



UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY



FACULTAD DE
INGENIERÍA

ProjectHub: Una herramienta para la gestión de proyectos basada en Valor Ganado

Informe de Proyecto de Grado presentado por

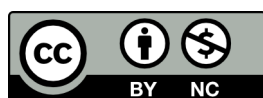
María Pía Delgado, Federico Martínez y Federico Tourn

en cumplimiento parcial de los requerimientos para la graduación de la carrera
de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de
la República

Supervisor

Diego Vallespir

Montevideo, 1 de julio de 2026



ProjectHub: Una herramienta para la gestión de proyectos basada en Valor Ganado por María Pía Delgado, Federico Martínez y Federico Tourn tiene licencia [CC Attribution-NonCommercial 4.0 International](https://creativecommons.org/licenses/by-nc/4.0/).

Agradecimientos

En primer lugar, queremos agradecer a nuestro tutor, Diego Vallespir, por su acompañamiento constante durante todo el desarrollo del proyecto, por su guía en la toma de decisiones y por su aporte académico y técnico, que fueron fundamentales para la concreción del mismo. Además, extendemos nuestro agradecimiento a la Facultad de Ingeniería por el apoyo institucional a lo largo de toda la carrera.

También queremos agradecer a Pyxis por abrirnos las puertas de la organización y brindarnos la posibilidad de trabajar con ellos. En particular, queremos destacar la participación de los profesionales de la empresa que colaboraron activamente en el proyecto aportando su experiencia en distintas áreas, así como también su valioso tiempo.

Finalmente, le agradecemos a nuestras familias y amigos por el apoyo incondicional durante todo el proceso de la carrera y, en particular, durante el desarrollo de este proyecto, brindándonos motivación y acompañamiento en cada etapa.

Resumen

El seguimiento y control de proyectos de software requiere contar con información confiable que permita evaluar el avance real de los proyectos, detectar desvíos y apoyar la toma de decisiones. En este contexto, la técnica de Valor Ganado constituye una alternativa para integrar planificación, seguimiento y análisis del desempeño de los proyectos. Sin embargo, las herramientas existentes no siempre se ajustan a las necesidades específicas de cada organización, tanto desde el punto de vista funcional como de experiencia de usuario.

Este proyecto surge a partir del interés de Pyxis, empresa uruguaya de desarrollo de software, en contar con una herramienta adaptada a su contexto organizacional. Como referencia conceptual se tomó la herramienta Process Dashboard, de la cual se identificaron oportunidades de mejora relacionadas con la experiencia de usuario, la adaptación a las necesidades de la organización y la incorporación de funcionalidades específicas para el seguimiento de proyectos.

El objetivo principal fue desarrollar una aplicación web multiusuario y multiproyecto que permita gestionar proyectos de software utilizando la técnica de Valor Ganado, brindando soporte a la planificación, el seguimiento del avance, la gestión de la capacidad de los equipos y la visualización de métricas en tiempo real. Adicionalmente, el proyecto se planteó como una oportunidad para recorrer integralmente las distintas etapas de la Ingeniería de Software en el contexto de un producto real desarrollado en colaboración con la industria, analizando tanto los resultados obtenidos como el proceso seguido.

El desarrollo se organizó en dos liberaciones incrementales siguiendo un proceso ágil. A lo largo del proyecto se realizaron actividades de relevamiento y validación de requisitos, diseño arquitectónico, implementación, despliegue en infraestructura de nube y testing. La participación activa de Pyxis permitió validar continuamente el producto mediante el intercambio con profesionales especializados en UX/UI, arquitectura de software e infraestructura. Además, una vez disponible una versión estable de la aplicación, esta comenzó a utilizarse por el equipo y por la organización.

Como resultado, se desarrolló una herramienta web que permite gestionar proyectos utilizando métricas de Valor Ganado, administrar equipos y calendarios de trabajo, registrar esfuerzo y visualizar indicadores de seguimiento y proyecciones de finalización. La solución fue desplegada en la infraestructura de Pyxis y validada junto a sus usuarios. Asimismo, el proyecto permitió analizar los beneficios y desafíos del trabajo conjunto entre academia e industria, desta-

cando el valor de la retroalimentación continua, la participación de especialistas y la validación del producto durante su construcción.

Palabras clave: Valor ganado, Gestión de proyectos, Planificación, Estimación, Seguimiento, Proyección

Índice general

1. Introducción	1
1.1. Objetivos del proyecto	2
1.2. Estructura del documento	3
2. Marco teórico	5
2.1. Generalidades	5
2.2. Técnica de Valor Ganado	6
2.3. Ejemplo de Valor Ganado	8
2.4. Personal Software Process y Team Software Process	17
2.5. La herramienta Process Dashboard	18
2.5.1. Limitaciones y necesidad de adaptación	19
2.5.2. Process Dashboard como referencia	20
3. Proceso de desarrollo de software	21
4. Requisitos	27
4.1. Primera liberación	27
4.1.1. Gestión de requisitos de la primera liberación	27
4.1.2. Requisitos de la primera liberación	30
4.2. Segunda liberación	31
4.2.1. Gestión de requisitos de la segunda liberación	31
4.2.2. Requisitos de la segunda liberación	32
5. Diseño arquitectónico	35
5.1. Primera liberación	38
5.1.1. Modelo de datos	38
5.1.2. Arquitectura de la lógica	39
5.2. Segunda liberación	40
5.2.1. Modelo de datos	41
5.2.2. Arquitectura de la lógica	42
6. Proceso de construcción	45
6.1. Tecnologías utilizadas	45
6.2. Control de versiones	46

6.3. Despliegues automáticos	46
6.4. Evolución de los diseños	47
6.5. Proceso	51
6.6. Implementación	53
6.6.1. Primera liberación	53
6.6.2. Segunda liberación	62
6.7. Comparación con Process Dashboard	72
7. Calidad y testing	75
7.1. Calidad del producto	75
7.2. Testing	76
7.2.1. Primera liberación	77
7.2.2. Segunda liberación	79
7.3. Consideraciones finales	84
8. Lecciones aprendidas	87
8.1. Gestión del alcance y definición de requisitos	87
8.2. Diseño incremental y decisiones arquitectónicas	88
8.3. Testing y validación continua	88
8.4. Colaboración con profesionales de Pyxis	89
8.4.1. Trabajo con UX/UI	89
8.4.2. Arquitectura e infraestructura	90
8.4.3. Encuestas y feedback del proceso	90
8.5. Evaluación del proceso de desarrollo utilizando la herramienta creada	91
8.5.1. Primera liberación	91
8.5.2. Segunda liberación	93
8.5.3. Comentarios finales	94
9. Conclusiones y Trabajo Futuro	97
9.1. Conclusiones	97
9.2. Trabajo a futuro	99
Referencias	101
A. Casos de uso de la primera liberación	103
A.1. Caso de uso de inicio de sesión	104
A.2. Caso de uso de cierre de sesión	105
A.3. Caso de uso de crear proyecto	106
A.4. Caso de uso de crear tarea	107
A.5. Caso de uso de eliminar tarea	108
A.6. Caso de uso de reordenar tarea	109
A.7. Caso de uso de editar horas semanales de un usuario	110

B. Cálculos para las métricas mostradas en la página de Dashboard	111
B.1. Métricas totales	111
B.1.1. Valor ganado (EV) acumulado a cierta fecha	111
B.1.2. Valor planificado (PV) acumulado a cierta fecha	111
B.1.3. Tiempo calendario planificado acumulado a cierta fecha	112
B.1.4. Tiempo calendario real acumulado a cierta fecha	112
B.1.5. Estimación total para tareas finalizadas	112
B.1.6. Esfuerzo real total en tareas finalizadas	113
B.2. Métricas de la última semana	113
B.2.1. Valor ganado de la última semana	113
B.2.2. Valor planificado de la última semana	113
B.2.3. Esfuerzo estimado en la última semana	114
B.2.4. Esfuerzo real en la última semana	114
B.2.5. Estimación para tareas finalizadas en la última semana	114
B.2.6. Esfuerzo real en tareas finalizadas en la última semana	115
B.3. Fechas previstas de fin	115
B.3.1. Fecha prevista de fin del proyecto según replanificación	115
B.3.2. Fecha prevista de fin del proyecto según pronóstico	117
B.4. Líneas mostradas en la gráfica	120
B.4.1. Línea de Valor Planificado	120
B.4.2. Línea de Valor Ganado	120
B.4.3. Línea de Valor Ganado replanificado	120
B.4.4. Línea de Valor Ganado pronosticado	120
C. Diseños de UI	121
C.1. Primera liberación	121
C.2. Segunda liberación	125
D. Manual de usuario	133
D.1. Requisitos previos	133
D.2. Acceso a la plataforma	133
D.3. Funcionalidades por roles	134
D.4. Crear nuevo proyecto	135
D.5. Página de proyecto	135
D.6. Página de Dashboard	137
D.6.1. Vista de las métricas del proyecto	138
D.6.2. Vista de las métricas individuales	139
D.7. Página de Tareas	140
D.7.1. Vista de las tareas del proyecto	141
D.7.2. Vista de las tareas individuales	141
D.7.3. Temporizador y horas realizadas	141
D.8. Página del Calendario	143
D.9. Página de Proyectos	144
D.10. Página de administrador	145

E. Pruebas de sistema	147
E.1. Caso de prueba: Inicio de sesión	147
E.2. Caso de prueba: Cierre de sesión	148
E.3. Caso de prueba: Multisesión	148
E.4. Caso de prueba: Vistas principales y Barra lateral	149
E.5. Caso de prueba: Pantalla de Administrador	150
E.6. Caso de prueba: Crear nuevo proyecto	151
E.7. Caso de prueba: Pantalla de Proyecto	152
E.8. Caso de prueba: Pantalla de Dashboard	152
E.9. Caso de prueba: Dashboard - Proyecto sin tareas	153
E.10. Caso de prueba: Dashboard - Proyecto finalizado	153
E.11. Caso de prueba: Dashboard - Métricas	154
E.11.1. Dashboard - Métricas Individuales	155
E.11.2. Dashboard - Métricas de Proyecto	156
E.12. Caso de prueba: Proyecto sin horas calendario	157
E.13. Caso de prueba: Proyecto - Editar fecha de comienzo	158
E.14. Caso de prueba: Pantalla de Calendario	159
E.15. Caso de prueba: Tareas - Registro de esfuerzo de forma manual	160
E.16. Caso de prueba: Tareas - Pausar Temporizador	161
E.17. Caso de prueba: Tareas - Temporizador persistente entre sesiones	161
F. Encuesta a profesionales de Pyxis	163
F.1. Encuesta a diseñadora UX/UI	163
F.2. Encuesta a arquitecto de software	165

Capítulo 1

Introducción

El software se ha convertido en un componente fundamental de la sociedad moderna y desempeña un papel central en el funcionamiento de numerosos sistemas y servicios utilizados cotidianamente. La evolución de las tecnologías informáticas y de comunicación ha incrementado su presencia e importancia, hasta consolidarlo como un elemento central de la vida diaria y del funcionamiento de los sistemas digitales contemporáneos (Sommerville, 2016; Jacobson, Lawson, Ng, McMahon, y Goedicke, 2019).

La creciente relevancia del software ha dado lugar al desarrollo de una industria dedicada a la creación, mantenimiento y comercialización de productos y servicios basados en software. En este contexto surge la Ingeniería de Software, disciplina encargada de aplicar principios, métodos y prácticas de ingeniería al desarrollo de sistemas de software. Según el *Cuerpo de Conocimientos de Ingeniería de Software (SWEBOK)*, la Ingeniería de Software se define como la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software (IEEE Computer Society, 2024).

El desarrollo de software suele organizarse mediante proyectos, por lo que las prácticas de Ingeniería de Software se complementan con técnicas de gestión que permitan planificar, controlar y evaluar su ejecución. Un proyecto es un esfuerzo temporal emprendido para crear un producto, servicio o resultado único, mientras que la gestión de proyectos consiste en la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades del proyecto para cumplir sus requisitos. Por su parte, el seguimiento de proyectos comprende la recopilación de datos sobre el desempeño del proyecto, la generación de medidas de desempeño y la comunicación de dicha información con el fin de evaluar su avance respecto a la planificación (Project Management Institute, 2021).

En este contexto, la necesidad de cumplir plazos, administrar recursos limitados y adaptarse a cambios durante la ejecución de los proyectos de software hace que la planificación inicial deba revisarse y ajustarse de manera continua. Contar con información precisa y actualizada sobre el estado de los proyectos resulta esencial para detectar desvíos, comprender sus causas y tomar decisiones oportunas.

Entre las técnicas orientadas a este propósito se encuentra la gestión del Valor Ganado (Earned Value Management o EVM), una metodología que combina medidas de alcance, cronograma y recursos para evaluar el desempeño y el avance de un proyecto. Es una técnica de dirección de proyectos que requiere el establecimiento de una línea base integrada respecto de la cual pueda medirse el desempeño a lo largo del proyecto. El EVM establece y monitorea tres métricas clave: Valor Planificado (PV), Valor Ganado (EV) y Costo Real (AC). La gestión del Valor Ganado compara la línea base para la medición del desempeño con el desempeño real del cronograma y de los costos. (Project Management Institute, 2021). Los principios del EVM pueden aplicarse a proyectos de cualquier tipo y sector, permitiendo evaluar avances reales, identificar desviaciones y generar proyecciones que apoyan la toma de decisiones.

Si bien existen herramientas que implementan este tipo de enfoques, sus características y funcionalidades no siempre se ajustan a las necesidades particulares de cada organización. Diferencias en los procesos de trabajo, en la forma de gestionar equipos y proyectos o en las expectativas respecto a la experiencia de usuario pueden hacer necesaria una adaptación al contexto específico de cada empresa.

1.1. Objetivos del proyecto

En este contexto surge el presente proyecto, impulsado por el interés de Pyxis, una empresa del sector del desarrollo de software de nuestro país, en contar con una herramienta adaptada a sus necesidades. El proyecto tiene como **primer objetivo** desarrollar un sistema de software adaptable a las necesidades de equipos de desarrollo de software, orientado al seguimiento y gestión de proyectos. La herramienta debe brindar soporte tanto a la planificación como al control del avance de los proyectos mediante la aplicación del método de Valor Ganado.

La aplicación debe permitir la gestión simultánea de múltiples proyectos, así como la participación de distintos equipos de trabajo. En este sentido, se busca construir una herramienta multiusuario y multiproyecto que aplique la técnica de Valor Ganado. La herramienta debe permitir el ingreso sencillo, desde un punto de vista de experiencia de usuario, de tareas a realizar. Debe permitir replanificar un proyecto mientras está en ejecución y el cálculo dinámico de métricas del mismo. Tiene que brindar una visualización en tiempo real del avance de los proyectos de forma de ser una ayuda para la detección temprana de desvíos y sus posibles causas. También debe facilitar la gestión de la capacidad de los equipos recalculando las métricas de planificación y seguimiento cuando la capacidad del equipo se modifica.

El **segundo objetivo** de este proyecto de grado es recorrer de manera integral las distintas etapas de la Ingeniería de Software aplicadas al desarrollo de un producto real, con el propósito de analizar y discutir los aciertos y oportunidades de mejora en este tipo de vínculo entre industria y academia. El foco está puesto tanto en la solución desarrollada como en el proceso seguido y en

las decisiones técnicas, metodológicas y organizacionales que surgen durante el desarrollo.

En este sentido, el proyecto se planteó explícitamente como la construcción de un producto, y no de un software experimental ni de un prototipo. El sistema a desarrollar corresponde a un producto empresarial de uso crítico, destinado al seguimiento de proyectos con clientes y a la toma de decisiones basada en datos y métricas. Esta definición condiciona todo el proceso, desde la forma de relevar requisitos hasta las decisiones arquitectónicas, los mecanismos de validación y las estrategias de testing.

1.2. Estructura del documento

El documento se estructura siguiendo las distintas etapas del ciclo de vida de un producto de software. En el Capítulo 2 se presenta el marco teórico que sirvió como base conceptual para el proyecto. En el Capítulo 3 se describe el proceso de trabajo adoptado por el equipo. El Capítulo 4 presenta el relevamiento, análisis, especificación y validación de los requisitos del sistema, detallando el proceso seguido para cada una de las liberaciones. En el Capítulo 5 se expone el diseño arquitectónico de la solución, incluyendo la arquitectura general del sistema, el diseño de la base de datos y las decisiones técnicas más relevantes. El Capítulo 6 describe la implementación realizada, las tecnologías utilizadas y los principales desafíos enfrentados durante el desarrollo de cada liberación. Posteriormente, el Capítulo 7 presenta las actividades de calidad y testing llevadas a cabo para validar el correcto funcionamiento del sistema y verificar el cumplimiento de los requisitos definidos. Las lecciones aprendidas a lo largo del proyecto de grado se presentan en el Capítulo 8. En el Capítulo 9 se presentan las conclusiones y el posible trabajo a futuro.

Capítulo 2

Marco teórico

En esta sección se presentan los principales conceptos teóricos que sustentan el desarrollo del presente proyecto. En particular, se introducen las nociones generales de Ingeniería de Software y gestión de proyectos, la técnica de Valor Ganado como herramienta central para el seguimiento del progreso, y los enfoques PSP y TSP orientados a la medición y mejora del proceso de desarrollo. Asimismo, se describe la herramienta Process Dashboard como referencia conceptual para la construcción de la solución propuesta.

2.1. Generalidades

El presente proyecto se enmarca dentro de la Ingeniería de Software, entendida como el conjunto estructurado de actividades, métodos y prácticas orientadas al desarrollo sistemático de sistemas de software de calidad.

Todo proceso de Ingeniería de Software comprende cuatro actividades fundamentales. La primera de estas es la especificación, donde se definen los servicios que el sistema debe brindar y las restricciones bajo las cuales debe operar. El diseño e implementación es la segunda actividad, que involucra la definición de la arquitectura del sistema y su construcción. Luego se encuentra la validación, cuyo propósito es verificar que el producto desarrollado cumple con los requisitos establecidos y satisface las necesidades del cliente. La última actividad es la evolución, que contempla los cambios necesarios para adaptar el sistema a nuevas necesidades o contextos tecnológicos. Además, de forma transversal a estas macro actividades, y para cada una de ellas, se encuentra la gestión de software. Dentro de la generalidad de la gestión se encuentran, entre otras, las disciplinas de gestión de la configuración, gestión de la calidad y la gestión de proyectos. En la gestión de proyectos, entre otros aspectos, se encuentran los de planificación, estimación de tareas y seguimiento de proyectos (Sommerville, 2016). Estos aspectos están fuertemente ligados a este proyecto de grado.

Se entiende como proyecto de software a un esfuerzo planificado y temporal cuyo objetivo es desarrollar un sistema que satisfaga un conjunto de requisitos

definidos, considerando restricciones como el tiempo, el costo y los recursos disponibles ([Project Management Institute, 2021](#)).

Para llevar adelante este tipo de proyectos, existen distintos enfoques y metodologías de desarrollo. Entre ellos, las metodologías ágiles constituyen un conjunto de enfoques de desarrollo de software caracterizados por ciclos iterativos cortos, participación frecuente del cliente y una fuerte orientación a la entrega continua de valor. Entre las metodologías ágiles más utilizadas se encuentra Scrum ([IEEE Computer Society, 2024](#)).

Scrum es un marco de trabajo ágil que permite desarrollar productos de forma incremental y colaborativa, mediante iteraciones cortas y de duración fija, denominadas *sprints*. Esta forma de trabajo promueve la inspección y adaptación continua a partir de la retroalimentación obtenida durante el proceso.

Scrum define ciertos roles, dentro de los cuales se encuentran *product owner*, *scrum master* y el equipo de desarrollo ([Scrum.org, s.f.](#)). El *product owner* es la persona responsable de maximizar el valor del producto y responsable final del producto desarrollado ([Project Management Institute, 2021](#)). Por otro lado el *scrum master* es responsable de promover y facilitar la correcta aplicación de Scrum dentro del equipo y la organización ([Scrum.org, s.f.](#)).

En este contexto, el punto de partida para la planificación es el *backlog* del producto, el cual constituye una lista priorizada de elementos que incluyen funcionalidades, requisitos y mejoras que deben ser abordadas por el equipo de desarrollo y que aportan valor de negocio al cliente ([Sommerville, 2016](#)).

En proyectos de la industria, independientemente de la metodología utilizada, los desvíos en calendario, esfuerzo o alcance son inevitables, por lo que resulta fundamental contar con mecanismos que permitan medir el desempeño del proyecto y tomar decisiones basadas en información objetiva ([The Standish Group, 2022](#)).

En este contexto, las métricas de seguimiento de proyecto constituyen herramientas clave para conocer el avance de los proyectos, detectar tempranamente posibles desvíos y facilitar la toma de decisiones. Una de las técnicas existentes para el control de proyectos es la gestión del Valor Ganado (Earned Value Management – EVM).

2.2. Técnica de Valor Ganado

EVM es una técnica que integra el alcance, el cronograma y los costos de un proyecto para evaluar de manera objetiva su desempeño y progreso. Esta técnica se basa en la definición de una línea base de desempeño contra la cual se mide el avance real del proyecto, permitiendo comparar el trabajo planificado con el efectivamente realizado y el costo incurrido. Para ello, el EVM utiliza tres métricas fundamentales: el Valor Planificado (PV), el Valor Ganado (EV) y el Costo Real (AC). A partir de estas variables, es posible analizar el estado del proyecto, identificar desvíos y proyectar tendencias, facilitando la toma de decisiones informadas ([Project Management Institute, 2021](#)).

Para el cálculo de las métricas asociadas a la gestión del Valor Ganado

(EVM), es necesario definir las unidades de trabajo del proyecto, comúnmente representadas como tareas o paquetes de trabajo. Estas permiten estructurar el alcance y establecer una base para la planificación y el seguimiento del progreso. A cada una de estas tareas se le asigna un tiempo planificado, que representa la cantidad de horas estimadas de esfuerzo para su realización, así como un registro del tiempo efectivamente dedicado a su ejecución.

Asimismo, resulta relevante considerar tanto el tiempo calendario planificado, definido como la cantidad de horas que se espera que cada integrante dedique al proyecto, como el tiempo calendario real, correspondiente a la cantidad de horas efectivamente dedicadas al mismo.

A partir de esta información, es posible determinar el Valor Ganado (EV) y el Valor Planificado (PV) del proyecto. El Valor Ganado se define como el porcentaje del trabajo efectivamente completado, calculado en función del tiempo planificado de las tareas finalizadas en relación con el tiempo total planificado del proyecto. Por su parte, el Valor Planificado representa el porcentaje del trabajo que debería haberse completado en un momento determinado, considerando el tiempo planificado de las tareas que, según el cronograma, tendrían que estar finalizadas.

En adición, con el objetivo de determinar la precisión en la estimación de las tareas y del tiempo calendario, se consideran tanto el tiempo planificado de las tareas finalizadas como el esfuerzo real invertido en ellas. A partir de estos valores, es posible calcular el porcentaje de desvío en la estimación. Este indicador mide qué tan precisa fue la estimación del esfuerzo para las tareas completadas y se obtiene como la relación porcentual entre el esfuerzo real de las tareas finalizadas y el tiempo planificado para dichas tareas. Un valor superior al 100 % indica que se dedicaron más horas de las estimadas (subestimación), mientras que un valor inferior al 100 % indica que se dedicaron menos horas (sobreestimación).

Además, a partir de los factores mencionados, es posible estimar de distintas formas la fecha de finalización del proyecto. A continuación, se presentan dos métodos de cálculo: según replanificación y según pronóstico.

La fecha prevista de finalización del proyecto según la replanificación se basa en la suposición de que, independientemente de los desvíos observados hasta el momento —ya sea en la estimación de las tareas o en el tiempo dedicado según el calendario—, las tareas restantes requerirán el tiempo originalmente estimado y, a partir de la semana actual, se dedicará el esfuerzo planificado en el calendario original.

Por otro lado, la fecha prevista de finalización del proyecto según pronóstico considera si las tareas finalizadas fueron subestimadas o sobreestimadas, así como si existieron desvíos en las horas efectivamente trabajadas respecto a lo planificado en el calendario. A partir de estos datos, se asume que dichos desvíos se mantendrán constantes en el futuro con respecto a lo planificado y se estima cómo evolucionará el proyecto.

2.3. Ejemplo de Valor Ganado

En esta sección se presenta un ejemplo (ProyectoEjemplo) que ilustra los conceptos teóricos mencionados. Para simplificar los cálculos, el ejemplo es de un proyecto que es llevado adelante por una única persona.

Entonces, el proyecto ProyectoEjemplo, tiene un único usuario. Esta persona definió su calendario de horas disponibles para el proyecto, desde la semana 1 (comienzo del proyecto) hasta la semana 5. Para cada semana, se cuenta con la estimación de horas que podrá dedicar la persona al proyecto. En la semana 1 estima dedicar 40 horas al proyecto, para la semana 2 30 horas, en la semana 3 nuevamente 40 horas, en la semana 4 estima dedicar 35 horas y, finalmente, en la semana 5 estima contar con 10 horas de esfuerzo disponibles. La figura 2.1 presenta el calendario de este proyecto. La misma muestra una semana adicional con una estimación de seis horas, que será utilizada más adelante en el ejemplo.

Semana	María Pía Delgado 40h std
20/4 - 26/4	40h
27/4 - 3/5	30h
4/5 - 10/5	40h
11/5 - 17/5	35h
18/5 - 24/5	10h
25/5 - 31/5	6h

Figura 2.1: Calendario de ProyectoEjemplo.

Para culminar el proyecto el usuario tiene que realizar cuatro tareas (Tarea 1, Tarea 2, Tarea 3 y Tarea 4). Estas tareas deben ser realizadas en exactamente ese orden presentado ya que tienen precedencia entre ellas. Cada una de estas tareas fue estimada al momento de planificar el proyecto según se puede ver en la figura 2.2 en la columna “Horas planeadas”.

El valor que se planifica para cada tarea corresponde a la cantidad de horas que se estiman que puede llevar la tarea para completarse. Para la Tarea 1 se le planificó un total de 25 horas, la Tarea 2 fue estimada en 30 horas, la Tarea 3 en 35 horas y por último la Tarea 4 fue estimada en 45 horas totales. El usuario del proyecto tiene un total de 135 horas planificadas, lo que implica que se estima que es necesario realizar ese esfuerzo para lograr finalizar el proyecto.

Con esta información, calendario, tareas y valor planificado para cada tarea, se puede desarrollar el plan previsto al comienzo del proyecto. En la primera

Tarea	Persona asignada	Horas planeadas	Horas realizadas	Estado	Fecha de fin	Acciones
☰ Tarea 1 Haz clic para agregar descripción	María Pía Delgado	25	0	Pendiente	-	✓ 🔊 ▶ 🗑
☰ Tarea 2 Haz clic para agregar descripción	María Pía Delgado	30	0	Pendiente	-	✓ 🔊 ▶ 🗑
☰ Tarea 3 Haz clic para agregar descripción	María Pía Delgado	35	0	Pendiente	-	✓ 🔊 ▶ 🗑
☰ Tarea 4 Haz clic para agregar descripción	María Pía Delgado	45	0	Pendiente	-	✓ 🔊 ▶ 🗑

Figura 2.2: Tareas de ProyectoEjemplo.

semana se debería finalizar la Tarea 1 y dedicar 15 horas a la Tarea 2. Por lo tanto, el Valor Planificado (PV) al final de la primera semana es de 18.5 %. Este valor, como ya se mencionó, indica el avance planificado en porcentaje (el que se espera obtener) al finalizar la primera semana. En la segunda semana se debería completar la Tarea 2 (realizando las 15 horas restantes) y avanzar 15 de las 35 horas de la Tarea 3. En consecuencia, el PV acumulado al final de la segunda semana es de 40.7 % (correspondiente a terminar las Tareas 1 y 2). En la tercera semana se deberían completar las 20 horas restantes de la Tarea 3 y dedicar 20 horas a la Tarea 4, alcanzando un PV de 66.6 %. Por último, en la cuarta semana se deberían completar las 25 horas restantes de la Tarea 4, alcanzando el 100 % de Valor Planificado. La quinta semana, que también tenía horas disponibles planificadas en el calendario, no es necesaria considerarla a los efectos del plan ya que todas las tareas estarían finalizadas en la cuarta semana. La figura 2.3 presenta una gráfica que muestra la evolución del valor planificado acumulado (línea azul) en función de cada semana del proyecto.

Lo presentado anteriormente constituye todos los elementos de un plan con valor ganado. A continuación, se presenta lo que fue sucediendo a lo largo del transcurso de las semanas para introducir los conceptos de seguimiento de proyecto con la técnica de VG. Para el ejemplo, se va a suponer que acaba de culminar la cuarta semana del proyecto.

En la semana 1 el usuario logró dedicar 38 horas al proyecto, pudiendo:

- Comenzar y finalizar la Tarea 1 en 30 horas
- Comenzar la Tarea 2, a la cual le dedicó 8 horas

La figura 2.4 presenta las tareas en las que el usuario realizó horas durante la semana 1. Como se observa, las horas planificadas para cada tarea se encuentran en la columna “Horas planeadas”, mientras que las horas efectivamente trabajadas se muestran en la columna “Horas realizadas”. Para aquellas tareas que registran esfuerzo, se incluyen los correspondientes registros de tiempo, los cuales indican el momento de inicio y de finalización del esfuerzo, así como su duración.

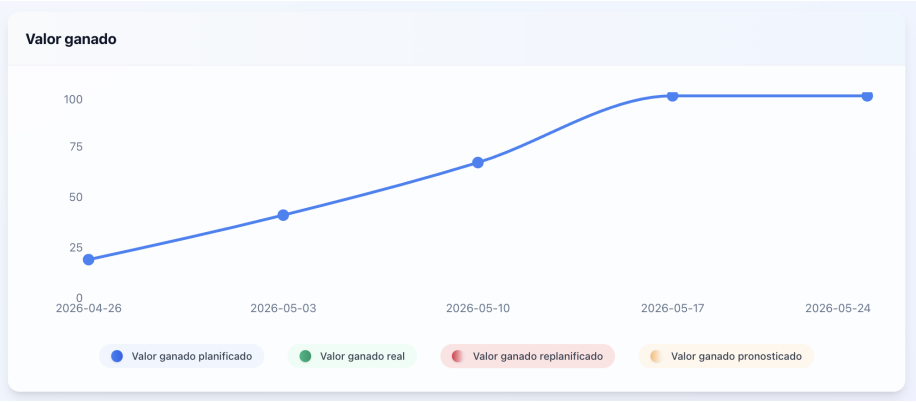


Figura 2.3: Gráfica con la evolución del valor planificado acumulado de ProyectoEjemplo.

Tarea	Persona asignada	Horas planeadas	Horas realizadas	Estado	Fecha de fin	Acciones
Tarea 1	Maria Pia Delgado	25	30	Completado	23/04/2026	
Registros de tiempo		Usuario	Fecha de inicio	Fecha de fin	Duración	
		Maria Pia Delgado	21/04/2026 10:00	21/04/2026 20:00	10:00:00	
		Maria Pia Delgado	22/04/2026 10:00	22/04/2026 20:00	10:00:00	
		Maria Pia Delgado	23/04/2026 10:00	23/04/2026 20:00	10:00:00	
Tarea 2	Maria Pia Delgado	30	8	En progreso	-	
Registros de tiempo		Usuario	Fecha de inicio	Fecha de fin	Duración	
		Maria Pia Delgado	24/04/2026 10:00	24/04/2026 18:00	08:00:00	
		+ Agregar registro de tiempo				
Tarea 3	Maria Pia Delgado	35	0	Pendiente	-	
Registros de tiempo		Usuario	Fecha de inicio	Fecha de fin	Duración	
		Maria Pia Delgado	24/04/2026 10:00	24/04/2026 18:00	08:00:00	
Tarea 4	Maria Pia Delgado	45	0	Pendiente	-	
Registros de tiempo		Usuario	Fecha de inicio	Fecha de fin	Duración	
		Maria Pia Delgado	24/04/2026 10:00	24/04/2026 18:00	08:00:00	

Figura 2.4: Tareas con horas realizadas en la semana 1.

Como se muestra en la misma figura, la Tarea 1 fue finalizada esta primera semana, pero a la Tarea 2 aún le quedan pendientes 22 horas a realizar según lo planificado.

La figura 2.5 presenta la gráfica de Valor Ganado luego de finalizar la primera semana. Se observa que el valor planificado coincide con el valor ganado debido a que el usuario logró completar la Tarea 1. La línea de valor ganado planificado coincide con la línea de valor ganado replanificado.

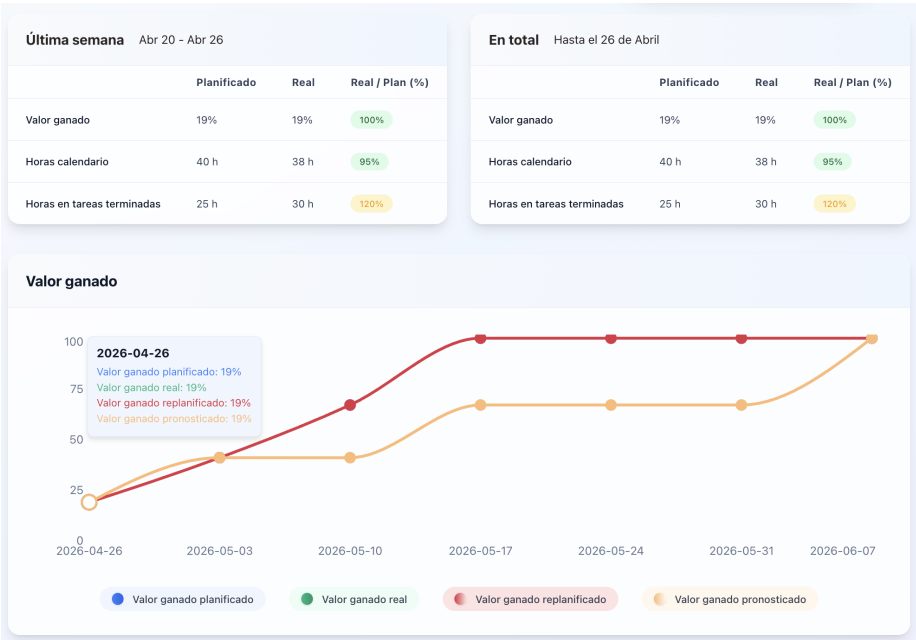


Figura 2.5: Gráfica de valor ganado luego de finalizar la semana 1.

En la semana 2 el usuario logró dedicar las 30 horas planificadas al proyecto, pudiendo:

- Finalizar la Tarea 2 realizando 12 horas adicionales (20 horas realizadas en total)
- Comenzar la Tarea 3, dedicándole 18 horas

La figura 2.6 presenta las tareas en las que el usuario logró realizar horas en la semana 2. La Tarea 2 fue finalizada esta segunda semana, pero a la Tarea 3 aún le quedan pendientes 17 horas a realizar según lo planificado.

La figura 2.7 presenta la gráfica de Valor Ganado luego de finalizar la segunda semana. Se observa que el valor planificado coincide con el valor ganado debido a que el usuario logró completar la Tarea 2 y acumular de la semana 1 la Tarea 1 completada.

Tarea	Persona asignada	Horas planeadas	Horas realizadas	Estado	Fecha de fin	Acciones
Tarea 1	María Pía Delgado	25	30	Completado	23/04/2026	
Tarea 2	María Pía Delgado	30	20	Completado	28/04/2026	
Registros de tiempo		Usuario	Fecha de inicio	Fecha de fin	Duración	
	María Pía Delgado	24/04/2026 10:00	24/04/2026 18:00	08:00:00		
	María Pía Delgado	27/04/2026 10:00	27/04/2026 16:00	06:00:00		
	María Pía Delgado	28/04/2026 10:00	28/04/2026 16:00	06:00:00		
Tarea 3	María Pía Delgado	35	18	En progreso	-	
Haz clic para agregar descripción						
Registros de tiempo		Usuario	Fecha de inicio	Fecha de fin	Duración	
	María Pía Delgado	29/04/2026 10:00	29/04/2026 20:00	10:00:00		
	María Pía Delgado	30/04/2026 10:00	30/04/2026 18:00	08:00:00		
+ Agregar registro de tiempo						
Tarea 4	María Pía Delgado	45	0	Pendiente	-	
Haz clic para agregar descripción						

Figura 2.6: Tareas con horas realizadas en la semana 2.

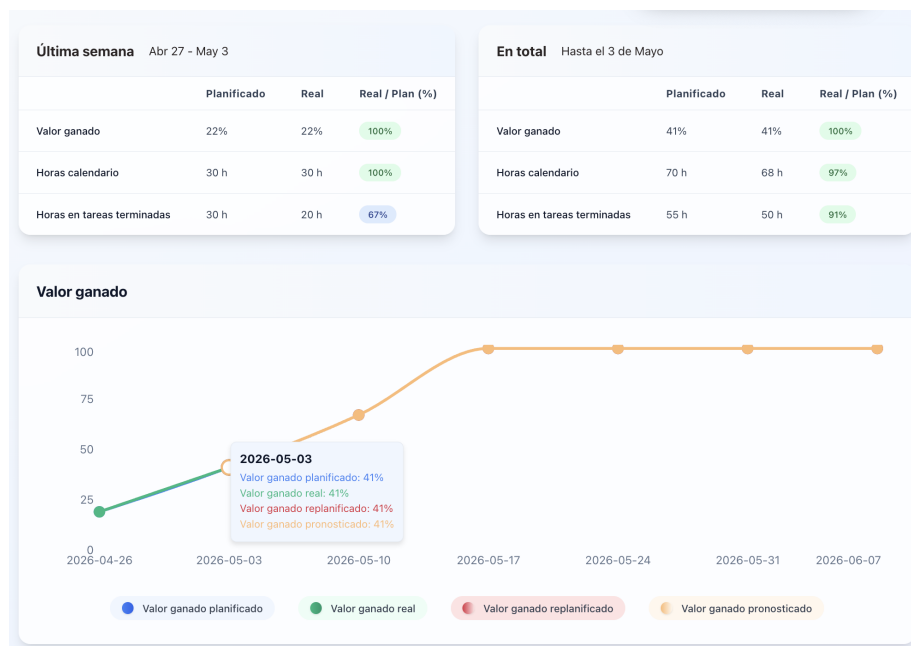


Figura 2.7: Gráfica de valor ganado luego de finalizar la semana 2.

En la semana 3 el usuario logró dedicar 40 horas al proyecto, pudiendo:

- Finalizar la Tarea 3 realizando 15 horas adicionales (33 horas realizadas en total)
- Comenzar la Tarea 4, dedicándole 25 horas

La figura 2.8 presenta las tareas en las que el usuario logró realizar horas en la semana 3. La Tarea 3 fue finalizada esta semana, pero a la Tarea 4 aún le quedan pendientes 20 horas a realizar según lo planificado.

Tarea	Persona asignada	Horas planeadas	Horas realizadas	Estado	Fecha de fin	Acciones
Tarea 1	Maria Pia Delgado	25	30	Completado	23/04/2026	
Tarea 2	Maria Pia Delgado	30	20	Completado	28/04/2026	
Tarea 3	Maria Pia Delgado	35	33	Completado	05/05/2026	
Registros de tiempo		Usuario	Fecha de inicio	Fecha de fin	Duración	
		Maria Pia Delgado	29/04/2026 10:00	29/04/2026 20:00	10:00:00	
		Maria Pia Delgado	30/04/2026 10:00	30/04/2026 18:00	08:00:00	
		Maria Pia Delgado	04/05/2026 10:00	04/05/2026 18:00	08:00:00	
		Maria Pia Delgado	05/05/2026 10:00	05/05/2026 17:00	07:00:00	
Tarea 4	Maria Pia Delgado	45	25	En progreso	-	
Registros de tiempo		Usuario	Fecha de inicio	Fecha de fin	Duración	
		Maria Pia Delgado	06/05/2026 10:00	06/05/2026 18:00	08:00:00	
		Maria Pia Delgado	07/05/2026 10:00	07/05/2026 17:00	07:00:00	
		Maria Pia Delgado	08/05/2026 10:00	08/05/2026 20:00	10:00:00	

Figura 2.8: Tareas con horas realizadas en la semana 3.

La figura 2.9 presenta la gráfica de Valor Ganado luego de finalizar la tercera semana. Se observa que el valor planificado coincide con el valor ganado debido a que el usuario logró completar la Tarea 3.

En la semana 4 el usuario logró dedicar 10 horas al proyecto, pudiendo:

- Continuar (pero no finalizar) la Tarea 4, realizando 10 horas adicionales

La figura 2.10 presenta la tarea en las que el usuario logró realizar horas en la semana 4, a esta tarea aún le quedan pendientes 10 horas a realizar según lo planificado.

La figura 2.11 presenta la gráfica de Valor Ganado luego de finalizar la cuarta semana. Se observa que el valor ganado no coincide con el valor planificado en esta semana, debido a que, según las horas disponibles del usuario en el calendario y las horas restantes planificadas, la Tarea 4 podría haberse completado durante esta semana, sin embargo, esto no ocurrió.

Al momento de empezar la quinta semana, las tareas 1, 2 y 3 ya se encuentran completadas. La única tarea pendiente de completarse es la Tarea 4, de la cual se

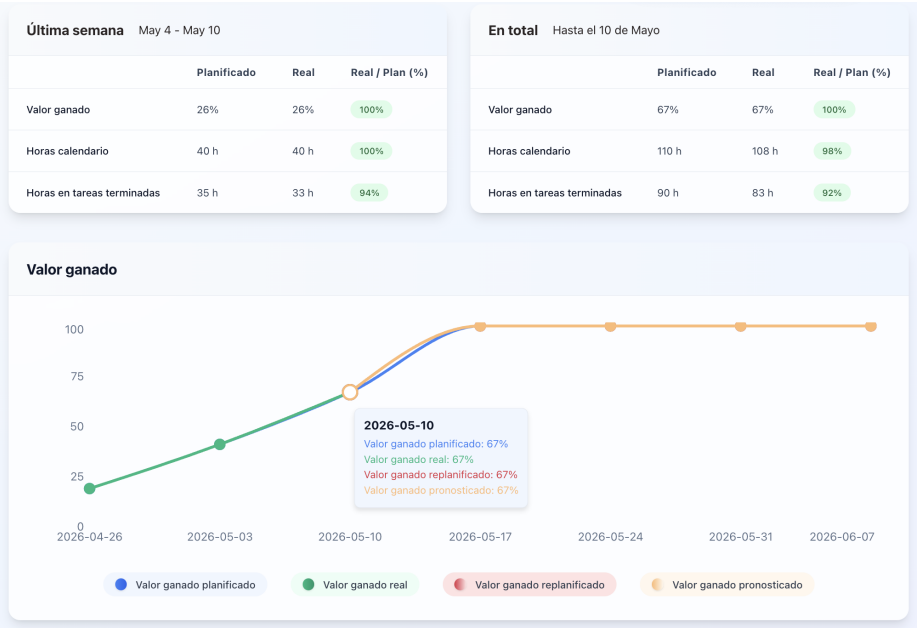


Figura 2.9: Gráfica de valor ganado luego de finalizar la semana 3.

Tarea	Persona asignada	Horas planeadas	Horas realizadas	Estado	Fecha de fin	Acciones
Tarea 1	Maria Pia Delgado	25	30	Completado	23/04/2026	
Tarea 2	Maria Pia Delgado	30	20	Completado	28/04/2026	
Tarea 3	Maria Pia Delgado	35	33	Completado	05/05/2026	
Tarea 4 Haz clic para agregar descripción	Maria Pia Delgado	45	35	En progreso	-	
Registros de tiempo		Usuario	Fecha de inicio	Fecha de fin	Duración	
		Maria Pia Delgado	06/05/2026 10:00	06/05/2026 18:00	08:00:00	
		Maria Pia Delgado	07/05/2026 10:00	07/05/2026 17:00	07:00:00	
		Maria Pia Delgado	08/05/2026 10:00	08/05/2026 20:00	10:00:00	
		Maria Pia Delgado	12/05/2026 10:00	12/05/2026 20:00	10:00:00	
						+ Agregar registro de tiempo

Figura 2.10: Tareas con horas realizadas en la semana 4.

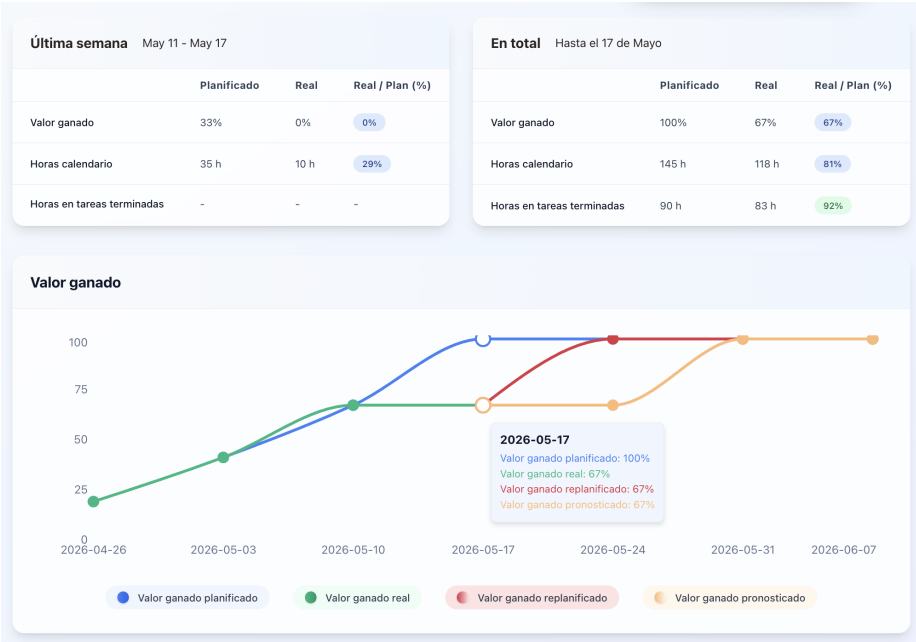


Figura 2.11: Gráfica de valor ganado luego de finalizar la semana 4.

observa en la figura 2.10 que le quedan por realizar 10 horas según lo planeado. A partir de esta información, se puede visualizar en el cuadro de las métricas totales que se encuentra en la figura 2.11 los siguientes datos:

- Valor Ganado: 66.6 % (las tareas 1, 2 y 3 tienen un total de 90 horas estimadas, lo cual representa un 66.6 % del total de 135 horas estimadas en tareas)
- Valor Planificado: 100 % (se planificaba finalizar todas las tareas)
- Esfuerzo Planificado: 145 horas (en base al calendario)
- Esfuerzo Real: 118 horas (en base a lo realmente dedicado a las tareas)
- Estimación para tareas finalizadas: 90 horas
- Esfuerzo real en tareas finalizadas: 83 horas

Al comenzar la cuarta semana del proyecto, las tareas 1, 2 y 3 ya se encontraban completadas. En ese momento la Tarea 4 tenía pendiente 20 horas por realizar y el usuario tenía disponibles en el calendario 35 horas. A partir de esta información, se puede visualizar en el cuadro de las métricas de la última semana que se encuentra en la figura 2.11 los siguientes datos:

- Valor Ganado: 0 % (en la semana 4 no se finalizó ninguna tarea)
- Valor Planificado: 33.3 % (al comenzar la semana 4, se planificaba finalizar la Tarea 4, que tiene 45 horas estimadas, lo cual representa un 33.3 % del total)
- Esfuerzo Planificado: 35 horas
- Esfuerzo Real: 10 horas
- Estimación para tareas finalizadas: no se finalizaron tareas durante la semana 4
- Esfuerzo real en tareas finalizadas: no se finalizaron tareas durante la semana 4

A continuación se explica cómo se calcula, al momento de comenzar la quinta semana, la fecha prevista de finalización del proyecto.

En primer lugar, para el cálculo de la fecha prevista de fin según replanificación, se considera el trabajo restante en función de la estimación original de la o las tareas en curso o aún pendientes. En este caso, la Tarea 4 tiene una estimación de 45 horas y, al inicio de la semana 5, se han dedicado 35 horas, por lo que restan 10 horas de trabajo.

Dado que en la semana 5 el usuario tiene planificadas 10 horas de dedicación al proyecto, se asume que la Tarea 4 podrá finalizarse exactamente al cierre de dicha semana. En consecuencia, la fecha prevista de fin según replanificación corresponde al fin de la semana 5.

Para el cálculo de la fecha prevista de fin según pronóstico, se consideran tanto la dedicación real observada como la precisión en las estimaciones de las tareas.

En cuanto a la dedicación semanal, hasta el final de la semana 4 se planificaba un total de 145 horas, mientras que el esfuerzo real fue de 118 horas. Esto permite obtener un ratio de dedicación de $118 / 145 = 0,81$. A partir de este valor, se estima que en la semana 5 se dedicarán efectivamente $10 \times 0,81 = 8,14$ horas al proyecto.

Por otra parte, para calcular el índice de desempeño de costos (CPI), se considera la relación entre el esfuerzo planificado y el esfuerzo real en las tareas finalizadas. Las tareas completadas hasta el momento son las tareas 1, 2 y 3, cuya estimación total es de 90 horas, mientras que el esfuerzo real dedicado a las mismas fue de 83 horas. De este modo, el CPI resulta $90 / 83 = 1,08$. Es decir, las tres tareas consideradas en conjunto fueron sobrestimadas.

A partir de este valor, se estima que el trabajo restante de la Tarea 4 (10 horas) requerirá $10/1,08 \approx 9,26$ horas reales.

Durante la semana 5, la capacidad estimada es de 8,14 horas, por lo que no será posible completar la tarea en dicha semana. En consecuencia, se utilizará la totalidad de la capacidad disponible y quedará un remanente de aproximadamente $9,26 - 8,14 = 1,12$ horas de trabajo.

Por lo tanto, la finalización de la Tarea 4 se trasladará a la semana 6, evidenciando un desvío respecto a la planificación original.

Supongamos ahora que la dedicación semanal planificada para la semana 6 es de 6 horas. Considerando el ratio de dedicación previamente calculado (0,81), se estima que la dedicación real durante dicha semana será de $6 \times 0,81 = 4,86$ horas.

En términos relativos, las 1,12 horas restantes representan $1,12/4,86 \approx 0,23$, es decir, un 23 % de la capacidad efectiva de dicha semana.

Si se considera la semana completa como un período de 7 días (168 horas), esta fracción corresponde a aproximadamente 38,6 horas desde el inicio de la semana.

Por lo tanto, la Tarea 4 se finalizará aproximadamente durante el transcurso del martes de la semana 6. En consecuencia, la fecha prevista de fin según pronóstico se sitúa en ese momento.

2.4. Personal Software Process y Team Software Process

Como parte del marco conceptual del proyecto, también se presentan el Personal Software Process (PSP) y el Team Software Process (TSP), creados por Watts Humphrey en el Software Engineering Institute.

El PSP es una metodología orientada al trabajo individual del ingeniero de software. Propone que cada desarrollador siga un proceso definido y registre métricas detalladas sobre el tiempo dedicado a cada tarea, los defectos introdu-

cidos y removidos en las distintas fases del desarrollo, y el tamaño del producto construido. A partir del análisis de estos datos históricos es posible realizar estimaciones más precisas, hacer seguimiento del avance del proyecto y evaluar objetivamente la calidad del producto. Asimismo, el PSP proporciona un marco estructurado que permite a los ingenieros planificar su trabajo, medir su desempeño y mejorar progresivamente sus habilidades mediante el análisis de datos propios. (Humphrey, 2005)

Por su parte, el TSP extiende estos mismos principios al trabajo en equipo y está pensado para grupos de ingenieros que aplican PSP. Está orientado a guiar equipos de ingeniería en la construcción de software de manera disciplinada, estableciendo procedimientos concretos que permiten estructurar el trabajo, mejorar la eficiencia y reducir la variabilidad en los resultados. Su aplicación busca lograr una mayor previsibilidad en cronogramas y presupuestos, mejorar la productividad del equipo y disminuir significativamente los defectos detectados en etapas tardías del desarrollo. (Humphrey, 2006)

El TSP organiza el trabajo en ciclos iterativos, donde cada ciclo comienza con un launch y puede incluir relanzamientos (relaunch) en función de la evolución del proyecto. Durante el launch, el equipo define metas, roles, estrategias, procesos y planes de trabajo, generando tanto un plan general como planes detallados para las etapas siguientes. Este proceso es colaborativo, de modo que todos los integrantes participan en la planificación y se comprometen con los resultados definidos. A lo largo del ciclo, los miembros del equipo registran de manera sistemática el tiempo dedicado a sus tareas y otros datos del proceso, lo que permite medir el desempeño, comparar lo planificado con lo efectivamente realizado e identificar desvíos. Esta información es utilizada para realizar ajustes, mejorar la planificación y estimar de manera más precisa el trabajo futuro. (Humphrey, 2006)

En conjunto, el TSP no solo define una estructura de trabajo para los equipos, sino que también establece un enfoque basado en la medición y el seguimiento continuo del desempeño. A través del registro sistemático de datos del proceso, como el tiempo dedicado a las tareas y el progreso alcanzado, los equipos pueden evaluar objetivamente su avance, anticipar posibles desvíos y tomar decisiones informadas. Este enfoque permite no solo mejorar la planificación y ejecución del proyecto en curso, sino también fortalecer la capacidad de estimación y gestión en futuros ciclos de desarrollo. (Humphrey, 2006)

2.5. La herramienta Process Dashboard

Process Dashboard es una herramienta que da soporte al proceso TSP, al PSP y también a los cursos para aprender el PSP. La primera versión de esta herramienta es de 1998 y aún sigue en evolución continua. Esta herramienta sirvió como referencia para la construcción de la aplicación desarrollada en el marco de este proyecto de grado.

Como se describió en este capítulo, tanto el PSP como el TSP requieren la recolección y el análisis de métricas a un nivel de gran detalle. En el caso

del TSP, además, es necesario consolidar las métricas individuales para generar métricas a nivel de equipo. Cuando los datos se recopilan con este grado de precisión, es posible aplicar análisis que permiten mejorar significativamente la planificación, el seguimiento, la predicción y el control de productos y proyectos de software.

No obstante, estos procesos de recolección y análisis de métricas no son triviales. En proyectos reales, el soporte mediante herramientas se vuelve un factor relevante para poder sostener la aplicación de estos enfoques. Si bien, en la teoría, es posible utilizar PSP sin herramientas específicas, el esfuerzo adicional requerido para mantener manualmente los registros hace que esto sea imposible en la práctica industrial. En este contexto surge Process Dashboard, como una herramienta diseñada para facilitar la recolección, consolidación y análisis de métricas necesarias para la implementación de PSP y TSP.

Process Dashboard puede utilizarse tanto en proyectos individuales como en proyectos grupales. Entre sus principales funcionalidades se encuentran:

- Recolección de datos: registro de tiempo, defectos y tamaño del producto, diferenciando datos planificados y reales.
- Planificación: utilización de *scripts*, plantillas, formularios y resúmenes integrados, incluyendo soporte para métricas de valor ganado.
- Seguimiento: herramientas para monitorizar el avance del proyecto mediante indicadores cuantitativos.
- Análisis de datos: generación de gráficos e informes que permiten analizar tendencias históricas y evaluar el desempeño.
- Exportación de datos: posibilidad de exportar información a planillas electrónicas o a formatos de texto para su utilización con herramientas externas.

De esta manera, la herramienta cumple un rol fundamental como soporte operativo para la aplicación de PSP y TSP, reduciendo la complejidad asociada al manejo manual de métricas y permitiendo una gestión basada en datos objetivos.

2.5.1. Limitaciones y necesidad de adaptación

Process Dashboard constituye un antecedente relevante y una solución consolidada para la implementación de estos enfoques. Sin embargo, el desarrollo del presente proyecto de grado surgió de la necesidad de contar con una herramienta adaptada al contexto específico de Pyxis.

En primer lugar, la interfaz visual de Process Dashboard se encuentra desactualizada en comparación con otras aplicaciones web, lo que motivó la búsqueda de una solución con una experiencia de usuario más moderna validada por un experto UX/UI. Solo a modo de ejemplo, la figura 2.12 presenta una imagen de una de las pantallas de esta herramienta. Según la mirada de los expertos de

Pyxis, este tipo de experiencias de usuario y de interfaz gráfica son de alrededor de la década de 1990 y está muy desactualizada al tipo de experiencia que hoy se espera.

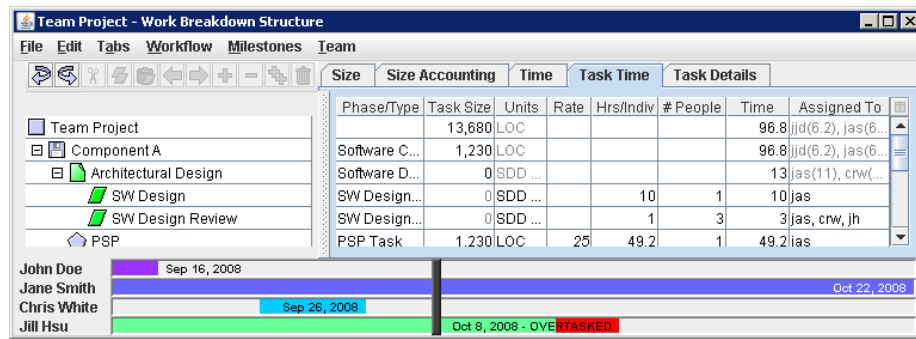


Figura 2.12: Herramienta de Process Dashboard.
Nota: Imagen tomada de Process Dashboard (Tuma Solutions, s.f.).

En segundo lugar, a partir del análisis realizado junto a Pyxis, surgió la necesidad de una herramienta que contemple funcionalidades específicas para el seguimiento de proyectos, incorporando únicamente aquellas características relevantes para el contexto particular de la organización.

Finalmente, si bien Process Dashboard cuenta con una versión gratuita orientada al uso individual, la versión destinada a organizaciones es de pago. Esta última incorpora funcionalidades adicionales, como la gestión centralizada de equipos, la administración de usuarios, el almacenamiento en servidor y el soporte para entornos corporativos, las cuales no están disponibles en la versión gratuita.

2.5.2. Process Dashboard como referencia

Se propone utilizar la documentación oficial de Process Dashboard como guía conceptual y funcional para el diseño de la solución a desarrollar.

En este sentido, Process Dashboard se considera un antecedente y una referencia técnica a partir de la cual se busca orientar el desarrollo de una solución personalizada, alineada a los objetivos, requerimientos y contexto organizacional definidos junto al cliente.

Capítulo 3

Proceso de desarrollo de software

El proyecto se desarrolló siguiendo el ciclo completo de la Ingeniería de Software, abarcando desde la obtención y análisis de requisitos hasta la validación y retroalimentación con usuarios provenientes de la industria.

El desarrollo se organizó en dos liberaciones principales, que se detallarán más adelante. Cada liberación tuvo como objetivo generar entregables con valor significativo para el cliente (Pyxis). Para ello, se adoptó un proceso ágil con desarrollo incremental, realizando entregas frecuentes y ajustes continuos en función del *feedback* recibido.

La gestión del desarrollo no se realizó siguiendo íntegramente el marco de trabajo Scrum. Sin embargo, se adoptaron varios de los conceptos presentados en la sección 2.1 del Marco Teórico, entre ellos los roles de *product owner* y equipo de desarrollo, así como el uso de un *backlog* de producto. Por otra parte, se decidió no incorporar el rol de *scrum master*, ya que el tamaño reducido del equipo y la naturaleza del proyecto no justificaban asignar a un integrante responsabilidades específicas de facilitación y coordinación. En su lugar, dichas tareas fueron asumidas de manera compartida por los miembros del equipo.

En cuanto a la distribución de responsabilidades, el equipo adoptó un enfoque colaborativo en el que todos los integrantes participaron en las distintas actividades del proyecto. Esto incluyó actividades de desarrollo tanto de *frontend* como de *backend*, testing y revisiones de código. Asimismo, los miembros del equipo trabajaron de forma conjunta en las tareas de planificación, gestión y elaboración de la documentación del proyecto. La principal excepción estuvo vinculada al despliegue y la infraestructura de la nube, donde un integrante asumió un rol más activo debido a la necesidad de adquirir conocimientos específicos sobre AWS, Kubernetes y los mecanismos de despliegue, que se mencionarán posteriormente en los capítulos 5 y 6.

Pyxis mostró un fuerte interés en participar activamente como cliente del proyecto de grado y brindó horas de acompañamiento a lo largo del proce-

so. Dichas horas fueron destinadas principalmente a actividades de gestión del proyecto, diseño UX/UI, arquitectura e infraestructura. En cuanto a los roles definidos dentro del marco ágil adoptado, Diego Vallespir (tutor de este proyecto de grado) asumió el rol de *product owner*. Además, colaboraron profesionales de Pyxis, entre ellos una diseñadora UX/UI, arquitectos de software y especialistas en infraestructura. La interacción constante con estos actores permitió contrastar decisiones que se fueron tomando a lo largo del proyecto con prácticas profesionales de primer nivel, enriqueciendo el proyecto y fortaleciendo la calidad del producto final.

Con el objetivo de complementar la descripción anterior, en la figura 3.1 se presenta los roles involucrados en el proyecto y su descripción.

El proceso de desarrollo se organizó en ciclos de trabajo semanales. Al inicio de cada semana, el equipo realizaba una reunión interna en la que se definían las tareas a abordar, las cuales habían sido previamente identificadas y priorizadas en función del valor para el cliente. Durante las primeras etapas del proyecto, estas tareas se gestionaban utilizando la aplicación Jira y una planilla de Excel para el registro de horas.

Durante la semana, cada integrante del equipo desarrollaba sus tareas y registraba el tiempo dedicado a las mismas. Con el avance del proyecto, la gestión de tareas y el registro de tiempo se trasladó a la propia aplicación desarrollada (ProjectHub), aplicando el dicho “comer de nuestra propia torta”. Esto permitió administrar las tareas, visualizar el avance del trabajo y evaluar el progreso de cada liberación utilizando la herramienta en un contexto real de uso.

Además, se trabajó con una dinámica de reuniones mayormente quincenales con el *product owner*. En estas reuniones se presentaban avances, se validaban los requisitos y se recibía retroalimentación, la cual era utilizada para ajustar la planificación de las siguientes semanas. Adicionalmente, se realizaron encuentros con otros actores del proyecto cuando fue necesario.

Por último, durante ambas liberaciones el equipo de desarrollo realizó despliegues mayormente semanales a producción. Esto permitió que las funcionalidades completadas quedaran disponibles de forma progresiva. Además, luego de finalizada la primera liberación y contando con una versión estable del sistema, el cliente comenzó a utilizar la plataforma de manera continua, lo que permitió recibir retroalimentación temprana y validar los cambios incorporados a través de los sucesivos despliegues.

En la figura 3.2 se presenta el ciclo adoptado durante el desarrollo. En la misma se ilustran las principales etapas del proceso, el flujo iterativo de trabajo y la interacción continua con los expertos en UX/UI, arquitectura de software y tecnologías de la información del cliente.

Por otra parte, al inicio del proyecto, se realizó un análisis de los requisitos del sistema, relevando las necesidades del cliente y definiendo las funcionalidades necesarias. Este relevamiento se llevó a cabo principalmente mediante reuniones con el *product owner*, quien aportó la visión del negocio y las prioridades del cliente.

Adicionalmente, antes de comenzar el trabajo de cada una de las dos liberaciones se realizaron instancias de planificación en las que se definieron las tareas

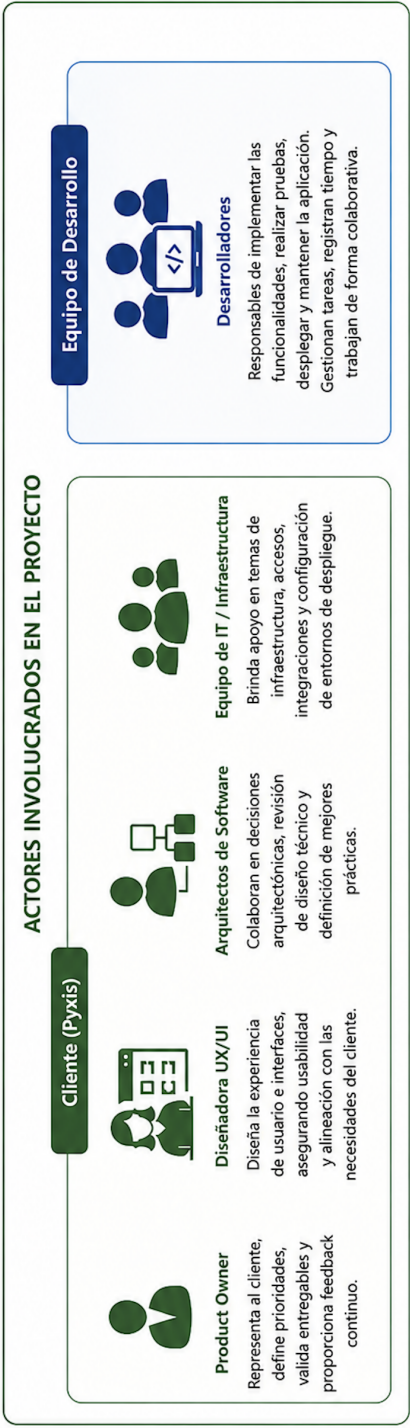


Figura 3.1: Actores involucrados en el proyecto.

Nota. Imagen generada mediante ChatGPT ([OpenAI, 2026](#)) a partir de un prompt elaborado por los autores.

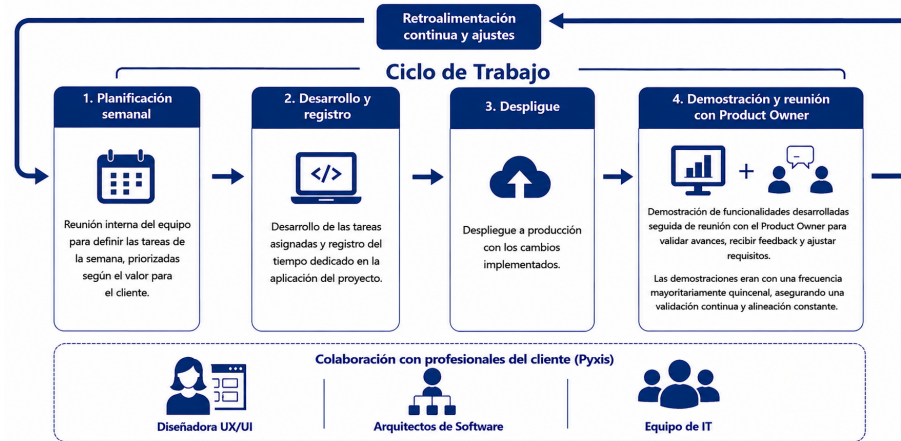


Figura 3.2: Ciclo de desarrollo adoptado en el proyecto.

Nota. Imagen generada mediante ChatGPT (OpenAI, 2026) a partir de un prompt elaborado por los autores.

a desarrollar, se estimaron en horas y se asignaron a los integrantes del equipo. Estas instancias fueron similares al concepto de despegue (*launch*) del Team Software Process (TSP).

El equipo tomó la decisión de registrar en la herramienta únicamente las tareas relacionadas con la implementación de funcionalidades, dejando fuera otras actividades como las revisiones de código, la elaboración de documentación y la planificación de las liberaciones. En consecuencia, al planificar las horas semanales de cada integrante, se consideró únicamente el tiempo destinado a tareas de implementación, asumiendo que las horas restantes que se debían dedicar semanalmente al proyecto de grado, se invertirían en las demás actividades necesarias para el desarrollo del proyecto.

El equipo consideró que actividades como las revisiones de código, las reuniones de planificación y la elaboración de documentación eran necesarias para alcanzar los objetivos del proyecto y contribuir a la calidad del producto desarrollado. Sin embargo, al tratarse de actividades cuyo esfuerzo resultaba más difícil de estimar de forma consistente, se decidió dejarlas fuera de las estimaciones y del seguimiento realizado mediante la plataforma, enfocando la misma en las tareas de implementación. Esta decisión y sus impactos se analizan críticamente en el capítulo 8.

Además de las instancias de planificación inicial, se incorporaron otras prácticas inspiradas en el TSP. En particular, al utilizar la propia aplicación desarrollada el equipo pudo registrar y realizar el seguimiento de métricas individuales y de proyecto. Esto permitió monitorear el avance respecto al plan y detectar posibles desvíos a partir de datos reales.

En cuanto a las liberaciones, el objetivo de la primera fue construir una

aplicación que permitiera a un usuario gestionar, mediante la técnica de Valor Ganado, un proyecto cuyo único integrante registrado en el sistema fuera el usuario. Dentro de la gestión del proyecto se incluyeron funcionalidades como la creación y edición de tareas, la gestión del calendario y la visualización de métricas relevantes, las cuales se explicarán más adelante. Asimismo, el usuario debía poder acceder a la plataforma utilizando su cuenta de Microsoft 365. Por último, esta liberación incluyó la configuración del entorno de despliegue y el despliegue de la aplicación en dicho ambiente.

El objetivo de la segunda liberación fue adaptar la plataforma para permitir la gestión de múltiples usuarios para un mismo proyecto y múltiples proyectos para un mismo usuario. En una primera instancia, esta liberación estaba planificada en dos iteraciones: la primera orientada a soportar múltiples usuarios asociados a un único proyecto, y la segunda enfocada en la incorporación de la gestión de múltiples proyectos. Sin embargo, se decidió unificar ambas iteraciones, ya que los cambios necesarios para soportar más de un proyecto resultaron ser relativamente menores, dado que desde la primera liberación el código había sido diseñado de forma suficientemente flexible como para facilitar la incorporación de múltiples proyectos.

En la figura 3.3 se presenta una síntesis cronológica de los principales hitos y actividades que marcaron el desarrollo del proyecto.

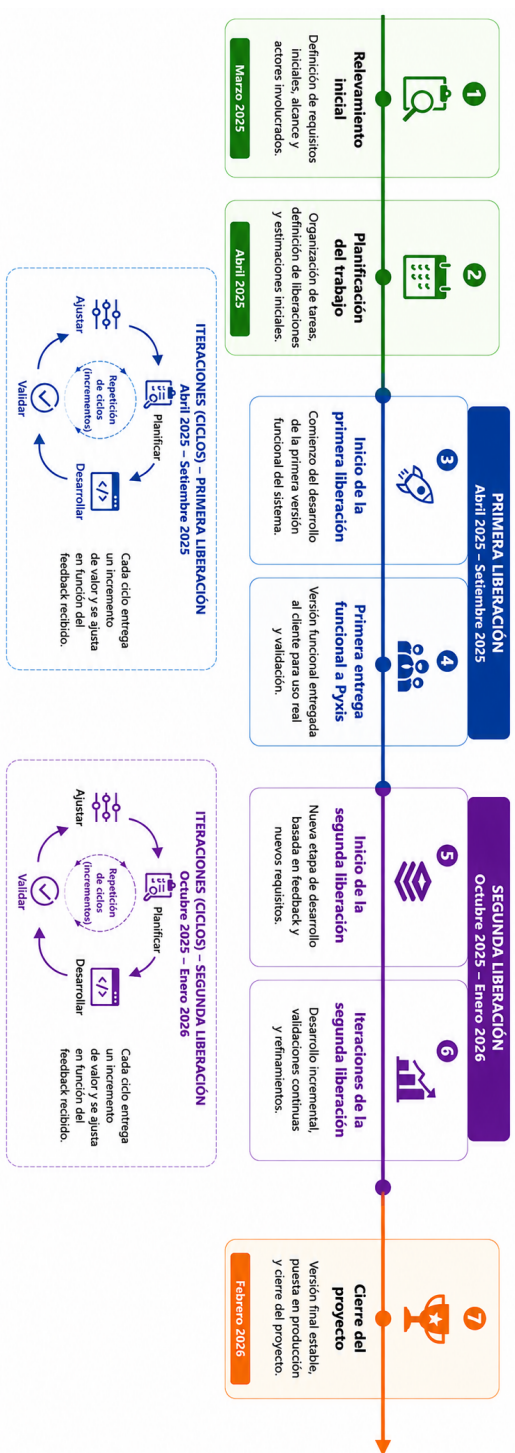


Figura 3.3: Evolución del proceso de trabajo a lo largo del proyecto.
 Nota. Imagen generada mediante ChatGPT ([OpenAI, 2026](#)) a partir de un prompt elaborado por los autores.

Capítulo 4

Requisitos

En este capítulo se describe el proceso de obtención, análisis, especificación y validación de los requisitos del sistema así como la especificación de requisitos en lenguaje natural. En este proyecto, la gestión de requisitos se llevó adelante de manera incremental, alineado con el enfoque ágil adoptado.

La obtención de requisitos se realizó de forma progresiva a lo largo de todo el proyecto. En una primera instancia, los requisitos fueron identificados a partir de reuniones con el tutor, quien tuvo el rol de *product owner*, y mediante discusiones internas del equipo, donde se definieron los objetivos generales del sistema y las funcionalidades principales que debía proveer.

Posteriormente, se incorporaron instancias de validación con stakeholders del lado de Pyxis, como los arquitectos de software, quienes realizaron observaciones y sugerencias que permitieron refinar y mejorar los requisitos especificados. También se utilizó la herramienta Process Dashboard como marco de referencias y para obtener posibles nuevas funcionalidades para validar con el *product owner*.

Presentamos a continuación la gestión de los requisitos y su especificación dividido por las dos liberaciones ya mencionadas.

4.1. Primera liberación

La subsección 4.1.1 presenta la gestión de requisitos mientras que la subsección 4.1.2 presenta los requisitos de la primera liberación.

4.1.1. Gestión de requisitos de la primera liberación

Para la primera liberación, el ciclo de desarrollo de los requisitos se realizó en gran parte mediante reuniones con el *product owner*. En una primera instancia el *product owner* presentó la aplicación mencionada en el marco teórico, Process Dashboard, de la cual se presentaron sus funcionalidades principales y, a partir de estas, las necesidades para la nueva aplicación. Asimismo, se mencionaron

aspectos de usabilidad que brinda Excel y que se debían incluir en la herramienta para que los usuarios pudieran llenar campos de forma rápida y solamente manejando el teclado.

Además de las reuniones, se estudiaron las funcionalidades principales de la aplicación Process Dashboard mediante videos. A partir de este proceso se recolectaron la mayoría de los requisitos funcionales de la primera liberación, así como varios que luego formaron parte de la segunda liberación.

Por otro lado, además de los requisitos obtenidos mediante el análisis de la aplicación, el equipo contó con algunos requerimientos iniciales ya establecidos por el *product owner* y por la empresa Pyxis. Entre ellos, se definieron los frameworks de programación a utilizar (Next.js para el *frontend* y Spring Boot para el *backend*), así como la implementación del login con Microsoft 365, particularmente para usuarios pertenecientes a la organización de Pyxis. Además, se especificó que la infraestructura de despliegue debía utilizar Amazon Web Services (AWS).

La mayoría de los requisitos para esta liberación, una vez obtenidos, fueron especificados utilizando casos de uso e historias de usuario para su validación.

En una primera instancia, se utilizó historias de usuario como mecanismo principal de especificación, dado que el equipo estaba más acostumbrado a la utilización de esta técnica en otros proyectos. Sin embargo, tras diversas instancias de revisión con el tutor, se determinó que la utilización de casos de uso permitía una mejor identificación de actores, escenarios e interacciones del sistema. Debido a esto, se decidió emplear esta técnica como herramienta principal para la especificación y validación de los requisitos en la primera liberación.

Cada caso de uso incluyó: nombre, objetivo, actores involucrados, descripción general, precondiciones, flujo principal, flujos alternativos y postcondiciones.

La tabla 4.1 presenta a modo de ejemplo el caso de uso para el inicio de sesión con Microsoft 365. Los casos de uso restantes se encuentran en el anexo A.

Esta formalización permitió no solo delimitar el alcance de la liberación sino también detectar inconsistencias y ambigüedades entre la visión de la aplicación del equipo y la del *product owner*. También sirvió para identificar escenarios alternativos.

Los documentos de requisitos y diagramas de arquitectura obtenidos fueron compartidos con los arquitectos de software de Pyxis, quienes realizaron una revisión más técnica. Esta instancia funcionó como mecanismo de validación de algunos requisitos, permitiendo verificar consistencia y ayudar a mejorar la calidad de los mismos. Entre los diagramas de arquitectura, los de componentes y de despliegue, que se presentarán en el capítulo 5 de Diseño Arquitectónico del sistema, fueron los más relevantes para la validación junto a los arquitectos. En particular, estos diagramas permitieron validar los componentes involucrados en el despliegue del sistema en AWS, así como la comunicación entre el *frontend* y el *backend*.

Sumado a los casos de uso, se utilizaron para la validación de ciertos requisitos prototipados de las pantallas creados por el equipo, los cuales permitieron validar con el *product owner* aspectos relacionados con la disposición de las

Nombre	Inicio sesión con Microsoft 365
Objetivo	Iniciar sesión en el sistema.
Actores	Usuario, sistema Microsoft 365.
Descripción	Este caso de uso consiste en iniciar sesión en el sistema a través de Microsoft 365.
Precondiciones	■ Microsoft 365 se encuentra funcionando. ■ El usuario no está logueado en el sistema.
Flujo principal	1. Usuario: ingresa a la URL correspondiente al login. 2. Usuario: hace clic en el botón para iniciar sesión con Microsoft. 3. Sistema: redirige al usuario al sitio de Microsoft 365. 4. Microsoft 365: solicita las credenciales. 5. Usuario: ingresa sus credenciales en Microsoft 365. 6. Microsoft 365: redirige al usuario al sistema. 7. Sistema: completa el inicio de sesión y redirige a la página principal.
Flujo alternativo	■ 5.A El usuario no tiene usuario de Microsoft 365 asociado a Pyxis: ● 5.A.1 Microsoft 365: muestra un mensaje de error y finaliza sin éxito el inicio de sesión. ■ 6.A El usuario ingresa credenciales incorrectas: ● 6.A.1 Microsoft 365: solicita nuevamente las credenciales. ● 6.A.2 Se vuelve al paso 5.
Postcondiciones	■ El usuario queda con su sesión iniciada en el sistema, pudiendo ver la información asociada al mismo. ■ Si es el primer inicio de sesión del usuario, se almacenan sus datos en el sistema.

Tabla 4.1: Caso de uso: Inicio de sesión con Microsoft 365

pantallas, funcionalidades y flujos de usuario. Posteriormente, estos prototipos también sirvieron como base para las reuniones con la diseñadora UX/UI de Pyxis, en las cuales se trabajó en la mejora de la experiencia de usuario y en la definición final de los diseños de la interfaz.

En base a lo definido en las etapas iniciales del proyecto, se especificaron los requisitos correspondientes a la primera liberación, orientados a construir una versión funcional del sistema que permita gestionar un proyecto individual utilizando métricas básicas para su planificación y seguimiento.

4.1.2. Requisitos de la primera liberación

En primer lugar, en cuanto a la autenticación, el sistema debe permitir a los usuarios iniciar sesión utilizando sus credenciales de Microsoft 365. El sistema debe permitir cerrar sesión de forma segura. Asimismo, el acceso debe estar restringido a usuarios autorizados de Pyxis, lo que simplifica la gestión de autenticación dentro de la aplicación.

En cuanto a la administración de proyectos, el sistema debe permitir a cada usuario crear un único proyecto. Dentro de este proyecto, el sistema debe permitir crear una o varias tareas, las cuales representan las unidades básicas de trabajo. Cada tarea debe contar con distintos estados que reflejen su progreso, incluyendo la posibilidad de marcarla como completada. Una vez que una tarea pasa al estado de “Completada”, no puede ser editada, salvo que sea desmarcada como completada. El sistema debe permitir también eliminar tareas en caso de ser necesario.

Asimismo, se debe poder reordenar las tareas dentro de la página correspondiente, ya que el orden de las mismas representa la prioridad que se tiene para dedicarle esfuerzo. En este sentido, las tareas completadas deben ubicarse al inicio de la lista y no pueden ser reordenadas, mientras que las tareas pendientes o en progreso pueden moverse libremente, manteniendo la posición definida por el usuario.

Por otro lado, en cuanto a la gestión del calendario, el sistema debe permitir editar las horas del calendario, las cuales representan la cantidad de tiempo semanal que un integrante puede dedicar al proyecto. Estas horas deben definirse inicialmente al momento de crear el proyecto y el sistema debe permitir ajustarlas posteriormente según cambios en la planificación, permitiendo reflejar de forma más realista la disponibilidad de cada usuario.

En relación a la visualización de métricas, el sistema debe mostrar información que permita comparar lo planificado con lo realmente ejecutado. En particular, se deben considerar variables como el valor ganado, las horas de calendario y las horas registradas en tareas completadas. El sistema debe presentar esta información tanto para la última semana como de forma acumulada desde el inicio del proyecto hasta el último domingo, lo que permite analizar la evolución del proyecto en el tiempo.

El sistema debe incluir una representación gráfica que facilite el análisis visual del avance del proyecto. En este sentido, el sistema debe mostrar una gráfica con dos líneas: una que represente la evolución real del valor ganado a

lo largo del tiempo y otra que indique cómo debería haber evolucionado según la planificación inicial, considerando las tareas y las semanas definidas en el calendario (i.e. son las representaciones gráficas de las funciones del valor ganado planificado y el valor ganado en función del tiempo). Esto permite identificar desvíos entre lo planificado y lo ejecutado de forma clara y rápida.

Finalmente, el sistema debe permitir visualizar la plataforma en más de un idioma. En esta primera versión, el sistema debe soportar español e inglés mediante un sistema de traducciones, lo que mejora la experiencia de usuario.

En conjunto, estos requisitos definen una primera versión del sistema que, si bien fue acotada en alcance, permitió cubrir las funcionalidades esenciales de gestión y seguimiento de un proyecto, sentando las bases para su evolución en la siguiente liberación.

4.2. Segunda liberación

La subsección 4.2.1 presenta la gestión de requisitos mientras que la subsección 4.2.2 presenta los requisitos de la segunda liberación.

4.2.1. Gestión de requisitos de la segunda liberación

En la segunda liberación, el proceso de obtención de requisitos tuvo un enfoque combinado. Por un lado, varios requisitos fueron obtenidos antes de comenzar la liberación, a partir del *feedback* recibido tras la primera versión del sistema. Por otro lado, nuevos requisitos surgieron mediante instancias con el *product owner* y de discusiones internas del equipo.

Durante esta liberación se destinó un tiempo inicial de aproximadamente dos semanas al descubrimiento y definición de los nuevos requisitos, así como a la consolidación de la visión del alcance final del proyecto de grado. Esto permitió delimitar qué requisitos se iban a dejar fuera debido a que era imposible, por temas de esfuerzo y cronograma, abordarlos en el proyecto de grado. También, como normalmente sucede en el desarrollo de software, surgieron algunos requisitos adicionales mientras el equipo ya se encontraba trabajando en esta última liberación de la aplicación.

En una primera instancia, los requisitos fueron especificados como historias de usuario. Luego, a solicitud del *product owner*, se decidió crear un manual de usuario de la plataforma, el cual permitió complementar la especificación funcional desde una perspectiva orientada al uso final del sistema. El manual fue elaborado con el objetivo de validar los requisitos junto al *product owner* y, al mismo tiempo, servir como herramienta de apoyo para los usuarios de Pyxis que participarían en las pruebas iniciales de uso de la aplicación, facilitando la comprensión y el uso de la misma.

El objetivo principal fue evolucionar lo construido en la primera liberación para acercarlo a la necesidad de la organización de Pyxis, incorporando soporte para múltiples usuarios y múltiples proyectos.

4.2.2. Requisitos de la segunda liberación

En primer lugar, en cuanto a la gestión de proyectos y usuarios, el sistema debe permitir que un usuario pueda pertenecer a múltiples proyectos y que cada proyecto pueda contar con varios integrantes. Asociado a esto, el sistema debe contar con un manejo de roles tanto a nivel global como a nivel de proyecto.

A nivel global, el sistema debe contemplar tres roles: administrador, operador e integrante de equipo. Estos roles y sus principales funcionalidades se muestran en la figura 4.1.



Figura 4.1: Roles a nivel del sistema.

Nota. Imagen generada mediante ChatGPT (OpenAI, 2026) a partir de un prompt elaborado por los autores.

El rol de operador debe permitir la gestión de los distintos proyectos del sistema. Esta gestión incluye la creación de nuevos proyectos, la edición de la información de los mismos y la posibilidad de agregar y eliminar integrantes del equipo.

El rol de administrador debe incluir las capacidades del operador y, adicionalmente, debe poder asignar roles a nivel de sistema a otros usuarios. Los integrantes de equipo corresponden a aquellos usuarios que no se encuentran asignados como operadores ni como administradores y no tienen asociada ninguna funcionalidad a nivel de sistema.

Por otro lado, a nivel de proyecto, el sistema debe contemplar los roles de líder e integrante. Estos roles y sus principales funcionalidades se muestran en la figura 4.2.



Figura 4.2: Roles a nivel del proyecto.

Nota. Imagen generada mediante ChatGPT (OpenAI, 2026) a partir de un prompt elaborado por los autores.

Los usuarios con rol de líder deben poder administrar el proyecto que lideran y gestionar las tareas relacionadas al mismo. Esta gestión incluye la edición de su información, la posibilidad de agregar y eliminar integrantes del equipo y la creación y edición de los campos de las tareas asociadas. Además, los integrantes con este rol también tienen la posibilidad de modificar el plan de cualquiera de los integrantes del equipo. El sistema debe permitir asignar a un usuario como líder al momento de la creación del proyecto y se debe poder también asignar este rol a otros integrantes con posterioridad. Cada proyecto debe tener en todo momento al menos un usuario asignado como líder.

En cuanto a la gestión de tareas, el sistema debe incorporar mejoras orientadas a un seguimiento más detallado del trabajo. En particular, el sistema debe permitir registrar el tiempo dedicado a cada tarea mediante múltiples registros de esfuerzo. Estos registros deben poder crearse tanto de forma manual como a través de un mecanismo de temporizador. El sistema debe permitir iniciar y detener este temporizador desde la interfaz, generando automáticamente los registros correspondientes. Esto permite obtener métricas más precisas sobre el esfuerzo real de cada usuario en sus tareas.

Asimismo, a partir de esta liberación y con respecto a las tareas, el sistema debe permitir editar la fecha de finalización de las mismas, reasignar tareas que aún no se encuentren completadas a un integrante del equipo del proyecto y por último cada usuario debe poder visualizar una vista individual de sus propias tareas y medidas de gestión (e.g. valor ganado), facilitando el seguimiento del avance personal dentro del proyecto.

En cuanto al calendario, el sistema debe brindar a los líderes la posibilidad de editar la disponibilidad horaria de todos sus integrantes. Para quienes no cumplan este rol dentro del proyecto, el sistema debe solamente permitirles ver su propia disponibilidad horaria dentro del mismo y poder cambiarla.

En relación a las métricas, el sistema debe ampliar las funcionalidades de la primera liberación incorporando nuevas visualizaciones y cálculos. En particular, el sistema debe permitir visualizar métricas individuales por usuario, además de las métricas globales del proyecto. También se debe incorporar mejoras en las gráficas, incluyendo dos nuevas líneas de proyección que permitan estimar la evolución futura del proyecto según distintos criterios.

Ambas líneas representan diferentes formas de calcular proyecciones en función del valor ganado. La primera se basa en un cálculo de replanificación, que asume que, a partir del estado actual del proyecto, el trabajo restante se ejecutará conforme a las estimaciones originales. Es decir, si bien se reconocen posibles sobreestimaciones o subestimaciones en tareas ya finalizadas, así como desvíos respecto al calendario, estos no se trasladan al resto del plan. En consecuencia, la proyección se construye considerando que el trabajo pendiente seguirá el comportamiento originalmente planificado. El cálculo completo de la línea de proyección correspondiente a la replanificación se presenta en el [anexo B.4.3](#).

Por su parte, la segunda línea, basada en un cálculo de pronóstico, considera los desvíos observados hasta el momento —tanto en la estimación de tareas como en el tiempo efectivamente trabajado— y asume que estos se mantendrán

constantes en el futuro con respecto a lo planificado. El cálculo completo de la línea de proyección correspondiente al pronóstico se presenta en el anexo [B.4.4](#).

Además, para cada una de las líneas de proyección mencionadas, se debe indicar una fecha prevista de finalización del proyecto, basándose en los cálculos correspondientes. Estas fechas deben estar disponibles tanto en la vista de proyecto como en la vista individual. Los cálculos para ambas fechas de fin se presentan en el anexo [B.3](#).

Por otro lado, el sistema debe incorporar funcionalidades de administración. En particular, solo los usuarios con roles de administrador deben poder acceder a una vista de administración que permita gestionar los roles globales de los usuarios.

Para mejorar la administración de proyectos por parte de usuarios con roles de operador y administrador, el sistema debe incluir una vista de proyectos que permita visualizar todos los proyectos existentes.

En relación a la administración de proyectos, se debe poder modificar los datos principales de un proyecto y gestionar los integrantes del equipo desde una misma sección. En este contexto, el sistema no debe restringirse a un único líder por proyecto. Los usuarios con rol de líder deben poder agregar nuevos integrantes, modificar sus roles y ajustar su dedicación semanal estándar.

Finalmente, en cuanto a la navegación general, el sistema debe permitir seleccionar el proyecto activo desde la interfaz, facilitando el trabajo en contextos donde un usuario participa en múltiples proyectos. Asimismo, los usuarios con los permisos adecuados deben poder crear nuevos proyectos, definiendo sus datos básicos y configurando inicialmente el equipo de trabajo.

Capítulo 5

Diseño arquitectónico

En esta sección se expone la arquitectura y el diseño de la base de datos propuestos y utilizados en el proyecto. El diseño arquitectónico del sistema fue abordado con la intención de construir una solución modular, clara y escalable, alineada con prácticas utilizadas en Pyxis. Desde el comienzo, el objetivo no fue simplemente desarrollar una aplicación que funcionara, sino diseñar una arquitectura que pudiera evolucionar a lo largo de las distintas liberaciones del proyecto e incluso más allá del alcance establecido para este proyecto de grado.

La arquitectura general del sistema fue definida siguiendo un enfoque cliente-servidor. A nivel global, el sistema se compone de un cliente web que interactúa con los usuarios y un servidor que expone una API mediante la cual se gestionan las operaciones del sistema. La comunicación entre ambos componentes se realiza a través de protocolos HTTP/HTTPS y utilizando formatos de intercambio de datos estructurados. El diagrama de componentes del sistema se presenta en la figura 5.1.

Este enfoque fue utilizado ya que permite separar claramente las responsabilidades entre la interfaz de usuario y la lógica del negocio. Esta separación facilita el mantenimiento del sistema, la escalabilidad y la posibilidad de evolucionar cada componente de forma relativamente independiente mediante cada iteración.

Internamente, el servidor se organizó siguiendo una arquitectura en capas. En particular, se definieron tres niveles principales:

- Una capa de presentación, encargada de recibir las solicitudes externas y exponer los *endpoints* del sistema
- Una capa de lógica de negocio, donde se implementan las reglas del dominio, los cálculos de métricas y las validaciones necesarias.
- Una capa de acceso a datos, responsable de la interacción con la base de datos y la persistencia de la información.

Esta organización en capas permite mantener una clara separación de responsabilidades dentro del *backend*, reduciendo el acoplamiento entre componentes

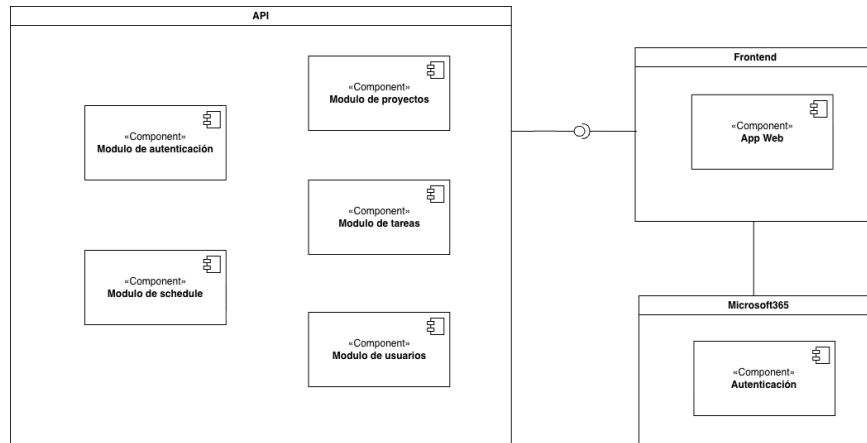


Figura 5.1: Diagrama de componentes del sistema.

y favoreciendo la mantenibilidad del código.

En cuanto a la persistencia, el sistema utiliza una base de datos relacional para almacenar la información de usuarios, proyectos, tareas, esfuerzos y calendarios. La base de datos elegida fue PostgreSQL. Esta elección se realizó por recomendación del equipo de Pyxis.

De esta forma, la arquitectura combina un esquema cliente-servidor a nivel de sistema con una organización en capas a nivel interno del servidor, permitiendo una estructura clara y alineada con prácticas profesionales de Pyxis.

Desde el punto de vista del despliegue, el sistema se ejecuta sobre infraestructura en AWS, utilizando Kubernetes (EKS con Fargate) para la orquestación de contenedores. La figura 5.2 presenta el diagrama de despliegue.

Esta estructura de despliegue fue recomendada por el equipo de arquitectura de Pyxis, con quienes se mantuvieron varias instancias de intercambio a lo largo del proyecto. Durante estas instancias, los arquitectos revisaron los diagramas propuestos, realizaron sugerencias sobre la arquitectura general del sistema y brindaron recomendaciones específicas relacionadas con el uso de servicios de AWS y la organización del despliegue.

Además, el equipo de arquitectura e infraestructura acompañó activamente el proceso de despliegue, ayudando a resolver distintos problemas vinculados a la configuración de la infraestructura en la nube.

En particular, era la primera experiencia de Pyxis brindando recursos a un proyecto externo a la organización, por lo que no contaban con un conjunto de permisos previamente definidos y aprobados. Debido a esto, fue necesario trabajar en conjunto con Pyxis en la definición y ajuste de los permisos de acceso dentro de la nube, a medida que el equipo avanzaba en la configuración del despliegue. Estos permisos determinan las acciones que cada usuario o servicio puede realizar dentro del entorno.

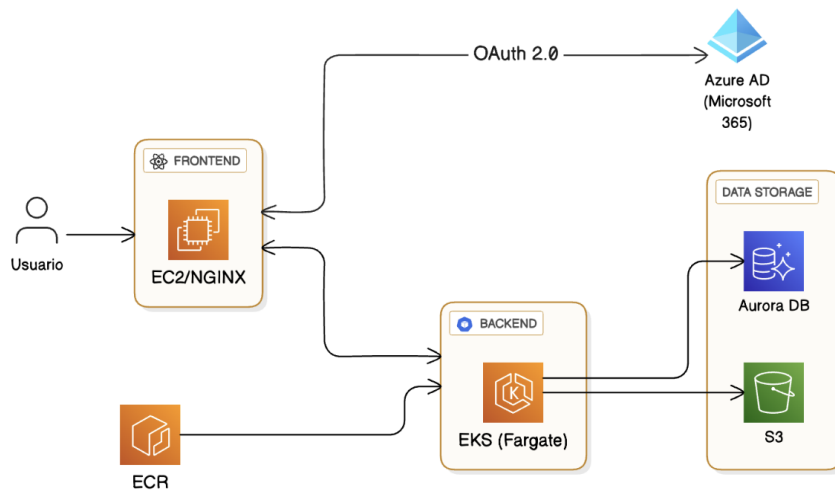


Figura 5.2: Diagrama de despliegue del sistema.

También se trabajó en la configuración de OIDC (OpenID Connect), que es un mecanismo que permite establecer una relación segura de confianza entre el clúster donde se ejecuta la aplicación y los servicios de la nube. En términos simples, esta configuración permite que los componentes del sistema puedan autenticarse correctamente y acceder a otros servicios sin necesidad de almacenar credenciales de forma insegura.

Otro punto relevante fue la instalación y configuración del Load Balancer Controller. Este componente se encarga de crear y administrar automáticamente un balanceador de carga, es decir, un mecanismo que recibe las solicitudes externas de los usuarios y las distribuye hacia la aplicación que se encuentra ejecutándose en la nube. Sin este componente, la aplicación no podría ser accesible desde internet de forma adecuada.

Finalmente, también se validó el uso de EKS, que es el servicio administrado de orquestación de contenedores. Este servicio permite ejecutar la aplicación en contenedores y gestionar su escalabilidad y disponibilidad. La validación por parte del equipo de arquitectura fue importante para confirmar que el enfoque adoptado era correcto y estaba alineado con buenas prácticas.

Este acompañamiento fue clave para poder finalizar exitosamente la puesta en producción del sistema, ya que permitió resolver obstáculos técnicos que excedían la experiencia inicial del equipo y garantizar que la infraestructura estuviera configurada de manera adecuada.

Las decisiones arquitectónicas finales son el resultado de un proceso iterativo que combinó el trabajo del equipo del proyecto con la visión y experiencia del equipo de arquitectura de Pyxis.

5.1. Primera liberación

En la primera liberación se definió un modelo de datos pensado para un escenario simple: un único usuario trabajando sobre un único proyecto. Aún así, desde el comienzo se buscó que la base de datos no quedara “cerrada” a esa situación, sino que pudiera adaptarse a futuros cambios sin tener que volver a diseñar todo desde cero. Por este motivo, varias decisiones se tomaron considerando la evolución prevista del sistema en las liberaciones siguientes.

5.1.1. Modelo de datos

El modelo se organizó alrededor de cuatro entidades principales: usuarios, proyectos, tareas y schedules. La figura 5.3 presenta un diagrama con las tablas creadas en la base de datos.

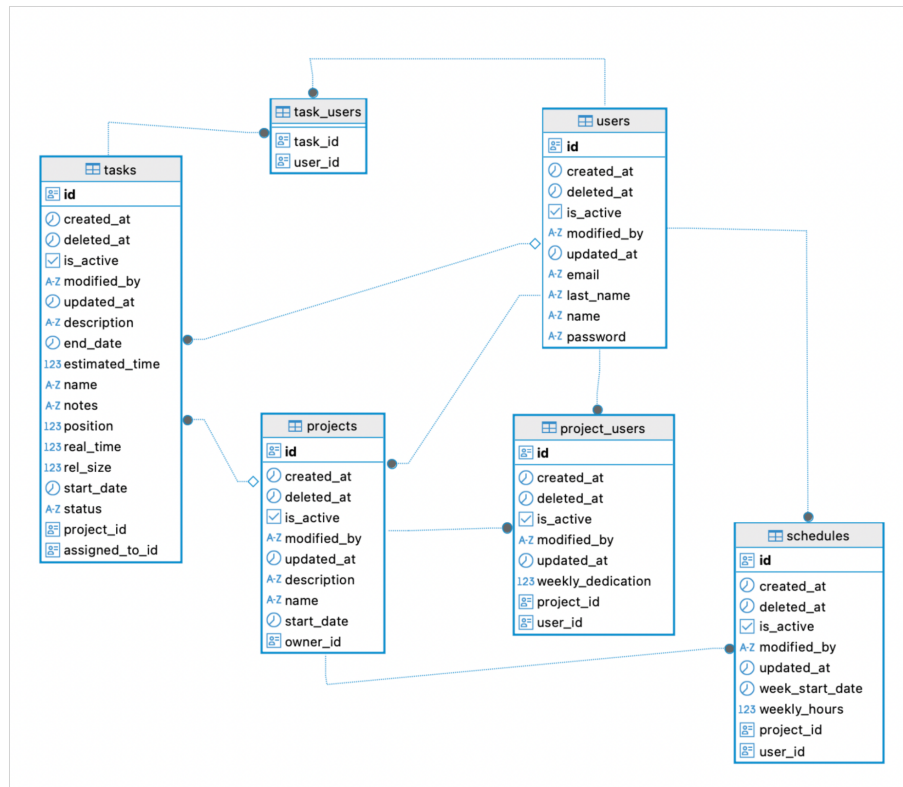


Figura 5.3: Diagrama del modelo de datos relacional correspondiente a la primera liberación.

La tabla *users* representa a los usuarios del sistema e incluye datos básicos

como nombre y correo electrónico. En esta primera iteración todavía no existía un manejo complejo de roles a nivel de sistema, y la lógica estaba orientada a permitir que cada usuario operará con un único proyecto activo.

La entidad *projects* representa los proyectos que se crean dentro de la plataforma. Para cada proyecto se almacena información general como nombre, descripción y fecha de inicio. Además, se incluyó el campo *owner_id* con el objetivo de identificar al usuario propietario del proyecto y dejar establecida una referencia clara a nivel de datos.

Por su parte, la tabla *tasks* representa las tareas asociadas a un proyecto. Cada tarea incluye información necesaria para el funcionamiento principal de la aplicación en esta liberación: el tiempo estimado (*estimated_time*), el tiempo real (*real_time*), el estado, una posición para permitir el ordenamiento de tareas, y el usuario asignado. En este modelo, las tareas no existen de forma independiente, sino que forman parte de un proyecto, estableciendo una relación directa mediante el campo *project_id*. Esto refleja que el trabajo gestionado en la herramienta se organiza siempre en el contexto de un proyecto, y no como tareas aisladas.

Aunque en esta primera liberación el sistema se manejaba con un solo usuario por proyecto, se incluyó desde el comienzo una tabla intermedia llamada *project_users*. Esta decisión fue tomada para preparar el modelo para una futura versión con múltiples usuarios por proyecto. De esta forma, cuando se incorporó esa funcionalidad en la segunda liberación, fue posible avanzar sin necesidad de un rediseño estructural completo.

Además, se definió la entidad *schedules*, que permite almacenar cuántas horas semanales tiene planificadas cada usuario para un proyecto en una semana determinada. Esta tabla es clave para varias métricas del sistema, ya que el calendario de dedicación afecta directamente los cálculos de planificación y proyecciones.

Finalmente, se agregó también una tabla intermedia *task_users*, pensada para asociar tareas con usuarios. Si bien en la primera versión lo habitual era que cada tarea tuviera un único usuario asignado, la estructura ya quedaba preparada para soportar múltiples relaciones en el futuro.

En cuanto a criterios generales del diseño, desde esta primera versión se decidió trabajar con un esquema de “soft delete”, utilizando campos como *is_active* y *deleted_at*, además de incorporar campos básicos de auditoría. El objetivo de estas decisiones fue mantener trazabilidad y evitar la pérdida de información ante eliminaciones, algo relevante para un producto donde el histórico de datos puede tener valor.

5.1.2. Arquitectura de la lógica

Siguiendo la arquitectura en capas previamente mencionada, cada uno de los módulos principales del sistema —usuarios, proyectos, tareas y calendario— fue implementado mediante un conjunto de clases organizadas en controladores, servicios, repositorios y entidades.

En el caso del módulo de usuarios, se definieron clases como *UserController*, *UserService*, *UserRepository* y la entidad *User*. El *UserController* expone los *endpoints* relacionados a la obtención de información del usuario, mientras que *UserService* centraliza la lógica asociada, como la identificación del usuario autenticado. Por su parte, *UserRepository* permite acceder a la base de datos. Dado que en esta primera liberación el manejo de usuarios era acotado, este módulo se centraba principalmente en identificar al usuario y su relación con el proyecto activo.

Para el módulo de proyectos, se implementaron clases como *ProjectController*, *ProjectService*, *ProjectRepository* y la entidad *Project*. El *ProjectController* define *endpoints* para la creación, edición y consulta de proyectos, mientras que *ProjectService* contiene la lógica de negocio asociada, incluyendo validaciones y la gestión de la relación entre el proyecto y su usuario propietario. Este módulo constituye el eje central del sistema, ya que organiza el resto de la información.

Por su parte, el módulo de tareas concentra gran parte de la funcionalidad de la aplicación. En este se definieron clases como *TaskController*, *TaskService*, *TaskRepository* y la entidad *Task*. El *TaskController* expone *endpoints* para la gestión de tareas, mientras que *TaskService* implementa la lógica asociada a la creación, edición, eliminación y actualización de estado de las tareas, así como su ordenamiento dentro del proyecto. Esta lógica se centraliza en la capa de servicios para evitar duplicación y facilitar el mantenimiento.

Finalmente, el módulo de calendario (*schedule*) fue implementado mediante clases como *ScheduleController*, *ScheduleService*, *ScheduleRepository* y la entidad *Schedule*. Estas clases permiten gestionar la planificación semanal de horas de trabajo por usuario, información que resulta fundamental para el cálculo de métricas y proyecciones del sistema.

Este modelo inicial fue compartido y revisado con los arquitectos de Pyxis, quienes aportaron su visión y validaron que la estructura fuera coherente y escalable para el crecimiento esperado en las siguientes liberaciones.

5.2. Segunda liberación

Una vez finalizada la primera liberación, el equipo comenzó a trabajar en la evolución del modelo de datos para soportar nuevos requerimientos. La segunda liberación incorporó múltiples usuarios, posibilidad de pertenecer a distintos proyectos, manejo de roles, y registro detallado del esfuerzo real. Como parte de este proceso, también se revisaron y ajustaron los diagramas de arquitectura, para asegurarse de que reflejaran correctamente lo que se estaba construyendo.

Luego de una reunión con los arquitectos de Pyxis con el objetivo de validar los cambios realizados tanto al diagrama del modelo de datos como al diagrama de componentes, y confirmar que el enfoque adoptado para AWS y Kubernetes fuera el correcto, se obtuvo el diagrama de modelo de datos que se presenta en la figura 5.4.

5.2.1. Modelo de datos

En esta iteración, el modelo se amplió para cubrir cuatro necesidades principales: permitir que un usuario pueda pertenecer a varios proyectos, manejar roles tanto a nivel de sistema como a nivel de proyecto, registrar esfuerzos reales en tareas con mayor detalle y mejorar el control de permisos asociado a cada acción. La figura 5.4 muestra el diagrama del modelo de datos relacional construido para esta versión del sistema.

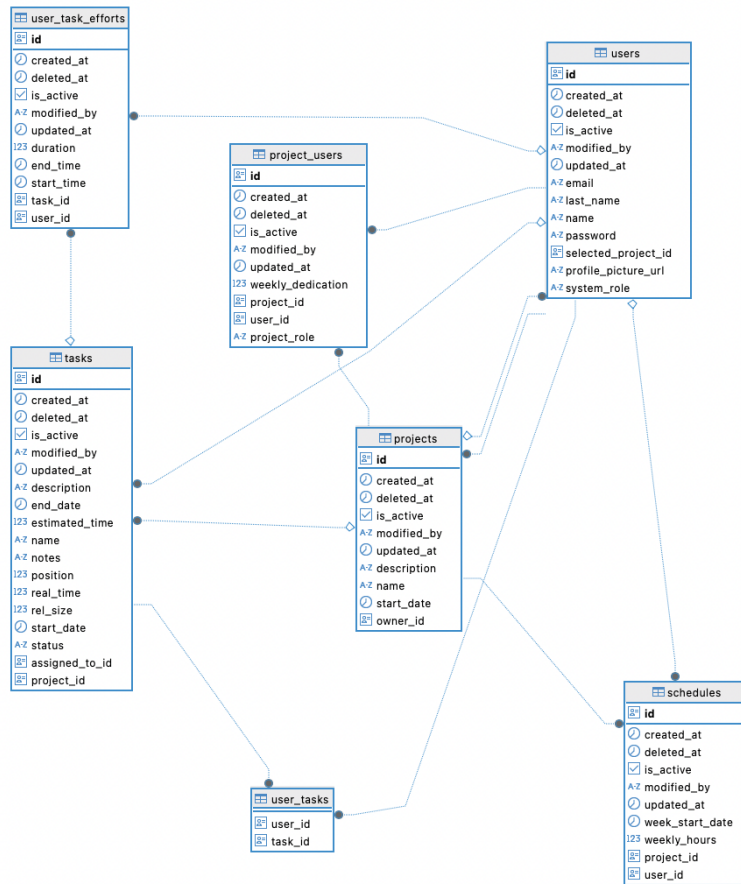


Figura 5.4: Diagrama del modelo de datos relacional correspondiente a la segunda liberación.

Uno de los cambios más importantes fue la incorporación de una nueva entidad llamada *user-task-efforts*. Esta tabla permite registrar cada esfuerzo

individual realizado por un usuario en una tarea, incluyendo tiempos de inicio y fin, duración, y referencias tanto al usuario como a la tarea. La incorporación de este registro detallado fue clave para poder calcular métricas con mayor precisión, como el esfuerzo real acumulado, desvíos de estimación y líneas de proyección a futuro.

Además, la tabla *project_users* se extendió incorporando el campo *project_role*, lo que permitió representar roles a nivel de proyecto. Esto fue necesario para manejar situaciones donde un usuario puede tener distintos permisos dependiendo del proyecto en el que está trabajando. Este campo fue definido como un enumerado, cuyas opciones son los posibles roles a nivel de proyecto.

Por otro lado, la entidad *users* también fue ampliada con el campo *system_role*, para reflejar roles globales del sistema. Similar al campo *project_role* en *project_users*, este campo también fue definido como un enumerado, cuyas opciones son los posibles roles a nivel de sistema. Adicionalmente, se incluyó un campo *selected_project_id* para representar el proyecto seleccionado por el usuario dentro de la aplicación.

5.2.2. Arquitectura de la lógica

Además de los cambios en el modelo de datos, la segunda liberación implicó una evolución significativa en la arquitectura de la lógica del sistema. Los módulos existentes fueron extendidos para soportar escenarios multiusuario, manejo de roles y registro detallado del esfuerzo real, incorporando nuevas responsabilidades tanto en la capa de servicios como en la de controladores.

En esta etapa, la gestión de permisos se consolidó mediante un enfoque combinado: restricciones declarativas en los controladores, validaciones de negocio en los servicios y verificaciones de pertenencia y rol dentro del proyecto mediante componentes de seguridad específicos. Esto permitió mantener una separación clara entre autorización técnica y reglas funcionales del dominio.

En el módulo de usuarios, la principal evolución fue la incorporación de roles a nivel de sistema. A partir de ello, se habilitaron controles de acceso diferenciados para operaciones administrativas, incluyendo la actualización de roles de sistema bajo restricciones explícitas.

El módulo de proyectos incorporó soporte completo para múltiples usuarios por proyecto, junto con la administración de roles de proyecto. La lógica de negocio se amplió para permitir altas, actualizaciones y bajas de integrantes, contemplando validaciones clave como la existencia de al menos un líder activo por proyecto.

El módulo de tareas fue extendido para soportar reasignación de tareas, validaciones basadas en rol y control de acciones según el vínculo del usuario con la tarea y el proyecto. Asimismo, se fortaleció su integración con los cálculos de métricas del sistema.

Además, se incorporó un módulo específico para el registro de esfuerzos reales (*UserTaskEffort*), implementado mediante controlador, servicio y repositorio. Este módulo permite registrar, actualizar y eliminar dedicaciones de tiempo, y constituye una fuente central para el cálculo de métricas y proyecciones.

Por su parte, el módulo de calendario (*schedule*) también evolucionó para contemplar escenarios multiusuario, permitiendo que usuarios con rol de líder gestionen la disponibilidad de otros integrantes del proyecto, además de mantener las operaciones individuales de cada usuario sobre su propio calendario.

Desde el punto de vista arquitectónico, la segunda liberación no solo representó una expansión del modelo de datos, sino también una consolidación de la lógica del sistema para escenarios de mayor complejidad. En particular, la capa de servicios asumió un rol más fuerte en la aplicación de reglas de negocio y validaciones de permisos, mientras que la capa de controladores reforzó el acceso seguro a los *endpoints* mediante políticas de autorización explícitas.

Las modificaciones del modelo y de la arquitectura fueron presentadas y discutidas con los arquitectos de Pyxis durante una reunión. En esa instancia validaron que el enfoque adoptado para roles y asociaciones múltiples era correcto. En particular, sugirieron utilizar anotaciones propias de Spring Boot para controlar la autorización en los *endpoints* del *backend*, recomendación que luego fue aplicada y resultó útil para ordenar el control de permisos dentro del código.

También se mencionó la posibilidad de asignar *scopes* o privilegios dentro del token JWT. Sin embargo, para aplicar este enfoque era necesario que el proveedor de identidad (Azure AD) incluyera dichos *scopes* en el token, lo cual no estaba configurado para este proyecto. Por ese motivo, se decidió no implementar *scopes* y manejar los permisos de los usuarios desde la aplicación.

Otro punto importante que surgió de la revisión fue el manejo del “proyecto seleccionado”. Inicialmente, este valor se almacenaba en la base de datos dentro de la tabla *users*, pero los arquitectos señalaron que esto podía generar inconsistencias si un usuario utilizaba el sistema en múltiples navegadores o sesiones simultáneamente, ya que el valor se sobrescribiría.

A partir de esta observación, se ajustó la implementación para utilizar la sesión de NextAuth.js como fuente de verdad del proyecto activo en cada sesión. NextAuth.js es una solución de autenticación de código abierto para aplicaciones Next.js, cuyo uso se detalla más adelante en el capítulo de construcción.

De esta forma, el campo en la base de datos se mantuvo únicamente como referencia inicial, mientras que el *frontend* pasó a obtener el proyecto activo directamente desde la sesión, evitando inconsistencias entre distintas sesiones del mismo usuario.

Durante la misma reunión también se validó que el uso de AWS, EKS con Fargate y el enfoque de despliegue adoptado fueran adecuados. Los arquitectos confirmaron que la estructura utilizada estaba alineada con prácticas profesionales y que el camino elegido era correcto para el tipo de sistema desarrollado.

En conclusión, tanto la arquitectura como el modelo de datos fueron evolucionando a lo largo del proyecto y no se plantearon como un diseño cerrado desde el inicio. Cada modificación relevante fue discutida y validada con el equipo de arquitectura de Pyxis, lo cual permitió mejorar la calidad del diseño y reducir riesgos técnicos.

Capítulo 6

Proceso de construcción

En este capítulo se describe el proceso de construcción e implementación de la herramienta. A lo largo de esta etapa, el equipo no solo se enfocó en desarrollar las funcionalidades definidas en los requisitos, sino también en establecer una forma de trabajo ordenada y consistente que permitiera integrar cambios de manera continua, mantener la estabilidad del sistema y facilitar su despliegue en un entorno real.

En este sentido, se adoptaron prácticas comunes en el desarrollo de software, como el uso de control de versiones, la definición de flujos de trabajo basados en ramas, la revisión de código entre integrantes del equipo y la automatización de procesos mediante pipelines de integración y despliegue continuo.

Asimismo, el proceso de construcción estuvo fuertemente influenciado por la evolución de los diseños de la interfaz de usuario, los cuales fueron refinados de manera iterativa en conjunto con la diseñadora UX/UI. Esto implicó no solo implementar las interfaces propuestas, sino también adaptarlas y ajustarlas a medida que surgían nuevos requisitos o cambios en la definición del producto.

En las siguientes secciones se describen en detalle los distintos aspectos que conformaron este proceso, incluyendo las tecnologías utilizadas, la estrategia de control de versiones adoptada, la implementación de despliegues automáticos y la evolución de los diseños a lo largo de las distintas liberaciones.

6.1. Tecnologías utilizadas

Las tecnologías utilizadas en este proyecto de grado fueron definidas al inicio del mismo en conjunto con el *product owner*, quien indicó trabajar con herramientas alineadas al stack tecnológico utilizado en Pyxis.

Para el desarrollo del sistema se utilizó un conjunto de tecnologías orientadas a la construcción de aplicaciones web. En particular, el *frontend* fue desarrollado utilizando Next.js (versión 15.2.6), mientras que el *backend* se implementó utilizando Spring Boot (versión 3.4.4), encargado de la lógica de negocio, la exposición de *endpoints* y la gestión de la autenticación.

En cuanto a la persistencia de datos, se utilizó PostgreSQL como sistema de base de datos relacional. Esta elección permitió modelar de forma estructurada las entidades del sistema, así como garantizar la integridad y consistencia de la información almacenada.

Por otro lado, el despliegue de la aplicación se realizó sobre infraestructura en la nube, utilizando servicios de AWS junto con Kubernetes para la orquestación de contenedores. Esto permitió gestionar de forma centralizada los distintos componentes del sistema y facilitar su actualización.

6.2. Control de versiones

A lo largo de todo el proceso de desarrollo se utilizó GitHub como herramienta principal de control de versiones y colaboración. Esta plataforma permitió que cada integrante del equipo pudiera trabajar de forma independiente en distintas funcionalidades, integrando luego los cambios de manera controlada.

El flujo de trabajo se organizó principalmente en torno a dos ramas: *main* y *develop*. La rama *main* se utilizó para mantener la versión estable de la aplicación, es decir, aquella que se encontraba desplegada y disponible para ser utilizada en la web. Por otro lado, la rama *develop* se utilizó para integrar las nuevas funcionalidades que aún se encontraban en desarrollo.

Este esquema resultó especialmente útil hacia el final de la primera liberación. Mientras el equipo de Pyxis se encontraba probando la primera versión del sistema desplegado, el equipo de desarrollo continuaba trabajando en nuevas funcionalidades correspondientes a la segunda liberación dentro de la rama *develop*. De esta forma, fue posible avanzar en paralelo sin afectar la estabilidad de la versión que estaba siendo evaluada.

Una vez finalizadas las pruebas de la primera liberación y validadas las funcionalidades existentes, la rama *develop* fue integrada a *main*, lo que permitió unir las nuevas funcionalidades desarrolladas.

Adicionalmente, cada funcionalidad fue desarrollada en ramas específicas (*feature branches*), las cuales eran luego integradas mediante *pull requests*. Este mecanismo permitió realizar revisiones de código entre los integrantes del equipo, detectar errores de forma temprana y mantener una mejor calidad del código.

6.3. Despliegues automáticos

Como parte del proceso de construcción, se implementó un mecanismo de integración continua y despliegue automático (CI/CD) utilizando GitHub Actions. Este mecanismo permitió automatizar completamente el proceso de construcción y despliegue tanto del *backend* como del *frontend*, evitando tareas manuales y asegurando consistencia entre el código fuente y la versión desplegada.

Para esto, se definieron dos *workflows* (flujos de trabajo) independientes: uno para el *backend* y otro para el *frontend*. Ambos *workflows* se ejecutan au-

tomáticamente cuando se detectan cambios en la rama *main* del repositorio, siempre que dichos cambios afecten a la parte correspondiente del sistema.

En ambos casos, el proceso comienza con la obtención del código fuente actualizado y la autenticación contra la infraestructura de AWS. Para esto se utilizó un mecanismo basado en OIDC (OpenID Connect), que permite a GitHub Actions asumir un rol dentro de AWS sin necesidad de manejar credenciales estáticas. Esto mejora la seguridad del sistema, ya que evita exponer claves sensibles en el repositorio.

Una vez autenticado, el *workflow* realiza el proceso de construcción de una imagen de contenedor (Docker) para cada componente. En el caso del *backend*, la imagen se construye a partir del código del servidor, mientras que en el *frontend* se genera una imagen correspondiente a la aplicación web. Ambas imágenes son etiquetadas utilizando el identificador único del *commit* (SHA), lo que permite versionar cada despliegue de forma precisa, además de mantener una etiqueta adicional “latest”.

Luego de construir las imágenes, estas son publicadas en un repositorio de contenedores en la nube (ECR – Elastic Container Registry). Este repositorio actúa como almacenamiento centralizado de las imágenes que serán utilizadas posteriormente en el despliegue.

Una vez que la nueva imagen se encuentra disponible, el *workflow* continúa con la actualización del sistema en ejecución. Para esto, se establece conexión con el clúster de Kubernetes (EKS) y se actualiza la imagen utilizada por el deployment correspondiente, tanto para el *backend* como para el *frontend*.

Este proceso se realiza mediante comandos que indican al clúster que debe utilizar la nueva versión de la imagen. A partir de ese momento, Kubernetes se encarga automáticamente de reemplazar los contenedores existentes por nuevas instancias basadas en la imagen actualizada, asegurando que la transición se realice de forma controlada.

Además, el *workflow* espera a que el proceso de actualización (*rollout*) finalice correctamente antes de marcar la ejecución como exitosa. Esto permite detectar posibles errores durante el despliegue y asegurar que la nueva versión del sistema se encuentre funcionando correctamente.

Para parametrizar estos procesos, se configuraron variables de entorno y secretos dentro de GitHub, incluyendo datos como la región de AWS, el nombre del clúster, los repositorios de imágenes y los nombres de los *deployments*. Esta configuración permitió reutilizar los *workflows* y mantener separada la lógica del despliegue de los datos sensibles o específicos del entorno.

La implementación de este mecanismo permitió que cada cambio integrado en la rama principal del proyecto se reflejara automáticamente en el entorno desplegado, logrando un flujo continuo de integración y entrega.

6.4. Evolución de los diseños

Durante las primeras semanas del proyecto de grado, se planteó que, a lo largo de su desarrollo, probablemente se contaría con el apoyo de una diseñadora

UX/UI, quien asistiría al equipo en lo relativo al diseño de las distintas pantallas. Esto dependía de la asignación de horas por parte de Pyxis para su participación en el proyecto. Asumiendo que dichas horas serían aprobadas, y con el objetivo de brindarle contexto sobre las necesidades del sistema, el tutor sugirió al equipo analizar herramientas existentes y elaborar un primer borrador de las principales pantallas previstas, al menos para la primera liberación.

Una vez que se llegó a una versión estable de estos bosquejos, y habiéndolo validado con el *product owner*, el grupo tuvo una primera reunión con la experta en UX/UI. En dicha reunión se le pudo explicar en qué consistía la herramienta que se iba a construir, logrando que la diseñadora se empiece a familiarizar con las funcionalidades y pantallas que debía diseñar. Luego de dicho encuentro virtual, se le envió un documento con información de cada pantalla.

Además de funcionar como primer contacto entre el equipo y la diseñadora, el objetivo de la reunión introductoria fue que ella pudiera recopilar toda la información necesaria para estimar ante Pyxis la cantidad de horas requeridas para la elaboración de los diseños.

Tras confirmarse que Pyxis asignaría 25 horas de trabajo de la diseñadora al proyecto de grado, y luego de algunos intercambios y mejoras sobre los bosquejos elaborados, la diseñadora hizo llegar al equipo los diseños iniciales correspondientes a las pantallas solicitadas.

Una vez obtenidos estos diseños, el equipo brindó retroalimentación a la diseñadora respecto a cambios requeridos tanto en funcionalidades como en aspectos de diseño. Además, también se le facilitaron nuevos bosquejos de otras pantallas necesarias para esta primera liberación. Esta comunicación se llevó a cabo tanto por correo electrónico como mediante reuniones virtuales.

Tras varias rondas de revisiones, se llegó a la versión final de los diseños para la primera liberación. A modo de ejemplo, en la figura 6.1 se presenta el bosquejo creado por el equipo para la pantalla de Tareas, mientras que en la figura 6.2 se muestra el diseño realizado para por la experta en UX/UI. Los diseños de las pantallas restantes pueden encontrarse en el anexo C, sección C.1.

La herramienta utilizada por la diseñadora para la elaboración de los diseños a lo largo del proyecto fue Lovable. Según su documentación oficial, Lovable es una “plataforma de desarrollo full-stack impulsada por IA para crear, iterar y desplegar aplicaciones web utilizando lenguaje natural, con código real, seguridad y gobernanza empresarial.” (Lovable, 2026).

Los diseños iniciales obtenidos mediante esta plataforma resultaron de gran valor para el equipo. Además de proporcionar las interfaces, Lovable permite generar el código correspondiente en el lenguaje de programación seleccionado. Para este proyecto, se acordó con la diseñadora utilizar Next. Sin embargo, tras obtener el código y realizar una breve investigación, se concluyó que la herramienta brinda soporte para React como librería, pero no específicamente para Next.js como framework basado en React. Adicionalmente, Lovable utiliza Tailwind (un framework de CSS) en lugar de CSS puro.

Si bien el uso de Lovable permitió ahorrar varias horas de desarrollo, el equipo debió igualmente dedicar tiempo a adaptar el código generado en React a Next.js, dado que existen diferencias significativas entre ambos.

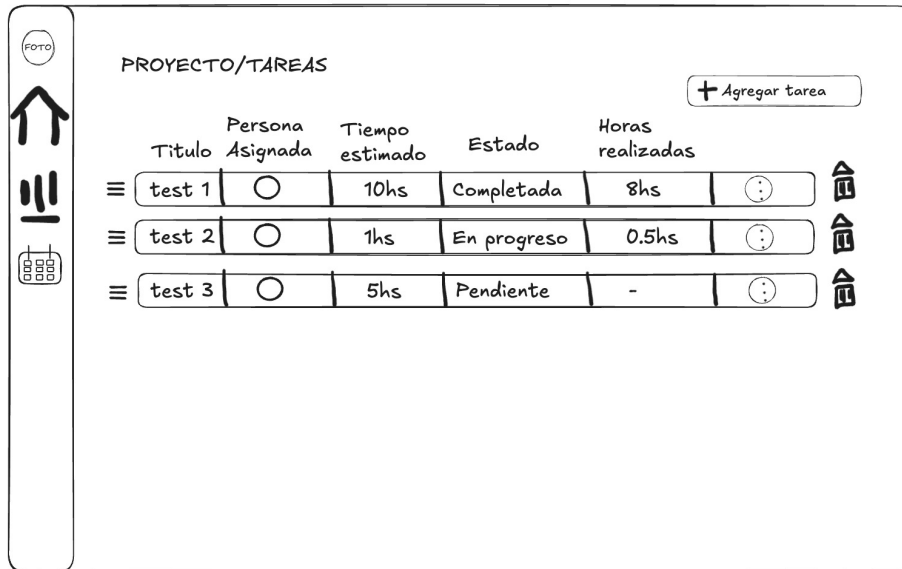


Figura 6.1: Bosquejo de la pantalla de Tareas.

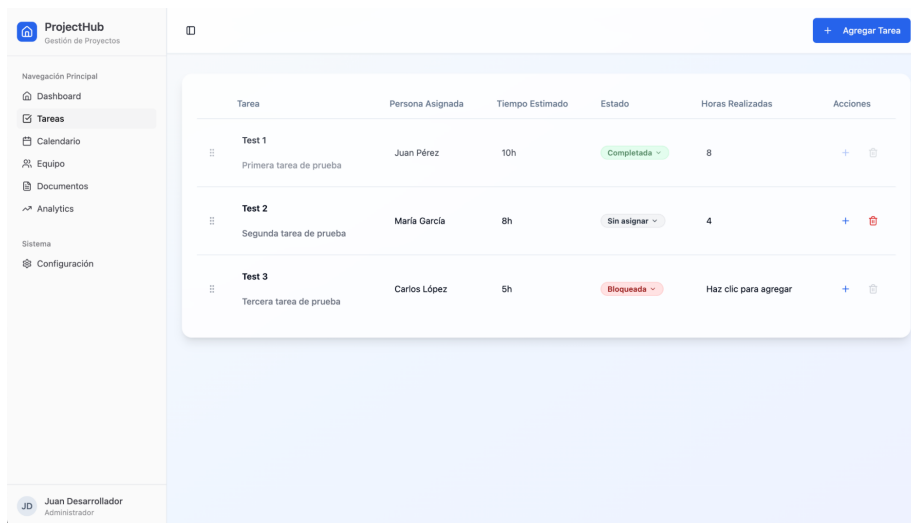


Figura 6.2: Diseño de la pantalla de Tareas.

Otro aspecto que el equipo debió analizar en relación al código generado por Lovable fue si convenía adoptar Tailwind, a pesar de no contar con experiencia previa en su uso, o si era preferible continuar utilizando CSS a través de Styled Components. Luego de intercambiar opiniones con el *product owner* y con la experta en UX/UI, se decidió mantener el uso de CSS.

Esta decisión puede interpretarse como una medida de mitigación de riesgos, ya que existía incertidumbre respecto a la disponibilidad de horas de la diseñadora para cubrir la totalidad de las pantallas necesarias, incluidas aquellas para la segunda liberación. En caso de no contar con dicho apoyo, la adopción de Tailwind implicaba asumir una curva de aprendizaje adicional y un posible impacto en los tiempos de desarrollo. En este contexto, optar por CSS resultaba una alternativa más segura, al tratarse de un lenguaje ya conocido por el equipo, lo que permitía reducir riesgos asociados a demoras y dificultades en la implementación.

Luego de obtenidos los diseños, el equipo comenzó a trabajar sobre los mismos, integrando lo ya implementado con las nuevas interfaces. La comunicación con la diseñadora continuó durante algunas instancias posteriores a la entrega de los primeros diseños, con el objetivo de realizar refinamientos surgidos a partir del relevamiento de nuevos requisitos o de funcionalidades cuyos comportamientos no coincidían completamente con lo planteado en los diseños iniciales. Sin embargo, las horas asignadas por Pyxis a la diseñadora alcanzaron solo para cubrir los diseños de la primera liberación.

En el contexto de la segunda liberación, y una vez relevados los nuevos requisitos asociados a la misma, el equipo se puso en contacto con el tutor para gestionar con Pyxis la asignación de horas adicionales de diseño destinadas a las nuevas funcionalidades y pantallas. Siguiendo una dinámica similar a la adoptada en la primera liberación, el equipo se reunió con la diseñadora y se presentaron bosquejos de las nuevas pantallas requeridas, acompañados de un documento con información detallada para facilitar la estimación de las horas necesarias.

Finalmente, Pyxis confirmó la asignación de horas adicionales de trabajo de la diseñadora, las cuales resultaron de gran valor para completar los diseños correspondientes a la última liberación.

Para esta liberación también hubo instancias de refinamientos, tanto para nuevas pantallas como para agregados en pantallas existentes. Luego de dichos refinamientos se llegó a una versión estable de los diseños, la cual cubría para las distintas pantallas los requerimientos necesarios para poder cumplir con las distintas funcionalidades que serían agregadas.

En la figura [6.3](#) se muestra como ejemplo el diseño final de la pantalla de tareas elaborado por la diseñadora para el equipo. Los diseños para las pantallas restantes se pueden encontrar en el anexo [C](#), en la sección [C.2](#).

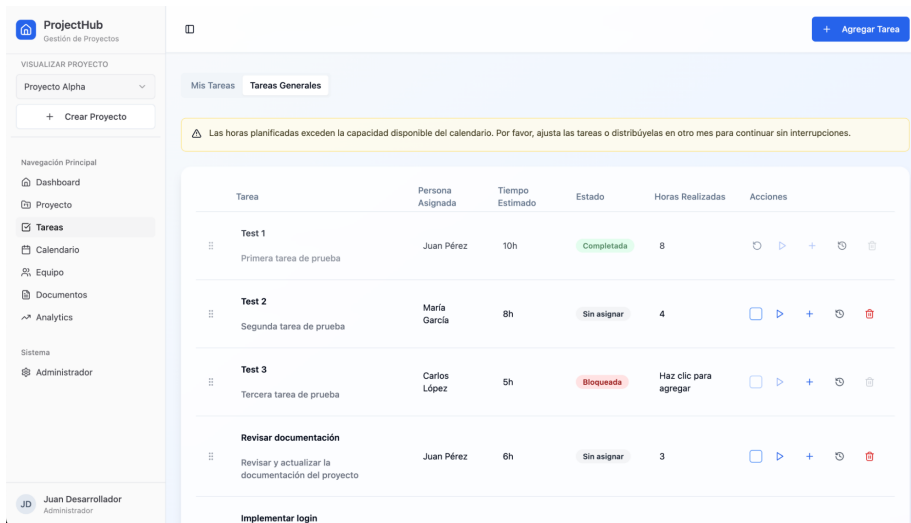


Figura 6.3: Diseño final de la pantalla de Tareas.

6.5. Proceso

Fue mencionado en el capítulo 3 que el proceso de desarrollo se dividió en dos grandes liberaciones, con el objetivo de lograr una entrega de valor temprana para el cliente mediante una versión funcional de la plataforma. Antes de comenzar con la implementación, el equipo utilizó las primeras semanas del proyecto para relevar los requisitos iniciales, planificar las liberaciones y comunicarse con los actores involucrados del lado de Pyxis.

El equipo de desarrollo se organizó principalmente mediante reuniones semanales fijas, complementadas con reuniones quincenales con el *product owner*, en las cuales se presentaban avances, se validaban funcionalidades y se relevaban nuevos requisitos.

Al inicio del proceso, se utilizó Jira como herramienta para centralizar las tareas a desarrollar, las cuales fueron escritas en forma de historias de usuario. Asimismo, se utilizó una planilla de Excel para registrar la cantidad de horas que cada integrante del equipo dedicaba semanalmente a cada tarea.

Avanzada la primera liberación, el equipo logró concluir el despliegue automático de la plataforma mediante GitHub Actions, como se mencionó en la sección 6.3 de este capítulo. A partir de ese momento, el equipo se comenzó a reunir al final de cada semana para desplegar los cambios realizados desde el entorno de desarrollo al entorno de producción. Durante estas instancias, también se planificaban las tareas a abordar en la semana siguiente.

Además de estas reuniones fijas, el equipo se reunía el día previo a los encuentros con el *product owner*, con el objetivo de preparar las demostraciones que se iban a realizar. Luego de estas reuniones, en las que se validaban re-

quisitos, el equipo priorizaba y generaba nuevas tareas para continuar con el desarrollo.

Hacia las últimas semanas de la primera liberación, el equipo comenzó a utilizar la aplicación de forma activa como usuarios finales, con el objetivo de validar su funcionamiento en un entorno real. Si bien esta primera liberación contemplaba inicialmente un único usuario por proyecto, la estructura de datos ya se encontraba preparada para soportar múltiples usuarios, tal como se definió en el diseño del modelo de datos.

Aprovechando esta característica, se realizó una asignación manual directamente en la base de datos para que todos los integrantes del equipo pertenecieran al mismo proyecto. Esto permitió simular un escenario multiusuario y validar el comportamiento de la aplicación bajo este contexto, aun cuando dicha funcionalidad no estaba completamente expuesta a nivel de interfaz de usuario.

A partir de este punto, se dejó de utilizar Jira para la gestión de tareas y se comenzó a utilizar la propia plataforma desarrollada. Inicialmente, se realizó una reunión de despegue en la cual se definieron todas las tareas restantes de esta liberación, junto con su estimación en horas. En esta instancia, también se asignaron las tareas a los integrantes del equipo. Con el avance de las semanas, cada integrante registraba el tiempo dedicado a sus tareas mientras trabajaba en ellas y, una vez finalizadas, las marcaba como completadas dentro de la aplicación.

Cuando se consideró que las funcionalidades desplegadas en el entorno de producción cumplían con los requisitos definidos para la primera liberación, se notificó al *product owner* y al equipo de Pyxis para que comenzaran a utilizar la herramienta. Para esta etapa, se elaboró un breve manual de usuario, en el cual se describían las principales funcionalidades disponibles y los comportamientos esperados. Este manual evolucionó posteriormente hacia un manual más completo como fue mencionado en el capítulo 4, en la sección 4.2.1, el cual abarca también las funcionalidades de la segunda liberación. Dicho manual está disponible en el anexo D .

La herramienta comenzó a ser utilizada por usuarios dentro de Pyxis, quienes reportaron fallas y sugirieron mejoras. Estas observaciones fueron priorizadas durante las semanas en las que se realizó la transición hacia la segunda liberación.

Para la segunda liberación, la plataforma fue utilizada desde el inicio como herramienta principal de gestión de este proyecto de grado. Durante las primeras semanas, se decidió pausar parcialmente el desarrollo de nuevas funcionalidades para enfocarse en la resolución de defectos reportados por los usuarios de Pyxis. En paralelo, se revisaron los requisitos que se habían definido al inicio del proyecto y se establecieron nuevos. También se planificaron los cambios necesarios en la arquitectura y en los diseños para soportar las nuevas funcionalidades.

Durante este período, se mantuvo un contacto frecuente con la diseñadora, como se mencionó en la sección 6.4 de este capítulo. En cuanto al equipo de arquitectura, fue consultado una vez que se contaba con una versión avanzada del nuevo diagrama de clases y con una propuesta encaminada para la incorporación de roles y permisos en la aplicación. A partir de esta instancia, se adoptaron

las recomendaciones que realizaron los arquitectos, las cuales se detallan en la sección 6.6 de este capítulo.

La segunda liberación también incluyó una reunión de despegue en la cual se definieron las tareas, categorizando cada una según la funcionalidad o pantalla correspondiente. En esta instancia, las tareas fueron estimadas en conjunto y asignadas a los integrantes del equipo.

Hasta la mitad de esta liberación, no fue posible realizar despliegues en el entorno de producción, ya que esto interferiría con las pruebas y los datos generados tanto por el equipo de desarrollo como por los usuarios de Pyxis. Durante este período, únicamente se desplegaba de forma inmediata las correcciones críticas correspondientes a la primera liberación.

Una vez alcanzada una versión estable de las funcionalidades de esta liberación, se consultó al equipo de Pyxis si continuaban realizando pruebas activamente en la aplicación, con el objetivo de coordinar un despliegue de mayor alcance que incluyera los avances acumulados hasta ese momento.

En esta etapa, se requirió un cuidado especial del entorno de producción, dado que estaba siendo utilizado por usuarios reales de la industria, con datos sensibles.

6.6. Implementación

6.6.1. Primera liberación

El objetivo principal de la primera liberación fue construir una versión inicial funcional del sistema que permitiera gestionar un único proyecto asociado a un único usuario, cubriendo las funcionalidades básicas de planificación y seguimiento del mismo.

En esta etapa, el foco estuvo puesto en implementar los flujos principales de la aplicación, incluyendo la creación y gestión de tareas, la dedicación semanal del usuario y la visualización de métricas que permitieran analizar el avance del proyecto. Asimismo, se buscó establecer una base sólida tanto a nivel de arquitectura como de modelo de datos, que permitiera extender el sistema en futuras iteraciones sin necesidad de realizar cambios estructurales significativos.

Adicionalmente, esta liberación tuvo como objetivo validar los principales requisitos del sistema en un contexto de uso real, permitiendo al equipo y al cliente interactuar con la herramienta.

Uno de los primeros desafíos durante la implementación fue definir una estructura clara y mantenible para el *backend*. Para ello, se optó por organizar la aplicación siguiendo una arquitectura en capas, separando responsabilidades entre los distintos componentes.

La estructura del *backend* se organizó siguiendo el patrón de arquitectura en capas, separando responsabilidades en controladores, servicios, repositorios y modelos. Para cada entidad del dominio se definieron clases correspondientes en cada capa. A modo de ejemplo, en la figura 6.4 se muestran las capas implementadas para la entidad Project y cómo estas interactúan entre sí, con el

cliente y con la base de datos.

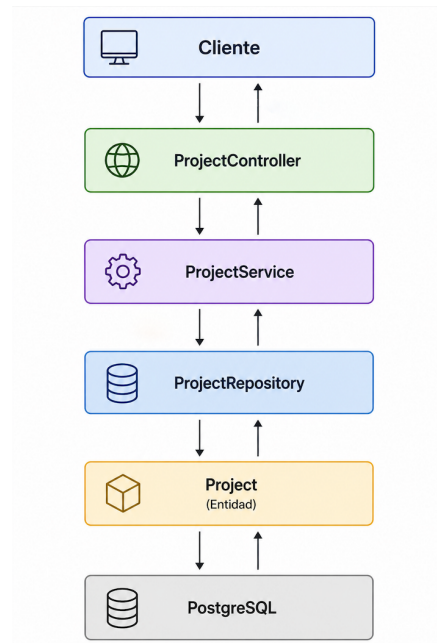


Figura 6.4: Interacción entre capas del backend para la entidad Project
Nota. Imagen generada mediante ChatGPT (OpenAI, 2026) a partir de un prompt elaborado por los autores.

En primer lugar, para las entidades contempladas en esta liberación, se desarrollaron los controladores *AuthController*, *ProjectController*, *TaskController*, *ScheduleController* y *UserController*. Cada uno de estos controladores expone los *endpoints* asociados a su entidad, definiendo las rutas a través de las cuales el *frontend* interactúa con la API. De esta manera, las operaciones relacionadas con proyectos, tareas o calendario quedan agrupadas en sus respectivos controladores, manteniendo una organización clara, consistente y fácil de mantener.

Por debajo de esta capa se encuentra la lógica de negocio, implementada en el paquete *service*. En este nivel se definieron servicios como *ProjectService*, *TaskService*, *ScheduleService* y *UserService*. Los servicios actúan como intermediarios entre los controladores y la capa de persistencia, centralizando las reglas del sistema y evitando duplicación de lógica.

Por debajo de la capa de servicios se ubica la capa de persistencia, implementada principalmente en el paquete *biz.repository*. Allí se definieron repositorios basados en Spring Data (por ejemplo *ProjectRepository*, *TaskRepository* y *ScheduleRepository*), responsables de encapsular el acceso a la base de datos y proveer métodos de consulta y guardado sobre las entidades.

En el modelo del dominio se encuentran las entidades que representan los

conceptos principales del sistema como son *Project*, *Task*, *User*, *Schedule*. Estas entidades son las que se persisten y sobre las cuales operan los repositorios y servicios.

Finalmente, para desacoplar el modelo interno de persistencia de los datos expuestos por la API, se utilizaron DTOs (objetos de transferencia de datos) en el paquete `dto` junto con sus clases de transformación en `dto.mappers`. De esta manera, los controladores intercambian DTOs con el *frontend*, mientras que los servicios y repositorios operan sobre entidades, reduciendo el acoplamiento.

En cuanto al diseño de la API, cada controlador expone operaciones sobre los recursos utilizando los métodos HTTP correspondientes (GET, POST, PUT, DELETE). En Spring Boot se utilizaron las anotaciones `@RestController` y `@RequestMapping` (y sus variantes por método), donde esta última define la ruta base de cada recurso y `@RestController` permite indicar que los métodos del controlador retornan datos, en este caso JSON, que Spring Boot serializa y escribe directamente en el cuerpo de la respuesta HTTP, en lugar de devolver una vista.

En lo que refiere a la comunicación entre el *frontend*, el *backend* y la base de datos, la misma se implementó siguiendo el enfoque cliente-servidor definido en la arquitectura del sistema.

El *frontend* se encarga de interactuar con el usuario y de realizar solicitudes al *backend* a través de la API expuesta. Estas solicitudes se realizan utilizando HTTP, enviando y recibiendo datos en formato JSON. De esta forma, cada acción del usuario en la interfaz (como crear una tarea, actualizar información o consultar métricas) se traduce en una llamada a un *endpoint* específico del *backend*.

Por su parte, el *backend* actúa como intermediario entre el *frontend* y la base de datos. Al recibir una solicitud, procesa la lógica de negocio correspondiente y, en caso de ser necesario, interactúa con la base de datos para obtener o persistir información. La conexión con la base de datos se gestionó mediante configuraciones definidas en variables de entorno, lo que permitió separar la lógica de la aplicación de los datos sensibles, como credenciales o URLs de conexión.

En cuanto a la configuración del sistema, tanto el *frontend* como el *backend* utilizan archivos de entorno (`.env`) para definir parámetros específicos de ejecución. En el caso del *frontend*, se configuró la URL base de la API, lo que permitió que la aplicación pudiera comunicarse correctamente con el *backend* en distintos entornos (desarrollo o producción). En el *backend*, se definieron variables relacionadas con la conexión a la base de datos, como el host, el puerto, el nombre de la base y las credenciales de acceso.

La base de datos, por su parte, se utiliza como el componente encargado de la persistencia de la información. Todas las operaciones de lectura y escritura se realizan a través del *backend*, evitando el acceso directo desde el *frontend* y asegurando así un mayor control sobre la lógica y la seguridad del sistema.

Esta separación de responsabilidades permitió mantener un acoplamiento bajo entre los distintos componentes del sistema, facilitando tanto el desarrollo como el despliegue. Además, el uso de variables de entorno permitió adaptar

fácilmente la configuración del sistema a distintos contextos sin necesidad de modificar el código.

Inicio de sesión

En cuanto a los requisitos para esta liberación, uno de los principales era que los usuarios pertenecientes a Pyxis pudieran acceder de forma segura a la plataforma con su cuenta de la organización. Para ello, fue necesario investigar la mejor manera de implementar el inicio de sesión, ya que este no debía realizarse simplemente con credenciales creadas en el sistema.

Tras un proceso de investigación, se decidió utilizar para el inicio de sesión la librería NextAuth.js, la cual se encarga de gestionar toda la autenticación con proveedores externos.

La decisión de utilizar esta librería surgió a partir de una recomendación de los arquitectos de Pyxis. Inicialmente, el equipo había planteado implementar el inicio de sesión y la autenticación directamente desde el *backend*, integrando el servicio de Microsoft 365. Sin embargo, los arquitectos sugirieron el uso de NextAuth.js no solo por su fácil integración con Next.js, sino también porque la librería ya cuenta con soporte para Microsoft Entra ID (anteriormente, Azure AD). Esto implica que gran parte de la configuración necesaria para trabajar con este proveedor está preconfigurada, requiriendo únicamente parámetros básicos, y evitando así tener que implementar manualmente el flujo completo de autenticación.

Para habilitar la autenticación mediante Microsoft Entra ID, es necesario registrar una aplicación en el portal de Azure. Este proceso implica definir el nombre y la descripción de la aplicación, establecer quiénes podrán acceder, configurar la URL de redirección que será utilizada por NextAuth.js, y especificar los permisos requeridos para el acceso a recursos de Microsoft. Además, se deben generar las credenciales correspondientes, como el Client ID, Tenant ID y Client Secret, que luego serán utilizadas por la aplicación para completar el flujo de autenticación.

Debido a políticas de seguridad y restricciones de permisos dentro de Pyxis, el equipo de desarrollo no contaba con los accesos necesarios para realizar directamente este registro. Por este motivo, el equipo de tecnologías de la información (IT) de la organización fue el encargado de llevar a cabo la creación y configuración de la aplicación.

Por parte del equipo de desarrollo, se le proporcionó la información necesaria para dicha configuración. Una vez creada la aplicación, el equipo de IT envió las credenciales generadas, las cuales fueron integradas en el proyecto mediante variables de entorno, permitiendo completar la configuración del inicio de sesión con Microsoft 365.

La autenticación de usuarios requirió la integración de distintos componentes, incluyendo el *frontend*, Microsoft Entra ID como proveedor de identidad, el *backend* de la aplicación y la base de datos. Debido a la cantidad de actores involucrados y a las responsabilidades que cada uno asume durante el proceso, se elaboró un diagrama para representar las principales interacciones que ocu-

ren durante el inicio de sesión. La figura 6.5 presenta una visión general de este flujo.

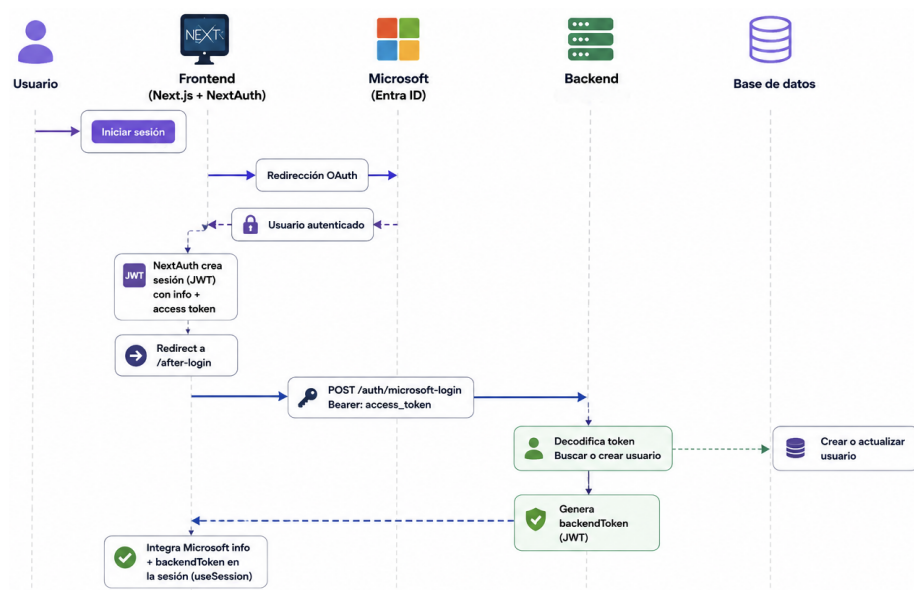


Figura 6.5: Diagrama de interacción del flujo de autenticación.

Nota. Imagen generada mediante ChatGPT (OpenAI, 2026) a partir de un prompt elaborado por los autores.

Cuando el usuario selecciona la opción de iniciar sesión, NextAuth inicia el flujo OAuth y redirige al usuario hacia Microsoft para realizar la autenticación. Una vez validada la identidad del usuario, NextAuth crea una sesión en el *frontend* que almacena la información básica del usuario junto con el *access token* proporcionado por Microsoft.

Posteriormente, el *frontend* utiliza dicho token para comunicarse con el *backend*. A partir de la información contenida en él, el *backend* ejecuta una lógica de “buscar o crear usuario” en la base de datos. En el caso de que el usuario no exista, se crea automáticamente con los datos recibidos y si ya existe, se actualiza la información relevante, asegurando que cada usuario autenticado mediante Microsoft cuente con una representación dentro del sistema. Una vez completado este proceso, el *backend* genera un token propio de la aplicación que es incorporado a la sesión mantenida por NextAuth.

A partir de ese momento, todas las solicitudes realizadas desde el *frontend* hacia la API incluyen el token generado por el *backend* en el encabezado de autorización. Antes de procesar cada petición, el *backend* valida dicho token para garantizar que únicamente usuarios autenticados y autorizados puedan acceder a los recursos protegidos de la aplicación.

La solución implementada permitió integrar el sistema con las cuentas corporativas de Microsoft 365 utilizadas por Pyxis. No obstante, su implementación presentó diversos desafíos técnicos. Si bien NextAuth simplificó considerablemente la integración con Microsoft 365, fue necesario coordinar la autenticación administrada por un proveedor externo con la gestión de usuarios propia de la aplicación, además de diseñar una estrategia de autorización independiente para las comunicaciones con el *backend*. Asimismo, debido a las políticas de seguridad de Pyxis, la configuración inicial de la aplicación en Azure debió realizarse en conjunto con el equipo de IT, lo que agregó un nivel adicional de coordinación durante el proceso de desarrollo.

Traducciones

Durante la implementación, se consultó al *product owner* si resultaba de valor para el producto contar con la página web disponible tanto en inglés como en español. A partir de esto, se le solicitó al equipo realizar una investigación para estimar el esfuerzo necesario para incorporar esta funcionalidad y evaluar si debía implementarse desde el inicio del proyecto o si podía postergarse para etapas posteriores.

La investigación realizada concluyó que Next.js ofrece soporte para múltiples idiomas, incluyendo la gestión del contenido traducido. Con base en estos resultados, se estimó que el esfuerzo requerido para integrar soporte multilingüe desde etapas tempranas del proyecto era bajo, por lo que el *product owner* decidió incluir esta funcionalidad en la presente liberación.

La funcionalidad se implementó a nivel de *frontend* utilizando el soporte de internacionalización provisto por Next.js, lo que permitió que los textos de la aplicación pudieran visualizarse tanto en inglés como en español.

Para lograrlo, se definió un mecanismo basado en archivos de diccionario, donde cada idioma cuenta con su propio archivo de configuración (en.json para inglés y es.json para español). En estos archivos se organizan todas las cadenas de texto utilizadas en la aplicación, agrupadas por secciones funcionales como errores, navegación, tareas, proyectos y métricas.

Cada texto se identifica mediante una clave única, lo que permite que las modificaciones se gestionen exclusivamente a través de los archivos de traducción, evitando la necesidad de alterar la lógica y facilitando la incorporación de nuevos idiomas en futuras iteraciones.

La aplicación ofrece soporte para los idiomas español e inglés, determinando el idioma a partir de la configuración del navegador del usuario. En caso de no poder determinarlo, se utiliza español como idioma por defecto. Para reflejar esto en la navegación, todas las rutas incluyen un prefijo de idioma (/es/ o /en/) luego de la URL base.

Pantalla de Dashboard y sus métricas

Una parte fundamental del proyecto fue la definición y el cálculo de métricas, debido a que, a partir de ellas se toman decisiones importantes sobre los

proyectos. Por este motivo, fue necesario asegurar que los cálculos y los datos mostrados fueran consistentes, correctos y confiables.

A partir de esto, se puede afirmar que la implementación de los diferentes cálculos era fundamental para esta primera liberación, ya que no solo representaban una gran parte de las funcionalidades de esta etapa, sino que además constituían la base para la segunda liberación.

El proceso de implementación de los cálculos comenzó al inicio de la liberación, donde el equipo de desarrollo se tomó inicialmente varias semanas para estudiar las diferentes métricas. En ese momento, el equipo comenzó a elaborar un documento describiendo cómo se implementaría cada una de esas métricas, el cual se iba validando con el *product owner*. Este documento estuvo enfocado en las métricas a incluir en la primera liberación y, posteriormente, fue evolucionando para incorporar los cálculos correspondientes a la versión final del sistema. El mismo se puede encontrar en el anexo B.

Cabe destacar que, si bien la primera liberación estaba orientada a un único usuario y un único proyecto, los cálculos fueron definidos desde un inicio de manera tal que pudieran ser reutilizados al extender el sistema a múltiples usuarios y múltiples proyectos.

En base a este documento, se llevó a cabo la implementación de las métricas correspondientes a la primera liberación. El mismo funcionó como guía para definir la lógica de cálculo de cada indicador, permitiendo trasladar de forma consistente lo especificado a su implementación en el sistema.

En la figura 6.6 se puede observar la pantalla de dashboard implementada al finalizar la primera liberación. En la misma se visualizan dos tablas de métricas, junto con una gráfica que presenta datos sobre el valor ganado planificado y real. Asimismo, en esta página se muestra la fecha estimada de finalización del proyecto.

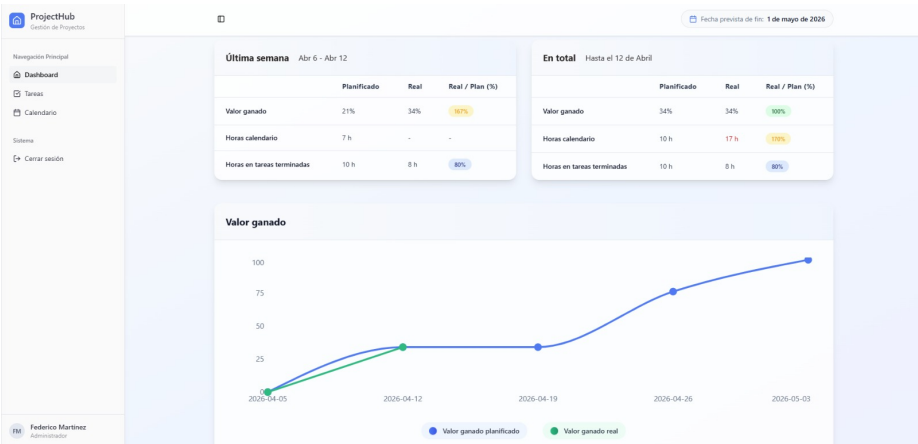


Figura 6.6: Pantalla de Dashboard implementada en la primera liberación.

Las tablas presentes en la página corresponden, por un lado, a las métricas de la última semana y, por otro, a las métricas acumuladas desde el inicio del proyecto hasta el domingo de la última semana. La decisión de incluir ambas tablas surge de la necesidad de Pyxis de no solo disponer de datos totales del proyecto, sino también de contar con información que permita evaluar las estimaciones más recientes y facilitar la planificación a partir de la semana actual.

En ambas tablas se presentan las mismas métricas, considerando diferentes rangos temporales en cada caso. A continuación, se detallan dichas métricas y sus respectivos cálculos para la tabla de métricas totales del proyecto. Las métricas correspondientes a la última semana se calculan de manera análoga, con ciertas excepciones que se describirán más adelante, teniendo en cuenta un rango de tiempo distinto.

El Valor Ganado Planificado se calcula en base al plan del proyecto, considerando las semanas del calendario y las tareas definidas, y representa el valor que se esperaba obtener desde el inicio del proyecto hasta el último domingo inclusive. Este valor se obtiene a partir de las horas estimadas de las tareas cuya finalización estaba prevista antes de dicha fecha.

Por su parte, el Valor Ganado real refleja el valor efectivamente obtenido en ese mismo período, considerando las horas estimadas de las tareas que fueron finalizadas dentro de dicho rango temporal. A partir de estos dos indicadores, se calcula el porcentaje de Valor Ganado respecto al Valor Planificado, el cual permite medir qué porcentaje de lo planificado fue realmente alcanzado.

En relación con la dedicación de tiempo, las horas calendario planificadas totales corresponden a la cantidad de horas que se estimó destinar al proyecto según el calendario definido. En contraste, las horas totales realmente dedicadas representan el tiempo efectivamente invertido en las tareas del proyecto. La relación entre ambas se expresa mediante el porcentaje de horas reales respecto a las planificadas, indicador que permite evaluar qué parte del tiempo planificado fue utilizado en la práctica.

Finalmente, se consideran métricas específicas relacionadas con las tareas ya finalizadas en el proyecto. Las horas estimadas de tareas finalizadas corresponden a la suma de las estimaciones de esfuerzo de las tareas completadas, mientras que las horas reales de tareas finalizadas reflejan el tiempo que efectivamente se requirió para finalizarlas. A partir de estos valores, se calcula el porcentaje de horas reales respecto a las estimadas para tareas finalizadas, lo cual permite evaluar la precisión de las estimaciones realizadas.

En relación con la tabla de la última semana, resulta pertinente realizar algunas aclaraciones respecto a decisiones tomadas en esta liberación. Como se mencionó previamente, no era requisito contar con un temporizador que registrara el tiempo dedicado a cada tarea, sino que las horas se ingresaban manualmente. Esta decisión implicó que no fuera posible determinar con exactitud la fecha y el horario en que se realizaban dichas horas.

Como consecuencia, puede observarse que la tabla de la última semana presenta dos campos sin datos en la fila correspondiente a “Horas realizadas”. Al no poder determinar cuántas de las horas ingresadas para una tarea fueron efectivamente realizadas en la última semana, no resultaba posible calcular las

horas reales para dicho período. Sin embargo, sí era posible calcular las horas planificadas, dado que estas se basaban en el calendario del usuario.

Otra consideración relevante es que el Valor Ganado Planificado no refleja completamente la realidad. Esto se debe a que, por la misma limitación mencionada anteriormente, no era posible determinar si al inicio de la última semana una tarea ya contaba con horas dedicadas, dado que únicamente se almacenaba el esfuerzo total ingresado manualmente. En consecuencia, en esta primera liberación para el cálculo de esta métrica se asumía que todas las tareas no finalizadas no tenían horas dedicadas.

En cuanto a la gráfica, en la misma se visualizan dos líneas: una correspondiente al valor ganado planificado y otra al valor ganado real. La línea de Valor Planificado representa, de acuerdo con el plan, cuánto valor debería haberse obtenido semana a semana, considerando las estimaciones de horas y la dedicación planificada para cada período. Por su parte, la línea de Valor Ganado refleja el valor efectivamente obtenido semana a semana, desde el inicio del proyecto hasta el último domingo, teniendo en cuenta las tareas finalizadas en cada intervalo.

La implementación del gráfico en el *frontend* se realizó utilizando la librería *Recharts*. El *backend* le envía al *frontend* dos listas de puntos (planificado y real), cada una compuesta por la fecha de la semana y el valor numérico correspondiente. El *backend* expone estos datos mediante un *endpoint*.

Por último, tal como se mencionó anteriormente, en esta liberación se tenía una primera versión de una fecha prevista de fin. La misma consideraba, para el usuario, las tareas que aún tenía pendientes. A partir del tiempo total estimado de dichas tareas, se recorrían las semanas del calendario comenzando desde la semana actual. En cada iteración, se descontaba del total de horas pendientes el esfuerzo planificado para la semana correspondiente.

Si el esfuerzo planificado en el calendario no alcanzaba para cubrir la totalidad de las horas estimadas de las tareas pendientes, se informaba al usuario que no había disponibilidad suficiente. En caso contrario, y asumiendo una jornada laboral de lunes a viernes, se determinaba como fecha prevista de finalización el viernes de la semana en la que se completaban todas las horas necesarias.

Todas las implementaciones de los cálculos mencionados anteriormente pueden consultarse en el anexo [B](#).

Las métricas mencionadas dependen directamente tanto de las horas estimadas de las tareas como de las horas planificadas en cada semana del calendario. Por este motivo, se consideró necesario advertir al usuario cuando, al modificar alguna de estas horas, se generaba una inconsistencia entre ambos valores. Por ejemplo, cuando las horas en semanas aún no transcurridas en el calendario del usuario fueran insuficientes para cubrir el total de horas estimadas de las tareas pendientes.

En función de esto, se decidió incorporar un mensaje de alerta visible tanto en la página de tareas como en la de calendario. Dicho mensaje se muestra cuando corresponde, para informar al usuario sobre la insuficiencia de horas en el calendario. La decisión de mostrar o no esta alerta se determina dinámicamente a partir de la información provista por el *backend* cada vez que se realizan modificaciones sobre las tareas o las semanas del calendario.

6.6.2. Segunda liberación

El objetivo principal de la segunda liberación fue adaptar el sistema para permitir la gestión de múltiples proyectos con equipos compuestos por múltiples usuarios, continuando con el soporte a la planificación y seguimiento de los mismos, y permitiendo a cada usuario realizar un seguimiento individual de su participación en el proyecto.

Como se mencionó en la sección 6.5, una vez finalizada la primera liberación, la aplicación comenzó a ser utilizada por usuarios dentro de Pyxis. Esto representó un desafío para el equipo, ya que únicamente se contaba con entornos de desarrollo y producción. Por este motivo, mientras los usuarios de la organización realizaban pruebas, no era posible desplegar nuevos cambios, sino únicamente correcciones correspondientes a la primera liberación. A su vez, tal como se mencionó previamente, el equipo utilizó la herramienta durante el final de la primera etapa para gestionar las distintas tareas, por lo que cualquier cambio podría incluso afectar los datos del propio equipo.

Como se mostró en el capítulo de arquitectura, para esta liberación se planificaron diversos cambios a nivel de base de datos. Estos cambios interferían con los datos existentes, por lo que fue necesario considerar la implementación de migraciones.

En total, el proyecto cuenta con doce migraciones, las cuales se encuentran ubicadas en el directorio *resources/db/migration*. En este directorio se definieron *scripts* versionados. La implementación de migraciones permitió evolucionar el esquema de la base de datos de forma controlada a lo largo de la segunda liberación.

Roles definidos

Una de las iteraciones más importantes de esta liberación fue adaptar la aplicación para soportar múltiples usuarios perteneciendo a múltiples proyectos. Uno de los primeros desafíos a abordar fue cómo implementar el manejo de roles y permisos a nivel de usuarios dentro del sistema.

El equipo de desarrollo planteó dos posibles enfoques para resolver este problema. El primero se centraba en la definición de roles, donde cada rol tenía predefinido qué tipo de acciones podía realizar en la plataforma y a qué funcionalidades tenía acceso. El segundo enfoque se basaba en la definición de permisos, donde estos estaban asociados a funcionalidades específicas y cada usuario podía contar con un conjunto de permisos que le permitieran operar dentro de la plataforma.

Ambos enfoques fueron documentados y discutidos con el *product owner*, con el objetivo de determinar cuál se ajustaba mejor a las necesidades de la empresa en el corto y mediano plazo. Si bien el enfoque basado en permisos resultaba más escalable a futuro, ya que permitiría agregar nuevos permisos y asignarlos dinámicamente a los usuarios, luego de varias discusiones se decidió optar por el enfoque basado en roles. Esta decisión se tomó debido a que el *product owner* quería únicamente tres roles bien definidos, cuyos permisos ya

estaban claramente establecidos.

En conjunto con el *product owner*, se definieron los siguientes roles a nivel de sistema: Integrante de equipo, Operador y Administrador. El rol de administrador tiene como principal función la gestión de usuarios, siendo el único que puede asignar o modificar los roles de otros usuarios dentro de la plataforma. Por su parte, el operador tiene como principal responsabilidad la gestión de proyectos, enfocándose principalmente en la creación de nuevos proyectos. Finalmente, el rol de integrante de equipo es asignado a todos los usuarios por defecto, y cuyas funcionalidades son las básicas, sin permisos para asignar roles ni crear nuevos proyectos.

Durante la definición de los roles del sistema, surgió la necesidad de considerar también roles a nivel de cada proyecto individual. En particular, se identificó el caso en el que un operador podría crear un proyecto sin necesariamente formar parte del mismo, lo que implicaría la ausencia de un usuario que gestione el proyecto. A partir de esta situación, se definieron los roles a nivel de proyecto: líder e integrante del proyecto. El líder tiene como principal responsabilidad la gestión del mismo y de las tareas asociadas. Al momento de crear un proyecto, un administrador u operador debe definir quiénes serán los líderes (pudiendo asignarse a sí mismo). Por otro lado, el integrante del proyecto cuenta con funcionalidades básicas dentro del proyecto al que pertenece.

Una vez definidos los roles tanto a nivel de sistema como de proyecto, el equipo procedió con su implementación. Para ello, se decidió almacenar el rol a nivel de sistema en la entidad *User*, mientras que el rol a nivel de proyecto se modeló en la entidad *ProjectUser*.

Tal como se mencionó en la sección de diseño arquitectónico, se utilizaron anotaciones de Spring Boot en los controladores, por sugerencia de los arquitectos de Pyxis, para controlar que ciertos *endpoints* solo puedan ser accesibles para usuarios con cierto rol a nivel de sistema o de proyecto. Por otro lado, también se hicieron verificaciones adicionales a nivel de servicio, para que ciertas acciones estén restringidas a ciertos roles.

Por otro lado, a nivel de *frontend*, las funcionalidades disponibles se adaptan dinámicamente según el rol del usuario. Aquellas acciones restringidas se ocultan para los usuarios que no cuentan con los permisos necesarios. Por ejemplo, el botón para crear un nuevo proyecto solo es visible para administradores y operadores a nivel de sistema, mientras que funcionalidades como la asignación de tareas dentro de un proyecto solo están disponibles para los líderes del mismo.

A continuación, se detallan las funcionalidades y pantallas disponibles para cada uno de los roles definidos, tanto a nivel de sistema como de proyecto.

El administrador del sistema tiene como principal responsabilidad la gestión de usuarios dentro de la plataforma. En la figura 6.7 se muestra la pantalla del administrador, desde la misma se puede asignar o desasignar los roles de administrador u operador a usuarios existentes, siendo precondition que estos ya estén registrados en el sistema.

Además, el administrador cuenta con los mismos permisos que un operador, por lo tanto puede acceder también a las pantallas de administración de proyectos, incluyendo la creación y administración de los mismos y sus integrantes.

La pantalla en donde se listan todos los proyectos presentes en la plataforma se puede observar en la figura 6.8. En términos generales, el administrador tiene acceso completo tanto a la gestión de usuarios como a la gestión de proyectos dentro de la aplicación.

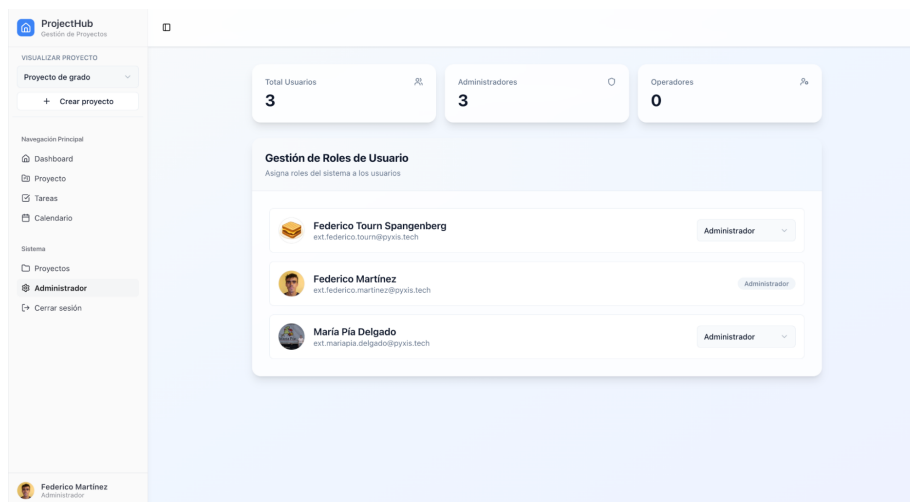


Figura 6.7: Pantalla de Administrador.

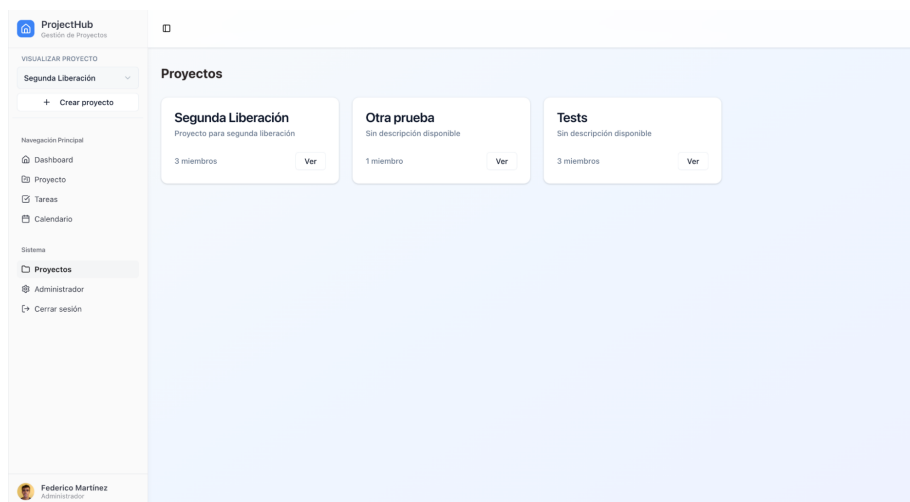


Figura 6.8: Pantalla de Proyectos.

El operador tiene como principal función la administración de proyectos.

Estos usuarios pueden crear nuevos proyectos y asignarle integrantes, definiendo para cada uno sus horas semanales y si serán líderes del proyecto. La figura 6.9 muestra el formulario dentro de la plataforma para la creación de nuevos proyectos.

El operador también puede editar proyectos existentes y gestionar a los equipos de un proyecto, asegurándose que cada uno cuente con al menos un líder. En resumen, el operador tiene control total sobre la configuración y mantenimiento de los proyectos.

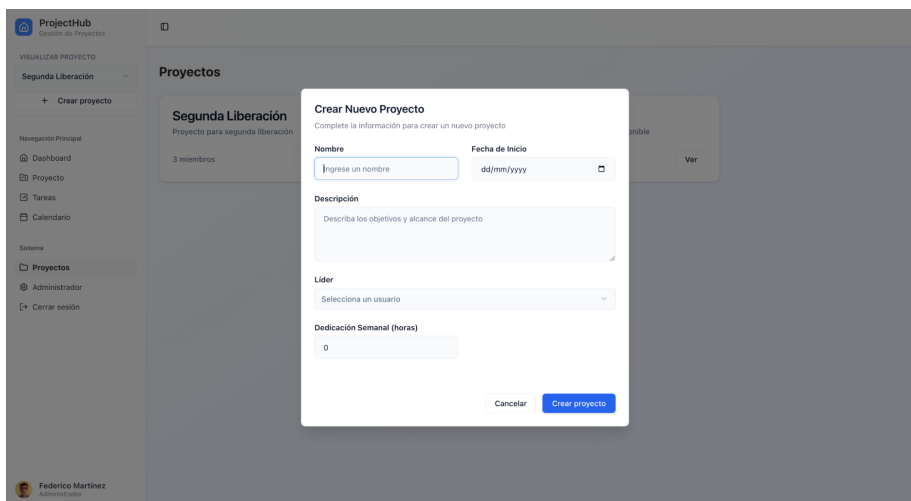
The image shows a web application interface for ProjectHub. On the left is a sidebar with navigation links: Dashboard, Proyecto, Tareas, Calendario, and a 'Proyectos' section containing 'Administrador' and 'Cerrar sesión'. The main area displays a 'Segunda Liberación' project card with 3 members. Overlaid on this is a 'Crear Nuevo Proyecto' modal form. The form has fields for 'Nombre' (with a placeholder 'Ingrese un nombre'), 'Fecha de inicio' (with a date picker showing 'dd/mm/yyyy'), a 'Descripción' text area (placeholder: 'Describe los objetivos y alcance del proyecto'), a 'Líder' dropdown menu (placeholder: 'Selecciona un usuario'), and a 'Dedicación Semanal (horas)' input field (value: '0'). At the bottom of the modal are 'Cancelar' and 'Crear proyecto' buttons.

Figura 6.9: Formulario para crear un nuevo proyecto.

El líder del proyecto es un integrante del mismo con permisos adicionales de gestión dentro de este. Desde la pantalla del proyecto seleccionado, puede editar la información del proyecto (figura 6.10) y gestionar sus integrantes (figura 6.11). Desde la pantalla del calendario puede administrar los calendarios de todos los integrantes del proyecto. Además, puede acceder a la pantalla de gestión de tareas, donde puede crear nuevas tareas y editar cualquier tarea del proyecto, independientemente de a quién esté asignada. También puede visualizar y gestionar los registros de esfuerzo asociados a las tareas, con las restricciones correspondientes cuando estos afectan a otros integrantes. Al ser también un integrante, el líder conserva todas las funcionalidades propias de ese rol dentro del proyecto.

El integrante de proyecto tiene acceso a las funcionalidades dentro del proyecto. Desde las pantallas principales, puede visualizar métricas generales del proyecto y métricas personales. En las pantallas de tareas, puede crear nuevas tareas asignadas a sí mismo y editar únicamente aquellas en las que está asignado, pudiendo modificar nombre, descripción, horas planificadas, registrar horas trabajadas (mediante temporizador o carga manual) y gestionar el estado de

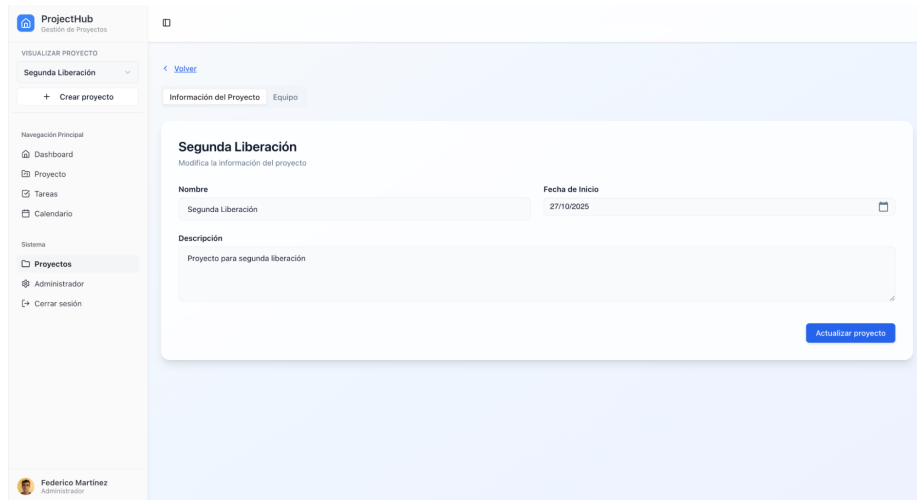


Figura 6.10: Pantalla de Información del Proyecto.

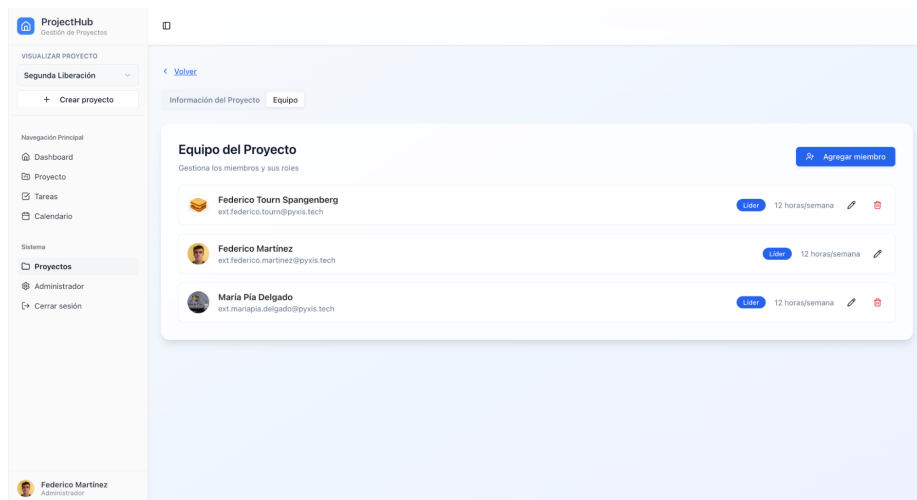


Figura 6.11: Pantalla de Equipo del Proyecto.

finalización. También puede visualizar todas las tareas del proyecto y sus registros de esfuerzo, aunque sin posibilidad de edición sobre tareas ajenas. Además, desde su calendario personal, puede ver y ajustar su dedicación semanal al proyecto, sin acceso a los calendarios de otros integrantes.

Por último el integrante de equipo es el rol por defecto cuando un usuario se crea en la plataforma. El mismo no tiene ningún permiso y no puede ver ningún dato dentro de la aplicación hasta que esté asignado a un proyecto.

Por otro lado, como se mencionó anteriormente, distintas pantallas implementadas en la primera liberación debieron adaptarse para distinguirse entre una vista individual y una vista general del proyecto. Esta decisión surgió de la necesidad de permitir a los integrantes visualizar sus propias métricas y gestionar sus tareas de forma independiente. Las pantallas a las que se les incorporó esta doble vista fueron la del Dashboard y la de Tareas, las cuales se muestran en la figura 6.12 y 6.13 respectivamente.

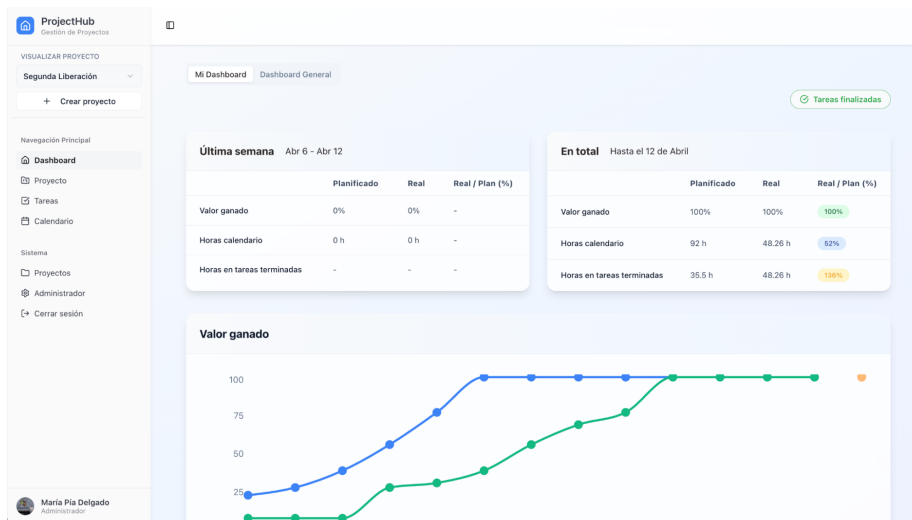


Figura 6.12: Pantalla de Dashboard implementada en la segunda liberación.

Sesión de NextAuth

Para lograr un correcto manejo de los roles según el usuario autenticado y el proyecto seleccionado, era indispensable disponer de esta información a nivel de *frontend*, de modo de poder mostrar las pantallas y funcionalidades correspondientes. Luego de una breve investigación para determinar la mejor forma de gestionar esto, se decidió incorporar los roles dentro de la sesión de NextAuth.

Al finalizar la primera liberación, la sesión ya almacenaba, además de los datos del usuario obtenidos desde Microsoft 365, el token del *backend*. En esta

Tarea	Persona asignada	Horas planeadas	Horas realizadas	Estado	Fecha de fin	Acciones
Arreglo de errores con codigos	María Pía Delgado	3	3	Completado	30/10/2025	
[PROYECTO] Página de nuevo proyecto	María Pía Delgado	5	12	Completado	17/11/2025	
[PROYECTO] Edición de nombre del proyecto y fecha de inicio del proyecto	María Pía Delgado	1	1	Completado	17/11/2025	
[BUG][TAREAS] Cuando se toca el timer no se pasa la tarea al estado en progreso	María Pía Delgado	0.5	0.3	Completado	20/11/2025	
[DISEÑOS INES] tabs pagina dashboard	María Pía Delgado	0.5	0.5	Completado	23/11/2025	
[DISEÑOS INES] Visualizar proyectos	María Pía Delgado	1	2	Completado	27/11/2025	
[PROYECTO] Crear tab del equipo en la pagina de editar proyecto	María Pía Delgado	3	3.5	Completado	03/12/2025	
[SCHEDULE] A confirm: aplicar diseños a vista general	María Pía Delgado	2	1.5	Completado	09/12/2025	
[SCHEDULE] Agregar tabs a la pagina del schedule	María Pía Delgado	2	2.5	Completado	09/12/2025	
[ADMIN] Agregar diseños de ines a la pagina del admin	María Pía Delgado	2	2	Completado	09/12/2025	
[PROYECTOS] A definir: Agregar funcionalidad de editar proyecto	Se debería de poder editar los campos y usuarios del proyecto A definir: como mostrar esto si					

Figura 6.13: Pantalla de Tareas implementada en la segunda liberación.

última liberación, se agregó también el identificador del proyecto seleccionado, siguiendo una sugerencia del equipo de arquitectura. Como se mencionó en la sección de arquitectura, el objetivo de esta decisión fue lograr consistencia en el uso de múltiples navegadores simultáneamente.

También se incorporaron a la sesión de NextAuth los roles del usuario autenticado, tanto a nivel de sistema como de proyecto. Esto permitió evitar consultas constantes al *backend* para determinar qué funcionalidades debían estar disponibles para cada usuario en cada pantalla. Cada vez que un usuario modifica el proyecto seleccionado, el rol a nivel de proyecto se actualiza en la sesión, almacenando el valor correspondiente al nuevo contexto.

Adicionalmente, para cubrir el caso en el cual el rol de un usuario ya autenticado sea modificado por otro usuario (ya sea a nivel de sistema o de proyecto), los roles almacenados en la sesión de NextAuth se sincronizan periódicamente con el *backend*, actualizándose cada cinco minutos.

Horas del calendario sin asignar

Uno de los principales requisitos de esta liberación era soportar múltiples usuarios dentro de un mismo proyecto. A partir de esta necesidad, el *product owner* planteó que es común que nuevos integrantes se incorporen a proyectos ya iniciados. En este contexto, se propuso la incorporación de una bolsa de horas correspondiente a semanas futuras, aún sin asignación a usuarios específicos. Es decir, si ya se conoce que va a haber en el futuro asignaciones de personas al proyecto, poder simular esta situación utilizando una bolsa de horas disponibles para el proyecto.

En respuesta a este requerimiento, se incorporó una columna adicional en la página de calendario, no asociada a ningún usuario en particular. Esta columna es visible únicamente para los líderes del proyecto y refleja la cantidad de horas que potenciales futuros integrantes podrían dedicar en cada semana. La misma se presenta bajo el título “Horas disponibles sin asignar”.

Estas horas sin asignar son consideradas en los cálculos del sistema para aquellas tareas que no tienen un usuario asignado. En estos casos, se asume que dichas tareas serán realizadas por un integrante futuro del proyecto, cuyas horas disponibles se reflejan en esta nueva columna.

Registros de esfuerzo

Otra funcionalidad abarcada en esta segunda liberación fue la posibilidad de registrar el tiempo efectivamente dedicado a las tareas mediante registros de esfuerzo. Este requisito surge principalmente de la necesidad de conocer en qué momentos se trabajó sobre cada tarea, permitiendo así la posterior obtención de métricas basadas en dicha información.

En conjunto con esta funcionalidad, se añadió un temporizador que puede iniciarse al comenzar a trabajar en una tarea y pausarse cuando se interrumpe su ejecución, facilitando el registro preciso del tiempo dedicado. El temporizador se puede visualizar en la figura 6.14.

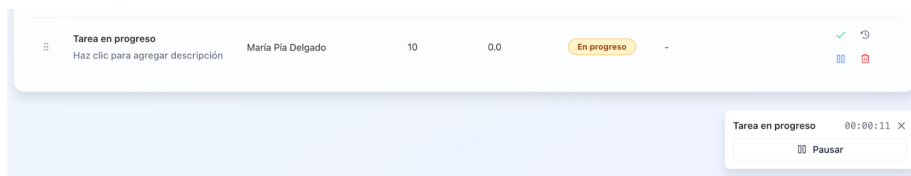


Figura 6.14: Temporizador en progreso.

Tal como se mencionó en la sección de arquitectura, los registros de esfuerzo fueron representados a nivel de base de datos mediante una nueva entidad denominada *UserTaskEffort*. Con el objetivo de no perder la información previamente disponible sobre el esfuerzo dedicado a cada tarea, se implementó una migración que toma dicho valor y genera, a partir de él, una instancia correspondiente de esta nueva entidad.

En dicha migración, las fechas de los registros de esfuerzo se definieron a partir de la información ya existente en las tareas. En particular, se tomó la fecha de creación de cada tarea como instante de inicio del esfuerzo (*start_time*), mientras que la fecha de finalización (*end_time*) se calculó sumando a dicho instante la duración total registrada en la tarea (*real_time*). De esta forma, cada esfuerzo migrado representa un bloque continuo de trabajo cuyo inicio coincide con la creación de la tarea y cuya duración refleja el tiempo real previamente almacenado.

Contar con estos registros permitió mejorar el cálculo de diversas métricas en la página de Dashboard. En particular, el tiempo calendario total efectivamente dedicado pasó a calcularse hasta el último domingo, en lugar de la fecha actual, alineándose así con el mismo rango temporal utilizado por las demás métricas de la tabla. Asimismo, estos registros hicieron posible disponer de información precisa sobre el tiempo realmente dedicado al proyecto durante la semana anterior a la actual.

En relación con el uso del temporizador y los registros de esfuerzo, un integrante del proyecto solo puede registrar esfuerzo en tareas asignadas a sí mismo. Sin embargo, el líder del proyecto tiene la capacidad de registrar esfuerzos para todos los integrantes, aunque no puede iniciar el temporizador en su nombre. Antes de crear un registro de esfuerzo, el líder es advertido de que está registrando horas a otro usuario, dado que esto impacta tanto en las métricas personales del otro usuario como en las del proyecto.

A continuación, se detallan algunas consideraciones relevantes sobre el manejo de los registros de esfuerzo. El sistema permite iniciar y pausar el temporizador manualmente tantas veces como sea necesario. En caso de que una tarea se complete con el temporizador en ejecución, este se detiene automáticamente. Asimismo, si se intenta cambiar de proyecto mientras el temporizador está en ejecución, el sistema muestra un mensaje de alerta y, en caso de confirmarse el cambio, el temporizador se pausa.

Al registrar esfuerzo, se asume que el integrante asignado a la tarea es quien realizó dichas horas. Cuando una tarea se completa (y se marca en la aplicación), el sistema asigna como fecha de fin la correspondiente al último esfuerzo registrado. En caso de que se desee registrar esfuerzo en una tarea ya completada, es necesario marcarla previamente como no completada.

Por otro lado, no es posible registrar esfuerzo en fechas futuras, y la fecha de inicio de cada registro debe ser anterior a su fecha de fin. Tampoco se permite completar tareas para proyectos que aún no han comenzado. Además, la fecha de fin de una tarea es editable, siempre que sea posterior al último esfuerzo registrado y anterior a la fecha actual.

A modo de ejemplo, en la figura 6.15 se muestra una tarea (Tarea de prueba) en progreso. En ella pueden observarse los distintos campos de la tarea, así como el historial de registros de esfuerzo, donde se detalla para cada registro su fecha de inicio, fecha de fin y duración.

Líneas de Replanificación y Pronóstico

Cuando se definió el alcance de ambas liberaciones al inicio del proyecto, se planteó la posibilidad de incorporar, en esta segunda liberación, múltiples gráficas basadas en distintas métricas, además del valor ganado. Sin embargo, al revisar los requisitos antes de comenzar esta etapa, se acordó con el *product owner* no agregar nuevas gráficas, sino incorporar líneas de proyección a futuro sobre la gráfica existente.

A partir de esta decisión, se investigaron las alternativas de cálculo de fechas previstas de fin disponibles en Process Dashboard. Entre las opciones analizadas,

The screenshot shows a web application interface for task management. At the top, there are tabs for 'Mis Tareas' and 'Tareas Generales', with 'Tareas Generales' selected. Below this, there are filters for 'Todas', 'Completadas', and 'Sin completar'. The main content area displays a table with the following columns: 'Tarea', 'Persona asignada', 'Horas planeadas', 'Horas realizadas', 'Estado', 'Fecha de fin', and 'Acciones'. A task titled 'Tarea de prueba' is shown, assigned to 'María Pía Delgado', with 3 planned hours and 2.0 hours completed. The status is 'En progreso'. Below the task row, there is a section for 'Registros de tiempo' (Time Records) with columns for 'Usuario', 'Fecha de inicio', 'Fecha de fin', and 'Duración'. A record is shown for 'María Pía Delgado' starting on '17/04/2026 18:32' and ending on '17/04/2026 20:33', with a duration of '02:00:01'. A button '+ Agregar registro de tiempo' is at the bottom right.

Tarea	Persona asignada	Horas planeadas	Horas realizadas	Estado	Fecha de fin	Acciones
Tarea de prueba Haz clic para agregar descripción	María Pía Delgado	3	2.0	En progreso	-	✓ ⚙
Registros de tiempo						
Usuario		Fecha de inicio		Fecha de fin		Duración
María Pía Delgado		17/04/2026 18:32		17/04/2026 20:33		02:00:01

+ Agregar registro de tiempo

Figura 6.15: Tarea en progreso con registro de tiempo.

se definió implementar dos enfoques: uno basado en un cálculo de replanificación y otro basado en un cálculo de pronóstico. En base a esta investigación, se incorporaron las descripciones de estos cálculos, junto con sus respectivas representaciones en la gráfica, al documento de cálculos que ya contenía las métricas definidas en la primera liberación.

Tras diversos intercambios con el *product owner*, orientados a validar la correcta comprensión de estos nuevos cálculos, se procedió a implementar las métricas definidas para esta segunda liberación. En particular, se implementaron dos fechas previstas de fin y sus líneas de proyección en la gráfica, cada una correspondiente a uno de los enfoques analizados: replanificación y pronóstico.

Se considera pertinente mencionar que, durante el proceso de implementación de estas líneas, surgieron algunas dudas que fueron abordadas por el equipo contactando directamente a David Tuma, principal creador de Process Dashboard. En base a estos intercambios, varias decisiones de implementación que se mencionan a continuación fueron sugeridas por el mismo.

Para el cálculo de la fecha de finalización según replanificación, se asume que las tareas pendientes o en progreso requerirán exactamente el tiempo actualmente estimado. Asimismo, se considera que en las semanas en curso y futuras del calendario se dedicará al proyecto el tiempo planificado. A partir de estas suposiciones, la fecha prevista de finalización se obtiene mediante la asignación y consumo de las horas planificadas en el calendario, recorriendo las tareas pendientes o en progreso hasta completar su esfuerzo estimado.

A nivel de proyecto, la fecha de finalización según replanificación se determina como la más tardía entre las fechas previstas de finalización de todos los usuarios involucrados.

Para el cálculo de la fecha según pronóstico, se toman como referencia las tareas ya finalizadas y se analiza la diferencia entre el tiempo estimado y el tiempo realmente dedicado a las mismas. A partir de esta información, se proyecta cuánto demandarán en la práctica las tareas que se encuentran en progreso o aún pendientes.

Para dichas tareas, se determina si el esfuerzo realmente dedicado ha supera-

do o no el esfuerzo planificado. Aquellas cuyo esfuerzo real excede al planificado se clasifican como tareas con sobreconsumo (*overspent*), mientras que las restantes se clasifican como tareas con subconsumo (*underspent*).

En el caso de las tareas en progreso que presentan sobreconsumo, el cálculo del esfuerzo restante puede resultar inconsistente si se basa únicamente en la diferencia entre el tiempo estimado y el tiempo real, ya que dicha diferencia puede ser negativa. Para resolver este problema, se introduce un mecanismo de redistribución de esfuerzo entre tareas, mediante el cual las tareas con sobreconsumo reciben tiempo estimado adicional proveniente de aquellas que no han agotado su estimación.

En particular, para cada tarea *overspent* se asume que el tiempo ya invertido representa aproximadamente un 90 % de su esfuerzo total. En el contexto del proceso de redistribución, el objetivo consiste en ajustar el tiempo estimado de la tarea hasta alcanzar este nuevo valor total proyectado.

En consecuencia, la cantidad de tiempo que cada tarea *overspent* debe recibir mediante el mecanismo de “préstamo” se calcula como la diferencia entre el esfuerzo total proyectado y su tiempo estimado actual. De esta forma, se incrementa su estimación hasta reflejar un valor más realista, asegurando que la tarea conserve un esfuerzo restante positivo.

Por otro lado, para las semanas del calendario que ya transcurrieron, se observa la diferencia entre el tiempo planificado para las mismas y el tiempo total realmente dedicado. En base a esta tasa de diferencia, se ajusta el tiempo planificado para la semana actual y semanas futuras.

Luego de ajustar tanto las tareas como las semanas del calendario, se procede a consumir para cada usuario las horas ajustadas de su calendario a través de las horas ajustadas de sus tareas. Al igual que en la otra fecha prevista de fin, a nivel de proyecto la fecha de finalización según pronóstico se determina como la más tardía entre las fechas previstas de finalización de todos los usuarios involucrados.

Cabe aclarar que ambas líneas de proyección a futuro, asociadas a las fechas previstas de fin mencionadas, se implementan siguiendo la lógica previamente mencionada para el cálculo de dichas fechas.

A modo de ejemplo se presenta la figura 6.16, en donde se muestra la pantalla de Dashboard con las nuevas líneas de proyección y fechas de fin del proyecto.

6.7. Comparación con Process Dashboard

A continuación se presentan algunas diferencias relevantes entre Process Dashboard y la herramienta desarrollada. Estas diferencias se encuentran asociadas tanto a las decisiones tecnológicas adoptadas como al alcance funcional definido para el proyecto, el cual estuvo guiado por las necesidades identificadas junto a Pyxis y al objetivo de desarrollar una solución adaptada a dicho contexto.

La aplicación desarrollada fue diseñada como una aplicación web con un fuerte foco en la experiencia de usuario, priorizando la usabilidad y la interacción

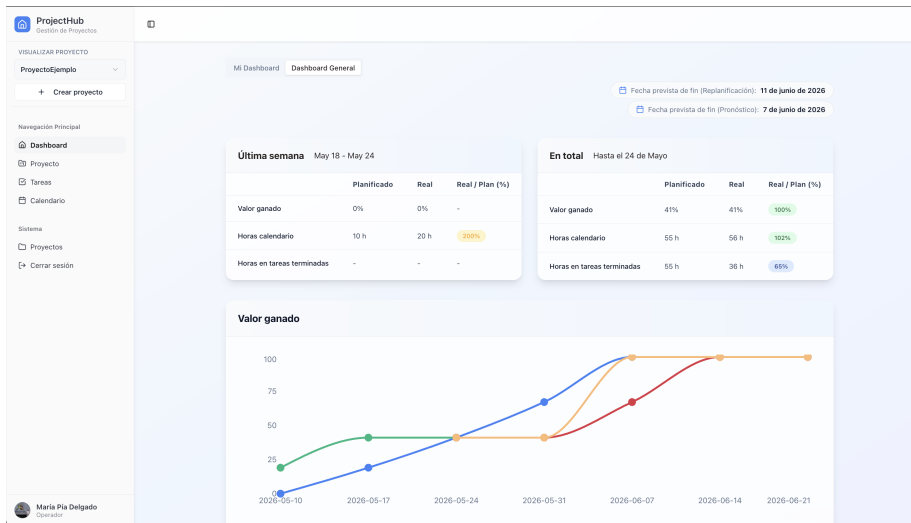


Figura 6.16: Pantalla de Dashboard con las nuevas líneas de proyección y fechas de fin del proyecto.

del usuario. Este aspecto fue trabajado y validado junto a la especialista en UX/UI, lo que permitió orientar las decisiones de diseño hacia una interfaz más clara y consistente. Esta diferencia resulta relevante en el contexto del proyecto, ya que uno de los objetivos fue adaptar la herramienta al uso específico definido junto a Pyxis, asegurando su alineación con las necesidades del entorno de trabajo.

En cuanto al alcance funcional, Process Dashboard ofrece una amplia variedad de funcionalidades orientadas a distintos tipos de proyectos y usuarios. La aplicación desarrollada, en cambio, fue concebida como una solución más acotada, enfocada específicamente en los procesos de planificación, seguimiento, control y replanificación de proyectos mediante la técnica de Valor Ganado. Esta decisión permitió concentrar el esfuerzo de desarrollo en un conjunto de funcionalidades alineadas con las necesidades relevadas junto a la organización, evitando la incorporación de características que no resultaban relevantes para el contexto del proyecto.

Asimismo, mientras Process Dashboard incorpora funcionalidades avanzadas como la gestión de líneas base (*baseline*), estructuras de desglose del trabajo (WBS) y otras capacidades de análisis, estas características no fueron incluidas en la herramienta desarrollada debido a restricciones de alcance. Sin embargo, se reconoce que este tipo de funcionalidades aportan valor en escenarios más complejos de gestión de proyectos, por lo que su incorporación constituye una posible línea de trabajo a futuro para la evolución de la herramienta.

Finalmente, una diferencia relevante se encuentra en el modelo de uso y escalabilidad. Process Dashboard presenta distintas versiones orientadas al uso

individual y organizacional, incorporando en sus versiones más completas funcionalidades de gestión centralizada de usuarios, proyectos y almacenamiento en servidor. La herramienta desarrollada, en cambio, fue diseñada desde su concepción como una solución multiusuario y multiproyecto integrada, simplificando el modelo de despliegue y evitando la necesidad de separar funcionalidades por tipo de licencia o entorno.

Capítulo 7

Calidad y testing

En esta sección se describen las actividades de verificación y validación realizadas a lo largo del proyecto, así como las decisiones adoptadas en relación al testing del sistema.

La calidad del producto fue abordada mediante distintas actividades realizadas durante todo el desarrollo. Además de las instancias de testing, se llevaron a cabo revisiones y validaciones de requisitos, evaluaciones de funcionalidades junto al *product owner*, revisiones de aspectos relacionados con la experiencia de usuario y la arquitectura de la solución, así como distintas actividades de revisión técnica. Estas instancias permitieron detectar defectos y oportunidades de mejora de forma temprana, contribuyendo a la calidad del producto desarrollado.

Dentro de los distintos niveles de prueba, el proyecto se enfocó principalmente en pruebas de aceptación, las cuales tienen como objetivo validar que el sistema desarrollado es el “producto correcto”, es decir, que satisface las necesidades del cliente y de los usuarios finales. Estas pruebas se realizaron considerando el sistema desde la perspectiva del usuario, utilizando como base de validación los casos de uso, historias de usuario y procesos definidos durante la etapa de requisitos.

7.1. Calidad del producto

La calidad del producto fue abordada mediante distintas actividades realizadas a lo largo del proyecto. A continuación se describen algunas de las principales actividades llevadas a cabo y su contribución a la calidad del sistema desarrollado.

En primer lugar, durante la etapa de requisitos, los casos de uso, el manual de usuario y el documento con los cálculos fueron revisados y validados junto al *product owner*. Estas revisiones permitieron identificar inconsistencias y aclarar aspectos funcionales antes de comenzar su implementación.

Asimismo, a medida que se desarrollaban nuevas funcionalidades, se rea-

lizaron validaciones con el *product owner* utilizando versiones funcionales del sistema. Estas instancias permitieron contrastar el comportamiento implementado con las necesidades y expectativas definidas para el producto.

También se realizaron instancias de revisión relacionadas con la experiencia de usuario y el diseño de la interfaz. En particular, se trabajó junto a la experta en UX/UI en la evaluación de distintas pantallas y flujos de navegación, identificando aspectos que podrían dificultar el uso de la herramienta o generar confusión en los usuarios.

Por otra parte, los arquitectos de Pyxis participaron en revisiones de la solución propuesta, analizando aspectos relacionados con la arquitectura y determinadas decisiones de implementación. Estas actividades permitieron detectar oportunidades de mejora antes de avanzar con las funcionalidades.

Adicionalmente, se realizaron revisiones manuales del código y de la implementación de los cálculos asociados a las métricas del sistema. Estas actividades estuvieron orientadas a verificar la correcta implementación de la lógica de negocio e identificar posibles inconsistencias o defectos en componentes críticos de la aplicación. De esta forma, se contribuyó a mejorar la calidad del producto al revisar en detalle la implementación de aspectos asociados a las métricas y a la lógica de negocio de la herramienta. Los principales hallazgos obtenidos de estas revisiones se describen posteriormente en el apartado *Revisiones de código* de la sección 7.2.2

Finalmente, muchas de las decisiones tomadas durante el desarrollo fueron analizadas y discutidas por el equipo antes de su implementación. Estas instancias permitieron evaluar distintas alternativas de solución y seleccionar aquellas que se consideraron más adecuadas para los requisitos y características del sistema.

7.2. Testing

En términos generales, el testing consiste en la ejecución del software con el objetivo de evaluar su comportamiento y detectar posibles defectos, permitiendo verificar que el sistema cumple con los requisitos establecidos y resulta adecuado para su uso (Sommerville, 2016).

A lo largo del proyecto se realizaron distintas instancias de testing con el objetivo de validar el correcto funcionamiento del sistema e identificar defectos en las funcionalidades implementadas.

Las actividades de testing variaron entre las distintas liberaciones. Mientras que en la primera liberación se utilizaron principalmente pruebas manuales orientadas a validar los flujos funcionales de la aplicación, en la segunda liberación se incorporaron pruebas unitarias, pruebas de sistema adicionales y revisiones técnicas sobre distintos componentes de la aplicación.

En las siguientes secciones se describen las actividades realizadas en cada liberación, así como los principales defectos identificados y las acciones tomadas para corregirlos.

7.2.1. Primera liberación

Una vez finalizada la primera liberación, se llevaron a cabo diversas instancias de prueba orientadas a validar el funcionamiento general del sistema. Estas pruebas fueron realizadas principalmente de forma manual, ejecutando los casos de uso definidos previamente y recorriendo los distintos flujos funcionales de la aplicación.

A través de estas pruebas se logró verificar que los flujos principales del sistema funcionaban correctamente, que la integración entre *frontend*, *backend* y servicios externos era consistente, y que los requisitos definidos para esta liberación habían sido implementados de forma adecuada.

Las actividades de testing fueron realizadas tanto por el equipo de desarrollo como por el *product owner*. Adicionalmente, se contó con la colaboración de un integrante del equipo de Pyxis, quien utilizó la herramienta mediante la creación y gestión de un proyecto propio. Esta instancia resultó especialmente valiosa, ya que permitió evaluar el sistema en condiciones cercanas a su uso real e identificar aspectos de mejora que no habían sido detectados previamente.

A continuación, se describen algunos de los defectos y oportunidades de mejora identificados durante las instancias de prueba realizadas sobre la plataforma. Estos hallazgos surgieron tanto de las pruebas ejecutadas por el equipo como del uso por parte de usuarios externos, y permitieron mejorar significativamente la calidad del sistema antes de finalizar la liberación.

Uno de los errores más relevantes se relacionó con el cálculo del Valor Planificado, donde en determinados casos se obtenían valores incoherentes, incluso superiores al 100 %. Si bien en esta primera liberación el sistema no contemplaba el soporte para múltiples proyectos por usuario, el problema se manifestó cuando un usuario que ya tenía un proyecto fue asociado manualmente a otro mediante modificaciones directas en la base de datos.

Este escenario evidenció una limitación en la implementación, ya que el cálculo del Valor Planificado no diferenciaba correctamente las tareas según el proyecto al que pertenecían. Como consecuencia, se estaban considerando tareas de distintos proyectos en un mismo cálculo, generando resultados incorrectos. Este inconveniente fue posteriormente corregido incorporando la validación correspondiente para filtrar las tareas por proyecto.

En segundo lugar, se detectó un inconveniente en el dashboard al crear un proyecto cuya fecha de inicio era futura. En estos casos, el sistema mostraba como fecha final el 31 de diciembre, lo cual resultaba inconsistente con la planificación definida. Este comportamiento fue corregido ajustando la lógica de cálculo para contemplar correctamente proyectos que aún no han comenzado.

Por otro lado, se identificaron varios problemas relacionados con la usabilidad del calendario, especialmente en la interacción mediante teclado. En particular, al presionar la tecla “Enter” para agregar una nueva semana en el calendario, el foco permanecía en la fila anterior, dificultando la carga secuencial de datos. Asimismo, se observó que no existía una forma clara de eliminar semanas, y que el comportamiento de la tecla “Enter” en la última fila no era consistente. Estos aspectos fueron corregidos, mejorando la experiencia de uso del calendario.

También se detectaron problemas visuales en la interfaz, como la aparición anticipada del *scroll* (desplazamiento) en ciertas pantallas, aun cuando existía espacio disponible. Este comportamiento generaba una experiencia poco natural para el usuario y fue corregido mediante ajustes en el *layout* (disposición) del calendario.

En relación a la visualización de métricas, se observó que la gráfica de valor planificado no se mostraba en proyectos que aún no habían comenzado. Si bien esto respondía a la lógica inicial del sistema, se determinó que la gráfica debía visualizarse igualmente, ya que representa la planificación futura del proyecto. Este comportamiento fue ajustado para contemplar correctamente estos casos.

Durante las pruebas también surgieron sugerencias funcionales que no fueron incorporadas en esta liberación por motivos de alcance. Entre ellas, se destacó la posibilidad de trabajar con *sprints* de distinta duración. Si bien se reconoció el valor de esta funcionalidad, se decidió no incluirla en el proyecto de grado, ya que el enfoque principal estaba orientado al seguimiento de métricas y no a la gestión de iteraciones tipo *sprint*. Esta mejora fue considerada como una posible línea de trabajo futuro.

Asimismo, se sugirió la incorporación de nuevas visualizaciones, como una gráfica comparativa de horas planificadas versus horas reales por semana. Esta funcionalidad tampoco fue incluida en el alcance de esta liberación, aunque se reconoce su utilidad para futuras versiones del sistema.

Otras sugerencias sí fueron incorporadas en etapas posteriores del proyecto. Entre ellas, se incluyó la posibilidad de modificar la fecha de inicio del proyecto, editar la fecha de finalización de las tareas y registrar esfuerzo en tareas por parte de otros integrantes del equipo. Estas funcionalidades fueron agregadas al *backlog* e implementadas en la segunda liberación.

Finalmente, se realizaron ajustes en la forma de representar ciertos valores en las tablas que se encuentran en la pantalla de Dashboard. En particular, se definieron criterios para diferenciar cuándo mostrar un valor numérico cero y cuándo utilizar un indicador visual, como un guion, dependiendo del contexto de la información. Por ejemplo, en el caso de las horas de calendario, se determinó que debía mostrarse el valor cero cuando no existiera dedicación planificada, ya que representa explícitamente la ausencia de horas. En cambio, para las horas correspondientes a tareas finalizadas, se optó por utilizar un guion en aquellos casos en los que no se hubieran completado tareas, con el objetivo de diferenciar entre la ausencia de datos y un valor numérico válido. Esta decisión permitió mejorar la claridad en la interpretación de la información presentada al usuario.

Los defectos identificados durante estas instancias de uso real de la plataforma estuvieron asociados principalmente a situaciones que no habían sido contempladas en las pruebas realizadas durante el desarrollo. Mientras que las actividades de testing llevadas a cabo por el equipo se enfocaron en validar los flujos definidos y el cumplimiento de los requisitos, el uso de la herramienta en un proyecto real permitió observar distintos escenarios y aspectos de usabilidad que no habían sido detectados previamente. Si bien estos escenarios podrían haberse reproducido mediante pruebas más específicas, surgieron principalmente a partir de la utilización de la herramienta y de la interacción cotidiana de los

usuarios con el sistema. Asimismo, estas instancias permitieron recibir sugerencias y observaciones relacionadas con la experiencia de usuario y las formas de trabajo de Pyxis, aportando una perspectiva complementaria a la visión técnica del equipo de desarrollo. Esta experiencia mostró la importancia de complementar las pruebas funcionales con instancias de uso real, ya que permiten identificar problemas y oportunidades de mejora difíciles de detectar en un entorno controlado.

7.2.2. Segunda liberación

Durante el desarrollo de la segunda liberación, el equipo comenzó a incorporar prácticas más estructuradas de testing, abarcando tanto pruebas unitarias como pruebas de sistema y revisiones manuales del código y del comportamiento general de la aplicación.

En primer lugar, se introdujeron pruebas unitarias sobre el *backend* del sistema. En esta etapa, se definió que no era necesario alcanzar un nivel específico de cobertura de código, pero sí asegurar la validación de los flujos más críticos de la aplicación. En particular, se priorizó la cobertura de la lógica asociada a las métricas y a las gráficas del sistema, dado que representan las partes más complejas desde el punto de vista del cálculo y procesamiento de datos. Para organizar estas tareas, el equipo utilizó la propia herramienta desarrollada, creando un proyecto específico donde cada tarea correspondía a la implementación de pruebas para una funcionalidad concreta.

En cuanto al *frontend*, y en acuerdo con los arquitectos de Pyxis, se decidió no implementar pruebas unitarias, enfocando los esfuerzos en la validación manual de la interfaz y la experiencia de usuario.

Revisiones de código

Se realizaron revisiones específicas del código, enfocadas principalmente en la validación de los cálculos de métricas, las líneas de las gráficas y el control adecuado de permisos según correspondiera. Estas revisiones permitieron identificar y corregir diversos defectos que no habían sido detectados en etapas anteriores.

En primer lugar, se detectaron inconsistencias en el manejo de fechas en distintas funcionalidades. Por ejemplo, el rango de fechas mostrado en el encabezado de las tablas del dashboard no siempre se devolvía correctamente si el proyecto no había comenzado aún.

También se identificaron errores en los cálculos de las métricas. En particular, en el método encargado de generar la información de la gráfica individual (*getIndividualEarnedValueChartData*), no se consideraba correctamente el cierre de semana. Se utilizaban las 00:00 del domingo en lugar de las 23:59, lo que podía provocar que ciertas tareas no fueran tenidas en cuenta. De forma similar, en el cálculo del esfuerzo real total de tareas finalizadas (*getTotalUserRealEffortForFinishedTasksByProjectId*), que se utiliza para obtener las líneas

de proyección, tampoco se contemplaban correctamente las tareas completadas en la fecha actual. Se obtenían las tareas finalizadas hasta el día anterior al actual, sin contemplar aquellas que pudieran haber sido completadas en el transcurso de la fecha actual. Ambos problemas fueron corregidos ajustando las referencias temporales utilizadas.

Por otro lado, se detectaron otras inconsistencias en la lógica de proyección de métricas a futuro. En particular, se corrigieron errores en los métodos responsables de redistribuir horas estimadas entre tareas (*applyBorrowingAdjustmentsToTasks*) y de calcular la disponibilidad restante de horas de calendario para la semana actual (*getUserAdjustedWeeksForProjection*), mejorando la precisión de las proyecciones.

En cuanto al control de accesos, se observó que algunas validaciones de roles no estaban siendo aplicadas en ciertos *endpoints*. Estas verificaciones fueron incorporadas tanto en la capa de controladores como en la de servicios, fortaleciendo la seguridad del sistema. Durante este proceso, también se identificó y eliminó código que no estaba siendo utilizado.

A nivel general, el equipo considera que las revisiones de código resultaron especialmente valiosas para detectar defectos que difícilmente hubieran sido identificados mediante pruebas funcionales convencionales. Muchos de los problemas encontrados estaban asociados a casos límite en la lógica de negocio, como el manejo preciso de fechas y horarios o los cálculos utilizados para proyectar métricas futuras. Si bien estos defectos podían manifestarse en determinados escenarios, reproducir sistemáticamente las condiciones necesarias para evidenciarlos mediante pruebas requería un nivel de detalle mayor y la construcción de casos de prueba más específicos.

Asimismo, las revisiones permitieron identificar aspectos que no necesariamente generan un comportamiento visible e inmediato para el usuario, como la ausencia de validaciones de permisos en determinados puntos del sistema o la presencia de código sin utilización. En estos casos, el análisis directo de la implementación permitió detectar problemas que no habían surgido durante las pruebas realizadas sobre el sistema. Esto mostró que la revisión del código permite detectar problemas que pueden pasar desapercibidos durante la ejecución de pruebas funcionales.

Rendimiento

Respecto a los tiempos de respuesta de la aplicación, luego de discutirlo con el *product owner* y los arquitectos de Pyxis, se optó por una evaluación basada en pruebas manuales. Como referencia, se utilizó la heurística de Jakob Nielsen, la cual establece que si una aplicación demora más de diez segundos en responder, se debe brindar al usuario algún tipo de retroalimentación indicando el estado de la solicitud (Nielsen, 2014).

A partir de pruebas realizadas con volúmenes de datos representativos de un uso habitual del sistema, incluyendo volúmenes típicos de uso, con decenas de tareas por proyecto, múltiples usuarios por equipo y varias semanas de planificación visibles en el calendario, se observó que en ningún caso se superaba

dicho umbral.

En determinadas situaciones, el tiempo de respuesta se ubicaba entre uno y diez segundos. A modo de ejemplo el proyecto utilizado para la segunda liberación, cuenta con 71 tareas y el tiempo de respuesta promedio en la página de tareas es de 3.1 segundos. En consecuencia, para estas vistas donde la carga puede demorar algunos segundos, como la pantalla de Dashboard, la de Tareas o el Calendario, se incorporaron indicadores visuales de carga con el objetivo de mejorar la experiencia de usuario.

Pruebas del sistema

Dentro de esta segunda liberación, además de las pruebas unitarias y revisiones de código, se llevaron a cabo pruebas de sistema orientadas a validar el correcto funcionamiento de los principales flujos de la aplicación. Estas pruebas fueron diseñadas tomando como referencia los requisitos definidos en ambas liberaciones y se ejecutaron una vez finalizada la implementación de las funcionalidades. Durante su ejecución, se verificaron tanto aspectos funcionales como de integración y usabilidad, y en algunos casos se detectaron defectos que requirieron ajustes en la implementación.

En este contexto, se validaron flujos clave como la autenticación, la navegación general de la aplicación, el control de accesos, la creación y gestión de proyectos, el uso del calendario y la visualización de métricas. Asimismo, se evaluaron distintos escenarios representativos, como proyectos sin tareas, proyectos finalizados o con insuficiencia de horas en el calendario, verificando que el sistema respondiera correctamente en cada caso.

En primer lugar, se validaron los flujos relacionados con la autenticación y gestión de sesiones. Esto incluyó el inicio de sesión mediante Microsoft, el cierre de sesión y el comportamiento del sistema ante múltiples sesiones simultáneas en distintos navegadores.

En todos los casos se verificó que la autenticación se realizara correctamente, que los tokens de sesión fueran gestionados de forma adecuada y que las rutas protegidas no fueran accesibles sin credenciales válidas. Asimismo, se comprobó el correcto funcionamiento de mensajes de error y traducciones en la pantalla de inicio de sesión.

Por otro lado, se evaluaron los flujos de navegación general de la aplicación, incluyendo la visualización de las distintas vistas y el comportamiento del sidebar. Se verificó que los distintos roles de usuario accedieran correctamente a las funcionalidades disponibles, que las rutas funcionaran correctamente con soporte multilenguaje y que el contexto del proyecto activo se actualizara de forma consistente. También se comprobó la correcta visualización de información del usuario, como su rol y foto de perfil.

En relación al control de accesos, se realizaron pruebas específicas sobre la pantalla de administración, verificando que únicamente los usuarios con permisos adecuados pudieran acceder a esta funcionalidad, tanto desde la interfaz como mediante acceso directo por URL. Asimismo, se validó que las operaciones disponibles, como la asignación de roles, funcionaran correctamente y respetaran

las restricciones definidas.

Se probaron también los flujos de creación y gestión de proyectos, verificando la correcta creación de nuevos proyectos, las validaciones de los formularios, la persistencia de datos y la actualización del proyecto activo dentro de la plataforma.

En cuanto a las vistas principales del sistema, se validó el correcto funcionamiento del dashboard, la página de proyecto, la gestión de tareas y el calendario. En el caso del dashboard, se comprobó la correcta carga de datos y la visualización de métricas, aunque no se validó en esta instancia la exactitud de los cálculos. También se probaron distintos escenarios particulares, como proyectos sin tareas, proyectos finalizados y proyectos con insuficiencia de horas en el calendario, verificando que el sistema mostrara los mensajes y alertas correspondientes en cada caso.

El módulo de calendario fue objeto de pruebas específicas, validando la correcta visualización y modificación de la dedicación horaria de los usuarios, la posibilidad de agregar nuevas semanas y la actualización de las métricas en función de los cambios realizados. Se comprobó que las modificaciones realizadas en el calendario impactaran correctamente en la planificación del proyecto y en las alertas del sistema.

Finalmente, se realizaron pruebas más detalladas sobre el cálculo de métricas y la consistencia de las gráficas del sistema. Para ello, se definieron escenarios controlados con valores conocidos de tareas, esfuerzos y disponibilidad horaria, permitiendo verificar que los indicadores calculados se comportaran según lo esperado tanto a nivel individual como a nivel de proyecto.

Durante la ejecución de estas pruebas se detectaron algunos defectos y oportunidades de mejora. En los flujos de creación y edición de proyectos se identificaron problemas relacionados con la validación de ciertos campos, como la posibilidad de ingresar valores no válidos en la dedicación semanal, así como inconsistencias en la actualización de datos al modificar miembros de un equipo. Estos problemas fueron posteriormente corregidos mediante ajustes en las validaciones y en la lógica de actualización de la información.

Asimismo, durante las revisiones realizadas por el *product owner* se identificaron distintos aspectos relacionados con la experiencia de usuario. En particular, se detectaron problemas al ingresar fechas manualmente mediante teclado, ya que el sistema no permitía escribir la fecha correctamente en su totalidad. Este defecto no había sido identificado previamente debido a que las distintas pruebas relacionadas a fechas fueron realizadas seleccionando la fecha en el calendario y no escribiendo la misma mediante el teclado.

También se observaron inconvenientes al navegar entre campos de formularios utilizando las teclas “Tab” y “Enter”, especialmente en escenarios de interacción rápida. Este comportamiento se manifestaba principalmente en el entorno de producción, donde los tiempos de respuesta son mayores a los del entorno local, y fue corregido ajustando la gestión del foco en la interfaz.

Por último, el *product owner* también identificó que, luego de editar la descripción de una tarea, al presionar “Tab”, la navegación mediante teclado no seguía el orden esperado de los campos. En particular se seleccionaba el campo

de horas estimadas y no el del usuario asignado (el cual es el siguiente campo luego de la descripción). Esto no era un error sino que no estaba implementado poder manejar el selector de usuarios mediante el teclado. Luego de la sugerencia del *product owner*, se decidió incorporar el soporte necesario para mejorar la experiencia de usuario.

Estas pruebas permitieron validar tanto el correcto funcionamiento de los flujos y funcionalidades implementadas como distintos aspectos relacionados con la experiencia de usuario. Mientras que las pruebas definidas por el equipo de desarrollo estuvieron orientadas principalmente a verificar el cumplimiento de los requisitos funcionales, las validaciones realizadas por el *product owner* se enfocaron en asegurar que la aplicación respondiera adecuadamente a las necesidades y expectativas de Pyxis respecto al producto.

A modo de ejemplo, se presenta a continuación el caso de prueba definido por el equipo de desarrollo para validar el flujo de creación de proyectos. El resto de los casos de prueba pueden consultarse en el anexo [E](#). En el mismo, se cuenta con 17 casos de prueba en donde se evalúan las funcionalidades principales del sistema.

Caso de prueba: Crear un nuevo proyecto

Descripción

Este caso de prueba tiene como objetivo validar que un usuario pueda crear un nuevo proyecto desde la interfaz de la aplicación, verificando tanto el correcto funcionamiento del flujo como las validaciones asociadas al formulario.

Condiciones previas

- El usuario se encuentra autenticado en el sistema.
- El usuario posee rol de operador o administrador.
- El sistema se encuentra disponible y funcionando correctamente.

Datos de prueba

- Nombre del proyecto válido.
- Fecha de inicio válida.
- Líder existente en la plataforma.
- Dedicación semanal válida: valor numérico positivo y menor a 168 horas (equivalente a la cantidad de horas máxima en una semana)

También se consideran datos inválidos para validación:

- Nombre del proyecto vacío o ya existente en la plataforma.
- Fecha de inicio inválida.
- Ausencia de líder.

- Dedicación semanal nula, negativa, no numérica o mayor a 168 horas.

Pasos a ejecutar

1. Apretar el botón “Crear proyecto”.
2. Ingresar los datos requeridos en el formulario.
3. Crear proyecto.

Resultado esperado

- El sistema crea el proyecto correctamente.
- Si el usuario se asignó a sí mismo como líder, el proyecto queda seleccionado como proyecto activo.
- El proyecto se visualiza en la lista de proyectos.
- El botón de creación sólo está disponible para usuarios con rol adecuado.
- Al cancelar la operación, no se persisten los datos ingresados.
- Las validaciones del formulario impiden el ingreso de datos inválidos.

Resultado obtenido

El flujo fue ejecutado exitosamente en términos generales. El proyecto se creó correctamente y se reflejó en la interfaz como se esperaba. En la página de Proyectos se muestra el proyecto creado y, en caso de haberse asignado como líder, el proyecto queda seleccionado en la plataforma. El botón para crear el nuevo proyecto se encuentra disponible solo para los roles de Administrador u Operador.

Sin embargo, durante la ejecución se detectaron defectos en la validación de la dedicación semanal, ya que inicialmente se permitía ingresar valores no numéricos o superiores al máximo esperado. Estos inconvenientes fueron corregidos mediante la incorporación de validaciones más estrictas.

7.3. Consideraciones finales

Las actividades de calidad y testing realizadas a lo largo del proyecto permitieron evaluar el funcionamiento de la herramienta desde distintas perspectivas. Las pruebas manuales, las revisiones de código, las pruebas unitarias incorporadas en la segunda liberación y las pruebas de sistema resultaron complementarias entre sí, permitiendo detectar defectos tanto a nivel funcional como técnico.

Asimismo, la participación del *product owner*, la experta UX/UI, los arquitectos de Pyxis y otros usuarios que utilizaron la aplicación durante el desarrollo, aportó una visión externa que permitió identificar oportunidades de mejora que no siempre eran evidentes para el equipo de desarrollo.

Por otra parte, el hecho de utilizar la propia herramienta para gestionar el proyecto de grado permitió ayudar a detectar problemas de usabilidad, realizar ajustes sobre la experiencia de usuario y verificar el comportamiento de las métricas y funcionalidades implementadas.

En conjunto, estas actividades contribuyeron a mejorar progresivamente la calidad de la aplicación y permitieron evaluar el producto desde distintas perspectivas.

Capítulo 8

Lecciones aprendidas

En este capítulo se presentan distintos aprendizajes identificados a lo largo del desarrollo del proyecto, tanto a nivel técnico como organizacional. Muchos de estos aprendizajes surgieron a partir de decisiones tomadas durante la planificación, la implementación y la validación del sistema, así como también del intercambio continuo con profesionales de distintas áreas dentro de Pyxis.

8.1. Gestión del alcance y definición de requisitos

Uno de los principales aprendizajes del proyecto estuvo relacionado con la planificación del alcance y la definición progresiva de requisitos. Al comienzo del proyecto se definieron distintas funcionalidades y objetivos para las liberaciones, pero a medida que avanzó el desarrollo el equipo identificó que varias de ellas requerían más esfuerzo del esperado, tanto desde el punto de vista técnico como organizacional. Esto llevó a priorizar funcionalidades y ajustar parte del alcance inicialmente planteado.

Además, durante distintas etapas del proyecto existieron funcionalidades cuya idea general estaba clara, pero cuyos detalles todavía necesitaban mayor definición. Un ejemplo de esto ocurrió durante la segunda liberación con la definición de roles a nivel de sistema y proyecto. La forma en que se modelaran estos roles afectaba directamente el *backend*, el *frontend*, la arquitectura de permisos y los distintos flujos de usuario, por lo que fue necesario dedicar tiempo adicional a validar el comportamiento esperado antes de comenzar la implementación.

Por otro lado, el equipo considera que haber planificado y validado previamente ciertos aspectos importantes del sistema resultó útil para reducir problemas posteriores. En particular, disponer desde etapas tempranas de una idea relativamente clara de las funcionalidades principales de cada liberación ayudó a mantener una dirección consistente durante el desarrollo, incluso cuando posteriormente surgían ajustes o nuevos requerimientos.

Asimismo, resultó especialmente útil validar previamente la lógica de las

métricas y cálculos del sistema antes de comenzar su implementación. El documento elaborado por el equipo para describir el funcionamiento de dichas métricas permitió detectar inconsistencias, discutir distintos enfoques y confirmar junto al *product owner* que el comportamiento esperado era correcto antes de escribir código.

8.2. Diseño incremental y decisiones arquitectónicas

Otro aprendizaje importante estuvo relacionado con la forma en que se diseñó la arquitectura del sistema. Desde la primera liberación se tomaron varias decisiones pensando en funcionalidades que serían incorporadas más adelante. Esto permitió evitar cambios grandes de arquitectura entre liberaciones y facilitó la evolución progresiva del sistema.

Por ejemplo, aunque inicialmente el sistema manejaba un único usuario por proyecto, varias partes de la arquitectura y del modelo de datos ya contemplaban escenarios con múltiples usuarios y múltiples proyectos. Algo similar ocurrió con algunos cálculos relacionados con métricas y gráficas, que desde etapas tempranas fueron diseñados considerando escenarios multiusuario.

Si bien igualmente fue necesario realizar ajustes y mejoras durante la segunda liberación, el hecho de haber pensado ciertas estructuras desde el comienzo simplificó considerablemente la evolución posterior del sistema.

Otro aspecto que el equipo considera positivo fue haber implementado despliegues automáticos desde etapas tempranas del proyecto. Si bien configurar estos procesos implicó esfuerzo al inicio, posteriormente permitió validar cambios rápidamente y mantener una versión actualizada de la aplicación desplegada de forma continua. Esto facilitó las pruebas, revisiones y validaciones realizadas durante el desarrollo.

8.3. Testing y validación continua

A medida que se fueron implementando funcionalidades, integrantes de Pyxis comenzaron a utilizar la herramienta y reportar comportamientos no esperados, errores o posibles mejoras. Estas pruebas permitieron detectar situaciones que no habían aparecido durante el desarrollo o las validaciones internas realizadas por el equipo.

En varios casos, los problemas identificados surgieron recién cuando la herramienta empezó a utilizarse en escenarios más cercanos a un uso real. Esto ocurrió, por ejemplo, con algunos comportamientos relacionados con cálculos, manejo de fechas, navegación mediante teclado o interacción rápida entre distintos campos de la interfaz.

El equipo considera que el hecho de utilizar la propia herramienta desarrollada durante el proyecto resultó especialmente útil para detectar este tipo de situaciones. Varias funcionalidades fueron refinadas luego de ser utilizadas en

contextos reales o semireales, algo que permitió validar no solamente el correcto funcionamiento técnico, sino también aspectos relacionados con experiencia de usuario y usabilidad.

Asimismo, las reuniones frecuentes con el *product owner* resultaron importantes para validar continuamente el estado del producto y discutir decisiones funcionales y técnicas. Estas reuniones permitían revisar avances, aclarar dudas y validar funcionalidades implementadas. En determinados momentos del proyecto, especialmente durante la segunda liberación, las reuniones pasaron a realizarse con mayor frecuencia para poder resolver temas pendientes de forma más rápida.

8.4. Colaboración con profesionales de Pyxis

A lo largo del proyecto el equipo trabajó junto a distintos profesionales de Pyxis, incluyendo arquitectos de software, una profesional de UX/UI y el *product owner*. Estas interacciones permitieron validar decisiones técnicas y funcionales, además de obtener distintas perspectivas sobre el sistema y el proceso de trabajo.

Como parte del cierre del proyecto, el equipo realizó encuestas a varios de estos profesionales con el objetivo de recopilar observaciones sobre la comunicación mantenida, la dinámica de trabajo, las decisiones tomadas y el resultado final del sistema. Las preguntas realizadas y las respuestas completas obtenidas se presentan en el anexo [F](#).

8.4.1. Trabajo con UX/UI

El trabajo junto a la diseñadora UX/UI implicó varias iteraciones antes de llegar a las versiones finales de las pantallas y flujos del sistema. Generalmente el equipo preparaba bocetos iniciales y posteriormente estos eran discutidos y refinados en reuniones o mediante intercambio de correos.

Uno de los aprendizajes relacionados con este proceso fue entender que el diseño no dependía únicamente del trabajo interno del equipo. Para avanzar correctamente fue necesario coordinar reuniones, validar flujos, discutir detalles funcionales y esperar nuevas versiones de diseños antes de implementar determinadas pantallas. En algunos casos esto llevó a postergar implementaciones para evitar retrabajo posterior.

Además, durante el proyecto se utilizaron herramientas basadas en inteligencia artificial para generar propuestas de interfaces y componentes reutilizables. Según comentó la diseñadora en la encuesta realizada al finalizar el proyecto, este enfoque permitió acelerar parte del trabajo relacionado con UX/UI y facilitar la reutilización de componentes por parte del equipo de desarrollo.

En dicha encuesta también destacó positivamente la dinámica de trabajo mantenida durante el proyecto, mencionando que la comunicación fue fluida y que las distintas instancias de interacción permitieron avanzar sin grandes fricciones. Asimismo, señaló que los distintos bocetos realizados por el equipo ayudaron a discutir necesidades y decisiones de diseño antes de cada reunión.

Otro aspecto mencionado en la encuesta fue que, desde su perspectiva, el proyecto logró mejorar considerablemente la usabilidad y presentación visual de la herramienta respecto a las ideas iniciales planteadas al comienzo del desarrollo.

8.4.2. Arquitectura e infraestructura

Las reuniones mantenidas con arquitectos de software de Pyxis fueron utilizadas principalmente para validar decisiones relacionadas con autenticación, infraestructura, despliegue y organización general de la arquitectura del sistema.

Uno de los ejemplos más relevantes surgidos a partir de estas reuniones fue la integración con Microsoft 365. Inicialmente el equipo planeaba implementar esta funcionalidad desde el *backend*, pero luego de discutir distintas alternativas junto a los arquitectos se decidió utilizar NextAuth.js desde el *frontend*, enfoque que posteriormente resultó más adecuado para el tipo de aplicación desarrollada.

Asimismo, las distintas reuniones ayudaron a validar el uso de AWS, Kubernetes y varios aspectos relacionados con despliegue e infraestructura en la nube. En algunos casos estas instancias permitieron detectar enfoques que podían simplificarse antes de comenzar su implementación.

A partir de la encuesta realizada al finalizar el proyecto, los arquitectos destacaron positivamente la predisposición del equipo para investigar problemas y comprender situaciones técnicas antes de realizar consultas. También mencionaron la importancia de contar, en futuras experiencias similares, con instancias más formales de nivelación técnica y organizacional entre estudiantes y tutores desde etapas tempranas del proyecto.

Por otro lado, también surgió como aprendizaje la importancia de contar con tiempos y recursos suficientemente planificados para este tipo de acompañamiento técnico, ya que parte de las interacciones dependían de la disponibilidad particular de los distintos profesionales involucrados.

8.4.3. Encuestas y feedback del proceso

El equipo considera que las encuestas realizadas al finalizar el proyecto fueron útiles no solamente como instancia de cierre, sino también como forma de reflexionar sobre cómo fue percibido el proceso desde distintos roles externos al desarrollo técnico del sistema.

En general, las respuestas obtenidas coincidieron en varios puntos. Por un lado, distintos participantes destacaron positivamente el compromiso del equipo, la capacidad de adaptación ante cambios y la predisposición para incorporar sugerencias y ajustes durante el proyecto.

Por otro lado, también surgieron observaciones relacionadas con la dificultad de estimar correctamente tiempos y esfuerzo al comienzo del proyecto, especialmente debido a la falta de experiencias previas similares y a la naturaleza cambiante de algunos requisitos.

Además, las respuestas reforzaron algo que el equipo también identificó que fue la importancia de mantener comunicación frecuente entre todos los actores

involucrados y validar continuamente tanto decisiones funcionales como técnicas para reducir retrabajo posterior.

8.5. Evaluación del proceso de desarrollo utilizando la herramienta creada

Como parte del proyecto, el equipo decidió utilizar la propia herramienta desarrollada para gestionar parte del trabajo realizado durante ambas liberaciones. Esto permitió analizar retrospectivamente el proceso de desarrollo utilizando las métricas implementadas en la plataforma.

A continuación se presentan los resultados obtenidos al finalizar cada liberación, junto con un análisis basado en las métricas de valor ganado, horas planificadas y horas reales registradas durante el proyecto.

8.5.1. Primera liberación

La figura 8.1 muestra el dashboard correspondiente al tramo final de la primera liberación del proyecto. La duración del mismo fue de aproximadamente cinco semanas. Se muestran únicamente dichas semanas debido a que fue en ese período cuando el equipo comenzó a utilizar la plataforma para la gestión del Proyecto de Grado. Además, la imagen fue tomada varias semanas posteriores a la finalización de la implementación del Proyecto de Grado, por lo que la tabla correspondiente a la última semana no presenta información.

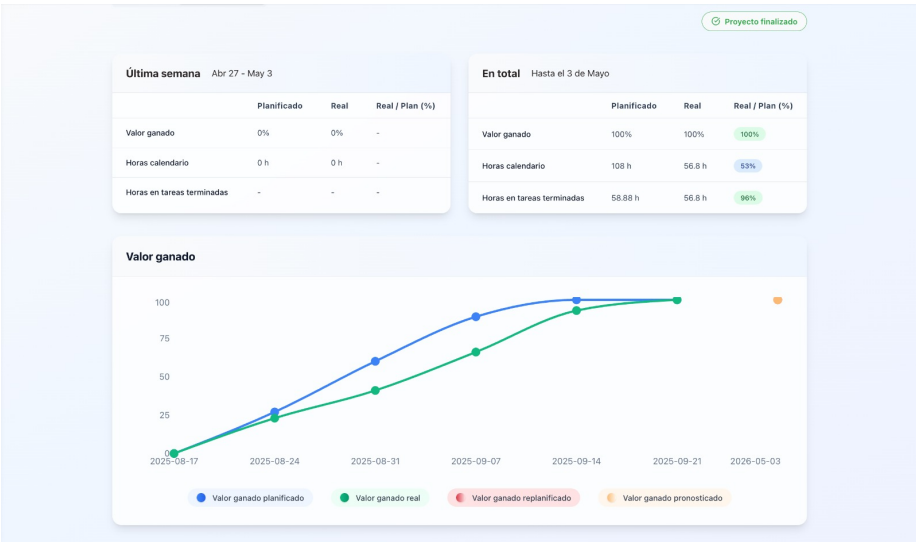


Figura 8.1: Dashboard con las métricas de la primera liberación del Proyecto de Grado.

A nivel general, puede observarse que el proyecto alcanzó, tal como estaba planificado, el 100 % de valor ganado, indicando que las funcionalidades previstas para esta liberación fueron completadas. Sin embargo, el comportamiento de las métricas permite extraer varias conclusiones sobre cómo se desarrolló el trabajo del equipo.

En primer lugar, se observa que las horas calendario planificadas fueron mayores que las horas reales registradas. Mientras que el calendario de la liberación indicaba una disponibilidad total de 108 horas para las tareas planificadas, las horas efectivamente registradas fueron aproximadamente 57.

Tal como se mencionó en el capítulo 3, el equipo utilizó la plataforma para planificar y realizar el seguimiento de las tareas vinculadas al desarrollo e implementación de la herramienta. Para construir el calendario de la liberación, se partió de la cantidad de horas semanales que cada integrante debía dedicar al Proyecto de Grado y, a partir de ellas, se estimó cuántas podrían estar disponibles para las tareas planificadas de desarrollo.

La diferencia observada entre las horas calendario y las horas reales registradas indica que dicha estimación no fue acertada. Al elaborar el cronograma, se sobrestimó la cantidad de horas que podrían dedicarse a las tareas planificadas, debido a que se subestimó el esfuerzo requerido por otras actividades necesarias para el proyecto que no fueron consideradas al momento de realizar esa estimación.

Entre las actividades que se sabía que formarían parte del proyecto, pero que fueron excluidas intencionalmente de la planificación en la plataforma, se encontraron las reuniones con el *product owner*, las reuniones internas del equipo y otras actividades generales de gestión. Por otra parte, hubo actividades cuyo esfuerzo no fue considerado adecuadamente al construir el calendario. Entre ellas se encontraron la planificación de la segunda liberación y la elaboración de documentación asociada a la primera liberación.

Como puede observarse en la figura 8.1, las horas de implementación consideradas al construir el calendario permitían completar la liberación en cuatro semanas. Sin embargo, debido a que las actividades no planificadas requirieron una dedicación mayor a la estimada, el equipo pudo dedicar menos horas semanales al desarrollo de la herramienta, lo que extendió la duración real de la liberación a cinco semanas.

Es importante destacar que las estimaciones realizadas para las tareas de implementación fueron adecuadas, ya que estas pudieron completarse con una cantidad de horas similar a la planificada. Si bien existieron desvíos puntuales, las estimaciones realizadas inicialmente estuvieron relativamente alineadas con el esfuerzo finalmente requerido para completar las funcionalidades implementadas. La diferencia se produjo en la distribución semanal de dichas horas. Como resultado, las tareas planificadas se completaron en el esfuerzo estimado, pero debieron ejecutarse a lo largo de un período más extenso que el previsto inicialmente.

En relación con la gráfica, se observa que la curva correspondiente al valor ganado se mantuvo por debajo de la curva de valor planificado durante gran parte de la etapa final de la primera liberación. Esto refleja que el avance real

del proyecto fue más lento de lo esperado, como consecuencia de lo mencionado anteriormente respecto a que las horas calendario reales resultaron inferiores a las planificadas.

Aun así, hacia el cierre de la liberación puede observarse que ambas líneas convergen en el 100 %, indicando que finalmente las tareas definidas para esta etapa lograron completarse en su totalidad.

8.5.2. Segunda liberación

La figura 8.2 presenta el dashboard correspondiente a la segunda liberación. La duración de esta etapa fue de aproximadamente diez semanas. La imagen fue tomada varias semanas posteriores a la finalización de la implementación del Proyecto de Grado, por lo que la tabla correspondiente a la última semana no presenta información.

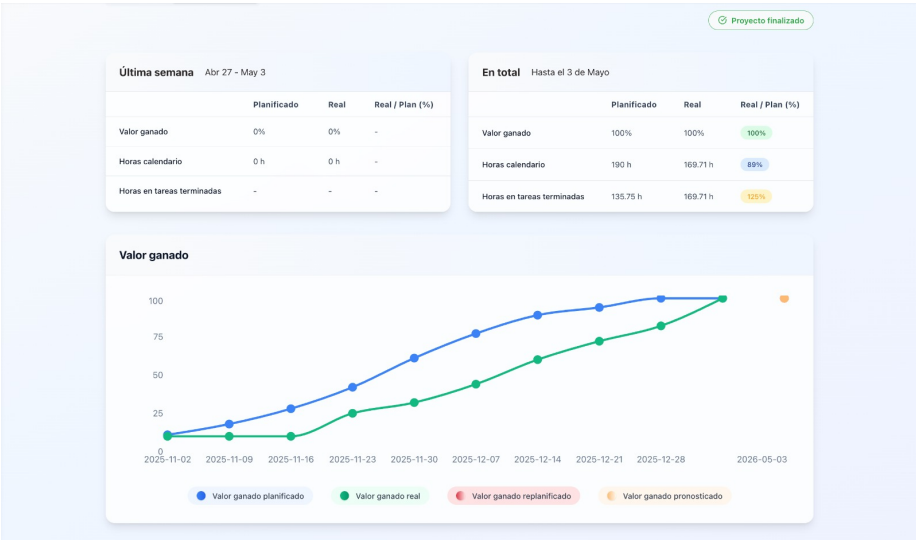


Figura 8.2: Dashboard con las métricas de la segunda liberación del Proyecto de Grado.

Como puede observarse, existe una diferencia entre la fecha de finalización de la primera liberación (figura 8.1) y la fecha de inicio de la segunda liberación (figura 8.2). Durante ese período el equipo no comenzó tareas de implementación asociadas a esta nueva etapa, sino que estuvo enfocado en realizar correcciones puntuales reportadas por Pyxis sobre la primera liberación y en actividades de planificación. Entre estas actividades se incluyeron la obtención de nuevos requisitos, el diseño del nuevo modelo de datos y la definición de la arquitectura necesaria para esta segunda etapa, entre otras tareas.

Al igual que en la primera liberación, se observa que las horas calendario pla-

nificadas fueron superiores a las horas reales registradas. En este caso, el calendario acumulado mostraba aproximadamente 190 horas planificadas, mientras que las horas reales registradas fueron cercanas a 170 horas.

Sin embargo, a diferencia de lo ocurrido en la primera liberación, en esta etapa las horas reales registradas superaron a las horas planificadas para las tareas finalizadas. Mientras las tareas completadas acumulaban alrededor de 136 horas estimadas, el esfuerzo real registrado fue cercano a 170 horas. Esto evidencia que varias funcionalidades requirieron un esfuerzo mayor al inicialmente estimado.

El equipo considera que esta situación estuvo principalmente relacionada con la incorporación de funcionalidades técnicamente más complejas que las desarrolladas en la primera etapa. Entre ellas se destacan las fechas previstas de finalización, las nuevas líneas de proyección y el manejo de roles a través de las distintas funcionalidades de la herramienta. Otra funcionalidad que demandó más horas de las estimadas fue el registro de esfuerzos para una tarea. Esta funcionalidad se dividió en varias tareas relacionadas, como la creación del temporizador, la lógica de *backend* y la visualización de los registros de esfuerzo, algunas de las cuales requirieron más tiempo del originalmente planificado. En general, varias tareas asociadas a estas funcionalidades implicaron un mayor esfuerzo debido a que se desconocía en detalle la complejidad técnica que involucraban a nivel de implementación.

En cuanto a la gráfica de valor ganado, puede observarse nuevamente que el avance real del proyecto se mantuvo por debajo del avance planificado durante toda la liberación. Esto refleja que no se logró alcanzar el ritmo de avance esperado, principalmente debido a la complejidad de las funcionalidades incorporadas y a la aparición de tareas adicionales que no habían sido previstas originalmente. No obstante, también puede apreciarse que hacia el final de la liberación el proyecto alcanza el 100 % de valor ganado.

Respecto a las líneas de proyección de replanificación y pronóstico incorporadas en esta liberación, las mismas no se visualizan en la gráfica debido a que, al momento de tomar la captura, el proyecto ya se encontraba finalizado y no existían tareas pendientes sobre las cuales realizar proyecciones futuras.

8.5.3. Comentarios finales

En términos generales, el uso de la herramienta durante el desarrollo permitió analizar de forma práctica cómo evolucionó el proyecto a lo largo de las liberaciones y observar el comportamiento de distintas métricas durante el trabajo del equipo. Además, permitió identificar comportamientos relacionados con estimaciones, planificación y dedicación real, especialmente en aquellas tareas cuya complejidad técnica era más difícil de prever al inicio.

También se pudo observar que, si bien en ambas liberaciones el proyecto logró completarse, existieron diferencias entre las horas estimadas y las horas finalmente registradas, principalmente en funcionalidades más complejas de la segunda liberación. Esto permitió obtener una visión más clara sobre cómo impactan las decisiones de planificación y estimación en el avance real del proyecto.

Un error, que representa a su vez una lección aprendida, fue haber planificado únicamente las tareas de implementación. Si bien las estimaciones realizadas para dichas tareas se consideran, en términos generales, adecuadas tanto para la primera como para la segunda liberación, posteriormente se concluyó que también habría sido beneficioso estimar y planificar las actividades vinculadas a la planificación, las revisiones, la documentación y otras tareas complementarias. Contar con esta información habría proporcionado una visión más completa del esfuerzo requerido por el proyecto y habría permitido obtener datos útiles para mejorar futuras estimaciones.

La decisión de no estimar este tipo de actividades se tomó al comenzar a utilizar la herramienta. En ese momento, varias de ellas eran desconocidas para el equipo o resultaban difíciles de estimar. Además, algunas actividades, como las reuniones semanales entre los integrantes del equipo, fueron excluidas deliberadamente luego de discutirlo con el *product owner*, ya que, si bien eran necesarias para alcanzar los objetivos del proyecto, se consideró que no aportaban valor directo al producto.

Sin embargo, otras actividades, como la creación y actualización del documento de cálculos del sistema o del manual de usuario, podrían haber sido incorporadas a la planificación y estimadas explícitamente. Esto habría permitido comprender mejor la distribución real del esfuerzo del equipo y adquirir experiencia en la estimación de este tipo de tareas.

Este aspecto fue identificado retrospectivamente como un aprendizaje para el equipo. En particular, se considera que habría sido beneficioso incluir y estimar estas tareas desde la primera liberación, ya que ello habría permitido analizar las diferencias entre las horas planificadas y las horas efectivamente dedicadas a las mismas. A partir de esa información, habría sido posible ajustar la planificación de la segunda liberación y obtener estimaciones más precisas basadas en la experiencia adquirida durante la primera.

Capítulo 9

Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones obtenidas a partir del desarrollo del proyecto, evaluando los resultados alcanzados en relación con los objetivos planteados inicialmente. Además, se analizan las principales dificultades encontradas durante el proceso, los aportes realizados y algunos aspectos que podrían haberse abordado de forma diferente o más completa.

Por otra parte, se describen posibles líneas de trabajo futuro y funcionalidades que podrían incorporarse en nuevas versiones de la herramienta, tanto para ampliar sus capacidades como para mejorar aspectos relacionados con la administración, escalabilidad y experiencia de uso.

9.1. Conclusiones

El proyecto cumplió con los objetivos y requerimientos planteados inicialmente, logrando desarrollar una herramienta funcional orientada a la gestión y seguimiento de proyectos mediante métricas basadas en la técnica de Valor Ganado.

Respecto al primer objetivo del proyecto, que consistía en *desarrollar un sistema de software adaptable a las necesidades de equipos de desarrollo de software, orientado al seguimiento y gestión de proyectos mediante la técnica de Valor Ganado*, se concluye que el mismo fue cumplido satisfactoriamente.

En particular, el sistema desarrollado satisface los requerimientos definidos para la herramienta. La solución implementada proporciona una plataforma multiusuario y multiproyecto, ya que permite la gestión simultánea de múltiples proyectos y la participación de distintos equipos de trabajo. Asimismo, se cumple con el objetivo de aplicar la técnica de Valor Ganado mediante la implementación del seguimiento de proyectos basado en esta metodología.

La herramienta también permite realizar la planificación y seguimiento de proyectos, así como con el cálculo de métricas fundamentales, tales como el Valor

Ganado, el Valor Planificado y las fechas previstas de fin del proyecto basadas en procesos de replanificación o pronóstico, entre otras métricas. Las mismas se actualizan de forma dinámica a lo largo de la ejecución del proyecto y permiten visualizar el avance en tiempo real, facilitando la detección temprana de desvíos y el análisis de sus posibles causas.

De igual forma, la herramienta contempla la gestión de capacidad de los equipos, permitiendo que las variaciones en la disponibilidad de recursos impacten en la planificación y en el cálculo de métricas. Para ello se dispone del calendario del equipo, el cual es tenido en cuenta para el cálculo de las distintas métricas ya mencionadas. Además, se permite la gestión de los integrantes del equipo para cada proyecto.

La herramienta se encuentra disponible en el ambiente de trabajo de Pyxis y actualmente se está comenzando un nuevo proyecto piloto que utilizaría la aplicación desarrollada. Esto apoya el principal objetivo de este proyecto de grado de poder contar con un sistema de software que permita la gestión con la técnica de Valor Ganado en el contexto de la empresa Pyxis.

El segundo objetivo del proyecto, que consistía en *recorrer de manera integral las distintas etapas de la Ingeniería de Software aplicadas al desarrollo de un producto real, con el propósito de analizar y discutir los aciertos y oportunidades de mejora de este tipo de vínculo entre industria y academia*, también se considera cumplido.

El desarrollo de este proyecto implicó atravesar todas las fases del ciclo de vida del software, incluyendo el relevamiento y análisis de requisitos, el diseño arquitectónico, la implementación, la validación y las actividades de testing. Además, fue necesario desarrollar funcionalidades tanto *frontend* como *backend*, desplegar la solución sobre infraestructura en la nube y resolver problemas técnicos relacionados con autenticación, persistencia de datos, despliegue sobre Kubernetes y cálculo de métricas.

Uno de los principales aciertos del proyecto fue la participación activa de Pyxis durante todo el proceso de desarrollo. La posibilidad de contar con instancias frecuentes de intercambio permitió validar requisitos, discutir decisiones técnicas y contrastar el trabajo realizado con necesidades reales de la organización, lo que contribuyó a orientar el desarrollo de la solución y obtener el producto esperado.

A su vez, el intercambio con especialistas en UX/UI, arquitectura e infraestructura constituyó uno de los aprendizajes más relevantes del proyecto. La interacción con profesionales experimentados permitió complementar los conocimientos académicos con criterios utilizados en entornos productivos, aportando una visión más amplia sobre la toma de decisiones técnicas y de diseño.

La experiencia también permitió identificar oportunidades de mejora para este tipo de iniciativas de colaboración entre la industria y la educación. Si bien la relación con Pyxis fue en general muy positiva, se considera que futuros proyectos podrían beneficiarse de una mayor definición inicial de algunos aspectos de la dinámica de trabajo, como la disponibilidad esperada de los referentes de la empresa, los accesos a las herramientas y recursos necesarios, y posibles instancias de nivelación sobre tecnologías y procesos. Una mayor formalización

de estos elementos podría facilitar la integración de los estudiantes al entorno de trabajo y mejorar la fluidez del desarrollo.

Finalmente, la experiencia permitió consolidar conocimientos prácticos mediante el uso de tecnologías y herramientas actuales de desarrollo e infraestructura en el contexto de un proyecto desarrollado en conjunto con una empresa. Esto complementó la formación académica al permitir la aplicación de conceptos vistos a lo largo de la carrera en un entorno con restricciones, decisiones y dinámicas propias de la industria. Asimismo, la herramienta desarrollada establece una base sólida sobre la cual podrían incorporarse nuevas funcionalidades y mejoras en futuras versiones.

9.2. Trabajo a futuro

Si bien el sistema desarrollado logró cumplir con los objetivos definidos para ambas liberaciones, durante las distintas instancias de prueba, uso interno y retroalimentación con Pyxis surgieron varias funcionalidades y mejoras que podrían incorporarse en futuras versiones de la herramienta.

Una de las funcionalidades que podría incorporarse a futuro es la posibilidad de eliminar proyectos del sistema. Actualmente, los proyectos siguen existiendo dentro del sistema incluso cuando ya no se encuentran activos o fueron finalizados. Incorporar esta funcionalidad permitiría mejorar la administración general de la plataforma y facilitar el manejo de proyectos obsoletos o creados únicamente con fines de prueba. En caso de implementarse, sería necesario analizar cuidadosamente cómo gestionar la información histórica asociada al proyecto, especialmente las tareas, métricas y registros de esfuerzo.

De forma similar, otra funcionalidad que podría incorporarse es la eliminación de usuarios del sistema. Actualmente, los usuarios permanecen registrados en la plataforma incluso cuando dejan de participar en proyectos o ya no la utilizan. No existe la posibilidad de removerlos del sistema. Poder gestionar correctamente el ciclo de vida de los usuarios permitiría mantener una administración más limpia y ordenada de la herramienta. Nuevamente, esto implicaría analizar el impacto sobre los datos históricos y las relaciones existentes con proyectos, tareas y registros de esfuerzo.

Otro aspecto identificado como posible mejora es la incorporación del concepto de *baseline*. Actualmente, la herramienta trabaja sobre la planificación vigente del proyecto, pero no conserva una línea base formal contra la cual comparar los cambios posteriores en la planificación. Incorporar esta funcionalidad permitiría almacenar una versión inicial aprobada del proyecto y luego comparar el avance real contra dicha planificación original, facilitando un análisis más preciso de desviaciones y cambios en el alcance o esfuerzo estimado.

También surgió la posibilidad de extender algunas métricas para que contemplen información hasta la fecha y hora actual, en lugar de calcularse únicamente hasta el último domingo. Actualmente, la mayoría de las métricas y gráficos trabajan con cortes semanales completos para simplificar los cálculos y mantener consistencia temporal. Sin embargo, en ciertos escenarios podría resultar útil

contar con métricas más cercanas al estado actual del proyecto, especialmente para realizar seguimientos diarios o detectar desvíos de forma más temprana.

Por otra parte, durante el análisis de funcionalidades presentes en Process Dashboard, como se mencionó en la sección 6.7 del capítulo de Proceso de construcción, se identificó como posible evolución de la herramienta la incorporación de una estructura jerárquica de tareas y subtareas (WBS). Actualmente, en la herramienta desarrollada las tareas representan la unidad mínima de planificación y seguimiento. Sin embargo, Process Dashboard permite organizar el trabajo mediante distintos niveles jerárquicos, donde una tarea principal puede contener múltiples subtareas asociadas.

Incorporar una funcionalidad similar permitiría descomponer actividades grandes o complejas en unidades de trabajo más pequeñas y específicas, facilitando una planificación más detallada y un seguimiento más preciso del avance real del proyecto. Además, esta estructura podría mejorar la organización del trabajo dentro de proyectos de mayor tamaño y permitir una representación más cercana a cómo suelen estructurarse las tareas en entornos reales de desarrollo de software.

Finalmente, otra mejora relacionada con la escalabilidad de la plataforma es la incorporación de paginado en la vista de tareas. En la versión actual, todas las tareas del proyecto se cargan simultáneamente, lo cual funciona correctamente para proyectos pequeños o medianos. Sin embargo, en proyectos con una gran cantidad de tareas, esto podría impactar tanto en el rendimiento como en la experiencia de usuario. Incorporar paginado o mecanismos de carga incremental permitiría mejorar la eficiencia y escalabilidad de esta sección de la aplicación.

En conjunto, estas mejoras representan posibles líneas de evolución futura para la herramienta, orientadas tanto a ampliar sus funcionalidades como a mejorar su rendimiento, mantenibilidad y experiencia de uso.

Referencias

- Humphrey, W. S. (2005). *Psp: A self-improvement process for software engineers*. Addison-Wesley.
- Humphrey, W. S. (2006). *Tsp: Coaching development teams*. Addison-Wesley Professional.
- IEEE Computer Society. (2024). *Guide to the software engineering body of knowledge (swebok guide)* (4.^a ed.). IEEE Computer Society.
- Jacobson, I., Lawson, H., Ng, P.-W., McMahon, P. E., y Goedicke, M. (2019). *The essentials of modern software engineering: Free the practices from the method prisons!* Association for Computing Machinery and Morgan & Claypool Publishers.
- Lovable. (2026). *Lovable*. <https://lovable.dev/>.
- Nielsen, J. (2014). *Response times: The 3 important limits*. <https://www.nngroup.com/articles/response-times-3-important-limits/>.
- OpenAI. (2026). *Chatgpt (gpt-5)*. <https://chatgpt.com>. ([Generative artificial intelligence tool])
- Project Management Institute. (2021). *A guide to the project management body of knowledge (pmbok guide)* (7.^a ed.). Project Management Institute.
- Scrum.org. (s.f.). *What is scrum?* <https://www.scrum.org/resources/what-scrum-module>.
- Sommerville, I. (2016). *Software engineering* (10.^a ed.). Pearson.
- The Standish Group. (2022). *Chaos report 2022*.
- Tuma Solutions, L. (s.f.). *Process dashboard*. <https://processdash.com>.

Anexo A

Casos de uso de la primera liberación

A continuación, se muestran los casos de usos creados para la primera liberación.

Cada caso de uso incluye: nombre, objetivo, actores involucrados, descripción general, precondiciones, flujo principal, flujos alternativos y postcondiciones.

A.1. Caso de uso de inicio de sesión

Nombre Inicio sesión con Microsoft 365.
Objetivo Iniciar sesión en el sistema.
Actores Usuario, sistema Microsoft 365.
Descripción Este caso de uso consiste en iniciar sesión en el sistema a través de Microsoft 365.
Precondiciones - Microsoft 365 se encuentra funcionando - El usuario no está logueado en el sistema
Flujo principal 1. Usuario: Ingresa a la URL correspondiente al Login. 2. Usuario: Clickea el botón correspondiente para iniciar sesión a través de Microsoft. 3. Sistema: Redirige al usuario al sitio de Microsoft 365. 4. Microsoft 365: Solicita al usuario sus credenciales. 5. Usuario: Ingresa sus credenciales en Microsoft 365. 6. Microsoft 365: Redirige al usuario al sistema. 7. Sistema: Completa el inicio de sesión del usuario, llevándolo a la página principal.
Flujo alternativo 5.A. El usuario no tiene usuario de Microsoft 365 asociado a Pyxis 5.A.1. Microsoft 365: Muestra un mensaje de error y finaliza sin éxito el inicio de sesión 6.A. El usuario no ingresa correctamente sus credenciales 6.A.1. Microsoft 365: Solicita nuevamente las credenciales. 6.A.2. Vuelve al paso 5.
Postcondiciones - El usuario queda con su sesión iniciada en el sistema, pudiendo ver la información asociada al mismo. - Si es el primer inicio de sesión del usuario, se guardan sus datos en el sistema.

A.2. Caso de uso de cierre de sesión

Nombre Cerrar sesión.
Objetivo Cerrar sesión del sistema.
Actores Usuario, sistema Microsoft 365
Descripción Este caso de uso consiste en cerrar sesión del sistema.
Precondiciones - Microsoft 365 se encuentra funcionando - El usuario está logueado en el sistema
Flujo principal 1. Usuario: Clickea el botón ubicado en la sidebar para cerrar sesión. 2. Sistema: Se redirige al cierre de sesión dentro de Microsoft 365 para expirar la sesión 3. Usuario: Confirma el cierre de sesión dentro de Microsoft 365 4. Sistema: Redirige al usuario a la página de inicio de sesión
Flujo alternativo 3.A: El usuario decide no cerrar sesión en Microsoft 365 3.A.1: Sistema: Redirige nuevamente a la pantalla de inicio
Postcondiciones - El sistema cerró la sesión del usuario y expiró el token asociado a la sesión de Microsoft 365. - El usuario fue redirigido a la pantalla de inicio de sesión

A.3. Caso de uso de crear proyecto

Nombre Crear proyecto
Objetivo Crear un nuevo proyecto en el sistema
Actores Usuario
Descripción Este caso de uso consiste en crear un nuevo proyecto en el sistema.
Precondiciones - El usuario inició sesión previamente en el sistema - El usuario no cuenta con ningún proyecto asociado <i>prox iteraciones: considerar múltiples proyectos y diferentes roles de usuarios</i>
Flujo principal 1. Usuario: Se dirige a la página principal del sistema. 2. Sistema: Le indica al usuario que no cuenta con un proyecto y le da la posibilidad de crear uno ingresando un nombre, una descripción y la cantidad de horas por defecto que el usuario va a dedicarle al proyecto por semana. <i>prox iteraciones: ingresar integrantes del proyecto y sus horas correspondientes</i> 3. Usuario: Ingresa el nombre del proyecto, descripción (opcional), las horas a dedicar por semana y clickea en "Crear proyecto". 4. Sistema: Crea un proyecto nuevo en el sistema con los datos ingresados y como único integrante asigna al usuario.
Flujo alternativo 4.A. Ya existe en el sistema un proyecto con el nombre ingresado 4.A.1. Sistema: Muestra un mensaje de error indicando que ya existe un proyecto con dicho nombre. 4.A.2. Vuelve al paso 3. <i>prox iteraciones: mostrar mensajes de error cuando alguno de los integrantes no sea válido</i>
Postcondiciones - Se crea en el sistema un nuevo proyecto con los datos ingresados conformado únicamente por ese usuario.

A.4. Caso de uso de crear tarea

Nombre Crear tarea
Objetivo Crear una nueva tarea asociada a cierto proyecto en el sistema
Actores Usuario
Descripción Este caso de uso consiste en crear una nueva tarea asociada a un proyecto existente en el sistema.
Precondiciones - El usuario inició sesión previamente en el sistema - El usuario cuenta con un proyecto asociado <i>prox iteraciones: considerar múltiples proyectos y diferentes roles de usuarios</i>
Flujo principal 1. Usuario: Se dirige a la página principal del sistema. 2. Sistema: Lista las tareas asociadas a ese proyecto (en caso de existir) y le da la posibilidad al usuario de crear una nueva. <i>prox iteraciones: considerar elegir el proyecto donde crear la tarea</i> 3. Usuario: Clickea en el botón correspondiente a crear una nueva tarea 4. Sistema: Agrega una nueva fila en la tabla de tareas existentes, con los siguientes campos en blanco: título, tiempo estimado, usuario asignado y descripción. El estado se crea por defecto como "pendiente". 5. Usuario: Ingresa al menos los datos obligatorios 6. Sistema: Crea una nueva tarea en el sistema con los datos ingresados, asociada al proyecto en el cual se encuentra el usuario y se recalculan las métricas del proyecto.
Flujo alternativo 5.A. El usuario no ingresa el nombre de la tarea 5.A.1. Sistema: aguarda a que la nueva tarea tenga un nombre ingresado. 5.A.2. Vuelve al paso 5. 5.B. El usuario no desea crear una tarea nueva 5.B.1. Usuario: Clickea en el botón correspondiente para eliminar la nueva fila creada. 5.B.2. Sistema: Elimina la fila agregada recientemente.
Postcondiciones - Se crea en el sistema una nueva tarea con los datos ingresados, asociada al proyecto en el cual se encuentra el usuario. - El sistema asigna fecha de inicio y fin estimada según la posición de la tarea, el usuario asignado y su cantidad de horas semanales.

A.5. Caso de uso de eliminar tarea

Nombre Eliminar tarea
Objetivo Eliminar del sistema una tarea existente asociada a un proyecto.
Actores Usuario
Descripción Este caso de uso consiste en eliminar una tarea existente asociada a un proyecto del sistema.
Precondiciones - El usuario inició sesión previamente en el sistema - El usuario cuenta con un proyecto asociado - Existen tareas creadas en dicho proyecto - La tarea a eliminar no se puede encontrar en estado "Completada" <i>prox iteraciones: considerar múltiples proyectos y diferentes roles de usuarios</i>
Flujo principal 1. Usuario: Se dirige a la página principal del sistema. 2. Sistema: Lista las tareas asociadas a ese proyecto. <i>prox iteraciones: considerar elegir el proyecto donde eliminar la tarea</i> 3. Usuario: Localiza la tarea que desea eliminar y clickea en el botón correspondiente a eliminar dicha tarea. 4. Sistema: Solicita confirmación al usuario para eliminar dicha tarea. 5. Usuario: Confirma que desea eliminar la tarea. 6. Sistema: Elimina la tarea, reordena las restantes si es necesario y se recalculan las métricas del proyecto.
Flujo alternativo 5.A Usuario rechaza eliminar la tarea 5.A.1 Vuelve al paso 2.
Postcondiciones - Se elimina del sistema la tarea seleccionada.

A.6. Caso de uso de reordenar tarea

Nombre Reordenar tareas
Objetivo Reordenar las tareas de un proyecto.
Actores Usuario
Descripción Este caso de uso consiste en cambiar la posición relativa de una tarea respecto a otras del mismo proyecto, clickeando sobre la misma y desplazándola con el mouse.
Precondiciones - El usuario inició sesión previamente en el sistema - El usuario cuenta con un proyecto asociado - Existen al menos dos tareas creadas en dicho proyecto. <i>prox iteraciones: considerar múltiples proyectos y diferentes roles de usuarios</i>
Flujo principal 1. Usuario: Se dirige a la página principal del sistema. 2. Sistema: Lista las tareas asociadas a ese proyecto. <i>prox iteraciones: considerar elegir el proyecto donde reordenar las tarea</i> 3. Usuario: Localiza la tarea que desea reubicar, mantiene clickeado sobre el icono que está sobre la izquierda de la fila, la desplaza hacia su nueva posición para reubicarla y suelta el click. 4. Sistema: Modifica las posiciones de las tareas afectadas y se recalculan las métricas del proyecto
Flujo alternativo 3.A El usuario quiere ubicar la tarea antes de una tarea ya finalizada 3.A.1 El sistema alerta que el movimiento no puede ser realizado. 3.A.2 Vuelve al paso 2.
Postcondiciones - Se modifican las posiciones de las tareas afectadas por la reordenación.

A.7. Caso de uso de editar horas semanales de un usuario

Nombre Editar horas semanales de un usuario
Objetivo Editar las horas de un usuario para un proyecto seleccionado en una semana en particular.
Actores Usuario
Descripción Este caso de uso consiste en editar las horas que un usuario le va a dedicar al proyecto en una semana en específico, previamente establecidas al indicar la cantidad de horas por defecto.
Precondiciones - El usuario inició sesión previamente en el sistema - El usuario cuenta con un proyecto asociado <i>prox iteraciones: considerar múltiples proyectos y diferentes roles de usuarios</i>
Flujo principal 1. Usuario: Se dirige a la pantalla de Schedule 2. Sistema: Muestra la tabla de las horas semanales de los usuarios que pertenecen al proyecto 3. Usuario: Selecciona la semana que quiere editar y setea la cantidad de horas para esa semana 4. Sistema: Se guarda la cantidad de horas seleccionadas para dicha semana y actualiza las métricas del proyecto
Postcondiciones - Se modifican las horas para dicha semana para el usuario en el proyecto seleccionado

Anexo B

Cálculos para las métricas mostradas en la página de Dashboard

A continuación, se detalla el razonamiento necesario para calcular cada métrica, incluyendo algunos valores intermedios requeridos para su obtención.

Cabe aclarar que la página de Dashboard cuenta con una vista grupal y una vista individual. Para las distintas métricas que se detallan a continuación existe una métrica equivalente que es utilizada para la vista individual. Dicha vista solo toma el calendario y las tareas asignadas al usuario que inició sesión.

Por otro lado, las tareas y horas del calendario no asignadas a ningún usuario solo son tomados en cuenta si se trata de la vista de proyecto y no de la vista individual.

B.1. Métricas totales

B.1.1. Valor ganado (EV) acumulado a cierta fecha

```
earned_value(date):  
    tasks = todas las tareas del proyecto  
    finished_tasks = tareas cuyo finished_at es previo a la semana en la que se encuentra date  
    total_planned_time = suma de horas estimadas de todas las tasks del proyecto  
    total_planned_time_finished_tasks = suma de horas estimadas de las finished_tasks  
    return (total_planned_time_finished_tasks * 100) / total_planned_time
```

B.1.2. Valor planificado (PV) acumulado a cierta fecha

```
planned_value(date):  
    tasks = todas las tareas del proyecto;  
    total_planned_time = tiempo total planificado para todas las tasks;
```

```

users = todos los usuarios pertenecientes al proyecto;
planned_time_for_planned_finished_tasks = 0;

for user in users:
    planned_calendar_time = tiempo calendario planificado hasta la semana
    anterior a date para user;
    assigned_tasks = todas las tasks asignadas a user;
    planned_finished_tasks = subconjunto máximo de assigned_tasks
    (empezando desde el comienzo y de forma secuencial) tal que el tiempo
    planificado total sea menor o igual a planned_calendar_time;
    planned_time_for_planned_finished_tasks =
    planned_time_for_planned_finished_tasks + suma del tiempo planificado de
    las planned_finished_tasks;

unassigned_planned_calendar_time = tiempo calendario no asignado hasta la semana
anterior a date;
unassigned_tasks = todas las tasks sin asignar;
planned_finished_tasks = subconjunto máximo de unassigned_tasks
(empezando desde el comienzo y de forma secuencial) tal que el tiempo
planificado total sea menor o igual a unassigned_planned_calendar_time ;
planned_time_for_planned_finished_tasks =
planned_time_for_planned_finished_tasks + suma del tiempo planificado de
las planned_finished_tasks ;

return (planned_time_for_planned_finished_tasks * 100) / total_planned_time ;

```

B.1.3. Tiempo calendario planificado acumulado a cierta fecha

```

planned_calendar_time(date):
    users = usuarios pertenecientes al proyecto;
    effort_from_users = suma de las horas totales que, según el schedule, cada
    usuario debería haber dedicado hasta la semana anterior a date inclusive;
    return effort_from_users;

```

B.1.4. Tiempo calendario real acumulado a cierta fecha

```

real_effort(date):
    tasks = tareas del proyecto creadas en una fecha anterior a date;
    effort_from_tasks = suma de las horas totales que realmente fueron dedicadas
    a tasks hasta la semana anterior a date inclusive;
    return effort_from_tasks;

```

B.1.5. Estimación total para tareas finalizadas

```

estimated_hours_for_finished_tasks(date):

```

```

finished_tasks = tareas del proyecto que fueron finalizadas en una fecha anterior a date;
estimated_hours_for_finished_tasks = suma de horas estimadas para finished_tasks;
return estimated_hours_for_finished_tasks;

```

Aclaración: para la tabla de métricas totales se tienen en cuenta sólo las tareas que fueron finalizadas antes de la semana actual

B.1.6. Esfuerzo real total en tareas finalizadas

```

real_effort_for_finished_tasks(date):
    finished_tasks = tareas del proyecto que fueron finalizadas antes de la semana
    correspondiente a date;
    real_effort_for_finished_tasks = suma de horas dedicadas a las finished_tasks;
    return real_effort_for_finished_tasks;

```

Aclaración: para la tabla de métricas totales se tienen en cuenta sólo las tareas que fueron finalizadas antes de la semana actual

B.2. Métricas de la última semana

Pasemos ahora a describir los cálculos realizados para obtener las métricas que corresponden a la última semana (es decir, a la semana anterior a la actual)

B.2.1. Valor ganado de la última semana

```

last_week_earned_value(date):
    tasks = todas las tareas del proyecto;
    last_week_finished_tasks = tareas cuyo finished_at es dentro de la semana previa a date;
    total_planned_time = suma de horas estimadas de las tasks;
    total_planned_time_finished_tasks = suma de horas estimadas de las last_week_finished_tasks;
    return (total_planned_time_finished_tasks * 100) / total_planned_time;

```

B.2.2. Valor planificado de la última semana

```

last_week_planned_value(date):
    tasks = todas las tareas del proyecto;
    total_planned_time = tiempo total planificado para todas las tasks;
    users = todos los usuarios pertenecientes al proyecto;
    planned_time_for_planned_finished_tasks = 0;
    for user in users:
        planned_calendar_time = tiempo calendario planificado en la semana
        anterior a date para user;
        assigned_pending_tasks = todas las tasks asignadas a user, que aún no
        habían sido finalizadas al comenzar la semana correspondiente a date ;
        planned_finished_tasks = subconjunto máximo de assigned_pending_tasks
        (empezando desde el comienzo y de forma secuencial) tal que el tiempo

```

```

planificado total sea menor o igual a planned_calendar_time ;
planned_time_for_planned_finished_tasks =
planned_time_for_planned_finished_tasks + suma del tiempo planificado
de las planned_finished_tasks ;

unassigned_planned_calendar_time = tiempo calendario no asignado para la
semana anterior a date
unassigned_pending_tasks = todas las tasks sin asignar, que aún no habían
sido finalizadas al comenzar la semana correspondiente a date ;
planned_finished_tasks = subconjunto máximo de unassigned_pending_tasks
(empezando desde el comienzo y de forma secuencial) tal que el tiempo
planificado total sea menor o igual a unassigned_planned_calendar_time ;
planned_time_for_planned_finished_tasks =
planned_time_for_planned_finished_tasks + suma del tiempo planificado de
las planned_finished_tasks ;
return (planned_time_for_planned_finished_tasks * 100) / total_planned_time;

```

B.2.3. Esfuerzo estimado en la última semana

```

estimated_effort_in_last_week(date):
    users = usuarios pertenecientes al proyecto;
    effort_from_users_in_last_week = suma de las horas totales que, según el
    schedule, se deberían haber dedicado al proyecto en la semana previa a date;
    return effort_from_users_in_last_week;

```

B.2.4. Esfuerzo real en la última semana

```

real_effort_in_last_week(date):
    tasks = tareas del proyecto creadas en una fecha anterior a date;
    effort_from_tasks_in_last_week = suma de las horas totales que realmente
    fueron dedicadas a tasks en la semana previa a date;
    return effort_from_tasks;

```

B.2.5. Estimación para tareas finalizadas en la última semana

```

estimated_hours_for_finished_tasks_in_last_week(date):
    finished_tasks_in_last_week = tareas cuya fecha de finalización fue en la
    semana anterior a date
    estimated_hours_for_finished_tasks = suma de horas estimadas para
    finished_tasks_in_last_week
    return estimated_hours_for_finished_tasks

```


B.2.6. Esfuerzo real en tareas finalizadas en la última semana

```
real_effort_for_finished_tasks_in_last_week(date):  
    finished_tasks_in_last_week = tareas cuya fecha de finalización fue en la semana  
    anterior a date  
    real_effort_for_finished_tasks = suma de horas dedicadas a las finished_tasks_in_last_week  
    return real_effort_for_finished_tasks
```

B.3. Fechas previstas de fin

B.3.1. Fecha prevista de fin del proyecto según replanificación

Fecha en la que se estima que se completarán todas las tareas del proyecto, partiendo del tiempo planificado para cada tarea y del tiempo calendario planificado, según un cálculo de replanificación. Este cálculo parte de la suposición de que, aunque hasta el momento las tareas finalizadas se hayan sobreestimado o subestimado, o que en el pasado se haya dedicado una cantidad de tiempo mayor o menor a la indicada en el calendario, a partir de ahora sí se cumplirá con el resto del plan.

A continuación se detalla como es la implementación de esta fecha prevista de fin.

Para las tareas en progreso debemos determinar cuánto tiempo restante les queda. En principio este valor es igual a `task.estimated_time - task.real_time`. Al hacer esto nos podemos enfrentar a dos situaciones: o que a la tarea ya se le haya dedicado más tiempo del estimado (llamémosles *overspent tasks*), o que la tarea no haya sido comenzada aún o que haya sido comenzada y su tiempo dedicado sea menor al estimado (llamémosles *underspent tasks*).

Para *overspent tasks*, para obtener cuánto tiempo restante les queda se hace una lógica de “pedir prestado” tiempo. Esto se hace bajo la suposición de que a partir de ahora se va a seguir el plan (ya que en esto se basa el *Replan date*). Si hay tareas que se subestimaron (*real > estimated*), significa que hay otras que se sobreestimaron (*estimated > time*), por lo que les podemos sacar tiempo estimado. La lógica de tomar prestado se hace quitándoles pequeñas fracciones de tiempo a cada tarea *underspent*, de forma proporcionalmente igual.

El tiempo que se toma prestado para las *overspent tasks* no es de forma tal que el tiempo estimado sea igual al real (ya que sería poco realista asumir que todas las *overspent tasks* están finalizadas), sino que se asume que las tareas *overspent* están un 90 % terminadas. Si no es posible prestar tiempo de esta forma (porque el tiempo que les sobra a las *underspent* no alcanza), se recurre a una lógica de que el tiempo restante de cada tarea es `max(0, task.estimated_time - task.real_time)`.

```
project_replanned_end_date():  
    // 1) Clasificar tareas
```

```

remaining_tasks = todas las tareas del proyecto no finalizadas
underspent = tareas no comenzadas + tareas en progreso con tiempo
estimado >= tiempo real
overspent = tareas en progreso con tiempo real > tiempo estimado

// 2) Calcular cuánto hay que "cubrir" para cada overspent
overspent_needed = []
for task in overspent:
    required_total = task.real_time / 0.9 // para dejarla 90% hecha
    borrow_needed = max(0, required_total - task.estimated_time)
    overspent_needed.push({
        task: task,
        borrow_needed: borrow_needed,
        borrow_remaining: borrow_needed
    })

// 3) Calcular tiempo disponible en underspent
total_available = suma( t.estimated_time - t.real_time para t en underspent )

// 4) Calcular tiempo total requerido
total_required = suma( entry.borrow_needed para entry en overspent_needed )

// 5) Calcular esfuerzo restante (Replan)
if total_available < total_required:
    // Fallback:
    // no es posible hacer borrowing, se usa estimación directa de trabajo
    // restante
    para cada task:
        task.remaining_raw =
            max(0, task.estimated_time - task.real_time)
else:
    // Inicialmente copiamos los valores actuales
    para cada task:
        task.adjusted_estimated_time = task.estimated_time

    // Borrowing proporcional (fracciones)
    for task in underspent:
        task_time_available = task.estimated_time - task.real_time

        if task_time_available <= 0:
            continue

        fraction = task_time_available / total_available

    // asignar a overspent según esta fracción
    for entry in overspent_needed:

```

```

        if entry.borrow_remaining <= 0:
            continue

        allocated = entry.borrow_needed * fraction
        allocated = min(allocated, entry.borrow_remaining)

        task.adjusted_estimated_time -= allocated
        entry.task.adjusted_estimated_time += allocated

        entry.borrow_remaining -= allocated

    // Una vez finalizada la redistribución,
    // calculamos el esfuerzo restante de cada tarea.
    // remaining_raw representa siempre el trabajo pendiente de la tarea,
    // independientemente de si se utilizó borrowing o fallback.
    for task in remaining_tasks:
        task.remaining_raw =
            task.adjusted_estimated_time - task.real_time

    // 6) Calcular cuánto queda de la semana actual
    // (según fecha actual, tomando 24h lun-vie)

    // 7) Recorrer semanas restando su capacidad hasta que remaining llegue a 0
    // para obtener el Replan Date por usuario

    // A nivel de proyecto, el Replan Date corresponde al último Replan Date de los usuarios

```

B.3.2. Fecha prevista de fin del proyecto según pronóstico

Fecha en la que se estima que se completarán todas las tareas del proyecto, partiendo del tiempo planificado para cada tarea y del tiempo calendario planificado, según un cálculo de pronóstico. Este cálculo considera si las tareas finalizadas fueron subestimadas o sobreestimadas, y si en el pasado se trabajaron más o menos horas que las planificadas en el calendario. A partir de estos datos, el sistema asume que ambos desvíos se mantendrán constantes en el futuro, respecto a lo planificado.

A continuación se detalla como es la implementación de esta fecha prevista de fin.

```

project_forecast_end_date():
    // 1) Clasificar tareas
    remaining_tasks = todas las tareas no finalizadas
    underspent = tareas no comenzadas + tareas en progreso con estimated > real
    overspent   = tareas en progreso con real > estimated

    // 2) Calcular cuanto hay que "cubrir" para cada overspent

```

```

overspent_needed = []
for task in overspent:
    required_total = task.real_time / 0.9 // para dejarla 90% hecha
    borrow_needed = max(0, required_total - task.estimated_time)
    overspent_needed.push({
        task: task,
        borrow_needed: borrow_needed,
        borrow_remaining: borrow_needed
    })

// 3) Calcular tiempo disponible en underspent
total_available = suma( t.estimated - t.real para t en underspent )

// 4) Calcular tiempo total requerido
total_required = suma( borrow_needed para cada overspent_needed )

// 5) Calcular el esfuerzo restante de las tareas
if total_available < total_required:
    // Fallback:
    para cada task:
        task.remaining_raw =
            max(0, task.estimated_time - task.real_time)
else:
    // Inicialmente copiamos los valores actuales
    para cada task:
        task.adjusted_estimated_time = task.estimated_time

    // Borrowing proporcional (fracciones)
    for task in underspent:
        task_time_available = task.estimated_time - task.real_time

        if task_time_available <= 0:
            continue

        fraction = task_time_available / total_available

        // asignar a overspent según esta fracción
        for entry in overspent_needed:
            if entry.borrow_remaining <= 0:
                continue

            allocated = entry.borrow_needed * fraction
            allocated = min(allocated, entry.borrow_remaining)

            task.adjusted_estimated_time -= allocated
            entry.task.adjusted_estimated_time += allocated

```

```

        entry.borrow_remaining -= allocated

    // Una vez finalizada la redistribución,
    // calculamos el esfuerzo restante
    for task in remaining_tasks:
        task.remaining_raw = task.adjusted_estimated_time - task.real_time

    // 6) Aplicamos CPI al esfuerzo restante
    CPI = sum(estimated_time for completed tasks) / sum(real_time for completed tasks)

    for task in remaining_tasks:
        task.remaining_forecast = task.remaining_raw / CPI

    // 7) Todas las tareas tienen ahora un remaining_forecast basado en el CPI

    // 8) Calcular cuánto queda de la semana actual
    // (según fecha actual, tomando 24h lun-vie)

    // 9) Ahora, para cada semana del calendario, ajustar su tiempo restante
    // disponible en base a un ratio historico, teniendo en cuenta hasta la
    // semana anterior a la actual
    ratio = sum(tiempo planificado para cada semana) / (tiempo realmente
    dedicado en cada semana)

    // actualizamos las semanas del calendario cuya fecha sea en la semana actual o
    // posterior
    for week in calendar_week_for_user:
        calendar_week_for_user.remaining_time = calendar_week_for_user.weekly_hours / ratio

    // 10) Se recorren las semanas y las tareas restantes
    // descontando el remaining_forecast de cada tarea del remaining_time disponible
    // de las semanas.
    // Cuando una semana se agota se continúa con la siguiente.
    // La fecha en la que termina la última tarea determina
    // el Forecast Date del usuario.
    // A nivel de proyecto, el Forecast Date corresponde al último Forecast Date de los usuarios

```

B.4. Líneas mostradas en la gráfica

B.4.1. Línea de Valor Planificado

Utilizando lo descrito en la función `planned_value(date)` (sección B.1.2), se realiza un cálculo que determina, para cada cierre semanal, cuál debería ser el Valor Planificado según el calendario y las tareas definidas. Es decir, se obtiene el valor que se planifica haber alcanzado al final de cada semana. Dichos valores se representan en la gráfica en la fecha correspondiente al domingo de finalización de cada una de ellas.

Esta línea se muestra desde el comienzo del proyecto hasta el domingo correspondiente a la última semana existente en el calendario.

B.4.2. Línea de Valor Ganado

Utilizando lo descrito en la función `earned_value(date)` (sección B.1.1), se realiza un cálculo que determina, para cada cierre semanal, cuál era el Valor Ganado acumulado hasta ese momento. Para ello, se consideran únicamente las tareas que ya se encontraban finalizadas en esa fecha, tomando como referencia el 100 % correspondiente al total de tareas actualmente registradas. De este modo, se obtiene el valor efectivamente ganado al final de cada semana. Dichos valores se representan en la gráfica en la fecha correspondiente al domingo de finalización de cada una de ellas.

Esta línea se muestra desde el comienzo del proyecto hasta el domingo correspondiente a la última semana existente en el calendario.

B.4.3. Línea de Valor Ganado replanificado

Utilizando lo descrito en la función `project_replanned_end_date(date)` (sección B.3.1), se realiza un cálculo que determina, para cada cierre semanal, cuál debería ser el Valor Ganado acumulado hasta ese momento, según el cálculo de replanificación. Dichos valores se representan en la gráfica en la fecha correspondiente al domingo de finalización de cada semana.

Esta línea se muestra desde el último domingo hasta la última semana existente en el calendario.

B.4.4. Línea de Valor Ganado pronosticado

Utilizando lo descrito en la función `project_forecast_end_date(date)` (sección B.3.2), se realiza un cálculo que determina, para cada cierre semanal, cuál debería ser el Valor Ganado acumulado hasta ese momento, según el cálculo de pronóstico. Dichos valores se representan en la gráfica en la fecha correspondiente al domingo de finalización de cada semana.

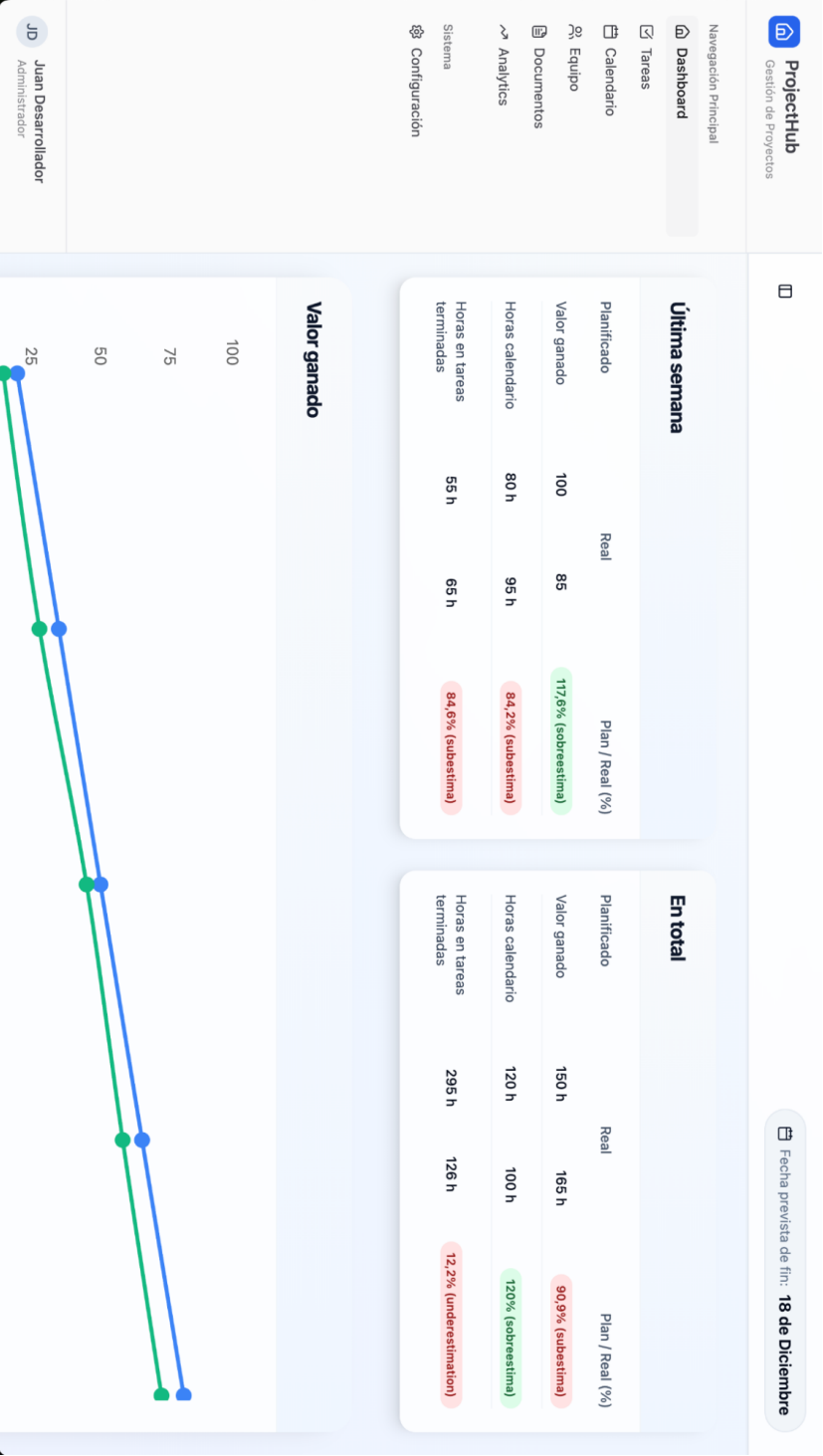
Esta línea se muestra desde el último domingo hasta la última semana existente en el calendario.

Anexo C

Diseños de UI

A continuación se muestran los diseños de UI para la primera y segunda liberación. Los diseños fueron creados por una experta en UX/UI de Pyxis utilizando la herramienta Lovable ([Lovable, 2026](#)).

C.1. Primera liberación



JD

Juan Desarrollador

Administrador

Figura C.1: Diseño de la pantalla de Dashboard.

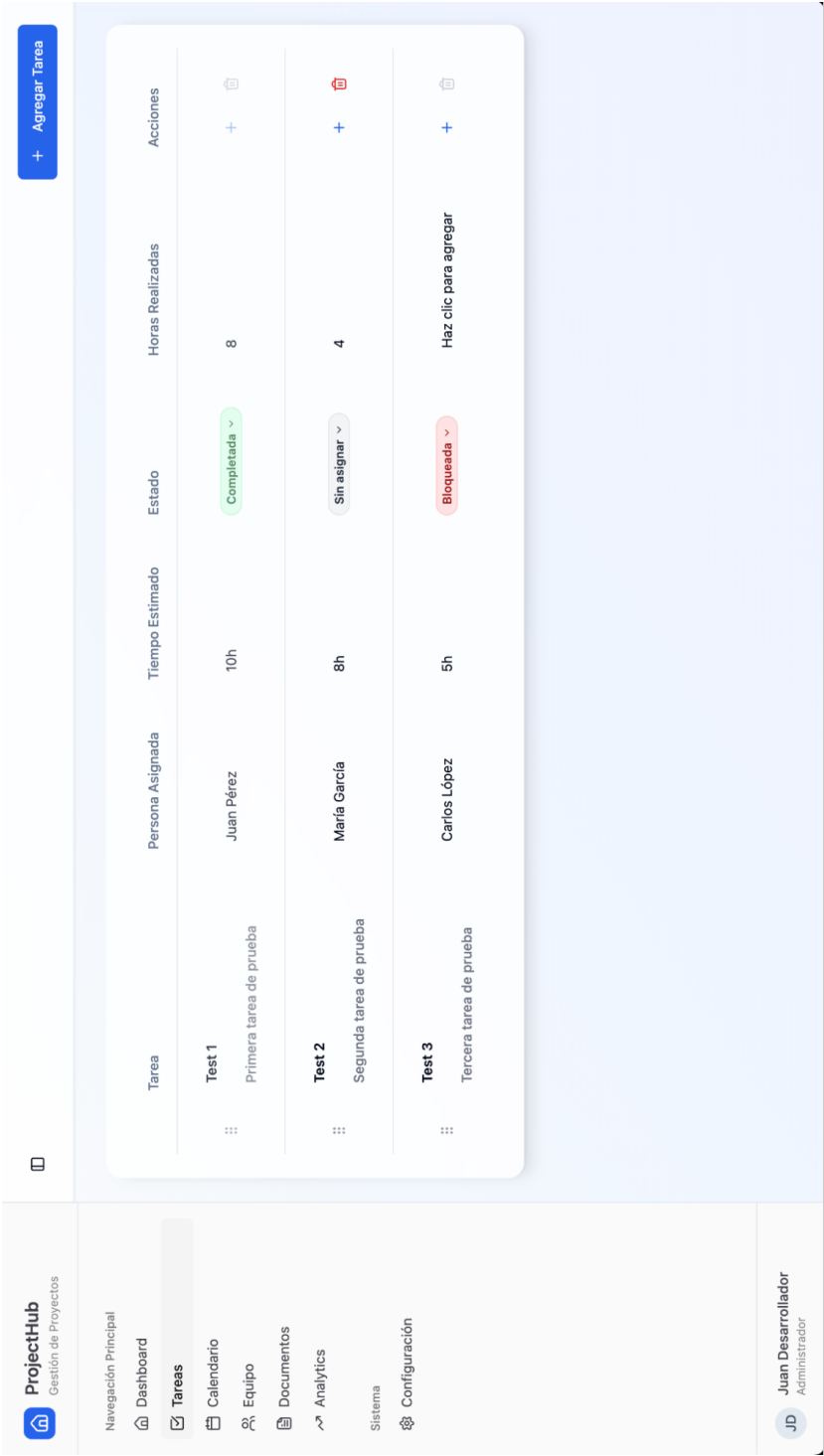


Figura C.2: Diseño de la pantalla de Tareas.

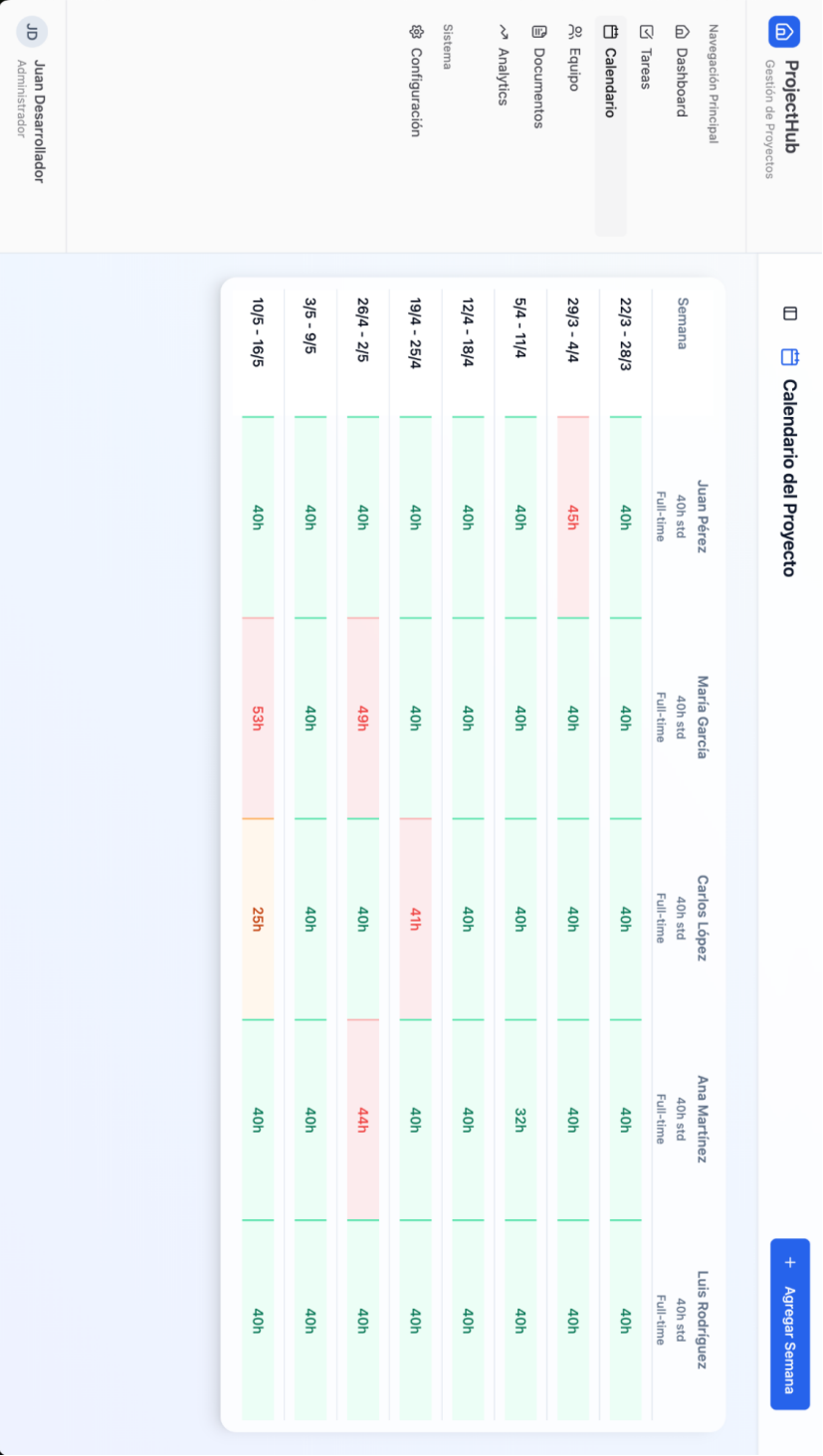


Figura C.3: Diseño de la pantalla de Calendario.

C.2. Segunda liberación

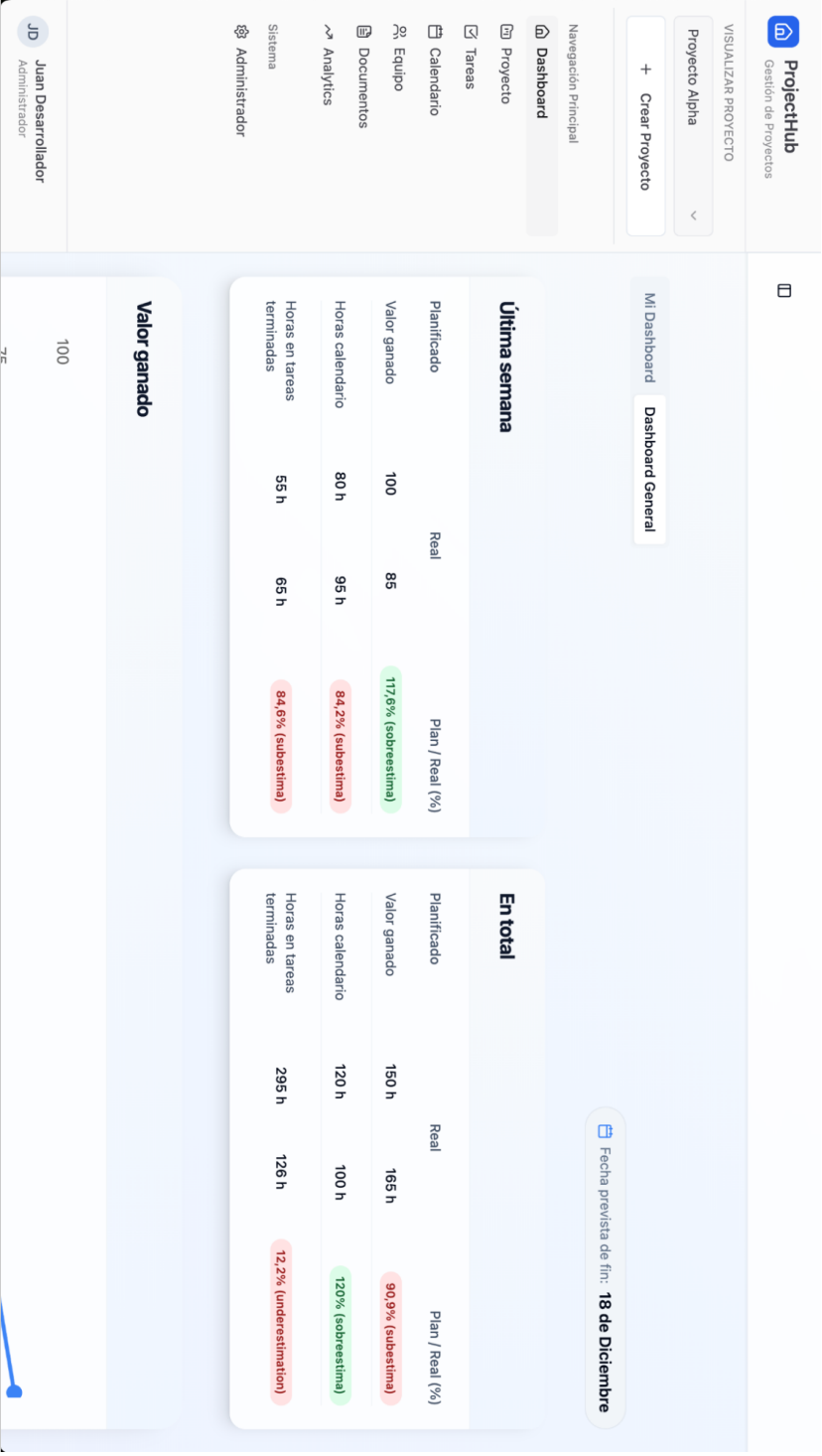


Figura C.4: Diseño final de la pantalla de Dashboard.

ProjectHub

Gestión de Proyectos

Projecto Alpha

+ Crear Proyecto

Navegación Principal

Dashboard

Proyecto

Tareas

Calendario

Equipo

Documentos

Analytics

Sistema

Administrador

Juan Desarrollador

Administrador

Mis Tareas

Tareas Generales

+ Agregar Tarea

⚠ Las horas planificadas exceden la capacidad disponible del calendario. Por favor, ajusta las tareas o distribúyelas en otro mes para continuar sin interrupciones.

Tarea	Persona Asignada	Tiempo Estimado	Estado	Horas Realizadas	Acciones
Test 1 Primera tarea de prueba	Juan Pérez	10h	Completada	8	
Test 2 Segunda tarea de prueba	María García	8h	Sin asignar	4	
Test 3 Tercera tarea de prueba	Carlos López	5h	Bloqueada	Haz clic para agregar	
Revisar documentación Revisar y actualizar la documentación del proyecto	Juan Pérez	6h	Sin asignar	3	
Implementar login					

Figura C.5: Diseño final de la pantalla de Tareas.

127



Figura C.6: Diseño final de la pantalla de Calendario.

ProjectHub

Gestión de Proyectos

Projecto Alpha

+ Crear Proyecto

Navegación Principal

Dashboard

Proyecto

Tareas

Calendario

Equipo

Documentos

Analytics

Sistema

Administrador

Juan Desarrollador

Administrador

Editar Proyecto

Ver todos los proyectos

Información del Proyecto

Equipo

Proyecto Alpha

Modifica la información del proyecto

Nombre

Proyecto Alpha

Fecha de Inicio

25/11/2025

Descripción

Describe los objetivos y alcance del proyecto

Responsable

Maria Pia Delgado

Dedicación Semanal (horas)

30

Eliminar proyecto

Actualizar proyecto

Figura C.7: Diseño de la sección de Información del Proyecto.

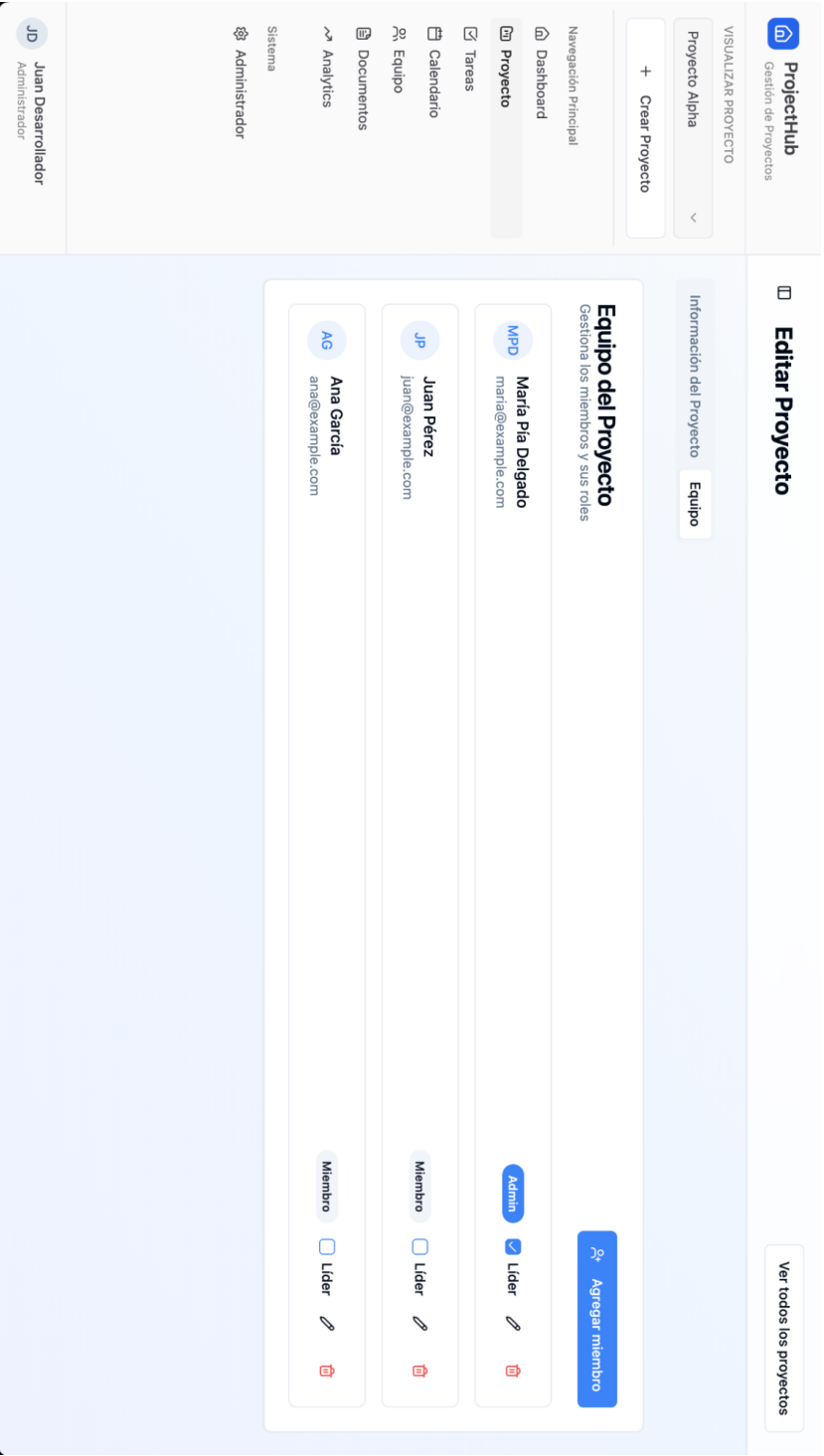


Figura C.8: Diseño de la sección de Equipo del Proyecto.

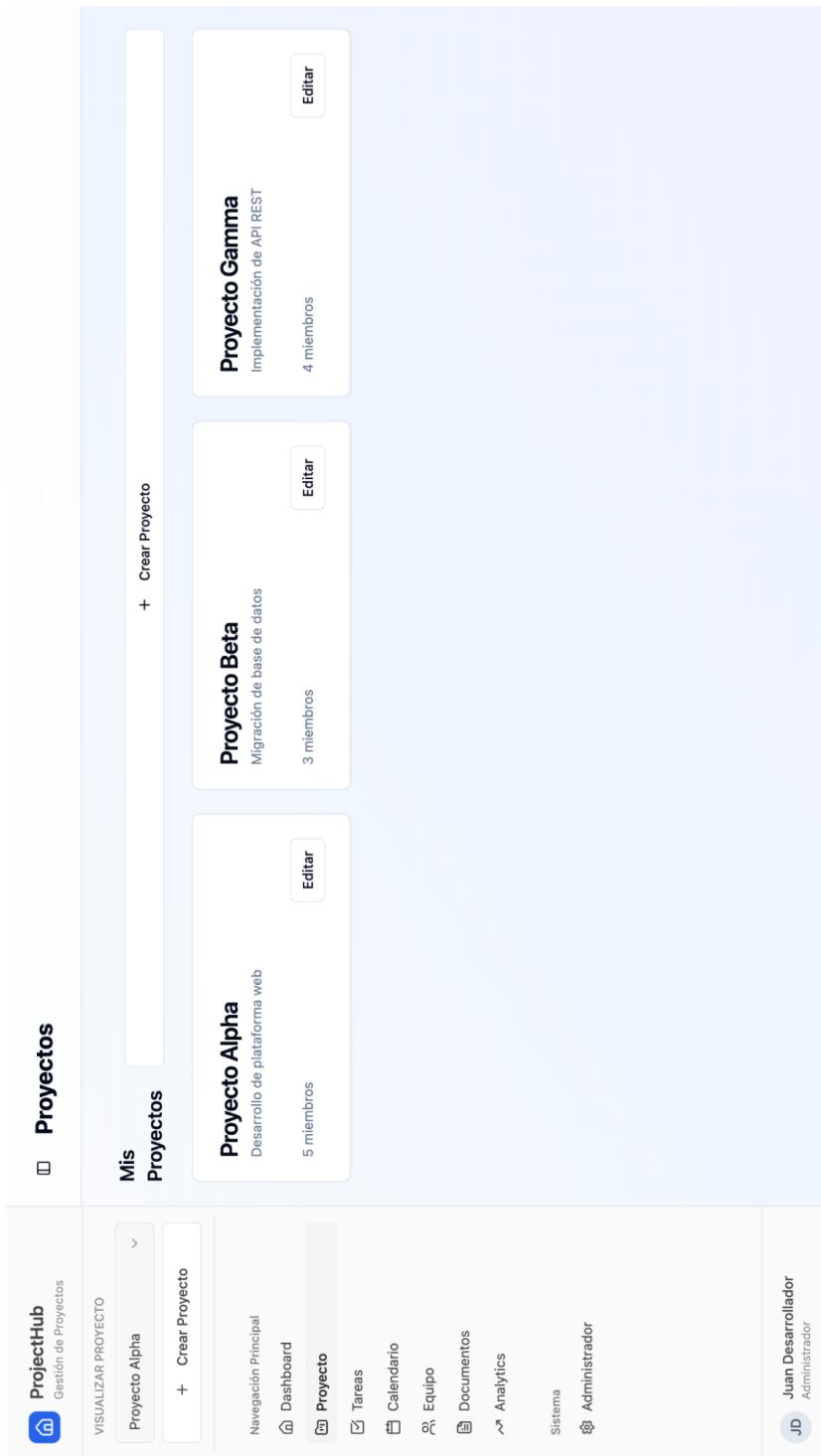


Figura C.9: Diseño de la pantalla de Proyectos.

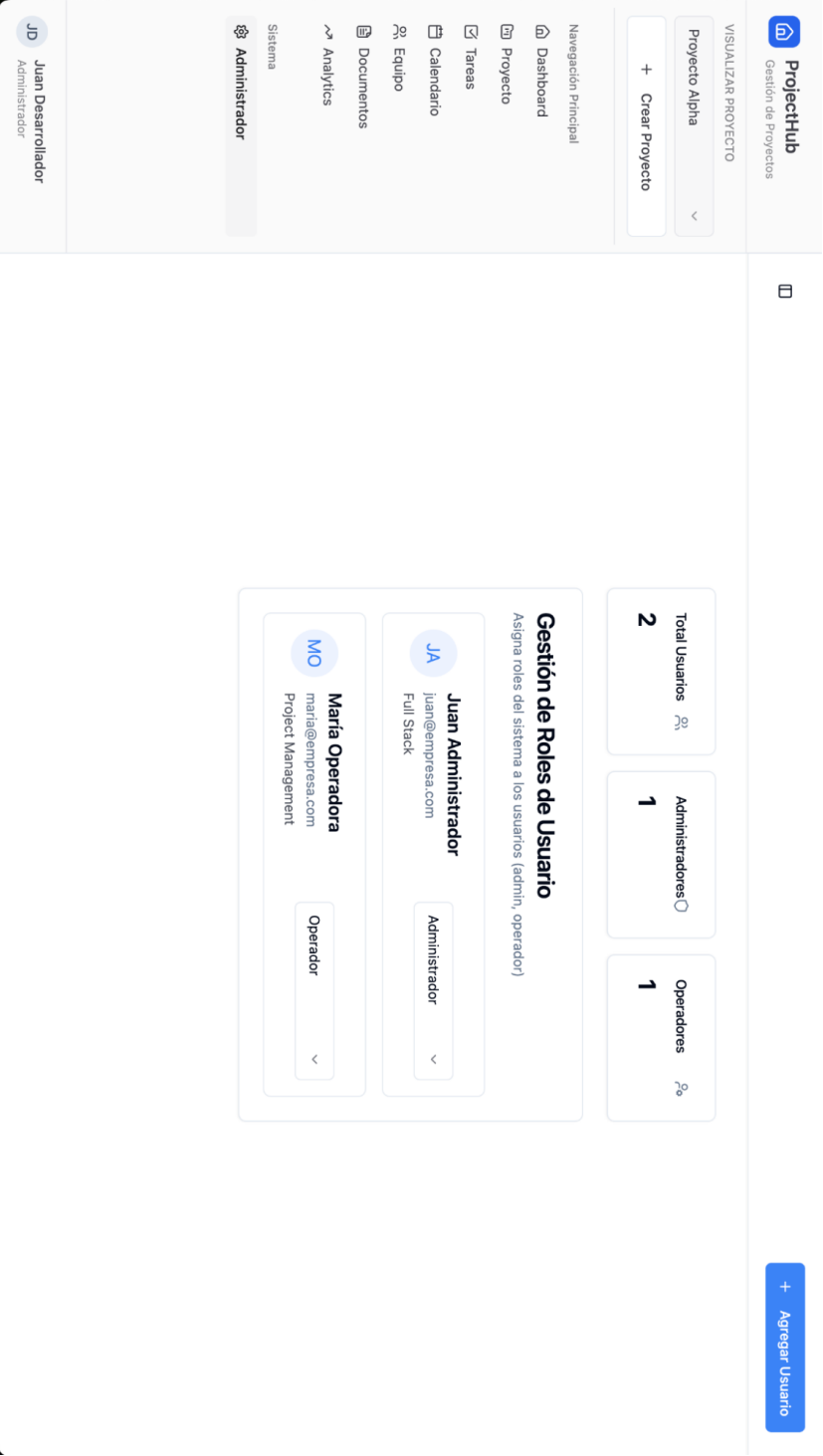


Figura C.10: Diseño de la pantalla de Administrador.

Anexo D

Manual de usuario

D.1. Requisitos previos

Antes de poder utilizar la aplicación, es necesario tener acceso a internet y un usuario de Microsoft 365 dentro de la organización Pyxis.

D.2. Acceso a la plataforma

Paso 1: Ingresar al link <https://proyectoev.pyxis.tech/> en donde se mostrará la pantalla de inicio de sesión como se observa en la figura D.1.



Figura D.1: Pantalla de inicio de sesión.

Paso 2: Hacer clic en el botón “Continuar con Microsoft 365” e ingresar con tu usuario de Pyxis en la pantalla que se muestra en la figura D.2.

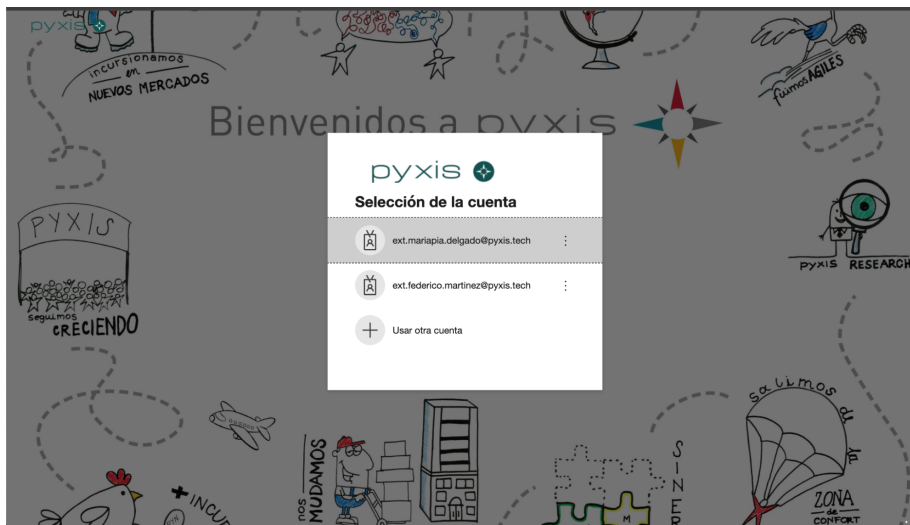


Figura D.2: Pantalla de inicio de sesión con Microsoft 365.

D.3. Funcionalidades por roles

Las funcionalidades disponibles para un usuario dentro de la plataforma dependen del rol que tenga en la aplicación (administrador, operador o integrante de equipo), del proyecto que tenga seleccionado y del rol que desempeñe dentro de ese proyecto (líder o integrante de proyecto).

El rol de administrador del sistema tiene como principal función la gestión de usuarios.

El rol de operador tiene como principal función la administración de proyectos.

También existe el rol de líder dentro de cada proyecto. Este rol tiene como principal función la administración del proyecto que lidera y la gestión de las tareas relacionadas al mismo. Cuando un administrador u operador crea un nuevo proyecto, debe definir quiénes serán los líderes del mismo (pudiendo asignarse a sí mismo).

El proyecto puede seleccionarse o cambiarse directamente desde la aplicación (a través de una lista desplegable en la barra lateral). Las diferentes páginas de la aplicación se actualizan según el proyecto seleccionado, mostrando la información correspondiente.

D.4. Crear nuevo proyecto

La funcionalidad de crear un nuevo proyecto está disponible solo para usuarios con roles de administrador u operador del sistema.

Para crear un proyecto nuevo se debe hacer clic en el botón “Crear proyecto” en la barra lateral. Al hacer clic se abre un formulario, el cual se muestra en la figura D.3, para completar los datos del proyecto a crear.

Los campos a completar son: nombre, fecha de inicio, descripción (opcional), líder asignado al proyecto y la dedicación semanal del mismo al proyecto.

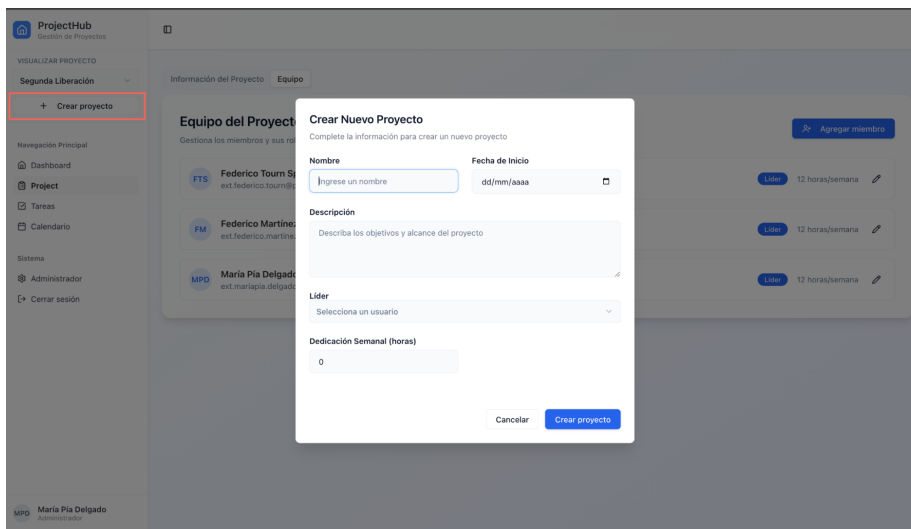
The image shows a web application interface for 'ProjectHub'. On the left is a sidebar with navigation links: 'Segunda Liberación', '+ Crear proyecto' (highlighted with a red box), 'Dashboard', 'Project', 'Tareas', 'Calendario', 'Sistema', 'Administrador', and 'Cerrar sesión'. The main area displays the 'Crear Nuevo Proyecto' modal form. The form has the following fields: 'Nombre' (text input with placeholder 'Ingrese un nombre'), 'Fecha de Inicio' (date picker with placeholder 'dd/mm/aaaa'), 'Descripción' (text area with placeholder 'Describe los objetivos y alcance del proyecto'), 'Líder' (dropdown menu with placeholder 'Selecciona un usuario'), and 'Dedicación Semanal (horas)' (text input with value '0'). At the bottom of the form are 'Cancelar' and 'Crear proyecto' buttons. The background shows a blurred view of the 'Equipo del Proyecto' section with user cards for Federico Tour, Federico Martine, and Maria Pia Delgado.

Figura D.3: Formulario para crear un nuevo proyecto.

D.5. Página de proyecto

Todos los usuarios asignados a un proyecto pueden ver la información relacionada a ese proyecto. Esta información incluye: nombre, fecha de inicio, descripción e integrantes del proyecto, con su respectivo rol y dedicación semanal dentro del proyecto.

La funcionalidad de editar proyecto se presenta para aquellos integrantes que sean líderes del mismo.

En la figura D.4 se muestra la pantalla del proyecto seleccionado y su información.

Todos los campos de un proyecto pueden ser editados. Además, se pueden agregar usuarios al proyecto (los usuarios deben preexistir en la plataforma), modificar integrantes pertenecientes al mismo (modificar sus dedicaciones semanales estándar y cambiarles el rol entre líder e integrante), o eliminar integrantes

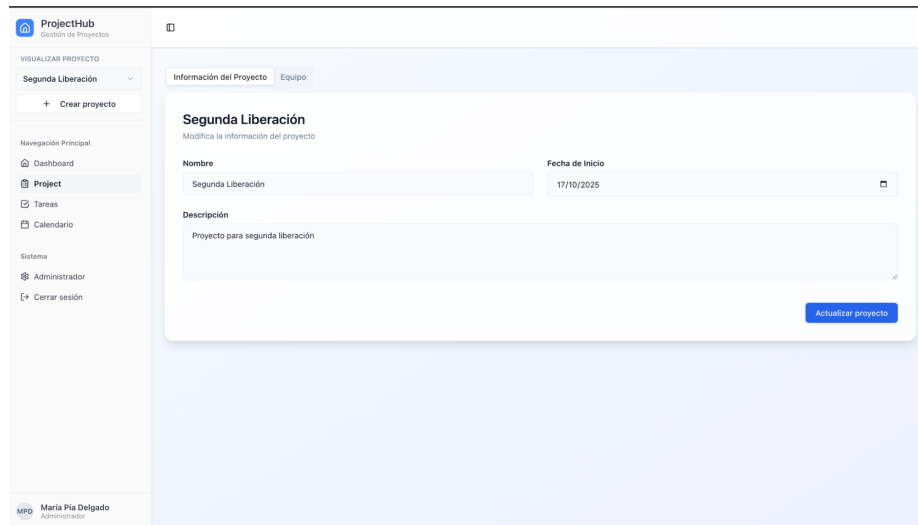


Figura D.4: Pantalla con la información de un proyecto.

del proyecto.

En la figura D.5 se muestra la pantalla con la información del equipo del proyecto seleccionado y en la figura D.6 se muestra el formulario para agregar o editar algún miembro del equipo.

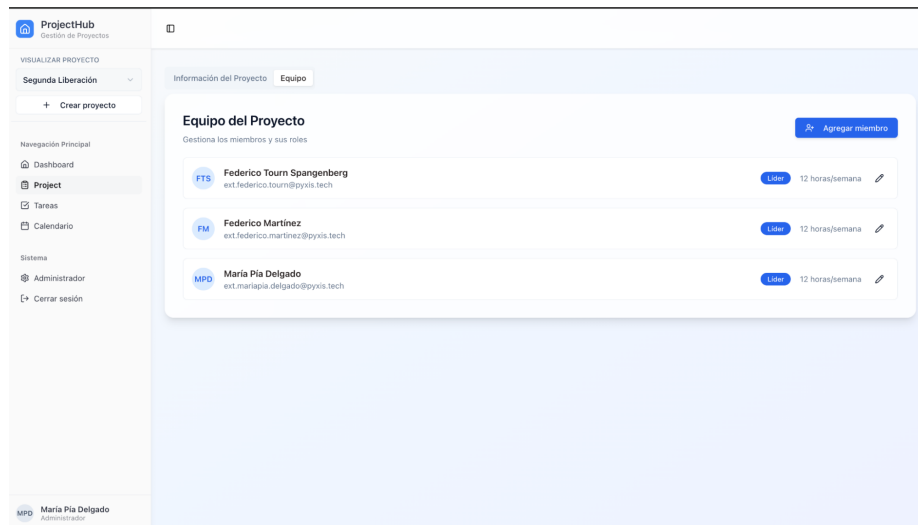


Figura D.5: Pantalla con la información del equipo de un proyecto.

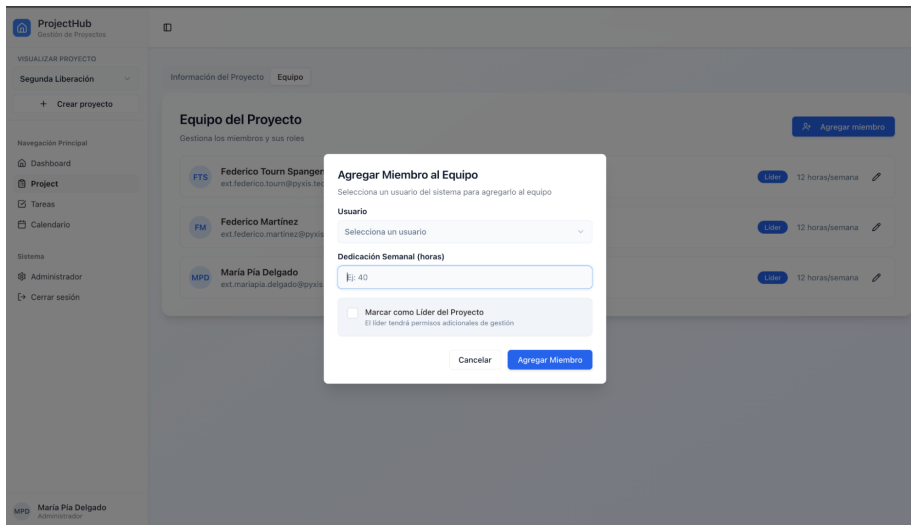


Figura D.6: Formulario para agregar un miembro al equipo de un proyecto.

Observaciones:

1. El nombre del proyecto y la fecha de inicio son campos obligatorios dentro de la información del proyecto
2. La dedicación semanal de cada integrante del equipo es un campo obligatorio dentro de la información del equipo.
3. Para el caso de que exista esfuerzo real dedicado al proyecto, el sistema no permite editar la fecha de inicio a una fecha posterior a la fecha de ese esfuerzo. Asimismo, la fecha de inicio del proyecto no puede ser posterior a ningún schedule existente.

D.6. Página de Dashboard

La página del Dashboard se puede visualizar en la figura D.7 tiene dos vistas: Individual y de Proyecto.

En esta página se visualizan métricas relacionadas con el proyecto seleccionado.

Tanto para la vista individual como para la vista del proyecto, en la tabla de la última semana y la total, las métricas que el sistema muestra son:

- Valor Ganado Planificado
- Valor Ganado Real

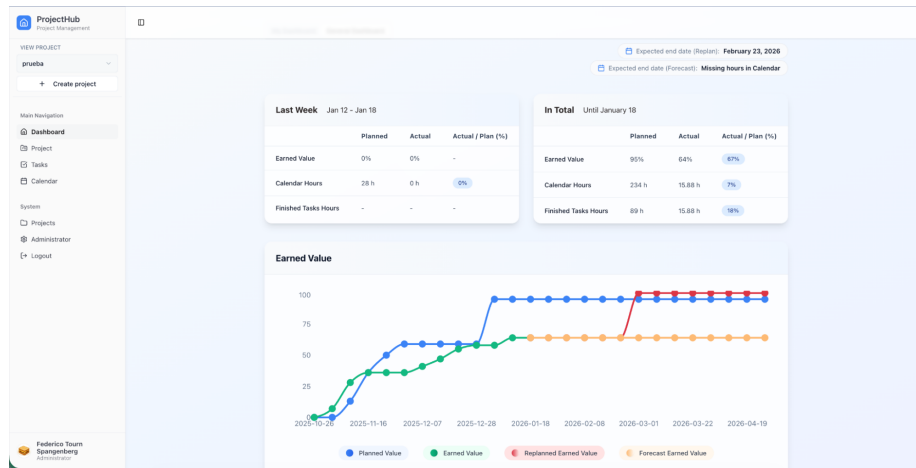


Figura D.7: Pantalla del Dashboard.

- Qué porcentaje representa el Valor Ganado Real respecto al Valor Ganado Planificado
- Horas Calendario Planificadas
- Horas Calendario Reales
- Qué porcentaje representan las Horas Calendario Reales respecto a las Horas Calendario Planificadas
- Horas Planificadas en las tareas finalizadas
- Horas Reales en las tareas finalizadas
- Qué porcentaje representan las Horas Reales en las tareas finalizadas, respecto a las Horas Planificadas en las tareas finalizadas

El significado de los colores asignados a los valores del porcentaje Actual / Plan (%) en las distintas tablas son: Verde: significa que el porcentaje está entre 90 % y 110 % Azul: representa un porcentaje menor a 90 % Amarillo: representa un porcentaje mayor a 110 %

D.6.1. Vista de las métricas del proyecto

En la vista del proyecto se ven las métricas asociadas al proyecto seleccionado teniendo en cuenta todos los integrantes del mismo y sus calendarios.

En esta vista se incluye:

- La fecha prevista de fin del proyecto, según un cálculo de replanificación. Este cálculo parte de la suposición de que, aunque hasta el momento las tareas finalizadas se hayan sobreestimado o subestimado, o que en el pasado se haya dedicado una cantidad de tiempo mayor o menor a la indicada en el calendario, a partir de ahora sí se cumplirá con el resto del plan. Si las horas del calendario no alcanzan para cubrir las tareas según este cálculo, el sistema muestra un mensaje indicando que no hay horas suficientes en el calendario.
- La fecha prevista de fin del proyecto, mediante un pronóstico que tiene en cuenta los desvíos históricos. En particular, considera si las tareas finalizadas fueron subestimadas o sobreestimadas, y si en el pasado se trabajaron más o menos horas que las planificadas en el calendario. A partir de estos datos, el sistema asume que ambos desvíos se mantendrán constantes respecto a lo planificado. Al igual que en el cálculo anterior, si las horas disponibles en el calendario no son suficientes para cubrir las tareas según este pronóstico, el sistema muestra un mensaje indicando que no hay horas suficientes.
- Una tabla que refleja el avance del proyecto desde su **fecha de inicio hasta el último domingo**.
- Una tabla que refleja el avance del proyecto en la **última semana** (desde el lunes hasta el domingo de la semana anterior a la semana actual).
- Una gráfica de valor ganado a lo largo del proyecto, desde su fecha de comienzo hasta la última semana del proyecto en el calendario. Dicha gráfica muestra una línea de valor ganado planificado, otra línea de valor ganado real y dos líneas de proyecciones a futuro, cada una basándose en los cálculos de replanificación y de pronóstico mencionados anteriormente.

Observación:

- Si aún no comenzó el proyecto, pero existen tareas con horas planificadas y horas en el calendario, esto se refleja en la gráfica.

D.6.2. Vista de las métricas individuales

En la vista individual se ven las métricas asociadas al proyecto seleccionado teniendo en cuenta solo el usuario logueado y su propio schedule.

En esta vista se incluye:

- Una fecha prevista de fin según el cálculo de replanificación mencionado anteriormente. En este caso se tienen en cuenta sólo las tareas asignadas al integrante en cuestión, y sus horas en el calendario.
- Una fecha prevista de fin según el cálculo de pronóstico mencionado anteriormente. En este caso se tienen en cuenta sólo las tareas asignadas al integrante en cuestión, y sus horas en el calendario.

- Una tabla que refleje el avance del integrante en el proyecto desde su **fecha de inicio hasta el último domingo**.
- Una tabla que refleje el avance del integrante en el proyecto en la **última semana** (desde el lunes hasta el domingo de la semana anterior a la semana actual).
- Una gráfica de valor ganado para el integrante a lo largo del proyecto, desde su fecha de comienzo hasta la última semana del integrante en el calendario. Dicha gráfica muestra una línea de valor ganado planificado, otra línea de valor ganado real y dos líneas de proyecciones a futuro, cada una basándose en los cálculos de replanificación y de pronóstico mencionados anteriormente.

Observación:

- Si aún no comenzó el proyecto, pero existen tareas con horas planificadas y horas en el calendario, esto se refleja en la gráfica.

D.7. Página de Tareas

La página de Tareas se puede visualizar en la figura D.8, la misma tiene dos vistas: Individual y de Proyecto.

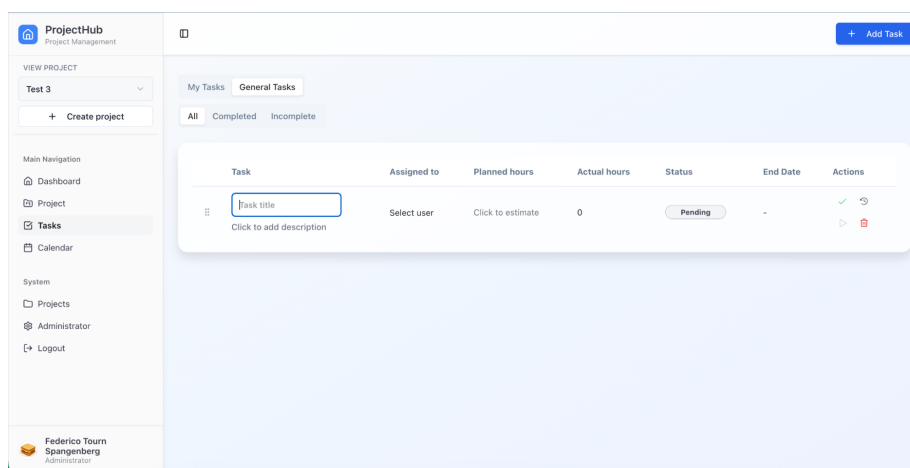


Figura D.8: Pantalla de Tareas.

En esta página se permite: crear tareas, editarlas, insertar horas realizadas (manualmente y a través de un temporizador), completarlas, visualizarlas y eliminarlas.

Las funcionalidades de la página de tareas estarán disponibles para todos los integrantes del proyecto, con algunas restricciones.

D.7.1. Vista de las tareas del proyecto

Los líderes del proyecto seleccionado pueden crear tareas y asignar todos los campos incluyendo el integrante asignado para esa tarea.

Los integrantes del equipo no pueden editar ninguna tarea en esta vista, pero sí pueden ver todos los campos disponibles de las tareas.

Si las horas planificadas en el calendario no alcanzan para cubrir las horas planificadas para las tareas asignadas al equipo el sistema muestra un mensaje de alerta.

La posición de las tareas en esta vista estará determinada por los siguientes criterios:

1. Primero se muestran las tareas completadas, ordenadas según su fecha de finalización.
2. Luego se muestran las tareas en estado “En progreso” o “Pendiente”, ordenadas según su posición.

Solo los líderes del proyecto pueden reordenar las tareas en la vista general.

D.7.2. Vista de las tareas individuales

Todos los integrantes pueden crear tareas en la vista individual. Se permite asignar todos los campos excepto por el integrante asignado que el sistema asigna automáticamente al usuario logueado.

Si las horas planificadas en el calendario no alcanzan para cubrir las horas planificadas para las tareas asignadas al integrante, el sistema muestra un mensaje de alerta.

D.7.3. Temporizador y horas realizadas

El sistema permite registrar el esfuerzo usando el temporizador o manualmente indicando la fecha de inicio y fin.

El sistema permite iniciar y parar el temporizador cuantas veces se quiera manualmente. En el caso de que la tarea se complete con el temporizador corriendo, el mismo se detiene automáticamente.

Si se desea cambiar de proyecto seleccionado y el temporizador está corriendo en una tarea, el sistema muestra un mensaje de alerta, y si el integrante de equipo confirma el cambio de proyecto, se pausa el temporizador.

Cuando se registra esfuerzo, se asumirá que el integrante asignado a la tarea es quien realizó dichas horas.

Cuando se completa una tarea el sistema le asigna la fecha de finalización como la fecha del último esfuerzo realizado.

Si se desea registrar esfuerzo en una tarea que ya está marcada como completada se debe marcar como no completada la tarea y agregar el esfuerzo.

No es posible registrar esfuerzo en una tarea, en una fecha posterior a la fecha actual (es decir, en una fecha en el futuro).

Observaciones:

- Una tarea es creada, y por lo tanto guardada, si se le asigna un título.
- Las tareas se crean desde el botón “Agregar tarea” o haciendo enter/tab desde el campo “Horas planeadas”.
- No se permite realizar horas ni completar tareas para un proyecto que todavía no comenzó.
- Es posible visualizar todos los registros de esfuerzo que haya asociados a una tarea.
- La fecha de finalización de cada tarea se puede editar. La nueva fecha de finalización debe ser posterior al último esfuerzo registrado y anterior a la fecha actual.
- El estado se actualiza automáticamente, el flujo de posibles estados es el siguiente: Cuando se crea la tarea se le asigna el estado “Pendiente”. Cuando se realizan horas se le asigna el estado “En progreso” (se puede visualizar en la figura D.9). Cuando se completa la tarea se le asigna el estado “Completado” (se puede visualizar en la figura D.10). Si se des-completa la tarea y tenía horas previamente realizadas entonces el estado vuelve a ser “En progreso”, si no, “Pendiente”.

AllCompletedIncomplete

Task	Assigned to	Planned hours	Actual hours	Status	End Date	Actions
Tarea de Prueba Descripción de la tarea de Prueba	Federico Tourn Spangenberg	4	1	In Progress	-	✓ ↻ ▶ ⛔
Time Logs	User	Start Date	End Date	Duration		
	Federico Tourn Spangenberg	19/01/2026 18:59	📅	19/01/2026 19:59	📅	01:00:00 ⛔
+ Add time log						

Figura D.9: Tarea en progreso.

AllCompletedIncomplete

Task	Assigned to	Planned hours	Actual hours	Status	End Date	Actions
Tarea de Prueba Descripción de la tarea de Prueba	Federico Tourn Spangenberg	4	1	Completed	19/01/2026	📅 ↺ ↻

Figura D.10: Tarea completada.

D.8. Página del Calendario

La página del Calendario se puede observar en la figura D.11. Esta página no tiene dos vistas separadas como las otras páginas.



Week	Federico Tourn Spangenberg 12h std	Federico Martínez 12h std	María Pía Delgado 12h std	Additional Hours
27/10 - 2/11	4h	8h	8h	0h
3/11 - 9/11	2h	2h	2h	0h
10/11 - 16/11	4h	8h	4h	0h
17/11 - 23/11	4h	8h	8h	0h
24/11 - 30/11	8h	8h	8h	0h
1/12 - 7/12	8h	8h	8h	0h
8/12 - 14/12	8h	8h	8h	0h
15/12 - 21/12	8h	8h	8h	0h
22/12 - 28/12	4h	6h	4h	0h
29/12 - 4/1	6h	6h	6h	0h
5/1 - 11/1	8h	8h	8h	0h

Figura D.11: Pantalla del Calendario.

Las funcionalidades de la página del Calendario estarán disponibles para todos los integrantes del proyecto con algunas restricciones.

Los líderes del proyecto seleccionados pueden ver todos los calendarios del equipo. También pueden editarlos.

Los integrantes del equipo sólo tendrán disponible para ver/editar su propio calendario.

Aparte de los calendarios de los integrantes, se puede agregar una instancia de calendario sin un usuario asociado dentro del proyecto. Las instancias de ese estilo serán para registrar, en cierta semana, el total de horas que dedicará uno o más integrantes del equipo que aún no fueron agregados al proyecto.

Cuando dichos usuarios sean agregados al proyecto, se debe modificar manualmente el calendario, quitando las horas de la columna de las horas adicionales sin asignar, y asignándolas al nuevo integrante.

Si las horas planificadas en el calendario no alcanzan para cubrir las horas planificadas para las tareas asignadas el sistema muestra un mensaje de alerta.

En esta página también es posible ver los calendarios existentes de los usuarios que fueron eliminados del proyecto. Si se desea modificarlos, el sistema muestra un mensaje de alerta.

Observaciones:

- Las semanas del calendario se agregan desde el botón “Agregar semana”, o haciendo tab/enter en la última semana en la tabla.

- La cantidad de horas por defecto para un integrante son las ingresadas en la dedicación semanal estándar del proyecto para el mismo.
- El sistema permite editar las dedicaciones semanales. Para el caso de las semanas anteriores o iguales a la actual el sistema pide confirmación antes de guardar las nuevas dedicaciones.
- Una semana puede ser eliminada únicamente si es la última semana registrada en el calendario y si aún no hay esfuerzo real registrado en la misma. También se puede eliminar una semana si es la primera semana del calendario y no hay esfuerzo realizado. En caso de que esto se cumpla, al eliminar la primera semana del calendario el sistema modifica automáticamente la fecha de comienzo del proyecto, y pasa a ser el lunes de la semana siguiente.

D.9. Página de Proyectos

La página de Proyectos es accesible para los usuarios con rol de administrador u operador del sistema.

En esta página, como se observa en la figura D.12, el sistema presenta el listado completo de proyectos existentes.

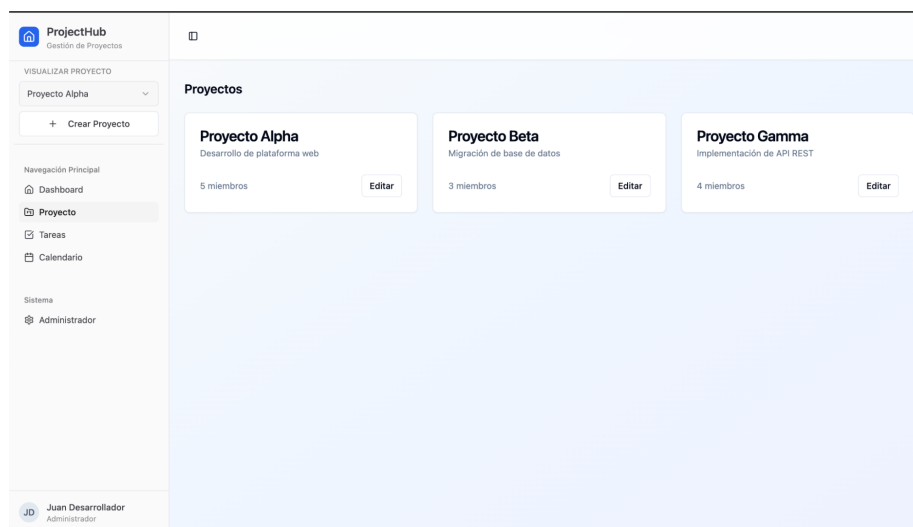


Figura D.12: Pantalla de Proyectos.

Además, el sistema permite a dichos usuarios editar la información de cada proyecto, incluyendo su nombre, descripción, fecha de inicio, y la dedicación semanal y el rol de los integrantes asignados. Además, se permite la incorporación

de usuarios previamente registrados en la plataforma, así como la eliminación de integrantes del proyecto.

Al eliminar un integrante del proyecto, se realizan las siguientes acciones:

- Eliminar todas las horas del calendario, a partir de la semana siguiente a la semana actual (las anteriores se mantienen creadas).
- Desasignar todas las tareas no completadas del integrante.

D.10. Página de administrador

En la página de administrador se pueden ver cuántos usuarios tiene la plataforma divididos en las siguientes secciones: Total de usuarios, Usuarios con rol de administrador y Usuarios con rol de operador.

Además, en esta página se puede gestionar los roles de los usuarios a nivel de sistema, donde un usuario administrador puede asignar a los demás usuarios como administrador, operador, o integrante de equipo.

La figura D.13 muestra esta página, la cual solo los administradores a nivel de sistema tienen acceso a esta página.

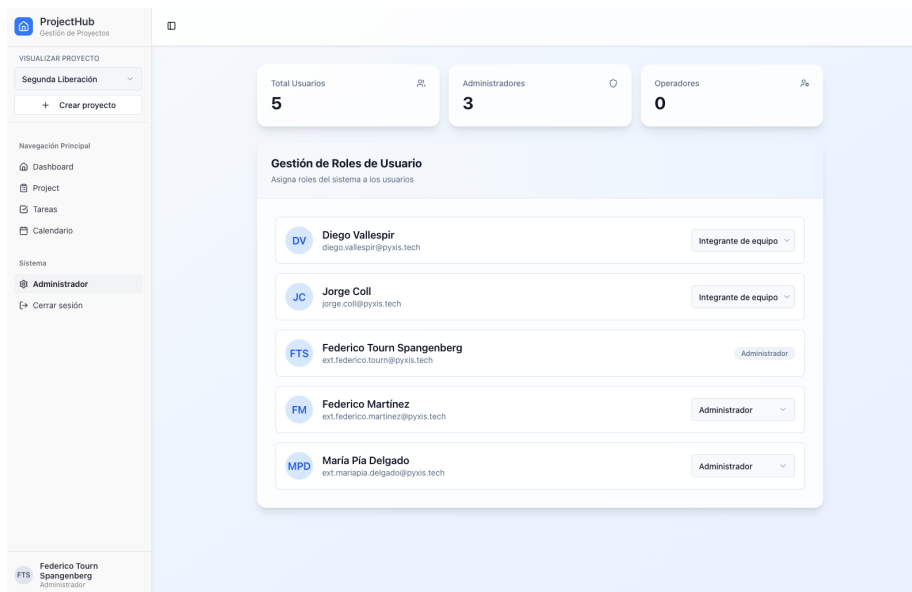


Figura D.13: Pantalla del Administrador.

Anexo E

Pruebas de sistema

A continuación se presentan las distintas pruebas de sistema realizadas por el equipo.

E.1. Caso de prueba: Inicio de sesión

Inicio de sesión	
Descripción	Usuario accede a la aplicación mediante inicio de sesión con Microsoft.
Datos de prueba	- Credenciales válidas - usuario dentro de la organización de Pyxis - Credenciales inválidas - usuario fuera del organización de Pyxis o usuario/contraseña incorrecto
Pasos a ejecutar	1. Navegar a la pantalla de inicio de sesión. 2. Hacer clic en "Iniciar sesión con Microsoft". 3. Completar autenticación con credenciales.
Resultado esperado	- Se redirige a /after-login y luego a la página principal con un usuario autenticado. - El token de sesión se guarda correctamente. - Se muestran mensajes de error (dentro de Microsoft) para los usuarios con credenciales inválidas. - Las traducciones en página de inicio de sesión se muestran correctamente. - Se muestra un mensaje de error cuando el servidor está caído.
Resultado obtenido	Flujo completado exitosamente.

E.2. Caso de prueba: Cierre de sesión

Cierre de sesión	
Descripción	Usuario autenticado cierra sesión.
Datos de prueba	- Usuario autenticado.
Pasos a ejecutar	1. Desde cualquier pantalla de la aplicación con sidebar, hacer clic en "Cerrar sesión".
Resultado esperado	- Se redirige a /login y se borra la sesión de next auth. - Verificar que el token se elimine y no se pueda acceder a rutas protegidas sin inicio de sesión.
Resultado obtenido	Flujo completado exitosamente.

E.3. Caso de prueba: Multisesión

Multisesión	
Descripción	Usuario accede a su cuenta en la aplicación desde varios navegadores
Datos de prueba	- Usuario con credenciales válidas
Pasos a ejecutar	1. Iniciar sesión en un navegador. 2. Iniciar sesión en otro navegador (o ventana de incógnito).
Resultado esperado	- Los datos que se muestran en ambas sesiones son correctos.
Resultado obtenido	Flujo completado exitosamente.

E.4. Caso de prueba: Vistas principales y Barra lateral

Vistas principales y Barra lateral	
Descripción	Usuario navega por la aplicación después de iniciar sesión, cambiando entre vistas personales y generales.
Datos de prueba	- Usuario autenticado - Proyecto válido
Pasos a ejecutar	1. Navegar a través de las distintas páginas disponibles. 2. Modificar el proyecto seleccionado. 3. Modificar el rol del usuario en el sistema.
Resultado esperado	- Los Administradores deben poder acceder a las siguientes páginas: Dashboard, Proyecto, Tareas, Calendario, Proyectos y Administrador. - Los Operadores deben poder acceder únicamente a las siguientes páginas: Dashboard, Proyecto, Tareas, Calendario y Proyectos. - Los Integrantes de equipo deben poder acceder únicamente a las siguientes páginas: Dashboard, Proyecto, Tareas y Calendario. - La barra lateral se muestra correctamente. - Las rutas en la url funcionan correctamente, y el idioma corresponde con lo indicado en la misma. - El contexto del proyecto se actualiza correctamente. - Los usuarios visualizan el selector de proyectos. - La foto del usuario y el rol a nivel de sistema del mismo se muestran correctamente.
Resultado obtenido	Flujo completado exitosamente.

E.5. Caso de prueba: Pantalla de Administrador

Pantalla de administrador	
Descripción	Usuario con rol de administrador accede al panel de gestión de usuarios.
Datos de prueba	- Usuario autenticado con rol de administrador. - Usuarios con roles de operador e integrante de proyectos.
Pasos a ejecutar	1. Seleccionar en la barra lateral la opción de administrador. 2. Ver lista de usuarios.
Resultado esperado	- Usuarios sin permisos no pueden acceder a esta página, ni siquiera a través de la url. - Los datos que se muestran son correctos (información de usuarios). - Se puede asignar roles a todos los usuarios, excepto al usuario que está autenticado.
Resultado obtenido	Flujo completado exitosamente.

E.6. Caso de prueba: Crear nuevo proyecto

Crear nuevo proyecto	
Descripción	Usuario crea un proyecto desde la interfaz.
Datos de prueba	- Usuario autenticado con rol de administrador u operador del sistema. - Datos válidos para un proyecto. - Datos inválidos para un proyecto - nombre de proyecto vacío o ya existente, dedicación semanal nula, negativa, no numérica, mayor a 168 horas.
Pasos a ejecutar	1. Clickear botón de "Crear proyecto". 2. Ingresar datos en los campos del formulario. 3. Clickear botón de "Crear proyecto".
Resultado esperado	- El botón solo aparece para usuarios con rol Operador o Administrador. - Al apretar el botón de cancelar no se guardan los datos ingresados en el formulario ni se crea el proyecto. - El proyecto creado aparece seleccionado en el visualizador de proyectos, solo si el usuario se asignó a sí mismo como líder del mismo. En caso contrario, el proyecto no es visible como opción en dicho visualizador. - El proyecto creado se visualiza en la página de proyectos. - Los distintos valores no son válidos para el campo correspondiente y se muestran mensajes de errores: nombre vacío, fecha de inicio vacía, no seleccionar líder, no asignar dedicación semanal al líder, asignar dedicación semanal negativa o 0.
Resultado obtenido	- El flujo fue ejecutado exitosamente en términos generales. - Durante la ejecución se detectaron defectos en la validación de la dedicación semanal, ya que inicialmente se permitía ingresar valores no numéricos o superiores al máximo esperado.

E.7. Caso de prueba: Pantalla de Proyecto

Pantalla de Proyecto	
Descripción	Usuario selecciona un proyecto y ve sus datos
Datos de prueba	- Usuario autenticado. - Proyecto existente.
Pasos a ejecutar	1. Seleccionar el proyecto deseado en la barra lateral. 2. Navegar a la página de Proyecto desde la barra lateral.
Resultado esperado	- Los datos que se muestran son correctos. - Se muestran ambas pestañas (Equipo e Información del proyecto). - Solo los usuarios con permisos pueden editar los campos y la edición funciona correctamente.
Resultado obtenido	- La dedicación semanal podía ser cualquier string, la dedicación semanal podía ser mayor a 168. - Si se editaba la dedicación semanal y se actualizaba el miembro del equipo se mostraba correctamente en la página del equipo pero si se volvía a abrir el formulario de actualizar miembro se mostraba la dedicación semanal NO actualizada.

E.8. Caso de prueba: Pantalla de Dashboard

Pantalla de Dashboard	
Descripción	Usuario selecciona un proyecto y ve sus métricas
Datos de prueba	- Usuario autenticado. - Proyecto existente.
Pasos a ejecutar	1. Seleccionar el proyecto deseado en la barra lateral. 2. Navegar a la página de Dashboard desde la barra lateral.
Resultado esperado	- Los datos se carguen correctamente en todos los campos (gráfica y tablas). - Se muestran ambas pestañas (general e individual). - Todos los textos tienen bien las traducciones.
Resultado obtenido	Flujo completado exitosamente.

E.9. Caso de prueba: Dashboard - Proyecto sin tareas

Dashboard - Proyecto sin tareas	
Descripción	Usuario selecciona un proyecto que no tiene ninguna tarea creada.
Datos de prueba	- Usuario autenticado. - Proyecto existente que no tenga tareas creadas.
Pasos a ejecutar	1. Seleccionar el proyecto en la barra lateral. 2. Navegar a la página de Dashboard.
Resultado esperado	- Verificar que se muestre un mensaje en las fechas previstas de fin: "Faltan tareas en el proyecto".
Resultado obtenido	Flujo completado exitosamente.

E.10. Caso de prueba: Dashboard - Proyecto finalizado

Dashboard - Proyecto finalizado	
Descripción	Usuario selecciona un proyecto finalizado.
Datos de prueba	- Usuario autenticado. - Proyecto existente que tenga: - Fecha de inicio en el pasado - Al menos una tarea - Todas las tareas completadas - Proyecto existente con una fecha de inicio en el futuro.
Pasos a ejecutar	1. Seleccionar el proyecto en la barra lateral. 2. Navegar a la página de Dashboard.
Resultado esperado	- Se debe mostrar el mensaje "Proyecto finalizado" en el lugar de las fechas previstas de fin. - Proyectos con fecha de inicio en el futuro no muestran ningún mensaje.
Resultado obtenido	Flujo completado exitosamente.

E.11. Caso de prueba: Dashboard - Métricas

A continuación se muestra la descripción y los pasos a seguir para el caso de prueba de las métricas en la pantalla de Dashboard. El mismo se separa en el caso de prueba de Métricas Individuales y Métricas de Proyecto.

Dashboard - Métricas	
Descripción	Verificar los correctos valores en los cálculos de las distintas métricas en la página de Dashboard
Datos de prueba	- Usuario autenticado. - Proyecto existente con varias tareas creadas.
Pasos a ejecutar	1. El proyecto creado debe haber empezado la semana anterior a la actual. 2. Crear 5 tareas, de las cuales tres están asignadas a mi usuario, una a otro usuario y la tarea restante sin asignar. 3. Estimar con 3, 2 y 5 horas las tareas asignadas a mi usuario, 4 horas para la tarea asignada al otro usuario, y 6 horas para la tarea sin asignar. 4. A la primera tarea, registrarle 9 horas de esfuerzo dedicadas en la semana anterior a la actual, y marcarla como finalizada. 5. En el calendario: a. asignarle a cada semana de mi usuario, a partir de la semana anterior a la actual: 2 horas, 3 horas, 1 hora, 3 horas y 3 horas b. al otro usuario, asignarle 2 horas a la semana anterior a la actual, esa misma cantidad a la semana actual, 3 horas a la semana posterior y 1 hora a la semana siguiente. c. En la columna de horas sin asignar, ingresar dos horas por semana a partir de la semana anterior a la actual y hasta la semana siguiente a la actual, y 3 horas a la semana posterior. Por último, asignar 1 hora a la semana siguiente.

E.11.1. Dashboard - Métricas Individuales

Dashboard - Métricas Individuales	
Resultado esperado	En la vista individual del dashboard debería verse: • Métricas totales: ◦ Valor planificado: 0% ◦ Valor ganado: 30 % ◦ Horas calendario planificadas: 2 horas ◦ Horas calendario reales: 9 horas ◦ Horas planificadas en tareas terminadas: 3 horas ◦ Horas reales en tareas terminadas: 9 horas • Métricas de la última semana: ◦ Valor planificado: 0% ◦ Valor ganado: 30 % ◦ Horas calendario planificadas: 2 horas ◦ Horas calendario reales: 9 horas ◦ Horas planificadas en tareas terminadas: 3 horas ◦ Horas reales en tareas terminadas: 9 horas • En la gráfica: ◦ La línea de Valor Ganado debería indicar que en la última semana se llegó a 30% ◦ La línea de Valor Planificado debería indicar que tres semanas después de la semana actual se llega al 100%. ◦ La línea de Valor Ganado Replanificado debería indicar también que se llega al 100% tres semanas después de la actual pero, dependiendo del día actual dentro de la semana actual, probablemente el domingo de la semana actual se tenga menos valor ganado que en la línea de Valor Planificado, ya que para la replanificación se tiene en cuenta cuántos días restan de la semana actual ◦ La línea de Valor Ganado Pronosticado debería indicar lo mismo mencionado para la línea de Valor Ganado Replanificado, y además teniendo en cuenta que, según el desvío histórico, las tareas finalizadas por mi usuario requirieron del triple de tiempo del estimado y mi usuario dedicó un 150% del tiempo indicado en el calendario. En base a estos datos: ■ se ajusta el calendario y a partir de la semana siguiente a la actual mi usuario debería dedicar 31,5 horas según el calendario (el 450% de lo planificado) ■ se ajustan las estimaciones de mis tareas no finalizadas, y deberían requerir 21 horas (el triple de lo planificado) ■ Analizando semana a semana los ajustes mencionados anteriormente, se puede deducir que el domingo que corresponde a dos semanas (o tres, dependiendo de la fecha actual) posteriores a la semana actual se debería finalizar la tarea de 2 horas estimadas (mi usuario llega al 50% de Valor Ganado), y en la semana posterior se llegaría al 100%. • Fechas previstas de fin: ◦ Fecha prevista de fin replanificada: Dentro de la semana que la gráfica indica que se llega al 100% de Valor Ganado Replanificado ◦ Fecha prevista de fin pronosticada: Dentro de la semana que la gráfica indica que se llega al 100% de Valor Ganado Pronosticado
Resultado obtenido	Flujo completado exitosamente.

E.11.2. Dashboard - Métricas de Proyecto

Dashboard - Métricas de Proyecto	
Resultado esperado	En la vista de proyecto del dashboard debería verse: • Métricas totales: ○ Valor planificado: 0 % ○ Valor ganado: 15% ○ Horas calendario planificadas: 6 horas ○ Horas calendario reales: 9 horas ○ Horas planificadas en tareas terminadas: 3 horas ○ Horas reales en tareas terminadas: 9 horas • Métricas de la última semana: ○ Valor planificado: 0% ○ Valor ganado: 15 % ○ Horas calendario planificadas: 6 horas ○ Horas calendario reales: 9 horas ○ Horas planificadas en tareas terminadas: 3 horas ○ Horas reales en tareas terminadas: 9 horas • En la gráfica: ○ La línea de Valor Ganado debería indicar que en la última semana se llegó a 15% ○ La línea de Valor Planificado debería indicar que tres semanas después de la actual se llega al 100%. ○ La línea de Valor Ganado Replanificado debería indicar también que se llega al 100% tres semanas después de la actual pero, dependiendo del día actual dentro de la semana actual, probablemente el domingo de la semana actual se tenga menos valor ganado que en la línea de Valor Planificado, ya que para la replanificación se tiene en cuenta cuántos días restan de la semana actual ○ Para la línea de Valor Ganado Pronosticado se sigue un razonamiento similar que para la línea de Valor Ganado Replanificado, y además teniendo en cuenta los ajustes mencionados para esta línea para mi usuario en la vista individual, y que además para el otro usuario del proyecto se tenían 2 horas para la semana anterior a la actual pero no se dedicó ninguna. Este último dato hace que, luego del ajuste, se llegue a la suposición de que dicho usuario no le dedica horas al proyecto y por lo tanto nunca finalizará sus tareas. Esto hace que, según este cálculo, este usuario nunca obtenga Valor Ganado Pronosticado, por lo que no se llegará al 100%. • Fechas previstas de fin: ○ Fecha prevista de fin replanificada: Dentro de la semana que ocurrirá tres semanas posteriores a la actual ○ Fecha prevista de fin pronosticada: No hay suficientes horas, por lo mencionado anteriormente
Resultado obtenido	Flujo completado exitosamente.

E.12. Caso de prueba: Proyecto sin horas calendario

Proyecto sin horas calendario	
Descripción	Usuario selecciona un proyecto pero las horas del calendario no son suficientes para las tareas que hayan estimadas.
Datos de prueba	- Usuario autenticado. - Proyecto existente que tenga tareas con horas estimadas pero que le falten horas en el calendario para poder cubrir esas tareas.
Pasos a ejecutar	1. Seleccionar el proyecto en la barra lateral. 2. Navegar a la página de Dashboard. 3. Navegar a la página de Tareas. 4. Navegar a la página de Calendario.
Resultado esperado	- En el Dashboard se muestra un mensaje en las fechas previstas de fin: "Faltan horas en el calendario". - En la página de tareas se muestra una alerta de que faltan horas en el calendario. - En la página de schedule se muestra una alerta de que faltan horas en el calendario. - Los mensajes de error deben también aparecer cuando no alcancen las horas calendario no asignadas con las tareas que no tienen usuario asignado.
Resultado obtenido	Flujo completado exitosamente.

E.13. Caso de prueba: Proyecto - Editar fecha de comienzo

Proyecto - No poder editar fecha de comienzo de proyecto posterior a esfuerzo ya realizado, ni posterior a la primera semana del calendario	
Descripción	Verificar que no se puede editar la fecha de comienzo de un proyecto asignándole una fecha posterior a un registro de esfuerzo existente para cierta tarea en dicho proyecto. Tampoco debe ser posible asignar una fecha de inicio del proyecto posterior a la primera semana del calendario que tenga horas asignadas a algún usuario.
Datos de prueba	- Usuario autenticado. - Proyecto existente que tenga al menos una tarea, completada o en progreso y con al menos un registro de esfuerzo. - El proyecto existente tiene al menos una semana en el calendario. - El usuario es el líder del proyecto.
Pasos a ejecutar	1. Seleccionar el proyecto en la barra lateral. 2. Navegar a la página de Proyecto.
Resultado esperado	- Intentar editar la fecha de comienzo del proyecto a una fecha posterior a la fecha del primer registro de esfuerzo existente para dicho proyecto. Debe mostrar un mensaje de error. - Intentar modificar la fecha de inicio del proyecto estableciéndose en una fecha posterior a la primera semana del calendario que tenga horas asignadas a algún usuario. El sistema debe impedir la operación y mostrar un mensaje de error indicando que la fecha seleccionada no es válida.
Resultado obtenido	Flujo completado exitosamente.

E.14. Caso de prueba: Pantalla de Calendario

Pantalla de Calendario	
Descripción	Usuario puede cambiar de horas en los schedules y se ve reflejado en la página de calendario
Datos de prueba	- Usuario autenticado. - Proyecto existente que tenga más de un integrante de equipo. - Usuario integrante de equipo.
Pasos a ejecutar	1. Seleccionar el proyecto en la barra lateral 2. Anotar las horas que cada integrante hace por defecto. 3. Navegar a la página de Calendario. 4. Cambiar las horas por defecto de al menos un integrante del proyecto. 5. Navegar a la página de Calendario.
Resultado esperado	- Los integrantes que se muestran en el calendario son aquellos pertenecientes al proyecto seleccionado. - Las horas que se muestran de cada integrante en cada semana son las correctas. - Si se cambia la dedicación de horas por defecto, se cambian las horas por defecto en la página del calendario al crear nuevas semanas. - Se pueden agregar semanas al calendario mediante "Tab". - Cuando se cambian las horas de calendario se ve reflejado en el plan. Para probar esto: cambiar horas según tareas disponibles y confirmar que se muestren o no alertas.
Resultado obtenido	Flujo completado exitosamente.

E.15. Caso de prueba: Tareas - Registro de esfuerzo de forma manual

Tareas - Agregar, editar y eliminar registros de esfuerzo de forma manual	
Descripción	El usuario debe poder agregar a cierta tarea un registro de esfuerzo de forma manual y no mediante el temporizador
Datos de prueba	- Usuario autenticado. - Proyecto existente que tenga al menos una tarea sin completar asignada al usuario autenticado.
Pasos a ejecutar	1. Seleccionar el proyecto en la barra lateral 2. Para cualquier tarea sin completar, clickear en el botón de ver el historial de registros de esfuerzo 3. Clickear sobre el botón de "Agregar registro de esfuerzo" 4. Asignarle al registro de esfuerzo fechas de comienzo y de fin en el pasado y posteriores a la fecha de inicio del proyecto.
Resultado esperado	- El registro de esfuerzo fue registrado correctamente, y el campo de horas realizadas en la tarea se actualizó al valor que corresponde. - El registro de esfuerzo creado puede ser editado, y el campo de horas realizadas en la tarea se actualiza al valor que corresponde. - El registro de esfuerzo creado puede ser eliminado, y el campo de horas realizadas en la tarea se actualiza al valor que corresponde.
Resultado obtenido	Flujo completado exitosamente.

E.16. Caso de prueba: Tareas - Pausar Temporizador

Tareas - Temporizador puede ser pausado desde el recuadro mostrado en la esquina inferior derecha	
Descripción	El usuario debe poder pausar el temporizador desde el recuadro mostrado en la esquina inferior derecha.
Datos de prueba	- Usuario autenticado. - Proyecto existente que tenga al menos una tarea sin completar asignada al usuario autenticado.
Pasos a ejecutar	1. Seleccionar el proyecto en la barra lateral. 2. Comenzar el temporizador en una tarea no completada.
Resultado esperado	- Es posible pausar el temporizador desde el recuadro mostrado en la esquina inferior derecha. Esto debería hacer que no aparezca más dicho recuadro, y además debería asignarle fecha de fin al registro de esfuerzo creado al comenzar el temporizador.
Resultado obtenido	Flujo completado exitosamente.

E.17. Caso de prueba: Tareas - Temporizador persistente entre sesiones

Tareas - Temporizador persistente entre sesiones	
Descripción	El temporizador de cierta tarea es persistente, incluso aunque el usuario ingrese a la plataforma desde una ventana nueva.
Datos de prueba	- Usuario autenticado. - Proyecto existente que tenga al menos una tarea sin completar asignada al usuario autenticado.
Pasos a ejecutar	1. Seleccionar el proyecto en la barra lateral. 2. Iniciar el temporizador en alguna tarea no completada. 3. En otra ventana, dirigirse a la aplicación, y en particular al mismo proyecto.
Resultado esperado	- El mismo temporizador comenzado previamente está corriendo, y ya lleva transcurrida la cantidad de tiempo que corresponde.
Resultado obtenido	Flujo completado exitosamente.

Anexo F

Encuesta a profesionales de Pyxis

F.1. Encuesta a diseñadora UX/UI

¿Entiendes que desde UX/UI se logró lo esperado? ¿Qué mejorarías?

“Sí, se logró mejorar y potenciar visualmente la idea que ya tenían, haciendo el producto más usable. Aplicaron todo lo que sugería en UX/UI, así que el resultado habla por sí solo.”

¿Qué hubieras hecho diferente en el proceso de colaboración con los estudiantes?

“El proceso me pareció súper adecuado dadas las circunstancias, especialmente considerando el tiempo limitado de consultoría UX/UI. Usamos estratégicamente cuándo reunirnos cuando mandar un correo. Hicieron varios bocetos a lo largo del proyecto, cada uno contenía una descripción explicando la necesidad y decisión de lo que se esperaba en cada interfaz de usuario antes de juntarnos. No cambiaría absolutamente nada, el equipo funcionó muy bien y las tareas se ejecutaron de manera efectiva y al resultado esperado.”

¿Entiendes que el compromiso y el trabajo fue el esperado por parte de los estudiantes?

“Sí, absolutamente. Como comenté antes, el nivel de compromiso, profesionalidad y apertura que tuvieron fueron excelentes. Se enfrentaron a un rol nuevo en su equipo UX/UI, asumieron responsabilidades diversas en cada etapa del proyecto, se los vio seguros

y con muchas ganas de aportar. Se notó la expertise técnica y soft skill: comunicación, trabajo en equipo, disposición en cada uno de ustedes.”

¿La asignación en horas desde Pyxis fue suficiente para atender las necesidades del proyecto? Si consideras que no, ¿cuál crees que fue la causa y cómo se podría haber previsto?

“Al principio fueron pocas horas, pero Diego hizo su magia y consiguió más horas y ahí sí se notó que eran más adecuadas para este proyecto. La causa asumo que puede ser la falta de experiencia y, por ende, la ausencia de datos históricos que nos hubieran permitido una mejor estimación.”

¿Cómo fue la experiencia desde el lado UX/UI en cuanto al uso de inteligencia artificial?

“Para mí fue el inicio de uso en todo el proceso de AI para generar interfaces de usuarios con impacto UX/UI. Es un camino de ida. Para ser sincera, siento que todas son necesarias para mi día a día: Granola, Perplexity, Cursor, Claude, Lovable, V0, Bolt, entre otras. La decisión de usar una u otra herramienta siempre está dado por los requerimientos del proyecto, la tecnología en que se desarrolle, la librería en que elijan. Porque mi foco no es solo optimizar mi trabajo en UX/UI, sino que lo que genere le sirva al equipo de desarrollo y puedan reutilizar parte del código.

En otros proyectos, los devs me dijeron que usaron hasta un 30 % de lo que generaba. En el último que hice, hace menos de 1 semana, era un MVP y por lo que dijeron usaron un 60 % porque me pasaron la estructura que definieron, usamos datos mockeados para que luego el team luego haga su magia y conecte con los servicios de API. Es muy importante para mí que lo que haga acelere a todo el equipo en lo que este a mi alcance y background, obviamente.”

¿Algún comentario adicional que te gustaría realizar?

“Agradecerle a cada uno de ustedes, por ser parte de este nuevo paradigma donde armamos equipos con estudiantes y están transitando la recta final de su carrera.

Para mí también fue un desafío muy lindo que disfruté mucho. El objetivo de potenciar el UX/UI usando herramientas AI para generar interfaces y código reutilizable no al 100 % pero un 1 % ya es positivo poder reducir el trabajo de todo el equipo no solo en UX/UI es algo que me parece muy valioso.”

F.2. Encuesta a arquitecto de software

¿Entiendes que el compromiso y el trabajo fue el esperado por parte de los estudiantes?

“Creo que el compromiso de los estudiantes fue lo esperado e incluso en algunos momentos superó las expectativas, ya que se mostraba una predisposición a solucionar los problemas y a entenderlos antes de venir con consultas.”

¿Qué hubieras hecho diferente en el proceso de colaboración con los estudiantes?

“Creo que se necesita una instancia de nivelación, a los hechos se terminó realizando, donde se pueda entender el problema y guiar el marco teórico y práctico que se debe aplicar, creo que al principio no lo tuvimos tan en cuenta y para la próxima me gustaría tener eso un poco más armado, no tengo claro si del lado de Pyxis o es algo que tengamos que pedir se realice a la facultad, o ambos.”

¿La asignación en horas desde Pyxis fue suficiente para atender las necesidades del proyecto?

“Acá tenemos un debe, ya que la asignación depende también de la disponibilidad particular de cada tutor, asegurar espacios y recursos es fundamental para que el ida y vuelta pueda ser un poco más fluido.”

¿Algún comentario adicional que te gustaría realizar?

“Pyxis tiene que trabajar más en armar un marco de tutoría para que no dependa tanto de la tracción de cada tutor, en Pyxis existe algo llamado las Horizontales, que son como un tipo de veedor tecnológico, dichas Horizontales deberían trabajar con Research para mancomunar esfuerzos en este aspecto.”