UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Proyekta: Sistema de gestión integral de proyectos y recursos humanos

Informe de Proyecto de Grado presentado por

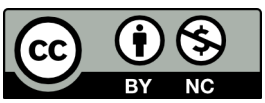
Luciano Carrasco, Maximiliano Jara y Santiago Pereira

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, 12 de junio de 2026



Agradecimientos

Agradecemos profundamente a nuestro tutor Diego Vallespir por su constante acompañamiento, sus aportes y su disposición permanente para guiarnos en cada etapa del proyecto. Su experiencia y claridad fueron esenciales para superar los desafíos técnicos y académicos que surgieron.

Queremos destacar también el valioso aporte de las personas de Pyxis que participaron activamente. A Claudia Martínez, Product Owner, por su visión estratégica, el seguimiento detallado de los requerimientos y su compromiso para que la solución se ajustara a las necesidades reales de la empresa. A Inés Aguiar, responsable de UX/UI, por su dedicación en el diseño de interfaces claras y profesionales. Y a Matías Fernández y Carlos Soderguit, responsables de arquitectura, por sus recomendaciones técnicas y su apoyo en la definición de una infraestructura sólida y escalable. Su experiencia profesional enriqueció notablemente el resultado final.

Por último, a nuestras familias y amigos, por su paciencia, comprensión y apoyo incondicional durante todo este proceso. Su aliento fue una fuente fundamental de motivación en cada momento.

Resumen

El presente proyecto consiste en el desarrollo de una herramienta web denominada *Proyekta*, creada para la empresa Pyxis con el objetivo de gestionar de forma integral los proyectos en curso y planificados, así como los colaboradores involucrados para su ejecución. La solución busca centralizar en un único sistema la información crítica de los proyectos, facilitando su seguimiento, análisis y asignación eficiente de recursos.

La iniciativa surge a partir de la necesidad identificada por Pyxis de contar con una solución a medida, adaptada a sus procesos internos, que le permita obtener una visión integral del estado de sus proyectos, detectar cuellos de botella y tomar decisiones respaldadas por datos.

Además de las funcionalidades operativas básicas, *Proyekta* incorpora un módulo de reportes personalizados que permite realizar análisis avanzados sobre la distribución de recursos, la evolución de los proyectos, la carga de trabajo y otros indicadores relevantes para la gestión estratégica. Entre las principales capacidades implementadas se destacan: la gestión de asignaciones, la detección de sobreasignaciones y tiempos ociosos, la visualización de lineas de tiempo (timelines) por proyecto y por colaborador, y la generación de reportes exportables en distintos formatos. Para llevar a cabo la construcción del sistema, el equipo debió abordar todas las etapas del ciclo de vida del desarrollo de software, incluyendo el análisis de requerimientos, diseño, definición e implementación de la arquitectura, desarrollo, integración de servicios, pruebas y despliegue en un entorno Cloud. Para ello, fue necesario estudiar y profundizar en diversas tecnologías modernas, que se detallaran a los largo de este informe, y que resultaron fundamentales para alcanzar la solución propuesta.

Por otro lado, el equipo definió pautas de trabajo claras en cuanto a roles y responsabilidades, planificación de reuniones y metodología de desarrollo, con el objetivo de mantener transparencia con los colaboradores de Pyxis, promover iteraciones cortas y facilitar validaciones y liberaciones continuas.

Más allá del desarrollo técnico, el proyecto de grado busca analizar el proceso de trabajo llevado a cabo, identificando los principales aciertos, desafíos y oportunidades de mejora detectados durante su ejecución. Este análisis pretende aportar insumos valiosos para futuras colaboraciones entre la Facultad y empresas interesadas en proyectos de grado conjuntos, fomentando la transferencia de conocimiento y la optimización de metodologías, tiempos y expectativas para equipos posteriores.

Palabras clave: Desarrollo de software, Gestión de proyectos, Gestión de colaboradores, colaboración empresa-universidad

Índice general

Capítulo 1

Introducción

En una empresa tecnológica de mediano porte, donde múltiples proyectos y equipos técnicos operan en paralelo, la gestión eficiente tanto de las iniciativas en curso como de los colaboradores involucrados constituye un factor crítico para alcanzar los objetivos estratégicos y operativos. En este tipo de organizaciones, la falta de una herramienta unificada que centralice la información suele generar dificultades en la planificación, el seguimiento y la toma de decisiones basada en datos, afectando la eficiencia general del proceso de gestión.

Pyxis es una empresa uruguaya de tecnología que ofrece soluciones en las áreas de desarrollo de software, servicios y consultoría de infraestructura de datos, ciberseguridad, inteligencia artificial, big data, soluciones de comunicación y marketing digital.

Desde el año 2009 actúa como agente de transformación digital que brinda soluciones 360° para organizaciones en múltiples industrias de mediano y gran porte. La matriz de sus servicios está diseñada para adaptarse a los requerimientos de cada nuevo proyecto (análisis, diseño, implementación y operación). Cuenta con un equipo multidisciplinario de más de 300 colaboradores distribuidos en diferentes lugares de América y Europa, que trabajan atendiendo una diversidad de proyectos de diferentes características. Esta realidad hace especialmente relevante contar con procesos de gestión sólidos y herramientas adecuadas para acompañar y potenciar la operativa diaria.

La gestión operativa de proyectos, colaboradores y asignaciones se ha realizado históricamente mediante diversas planillas de cálculo que buscaban dar respuesta a las necesidades existentes. Si bien en etapas iniciales estas herramientas resultaban suficientes, a medida que la realidad organizacional se ha ido complejizando, se evidencia la necesidad de implementar una solución más robusta. Esta solución permitiría centralizar la información en un único sistema, mejorando tanto la gestión como la visualización de los datos.

Con el fin de abordar estas necesidades, Pyxis propuso un proyecto de grado cuyo objetivo general fue la construcción de una solución integral que permitiera registrar, visualizar y gestionar los proyectos en curso, facilitando la asignación de colaboradores según disponibilidad, capacidades y conocimientos, y optimi-

zando la utilización de recursos para evitar sobrecargas u ociosidad.

El objetivo central fue desarrollar un producto funcional preparado para operar en un entorno productivo real, en lugar de limitarse a construir un prototipo o una prueba de concepto. En consecuencia, la empresa asignó horas profesionales de expertos en Arquitectura/Infraestructura, UX/UI y un Product Owner (PO) dedicado, con el objetivo de asegurar que el resultado final cumpliera con los estándares empresariales de calidad, seguridad y usabilidad. Esto dio lugar a un escenario híbrido: por un lado, un proyecto de grado con tiempos, entregables y alcance académico; por otro, un proyecto de desarrollo de software que debía integrarse en procesos internos ya establecidos y sostener operaciones críticas para Pyxis.

Esta dualidad planteó desafíos particulares en materia de Ingeniería de Software. Fue necesario definir un proceso de trabajo que conciliara prácticas académicas, practicas industriales y restricciones reales de disponibilidad tanto del equipo estudiantil como del personal de Pyxis. A lo largo del proyecto, se aplicaron metodologías ágiles con ciclos iterativos cortos, revisiones frecuentes con la PO, validaciones de interfaces con la especialista en UX/UI y consultas técnicas con Arquitectos. Este proceso no estuvo exento de dificultades: coordinación de agendas, cambios de enfoque producto de necesidades emergentes, solicitud de permisos y múltiples desafíos técnicos.

Además de desarrollar la solución tecnológica, el proyecto no solo dio respuesta a una necesidad concreta de Pyxis, sino que también permitió reflexionar sobre el proceso de ingeniería seguido. En este sentido, se analizan tanto los aspectos técnicos y metodológicos del desarrollo como las dinámicas de trabajo conjunto entre la facultad y el sector productivo, evaluando qué decisiones resultaron adecuadas, qué aspectos podrían haberse abordado de otra manera y qué elementos deberían considerarse en futuros proyectos con características similares. De este modo, se sientan las bases para el análisis detallado que se presenta a lo largo del presente documento.

Objetivos generales

Desarrollar un producto integral para la empresa Pyxis que permita gestionar de forma centralizada los proyectos, colaboradores y asignaciones, optimizando la planificación de recursos, visibilidad de operaciones, y toma de decisiones. Además, el proyecto busca analizar y documentar el proceso de trabajo colaborativo entre la facultad y la empresa, y más específicamente entre el equipo y los profesionales de las distintas áreas involucradas, con el fin de identificar buenas prácticas, desafíos y oportunidades de mejora que contribuyan a fortalecer futuros proyectos conjuntos donde se vincule lo académico y lo empresarial.

Objetivos específicos

- Analizar los procesos de gestión de proyectos, colaboradores y asignaciones a proyectos de la empresa Pyxis, que hasta el inicio del presente proyecto de grado se realizaban mediante el uso de planillas de cálculo, identificando limitaciones y oportunidades de mejora.
- Diseñar y desarrollar una aplicación web con un stack tecnológico dado, que replique y extienda las funcionalidades actualmente gestionadas mediante planillas, permitiendo administrar los proyectos, colaboradores, disponibilidades y reportes de forma centralizada, eficiente y segura.
- Implementar una arquitectura moderna y escalable que facilite el despliegue, mantenimiento y evolución del sistema, aplicando principios de ingeniería de software y buenas prácticas.
- Asegurar que el producto final cumpla con las expectativas y necesidades de Pyxis, trabajando en conjunto con los colaboradores de las distintas disciplinas.
- Aplicar metodologías ágiles de trabajo que promuevan la planificación iterativa, la revisión continua y la adaptación del producto durante todo el ciclo de desarrollo a partir de una puesta en producción temprana.
- Evaluar los resultados obtenidos, identificando beneficios del nuevo sistema, con respecto al enfoque anterior basado en planillas.
- Identificar los principales desafíos surgidos durante la colaboración entre la academia y Pyxis.
- Documentar lecciones aprendidas y las oportunidades de mejora relacionadas con el trabajo conjunto entre la Facultad y la empresa.
- Documentar las buenas prácticas que favorecieron el avance del proyecto y que puedan servir como guía o referencias para iniciativas similares.
- Proponer recomendaciones para fortalecer la colaboración entre la facultad y la empresa.
- Aspectos específicos del producto:
  - Gestión integral de los proyectos, colaboradores y asignaciones de Pyxis teniendo en cuenta todos los aspectos funcionales que se releven con la *product owner*.
  - Integrar el producto con los servicios de Microsoft 365, permitiendo la autenticación a través de Azure.
  - Desplegar en Amazon Web Services (AWS) o Microsoft Azure utilizando Kubernetes.

Resultados esperados

Se espera que el producto desarrollado permita a Pyxis contar con una herramienta centralizada, confiable y flexible para la gestión integral de sus proyectos y colaboradores. La solución deberá contribuir a una mejora en la planificación de cargas de trabajo, la asignación de colaboradores y la generación de reportes, reduciendo los tiempos administrativos y los riesgos asociados al manejo manual de la información. Desde la perspectiva académica, se busca obtener una experiencia de desarrollo integral, abarcando análisis, diseño, implementación, pruebas e implantación en un entorno productivo real. Al trabajar junto a una PO, especialista en UX/UI y expertos en Arquitectura, el proyecto aspira a reproducir prácticas profesionales reales y generar aprendizajes relevantes sobre la ejecución de proyectos de software con un cliente empresarial.

Finalmente, se espera que la colaboración entre la Facultad y la empresa brinde insumos valiosos para fortalecer futuros proyectos de grado que requieran interacción con organizaciones externas, especialmente cuando se trata de productos críticos que deben cumplir estándares de calidad empresarial.

Trabajo realizado

Se desarrolló una herramienta que permite cumplir con los objetivos planteados, entregando una solución actualmente capaz de soportar la gestión operativa de proyectos en Pyxis, y alineada con los estándares técnicos y funcionales definidos junto a los expertos de la empresa. La participación de los distintos expertos resultó fundamental para lograr un producto robusto y adecuado para un entorno real, más allá de un prototipo académico.

Asimismo, el proyecto posibilitó analizar el proceso de trabajo conjunto y reflexionar sobre cómo se articula un proyecto de grado cuando el producto final es crítico para la organización. Estas experiencias aportan aprendizajes significativos para la mejora de futuros proyectos colaborativos entre la Facultad y el sector empresarial.

Organización del documento

Este informe se estructura de la siguiente manera:

- **Capítulo 2: Revisión de antecedentes.** Presenta el contexto teórico y práctico relacionado con la gestión de proyectos y la asignación de colaboradores, así como las principales herramientas y enfoques existentes en el mercado.

- **Capítulo 3: Análisis y diseño del sistema.** Describe los requerimientos funcionales y no funcionales del sistema, el modelado de la solución, y las decisiones de diseño adoptadas.

- **Capítulo 4: Detalles de implementación.** Detalla las tecnologías utilizadas, la arquitectura del sistema y los aspectos más relevantes del desarrollo, tanto en el frontend como en el backend y la infraestructura Cloud.
- **Capítulo 5: Pruebas realizadas y resultados obtenidos.** Expone la estrategia de validación, los casos de prueba ejecutados, los resultados obtenidos y el análisis comparativo respecto al sistema previo basado en planillas.
- **Capítulo 6: Lecciones aprendidas y recomendaciones.** Releva los aprendizajes obtenidos durante el desarrollo, las interacciones con los profesionales de la empresa y las oportunidades de mejora detectadas en el proceso.
- **Capítulo 7: Conclusiones generales y trabajo futuro.** Resume los principales logros del proyecto, las conclusiones derivadas y las posibles líneas de continuidad o evolución del sistema.
- **Anexos.** Incluye todas las historias de usuario.

Capítulo 2

Revisión de antecedentes

Este capítulo presenta el análisis de las soluciones existentes y los enfoques conceptuales que luego sirvieron de base para el desarrollo de la herramienta. Por un lado, se estudian las herramientas disponibles en el mercado para la gestión de proyectos y recursos en entornos donde se ejecutan múltiples proyectos en paralelo, y sus principales limitaciones en el contexto de la empresa Pyxis. Por otro lado, se describen los conceptos técnicos y metodológicos que guiaron el diseño de la solución. Finalmente se expone la necesidad de desarrollar una herramienta personalizada que responda a las particularidades del flujo de trabajo de la organización.

Esta revisión se basa en un análisis comparativo realizado durante la fase inicial del proyecto, considerando tanto herramientas generales centradas en gestión operativa de proyectos individuales, con énfasis en tareas diarias y colaboración, como herramientas especializadas en portafolios orientadas a administración integral de múltiples proyectos, con foco en optimización estratégica de recursos y decisiones de alto nivel.

2.1. Herramientas de gestión de proyectos en el mercado

Durante la etapa de relevamiento se analizaron diferentes plataformas ampliamente utilizadas en la industria del software tales como: Jira, Trello, Asana, Monday.com y Microsoft Project. Estas herramientas ofrecen funcionalidades orientadas a la planificación y seguimiento de tareas, definición de cronogramas, visualización de avances, coordinación de equipos y colaboración en linea. El análisis se centró inicialmente en soluciones de uso generalizado para la gestión de proyectos individuales.

Jira es particularmente robusta para marcos ágiles como Scrum y Kanban; Trello propone una interfaz visual basada en tableros y tarjetas; Asana y Monday.com brindan enfoques generalistas para la gestión del trabajo; mientras que Microsoft Project se centra en la planificación tradicional mediante diagramas

de Gantt. Si bien estas plataformas permiten gestionar múltiples proyectos en paralelo y asignar personas a tareas, su foco principal no está puesto en la administración integral de recursos compartidos ni en la modelización de estructuras organizacionales complejas.

Adicionalmente, se consideró la existencia de soluciones pertenecientes a la categoría de *Project Portfolio Management* (PPM) o *Enterprise Project Portfolio Management* (EPPM), diseñadas específicamente para la gestión de carteras de proyectos a nivel organizacional. Entre estas herramientas se encuentran plataformas como Jira Align, Planview y Oracle Primavera P6, las cuales ofrecen funcionalidades avanzadas para la planificación de múltiples proyectos, la asignación de recursos compartidos y la visualización del estado global de la organización.

No obstante, este tipo de soluciones presentan una serie de desventajas en el contexto de Pyxis. En primer lugar, se trata de herramientas de elevada complejidad operativa y conceptual, orientadas a organizaciones de gran escala, lo que implica curvas de aprendizaje pronunciadas y una sobrecarga funcional respecto a las necesidades reales identificadas. En segundo lugar, su alto costo de licenciamiento y mantenimiento, así como su fuerte acoplamiento a ecosistemas tecnológicos específicos, limitan su viabilidad como solución flexible. Finalmente, estas plataformas ofrecen capacidades de personalización acotadas a los modelos y flujos predefinidos por el proveedor, dificultando la adaptación a procesos internos particulares, como la gestión de colaboraciones informales, la asignación basada en tecnologías específicas y la detección temprana de sobreasignaciones según criterios propios de Pyxis.

En síntesis, si bien existen herramientas consolidadas tanto para la gestión de proyectos individuales como para la administración de carteras de proyectos a nivel empresarial, ninguna de las soluciones analizadas se ajusta plenamente a los requerimientos específicos de Pyxis. Esta situación motiva el desarrollo de una solución propia, diseñada para integrarse con los procesos internos de la organización y ofrecer un mayor grado de flexibilidad, adaptación y evolución.

2.2. Fundamentación del desarrollo a medida

Pyxis manifestó la necesidad de modernizar su operativa, migrando desde un modelo basado en hojas de cálculo, hacía un sistema que reflejara con precisión su proceso de gestión de la cartera de proyectos y asignación de colaboradores. La información relevada durante entrevistas con los responsables del área permitió constatar que, si bien las planillas de Excel ofrecen rapidez, flexibilidad y un bajo umbral de aprendizaje, presentan importantes limitaciones: carecen de control de versiones, dificultan el trabajo colaborativo en simultaneo, no validan reglas de negocio complejas y no brindan visualizaciones dinámicas (como líneas de tiempo, alertas o detección automática de inconsistencias).

La información relevada identificó requerimientos funcionales que no podían resolverse de manera eficiente utilizando herramientas disponibles en el mercado sin configuraciones complejas o licencias avanzadas que pueden resultar costosas.

Entre los factores que justifican el desarrollo a medida se destacan los siguientes:

- **Adaptación total**: la herramienta se ajusta específicamente a los criterios de gestión, nomenclaturas y procesos definidos por Pyxis para alcanzar los objetivos estratégicos y operativos. por el área de Operaciones de Pyxis.
- **Integración corporativa**: autenticación de usuarios mediante Azure de Microsoft 365 utilizado por la empresa.
- **Stack tecnológico**: Herramienta desarrollada en el stack tecnológico en el cual la empresa tiene sólidos conocimientos, facilitando su mantenimiento y evolución futura.
- **Propiedad y control de la información**: todos los datos permanecen centralizados permitiendo trazabilidad y consistencia de la información.
- **Escalabilidad y disponibilidad**: la arquitectura Cloud sobre AWS garantiza un crecimiento flexible y alta disponibilidad.
- **Reportes y toma de decisiones:** exportación de reportes personalizados para apoyar la planificación estratégica y operativa.

La solución propuesta no solo reemplaza el sistema previo basado en planillas, combinando su agilidad con la robustez de un sistema especializado, sino que impulsa la digitalización de procesos internos y fortalece la capacidad de planificación y toma de decisiones estratégicas. Este conjunto de necesidades superaba las capacidades de las herramientas existentes, justificando la creación de una solución desarrollada específicamente para el contexto de Pyxis.

2.3. Conceptos técnicos relevantes

Para comprender el desarrollo de la solución *Proyekta*, resulta pertinente describir las tecnologías y enfoques adoptados:

- **Arquitectura Cloud**: infraestructura desplegada en AWS, aprovechando servicios gestionados, escalables y con alta disponibilidad.
- **Kubernetes**: orquestador de contenedores utilizado para asegurar disponibilidad y despliegue continuo.
- **Terraform**: herramienta para la definición de infraestructura como código (IaC), facilitando la automatización, versionado y reproducibilidad de entornos.
- **Autenticación mediante Azure AD**: implementación de Single Sign-On (SSO) con las credenciales corporativas de M365.
- **Diseño UX/UI**: desarrollo de interfaces acompañado por un profesional de Pyxis para garantizar consistencia visual y facilidad de uso.

Estas tecnologías permitieron construir una solución moderna, escalable e integrada a la infraestructura existente.

Capítulo 3

Análisis y diseño del sistema

En este capítulo se describen los roles del equipo, la metodología de trabajo, los requerimientos del sistema, el diseño funcional y las principales decisiones de diseño tomadas para el desarrollo de la herramienta *Proyekta*.

El objetivo principal es explicar la metodología de trabajo adoptada por el equipo, detallar la interacción con las distintas áreas involucradas del proyecto y exponer el proceso mediante el cual se definió la solución final. Partiendo de las necesidades identificadas de Pyxis, se definieron y priorizaron historias de usuario, que sirvieron como insumo principal para construir la arquitectura de software del sistema, la definición de las pantallas y la implementación de funcionalidades.

3.1. Metodología de trabajo

El desarrollo de la herramienta se llevó a cabo mediante una metodología iterativa y colaborativa, inspirada en principios ágiles que permitió avanzar de forma incremental y adaptar el producto a partir de la retroalimentación continua de los distintos actores involucrados (Beck y cols., 2001).

3.1.1. Roles dentro del equipo

Con el objetivo de organizar el trabajo y distribuir responsabilidades a lo largo del proyecto, se identificaron una serie de roles principales que los integrantes del equipo debían asumir durante las distintas etapas del desarrollo. Estos roles permitieron estructurar las tareas, facilitar la coordinación interna y asegurar la cobertura de los distintos aspectos técnicos y funcionales del sistema así como los aspectos comunicativos del equipo y con el equipo de Pyxis.

- **Encargado de planificación:** responsable del seguimiento de las iteraciones, la planificación de tareas, la estimación de esfuerzos y el control

del avance general del proyecto.

- **Encargado de arquitectura :** responsable del diseño e implementación de la arquitectura del sistema y la definición de componentes en conjunto con los arquitectos de Pyxis.

- **Analista funcional:** encargado del relevamiento, análisis y validación de los requerimientos funcionales, la elaboración y refinamiento de historias de usuario y la alineación de las funcionalidades desarrolladas con las necesidades del negocio.

- **Desarrollo frontend:** responsable del diseño e implementación de la interfaz de usuario, asegurando la correcta interacción con el sistema, la usabilidad y la coherencia visual de la aplicación.

- **Desarrollo backend:** encargado de la implementación de la lógica de negocio y la gestión de datos.

- **Verificación:** responsable de la validación de las funcionalidades implementadas, la ejecución de pruebas funcionales y la detección temprana de errores o inconsistencias.

- **Comunicación cliente/tutor:** encargado de centralizar la comunicación con la referente del proyecto y el tutor académico, coordinando reuniones, presentando avances y canalizando observaciones y ajustes solicitados.

- **Gestión de la configuración:** responsable del control de versiones del código fuente, la gestión de ramas, la integración de cambios y el mantenimiento de la estabilidad del repositorio del proyecto.

- **Documentación de usuario:** encargado de la elaboración de la documentación de manual de usuario y demostraciones.

Si bien existieron responsabilidades asociadas a cada uno de estos roles, debido a que la carga de trabajo de los mismo varió a lo largo del proyecto, los integrantes del equipo asumieron los distintos roles de manera flexible y rotativa. Esta dinámica promovió el intercambio constante de conocimiento, la reducción de dependencias individuales y la toma de decisiones conjunta.

A nivel de organización interna, se establecieron reuniones de coordinación periódicas entre los integrantes del equipo. Estas instancias incluyeron reuniones breves de seguimiento, orientadas a compartir el estado de las tareas, identificar bloqueos y coordinar el trabajo diario, así como reuniones semanales de mayor duración destinadas a la planificación, revisión de avances y definición de objetivos a corto plazo.

3.1.2. Dinámica de trabajo

A lo largo del proyecto se mantuvieron instancias de coordinación con los distintos referentes de la empresa, tanto mediante reuniones como a través de intercambio de correos electrónicos. La figura 3.1 presenta la cantidad de reuniones a distancia mantenidas a lo largo del proyecto con cada uno de los referentes.

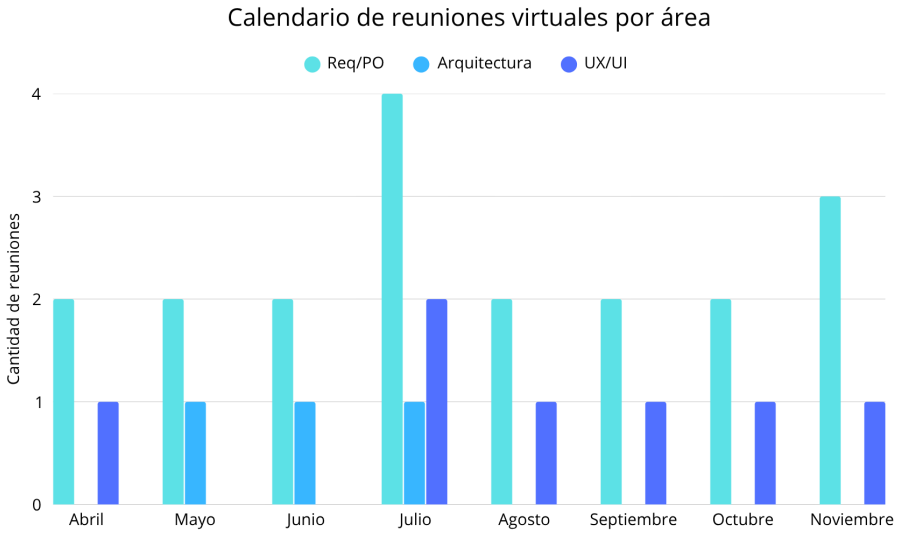


Figura 3.1: Cantidad de reuniones mantenidas con los distintos referentes del proyecto

Desde las etapas iniciales, se establecieron reuniones semanales de revisión y planificación con la referente del proyecto de Pyxis. En dichas instancias se discutían y ajustaban los objetivos de cada iteración, se refinaban y priorizaban las historias de usuario, y se presentaban los avances desarrollados para su validación. Una vez alcanzado un conjunto de funcionalidades o módulo y validado como correcto por la referente del proyecto, se acordaba su liberación en una nueva versión del sistema. Estas liberaciones no se encontraban asociadas a ciclos temporales rígidos, sino al cumplimiento de metas funcionales concretas previamente definidas.

De forma complementaria, se realizaron instancias de coordinación con el equipo de arquitectura de Pyxis para el análisis de aspectos vinculados al despliegue, la autenticación y las decisiones de infraestructura.

En contraste, la interacción con la referente de UX/UI se realizó en menor medida y de forma mas puntual, enfocándose principalmente en la validación de criterio de diseño visual y experiencia de usuario. En particular, se dieron al inicio del proyecto, para definir los componentes principales que debería tener la aplicación, y en el final donde se ajustaron criterios de diseño visual como la

paleta de colores, tipografía, etc.

Esta dinámica de trabajo favoreció una comunicación fluida tanto dentro del equipo de desarrollo como con los referentes de la empresa, permitiendo detectar desvíos de manera temprana y realizar ajustes oportunos sobre los requerimientos, la arquitectura del sistema o el diseño de la interfaz.

3.2. Requerimientos del sistema

El análisis de requerimientos constituyó una de las etapas más importantes del proyecto, ya que permitió comprender en profundidad la operativa interna de la empresa Pyxis y las necesidades especificas asociadas a la gestión de proyectos y asignaciones.

El proceso comenzó con reuniones con la Responsable de Operaciones y Product Owner (PO) de la herramienta, donde se relevaron los flujos de trabajo y las limitaciones del proceso basado en planillas. A partir de estos insumos, se identificaron los puntos críticos y las áreas donde el producto a desarrollar aportaría mayor valor.

Posteriormente, estas observaciones fueron analizadas en las sesiones de refinamiento semanales, donde se transformaron en requerimientos funcionales y no funcionales que orientaron la definición de la solución.

3.2.1. Requerimientos funcionales

Los requerimientos funcionales representan una visión de alto nivel de las capacidades que debe ofrecer la herramienta. Estos se agrupan en los principales módulos del sistema:

- **Gestión de proyectos:** creación, edición y visualización del estado general de los proyectos.
- **Gestión de áreas y servicios:** administración de la estructura operativa asociada a cada proyecto.
- **Colaboradores y asignaciones:** gestión de colaboradores y gestión de asignaciones identificando sobre asignaciones y recursos ociosos.
- **Reportes y análisis:** generación de información exportable con filtros avanzados que apoyen la toma de decisiones.
- **Configuración de catálogos:** creación, edición y visualización de los catálogos del sistema.

En la práctica, estos módulos permiten una operación integrada y coherente. El usuario puede registrar proyectos con sus datos básicos y definir su estructura operativa mediante áreas, entendidas como bloques de trabajo con roles específicos, cargas horarias y períodos de ejecución, así como mediante servicios que representan prestaciones especializadas orientadas a entregables o a la

atención del cliente. A partir de esta configuración, es posible asignar colaboradores a las distintas áreas o servicios, gestionando su disponibilidad, detectando automáticamente sobreasignaciones o tiempos ociosos y realizando ajustes en tiempo real.

Para ofrecer una visualización clara de la disponibilidad y asignación de recursos, el sistema incorpora líneas de tiempo de proyectos y de colaboradores. Estas presentan vistas cronológicas tipo Gantt que permiten identificar duraciones, solapamientos y carga horaria, además de posibilitar la creación y edición de asignaciones directamente sobre la visualización.

Para facilitar el análisis y la distribución de la información fuera del sistema, se incorpora un módulo de reportes orientado a la generación de salidas exportables. Su objetivo es ofrecer una versión simplificada y consolidada de los datos clave, como la disponibilidad de colaboradores, el avance de los proyectos y la asignación de recursos, permitiendo su revisión sin necesidad de acceder a la plataforma. De esta manera, se habilita tanto el análisis "fuera de línea" como la posibilidad de compartir información relevante con actores externos, manteniendo claridad y consistencia en los datos presentados.

Por su parte, el módulo de catálogos mantiene actualizadas las listas maestras de las principales entidades del sistema, garantizando consistencia y homogeneidad en la información utilizada a lo largo de toda la plataforma. Entre las entidades gestionadas se incluyen: agrupación, cliente, contrato de colaborador, contrato de proyecto, estado, experiencia, rol, antigüedad (*seniority*), tecnología, tipo de servicio y usuarios permitidos.

Estos requerimientos funcionales permitieron delimitar el alcance general del sistema. Sin embargo, la definición detallada de los flujos, reglas de negocio y criterios de aceptación se realizó mediante historias de usuario, elaboradas junto a la Product Owner durante las iteraciones de análisis.

La participación de la Product Owner resultó especialmente valiosa durante este proceso. Su perfil técnico, combinado con un profundo conocimiento del negocio, permitió revisar y ajustar las definiciones funcionales con un alto nivel de precisión, facilitando la identificación de reglas de negocio, excepciones operativas y criterios de aceptación que no surgían de forma inmediata en el análisis funcional.

Como resultado de estas iteraciones, se obtuvo un conjunto consolidado de requerimientos funcionales y criterios de aceptación, que sirvieron como base para la elaboración de las historias de usuario y para orientar el diseño funcional del sistema.

La siguiente sección presenta el conjunto de historias de usuario que permitió descomponer estos requerimientos generales en funcionalidades concretas, priorizables y verificables, definiendo así el backlog del proyecto.

3.2.2. Historias de usuario y definición del backlog

Una vez identificados los requerimientos principales del sistema y sus módulos funcionales, se procedió a detallarlos mediante historias de usuario. Esta

técnica permitió capturar la necesidad desde la perspectiva del usuario final y orientar el desarrollo a funcionalidades con valor real para la organización. Las historias se redactaron en el documento titulado Historias de Usuario (ver Anexo A) siguiendo el formato estándar:

Como [rol], quiero [función], para [objetivo].

El proceso de construcción del backlog se realizó en conjunto con la PO, lo que permitió priorizar las funcionalidades de acuerdo con sus objetivos de negocio.

A modo de ejemplo, se presenta a continuación una historia de usuario completa, y posteriormente se lista el encabezado de todas las historias de usuarios resultantes (muchos términos están en inglés debido a que es el lenguaje utilizado en Pyxis):

H14 – Crear área desde timeline de colaboradores

COMO: *Administrador del sistema, Responsable de operaciones, Project Manager*
QUIERO: Crear un área de proyecto para un colaborador directamente desde la timeline de colaboradores.
PARA: Asignar a la persona a un área de un proyecto existente de manera rápida y contextual.

DESCRIPCIÓN: Desde la timeline de colaboradores se debe permitir crear un área en un proyecto seleccionado, quedando el colaborador asignado como candidato del área.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe poder seleccionar el proyecto al que le corresponde el área desde un listado con los nombres de los proyectos registrados.
2. Se debe poder seleccionar el rol desde un listado con los roles definidos en el sistema, permitiendo que el usuario defina un nuevo rol.
3. Opcionalmente el usuario puede definir la carga horaria diaria, las fechas de inicio y fin, sí está confirmada y si es billable.
4. El colaborador seleccionado desde la timeline debe quedar registrado como candidato para el área.
5. Por defecto el campo confirmado es verdadero, salvo que el proyecto sea tentativo, en cuyo caso por defecto es no confirmado.
6. Por defecto el campo billable es verdadero.
7. Por defecto, las fechas de inicio y fin toman el valor de las fechas del proyecto al que pertenece el área a crear (si es que están definidas).

8. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.
9. Se debe validar que las fechas seleccionadas están dentro de la duración del proyecto.
10. El área debe registrarse como informal.
11. En caso de que se superen las fechas del proyecto, se debe permitir extenderlo para asegurar la consistencia

---

Módulo: Gestión de proyectos

- **H1. Registrar proyecto:** Como usuario, quiero registrar un nuevo proyecto para incorporarlo al portafolio general.
- **H2. Editar proyecto:** Como usuario, quiero editar la información de un proyecto existente para corregir o actualizar la información asociada.
- **H3. Listado de proyectos:** Como usuario, quiero acceder a un dashboard con el estado general de los proyectos.
- **H4. Detalle de proyecto:** Como usuario, quiero acceder al detalle completo de un proyecto desde el listado para consultar toda su información relevante, incluyendo alertas visuales sobre atrasos y asignaciones incompletas.
- **H5. Timeline de proyectos:** como usuario, quiero visualizar una timeline de los proyectos para obtener una vista clara de la duración de los proyectos ingresados en el sistema.
- **H6. Extender proyecto:** Como usuario quiero extender la fecha de fin del proyecto hacia adelante para arrastrar hacia adelante todas las fechas de fin de las áreas y servicios donde su fecha de fin coincide con la de fin de proyecto hacia adelante.

Módulo: Gestión de áreas y servicios

- **H7. Crear áreas de proyecto:** Como usuario, quiero registrar las áreas requeridas a un proyecto para definir la estructura del equipo que se necesita para su ejecución.
- **H8. Registrar servicios de proyecto:** Como usuario, quiero registrar los servicios requeridos de un proyecto para definir la estructura del equipo que se necesita para su ejecución.
- **H9. Crear colaboración para servicio:** Como usuario, quiero agregar a un colaborador a un servicio creado previamente, para definir la asignación de una persona al proyecto.

Módulo: Colaboradores y asignaciones

- **H10. Registrar colaborador:** Como usuario, quiero crear una nueva persona con información básica para registrar formalmente un colaborador dentro del sistema.
- **H11. Editar colaborador:** Como usuario, quiero editar la información de un colaborador, para corregir o actualizar la información asociada.
- **H12. Listado de colaboradores:** Como usuario, quiero visualizar un listado de todos los colaboradores registrados.
- **H13. Linea de tiempo de colaboradores:** Como usuario, quiero visualizar en una linea de tiempo, items que reflejan una timeline que muestre la carga de trabajo de cada colaborador con sus asignaciones actuales, tentativas y licencias, para poder analizar la distribución de trabajo, identificar sobre-asignaciones y tomar decisiones informadas de la planificación.
- **H14. Crear área desde linea de tiempo de colaboradores:** Como usuario, quiero crear un área de un proyecto para un colaborador directamente desde la linea de tiempo de colaboradores, para asignar a la persona a un área de un proyecto existente de manera rápida y contextual.
- **H15. Asignar a servicio desde linea de tiempo colaboradores:** Como usuario, quiero asignar a un colaborador a un servicio desde la linea de tiempo de colaboradores, para asignar a la persona a un servicios de un proyecto existente de manera rápida y contextual.
- **H16. Asignar licencia desde linea de tiempo de colaboradores:** Como usuario, quiero asignar licencia para un colaborador desde la linea de tiempo de colaborador para registrar y visualizar ausencias de un colaborador en un periodo correspondiente.
- **H17. Editar área desde la linea de tiempo colaboradores:** Como usuario, quiero editar los datos de un área ya registrada directamente desde la linea de tiempo, para mantener actualizada y corregida la información de asignaciones sin necesidad de salir de la vista de la linea temporal.
- **H18. Editar asignación a servicio desde linea de tiempo colaboradores.:** Como usuario, quiero editar los datos de una asignación de colaborador a un servicio directamente desde la linea temporal.
- **H19. Editar licencia desde linea de tiempo colaboradores:** Como usuario quiero editar los datos de un segmento de licencia de un colaborador desde la linea temporal de colaboradores.
- **H20. Linea temporal de tiempo libre:** Como usuario, quiero visualizar una linea temporal que muestre el tiempo libre de cada colaborador en rangos de tiempo, para poder analizar la distribución de trabajo, identificar recursos libres para próximos proyectos.

Módulo: Reportes

- **H21. Reporte de proyectos por finalizar:** Como usuario, quiero ver un listado de los proyectos que finalizan la próxima quincena, para poder evaluar los colaboradores que se liberarán para próximos proyectos.
- **H22. Reporte de colaboradores libres:** Como usuario, quiero ver un listado de los colaboradores con tiempo libre en una quincena dada para poder detectar recursos libres al momento de planificar.
- **H23. Reporte de total equipo disponible por período de tiempo:** Como usuario, quiero un reporte que muestra la disponibilidad total del equipo para cada periodo de tiempo(quincena), para poder visualizar de forma clara el nivel de asignación y designación en cada periodo y tomar decisiones de planificación.
- **H24. Reporte de total de equipo disponible por horizontal:** Como usuario, quiero un reporte que muestra la disponibilidad total del equipo por periodo de tiempo subdividida por horizontal, para poder identificar con precisión la evolución de la disponibilidad de recursos según la especialidad

Módulo: Configuración de catálogo

Las historias de este módulo aplican para las siguientes entidades: Estado de Proyecto, Cliente, Tipo de Contrato de Proyecto, Agrupación, Tipo de contrato de Colaborador, Seniority, Tecnología, Expertice, Tipo de Servicio, Rol y Usuario.

- **H25. Listar entidad:** Como usuario, quiero visualizar un listado de con los valores de la entidad disponible en el sistema.
- **H26. Agregar entidad:** Como usuario, quiero crear un nuevo valor de la entidad.
- **H27. Editar entidad:** Como usuario, quiero editar un valor de una entidad para corregir el valor previo.
- **H28. Eliminar entidad:** Como usuario, quiero eliminar el valor de una entidad para no visualizarlo mas en los listados del sistema.

El resultado de este proceso fue un backlog priorizado, que sirvió como hoja de ruta para el desarrollo incremental del sistema. Cada iteración abarcó un subconjunto de historias, permitiendo entregar valor de manera continua y validar tempranamente la solución junto a la Product Owner.

La elaboración del listado completo de historias de usuario requirió un trabajo progresivo que combinó la especificación detallada con el desarrollo iterativo de las funcionalidades.

El equipo adoptó una metodología de desarrollo iterativa e incremental, acordando con los interesados de Pyxis un proceso de priorización continua. Este enfoque permitió liberar versiones del producto de manera iterativa, facilitando la validación temprana y el refinamiento de las funcionalidades según la retroalimentación obtenida.

Desde las primeras sesiones de trabajo se logró identificar un conjunto central de funcionalidades prioritarias, lo que proporcionó una base sólida para la planificación. Esta identificación temprana permitió:

- Definir una hoja de ruta clara para el desarrollo incremental.
- Establecer un backlog priorizado y flexible.
- Realizar ajustes y añadir nuevas funcionalidades tras cada ciclo.

La aplicación de esta metodología resultó en la entrega continua de valor y en la validación progresiva de la solución, asegurando su alineación con las necesidades reales del negocio.

3.2.3. Requerimientos no funcionales y arquitectónicos

En paralelo al relevamiento funcional, se llevó a cabo un análisis orientado a los requerimientos no funcionales y a los aspectos arquitectónicos del sistema.

Para esta instancia, se mantuvieron reuniones con integrantes del equipo de arquitectura de la empresa, quienes definieron los lineamientos técnicos y las herramientas que debían emplearse durante el desarrollo. El objetivo fue asegurar la compatibilidad con el ecosistema tecnológico existente en Pyxis y garantizar que la solución cumpliera con sus estándares internos de calidad, mantenimiento y seguridad.

Estas sesiones permitieron establecer pautas relacionadas con la arquitectura general, la infraestructura en la nube, los mecanismos de autenticación corporativa y las tecnologías base para el desarrollo.

Requerimientos no funcionales

Entre los principales requerimientos no funcionales definidos, se destacan:

- **Rendimiento:** debe responder de forma ágil, con tiempos de respuesta inferiores a 1 segundo, incluso con grandes volúmenes de datos.
- **Escalabilidad:** la arquitectura debe permitir incorporar nuevas funcionalidades y soportar un mayor numero de usuarios sin requerir cambios estructurales significativos.
- **Usabilidad:** priorizar una experiencia de usuario fluida, con navegación clara, interfaz consistente y accesible.
- **Seguridad:** garantizar la confidencialidad e integridad de la información.

Requerimientos arquitectónicos y tecnológicos

Además se establecieron los siguientes lineamientos arquitectónicos:

- **Integración:** compatibilidad con *Microsoft 365* y autenticación mediante *Single Sign-On (SSO)* corporativo.
- **Lenguajes de programación:** *Next.js* para el frontend y *Springboot* para el backend.
- **Persistencia de datos:** Se deben alojar los datos en bases de datos relacionales.
- **Entorno Cloud:** La aplicación debe desplegarse en *Amazon Web Services (AWS)* o *Microsoft Azure*, utilizando *Kubernetes*
- **IaC:** La infraestructura debe gestionarse mediante *Infraestructura como código (IaC)* usando *Terraform*.

3.3. Proceso de diseño de la experiencia de usuario

Durante las primeras etapas del proyecto, el equipo realizó un primer acercamiento al diseño visual de la herramienta, proponiendo ideas para las pantallas principales y los componentes mas relevantes, basadas en las funcionalidades definidas durante el análisis.

Posteriormente, se contó con la participación de una responsable de UX designada por la empresa, quien colaboró activamente en la definición de los flujos de usuario, la organización visual de la información y el diseño final de las interfaces.

A lo largo del desarrollo, el equipo consultó en diversas ocasiones al área de UX/UI para validar diseños específicos, lo que permitió ajustar la estética y la experiencia de uso para acercarlas a las preferencias de los usuarios finales.

Hacía el cierre del proyecto, la responsable de UX entregó una guía de estilos detallada, que fue incorporada a la solución para consolidar la identidad visual y asegurar la coherencia entre los distintos componentes de la aplicación. Este proceso permitió alinear el diseño de la identidad de Pyxis, priorizando la claridad, la consistencia y la usabilidad.

3.4. Priorización del desarrollo

Con el objetivo de organizar el trabajo de manera eficiente y asegurar entregables funcionales en etapas intermedias, se definió una estrategia de priorización basada en el impacto de cada módulo sobre la operativa interna de Pyxis.

Esta planificación permitió estructurar el desarrollo de forma incremental, facilitando la validación temprana de funcionalidades y reduciendo los riesgos asociados a la integración de componentes futuros.

El orden de desarrollo establecido fue el siguiente:

- En primer lugar, se priorizó la implementación del Módulo de gestión de proyectos y el Módulo de gestión de áreas y servicios, al ser el núcleo y el punto inicial para registrar la estructura operativa (proyectos, áreas y servicios).
- En segundo lugar, se abordó el Módulo de colaboradores y asignaciones, incorporando la creación de colaboradores, su asignación a proyectos y la visualización de linea de tiempo de colaboradores.
- Finalmente, se desarrollaron dos módulos complementarios: el Módulo de configuración de catálogos - que permite administrar las codigueras del sistema- y el Módulo de reportes. Si bien ambos eran necesarios para el funcionamiento completo, se planificaron para las ultimas etapas debido a que el corazón funcional de la herramienta se centraba en los proyectos, los colaboradores y sus asignaciones.

Este enfoque incremental permitió realizar validaciones parciales con la Product Owner, asegurando que cada entrega aportara valor tangible y consolidara la base funcional del sistema para las siguientes etapas.

La Figura 3.2 presenta de manera visual el roadmap final del proyecto, donde se observa la secuencia en que fueron incorporadas las historias de usuario a lo largo del desarrollo.

El núcleo funcional —centrado en la gestión de proyectos y la gestión de áreas y servicios — fue implementado en las primeras etapas, seguido por la incorporación progresiva de la gestión de colaboradores y sus asignaciones. Finalmente, se añadieron los módulos complementarios de reportes y configuración, consolidando el sistema como una herramienta integral.

La Figura 3.2 permite comprender no solo el orden de desarrollo, sino también la estrategia incremental adoptada, en la cual cada bloque de funcionalidades liberadas sentó las bases para la siguiente etapa. De esta manera, el roadmap sintetiza el proceso evolutivo del producto y refuerza la coherencia entre la planificación inicial y la implementación efectiva del sistema.

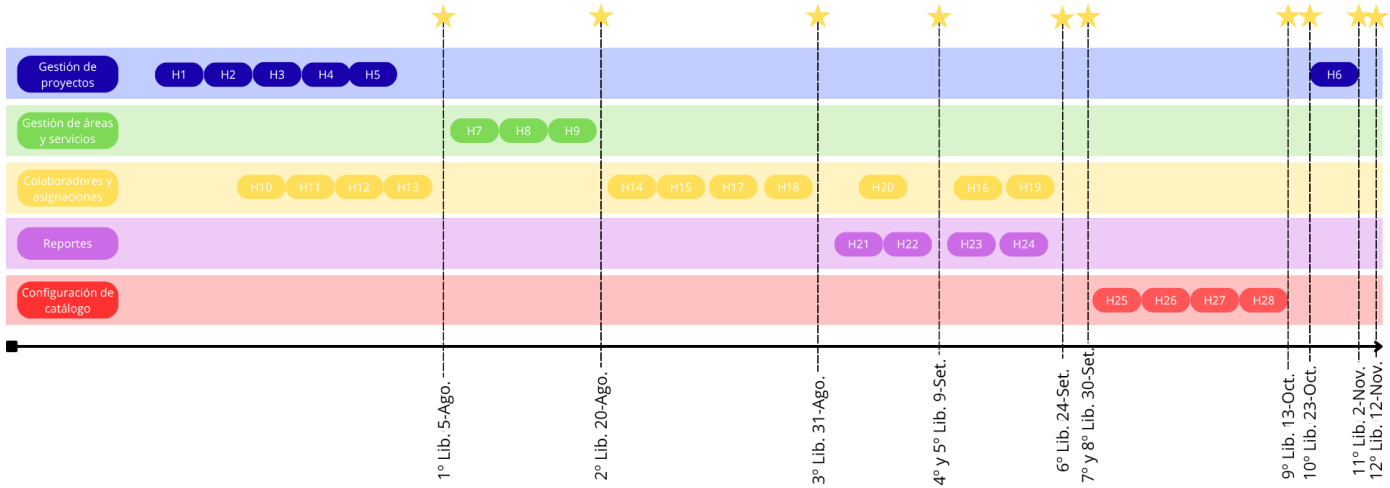


Figura 3.2: Roadmap de implementación del sistema según el orden de desarrollo de las historias de usuario.

Capítulo 4

Detalles de implementación

En este capítulo se describen las decisiones técnicas y de infraestructura que permitieron implementar y desplegar Proyekta en un entorno productivo, en conformidad con los estándares y restricciones tecnológicas definidos por Pyxis. Se presenta cómo se tradujeron los requerimientos funcionales y no funcionales definidos en capítulos anteriores en una arquitectura concreta, escalable y lista para operar en un entorno empresarial real. Asimismo, se busca ofrecer una visión clara y secuencial del camino técnico seguido, desde las decisiones estratégicas hasta la implementación operativa del producto. Se parte del contexto tecnológico inicial impuesto por la empresa, se aborda el diseño de la infraestructura en la nube junto con el proceso de contenerización y despliegue, se explica la integración de autenticación corporativa y se analizan las tecnologías utilizadas en cada capa del sistema.

4.1. Contexto tecnológico inicial

Antes de iniciar el desarrollo, la empresa Pyxis definió un conjunto de tecnologías obligatorias para el desarrollo y despliegue del sistema con el fin de garantizar la coherencia con su ecosistema actual y facilitar el mantenimiento futuro del sistema.

Como resultado, el equipo de proyecto no realizó una comparación exhaustiva entre alternativas tecnológicas, sino que adoptó directamente el *stack* establecido por la empresa. Esta decisión permitió alinear la solución con las prácticas internas, integrarlas sin fricciones con los servicios existentes y aprovechar la experiencia previa de los equipos técnicos de Pyxis.

Las tecnologías definidas fueron las siguientes:

- **Spring Boot** (Tanzu, 2026) ,utilizado para la implementación del backend y la lógica de negocio.
- **PostgreSQL** (*PostgreSQL*, 2026) como motor de base de datos, desplegado en AWS RDS (Relational Database Service) para garantizar alta

disponibilidad y administración automatizada.

- **Next.js** (Vercel, 2026) como framework principal para la construcción del frontend.
- **Docker** (Docker, 2026) para contenedorización de los distintos servicios de la solución.
- **AWS (Amazon Web Services)** (Amazon, 2026), como servicio de computación en la nube.
- **Kubernetes** (Google, 2026) mediante AWS EKS (Elastic Kubernetes Service) como plataforma de orquestación.
- **AWS Fargate** para la ejecución *serverless* de contenedores, reduciendo la gestión de nodos.
- **Microsoft 365 (Azure AD)** (Microsoft, 2026) como proveedor de identidad.
- **Terraform** (HashiCorp, 2026) para la gestión declarativa de la infraestructura.

Este conjunto tecnológico constituyó el punto de partida para las decisiones arquitectónicas y de desarrollo que se detallan en las siguientes secciones.

4.2. Diseño e infraestructura en la nube

Con el stack tecnológico definido, el equipo procedió a diseñar la infraestructura necesaria para desplegar la solución en la nube. Se adoptó una arquitectura basada en servicios gestionados de AWS, priorizando escalabilidad, disponibilidad y facilidad de mantenimiento.

A pesar de que el equipo carecía de experiencia previa en infraestructura en la nube, contenedorización, herramientas de *Infraestructura as Code (IaC)*, se llevó adelante un proceso de aprendizaje gradual, apoyado en:

- documentación oficial,
- asesoramiento del equipo de arquitectura de Pyxis, y
- prototipos técnicos para validar enfoques antes de su adopción definitiva

La infraestructura se definió bajo el enfoque de IaC, utilizando Terraform para describir y versionar los recursos desplegados en AWS. Los principales componentes definidos mediante esta herramienta fueron:

- Clúster EKS (Elastic Kubernetes Service).
- Recursos de red: VPC (nube virtual privada), subredes, internet gateways, y balanceadores de carga.

- Múltiples roles y políticas IAM.
- Instancia de AWS RDS para PostgreSQL.

En paralelo, se desarrollaron las imágenes Docker del frontend y backend, posteriormente desplegadas en clúster de Kubernetes. Esta integración requirió comprender el funcionamiento de *pods*, despliegues, servicios, e *ingress controllers* dentro del ecosistema de Kubernetes.

4.3. Contenerización y despliegue en Kubernetes

La aplicación *Proyekta* fue desplegada en un clúster de AWS EKS como entorno de ejecución *serverless*. Este enfoque permitió construir una solución escalable, y de bajo mantenimiento, alineada con los lineamientos técnicos definidos por Pyxis.

4.3.1. Arquitectura de Kubernetes

El clúster de Kubernetes constituye la plataforma donde se ejecutan los contenedores del backend y del frontend. En la figura 4.1 se presenta el diagrama general de la arquitectura interna.

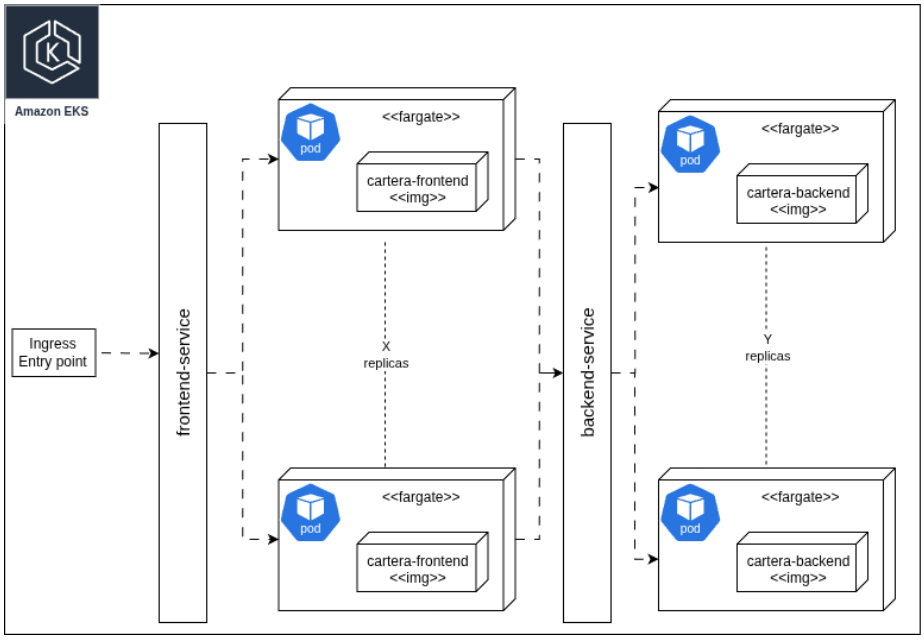


Figura 4.1: Arquitectura del clúster EKS

A continuación, se describen los componentes principales:

Deployments

Los objetos *Deployment* gestionan el ciclo de vida de los pods del backend y frontend. Cada uno referencia una imagen Docker correspondiente y define el número deseado de réplicas (una réplica inicial en esta etapa del proyecto). Kubernetes asegura que los pods se mantengan disponibles y operativos.

Services

Los *Services* proporcionan puntos de acceso estables a los pods:

- *frontend-service* expuesto mediante el *Ingress*
- *backend-service* accesible únicamente dentro del clúster, consumido por el frontend

ConfigMaps

Contienen parámetros de configuración no sensibles, por ejemplo: identificadores de aplicación, parámetros operativos y datos de integración con Microsoft 365.

Secrets

Almacenan información sensible como credenciales para servicios externos, *tokens* de autenticación con Microsoft 365 y claves o configuraciones críticas utilizadas por los pods.

Ingress

El *Ingress* actúa como punto de entrada al clúster desde Internet. Implementado con un Application Load Balancer (ALB) y certificados gestionados por AWS ACM, enruta tráfico HTTPS hacia el *frontend-service*.

4.3.2. Infraestructura Cloud en AWS

La solución utiliza distintos servicios administrados de AWS para garantizar seguridad, disponibilidad, y escalabilidad. en la figura 4.2 se ilustra la arquitectura completa.

A continuación, se describen sus componentes principales:

Route 53

Servicio de DNS administrado. Resuelve el dominio público y dirige tráfico entrante hacia el ALB.

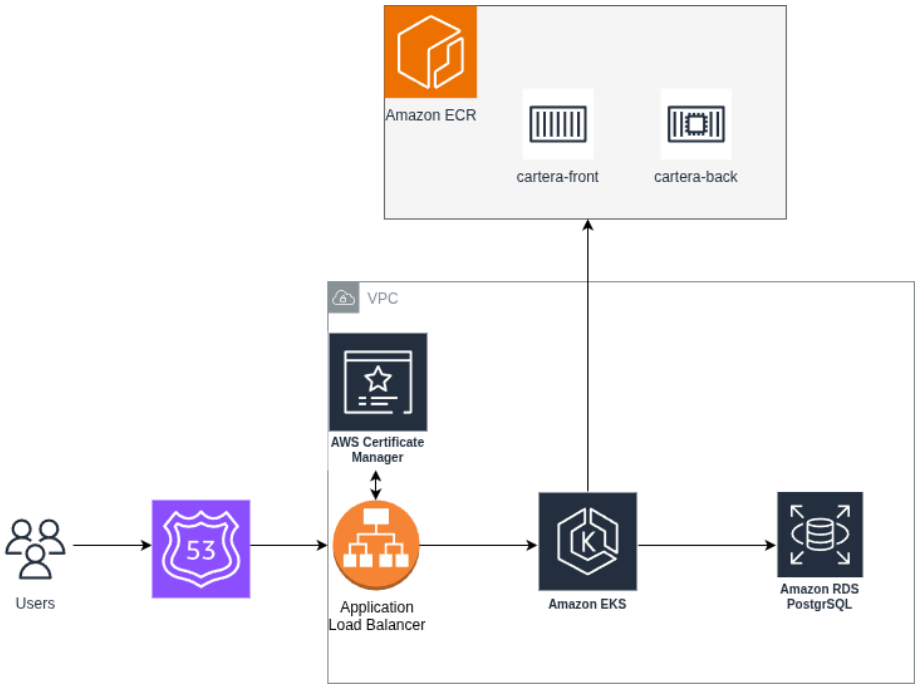


Figura 4.2: Infraestructura Cloud en AWS

Application Load Balancer (ALB)

Gestiona y enruta solicitudes HTTP/HTTPS hacia el cluster EKS, distribuyendo carga y garantizando disponibilidad.

AWS Certificate Manager (ACM)

Provee certificados SSL/TLS utilizados por el ALB para comunicaciones seguras vía HTTPS.

Amazon Elastic Kubernetes Service (EKS)

Orquesta los contenedores de la aplicación. Se utilizó un *Fargate profile* para ejecutar pods sin administrar nodos EC2.

Amazon Elastic Container Registry (ECR)

Repositorio privado para las imágenes Docker del frontend y backend.

Amazon Relational Database Service (RDS)

Base de datos PostgreSQL administrada por AWS, con respaldos automáticos, alta durabilidad y configuraciones de disponibilidad gestionadas.

Cierre de la sección

En conjunto, estos servicios de AWS conforman una arquitectura robusta, escalable y alineada con las prácticas modernas de despliegue en la nube. La utilización de componentes gestionados, como EKS con Fargate y RDS, permitió reducir la carga operativa del equipo y enfocar los esfuerzos en el desarrollo funcional de la aplicación. Asimismo, la integración entre servicios —desde la resolución DNS con Route 53 hasta la provisión de certificados con ACM y el enrutamiento mediante el Application Load Balancer— facilitó la construcción de una solución segura, modular y preparada para evolucionar en el tiempo.

4.4. Integración de autenticación con Microsoft 365

La empresa Pyxis definió que el acceso a *Proyekta* debía realizarse exclusivamente con credenciales corporativas. Para ello se integró autenticación mediante Azure Active Directory, utilizando OAuth 2.0 y (Hardt, 2012) OpenID Connect (OIDC) (OpenID Foundation, 2014).

Esta integración permitió centralizar la gestión de identidades, reforzar la seguridad y habilitar un esquema de *Single Sign-On* (SSO).

Flujo de autenticación

El flujo general se ilustra en la Figura 4.3.

Descripción del flujo

1. El usuario accede a *Proyekta* desde su navegador web.
2. El frontend (Next.js) detecta que no existe sesión valida y redirije a la pantalla de inicio de sesión de Microsoft 365.
3. Azure AD verifica las credenciales corporativas.
4. Si son correctas, Azure AD retorna un *Access Token* firmado.
5. El frontend almacena temporalmente el token y lo envia en cada solicitud al backend (Spring Boot).
6. El backend valida la firma y autoriza únicamente a usuarios autenticados.

Ventajas principales

La incorporación de Azure AD en el proceso de autenticación trajo consigo una serie de beneficios relevantes para la seguridad y la experiencia de uso:

- Unificación de credenciales: los usuarios acceden con su identidad corporativa, sin necesidad de gestionar nuevas cuentas locales.

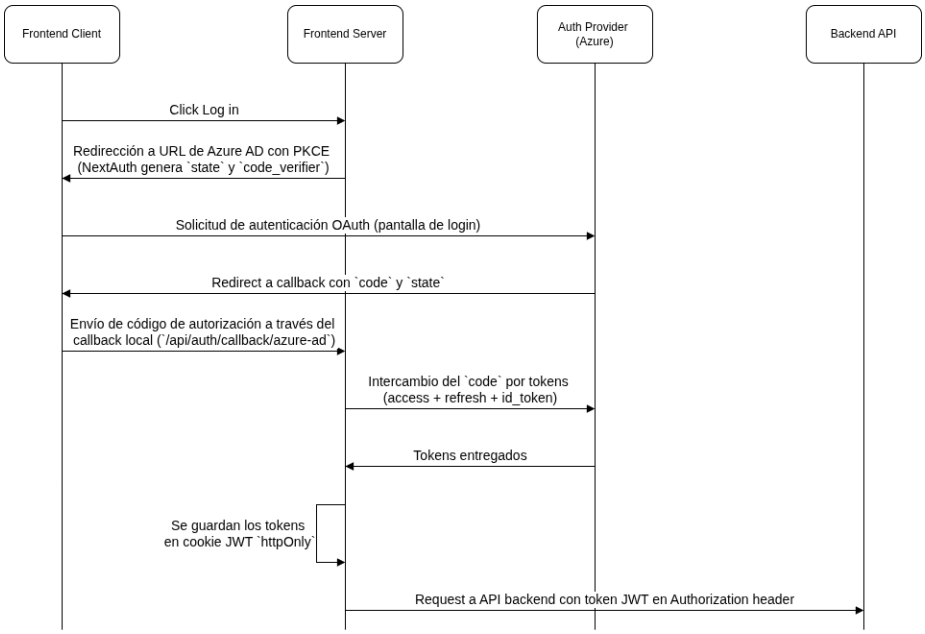


Figura 4.3: Flujo de autenticación

- Reducción de riesgos de seguridad: la plataforma no almacena contraseñas ni maneja procesos de recuperación de claves.

- Mejor experiencia de usuario: se habilita un esquema de Single Sign-On (SSO) con otros servicios corporativos, evitando múltiples inicios de sesión.

Esta integración permitió dotar al sistema de un mecanismo de autenticación robusto, alineado con los estándares de seguridad de la organización y completamente integrado con el ecosistema tecnológico corporativo.

4.5. Tecnologías utilizadas

Backend: Spring Boot

El backend fue desarrollado utilizando Spring Boot, un framework basado en Java que permite construir *APIs REST* robustas, escalables y mantenibles. Se aprovechó su integración con Spring Data JPA para el acceso a la base de datos y su sistema de configuración centralizada, lo que permitió un desarrollo más ágil y organizado.

Base de datos: PostgreSQL en AWS RDS

Se utilizó PostgreSQL como sistema de gestión de bases de datos relacional, pero la solución fue desplegada sobre Amazon RDS (*Relational Database Service*), lo que permitió abstraer la gestión de infraestructura, actualizaciones automáticas y respaldos programados.

El diseño de la base de datos incluyó relaciones entre proyectos, áreas, colaboradores y asignaciones, con índices estratégicos para optimizar la performance. Además, el uso de RDS facilitó la configuración de alta disponibilidad y escalabilidad futura.

Frontend: Next.js

Para la interfaz de usuario se empleó Next.js, un framework basado en React que facilita la construcción de interfaces dinámicas y permite renderizado híbrido (*SSR* y *CSR*). Además, su manejo nativo del enrutamiento y la optimización de rendimiento lo convirtieron en una herramienta ideal para cumplir con los requisitos de Pyxis.

Contenedorización y despliegue

La solución fue diseñada con una arquitectura basada en contenedores:

- **Docker** permitió empaquetar backend, frontend y servicios auxiliares en imágenes reproducibles.
- **Kubernetes** gestionó la orquestación de los contenedores, garantizando alta disponibilidad y escalabilidad.
- **AWS EKS** facilitó la administración del clúster de Kubernetes.
- **AWS Fargate** permitió ejecutar los *pods* de manera *serverless*, evitando la gestión manual de nodos.
- **Amazon RDS** gestionó la base de datos PostgreSQL como un servicio administrado.

La combinación de estas tecnologías permitió construir una plataforma moderna, segura y alineada con las prácticas actuales de desarrollo en la nube. El uso de servicios gestionados de AWS, junto con una arquitectura basada en contenedores y una integración nativa con Azure Active Directory, proporcionó una base sólida para el funcionamiento de *Proyekta*.

Asimismo, la adopción de frameworks consolidados como Spring Boot y Next.js facilitó la implementación de las funcionalidades centrales del sistema, garantizando mantenibilidad, escalabilidad y un ciclo de desarrollo ágil.

En conjunto, estas requerimientos y decisiones tecnológicas definieron la estructura fundamental de la solución y establecieron los pilares sobre los que se desarrollan los aspectos funcionales y operativos descritos en el capitulo anterior.

Capítulo 5

Pruebas realizadas y resultados obtenidos

En este capítulo se presentan las actividades de validación realizadas sobre el sistema, los resultados obtenidos y el análisis de calidad alcanzado durante el desarrollo. Además de las pruebas técnicas y funcionales, se llevó a cabo una encuesta de satisfacción dirigida a los referentes de Pyxis que participaron en el proyecto, con el propósito de evaluar la percepción general sobre la herramienta desarrollada, el proceso de trabajo conjunto y el valor aportado por la colaboración entre la universidad y la empresa. Dicha encuesta permitió complementar los resultados técnicos con una visión cualitativa sobre la experiencia de desarrollo y su impacto dentro de la organización.

5.1. Estrategia general de pruebas

El proceso de validación del sistema se basó en la ejecución de pruebas automatizadas sobre el backend desarrollado en *Spring Boot*, orientadas a garantizar la estabilidad del sistema y el cumplimiento de los requerimientos funcionales. Las pruebas se implementaron y ejecutaron mediante el sistema de construcción *Maven*, validando la correcta operación de las API REST en los módulos de proyectos, colaboradores, asignaciones y reportes.

El conjunto de casos incluyó verificaciones de creación, consulta, modificación y eliminación de entidades, asegurando la integridad de los datos y la correcta aplicación de las reglas de negocio. De forma complementaria, se realizaron validaciones manuales sobre la interfaz web, lo que permitió evaluar la coherencia visual y la integración entre las capas del sistema.

5.2. Pruebas funcionales y liberaciones iterativas

El desarrollo siguió un enfoque iterativo con entregas frecuentes y validaciones continuas junto al equipo de Pyxis. Este esquema de liberaciones tempranas permitió identificar mejoras de usabilidad y corregir errores de manera ágil, asegurando una evolución constante del producto. Las pruebas funcionales también sirvieron para confirmar el desempeño del sistema bajo condiciones de uso reales, verificando la correcta respuesta frente a operaciones concurrentes y volúmenes de datos crecientes.

5.3. Registro y análisis de incidencias

Durante las distintas liberaciones se registraron los errores y observaciones detectadas tanto por el equipo de desarrollo como por los usuarios validadores de Pyxis. A partir de estos registros se construyó un resumen general por versión, presentado en la Tabla 5.1. Allí se observa la cantidad total de bugs detectados y su distribución según la criticidad.

Versión	Fecha liberación	Bugs Totales	Criticidad Baja	Criticidad Media	Criticidad Alta
0.0.1	5-Ago	**1**	1	0	0
0.0.2	20-Ago	**0**	0	0	0
0.0.3	31-Ago	**0**	0	0	0
0.0.4	9-Set	**1**	0	1	0
0.0.5	9-Set	**3**	0	3	0
0.0.6	24-Set	**0**	0	0	0
0.0.7	30-Set	**0**	0	0	0
0.0.8	30-Set	**1**	1	0	0
0.0.9	13-Oct	**6**	1	3	2
0.0.10	23-Oct	**1**	0	1	0
0.0.11	02-Nov	**0**	0	0	0
0.0.12	12-Nov	**0**	0	0	0

Tabla 5.1: Resumen de bugs por liberación

La Figura 5.1 muestra gráficamente la evolución de los errores detectados a lo largo de las versiones, donde las barras representan las distintas criticidades agrupadas por liberación.

En líneas generales, puede observarse que la mayoría de las versiones presentaron pocos o ningún error reportado, lo cual evidencia un nivel de estabilidad sostenido a lo largo del desarrollo. Sin embargo, en la versión *0.0.9* se registró un incremento puntual en la cantidad de defectos, asociado a la incorporación simultánea de múltiples funcionalidades nuevas y a cambios estructurales en el sistema. Esta concentración de incidencias resultó esperable dentro del ciclo de integración continua y permitió detectar y corregir comportamientos anómalos en etapas tempranas, fortaleciendo la calidad de las versiones posteriores. A partir de la versión *0.0.10*, el sistema volvió a mostrar un comportamiento estable y libre de errores críticos, evidenciando la efectividad del proceso de pruebas y la madurez alcanzada por el producto.

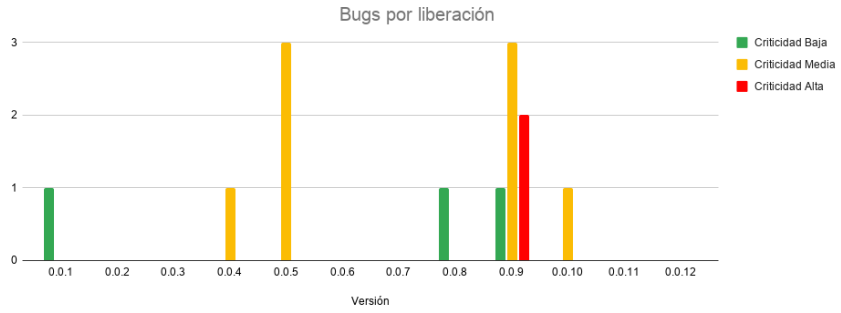


Figura 5.1: Evolución de bugs por liberación

Por otra parte, si bien se detectaron algunos defectos de criticidad alta por parte de los usuarios de Pyxis, los mismos no pudieron ser reproducidos posteriormente por el equipo de desarrollo, ya que las funcionalidades involucradas fueron probadas en múltiples escenarios y funcionaron correctamente. Luego de aplicar las correcciones correspondientes, estas fallas no se reprodujeron en las versiones posteriores.

El detalle completo de las incidencias se presenta en la Tabla 5.2, donde se especifica para cada caso la descripción, criticidad y tipo de error.

Versión	**Detalle**	**Criticidad**	**Tipo**
0.0.1	Timeline proyectos: en el caso de trimestre, no permite moverse de trimestre en trimestre.	Baja	Visualización
0.0.4	Timeline colaboradores: no se ven colaboradores sin asignación	Media	Información visible/no visible
0.0.5	Roles duplicados	Media	Datos duplicados
0.0.5	Proyectos duplicados	Media	Datos duplicados
0.0.5	Demora al crear nuevos proyectos	Media	Demora
0.0.8	Gráficos no se ven en producción	Baja	Visualización
0.0.9	Suma de horas en timeline colaboradores incorrecta	Media	Cálculos incorrectos
0.0.9	Error al intentar realizar asignación	Alta	Creación de datos
0.0.9	Error al crear colaborador	Alta	Creación de datos
0.0.9	No figuran proyectos: “BID/*” y “redgealc” en listado, ni en filtros	Media	Información visible/no visible
0.0.9	Filtro colaboradores por tecnología no funciona	Baja	Visualización
0.0.9	Indicador total de personas sin asignación con error	Media	Cálculos incorrectos
0.0.10	Usuario “Rafael Amor” con fecha de baja pero aún visible en timeline	Media	Información visible/no visible

Tabla 5.2: Detalle de errores detectados por versión

5.3.1. Análisis cualitativo de incidencias seleccionadas

A continuación se profundiza en cuatro incidencias representativas (ver Tabla 5.2), seleccionadas por su valor técnico, impacto en el proceso de validación y aprendizajes derivados. Estas no solo reflejan desafíos encontrados, sino también la proactividad del equipo en su detección, corrección y prevención.

Roles duplicados (versión 0.0.5, criticidad media)

Aunque inicialmente reportado por la PO en el contexto de creación de proyectos, el análisis reveló que el problema de roles duplicados se extendía a múltiples puntos de la interfaz. El origen estaba en los componentes de selección dinámica que permitían crear nuevas opciones en formularios con varios campos de selección (por ejemplo, al asignar roles por área). Cada nuevo valor se enviaba de forma independiente, generando duplicados aunque los nombres coincidieran.

Tras una revisión sistemática, se identificó el mismo patrón en vistas de edición de proyectos, detalles, listado y timeline de colaboradores. Se implementaron dos estrategias de corrección según el caso de uso:

- **Creación inmediata**: el nuevo rol se envía al backend al ingresarlo, actualizando el listado en tiempo real. Ideal cuando se reutiliza en el mismo formulario.
- **Actualización diferida**: se crea al enviar el formulario y se refresca el listado en la vista. Adecuada para reutilización en la misma página, pero no en el mismo formulario.

Ambas soluciones se aplicaron selectivamente, eliminando todas las instancias del defecto en el sistema. Este caso ilustra cómo un error reportado en un único contexto permitió al equipo identificar y corregir un comportamiento inconsistente presente en múltiples módulos, fortaleciendo la integridad global de los datos.

Gráficos no visibles en producción (versión 0.0.8, criticidad baja)

Tras el despliegue de la primera versión con reportes gráficos (*Chart.js*), el equipo detectó por su cuenta que las vistas con gráficos fallaban en producción, mostrando errores del tipo `"bar"/"line" is not a registered controller`. El problema no se manifestaba en el entorno de desarrollo local (`npm run dev`), donde la librería incluía automáticamente los controladores. Sin embargo, el build de *Next.js* eliminaba dependencias no importadas explícitamente, provocando la falla en el entorno de producción.

La solución fue agregar importaciones directas de los controladores requeridos, asegurando su inclusión en el bundle final. Este incidente resalta una diferencia crítica entre entornos de desarrollo y producción en frameworks modernos, y refuerza la necesidad de validar despliegues completos en entornos representativos.

Error al realizar asignación (versión 0.0.9, criticidad alta)

La PO reportó un fallo al intentar una asignación desde la timeline de colaboradores, mostrando un mensaje genérico de error por seguridad. A pesar de múltiples intentos, ni la PO ni el equipo lograron reproducirlo en entornos controlados, bajo diversas condiciones de datos, concurrencia o estado del sistema.

Dado que la funcionalidad fue validada exhaustivamente y no se reprodujo en liberaciones posteriores, se considera probable una condición transitoria externa (timeout, interrupción de red, estado momentáneamente inconsistente). Como mejora, se fortaleció el registro interno de errores (sin exponer datos sensibles), mejorando la trazabilidad futura. Este caso ilustra la dificultad de diagnosticar fallos intermitentes y la importancia de mecanismos robustos de observabilidad.

Proyectos cancelados no visibles en listados (versión 0.0.9, criticidad media)

Se reportó que los proyectos `BID/*` y `redgealc` no aparecían en listados ni filtros. Al verificar, ambos tenían estado *Cancelado*, y según los requerimientos validados con la PO, estos proyectos se ocultan por defecto y solo son visibles al filtrar explícitamente por dicho estado.

Tras confirmar con la PO que el comportamiento era correcto, se identificó una oportunidad de mejora: fortalecer la revisión conjunta de la documentación de requerimientos. Se acordó revisar activamente las reglas de visibilidad y filtrado al inicio de cada funcionalidad, evitando malentendidos derivados de especificaciones previas no revisadas en contexto.

5.4. Cobertura y calidad del sistema

En el proceso de validación se implementaron un total de 28 clases de pruebas automatizadas, distribuidas en los módulos de proyectos, colaboradores y colaboraciones (incluyendo áreas, servicios y licencias). Estas pruebas cubren las principales operaciones expuestas por las API REST, tales como creación, consulta, modificación, eliminación y filtrado de entidades.

Las pruebas automatizadas fueron diseñadas a nivel de controlador utilizando *MockMvc*, permitiendo validar las respuestas HTTP, estructuras JSON y códigos de estado, sin necesidad de desplegar la aplicación completa. Para la definición de los casos de prueba se utilizó un enfoque basado en datos, apoyado en un total de 87 archivos JSON que representan distintos escenarios de ejecución.

La organización de los casos de prueba se estructuró por módulo, operación y código de respuesta HTTP, permitiendo cubrir tanto escenarios exitosos (200, 201), como casos de validación incorrecta (400), ausencia de datos (204, 404) y errores internos (500). Este enfoque permitió validar no solo los flujos principales del sistema (*camino feliz*), sino también comportamientos ante condiciones inválidas o excepcionales.

Módulo	200/201	400	404/204	500	Total
Proyectos	13	12	2	0	27
Colaboradores	10	11	5	4	30
Colaboraciones	15	9	0	6	30
Total	38	32	7	10	87

Tabla 5.3: Distribución de casos de prueba por módulo y tipo de respuesta HTTP

La Tabla 5.3 resume la distribución de los casos de prueba por módulo y tipo de respuesta HTTP. Se observa una cobertura transversal del sistema, incluyendo tanto escenarios exitosos (200, 201) como validaciones de error (400), ausencia de datos (204, 404) y fallos internos (500).

En particular, los módulos críticos del sistema presentan casos de prueba que contemplan no solo el funcionamiento esperado, sino también situaciones inválidas o excepcionales, lo que evidencia un enfoque de validación integral. Esta distribución permitió asegurar que las funcionalidades principales fueran evaluadas bajo distintos contextos de ejecución, fortaleciendo la robustez del producto desarrollado.

En particular, se verificaron reglas de negocio críticas como la gestión de proyectos y las asignaciones de los colaboradores en las diferentes áreas y servicios, así como la correcta respuesta del sistema ante solicitudes inválidas o inconsistentes.

La ejecución de estos casos de prueba en etapas finales del desarrollo permitió validar el comportamiento del sistema en distintos escenarios y reforzar la consistencia de las funcionalidades implementadas. En particular, este proceso contribuyó a reducir la aparición de defectos en las últimas versiones liberadas, tal como se observa en el análisis de incidencias presentado previamente, donde a partir de la versión 0.0.10 no se registran errores de criticidad alta y se evidencia una estabilización del sistema.

5.5. Encuesta de satisfacción con el producto y el proceso de trabajo

Con el fin de obtener una evaluación cualitativa tanto de la herramienta desarrollada como del proceso de trabajo llevado adelante durante el proyecto, se realizaron encuestas a distintos referentes de Pyxis que participaron activamente en su desarrollo. La encuesta fue dirigida a los profesionales de las diferentes áreas involucradas en el proyecto, incluyendo Product Owner, Arquitectura, y diseño UX/UI, lo que permitió relevar la percepción del sistema desde múltiples perspectivas dentro de la organización.

A través de estas consultas se buscó evaluar diversos aspectos del proyecto, tales como el valor aportado por la herramienta, su comparación con el método previo basado en planillas de cálculo, la calidad de la solución técnica imple-

mentada, así como la experiencia de uso y el proceso de colaboración entre el equipo de estudiantes y los referentes de la empresa.

A continuación, se presentan las preguntas realizadas en cada caso, junto con los resultados obtenidos y comentarios aportados.

5.5.1. Visión del negocio (Product Owner)

La encuesta correspondiente a la perspectiva del negocio fue respondida por la PO del proyecto y referente de Pyxis, quien participó activamente durante todo el proceso de desarrollo. Su participación permitió relevar una evaluación directa desde el punto de vista del negocio, considerando tanto el nivel de satisfacción de la herramienta desarrollada, como su alineación con las necesidades operativas de la organización, la efectividad del proceso de colaboración con el equipo de estudiantes, el valor percibido de la herramienta y en particular el valor aportado comparado con el método previo basado en planillas.

A continuación, se presentan las preguntas formuladas junto con las respuestas obtenidas:

- *¿Qué tan satisfecho estás con la herramienta desarrollada?*

  Resultado: 4 sobre 5.

  Comentario: *"La herramienta es correcta, se adapta a las necesidades, se fueron respondiendo en tiempo y forma las solicitudes que surgieron. Se perciben algunos temas de performance en algunas ocasiones, pero se entiende que puedan estar asociadas a la infrastructura."*

- *El alcance definido reflejó las necesidades reales del negocio.*

  Resultado: 5 sobre 5.

  Comentario: *"La herramienta cumple con las necesidades del negocio, ya se encuentra en uso y resuelve satisfactoriamente las necesidades que surgen en la operativa diaria."*

- *¿Cuáles te parecen los principales beneficios/mejoras de la nueva herramienta con respecto a las planillas de cálculo?*

  Comentario: *"Mejora en la performance general, mejora en la visualización, unifica soluciones."*

- *La comunicación con el equipo fue clara, frecuente y efectiva.*

  Resultado: 5 sobre 5.

  Comentario: *"Siempre estuvieron disponibles para las reuniones. las veces que se vio demorada o postergada fue por necesidad del lado del cliente."*

- *Las historias de usuarios reflejaron lo acordado y lo necesario.*

  Resultado: 5 sobre 5.

- *"Fue de mucha utilidad en el proceso de desarrollo una puesta en producción temprana."*

  Resultado: 5 sobre 5.

  Comentario: *"Pudimos acceder a la herramienta en instancias tempranas que permitieron la realización de ajustes y mejoras durante el período de desarrollo."*

- *¿Cuáles te parecen los principales beneficios/mejoras de la nueva herramienta con respecto a las planillas de cálculo?*

  Comentario: *"La herramienta provee una solución integrada, en comparación con la solución compuesta apoyada en diversas planillas, confluence y jira."*

- *NPS - ¿Recomendarías esta modalidad de proyecto a otras áreas/empresas?*

  Resultado: 9 sobre 10.

- *¿Cómo describirías tu experiencia de colaboración con el equipo del proyecto y con la Universidad? ¿Qué aspectos te resultaron más valiosos y qué sugerencias darías para mejorar futuros proyectos similares?*

  Comentario: *"Fue muy interesante poder participar, aunque es desafiante encontrar el tiempo apropiado para darle el seguimiento necesario. De todos modos fue una experiencia muy gratificante."*

- *Satisfacción global del proyecto*

  Resultado: 10 sobre 10.

5.5.2. Visión arquitectónica

Con el objetivo de complementar la evaluación del sistema desde una perspectiva técnica, se realizó una encuesta a los integrantes del equipo de arquitectura de la empresa que participaron directamente en el proyecto. Esta instancia permitió relevar la percepción sobre la solución implementada desde el punto de vista arquitectónico, considerando aspectos como la adecuación de las decisiones técnicas adoptadas, el alineamiento de la arquitectura con los requerimientos del proyecto y la dinámica de colaboración establecida entre el equipo de estudiantes y los referentes técnicos. Asimismo, se buscó identificar posibles oportunidades de mejora tanto en la arquitectura de la herramienta como en el proceso de trabajo desarrollado durante el proyecto.

A continuación, se presentan las preguntas formuladas junto con los comentarios aportados por los referentes de arquitectura:

- *¿Entiendes que desde la arquitectura se logró lo esperado?*

  Comentario: *"El equipo escucho las recomendaciones y propuso una arquitectura pertinente a los requerimientos."*

- *¿Qué hubiesen hecho diferente en el proceso de colaboración con los estudiantes?*

  Comentario: *"Creo que el proceso fue en una cantidad justa y medida, no hubo excesos ni desde nosotros como cliente ni desde los estudiantes como implementadores/constructores de la solución."*

- *¿Entiendes que el compromiso y el trabajo fue el esperado por parte de los estudiantes?*

  Comentario: *"Si."*

- *¿La asignación en horas desde Pyxis fue suficiente para atender las necesidades del proyecto?*

  Comentario: *"Mas que un tema de horas creo que como Pyxis deberíamos apuntar a tener un proceso de asignación de recursos, permisos y gestión mas organizado."*

- *¿Qué mejorarían del producto final respecto a la arquitectura?*

  Comentario: *"Probablemente la división de micro-servicios para hacerlo un poco mas interesante, pero no es nada radical, ya que la solución propuesta es perfectamente aplicable a las necesidades actuales y futuras a largo plazo."*

- *¿Algún comentario adicional que te gustaría realizar?*

  Comentario: *"Fue una muy buena experiencia desde mi parte como colaborador desde el lado 'cliente', volvería a repetirla si continúan viniendo estudiantes de esta calidad humana."*

5.5.3. Visión de experiencia de usuario (UX/UI)

Con el objetivo de complementar la evaluación del sistema desde una perspectiva de la experiencia de usuario, se realizó una encuesta a la referente en UX/UI que participó en el proceso de consultoría durante el desarrollo del proyecto. Su participación permitió relevar una valoración especializada respecto a la calidad de la interfaz de usuario, la adecuación de las mejoras implementadas, a partir de las recomendaciones de la consultoría y la dinámica de colaboración establecida con el equipo de desarrollo.

Asimismo, se buscó conocer su percepción sobre el grado de compromiso del equipo de estudiantes, la suficiencia de los recursos asignados por la empresa y las posibles oportunidades de mejora del producto desde la perspectiva de experiencia de usuario.

A continuación, se presentan las preguntas formuladas junto con los comentarios aportados por la referente consultada.

- *¿Entiendes que desde UX-UI se logró lo esperado?*

  Comentario: *"¡Absolutamente! Team Proyekta logró el objetivo de mejorar su producto con los ajustes de la consultoría UX/UI."*

- *¿Qué hubiesen hecho diferente en el proceso de colaboración con los estudiantes?*

  Comentario: *"No lo haría diferente, todos dimos el 120 %. Y si, se puede hacer tanta cosas distintas como escenarios y contextos tengamos. Pero no cambiaría nada. Todos trabajaron con el mismo compromiso y con una gran apertura para aprender de cada uno de los que formamos este equipo."*

- *¿Entiendes que el compromiso y el trabajo fue el esperado por parte de los estudiantes?*

  Comentario: *"Absolutamente. Todo el equipo trabajo con compromiso, dedicación y profesionalismo por parte de los estudiantes."*

- *¿La asignación en horas desde Pyxis fue suficiente para atender las necesidades del proyecto?*

  Comentario: *"Pyxis sumó más hs a las pactadas inicialmente para poder lograr cubrir los requerimientos del proyecto."*

- *¿Qué mejorarían del producto final respecto a UX-UI?*

  Comentario: *"Sumaría una herramienta de monitoreo como Hotjar que permite detectar errores, fricciones y nuevas oportunidades, para iterar con mejoras continuas como cualquier producto digital."*

- *¿Algún comentario adicional que te gustaría realizar?*

  Comentario: *"Agradecer a Pyxis y a ustedes por incluirme en este proyecto. Fue divertido y desafiante hacer consultoría UX/UI 100 % con AI, integrando el equipo con estudiantes que por primera vez a dos externos: Owner y UX/UI. Vivieron la experiencia real de formar equipos con roles distintos, pero todos bajo un mismo objetivo de mejorar el producto."*

5.5.4. Análisis de resultados

A partir de las respuestas obtenidas en las encuestas realizadas a los referentes de las distintas áreas, es posible analizar la percepción general sobre la herramienta desarrollada y el proceso de trabajo llevado a cabo. Las distintas perspectivas relevadas permiten obtener una visión complementaria del sistema, considerando tanto su valor para la operativa del negocio como la adecuación de la solución técnica y la experiencia de uso del sistema.

En términos generales, los resultados obtenidos muestran una valoración altamente positiva tanto de la herramienta desarrollada como del proceso de trabajo llevado a cabo, considerando las tres perspectivas analizadas: negocio, arquitectura de software y experiencia de usuario.

Desde el punto de vista del negocio, la máxima puntuación obtenida en la adecuación del sistema evidencia un alto grado de alineación con las necesidades de la organización, mientras que la satisfacción global con el proyecto refleja una percepción muy favorable del desarrollo en su conjunto. Asimismo, el nivel de

recomendación alcanzado indica una alta probabilidad de replicar la experiencia en contextos similares.

Desde la perspectiva arquitectónica, las respuestas destacan la adecuación de la solución propuesta a los requerimientos planteados, así como la correcta interpretación de las recomendaciones realizadas durante el proceso. Se valora positivamente la arquitectura implementada, considerándola apropiada para las necesidades actuales y con capacidad de evolución, identificándose como posibles líneas futuras de mejora la incorporación de enfoques más desacoplados, como arquitecturas basadas en microservicios.

Por su parte, la visión de experiencia de usuario refleja una evaluación altamente positiva del trabajo realizado, destacándose el impacto de la consultoría UX/UI en la mejora del producto, así como el compromiso y la apertura del equipo durante el proceso de desarrollo. Como oportunidad de mejora, se sugiere la incorporación de herramientas de monitoreo que permitan detectar fricciones en el uso y facilitar la iteración continua sobre la experiencia del usuario.

De forma transversal, se resalta la dinámica de trabajo colaborativa entre el equipo del proyecto y los colaboradores de la empresa. En particular se destacan la comunicación fluida, la incorporación temprana de feedback y el enfoque iterativo adoptado durante el proyecto. Estos aspectos fueron identificados como factores clave para lograr una solución alineada con las necesidades reales de la organización.

En conjunto, los resultados de las encuestas complementan las pruebas técnicas y funcionales realizadas, aportando una validación cualitativa integral que respalda la utilidad, aceptación y valor del sistema desarrollado dentro de Pyxis. La incorporación de múltiples perspectivas permite fortalecer la validez de los resultados obtenidos, proporcionando una visión más completa del sistema tanto desde el punto de vista funcional como técnico y de experiencia de usuario.

5.6. Lecciones aprendidas y oportunidades de mejora del proceso de pruebas

En términos generales, el proceso de validación resultó efectivo para asegurar la estabilidad del sistema y reducir progresivamente la aparición de errores a lo largo de las distintas liberaciones. No obstante, se identifican oportunidades de mejora tanto a nivel técnico como metodológico. En particular, durante el desarrollo no se implementó un proceso sistemático de revisión de código (*code review*) entre los integrantes del equipo, lo cual podría haber contribuido a la detección temprana de defectos, inconsistencias de diseño y oportunidades de refactorización antes de la integración de nuevas funcionalidades. La incorporación de revisiones de código cruzadas, apoyadas en herramientas de control de versiones y flujos de aprobación, habría permitido mejorar la calidad del código y reducir la aparición de ciertos bugs funcionales.

Asimismo, si bien se lograron buenos resultados mediante pruebas automatizadas sobre el backend, la ausencia de validaciones automatizadas sobre el

frontend limitó la detección temprana de errores relacionados con la interfaz de usuario, la visualización de información y la experiencia de uso. La incorporación temprana de pruebas de integración y pruebas de usabilidad, junto con herramientas de testing end-to-end, podría haber permitido identificar algunos defectos en etapas iniciales del desarrollo, reduciendo el esfuerzo de corrección en versiones posteriores.

Finalmente, una planificación más exhaustiva de escenarios de prueba complejos —como condiciones de concurrencia, estados intermedios del sistema y combinaciones poco frecuentes de datos— habría fortalecido aún más la cobertura de validación. Estas observaciones constituyen aprendizajes valiosos para futuros proyectos con características similares, donde la adopción de prácticas de calidad de software más formales puede potenciar significativamente la robustez y mantenibilidad del producto final.

Capítulo 6

Lecciones aprendidas y recomendaciones

El presente proyecto consistió en el diseño, desarrollo y despliegue de una herramienta integral para la gestión de proyectos y recursos, orientada a su uso en un entorno empresarial. Durante su ejecución, el equipo trabajó en estrecha colaboración con profesionales de la empresa Pyxis, incluyendo una Product Owner, una responsable de experiencia de usuario y un equipo de arquitectura. Esta interacción permitió abordar el desarrollo en un contexto similar al de un proyecto profesional, caracterizado por interacción continua entre distintos perfiles, requisitos cambiantes, restricciones organizacionales y recursos acotados.

Este enfoque implicó abordar la totalidad del ciclo de vida del software, desde el relevamiento de requerimientos hasta el despliegue en nube, incluyendo las etapas de diseño, implementación, pruebas e integración con sistemas corporativos existentes. Como resultado, el desarrollo de la herramienta permitió no sólo alcanzar los objetivos técnicos planteados, sino también obtener una serie de aprendizajes relevantes vinculados al trabajo en equipo, la gestión del proyecto y la interacción en el ámbito empresarial.

Asimismo, una parte sustancial del aprendizaje obtenido estuvo asociada a la experiencia de colaboración entre estudiantes y una empresa del sector privado. Esta dimensión permitió explorar y aplicar buenas prácticas orientadas a la articulación entre el ámbito académico y el empresarial, identificando desafíos tales como la diferencia de prioridades, la coordinación de agendas y la adaptación de metodologías de desarrollo a contextos con recursos y tiempos acotados.

En este contexto, el presente capitulo expone las principales lecciones aprendidas durante el proyecto y propone una serie de recomendaciones orientadas a futuros trabajos de grado que busquen fortalecer la colaboración entre la Facultad y el sector productivo.

6.1. Aspectos positivos y buenas prácticas

Entre los principales aciertos del proyecto se destaca el trabajo colaborativo y coordinado del equipo. La definición clara de tareas y responsabilidades permitió avanzar en paralelo sobre distintos módulos del sistema, manteniendo una comunicación fluida y resolviendo los obstáculos con rapidez. Las reuniones periódicas, junto con el uso de canales de comunicación eficientes, facilitaron la alineación de criterios y la toma de decisiones conjuntas.

Otro aspecto positivo fue la liberación temprana de la herramienta, lo que posibilitó obtener retroalimentación real de los usuarios en etapas intermedias del desarrollo. Esta estrategia resultó clave para la detección temprana de errores, la validación del comportamiento del sistema y el ajuste de funcionalidades de acuerdo con las necesidades operativas de Pyxis, reduciendo riesgos en instancias posteriores del proyecto.

Por otro lado, las reuniones iniciales con el equipo de arquitectura resultaron especialmente útiles. En estas instancias se compartieron lineamientos técnicos, documentación, fuentes de consulta, la pila tecnológica definida y las prácticas utilizadas en Pyxis. Esto proporcionó un punto de partida claro para la investigación y el aprendizaje del equipo, particularmente en aspectos vinculados a la arquitectura de software y la infraestructura en la nube, donde ninguno de los integrantes contaba con suficiente experiencia previa.

El contacto permanente con los referentes técnicos y de negocio de la empresa (product owner, arquitectos y responsable de UX) permitió mantener una visión clara del propósito de la herramienta y asegurar que las decisiones técnicas estuvieran alineadas con las expectativas de uso. La apertura y disponibilidad de los representantes de Pyxis constituyeron un factor determinante para el avance sostenido del proyecto.

En particular, en el ámbito de experiencia de usuario e interfaz (UX/UI), resultó beneficioso el desarrollo inicial de prototipos estáticos en Figma para las vistas principales, los cuales sirvieron como base para para discutir flujos y funcionalidades con la Product Owner y la responsable de UX/UI. Además, la adopción de la herramienta *v0* por parte de la referente de UX/UI facilitó la integración de los diseños, dado que permitía generar código en React con estilos en Tailwind, compatibles con el stack tecnológico del equipo. Esto posibilitó una adaptación rápida y precisa de los elementos visuales durante la implementación. Complementariamente, la elaboración de documentación de vistas contribuyó a mantener informada a la responsable de UX/UI sobre el avance del desarrollo.

Finalmente, en lo que respecta al trabajo con la Product Owner, se mantuvo una comunicación fluida mediante reuniones quincenales y correspondencia regular por correo electrónico. Este vinculo permitió un seguimiento continuo del proyecto y una interacción más detallada en comparación con el resto de referentes, favoreciendo la clarificación temprana de requerimientos y la priorización adecuada de funcionalidades.

6.2. Oportunidades de mejora

Si bien los resultados alcanzados en el proyecto fueron satisfactorios, el proceso de desarrollo permitió identificar oportunidades de mejora relevantes para futuros proyectos de características similares. Una de las principales oportunidades detectadas fue la necesidad de destinar mayor tiempo en las etapas iniciales de análisis y diseño, en lo particular en lo referente a la experiencia de usuario (UX). Si bien las historias de usuario se mantuvieron actualizadas, una dedicación mayor al prototipado y a la validación visual temprana habría permitido anticipar ajustes de diseño y reducir retrabajo en fases posteriores. La definición de flujos de navegación más detallados desde el inicio habría contribuido también a una mayor coherencia entre las distintas vistas del sistema.

En cuanto a la coordinación con los referentes de la empresa, se observaron dificultades puntuales para obtener retroalimentación en tiempo oportuno. Consideramos que estos problemas de coordinación pueden ser producto de la disponibilidad acotada de los colaboradores u otras responsabilidades laborales. Estas situaciones, si bien no fueron críticas, afectaron en algunos momentos el ritmo de avance.

Dado que estos problemas de coordinación resultan inevitables e imprevisibles en este contexto, se recomienda trabajar asumiendo desde el inició que ocurrirán. Para minimizar el impacto, el equipo debería priorizar una comunicación concisa y estructurada: ante múltiples dudas, reunirse internamente para consolidarlas en un único mensaje claro y completo, evitando intercambios dispersos y difíciles de seguir. Además, siempre evaluar si la consulta realmente puede resolverse de manera asincrónica, cuando exista riesgo de que esta genere nuevas dudas en cadena, optar directamente por planificar una instancia sincrónica breve para resolverlas en el momento.

Otro aspecto a mejorar fue la comunicación del cronograma del proyecto de grado hacia los referentes de Pyxis. En las etapas finales del proyecto fue necesario reducir bruscamente el desarrollo de nuevas funcionalidades, para enfocarse en arreglo de errores y la redacción del informe y documentación técnica adicional. Esta transición, inherente al cierre de un proyecto de grado, no había sido comunicada correctamente a Pyxis, lo que generó una concentración de tareas en un periodo muy acotado y limitó el margen para iteraciones y ajustes.

Por último, se identifica como aprendizaje la importancia de planificar de forma explicita la gestión del tiempo, los recursos y los mecanismos de comunicación considerando la disponibilidad real tanto del equipo de estudiantes como la de los colaboradores de la empresa. Si bien el correo electrónico funcionó como canal formal de intercambio, en la práctica presentó limitaciones para el seguimiento continuo de consultas y tareas, ya sea por la dispersión de los mensajes o por respuestas demoradas u omitidas.

En este contexto, habría sido beneficioso contar con una herramienta de gestión compartida, como un tablero de tareas o un sistema de *tickets*, orientado específicamente a que los colaboradores de la empresa pudieran registrar de forma estructurada los errores y observaciones detectadas durante el uso de la herramienta. Esto habría permitido centralizar reportes, estandarizar la información

relevante (pasos de reproducción, contexto, capturas) y mejorar la trazabilidad de la retroalimentación, complementando los mecanismos de gestión interna ya utilizados por el equipo del proyecto. Este tipo de apoyo habría contribuido a una coordinación más previsible y a un uso más eficiente del tiempo disponible, sin comprometer la calidad de la entrega.

6.2.1. Proceso con responsable de UX/UI

El diseño y pulido de UX/UI quedó relegado a las últimas etapas del proyecto, puesto que el equipo decidió priorizar la implementación de nuevas funcionalidades solicitadas por la PO. Esto llevó a que la implementación de diseños y la coordinación con la responsable de UX/UI se apresurara, quizás dejando potenciales ajustes finos o mejoras que, en condiciones más holgadas, podrían haberse abordado. En retrospectiva, habría sido más efectivo involucrar a la responsable de UX/UI desde fases tempranas (más allá de la reunión de introducción al proyecto), al menos para definir y trabajar algunas vistas clave de forma iterativa, evitando la concentración de diseño en el tramo final.

Por otro lado, para mantener a la diseñadora al tanto del avance del proyecto, se desarrolló y actualizó regularmente una documentación de vistas del sistema. Sin embargo, apoyarse principalmente en documentación no resultó suficiente. Habría sido más eficiente mantener reuniones sincrónicas breves para revisar avances, priorizar que vistas requerían atención y clarificar que ajustes eran realmente necesarios dentro del tiempo disponible. Esto habría resultado en diseños detallados para las vistas clave, pudiéndose usar estas como base para adaptar los diseños de vistas con menor prioridad. Cabe destacar, que esta estrategia se aplicó finalmente, pero de manera tardía: se obtuvieron diseños de las vistas más importantes y se extrapolaron para el resto, aunque todo ocurrió bajo presión en la etapa final del proyecto.

6.2.2. Proceso con equipo de arquitectura

La coordinación con el equipo de arquitectura presentó algunas dificultades, principalmente asociadas a la disponibilidad para validar configuraciones y habilitar accesos. Esto impactó en el ritmo del despliegue en la nube, generando retrasos significativos.

Por un lado, el equipo de desarrollo contaba con experiencia limitada en tecnologías de despliegue en la nube como Kubernetes, Docker y el ecosistema de AWS. Si bien se realizó un estudio inicial de dichas tecnologías, se subestimó la complejidad de configurar un entorno productivo completo, lo que derivó en la aparición progresiva de obstáculos técnicos. En retrospectiva, habría sido conveniente realizar pruebas de despliegue tempranas, incluso con aplicaciones simples, para identificar dificultades y reducir incertidumbre. Sin embargo, durante las etapas iniciales del proyecto aún no se encontraban definidas las tecnologías de despliegue, por lo que una profundización temprana podría haber implicado

una inversión de tiempo innecesaria.

Por otro lado, el equipo experimentó problemas con restricciones en los permisos disponibles, lo que impedía realizar ciertas configuraciones sin la intervención del equipo de arquitectura. Esto derivó en intercambios reiterados por correo electrónico, con tiempos de respuesta prolongados y que afectaron el avance del despliegue. Para mitigar esta situación, se complementó la comunicación asincrónica con reuniones breves con el equipo de arquitectura, lo que permitió resolver los bloqueos de forma más ágil y avanzar en las configuraciones necesarias.

En conjunto, estos aspectos evidencian la importancia de abordar el despliegue como una actividad temprana y continua, considerando tanto los desafíos técnicos como las restricciones asociadas a la gestión de permisos y roles en entornos corporativos.

6.2.3. Proceso con Product Owner

El trabajo con la PO se caracterizó por una comunicación fluida, con reuniones quincenales y seguimiento regular. Esta interacción permitió validar avances de forma continua y ajustar el desarrollo en función de las necesidades del negocio.

Durante el proceso surgieron ajustes de requerimientos en etapas avanzadas, algunos de los cuales implicaron cambios relevantes en funcionalidades ya implementadas. Estos casos evidencian que, en contextos donde el producto se define en paralelo a su desarrollo, muchas necesidades solo se identifican a partir del uso directo de la herramienta y no exclusivamente en instancias de relevamiento inicial.

En este sentido, una de las principales lecciones es la importancia de habilitar el uso temprano del sistema. El retraso en el despliegue limitó el tiempo disponible para que la PO explorara la herramienta en profundidad, lo que concentró la detección de ajustes en etapas más avanzadas, con mayor costo de implementación.

Asimismo, se identificó la necesidad de complementar las historias de usuario con materiales orientados al uso práctico, como guías breves de flujos clave. Este tipo de documentación facilita la evaluación del sistema desde la perspectiva operativa y ayuda a identificar inconsistencias, cuellos de botella o mejoras de experiencia de uso.

Otro aspecto que surgió durante el trabajo fue la cancelación ocasional de reuniones por imprevistos. A partir de esta experiencia, se recomienda planificar desde el inicio instancias alternativas ya coordinadas, para así evitar replanificaciones urgentes y mantener la continuidad del seguimiento.

Otro punto relevante fue la gestión del reporte de fallas. Al no contar con un canal estructurado, los reportes se realizaron por correo electrónico con niveles de detalle variables, lo que dificultó su reproducción y análisis en algunos casos. Como mejora, resulta conveniente definir un formato estandarizado de reporte

que incluya pasos de reproducción, contexto y material de apoyo (por ejemplo, capturas o registros de pantalla). Esto permite reducir ambigüedades y agilizar la identificación de la causa de los problemas.

En relación con este punto, se evaluó la posibilidad de pedir a la PO que grabase la pantalla durante su actividad en la plataforma para analizar los errores en contexto. Sin embargo, la idea se descartó por varios motivos:

1. Parte de la adopción del sistema implica el registro de información sensible de la empresa respecto a proyectos y personal. Mantener un registro de esa actividad presenta una vulnerabilidad de seguridad importante.

2. En estas etapas finales del proyecto se busca que la PO pueda usar la plataforma de la forma más fácil posible, pedirle que antes de ponerse a usar la plataforma tenga que prender cualquier programa para grabar agrega una barrera de fricción innecesaria.

3. Queremos evaluar que tanto se adecúa la plataforma a las necesidades del día a día de la PO y potenciales usuarios. Solicitar la grabación puede alterar el comportamiento de la PO y la forma en que interactúa con el sistema, sesgando los resultados.

Aunque en esta sección se mencionan diversas dificultades, es importante considerar que la PO fue el principal punto de contacto durante todo el proyecto. En este tipo de iniciativas, es natural que la mayor parte de los ajustes y aprendizajes surjan en esta interacción, lo que refuerza la importancia de estructurar adecuadamente los mecanismo de validación y comunicación.

6.2.4. Evaluación de la metodología adoptada

Inicialmente, se propuso adoptar Scrum (Schwaber y Sutherland, 2020) como marco de trabajo para estructurar el desarrollo, utilizando sprints de dos semanas como unidad de planificación. Durante la reunión inicial con la PO, se elaboró y validó un backlog inicial organizado en tres módulos principales: gestión de proyectos (creación, edición, listado y un timeline inicial sin asignaciones), gestión de colaboradores (ABM, asignaciones, áreas y servicios) y reportes (gráficos y estadísticas). A partir de este backlog se construyó un roadmap preliminar que distribuía funcionalidades previstas y estimaba en qué sprint se abordarían.

En las primeras etapas del proyecto, el equipo asumió de forma incorrecta, que el alcance funcional se encontraba relativamente definido, y que el roadmap constituiría una guía estable para la planificación del desarrollo. Sin embargo, al iniciar la implementación del sistema y compartir los avances con la PO, se evidenció que el alcance del sistema se expandía y los requisitos experimentaban modificaciones significativas. En múltiples instancia de revisión surgieron cambios sustanciales sobre lo ya desarrollado, así como ajustes derivados del uso directo de la herramienta por parte de la PO. Esta situación fue comprendida por parte del equipo considerando el objetivo inicial del proyecto: reemplazar el

sistema basado en planillas y mejorarlo, lo que parecía tener un alcance relativamente claro, pero que comenzó a evolucionar y modificarse a partir de nuevas funcionalidades que el sistema debía incorporar. A modo de ejemplo, surgieron requerimientos no previstos inicialmente, como la gestión de licencias, gestión de los catálogos del sistema, colaboradores que podían ser dados de baja, y muchas otras validaciones especificas derivadas de los flujos operativos reales de Pyxis, los cuales se hicieron evidentes al interactuar con el sistema en desarrollo.

En ese contexto, el roadmap definido en etapas tempranas quedó obsoleto y la planificación basada en sprints comenzó a resultar difícil de sostener debido a la constante incorporación de nuevos requerimientos y a la necesidad de modificar funcionalidades ya desarrolladas para adaptarlas a flujos reales de trabajo.

Si bien Scrum está diseñado para gestionar cambios en el alcance mediante la priorización continua del backlog, en este caso los cambios no se limitaron a ajustes o refinamientos de funcionalidades ya identificadas. Como se mencionó anteriormente, a lo largo del proyecto surgieron nuevos módulos y funcionalidades que no habían sido contemplados inicialmente. Esta situación llevó, en etapas avanzadas del proyecto, a dedicar esfuerzos significativos a la implementación de funcionalidades adicionales y a la realización de ajustes sustanciales sobre lo ya construido, alterando la planificación de los sprints definidos previamente.

Adicionalmente, la aplicación estricta de Scrum presentó dificultades prácticas relacionadas con el tamaño reducido del equipo y las restricciones propias del contexto académico (fecha fija de entrega y defensa), así como dificultades de agenda para mantener las ceremonias formales, particularmente las reuniones diarias (*daily stand-ups*) y planificación estricta de sprints.

Como consecuencia de estas dinámicas, el proceso de trabajo evolucionó de manera natural hacia un enfoque iterativo flexible. En la práctica el equipo adoptó la realización de reuniones semanales o quincenales para presentar avances, validar posibles escenarios de comportamiento de la aplicación, explorar opciones de diseño, detectar oportunamente cambios o modificaciones y tomar decisiones tempranas con el fin de ahorrar re-trabajo.

Esta forma de trabajo se adecuó mejor al contexto del proyecto, caracterizado por un proceso exploratorio orientado a diseñar una herramienta que reemplaza un sistema existente y al mismo tiempo mejora flujos de trabajo internos. No obstante, la falta de un marco más estructurado generó, en algunos casos, dificultades para visualizar el proceso global del proyecto.

Para proyectos futuros en un contexto academia–empresa, se recomienda evaluar con mayor detenimiento la estabilidad esperada de los requerimientos y los recursos disponibles antes de definir la metodología. En contextos con alto grado de incertidumbre, marcos más orientados al flujo continuo, como Kanban, pueden resultar más adecuados al priorizar la visualización del trabajo en curso y la flexibilidad, sin imponer iteraciones fijas (Anderson, 2010; Kniberg y Skarin, 2010).

6.3. Recomendaciones para futuros proyectos

A partir de la experiencia obtenida, se proponen las siguientes recomendaciones para proyectos de grado que se desarrollen en un contexto de colaboración con empresas:

- Definir desde el inicio instancias de validación con referentes externos, incluyendo frecuencia, tiempos de respuesta esperados y mecanismos de recuperación ante imprevistos. En particular, establecer instancias de respaldo o, en su defecto, un canal asincrónico con un protocolo claro para el envío de consultas y la recepción de retroalimentación.
- Comunicar explicitamente el cronograma del proyecto a todos los colaboradores, incluyendo hitos relevantes y, especialmente, el cambio de foco en etapas finales hacia la estabilización, corrección de errores y ajustes de diseño.
- Involucrar a los perfiles de UX/UI desde etapas tempranas en la definición de vistas clave, evitando concentrar el diseño en fases finales. Priorizar validaciones iterativas sobre las pantallas más críticas y reutilizar estos diseños como base para el resto del sistema.
- Mantener un esquema de comunicación continua con los referentes de la empresa, promoviendo reuniones breves y frecuentes, y optando por interacciones síncronas en áreas como UX/UI para maximizar el uso de horas asignadas, en lugar de depender solo de documentación asincrónica.
- Incorporar guías de uso simples para flujos clave del sistema, con el objetivo de facilitar la evaluación por parte de usuarios no técnicos y mejorar la calidad de la retroalimentación recibida.
- Realizar estudios profundos y pruebas prácticas de herramientas críticas (e.g., plataformas de nube) desde etapas iniciales, designando responsables para mitigar subestimaciones de complejidad, incluyendo despliegues de prueba para identificar limitantes tempranamente.
- Identificar y planificar explícitamente las dependencias externas (como validaciones o habilitación de permisos), considerando posibles tiempos de espera y evitando que se conviertan en bloqueos críticos.
- Establecer sistemas formales de reporte de bugs, como tickets con explicaciones claras de pasos reproducibles, capturas y detalles, para agilizar la identificación y resolución, evitando inconsistencias en feedback vía correo electrónico.
- Adoptar un enfoque de planificación acorde al nivel de incertidumbre del proyecto. En contextos exploratorios, priorizar esquemas flexibles basados en hitos y revisión continua, evitando depender de planificación rígida por iteraciones cuando el alcance evoluciona constantemente.

Estas recomendaciones buscan capitalizar la experiencia adquirida, reforzando las buenas prácticas observadas y abordando los puntos que presentaron mayores desafíos. En conjunto, las lecciones aprendidas constituyen una base sólida para el diseño y ejecución de futuros proyectos de grado orientados a la vinculación con empresas del sector tecnológico.

Capítulo 7

Conclusiones generales y trabajo futuro

El desarrollo de Proyekta cumplió los objetivos principales establecidos: construir una herramienta web integrada para Pyxis que centraliza la gestión de proyectos, colaboradores y asignaciones, sustituyendo planillas manuales por un sistema escalable. Esta solución optimiza la planificación de recursos en un equipo de más de 300 colaboradores distribuidos, facilitando decisiones basadas en datos y mejorando la eficiencia operativa.

Desde el punto de vista técnico, el proyecto implementó una arquitectura cloud sobre AWS con Kubernetes e integró autenticación mediante Microsoft 365. La aplicación de metodologías ágiles, con iteraciones cortas, rotación de roles y validaciones continuas, permitió una adaptación progresiva a los requerimientos del negocio. Las pruebas funcionales y la encuesta de satisfacción evidenciaron un nivel adecuado de robustez y aceptación. No obstante, los retrasos en determinadas integraciones resaltaron la necesidad de incorporar instancias de prototipado en etapas más tempranas.

La colaboración universidad-empresa mostró un impacto positivo en la transferencia de conocimiento, especialmente en áreas como UX/UI y arquitectura de software. No obstante, también expuso desafíos vinculados a la coordinación de agendas y a la alineación de expectativas, aspectos que podrían abordarse con definiciones iniciales más formales y acuerdos de trabajo explícitos.

7.1. Recomendaciones

Para nuevas colaboraciones entre Pyxis y proyectos de grado, se sugiere incorporar sesiones tempranas de prototipado con responsables de UX/UI para minimizar el retrabajo, establecer lineamientos de arquitectura desde el inicio para evitar cambios estructurales y adoptar enfoques metodológicos ágiles capaces de adaptarse a entornos con alta variabilidad de requerimientos. Dichas metodologías deberían contemplar las restricciones de recursos humanos, pro-

pias de equipos reducidos, así como las limitaciones como temporales derivadas de la duración acotada del proyecto. Estas acciones pueden contribuir a optimizar tiempos, reducir fricciones y fortalecer la colaboración entre estudiantes y empresa.

7.2. Trabajo futuro

Para las próximas iteraciones del sistema se proponen las siguientes líneas de evolución. Algunas de estas ideas surgieron en reuniones con los responsables de Pyxis y quedaron fuera del alcance del proyecto por limitaciones de tiempo y prioridad. Otras son propuestas del equipo, surgidas de la experiencia durante el desarrollo y del uso de la herramienta:

- Implementar un pipeline de CI/CD (Continuous Integration and Continuous Delivery) automático en el entorno de producción para facilitar la actualización del sistema.
- Definir un sistema de roles y permisos más granular: Actualmente existe un control básico de acceso al sistema, pero no se distinguen niveles de visibilidad ni permisos específicos por rol. Se recomienda implementar un modelo RBAC (Role-Based Access Control) que permita restringir el acceso a módulos, datos sensibles y acciones según el perfil del usuario. Esta funcionalidad podría implementarse usando el entorno existente de Microsoft 365 o mediante un sistema interno personalizado.
- Extender la plataforma con una aplicación móvil para facilitar la consulta de reportes.
- Integrar herramientas externas de seguimiento de tiempo para importar las horas reales trabajadas por los colaboradores y compararlas con las planificadas, permitiendo refinar estimaciones futuras.
- Implementar un modulo completo de auditoria que registre modificaciones a proyectos y asignaciones, junto con el usuario autor. Actualmente solo se cuenta con el sistema de auditoría a nivel de base de datos.

Estas mejoras permitirían que Proyekta se consolide como la herramienta principal de gestión integral de proyectos en Pyxis. De esta forma, servirá como apoyo estratégico al proporcionar datos clave para la toma de decisiones informadas a lo largo de todo el ciclo de vida de los proyectos.

Referencias

Amazon. (2026). *Amazon web services.* Descargado de https://aws.amazon.com/es/ (Accessed: 2026-04-20)

Anderson, D. J. (2010). *Kanban: Successful evolutionary change for your technology business.* Blue Hole Press.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... Thomas, D. (2001). *Manifesto for agile software development.* Descargado de https://agilemanifesto.org

Docker, I. (2026). *Docker.* Descargado de https://www.docker.com/ (Accessed: 2026-04-20)

Google. (2026). *Kubernetes.* Descargado de https://kubernetes.io/es/ (Accessed: 2026-04-20)

Hardt, D. (2012, octubre). *The OAuth 2.0 Authorization Framework* (n.º 6749). RFC 6749. RFC Editor. Descargado de https://www.rfc-editor.org/info/rfc6749 doi: 10.17487/RFC6749

HashiCorp. (2026). *Terraform.* Descargado de https://developer.hashicorp.com/terraform (Accessed: 2026-04-20)

Kniberg, H., y Skarin, M. (2010). *Kanban and scrum-making the most of both.* C4Media.

Microsoft. (2026). *Azure active directory.* Descargado de https://www.microsoft.com/es-es/security/business/identity-access/microsoft-entra-id (Accessed: 2026-04-20)

OpenID Foundation. (2014). *Openid connect core 1.0.* Descargado de https://openid.net/specs/openid-connect-core-1_0-final.html

Postgresql. (2026). Descargado de https://www.postgresql.org/ (Accessed: 2026-04-20)

Schwaber, K., y Sutherland, J. (2020). *The scrum guide.* Descargado de https://scrumguides.org

Tanzu, V. (2026). *Spring boot.* Descargado de https://spring.io/projects/spring-boot (Accessed: 2026-04-20)

Vercel. (2026). *Next.js.* Descargado de https://nextjs.org/ (Accessed: 2026-04-20)

Anexo A

Historias de Usuario

Usuarios del sistema

Los tipos de usuarios del sistema son los siguientes:

- Administrador del sistema (AS)
- Responsable de Operaciones (RO)
- Responsable de Delivery o Project Manager (PM)

Módulo: Gestión de proyectos

H1 – Registrar proyecto

COMO: AS, RO, PM
QUIERO: Registrar un nuevo proyecto con información básica.
PARA: Incorporarlo al portafolio general.

DESCRIPCIÓN: En el módulo de Proyectos se debe poder crear un nuevo proyecto ingresando los siguientes datos básicos: nombre, descripción, cliente, tipo de contrato, agrupación, estado, referente principal y fechas de inicio y fin. Adicionalmente, se debe permitir definir áreas de colaboración y servicios asociados al proyecto.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. El formulario debe incluir los campos: nombre, descripción, cliente, tipo de contrato, agrupación, estado, referente principal y fechas de inicio y fin.

2. Cliente, tipo de contrato y agrupación deben seleccionarse desde listados existentes, con la opción de crear nuevas entradas si no se encuentran en el listado.

3. El referente principal debe seleccionarse de un listado de colaboradores disponibles.

4. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.

5. Opcionalmente se debe permitir crear áreas para proyecto siguiendo los criterios definidos en “Crear áreas de proyecto”.

6. Opcionalmente se debe permitir crear servicios para el proyecto siguiendo los criterios definidos en “Crear servicios de proyecto”.

H2 – Editar Proyecto

COMO: AS, RO, PM
QUIERO: Editar la información de un proyecto existente.
PARA: Corregir o actualizar la información asociada.

DESCRIPCIÓN: Se debe poder modificar un proyecto ya creado. Esto incluye su información básica y la gestión de áreas y servicios asociados, respetando los permisos y validaciones correspondientes.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe permitir modificar la información básica: nombre, descripción, cliente, tipo de contrato, agrupación, estado, referente principal y fechas de inicio y fin.

2. Cliente, tipo de contrato y agrupación deben seleccionarse desde listados existentes, con la opción de crear nuevas entradas si no se encuentran en el listado.

3. El referente principal debe seleccionarse de un listado de colaboradores disponibles.

4. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.

5. Opcionalmente se debe permitir crear áreas para proyecto siguiendo los criterios definidos en P2.

6. Se debe permitir modificar un área existente del proyecto.

7. Se debe permitir eliminar un área existente del proyecto.

8. Opcionalmente se debe permitir crear servicios para el proyecto siguiendo los criterios definidos en P3.

9. Se debe permitir modificar un servicio existente del proyecto.

10. Se debe permitir eliminar un servicio existente del proyecto, lo cual también debe eliminar todas las colaboraciones asociadas a ese servicio.

11. Si se acorta la duración del proyecto, se deben alertar de que tanto las áreas, servicios y sus colaboraciones que tenían fecha igual a la de proyecto se verán modificadas por este cambio con el fin de mantener la coherencia. Pudiendo confirmar o revertir.

H3 – Listado de proyectos

COMO: AS, RO, PM
QUIERO: Acceder a un dashboard con el estado general de los proyectos.
PARA: Obtener una visión rápida del avance, estado y colaboraciones de los proyectos.

DESCRIPCIÓN: Se debe mostrar un listado de proyectos registrados en el sistema con su información básica. Además, el usuario debe poder aplicar filtros para facilitar el análisis y la toma de decisiones.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Para cada proyecto se debe mostrar: nombre, cliente, estado, agrupación, tipo de contrato y fechas de inicio y fin.
2. Se debe mostrar un listado de las colaboraciones del proyecto, incluyendo el nombre del colaborador, su rol/tipo de servicio, carga horaria diaria y fechas de inicio y fin.
3. Se debe permitir filtrar el listado por nombre, cliente, estado, tipo de contrato, agrupación y fecha.
4. El filtro por nombre debe ser un campo de texto y devolver los proyectos cuyos nombres coincidan total o parcialmente con lo ingresado.
5. Los filtros por cliente, estado, tipo de contrato y agrupación deben ser listas desplegables con las opciones disponibles en el sistema.
6. El filtro por fecha debe devolver los proyectos en curso en la fecha indicada.
7. Por defecto el listado de proyectos, debe mostrar todos los proyectos con estado distinto a “Cancelado”.

H4 – Detalle de proyecto

COMO: AS, RO, PM
QUIERO: Acceder al detalle completo de un proyecto desde el listado
PARA: Consultar toda su información relevante, incluyendo alertas visuales sobre atrasos y asignaciones incompletas.

DESCRIPCIÓN: Al hacer clic sobre un proyecto desde la timeline o el listado, se debe abrir una vista detallada con toda la información relevante del mismo. Además de la información básica, se deben visualizar las áreas y los servicios

requeridos, junto con las colaboraciones incluidas en estos. También se deben visualizar las alertas que advierten sobre posibles problemas operativos.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se mostrará la información básica: nombre, descripción, cliente, agrupación, tipo de contrato, referente, estado y fechas de inicio y fin.
2. Se debe mostrar un listado de las áreas definidas, incluyendo rol, candidato, fecha de inicio, fecha de fin, carga horaria diaria, si está confirmada y si es billable.
3. Se debe permitir definir nuevas áreas desde el listado de áreas, siguiendo los criterios definidos en P2.
4. Se debe permitir editar un área existente desde el listado de áreas.
5. Se debe permitir eliminar un área existente desde el listado de áreas.
6. Se debe mostrar un listado de los servicios definidos, incluyendo tipo de servicio, fecha de inicio, fecha de fin, carga horaria mensual y si está confirmado.
7. Para cada servicio, se debe mostrar un listado con sus colaboraciones, incluyendo el colaborador, fecha de inicio, fecha de fin, carga horaria diaria y si está confirmada.
8. Se debe permitir definir nuevas colaboraciones desde el listado de colaboraciones de un servicio, siguiendo los criterios definidos en P7.
9. Se debe permitir editar una colaboración de un servicio desde su listado de colaboraciones.
10. Se debe permitir eliminar una colaboración de un servicio desde su listado de colaboraciones.

H5 – Timeline de Proyectos

COMO: AS, RO, PM
QUIERO: Visualizar una timeline de los proyectos
PARA: Obtener una vista clara de la duración de los proyectos ingresados en el sistema

DESCRIPCIÓN: Necesito ver una timeline que permite filtrar y visualizar los proyectos para tener un panorama general de los proyectos ingresados en el sistema.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Los ítems de proyectos deben verse como bloques temporales con su fecha de inicio y fin.

2. Se debe permitir filtrar proyectos por nombre, cliente, estado, tipo de contrato, agrupación y fecha.

3. El filtro por nombre debe ser un campo de texto y devolver los proyectos cuyos nombres coincidan total o parcialmente con lo ingresado.

4. Los filtros por cliente, estado, tipo de contrato y agrupación deben ser listas desplegables con las opciones disponibles en el sistema.

5. El filtro por fecha debe devolver los proyectos en curso en la fecha indicada.

6. La timeline tiene que tener los distintos niveles de zoom (mes, trimestre, año)

H6 – Extender Proyecto

COMO: AS, RO, PM
QUIERO: Extender la fecha de fin del proyecto hacia adelante
PARA: arrastrar hacia adelante todas las fechas de fin de las áreas y servicios donde su fecha de fin coincide con la de fin de proyecto hacia adelante

DESCRIPCIÓN: Necesito poder extender la fecha de fin de un proyecto hacia adelante, actualizando la fecha de fin de todas las áreas y servicios donde la fecha coincidía, con el fin de ejecutar en una sola acción y no tener que extender cada área o servicio manualmente.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. La nueva fecha de fin debe ser posterior a la fecha de fin actual

2. Las fechas de fin de áreas que coinciden con la fecha de fin del proyecto deben actualizarse a la nueva fecha de fin

3. Las fechas de fin de los servicios y sus colaboraciones que coinciden con la fecha de fin del proyecto deben actualizarse a la nueva fecha de fin

Módulo: Gestión de áreas y servicios

H7 – Registrar áreas de Proyecto

COMO: AS, RO, PM
QUIERO: Registrar las áreas requeridas a un proyecto.
PARA: Definir la estructura del equipo que se necesita para su ejecución.

DESCRIPCIÓN: Al crear o editar un proyecto, se deben poder agregar áreas que se requieren para llevar adelante el proyecto. Cada área debe contar con un rol, candidato, carga horaria, fechas de inicio y fin, si es billable y si está confirmada.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se deben poder crear múltiples áreas por proyecto.
2. Cada área debe incluir obligatoriamente un rol.
3. Cada área puede registrar un candidato, una carga horaria diaria, un indicador de si es billable, un indicador de si está confirmada y fechas de inicio y fin.
4. El rol debe seleccionarse de un listado de roles registrados, con la posibilidad de crear un nuevo rol desde el mismo listado.
5. El candidato debe seleccionarse de un listado de colaboradores del sistema.
6. Las fechas de inicio y fin se inicializan con las del proyecto si están definidas, pero pueden modificarse.
7. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.
8. Se debe validar que las fechas de inicio y fin del área estén dentro del rango de fechas del proyecto.
9. Cada área guardada debe registrarse como formal en el sistema.
10. En caso de que el rango del área quede por fuera del rango del proyecto, se debe dar la opción de extenderla para mantener la coherencia.

Nota: El concepto de formal indica que una colaboración forma parte del plan del proyecto oficialmente

H8 – Registrar servicios de Proyecto

COMO: AS, RO, PM
QUIERO: Registrar servicios requeridos a un proyecto.
PARA: Definir la estructura del equipo que se necesita para su ejecución.

DESCRIPCIÓN: Al crear o editar un proyecto se deben poder agregar servicios necesarios. Cada servicio debe incluir el tipo de servicio, y opcionalmente si es billable, la carga horaria mensual esperada y las fechas de inicio y fin dentro del marco del proyecto.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se deben poder crear múltiples servicios por proyecto.
2. Cada servicio debe incluir obligatoriamente el tipo de servicio.
3. Se podrá indicar opcionalmente la carga horaria mensual, si es billable y las fechas de inicio y fin.
4. El tipo de servicio debe seleccionarse de un listado de servicios registrados, con posibilidad de crear uno nuevo.

5. Las fechas deben validarse para asegurar que la fecha de inicio sea anterior a la de fin.

6. Las fechas de inicio y fin deben estar comprendidas dentro de la duración del proyecto.

7. Las fechas de inicio y fin toman por defecto las del proyecto si están definidas.

8. En caso de que el rango del servicio quede por fuera del rango del proyecto, se debe dar la opción de extenderlo para mantener la coherencia.

H9 – Crear colaboración para servicio

COMO: AS, RO, PM
QUIERO: agregar a un colaborador a un servicio existente
PARA: Definir la asignación de la persona al proyecto.

DESCRIPCIÓN: Al ver los detalles de un proyecto, se deben poder agregar colaboraciones que se requieren para uno de sus servicios. Cada colaboración debe contar con un colaborador, carga horaria, si está confirmada y fechas de inicio y fin.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se deben poder crear múltiples colaboraciones por servicio.

2. Cada colaboración debe incluir obligatoriamente un colaborador.

3. Se podrá indicar opcionalmente la carga horaria diaria, si está confirmada y las fechas de inicio y fin.

4. El colaborador debe seleccionarse de un listado de colaboradores registrados.

5. Por defecto las fechas de inicio y fin toman el valor de las fechas del servicio, y si estas no están definidas, toman las fechas del proyecto (si están definidas).

6. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.

7. Se debe validar que las fechas de la colaboración estén contenidas dentro de la duración del servicio, o dentro de la duración del proyecto si el servicio no tiene fechas definidas.

8. En caso de superar el rango del servicio o proyecto, se debe dar la opción de extenderlos a el nuevo rango definido con el fin de mantener la coherencia.

Módulo: Colaboradores y asignaciones

H10 – Registrar Colaborador

COMO: AS, RO, PM
QUIERO: Crear una nueva persona con información básica.
PARA: Registrar formalmente un nuevo colaborador dentro del sistema.

DESCRIPCIÓN: Desde el módulo de Colaboradores se debe poder ingresar un nuevo colaborador indicando los datos mínimos necesarios: nombre, capacidad horaria, tecnologías, expertise, seniority, tipo de contrato, fecha de alta y fecha de baja.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. El formulario debe permitir ingresar nombre, capacidad horaria, tecnología, expertise, seniority, tipo de contrato, fecha de alta y fecha de baja.

2. El único campo obligatorio es el nombre.

3. Las tecnologías, expertise, seniority y tipo de contrato deben seleccionarse de un listado con las opciones registradas, permitiendo también crear nuevas entradas en caso de no encontrar la buscada.

4. Opcionalmente se debe dar la opción de indicar que el colaborador se debe ocultar de la timeline.

H11 – Editar Colaborador

COMO: AS, RO, PM
QUIERO: Editar la información de un colaborador existente.
PARA: Corregir o actualizar la información asociada.

DESCRIPCIÓN: Necesito poder modificar un colaborador ya creado. Esto incluye su información básica respetando los permisos y validaciones correspondientes.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe permitir modificar la información básica: nombre, capacidad horaria, tecnologías, expertise, seniority, tipo de contrato, fecha de alta, fecha de baja y ocultar en timeline.

2. El único campo obligatorio es el nombre.

3. Las tecnologías, expertise, seniority y tipo de contrato deben seleccionarse de un listado con las opciones registradas, permitiendo también crear nuevas entradas en caso de no encontrar la buscada.

H12 – Listado de Colaboradores

COMO: AS, RO, PM
QUIERO: Visualizar un listado de todos los colaboradores registrados
PARA: Obtener una visión rápida de la información de los colaboradores

DESCRIPCIÓN: Se debe mostrar un listado de colaboradores registrados en el sistema con su información básica. Además, el usuario debe poder aplicar filtros para facilitar el análisis y la toma de decisiones.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Para cada colaborador se debe mostrar: nombre, horas por día, fecha de alta, fecha de baja, tecnologías asignadas, horizontal, seniority, tipo de contrato y si se muestra en la timeline de colaboradores.

2. Se debe permitir filtrar el listado por nombre, tecnología, horizontal, seniority y tipo de contrato.

3. El filtro por nombre debe ser un campo de texto y devolver los colaboradores cuyos nombres coincidan total o parcialmente con lo ingresado.

4. Los filtros por tecnología, horizontal, seniority y tipo de contrato deben ser listas desplegables con las opciones disponibles en el sistema.

H13 – Timeline de Colaboradores

COMO: AS, RO, PM
QUIERO: Visualizar una timeline que muestre la carga de trabajo de cada colaborador con sus asignaciones actuales, tentativas y licencias
PARA: Poder analizar la distribución del trabajo, identificar sobreasignaciones y tomar decisiones informadas de planificación

DESCRIPCIÓN: Se debe mostrar una timeline que permita visualizar y filtrar las colaboraciones en las que participa cada colaborador, con distintos niveles de detalle y herramientas de gestión asociadas.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. La timeline tiene que tener los niveles de zoom mes, trimestre y año.

2. La vista debe permitir filtrar por nombre, tecnología, expertise, seniority, tipo de contrato, proyecto y estado de la asignación (confirmada o tentativa).

3. El filtro por nombre debe ser un campo de texto y devolver los colaboradores cuyos nombres coincidan total o parcialmente con lo ingresado.

4. Los filtros por tecnología, expertise, seniority y tipo de contrato deben ser listas desplegables con las opciones disponibles en el sistema.

5. El filtro por proyecto debe retornar los datos de todos los colaboradores involucrados en el proyecto seleccionado.

6. La timeline debe incluir una línea de resumen por colaborador que indique el total de horas asignadas a lo largo del tiempo.

7. La línea de resumen debe mostrar visualmente si hay subasignación, sobreasignación o asignación justa.

8. La línea de resumen debe incluir los periodos donde el colaborador tiene licencia.

9. La timeline debe contar con un panel lateral con el nombre del colaborador y sus datos clave (tecnologías, expertise, seniority, tipo de contrato y horas diarias).

10. El panel lateral correspondiente a un colaborador debe incluir la opción de crear un área con el colaborador seleccionado como candidato, como se detalla en C5.

11. El panel lateral correspondiente a un colaborador debe incluir la opción de asignar al colaborador a un servicio ya existente, como se detalla en C6.

12. El panel lateral correspondiente a un colaborador debe incluir la opción de crear licencia para el colaborador, como se detalla en C7.

13. Al hacer clic en un colaborador, deben desplegarse sus asignaciones individuales y períodos de licencia debajo de su línea principal.

14. Las barras asignaciones deben tener un indicador visual que distinga entre confirmadas y tentativas.

15. Se debe poder editar la duración de una colaboración o licencia arrastrando los bordes de la barra correspondiente.

16. En el panel lateral de cada asignación debe indicarse el área/servicio y el proyecto al que pertenece.

17. En el panel lateral de cada segmento de licencia debe indicarse que la entrada corresponde a licencia.

18. El panel lateral correspondiente a una asignación/licencia debe incluir un botón que permita editar la asignación/licencia, como se detalla en C8, C9 y C10.

19. El panel lateral correspondiente a una asignación/licencia debe incluir un botón que permita eliminar la asignación/licencia.

H14 – Crear área desde timeline colaboradores

COMO: AS, RO, PM
QUIERO: Crear un área de proyecto para un colaborador directamente desde la timeline de colaboradores.
PARA: Asignar a la persona a un área de un proyecto existente de manera rápida y contextual

DESCRIPCIÓN: Desde la timeline de colaboradores se debe permitir crear un área en un proyecto seleccionado, quedando el colaborador asignado como candidato del área.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe poder seleccionar el proyecto al que le corresponde el área desde un listado con los nombres de los proyectos registrados.
2. Se debe poder seleccionar el rol desde un listado con los roles definidos en el sistema, permitiendo que el usuario defina un nuevo rol.
3. Opcionalmente el usuario puede definir la carga horaria diaria, las fechas de inicio y fin, si está confirmada y si es billable.
4. El colaborador seleccionado desde la timeline debe quedar registrado como candidato para el área.
5. Por defecto el campo confirmado es verdadero, salvo que el proyecto sea tentativo, en cuyo caso por defecto es no confirmado.
6. Por defecto el campo billable es verdadero.
7. Por defecto, las fechas de inicio y fin toman el valor de las fechas del proyecto al que pertenece el área a crear (si es que están definidas).
8. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.
9. Se debe validar que las fechas seleccionadas están dentro de la duración del proyecto.
10. El área debe registrarse como informal.
11. En caso de que se superen las fechas del proyecto, se debe permitir extenderlo para asegurar la consistencia

H15 – Asignar a servicio desde timeline colaboradores

COMO: AS, RO, PM
QUIERO: Asignar un colaborador a un servicio existente desde la timeline de colaborador.
PARA: Asignar a la persona a un servicio de un proyecto existente de manera rápida y contextual

DESCRIPCIÓN: Desde la timeline de colaboradores se debe permitir crear una colaboración en un servicio de un proyecto seleccionado, quedando el colaborador asignado como candidato de dicha colaboración.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe poder seleccionar el proyecto desde un listado con los nombres de los proyectos registrados.
2. Se debe poder seleccionar el servicio desde un listado con los servicios definidos para el proyecto.
3. Opcionalmente se debe poder definir la carga horaria diaria, las fechas de inicio y fin y si la colaboración está confirmada.
4. Por defecto el campo confirmado es verdadero, salvo que el proyecto sea tentativo, en cuyo caso por defecto es no confirmado.
5. Por defecto las fechas de inicio y fin toman el valor de las fechas del servicio, y si estas no están definidas, toman las fechas del proyecto (si están definidas).
6. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.
7. Se debe validar que las fechas de la colaboración estén contenidas dentro de la duración del servicio, o dentro de la duración del proyecto si el servicio no tiene fechas definidas.
8. La colaboración debe registrarse como informal.
9. En caso de que se superen las fechas del servicio o proyecto, se debe permitir extenderlo/s para asegurar la consistencia

H16 – Asignar licencia desde timeline colaboradores

COMO: AS, RO, PM
QUIERO: Asignar licencia para un colaborador desde la timeline de colaborador.
PARA: Registrar y visualizar las ausencias de un colaborador en un período correspondiente.

DESCRIPCIÓN: Se debe mostrar una timeline que permita visualizar el tiempo libre de cada colaborador, con distintos niveles de detalle y herramientas de gestión asociadas.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe poder seleccionar la fecha de inicio y fin para la licencia a registrar.
2. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.

3. La licencia debe quedar registrada para el colaborador seleccionado en la timeline.

4. La licencia debe visualizarse como un bloque en la timeline del colaborador.

H17 – Editar área desde timeline colaboradores

COMO: AS, RO, PM
QUIERO: editar los datos de un área ya registrada directamente desde la timeline de colaboradores.
PARA: mantener actualizada y corregida la información de asignaciones sin necesidad de salir de la vista de timeline.

DESCRIPCIÓN: Desde la timeline de colaboradores se debe poder editar los campos de un área ya existente, incluyendo rol, candidato, fechas de inicio y fin, carga horaria diaria, si está confirmada y si es billable.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe habilitar la edición de área desde la timeline mediante un formulario asociado al bloque seleccionado.

2. Se debe permitir modificar el rol, candidato, fecha de inicio, fecha de fin, carga horaria diaria, si está confirmada y si es billable.

3. Se debe validar que el rol no quede vacío al guardar los cambios.

4. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.

5. Se debe validar que las fechas seleccionadas están dentro de la duración del proyecto.

6. En caso de que se superen las fechas del proyecto, se debe permitir extenderlo para asegurar la consistencia

H18 – Editar asignación a servicio desde timeline colaboradores.

COMO: AS, RO, PM
QUIERO: editar los datos de una asignación de colaborador a un servicio directamente desde la timeline de colaboradores.
PARA: mantener actualizada la información de las colaboraciones de manera rápida y sin salir de la vista de timeline.

DESCRIPCIÓN: Desde la timeline de colaboradores se debe poder editar los datos de una asignación existente a un servicio, incluyendo el colaborador, la carga horaria diaria, las fechas de inicio y fin y si está confirmada.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe habilitar la edición de asignaciones a servicios desde la timeline mediante un formulario o modal asociado al bloque seleccionado.
2. Se debe permitir modificar el colaborador, la carga horaria diaria, la fecha de inicio, la fecha de fin y si está confirmada.
3. Se debe validar que siempre se seleccione un colaborador.
4. Se debe validar que la fecha de inicio sea anterior a la fecha de fin.
5. Se debe validar que las fechas de la colaboración estén contenidas dentro de la duración del servicio, o dentro de la duración del proyecto si el servicio no tiene fechas definidas.
6. En caso de que se superen las fechas del servicio o proyecto, se debe permitir extenderlo/s para asegurar la consistencia

H19 – Editar licencia desde timeline colaboradores.

COMO: AS, RO, PM
QUIERO: editar los datos de un segmento de licencia de un colaborador desde la timeline de colaboradores.
PARA: mantener actualizada la información del colaborador de manera rápida y sin salir de la vista de timeline.

DESCRIPCIÓN: Desde la timeline de colaboradores se debe poder editar la duración de un periodo de licencia de un colaborador.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe poder modificar la fecha de inicio y fin de la licencia.

H20 – Timeline de tiempo libre

COMO: AS, RO, PM
QUIERO: Visualizar una timeline que muestre el tiempo libre de cada colaborador en rangos de tiempo
PARA: Poder analizar la distribución del trabajo, identificar recursos libres para próximos proyectos.

DESCRIPCIÓN: Se debe mostrar una timeline que permita visualizar el tiempo libre de cada colaborador, con distintos niveles de detalle, para poder planificar futuras asignaciones.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. La timeline tiene que tener los niveles de zoom mes, trimestre y año.
2. La timeline debe incluir una línea de resumen por colaborador que indique el total de horas libre (horas contractuales - horas libre) a lo largo del tiempo.

3. La línea de resumen debe incluir los periodos donde el colaborador tiene licencia.

4. La timeline debe contar con un panel lateral con el nombre del colaborador y sus datos clave (tecnología, expertise, seniority, tipo de contrato y horas diarias).

5. Se debe permitir seleccionar una fecha para poder ver solamente el tiempo libre de los colaboradores en la quincena correspondiente.

Módulo: Reportes

H21 – Reporte de proyectos por finalizar

COMO: AS, RO, PM
QUIERO: Un listado con los proyectos que finalizan la próxima quincena.
PARA: Poder evaluar los colaboradores que se liberarán para próximos proyectos.

DESCRIPCIÓN: Se debe generar un reporte que muestre los proyectos cuya fecha de fin se encuentra dentro de la próxima quincena. El reporte debe permitir visualizar datos básicos de cada proyecto y opcionalmente incluir detalles de áreas y servicios asociados.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se deben listar todos los proyectos cuya fecha de fin cae dentro de la quincena siguiente al momento de consultar el reporte.

2. Se deben mostrar los datos básicos de cada proyecto por finalizar: nombre, referente, cliente, estado, tipo de contrato, agrupación y fechas de inicio y fin.

3. Opcionalmente se deben mostrar las áreas registradas para el proyecto, incluyendo rol, carga horaria diaria, fechas de inicio y fin, y colaborador, junto con sus tecnologías, expertise y seniority.

4. Opcionalmente se deben mostrar los servicios registrados para el proyecto, incluyendo para cada servicio su tipo y un listado de colaboraciones. Cada colaboración debe incluir colaborador, fechas de inicio y fin y carga horaria diaria; además, se deben mostrar los datos del colaborador: tecnologías, expertise y seniority.

5. Se debe permitir exportar un listado simple con los datos básicos de proyectos, o uno completo, incluyendo las áreas y servicios definidos para cada uno.

6. Los datos deben exportarse en formato Excel (.xlsx).

H22 – Reporte de colaboradores libres

COMO: AS, RO, PM
QUIERO: Un listado con los colaboradores con tiempo libre en una quincena dada.
PARA: Poder detectar recursos libres al momento de planificar.

DESCRIPCIÓN: Se debe generar un reporte que muestre los colaboradores con tiempo libre en una quincena dada. El reporte debe permitir visualizar datos básicos de cada colaborador y la cantidad de horas disponibles.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se deben listar todos los colaboradores con tiempo libre en la quincena seleccionada, considerando tiempo contractual menos asignaciones.
2. Para cada colaborador se debe mostrar nombre, tecnologías, expertise, seniority, la cantidad de horas libres y la fecha en la que tuvo su última asignación completa (0 horas libres).
3. Se debe permitir filtrar el listado por fecha, tecnología, expertise, seniority y tipo de contrato.
4. Los filtros de tecnología, expertise, seniority y tipo de contrato deben ser listas con las opciones registradas en el sistema.
5. La fecha seleccionada en el filtro define la quincena a evaluar.
6. Por defecto, la fecha del filtro debe inicializarse en el primer día de la quincena siguiente al momento de consultar el reporte.
7. Se debe permitir exportar los datos del resumen.
8. El exportado puede hacerse con o sin los filtros.
9. Se debe permitir exportar los datos en formato Excel (.xlsx) o CSV (.csv).
10. Se debe indicar la suma de horas libres o FT de todos los colaboradores

H23 – Reporte de total equipo disponible por período de tiempo

COMO: AS, RO, PM
QUIERO: Un reporte que muestre la disponibilidad total del equipo por período de tiempo (quincena).
PARA: Poder visualizar de forma clara el nivel de asignación y desasignación en cada período y tomar decisiones de planificación.

DESCRIPCIÓN: Se debe generar un reporte gráfico que muestre, para cada período de tiempo (quincena), el porcentaje de asignación y desasignación de la

capacidad total del equipo. La visualización debe permitir entender la evolución de la disponibilidad a lo largo de diferentes períodos.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe mostrar un gráfico con períodos (quincenas) en el eje temporal.

2. Se debe poder mostrar el tiempo libre en horas y en FT (horas/8).

3. Para cada período se debe visualizar el porcentaje de asignación y desasignación del equipo.

4. Se debe permitir exportar la información subyacente en formato Excel (.xlsx) o CSV (.csv).

5. Se debe permitir exportar el gráfico como imagen.

H24 – Reporte de total equipo disponible por horizontal

COMO: AS, RO, PM
QUIERO: Un reporte que muestre la disponibilidad total del equipo por período de tiempo subdividida por horizontal.
PARA: Poder identificar con precisión la evolución de la disponibilidad de recursos según la especialidad.

DESCRIPCIÓN: Se debe generar un reporte gráfico que muestre, para cada período de tiempo (quincena), la disponibilidad total subdividida por expertise. El gráfico debe permitir apreciar cómo evoluciona la disponibilidad de cada expertise a lo largo de los diferentes períodos.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Se debe mostrar un gráfico de áreas apiladas por expertise, con períodos (quincenas) en el eje temporal.

2. Cada capa del gráfico debe representar la disponibilidad de un expertise en horas o en FT (horas/8).

3. Se debe permitir identificar la disponibilidad total y la correspondiente a cada expertise para un período específico.

4. Se debe permitir ocular los datos de expertises para enfocarse en otro subconjunto.

5. Se debe permitir exportar la información subyacente en formato Excel (.xlsx) o CSV (.csv).

6. Se debe permitir exportar el gráfico como imagen.

Módulo: Configuración de catálogo

Las historias de este módulo aplican para las siguientes entidades: Estado de Proyecto, Cliente, Tipo de Contrato de Proyecto, Agrupación, Tipo de contrato de Colaborador, Seniority, Tecnología, Expertise, Tipo de Servicio, Rol y Usuario.

H25 – Listar entidad

COMO: AS, RO, PM
QUIERO: Visualizar un listado con los valores de las entidad disponibles en el sistema
PARA: Obtener una visión general de las entidades configuradas y gestionarlas fácilmente.

DESCRIPCIÓN: Se debe mostrar un listado con todos los valores de la entidad disponibles en el sistema.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. Para cada entidad se debe mostrar el nombre, y dar la posibilidad de editarla o borrarla

H26 – Agregar entidad

COMO: AS, RO, PM
QUIERO: Crear un nuevo valor de la entidad
PARA: Incorporar nuevas opciones al catálogo para su uso en el sistema.

DESCRIPCIÓN: Se debe poder crear un nuevo valor para la entidad.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. No se debe poder crear una nueva entidad con el mismo nombre que otra.

H27 – Editar entidad

COMO: AS, RO, PM
QUIERO: Editar un valor de una entidad
PARA: corregir el valor que previo

DESCRIPCIÓN: Se debe poder editar el valor de una entidad.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. No se debe poder editar una nueva entidad con el mismo nombre que otra.

H28 – Eliminar entidad

COMO: AS, RO, PM
QUIERO: Eliminar el valor de una entidad
PARA: no visualizarlo mas en los listados del sistema

DESCRIPCIÓN: Se debe poder eliminar el valor de una entidad.

CRITERIOS DE ACEPTACIÓN FUNCIONALES:

1. No se debe visualizar en los listados la opción de la entidad eliminada.