UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Ambiente de datos en metodologías híbridas

Informe de Proyecto de Grado presentado por

Gimena Caroceli y Maximiliano Ustria

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

Supervisores

Regina Motz
Edelweis Rohrer

Montevideo, Noviembre 2025

Resumen

Este trabajo desarrolla un ambiente de datos para metodologías de investigación híbridas, integrando la gestión y el análisis de datos cualitativos y cuantitativos en un único sistema colaborativo. Se relevaron requerimientos con equipos de investigación, se diseñó una solución basada en una arquitectura monolítica con Ruby on Rails y una interfaz web centrada en los flujos de trabajo de proyectos, marcos teóricos, instrumentos y registros. La implementación incluye almacenamiento de documentos y encuestas, codificación de fragmentos, buscador de códigos y bitácora de eventos, así como soporte para colaboración por roles. Se aplicaron prácticas de ingeniería (control de versiones, contenedorización y despliegue) y pruebas manuales y de rendimiento. Los resultados muestran que la herramienta facilita el cruce de códigos entre múltiples recursos (entrevistas, encuestas y documentos) y mejora la trazabilidad del análisis respecto a soluciones dispersas. Finalmente, se discuten limitaciones (interoperabilidad, visualización avanzada) y líneas de trabajo futuro para ampliar métricas y la integración con otras plataformas.

Palabras clave: ambiente de datos, metodologías híbridas, investigación, análisis cualitativo, Ruby on Rails

Agradecimientos

Queremos expresar nuestro agradecimiento a nuestras tutoras, Regina Motz y Edelweis Rohrer, por su guía, apoyo y compromiso a lo largo del desarrollo del proyecto.
A nuestros profesores, compañeros, familia y amigos que nos apoyaron y motivaron a lo largo de estos años, sin ellos este trabajo no hubiera sido posible.

Índice general

Capítulo 1

Introducción

Este proyecto se centra en desarrollar un ambiente de gestión de datos para metodologías de investigación híbridas, es decir, un entorno digital que permita gestionar y analizar datos de manera eficiente en proyectos de investigación que combinan enfoques cualitativos y cuantitativos. Este tipo de metodologías híbridas es utilizado frecuentemente en investigaciones que buscan no solo describir fenómenos sociales sino también entender sus significados profundos y complejos. Por lo tanto, el ambiente de datos busca proporcionar a los investigadores las herramientas necesarias para capturar, organizar y analizar ambos tipos de datos en un solo sistema integrado.

Metodologías de investigación híbridas son enfoques que combinan métodos cualitativos y cuantitativos para estudiar fenómenos desde múltiples perspectivas. En estos proyectos, se recopilan tanto datos numéricos como narrativos, y su integración permite obtener resultados más enriquecedores y completos. El propósito de crear un ambiente de datos adecuado es optimizar el manejo de esta información diversa y facilitar su análisis en investigaciones científicas que requieren de la flexibilidad y profundidad que ofrece la metodología híbrida. (Creswell y Plano Clark, 2017)

Ambiente de datos se refiere al entorno digital que centraliza y facilita la recopilación, almacenamiento y análisis de datos de investigación. En este caso, se espera que dicho ambiente ofrezca soporte para almacenar datos en diversos formatos, que permita la colaboración entre investigadores, y que facilite el análisis y la visualización de patrones y tendencias.

En este proyecto la investigación y relevamiento de requerimientos, se rea-

lizaron con la ayuda de la investigadora Rosalía Winocur en la Facultad de Información y Comunicación (FIC) de la Universidad de la República. En sus investigaciones generalmente utiliza un ambiente de datos diseñado para almacenar y gestionar los datos obtenidos tanto de entrevistas y observaciones como de cuestionarios y encuestas. Esta base de datos se implementa actualmente mediante herramientas de almacenamiento como Google Drive, lo que permite una cierta organización de los documentos recolectados y su acceso compartido, aunque presenta algunas limitaciones.

Los puntos fuertes de este ambiente de datos incluyen su accesibilidad y facilidad de uso para el almacenamiento de los recursos de investigación. No obstante, se ha identificado una limitación en la capacidad de visualización y organización de los datos. Actualmente, la estructura dificulta la rápida identificación de patrones y la generación de nuevas ideas a partir de la información recolectada.

Este proyecto propone diseñar un ambiente de datos que aborde estas limitaciones, facilitando la organización, gestión, colaboración, el almacenamiento y el análisis integrado de datos cualitativos y cuantitativos.

Para solucionar estas limitaciones, proponemos desarrollar una herramienta web que facilite la gestión de los datos en un entorno centralizado e integrador. Los objetivos específicos son:

- **Análisis cualitativo**: Ofrecer la posibilidad de analizar y cruzar datos permitiendo visualizaciones dinámicas y adaptadas al tipo de datos.
- **Almacenamiento y organización eficientes**: Implementar un sistema de almacenamiento que permita una mejor organización y clasificacion de los datos según sus tipos y sus relaciones con otros datos, y que facilite la creación de vínculos entre entrevistas y observaciones.
- **Colaboración en tiempo real y accesibilidad**: Crear un sistema de trabajo colaborativo en tiempo real que permita a los investigadores compartir y modificar sus análisis y hallazgos dentro del mismo ambiente de datos, promoviendo una interacción constante y una contribución activa al proyecto. Asegurando la accesibilidad para todos los investigadores, independientemente de su ubicación, habilidades técnicas, fomentando así una participación plena en el proyecto.

Capítulo 2

Antecedentes

2.1. Descripción del Dominio

Las investigaciones con metodologías híbridas suelen seguir una estructura flexible de fases generales, que se adapta en función de cada caso particular. En esta sección, definimos las fases más comunes de un proyecto de investigación y la manera en que el ambiente de datos propuesto debe ajustarse a cada una para maximizar su funcionalidad.

- **Fase Inicial**: Define la hipótesis inicial, objetivos y participantes. Es aquí donde se establecen los fundamentos del proyecto y se clarifican las metas a alcanzar.
- **Recolección de Datos**: Se recopilan tanto datos cualitativos como cuantitativos, provenientes de entrevistas, encuestas, observaciones, entre otros métodos. La herramienta debe permitir un almacenamiento eficiente de datos heterogéneos.
- **Análisis de Datos**: La etapa de análisis integra ambos tipos de datos en un entorno de trabajo común, donde los investigadores puedan generar códigos, identificar patrones, y realizar cruces de información entre conjuntos de datos distintos.
- **Publicación de Resultados**: Finalmente, el proyecto debe ofrecer opciones para exportar los análisis y resultados, y publicar los hallazgos en formatos de fácil acceso para la comunidad académica.

(Creswell y Plano Clark, 2017)

2.2. Herramientas Actuales y Valoración

Actualmente, existen herramientas como Atlas.ti y Google Drive, que permiten gestionar ciertos aspectos del análisis de datos, aunque con limitaciones. Atlas.ti facilita el analisis de datos cualitativos pero tiene algunas limitaciones a la hora de gestion de un proyecto. Google Drive facilita el almacenamiento y el acceso compartido, pero carece de funcionalidades específicas para análisis de datos. Por ultimo queriamos mencionar el framework OSF, que provee una optima forma de gestionar un proyecto de investigacion pero carece en los aspectos de analisis. Este proyecto plantea una solución que combine las ventajas de ambos tipos de herramientas, además de integrar capacidades de análisis híbrido y colaboración.

- **Atlas.ti**

  Atlas.ti es una herramienta de software que organiza grandes cantidades de datos cualitativos y permite analizar, contrastar y gestionar dicha información. Es útil para organizar proyectos de investigación complejos con múltiples fuentes de datos en diferentes formatos, como texto y video. (*Atlas ti*, s.f.)

- **OSF**

  OSF es una plataforma de código abierto que facilita la colaboración en investigación científica. Ofrece opciones de gestión de proyectos, sincronización con otras herramientas, y visualización de estadísticas del proyecto, permitiendo etiquetar y comentar los proyectos para la interacción con otros investigadores. (*OSF*, s.f.)

- **Google Drive**

  Google Drive es una plataforma de almacenamiento en la nube que permite a los investigadores almacenar y compartir archivos de manera sencilla. Aunque es limitada en cuanto a las funciones específicas de investigación, es frecuentemente utilizada para almacenar datos recolectados. (*Google drive*, s.f.)

Capítulo 3

Requerimientos

3.1. Relevamiento de requerimientos

Para el relevamiento de requerimientos del proyecto, se lleva a cabo un seguimiento semanal con las tutoras asignadas, además de una reunión mensual con los equipos de los proyectos pilotos para la definición del framework, liderados por las investigadoras: Adriana Gewerc, Rosalia Winocur y Mariana Porta. Durante estas reuniones mensuales, se presenta el progreso del framework, se analizan sus opiniones y se obtiene su retroalimentación para asegurar que el producto se adapte a sus necesidades. La participación activa de estos investigadores es esencial tanto para la priorización como para la incorporación de nuevos casos de uso, ya que como usuarios finales, sus aportes desempeñan un papel crucial en el desarrollo del proyecto.

3.2. Requerimientos funcionales

Los requerimientos funcionales fueron especificados en iteraciones con los investigadores y tutoras, y posteriormente se modelaron utilizando casos de uso para detallar las funcionalidades necesarias del sistema.

El diseño del frontend y el backend se fundamentó en los casos de uso. En este enfoque, el modelo de datos se desarrolla a partir de la comprensión del dominio del problema, mientras que las interfaces de usuario y la lógica de ne-

gocio se diseñan e implementan para cumplir con los requerimientos funcionales identificados.

3.2.1. Sesión

RF1.1: Iniciar sesion

- **Descripción:** Este requerimiento funcional se refiere a la capacidad del sistema de permitir a los usuarios autenticarse proporcionando credenciales válidas (email y contraseña) para acceder a la aplicación.

- **Precondiciones:**
  El usuario debe haberse registrado previamente en la aplicación.

- **Flujo de Usuarios:**
  El usuario accede a la página de inicio de sesión.
  El usuario proporciona sus credenciales (email y contraseña).
  El sistema verifica la autenticidad de las credenciales.
  Si las credenciales son válidas, el usuario obtiene acceso a su cuenta y se le redirige a una página de inicio de la aplicación.
  Si las credenciales son inválidas, el sistema muestra un mensaje de error y da al usuario la oportunidad de intentar nuevamente.

- **Diagrama de Secuencia del Sistema**

  El siguiente diagrama de secuencia (Figura 3.1) ilustra el flujo de interacción entre el usuario y el sistema durante el proceso de inicio de sesión.

RF1.2: Registro usuario

- **Descripción:** Este requerimiento funcional se refiere a la capacidad del sistema de permitir a los usuarios (investigadores) crear una cuenta en la aplicación proporcionando:

  - email,

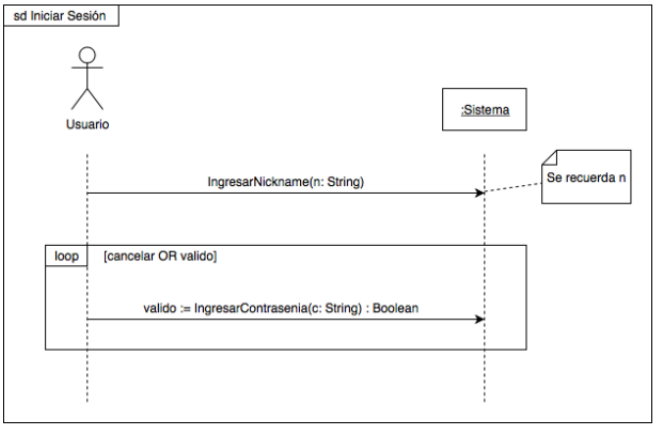

Figura 3.1: Diagrama de secuencia: Iniciar sesión

- nombre
- apellido,
- contraseña,
- confirmación de contraseña

- **Precondiciones:**
  No tiene precondiciones.

- **Flujo de Usuarios:**
  El usuario accede a la página de registro de sesión.
  El usuario completa el formulario con los datos correspondientes.
  Si los datos son válidos, se le crea una cuenta al usuario y redirige a la página de inicio de sesión de la aplicación.
  Si los datos son inválidos, el sistema muestra un mensaje de error y da al usuario la oportunidad de intentar nuevamente.

3.2.2. Proyecto

RF2.1: Crear Proyecto

- **Descripción:**
  Capacidad del sistema de permitir a un usuario crear un proyecto en la aplicación proporcionando los datos necesarios:

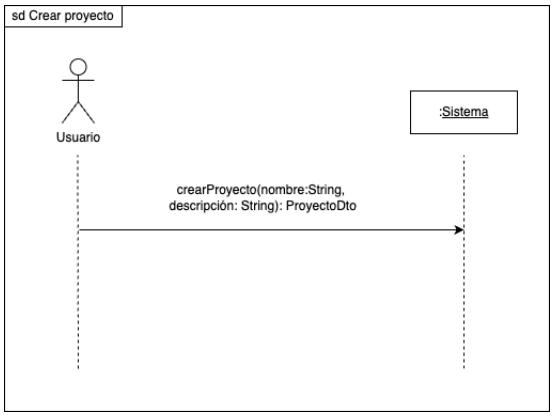

Figura 3.2: Diagrama de secuencia crear proyecto

- nombre: representa el nombre que tendrá el proyecto en el sistema
- descripción: representa una breve descripción del proyecto.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.

- **Flujo de Usuarios:**
  El usuario accede a la página crear proyecto.
  El usuario proporciona los datos para el nuevo proyecto.
  El sistema verifica los datos.
  Si los datos son válidos, el proyecto es creado y se redirige al usuario a la página de ver proyecto.
  Si los datos son inválidos, el sistema muestra un mensaje de error y da al usuario la oportunidad de intentar nuevamente.

- **Diagrama de Secuencia del Sistema**
  El siguiente diagrama de secuencia (Figura 3.2) ilustra el flujo de interacción entre el usuario y el sistema durante el proceso de crear un proyecto.

RF2.2: Usuarios y roles

El usuario creador del proyecto obtendra el rol de administrador en el mismo. Los roles que un usuario puede tener dentro de un proyecto son:

- **Administrador:**
  Creador del proyecto. Permite realizar todas las acciones dentro del proyecto.

- **Editor:**
  Debe recibir una invitación de editor al proyecto, una vez aceptada podrá editar todo el proyecto.

- **Comentador:**
  Debe recibir una invitación de comentador al proyecto, una vez aceptada podrá visualizar todo el proyecto pero no editar el mismo.

RF2.3: Explorar proyectos

- **Descripción:**
  Capacidad del sistema de permitir a un usuario(investigador) ver todos los proyectos en los que participa.
  Muestra datos generales de cada proyecto: nombre, descripción, fase, administrador, cantidad de participantes y fecha de creación.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.

RF2.4: Ver proyecto general

- **Descripción:**
  Capacidad del sistema de permitir a un usuario ver un proyecto en la aplicación. Esta sería la página principal de un proyecto en la que se encuentran la información general del mismo. Muestra los datos del proyecto: nombre, descripción, fase, marcos teóricos, y bitácora.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.

RF2.5: Actualizar fase

- **Descripción:**
  Capacidad del sistema de permitir a un usuario actualizar la fase en la que se encuentra un proyecto en la aplicación.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página cambiar fase.
  El usuario elige una fase entre las fases creadas del proyecto.
  Se redirige al usuario a ver proyecto general, en caso de que el cambio sea exitoso se visualiza la nueva fase, en caso de error el sistema muestra un mensaje de error.

RF2.6: Crear fase

- **Descripción:**
  Capacidad del sistema de permitir a un usuario crear una nueva fase para el proyecto.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página crear fase.
  El usuario proporciona un nombre para la nueva fase.
  Se redirige al usuario a ver configuración del proyecto, en caso de éxito la nueva fase queda disponible, en caso de error el sistema muestra un mensaje de error.

RF2.7: Agregar marco teórico

- **Descripción:**
  Capacidad del sistema de permitir a un usuario agregar un nuevo marco teórico al proyecto, proporcionando los datos necesarios:
  - nombre: representa el nombre que tendrá el marco teórico en el proyecto,
  - descripción: representa una breve descripción
  - documento: opcional, representa un documento asociado al marco teórico, los formatos aceptados son doc o pdf.
- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.
- **Flujo de Usuarios:**
  El usuario accede a la página crear marco teórico.
  El usuario proporciona un nombre, descripción y, opcionalmente, selecciona un documento desde su ordenador.
  Se redirige al usuario a ver proyecto general, en caso de éxito, se visualiza el nuevo marco teórico, en caso de error el sistema muestra un mensaje de error.

RF2.8: Agregar investigador

- **Descripción:**
  Capacidad del sistema de permitir a un usuario agregar un nuevo investigador a un proyecto.
  El usuario debe especificar qué rol tendrá el nuevo investigador y el email del mismo.
- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página, agregar investigador.
  El usuario proporciona un email, y selecciona uno de los roles posibles, editor o comentador.
  Se redirige al usuario a ver participantes del proyecto, al nuevo investigador le llegará un mail a su email informándole sobre la invitación al proyecto, y en caso de que dicho investigador ya tenga una cuenta en el sistema observará una nueva solicitud en la página invitaciones pendientes.

RF2.19Aceptar invitación a proyecto

- **Descripción:**
  Capacidad del sistema de permitir a un usuario aceptar la invitación a un proyecto.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener una invitación al proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página de invitaciones pendientes.
  El usuario puede aceptar o rechazar cada invitación.

RF2.10Ver participantes

- **Descripción:**
  Capacidad del sistema de permitir a un usuario ver los investigadores participantes e invitados de un proyecto en la aplicación. De cada investigador participante muestra: nombre, email, rol (administrador, editor, comentador) y estado (aceptado, pendiente, rechazado).

  También permite buscar entre los investigadores participantes e invitados de un proyecto en la aplicación. Debe permitir filtrar por email, fase y/o rol.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página ver participantes.
  Se le muestra al usuario un buscador por texto para email, un selector con los estados de los participantes y un selector con los roles de los participantes.
  Al seleccionar o escribir y buscar se le muestra los participantes que cumplen las condiciones de la búsqueda.

3.2.3. Instrumento

RF3.1Agregar Instrumento a proyecto

- **Descripción:**
  El sistema permitirá a un usuario crear instrumentos asignando determinados atributos del mismo.
  - nombre: representa el nombre del instrumento en el proyecto.
  - Descripción: breve descripción del instrumento.
  - tipo de instrumento: Encuesta, entrevista o customizado permitiendo seleccionar preguntas o cuadros de texto.
  - Preguntas o secciones: permite agregar las preguntas o secciones que debe presentar el instrumento.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página agregar instrumento.

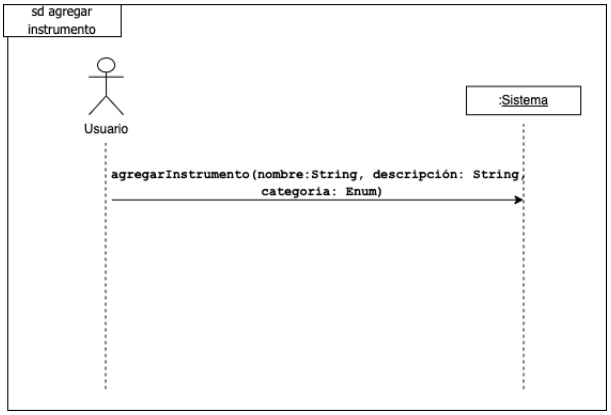

Figura 3.3: Diagrama de secuencia agregar instrumento

El usuario debe completar el formulario con los datos requeridos.
En caso de que los datos provistos por el usuario estén correctos, se crea el instrumento y se redirige al usuario a ver instrumentos donde se visualiza el nuevo instrumento.
En caso de algún error se redirigirá al usuario a ver instrumentos y se le mostrará un mensaje de error.

- **Diagrama de Secuencia del Sistema**
  El siguiente diagrama de secuencia (Figura 3.3) ilustra el flujo de interacción entre el usuario y el sistema durante el proceso de agregar un instrumento a un proyecto.

RF3.2Eliminar Instrumento a proyecto

- **Descripción:**
  El sistema permitirá a un usuario eliminar instrumentos ya creados.
- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.
  Debe existir en el proyecto el instrumento que se desea borrar.

RF3.3Ver instrumento

- **Descripción:**
  El sistema permitirá a un usuario ver un instrumento. La aplicación deberá mostrar el nombre del instrumento, descripción, tipo y sus preguntas o secciones.

  También deberá mostrar el listado de códigos vinculados a ese instrumento con su nombre y descripción.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.
  Debe existir en el proyecto el instrumento.

RF3.4Ver instrumentos

- **Descripción:**
  El sistema permitirá a un usuario ver el listado de instrumentos de un proyecto. Por cada uno se mostrará: nombre, descripción y tipo.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.

3.2.4. Código

Los códigos hacen referencia a clarificar o categorizar la información de manera más comprensible. Por ejemplo, si en un registro se encuentra la frase "lloro de felicidad"se le podría asociar el código .^{em}oción". Además, es posible definir una jerarquía de códigos para organizar mejor la información. En esta jerarquía, un código principal puede tener subcódigos que especifican detalles más concretos relacionados con el tema general del código principal.

Cada código tiene nombre, descripción y, en caso de ser necesario, si es un subcódigo de otro código.

RF4.1Crear código

- **Descripción:**
  Capacidad del sistema de permitir a un usuario crear un nuevo código para el proyecto.

  El código debe presentar nombre y descripción.

  Los códigos pueden ser asociados a registros y/o marcos teóricos.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página crear código.
  El usuario proporciona un nombre y descripción para el nuevo código.
  Se redirige al usuario a ver configuración del proyecto, en caso de éxito se observa el nuevo código en el listado de códigos, en caso de error el sistema muestra un mensaje de error.

RF4.2Importar códigos

- **Descripción:**
  Capacidad del sistema de permitir a un usuario crear códigos en el proyecto a partir de un documento excel.

  El documento debe tener formato excel y contener dos columnas, nombre y descripción.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página importar código.
  El usuario proporciona un documento de formato csv desde su ordenador, que cumpla las características necesarias.
  Se redirige al usuario a ver configuración del proyecto, en caso de éxito, se observan los nuevos códigos en el listado de códigos, en caso de error el sistema muestra un mensaje de error.

RF4.3Unir codigos

- **Descripción:**
  Capacidad del sistema de permitir a un usuario unificar dos códigos en un nuevo código.

  El usuario debe especificar los dos códigos que quiere convertir en uno nuevo y a éste definirle un nombre y descripción.

  Todas los registros, instrumentos y marcos teóricos relacionados con esos códigos se actualizan con el nuevo código.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.
  Deben existir al menos dos códigos en el sistema.

- **Flujo de Usuarios:**
  El usuario accede a la página unir códigos.
  El usuario selecciona los dos códigos a unificar y provee un nuevo nombre y descripción.
  Se redirige al usuario a ver configuración del proyecto, en caso de éxito se observa el nuevo código y la eliminación de los anteriores en el listado de códigos, en caso de error el sistema muestra un mensaje de error.

RF4.4Ver códigos

- **Descripción:**
  Capacidad del sistema de permitir a un usuario ver el listado de códigos

existentes en un proyecto y buscar por nombre de código.

Por cada código debe mostrar nombre, descripción y si es subcódigo de otro código.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.

RF4.5Asignación de código padre

- **Descripción:**
  Capacidad del sistema de permitir a un usuario asignarle a un código un código padre. Todos los instrumentos, registros y marcos teóricos relacionados con el primer código, quedan asociados al código padre.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.

3.2.5. Registro

RF5.1Crear carpeta

- **Descripción:**
  El sistema permitirá a un usuario crear carpetas con nombres en las diferentes rutas, con el fin de poder organizar de manera flexible y conveniente las instancias de los instrumentos.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.

RF5.2Agregar registro a instrumento

- **Descripción:**
  El sistema permitirá crear registros de un instrumento dentro del proyecto. Si el instrumento es una entrevista, los registros son las respuestas a la entrevista, en caso de encuestas, las respuestas a encuestas. Para crear un registro de instrumento, el usuario debe proporcionar el instrumento, nombre y un archivo en formato doc o csv con las respuestas al instrumento seleccionado o subir las respuestas manualmente.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe ser investigador del proyecto y tener rol administrador o editor.

- **Flujo de usuarios:**
  El usuario accede a la página de agregar registro de instrumento.
  El usuario selecciona entre los instrumentos disponibles en cual desea crear el registro, agregando un nombre y un documento seleccionado desde su ordenador en formato doc o excel.
  El usuario es redireccionado a la ruta donde cre el registro, en caso de éxito vera el nuevo registro en la carpeta correspondiente, en caso de error se le muestra un mensaje al usuario.

- **Diagrama de Secuencia del Sistema**
  El siguiente diagrama de secuencia (Figura 3.4) ilustra el flujo de interacción entre el usuario y el sistema durante el proceso de agregar registro a un instrumento en un proyecto.

RF5.3Ver registro

- **Descripción:**
  El sistema permitirá a un usuario ver un registro de un instrumento. La aplicación deberá mostrar el nombre del registro y del instrumento, con sus preguntas o secciones y sus respuestas obtenidas del archivo subido en Agregar registro. Además, deberá mostrar los códigos aplicados junto con su nombre y descripción. Estos códigos serán tratados en los casos de

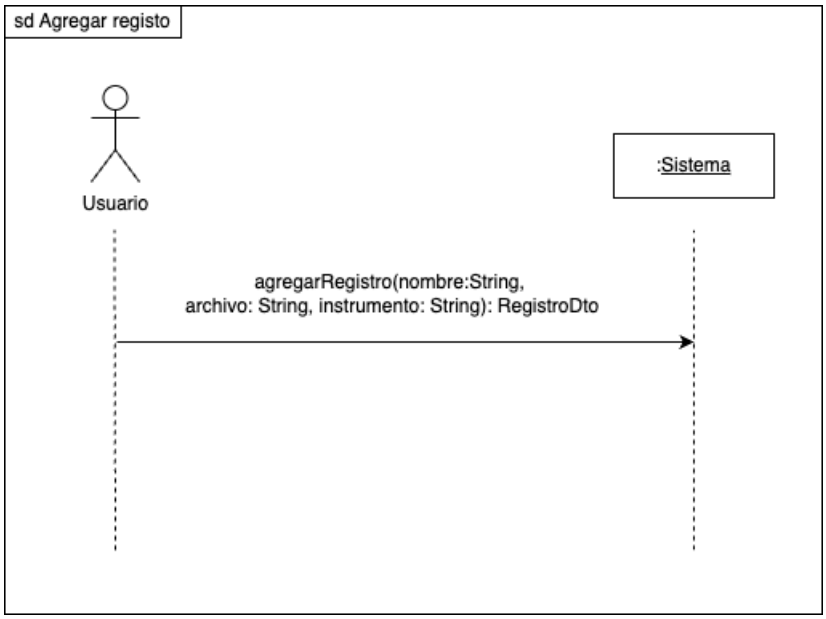

Figura 3.4: Diagrama de secuencia agregar registro a instrumento

uso siguientes, donde se explicará cómo agregarlos. Su función principal es asociar análisis o marcas a instrumentos, registros y marcos teóricos, asignando un significado específico a palabras o frases. Por ejemplo, en un registro donde se indique: .^{es}taba llorando de emoción ", podría asignarse el código "felicidad".

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.
  Debe existir en el proyecto el instrumento y el registro.

RF5.4Ver y filtrar registros

- **Descripción:**
  Capacidad del sistema de permitir a un usuario ver todos los registros y filtrar entre ellos. Debe permitir filtrar registros según su tipo de instrumento(encuesta, entrevista u otro).

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página ver registros.
  Se le muestra al usuario un selector con los tipos de instrumentos (encuestas o entrevistas).
  Al seleccionar un tipo de instrumento y buscar se le muestra los registros que pertenecen a un instrumento de ese tipo.

RF5.5Agregar código a registro

- **Descripción:**
  Capacidad del sistema de permitir a un usuario agregar un código a un registro.

  El usuario deberá especificar el código y el registro.

  El registro quedará marcado con ese código.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador o editor en el proyecto.
  Debe existir al menos un código en el sistema. Debe existir al menos un registro en el sistema.

- **Flujo de Usuarios:**
  El usuario accede al registro al que desea agregar un código.
  Selecciona la sección del registro a la que quiere asignar un código.
  Elige un código de los disponibles o crea uno nuevo.

RF5.6Ver bitácora

- **Descripción:**
  Capacidad del sistema de mostrar una línea temporal con acciones realizadas en un proyecto. Las acciones que forman parte de la bitácora son:

  - Proyecto creado
  - Cambio de fase
  - Investigador agregado
  - Investigador eliminado
  - Fase creada
  - Fase eliminada
  - Carpeta creada
  - Carpeta eliminada
  - Instrumento creado
  - Instrumento eliminado
  - Registro creado
  - Registro eliminado
  - Código agregado
  - Código eliminado
  - Código agrupado
  - Marco teórico agregado

  Por cada una de estas acciones se muestra en la línea temporal, fecha de realización e investigador responsable.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.

RF5.7Filtrar acción en bitácora

- **Descripción:**
  Capacidad del sistema de permitir filtrar por acción en la línea temporal con acciones realizadas en un proyecto.

- **Precondiciones:**
  El usuario debe haber iniciado sesión mediante su nombre de usuario/email y su contraseña.
  El usuario debe tener rol administrador, editor o comentador en el proyecto.

- **Flujo de Usuarios:**
  El usuario accede a la página bitácora.
  Se le muestra al usuario todas las acciones disponibles en la bitácora, el usuario selecciona una o más.
  Se muestra en la línea temporal solo las acciones seleccionadas por el usuario.

3.3. Requerimientos no funcionales

Los requerimientos no funcionales son aspectos críticos de una aplicación que no están relacionados directamente con las funcionalidades específicas de la aplicación, pero que son esenciales para garantizar su rendimiento, seguridad, escalabilidad y usabilidad.
Además, son transversales a todas las funcionalidades, asegurando que estas se ejecuten de manera eficiente y segura en todo el sistema.
(*Iso 25010*, s.f.)

Tiempo de respuesta La aplicación debe responder lo más rápido posible a las solicitudes del usuario. Se define un umbral de 5 segundos.

Escalabilidad La aplicación debe poder manejar un aumento en la carga de usuarios sin degradación significativa del rendimiento. Se define un umbral de 50 a 100, en base a los recursos disponibles en el servidor. En caso de querer

aumentar este número se deberá o bien mejorar los recursos del servidor o comenzar a utilizar una arquitectura multihilos.

Tiempo de disponibilidad La aplicación debe estar disponible y funcionando durante un porcentaje alto del tiempo. Se define que en horario de trabajo de los investigadores la disponibilidad debe ser de un 95 %.

Tolerancia a fallos Debe ser capaz de recuperarse de fallos y proporcionar una experiencia de usuario continua.

Autenticación y autorización Debe contar con mecanismos de autenticación, seguros y autorización para controlar el acceso a datos y funcionalidades.

Facilidad de mantenimiento El código de la aplicación debe ser fácil de entender y modificar para futuras actualizaciones y correcciones.

Interfaz de usuario intuitiva La interfaz de usuario debe ser fácil de usar y comprender para los investigadores.

Almacenamiento de datos Debe garantizarse un rendimiento eficiente para la lectura y escritura de datos en bases de datos u otros sistemas de almacenamiento, así como protección de los mismos.

Integridad de datos Se debe garantizar la precisión y coherencia de la información almacenada y procesada en el sistema.

Idioma español El idioma de la aplicación debe ser español.

Software libre y gratuito A lo largo del desarrollo se priorizó la utilización de software libre y gratuito. Entre sus ventajas se destaca la libertad de personalización, facilidad de integración, acceso al código fuente y que no genera costos adicionales al proyecto.

Capítulo 4

Diseño de la solución

En esta sección se describe el diseño de la solución. En primer lugar, se presenta el modelo conceptual del negocio y su implementación en una base de datos relacional. A continuación, se detalla la arquitectura elegida, mostrando un modelo de alto nivel, y finalmente se discuten las alternativas consideradas junto con sus ventajas y desventajas.

4.1. Modelo de Datos

En esta sección se describe el modelo conceptual del negocio y su implementación en una base de datos relacional. El modelo conceptual, mostrado en la Figura 4.1, representa una visión general de las principales entidades del sistema y sus relaciones, proporcionando una estructura lógica para la organización y gestión de los datos. Las entidades clave de este modelo incluyen Usuarios, Proyectos, Roles de Usuario, Hipótesis, Códigos, Instrumentos y Registros, entre otras, las cuales están interrelacionadas para reflejar las interacciones y flujos de trabajo en un proyecto de investigación.

Por ejemplo, un Proyecto puede estar asociado a varios Usuarios con diferentes Roles, lo que facilita el trabajo colaborativo dentro del sistema. Además, un Proyecto puede incluir múltiples Hipótesis e Instrumentos, cada uno con elementos específicos, como preguntas o ítems, que estructuran la información y permiten analizar los datos recopilados. Las relaciones muchos-a-muchos entre entidades, como la relación entre Códigos y Proyectos, o entre Instrumentos y

Registros, se gestionan mediante tablas intermedias que proporcionan flexibilidad y escalabilidad al sistema.

Este modelo conceptual se implementa en la base de datos relacional, que se refleja en el modelo de dominio de la Figura 4.2. En este modelo relacional, las entidades se transforman en tablas de la base de datos, y las relaciones entre ellas se implementan mediante claves foráneas.

A continuación, se describen las principales entidades que componen la base de datos:

- **Usuarios:** Almacena información sobre los usuarios de la aplicación, incluyendo datos de autenticación como correo electrónico y contraseña, así como otros campos como nombre, apellidos y roles.
- **Proyecto:** Almacena información sobre proyectos en la aplicación. Cada proyecto tiene un nombre, descripción y fase asociado.
- **Logs del proyecto:** Registra eventos y cambios en proyectos.
- **Fases:** Almacena las distintas fases que puede tener un proyecto, indicando la fase actual de un proyecto.
- **Configuración:** Almacena configuraciones y ajustes de la aplicación.
- **Carpetas:** Almacenan y organizan información dentro de la aplicación. Estas carpetas están asociadas a un proyecto, permitiendo una estructura organizada de los datos. Pueden ser utilizadas para categorizar y clasificar diversos tipos de información de manera jerárquica.
- **Hipótesis:** Almacena información sobre hipótesis, que son afirmaciones o suposiciones que se investigan en el contexto de proyectos.
- **Instrumentos:** Almacena información sobre los diferentes medios para obtener datos sobre la investigación.
- **Registros:** Almacena información sobre registros que hacen referencia a un instrumento de la aplicación. Cada registro está relacionado con usuarios, instrumentos y carpetas. Por ejemplo, un registro de un instrumento tipo encuesta contiene las respuestas de una persona en particular.

- **Códigos:** Almacena la codificación de los metadatos que se asocian a los registros, guardando un nombre y descripción. Estos códigos pertenecen a un proyecto y pueden estar asociados a marco teórico, instrumentos y/o registros específicos.

El modelo relacional implementa estas entidades de manera que cada una se corresponde con una tabla, y las relaciones entre ellas se gestionan mediante claves primarias y foráneas. Esto garantiza una estructura organizada, flexible y escalable para el manejo de grandes volúmenes de datos en el sistema.

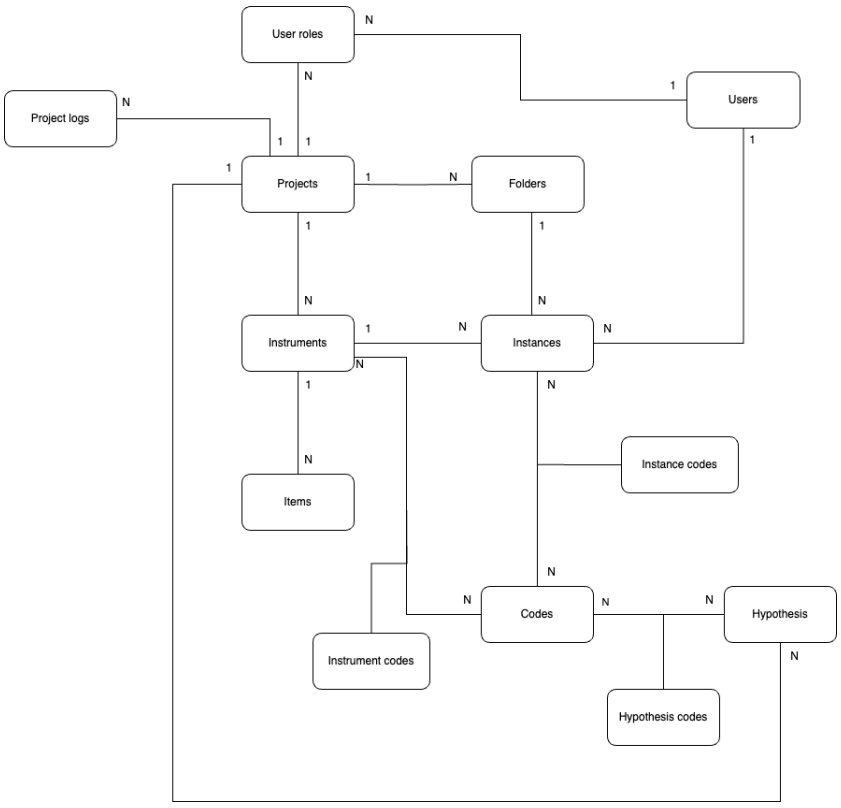

Figura 4.1: Modelo conceptual

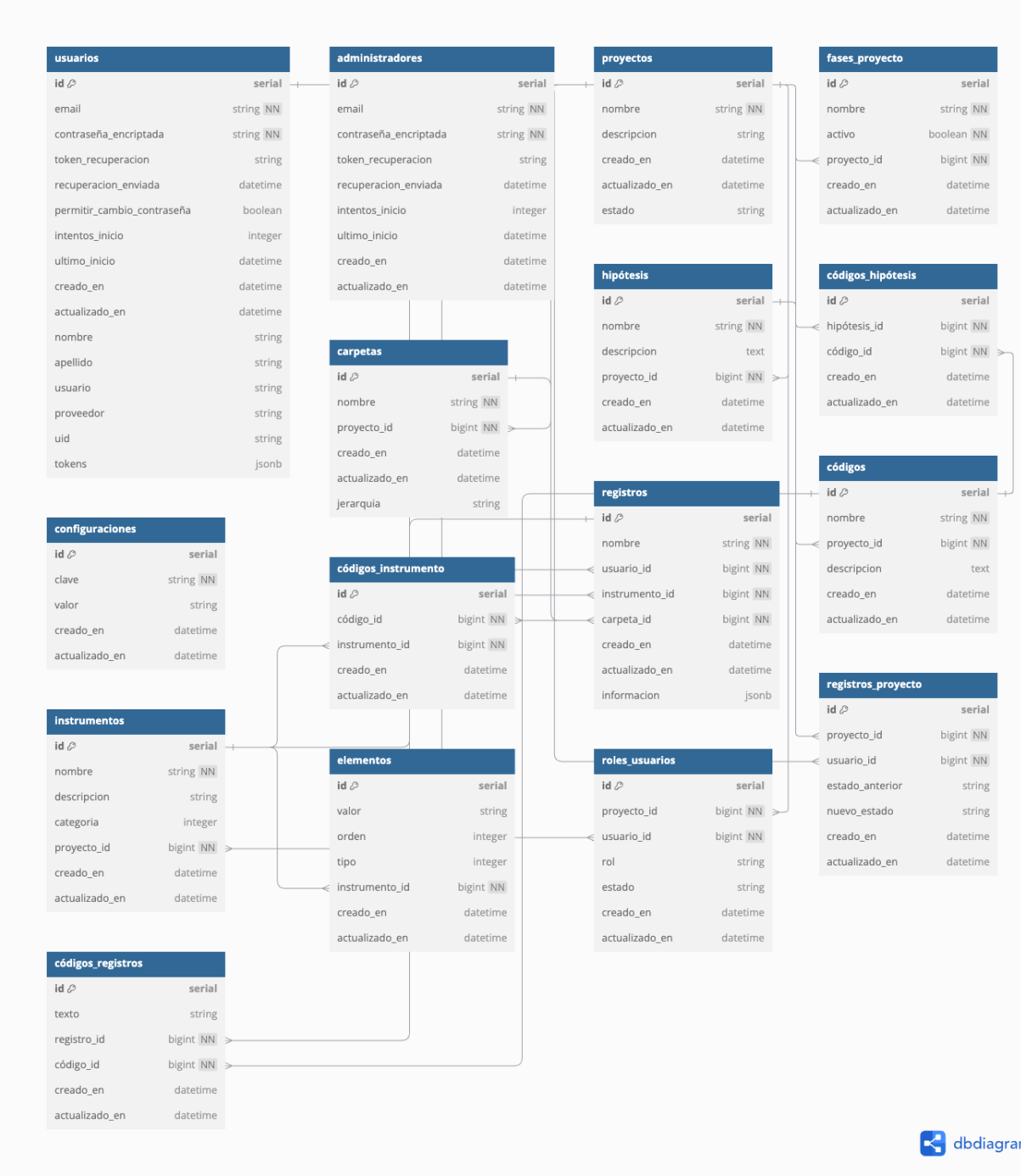

Figura 4.2: Modelo de dominio

4.2. Arquitectura

La arquitectura elegida es una arquitectura cliente-servidor monolítica, como se muestra en la Figura 4.3. Esta se basa en el framework Ruby on Rails para el desarrollo del backend y frontend, junto con PostgreSQL como sistema gestor de base de datos.

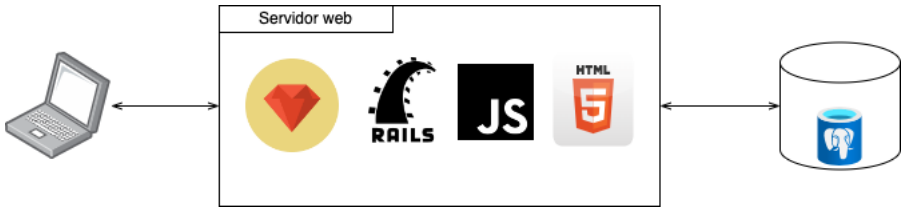

Figura 4.3: Arquitectura del sistema

Esta elección se basa en la consideración de que, en una fase inicial, debido al volumen de datos que maneja el proyecto, no se aprovechan plenamente las ventajas clave que ofrece la arquitectura de microservicios, y se agrega una complejidad innecesaria.

La arquitectura cliente-servidor monolítica basada en Ruby on Rails con PostgreSQL nos proporciona una base sólida que permite un desarrollo rápido de la aplicación y una gestión eficiente de los datos. No obstante, es importante tener en cuenta que, a medida que la aplicación crezca, podríamos necesitar evaluar enfoques de arquitectura más distribuidos y escalables para adaptarnos a las cambiantes necesidades del negocio.

4.3. Análisis de alternativas arquitectónicas

Se analizaron dos estilos de arquitecturas diferentes con el fin de elegir la más conveniente para el proyecto. Estas son la arquitectura monolítica cliente-servidor y la arquitectura basada en microservicios.

Arquitectura monolítica cliente-servidor: es un enfoque de diseño de software en el que una aplicación se divide en dos componentes principales: el cliente y el servidor. El cliente es la parte de la aplicación que se ejecuta en la máquina del usuario, mientras que el servidor es la parte que se ejecuta en un servidor remoto. Además, este incluirá una base de datos para almacenar

y recuperar datos. A continuación, se describen sus características, ventajas y desventajas.

- **Cliente:** Interfaz de la aplicación que se ejecuta en la máquina del usuario. Responsable de la presentación de la información y la interacción con la aplicación.
- **Servidor:** Parte central de la aplicación que procesa las solicitudes del cliente, realiza cálculos y accede a la base de datos. Contiene:
  - Controladores: Manejan las solicitudes HTTP y controlan las acciones correspondientes.
  - Modelos: Definen la estructura de datos y la lógica de negocio.
  - Vistas: Encargadas de la presentación de datos al usuario. En este proyecto, se utiliza Ruby on Rails para las vistas.
  - Rutas: Definen cómo se asignan las URL a las acciones de los controladores.
- **Base de datos:** Se utiliza PostgreSQL como sistema de gestión relacional, que ofrece alta confiabilidad, escalabilidad y soporte para grandes volúmenes de datos.

Ventajas:

- Escalabilidad vertical.
- Menor costo inicial.
- Facilidad de desarrollo para proyectos pequeños.
- Menor latencia.
- Control centralizado.
- Curva de aprendizaje baja.

Desventajas:

- Difícil de escalar y mantener a medida que crece.
- Punto único de falla.
- Dificultad para desarrollo concurrente.
- Flexibilidad tecnológica limitada.

Arquitectura de microservicios: se basa en desarrollar una aplicación como un conjunto de servicios pequeños e independientes, centrados en una función específica. Estos servicios se comunican a través de APIs y pueden escalarse y desplegarse de forma independiente.

Ventajas:

- Escalabilidad y flexibilidad.
- Resiliencia y tolerancia a fallos.
- Desarrollo ágil.
- Optimización individual del rendimiento.

Desventajas:

- Mayor complejidad operativa.
- Coordinación y comunicación entre servicios más compleja.
- Costos de infraestructura mayores.
- Curva de aprendizaje más alta.

Aspecto	**Arq. Monolítica**	**Arq. Microservicios**
Escalabilidad	Vertical	Horizontal
Costo inicial	Bajo	Moderado
Facilidad de Desarrollo	Alta	Moderada
Latencia	Baja	Moderada
Control centralizado	Sí	No
Curva de aprendizaje	Baja	Alta
Mantenibilidad	Moderada	Alta
Tolerancia a fallos	Baja	Alta
Flexibilidad tecnológica	Baja	Alta
Gestión de la complejidad	Baja	Alta
Desarrollo concurrente	Complicado	Facilitado

Tabla 4.1: Comparación entre arquitecturas

Conclusiones basadas en la comparación:

- **Escalabilidad:** Microservicios es superior en escalabilidad horizontal.

- **Costo inicial:** Monolítica es más económica inicialmente.
- **Facilidad de desarrollo:** Monolítica tiene menor complejidad inicial.
- **Tolerancia a fallos y mantenibilidad:** Microservicios es más robusta y modular.
- **Flexibilidad tecnológica:** Microservicios permite mayor diversidad tecnológica.

Capítulo 5

Metodología de desarrollo

Para este proyecto se decidió aplicar una metodología ágil y desarrollo incremental, particularmente la metodología SCRUM(*Scrum*, s.f.-a). Se llegó a esta decisión, ya que si bien los objetivos del framework a desarrollar están claros, entendemos que las necesidades y requerimientos del mismo pueden verse modificadas a medida que se avance en el tiempo.

La metodología SCRUM tiene ciertos beneficios que se adaptan muy bien a las necesidades del proyecto. A continuación se describen ventajas de aplicar esta metodología.

- Flexibilidad a los cambios: SCRUM permite una rápida reacción a los cambios que sugiera el cliente o el mercado.
- Desarrollo incremental: Los clientes pueden empezar a utilizar el proyecto antes de que esté terminado debido a su desarrollo incremental. Esta característica es de vital importancia, ya que nos ayudará a realizar el framework que más se adapte y entiendan los clientes.
- Reducción de riesgos y predicciones de tiempo: Al ir desarrollando de manera incremental, se permite conocer la velocidad del equipo. Cada sprint, que representa una iteración del proceso de desarrollo, permite perfeccionar las estimaciones. Al comenzar con las funcionalidades de mayor valor, se logran mitigar los mayores riesgos de forma anticipada.
- Mejora de la calidad: En cada incremento se introducen nuevas funcionalidades y puede llevar a una refactorización del código si se viera una

oportunidad, tanto en diseño como en arquitectura.

Para la aplicación de SCRUM, se trabaja con sprints de dos semanas. Se crea un backlog con todas las tareas a realizar, separadas por categorías como backend, frontend, configuraciones generales del proyecto e investigación. Cada tarea recibe un puntaje, aplicando el sistema de puntuación Fibonacci (*Trello*, s.f.). Al inicio de cada sprint, se seleccionan las tareas con mayor prioridad y valor para el cliente, con el objetivo de implementarlas en dicho ciclo de trabajo.

La herramienta Trello se utiliza para la gestión y administración de las tareas, permitiendo a los equipos idear, planificar, gestionar y celebrar sus objetivos de manera colaborativa, productiva y organizada. A través de Trello, se da seguimiento al avance y estado del desarrollo, y se añaden nuevas tareas cuando sea necesario, ya sea por errores en el código, fallos en el sistema o la incorporación de nuevas funcionalidades.

El desarrollo incremental permite mostrar el proyecto a los investigadores a medida que avanza, lo que facilita su interacción y retroalimentación constante. Esto es fundamental, ya que el objetivo principal es crear un software funcional que cumpla con sus necesidades. Para ello, se definió una reunión mensual con los investigadores para presentar los avances, explicar el software y obtener feedback. Este feedback incluye opiniones sobre nuevas funcionalidades que podrían facilitar su trabajo, las cuales se analizan, se evalúa su viabilidad y se priorizan dentro del backlog. También se recogen opiniones sobre la navegabilidad de la aplicación y los nombres de las acciones. Además, se cuenta con un glosario de los términos utilizados por los investigadores para referirse a distintos aspectos del proyecto, garantizando que el software esté lo más adaptado posible a su lenguaje y necesidades (*Scrum*, s.f.-b).

5.1. Integración Continua y entrega continua

En este proyecto se optó por seguir la práctica de integración continua, es una práctica de desarrollo de software en la cual los miembros de un equipo integran su trabajo frecuentemente, como mínimo de forma diaria. Cada integración se verifica mediante una herramienta de construcción automática para detectar los errores de integración tan pronto como sea posible. La experiencia de diversos equipos de trabajo indica que que este enfoque lleva a una reducción significativa

de los problemas de integración y permite a un equipo desarrollar software cohesivo de forma más rápida (Martin Fowler, 2006, «Continuos Integration»). (*CI*, s.f.)

Al incluir integración continua en nuestro proyecto nos aseguramos un estándar y calidad de código, ya que siempre que se agregue líneas de código estas tienen que pasar la línea de procesos especificados que corren automáticamente, por ejemplo, todas las pruebas unitarias deben ejecutarse correctamente, este enfoque asegura que se cumpla que cierto porcentaje de código tenga pruebas unitarias, se detectan errores de sintaxis, uso de malas prácticas, uso de estilos inconsistentes, entre otros.

Existen varias herramientas para utilizar esta práctica; En nuestro caso vamos a usar la herramienta circle CI.
(*Circle CI*, s.f.)

La entrega continua amplía la práctica de integración continua.
La entrega continua es una práctica de desarrollo de software mediante la cual se preparan automáticamente los cambios en el código y se entregan a la fase de producción. Esto implica la ejecución de un proceso de construcción que incluye la compilación del código, la ejecución de pruebas automatizadas y la generación de artefactos desplegables. Estos artefactos pueden ser binarios, como archivos ejecutables o paquetes de software, que están listos para ser desplegados en el entorno de producción. La entrega continua ayuda a reducir el tiempo y el esfuerzo necesarios para implementar cambios en el software, lo que aumenta la eficiencia del equipo de desarrollo y la calidad del producto final.

Los beneficios de esta práctica son:

- Automatizar el proceso de publicación de software
- Mejora la productividad de desarrollo
- Permite encontrar y arreglar errores con mayor rapidez
- Permite entregar las actualizaciones con mayor rapidez

(*CD*, s.f.)

5.2. Tipos de Pruebas

En primer lugar, cabe mencionar que las pruebas se dividen en dos grandes categorías: **manuales** y **automáticas**.

Pruebas Manuales

Las pruebas manuales son llevadas a cabo por una persona que interactúa directamente con el proyecto. Son costosas porque requieren un ambiente configurado y análisis de distintos casos a probar.

Pruebas Automáticas

Las pruebas automáticas las realiza una computadora mediante scripts u otras metodologías. Tienen ventajas significativas: son más rápidas, confiables y pueden ser ejecutadas con mayor regularidad. Varían en complejidad, desde muy sencillas hasta muy complejas. Además, los tests automáticos son una parte integral del proceso de integración continua y entrega continua, un componente que discutiremos en una sección más adelante.

Clasificación de Pruebas

Pruebas Funcionales Las pruebas funcionales se centran en los requerimientos de negocio de una aplicación. Se ejecutan para verificar el cumplimiento de los requerimientos funcionales, comprobando el resultado de una acción dada, con distintas entradas y acciones. Estas pruebas aseguran que el comportamiento de la aplicación cumpla con las expectativas del negocio.

Pruebas No Funcionales Dentro de las pruebas no funcionales, se incluyen varios tipos específicos que se ejecutan para verificar el cumplimiento de los requerimientos no funcionales. A continuación, se describen algunos de los tipos más relevantes:

- Pruebas de Seguridad: Evalúan la capacidad del sistema para proteger los datos y garantizar la confidencialidad, integridad y disponibilidad de la información. Estas pruebas también verifican la resistencia del sistema frente a ataques o vulnerabilidades, asegurando que los controles de acceso y las políticas de seguridad sean efectivas, conforme a las necesidades de protección definidas por el negocio.

- Pruebas de Rendimiento: Verifican cómo responde el sistema bajo diferentes condiciones de carga. Este tipo de pruebas mide aspectos clave como la fiabilidad, estabilidad, escalabilidad y disponibilidad de la aplicación cuando se enfrenta a situaciones de alta demanda. Se busca asegurar que el sistema pueda manejar picos de tráfico sin afectar su funcionamiento.

- Pruebas de Usabilidad: Aseguran que el sistema sea fácil de usar y que proporcione una experiencia de usuario óptima. En particular, se evalúa la capacidad de interacción del usuario con la aplicación, asegurando que sea intuitiva, eficiente y satisfactoria en términos de diseño de la interfaz, accesibilidad y la realización de tareas comunes. Estas pruebas también buscan mejorar la satisfacción del usuario y facilitar el aprendizaje del sistema.

- Pruebas de Performance: Se centran en la evaluación del rendimiento del sistema, evaluando factores como la capacidad de respuesta, el tiempo de carga y la utilización de recursos, con el fin de garantizar que el sistema funcione de manera eficiente incluso bajo condiciones extremas o con grandes volúmenes de datos.

Niveles de Pruebas

Pruebas Unitarias Las pruebas unitarias son de bajo nivel y se ubican en el código de la aplicación. Prueban individualmente las distintas funciones y/o métodos de las clases, componentes y servicios. Son económicas y generalmente se utilizan en la integración continua de los proyectos. Un enfoque particular es el Desarrollo Dirigido por Pruebas (TDD), que se basa en desarrollar las pruebas antes de la implementación de solución.

TDD consta de tres leyes:

- No escribirás código de producción sin antes escribir un test que falle.

- No escribirás más de un test unitario suficiente para fallar (y no compilar es fallar).

- No escribirás más código del necesario para hacer pasar el test.

Pruebas de Integración Las pruebas de integración verifican que los diferentes módulos y/o servicios de la aplicación funcionen correctamente cuando

trabajan juntos. Se realizan después de las pruebas unitarias. Aunque son costosas de implementar y ejecutar, son esenciales para asegurar la cohesión del sistema.

Pruebas de Punta a Punta Estas pruebas replican el comportamiento de los usuarios con el software en un entorno completo. Son muy costosas y útiles, pero difíciles de mantener cuando son automatizadas. Prueban todas las funcionalidades del sistema con sus diferentes variables y excepciones.

Pruebas de Regresión Las pruebas de regresión verifican un conjunto de escenarios que funcionaron correctamente en el pasado para asegurar que continúen funcionando. Son útiles para detectar si una nueva funcionalidad ha introducido un bug en una funcionalidad ya existente, aunque presentan un alto costo.

5.3. Selección de Pruebas

Después de profundizar en todos estos tipos de pruebas, analizar sus ventajas y desventajas, y considerando el tamaño del equipo, el tiempo y el alcance del proyecto, hemos decidido lo siguiente:

- Pruebas Unitarias con TDD: Incluir las pruebas unitarias utilizando la metodología TDD como un componente esencial del proyecto.
- Pruebas Manuales: Realizar pruebas manuales periódicamente siguiendo una serie de casos de pruebas para verificar la usabilidad en conjunto.
- Pruebas de Rendimiento: Implementar pruebas de rendimiento para asegurar cómo se comporta la aplicación en distintos escenarios. Definiremos los tiempos de respuesta óptimos y la cantidad de usuarios y peticiones que la aplicación debe tolerar.

Aunque sería ideal aplicar la mayor cantidad de pruebas posibles, debemos tener en cuenta el tiempo y el alcance del proyecto. (*Pruebas*, s.f.)

Capítulo 6

Implementación

En esta sección se detallan las tecnologías y herramientas utilizadas para el desarrollo de la aplicación. Se describe el entorno de desarrollo seleccionado, incluyendo el framework principal, lenguajes de programación, gemas relevantes, herramientas de estilo y lenguajes de marcado, así como los sistemas de gestión de bases de datos empleados. También se explican los mecanismos de control de versiones, contenedorización con Docker y despliegue del sistema. Todo ello con el fin de ofrecer una visión clara y completa del stack tecnológico utilizado en el proyecto y de cómo se integran sus distintos componentes para lograr una solución eficiente, escalable y mantenible.

También se detalla la interfaz y funcionamiento de la aplicación, a modo de una demostración de su funcionamiento.

6.1. Tecnología y herramientas

6.1.1. Ruby - Ruby on Rails

Ruby on Rails es un framework escrito en Ruby que facilita el desarrollo de aplicaciones web al proporcionar una estructura y conjunto de convenciones predefinidas. A su vez, Ruby es un lenguaje de programación orientado a objetos e interpretado, conocido por su simplicidad y elegancia. Juntos, Ruby y Ruby on Rails permiten la creación rápida y eficiente de aplicaciones web robustas y escalables. (*Ruby on Rails*, s.f.)

Ruby on Rails (a partir de ahora RoR) aplica el paradigma MVC, separando la aplicación en tres partes:

- Modelo: Parte de la aplicación donde se encuentra la lógica de la aplicación.
- Vista: Muestra el modelo de tal forma que el usuario pueda interactuar con él.
- Controlador: Responde a eventos activados por el usuario e interactúa con el modelo, es el encargado de enviar estos datos a la vista.

Algunas ventajas son:

- Facilidad de Desarrollo: Ruby on Rails proporciona un entorno de desarrollo ágil y una amplia gama de gemas (bibliotecas) que simplifican la creación de características y la gestión de la base de datos.
- Seguridad: Ruby on Rails incluye medidas de seguridad integradas para proteger contra ataques comunes, como inyecciones SQL y ataques de scripting entre sitios (XSS).
- Facilidad de Mantenimiento: Aunque es un monolito, Rails proporciona una estructura clara para organizar el código, lo que facilita el mantenimiento a largo plazo.

Se opta por Ruby on Rails (RoR) por dos razones fundamentales. La primera radica en su capacidad para acelerar el desarrollo de aplicaciones web al automatizar tareas repetitivas y, en general, simplificar aspectos inherentes a la mayoría de los proyectos. Esta ventaja permite al equipo de desarrollo concentrarse en el producto final, en lugar de perder tiempo en tareas tediosas. La segunda razón corresponde al hecho de que los integrantes del proyecto poseen experiencia y conocimientos sólidos en este lenguaje. Esta familiaridad con RoR evita la necesidad de invertir tiempo en aprender un lenguaje nuevo, agilizando así el proceso de desarrollo.

En cuanto a las versiones, se ha decidido trabajar con Ruby 3.0 y Ruby on Rails 7, que es la versión más reciente hasta la fecha.

La elección de desarrollar una arquitectura monolítica tambien se justifica por la capacidad de Ruby on Rails de integrar frameworks como React directamente en las vistas. Esto brinda una experiencia de usuario tipo SPA (Single Page Application) de manera eficiente, sin necesidad de dividir la lógica del

frontend. Esta decisión optimiza el tiempo de desarrollo y simplifica las tareas a realizar.

6.1.2. Gemas relevantes

- Active Admin: es una popular gema de Ruby on Rails que se utiliza para crear rápidamente una interfaz de administración para una aplicación web. Con Active Admin, puedes generar fácilmente un panel de administración completo y personalizado para gestionar los modelos de tu aplicación. (*Active Admin*, s.f.)

- Devise:

  Proporciona una solución completa y flexible para gestionar la autenticación de usuarios en aplicaciones web, ahorrándote tiempo y esfuerzo al implementar un sistema de autenticación desde cero. (*Devise*, s.f.)

- Rspec: Proporciona una sintaxis expresiva y legible que facilita la escritura y ejecución de pruebas automatizadas para una aplicación de Rails. RSpec se basa en el paradigma de pruebas de comportamiento y promueve un enfoque orientado al comportamiento en lugar de centrarse únicamente en las pruebas unitarias.

  Con RSpec, puedes escribir pruebas claras y concisas que describen el comportamiento esperado de tu código. La sintaxis de RSpec se asemeja a un lenguaje natural, lo que hace que las pruebas sean más legibles y comprensibles tanto para los desarrolladores como para otros miembros del equipo. (*RSpec*, s.f.)

6.2. CSS, HTML y JavaScript

HTML (HyperText Markup Language): HTML es el lenguaje de marcado estándar utilizado para crear la estructura y el contenido de las páginas web. Con HTML, puedes definir los elementos y componentes de una página, como

encabezados, párrafos, imágenes, enlaces y formularios. Proporciona una estructura semántica que ayuda a los navegadores y motores de búsqueda a entender y renderizar correctamente el contenido de una página web. (*HTML*, s.f.)
CSS (Cascading Style Sheets): CSS es un lenguaje de estilo utilizado para controlar la presentación y el diseño visual de las páginas web. Con CSS, puedes definir reglas que especifican cómo se deben mostrar los elementos HTML en términos de colores, fuentes, tamaños, márgenes, espaciados y otras propiedades visuales. CSS permite separar la estructura y el contenido de una página web de su apariencia visual, lo que facilita la personalización y el mantenimiento de estilos coherentes en todo el sitio. (*CSS*, s.f.)
JavaScript: JavaScript es un lenguaje de programación utilizado para agregar interactividad y funcionalidad dinámica a las páginas web. Con JavaScript, puedes manipular y modificar elementos HTML, responder a eventos del usuario, realizar validaciones de formularios, realizar peticiones a servidores y mucho más. JavaScript se ejecuta en el navegador del cliente y permite crear experiencias interactivas y en tiempo real en las aplicaciones web. (*Java Script*, s.f.)
En resumen, HTML se utiliza para definir la estructura y el contenido de una página web, CSS se utiliza para controlar el estilo y la presentación visual de la página, y JavaScript se utiliza para agregar interactividad y funcionalidad dinámica. Estos tres lenguajes trabajan juntos para crear páginas web atractivas y funcionales.

6.3. Bases de Datos

La base de datos constará de tres partes importantes. En primer lugar, se elegirá PostgreSQL(*Postgres SQL*, s.f.) como base de datos relacional, ya que permite una mayor flexibilidad y variedad de tipos de datos en comparación con las otras opciones. En segundo lugar, se usará Redis (*Redis*, s.f.) como base de datos basada en cache, ya que es la opción que más se acopla al lenguaje de programación elegido, y se usará para gestionar todos los trabajos asíncronos necesarios dentro de la plataforma. Finalmente, se usará una base de datos basada en grafos para poder representar las relaciones entre tipos de datos. Para esto se usará Neo4j(*Neo 4j*, s.f.-a), ya que es de las pocas habilitadas para trabajar con Ruby on Rails.

PostgreSQL es un sistema de gestión de bases de datos relacional de código

abierto y robusto. Es conocido por su confiabilidad, escalabilidad y capacidad de manejar grandes volúmenes de datos. PostgreSQL ofrece soporte para una amplia variedad de características avanzadas, como consultas complejas, transacciones ACID, integridad referencial y replicación. También es altamente compatible con los estándares SQL y admite funciones avanzadas como el soporte para tipos de datos personalizados y extensiones.

Neo4j es una base de datos de grafos de alto rendimiento y orientada a objetos. A diferencia de las bases de datos relacionales tradicionales, Neo4j almacena los datos en forma de nodos interconectados por relaciones, lo que permite representar y consultar relaciones complejas de manera eficiente. Los grafos son especialmente útiles para modelar datos altamente interconectados, como redes sociales, recomendaciones de productos, sistemas de recomendación y muchas otras aplicaciones.

6.3.1. Integración de PostgresQL y Rails

Ruby on Rails es un popular framework de desarrollo web que proporciona una capa de abstracción para interactuar con bases de datos. Rails tiene una excelente integración con PostgreSQL, lo que permite utilizar PostgreSQL como base de datos principal para tu aplicación Rails. (*Postgres SQL - Rails*, s.f.)

6.3.2. Integración Neo4J y Rails

Ruby on Rails ofrece una integración completa con Neo4j a través de la gema "neo4j". Esta gema proporciona un conjunto de herramientas y métodos que facilitan el desarrollo de aplicaciones Rails utilizando Neo4j como base de datos subyacente. (*Neo 4j*, s.f.-a)

6.4. Docker

Docker es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que

el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución.

Se decidió la utilización de la herramienta Docker por la facilidad que ofrece para empaquetar el software, lo que permite iniciar el proyecto en cualquier servidor de una forma más sencilla. (*Docker*, s.f.)

6.5. GitHub

GitHub, es una herramienta open-source creada por Linus Torvalds en 2005, tiene como objetivo facilitar la gestión de las versiones de código. Específicamente, Git es un sistema de control de versiones distribuido, lo que significa que toda la base de código y el historial se encuentran disponibles en la computadora de cada desarrollador, lo que permite que cada uno pueda gestionar de forma fácil la ramificación (branching) y fusión (merging) del código. En otras palabras, Git te permite realizar el seguimiento de cada cambio en el código, permitiendo ver el historial de cambios. (*GitHub*, s.f.)

Al integrar GitHub en el proyecto, se establece un GitFlow, un método que segmenta el trabajo en diferentes tipos de ramas para adaptarse al proceso colaborativo del equipo de desarrollo. En nuestro caso, hemos optado por seguir el modelo clásico, que comienza con dos ramas principales: (*GitHub Flow*, s.f.)

- Máster: contiene el código de producción.
- Develop: contiene el código en el que se considera que el desarrollo está completo.

A partir de estas ramas principales, se derivan otras ramas que se utilizarán en el proyecto. Todas estas ramas siguen el mismo patrón de nomenclatura, que es:

- feature/nombre-funcionalidad: Siempre se crea partiendo de la rama develop, y se usa para la creación de una nueva funcionalidad, una vez terminada se integra a la branch develop.

- release/numero-version : se crea desde develop para pasar a producción las nuevas funcionalidades que fueron finalizadas, se integra con la rama master.
- hotfix: Se crea desde master para solucionar un error o bug que se encuentra en producción, se integra con master y develop una vez terminado.

6.6. Despliegue de la aplicación

La facultad nos brindó un servidor en la nube de antel, este cuenta con Linux, Ubuntu y 4 GB de memoria RAM. La base de datos también será almacenada en este servidor para mayor privacidad de los datos.

6.7. Interfaz de la aplicación

A continuación, se presentan las pantallas correspondientes a los casos de uso descritos en la sección de requerimientos. Estas pantallas ilustran cómo la aplicación satisface las necesidades y funcionalidades solicitadas por los usuarios. Las subsecciones muestran de manera visual el flujo de interacción con la aplicación, en cada uno de los escenarios definidos previamente.

En algunas situaciones, los casos de uso descritos en la sección de requerimientos pueden haber sido agrupados en una única pantalla por razones de coherencia o de flujo de trabajo. En otros casos, puede haber pequeñas variaciones en los nombres de las subsecciones con respecto a los nombres de los casos de uso, ya que ciertos casos se han consolidado en una pantalla más general para facilitar la navegación o mejorar la experiencia del usuario. Cada subsección estará acompañada de una breve explicación sobre la relación con los casos de uso y la razón detrás de dicha organización.

6.7.1. Inicio sesión

La primera funcionalidad que encontramos en la aplicación es el inicio de sesión como se muestra en la imagen 6.1RF1.1. En este caso de uso, si el usuario ya tiene una cuenta registrada, deberá ingresar su correo electrónico y la contraseña asociada para acceder a todas las funcionalidades del sistema. Si

el usuario aún no tiene una cuenta, podrá hacer clic en el botón Regístrate", lo que lo redirigirá al proceso de registro de usuario para crear una nueva cuenta.

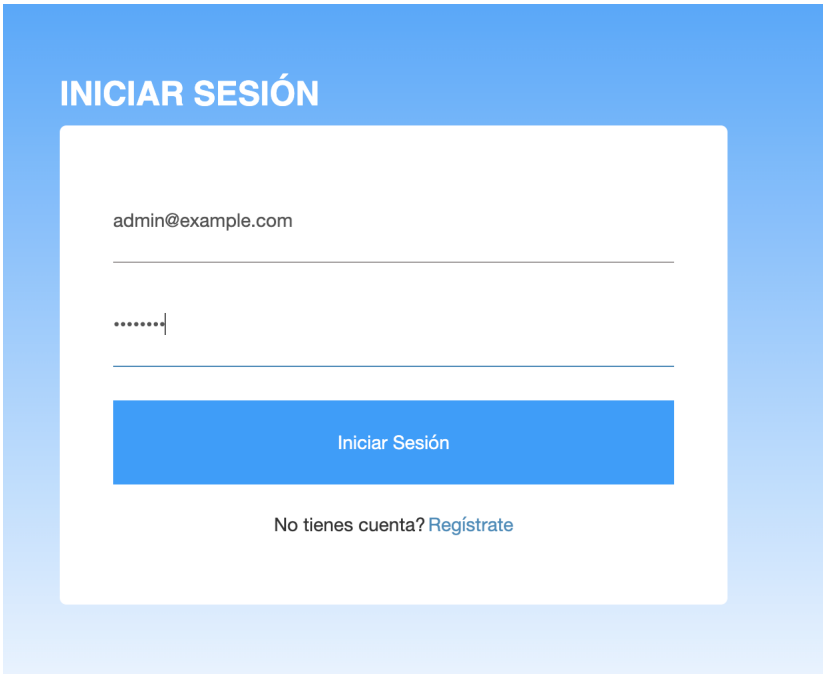

Figura 6.1: Inicio de sesion

6.7.2. Registro usuario

El caso de uso Registro de usuario presenta un formulario, como se muestra en la Figura 6.2[RF1.2]. en el cual el usuario debe completar sus datos personales, incluyendo correo electrónico, nombre, apellido y una contraseña que se utilizará para acceder a la plataforma posteriormente. En caso de que el usuario ingrese información incorrecta, como un correo electrónico que ya esté registrado en la plataforma o una contraseña que no coincida con el campo de confirmación de contraseña, se mostrarán los errores correspondientes para que el usuario los corrija.

Una vez que el formulario se complete correctamente, el sistema notificará al usuario que su cuenta ha sido creada exitosamente.

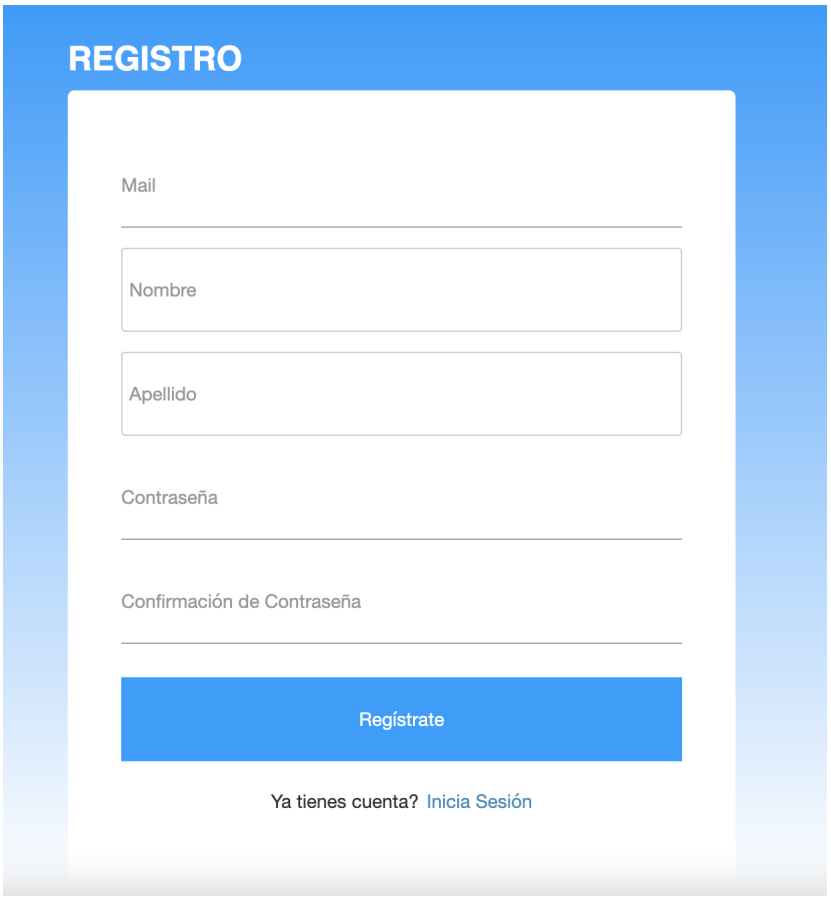

Figura 6.2: Registro de usuario

Tras iniciar sesión, el usuario es redirigido a la pantalla de *Explorar proyectos*RF2.3, donde podrá visualizar una lista de todos sus proyectos. En esta vista, se muestra información básica de cada proyecto, como se ilustra en la Figura 6.3.

Para cada proyecto, se podrá consultar el nombre, estado, creador, descripción, número de participantes y fecha de creación. En la barra lateral, el usuario encontrará tres opciones: *Mis proyectos*, que lo llevará a la vista de sus proyectos actuales; *Crear proyecto*, que redirige al caso de uso para crear un nuevo proyecto; e *Invitaciones pendientes*, que lo llevará al caso de uso donde podrá aceptar invitaciones para unirse a otros proyectos.

Figura 6.3: Explorar proyectos

6.7.3. Crear proyecto

RF2.1: Crear Proyecto El caso de uso *Crear proyecto*RF2.1 permite al usuario crear un nuevo proyecto en la plataforma. Para ello, el usuario debe completar un formulario en el que se solicita el nombre y la descripción del proyecto que desea crear, como se ilustra en la Figura 6.4.

Una vez que el proyecto haya sido creado exitosamente, el usuario será redirigido al caso de uso *Ver proyecto*, donde podrá visualizar los detalles del nuevo proyecto.

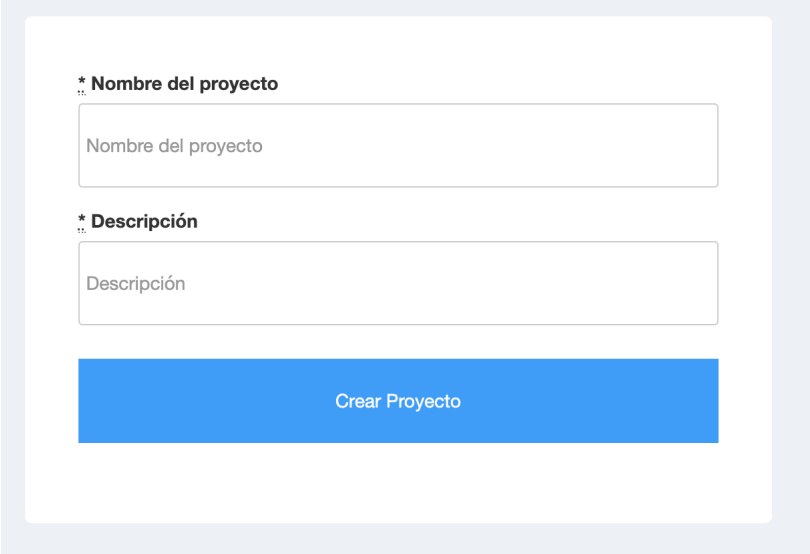

Figura 6.4: Formulario para crear un proyecto

6.7.4. General

En la barra superior de la pantalla principal de la aplicación se muestran seis pestañas que contienen las principales funcionalidades de la aplicacion.

En la pestaña *General* se implementan las funcionalidades básicas de la gestión de los proyectos. Se implementa el caso de uso *Ver proyecto*RF2.4, que muestra información general, como el nombre, la fase, la descripción y los marcos teóricos asociados, con su respectivo nombre, descripción y documento.
Además, se encuentra la opción *Cambiar fase*, que abre un modal para actualizar la fase del proyecto, lo que redirige al caso de uso *Actualizar fase del proyecto*RF2.5.
También se puede crear un nuevo marco teórico a través de la opción *Crear marco teórico*, lo que abre un modal para agregarlo al proyecto, además de ofrecer opciones para editar o eliminar marcos teóricos existentes.
Finalmente, se presenta una sección de *Bitácora*RF5.6, en la que se visualiza una línea temporal con todas las acciones realizadas en el proyecto. Esta sección incluye un filtro que permite organizar las acciones por tipo. Cada acción en la bitácora muestra una descripción básica, un símbolo de referencia, el usuario que la realizó y la fecha de realización.

A continuación se muestran algunas de las pantallas correspondientes a este caso de uso, Figura 6.5, Figura 6.6, Figura 6.7.

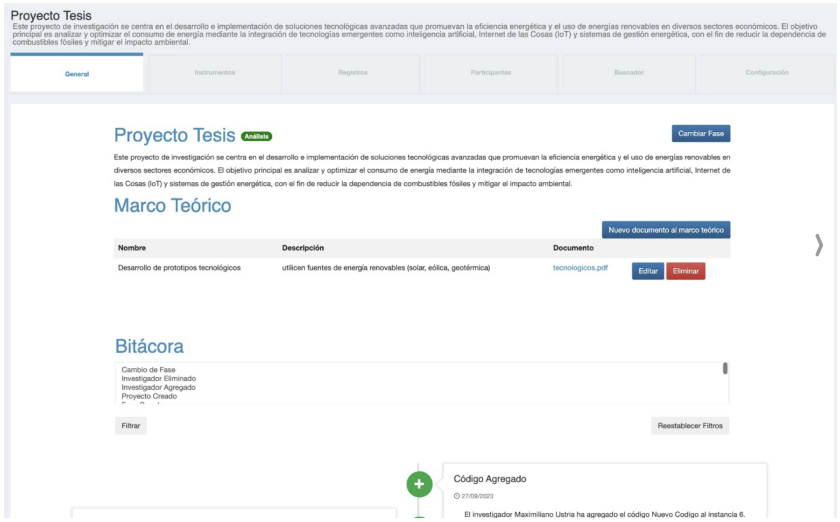

Figura 6.5: Pantalla *Ver proyecto* - Pestaña general

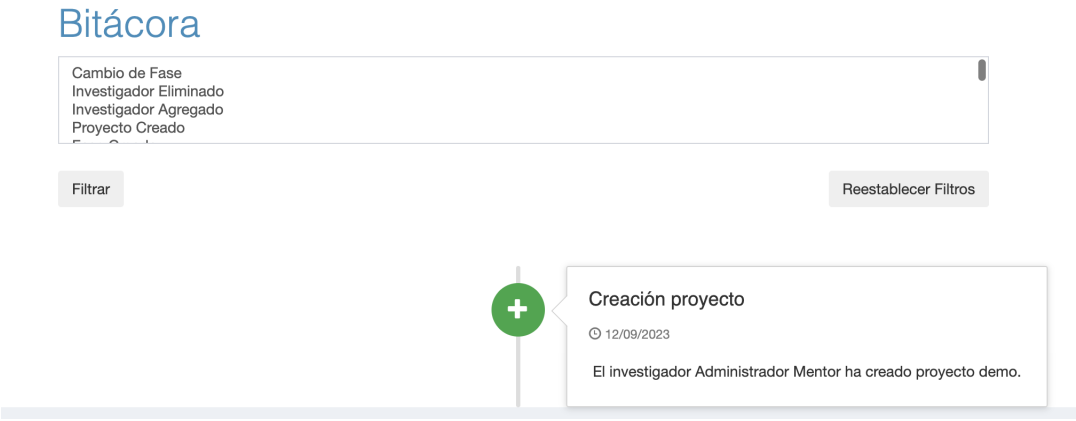

Figura 6.6: Filtrado de acciones en la bitácora

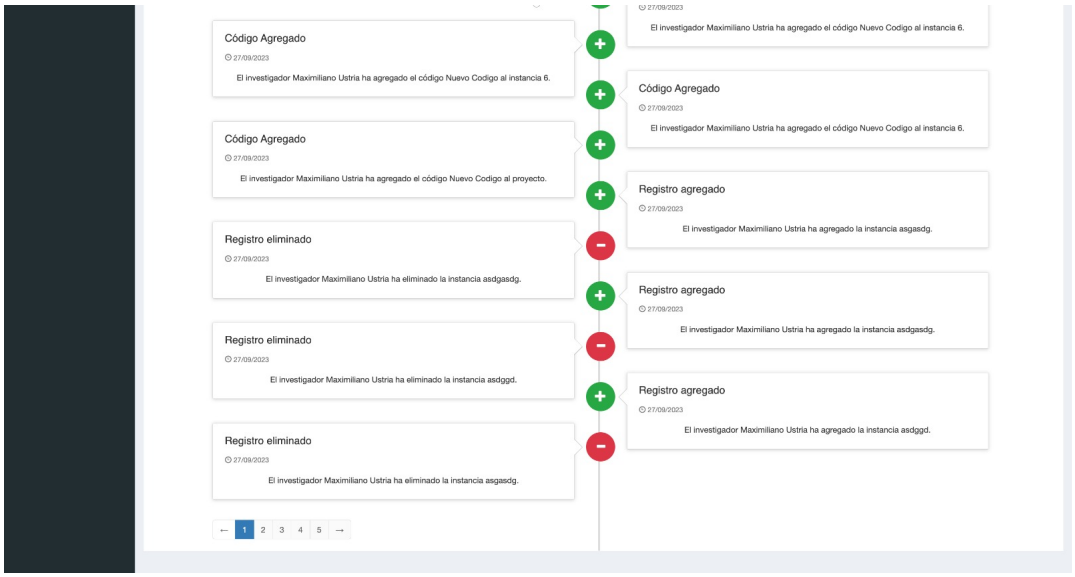

Figura 6.7: Eventos registrados en la bitácora

Cambiar fase Modal que despliega las fases disponibles en las que se puede encontrar el proyecto, permitiendo seleccionar una de estas como fase del proyecto, como se muestra en la Figura 6.8

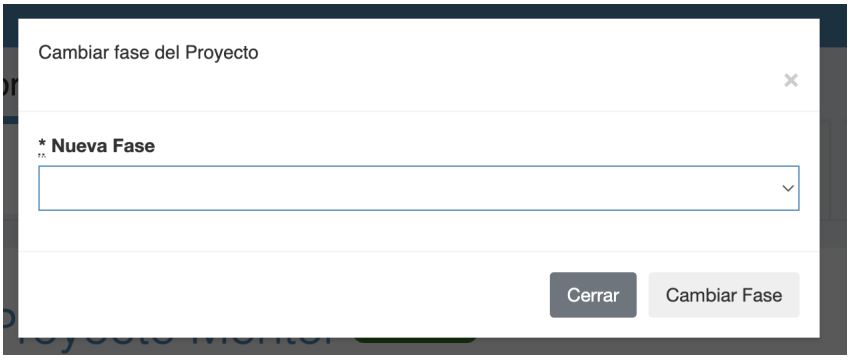

Figura 6.8: Formulario cambiar fase

Crear marco teórico El modal para agregar un nuevo marco teórico permite al usuario ingresar la información relevante para el proyecto. En este formulario, el usuario debe completar los siguientes campos: nombre, descripción y el archivo correspondiente, el cual puede ser cargado en cualquiera de los formatos

aceptados por la plataforma.

A continuación se muestra el formulario correspondiente para agregar un nuevo marco teórico en la Figura 6.9RF2.7.

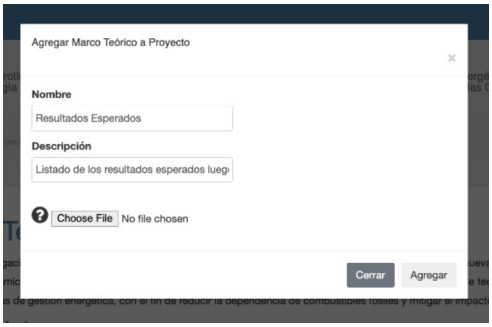

Figura 6.9: Formulario para agregar un nuevo marco teórico

Ver marco teórico Al presionar sobre un marco teórico, se redirige a la vista de marco teórico, la cual despliega nombre, descripción y códigos asociados, como muestra la Figura 6.10 .

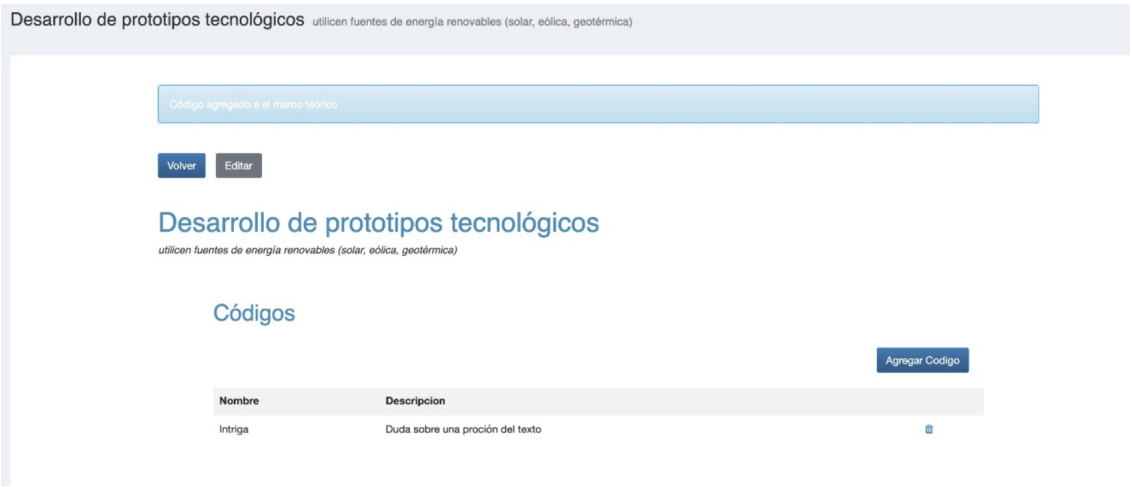

Figura 6.10: Ver marco teórico

Agregar código En el caso de uso *Agregar código*, genera un modal tal como el de la Figura 6.11 el cual muestra una lista de los códigos disponibles. Al seleccionar uno de estos códigos, este se asociará al marco teórico correspondiente.

Figura 6.11: Formulario para agregar un código a un marco teórico

Editar marco teórico El caso de uso *Editar marco teórico* permite modificar el nombre, la descripción y el documento asociado a un marco teórico. El usuario puede actualizar esta información según sea necesario, tal como se muestra en la Figura 6.12.

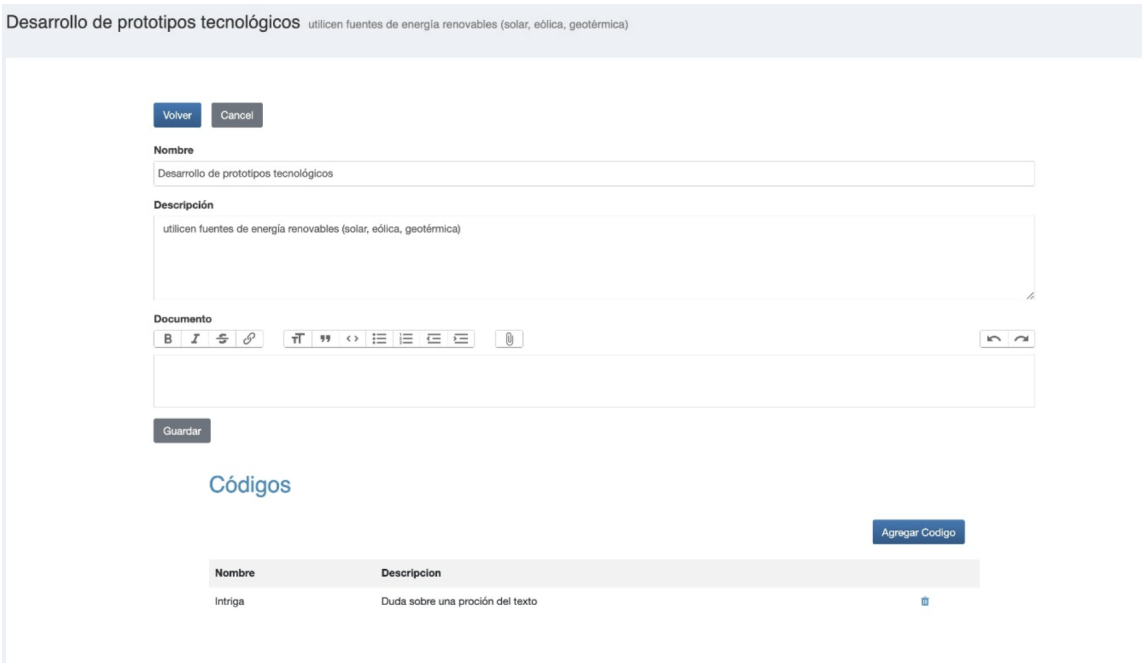

Figura 6.12: Formulario para editar un marco teórico

6.7.5. Instrumentos

En la pestaña *Instrumentos*, el sistema muestra un listado de los instrumentos disponibles en el proyecto, incluyendo su nombre, descripción y tipo, como se muestra en la Figura 6.13RF3.4. Además, se presentan varias acciones

disponibles:

- El botón *Agregar instrumento*RF3.1, que abre un modal para crear un nuevo instrumento.

- Para cada instrumento en la lista, hay dos acciones representadas por íconos: un ícono de ojo que redirige al caso de uso *Ver instrumento*, y un ícono de papelera que ejecuta el caso de uso *Eliminar instrumento*RF3.2 y elimina el instrumento del proyecto.

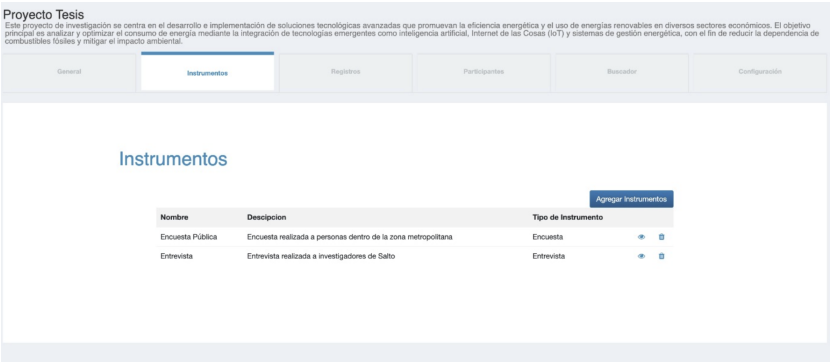

Figura 6.13: Pestaña de instrumentos en un proyecto

Agregar instrumento En el caso de uso *Agregar instrumento*, abre un modal que solicita información sobre el nuevo instrumento, incluyendo su nombre, descripción y tipo. Dependiendo del tipo seleccionado, pueden aparecer campos adicionales. En el caso de un instrumento de tipo *Entrevista*, se presentan opciones para crear secciones y preguntas, como se ilustra en la Figura 6.14.

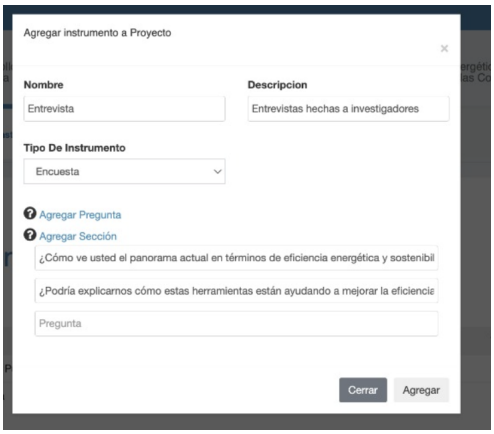

Figura 6.14: Formulario para agregar un nuevo instrumento

Ver instrumento Al seleccionar un instrumento de la lista, el usuario es redirigido a la vista *Ver instrumento*, que muestra información detallada sobre el instrumento, como su nombre, descripción y los códigos asociados. Además, ofrece la opción de agregar nuevos códigos a este instrumento. La vista se muestra en la Figura 6.15.

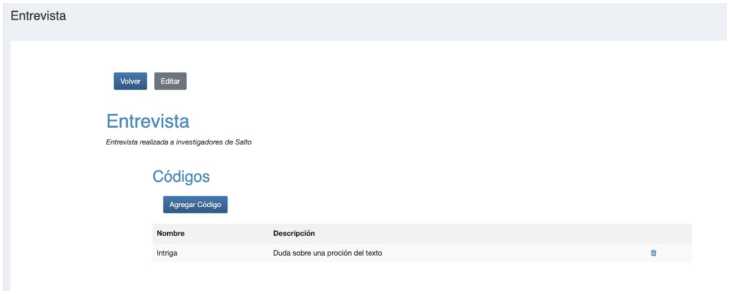

Figura 6.15: Vista de un instrumento

6.7.6. Registros

En la pestaña *Registros* permite gestionar el directorio de registros dentro del proyecto. Se presentan las carpetas y los registros creados, y el usuario puede acceder a cada uno de ellos. Además, se ofrece un filtro para buscar registros por tipo de instrumento, lo que facilita la navegación en proyectos grandes, como se muestra en la Figura 6.16. También se incluye un botón de acción *Agregar registro de instrumento*, que abre un modal al caso de uso *Agregar registro de*

instrumento.

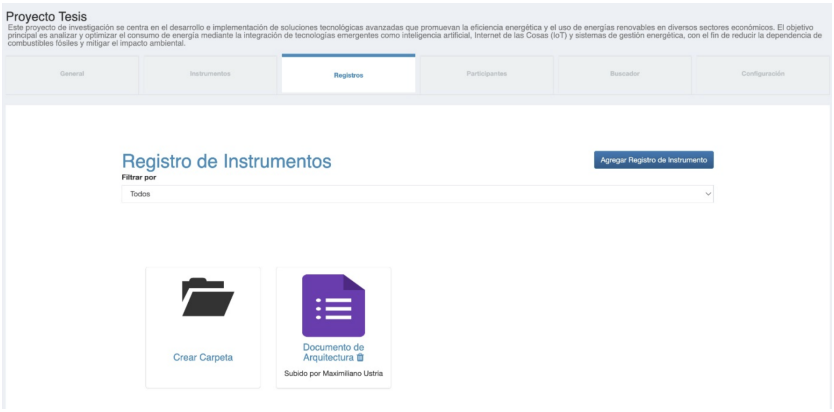

Figura 6.16: Pestaña de registros en un proyecto

Agregar registro de instrumento En el caso de uso *Agregar registro de instrumento*, el usuario debe seleccionar a qué instrumento se asociará el nuevo registro. Además, debe proporcionar un nombre y un archivo correspondiente al registro. Este proceso se ilustra en la Figura 6.17.

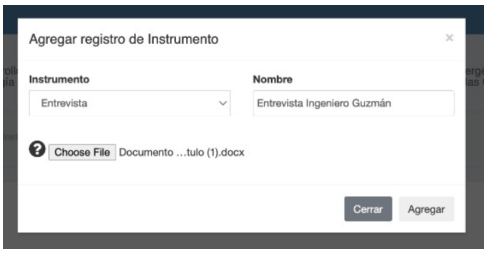

Figura 6.17: Formulario para agregar un registro a un instrumento

Ver registro Al presionar sobre un registro, el sistema redirige al usuario a la vista detallada del registro. En esta vista, se despliegan el nombre, la descripción y el contenido del documento ingresado. Además, se muestran los fragmentos asociados a los códigos del proyecto. Como se observa en la imagen 6.18, la interfaz presenta esta información organizada de forma clara.

En el caso de registros de tipo encuesta, la vista se ajusta para reflejar las preguntas y respuestas correspondientes, como se ilustra en la imagen 6.19.

Al presionar el botón .^Agregar código", se despliega un formulario al costado derecho que permite al investigador asociar un código al fragmento de texto seleccionado. En este formulario, el investigador puede proporcionar un nombre y una descripción para el código. Esto se visualiza en la imagen 6.20.

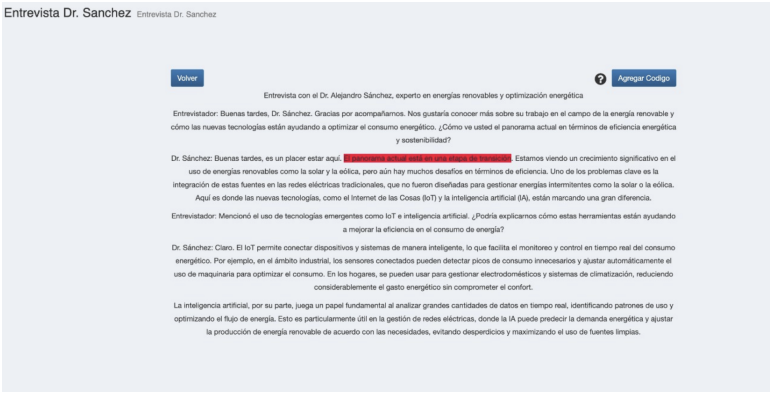

Figura 6.18: Ver registro

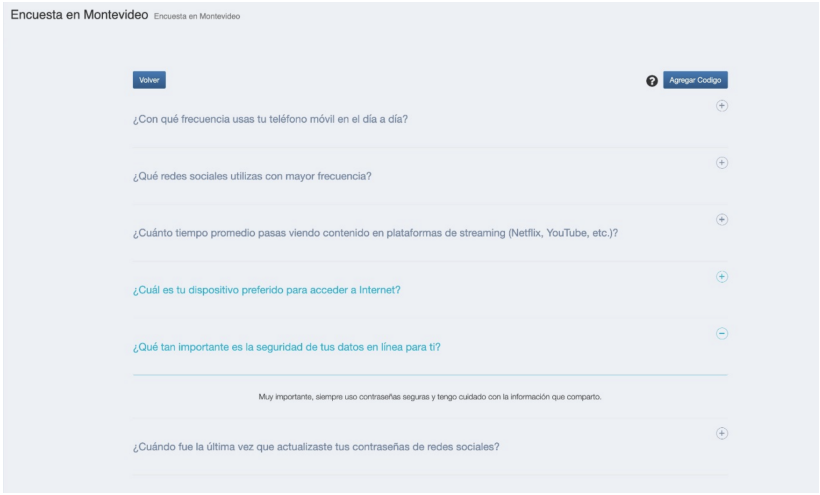

Figura 6.19: Ver registro de tipo encuesta

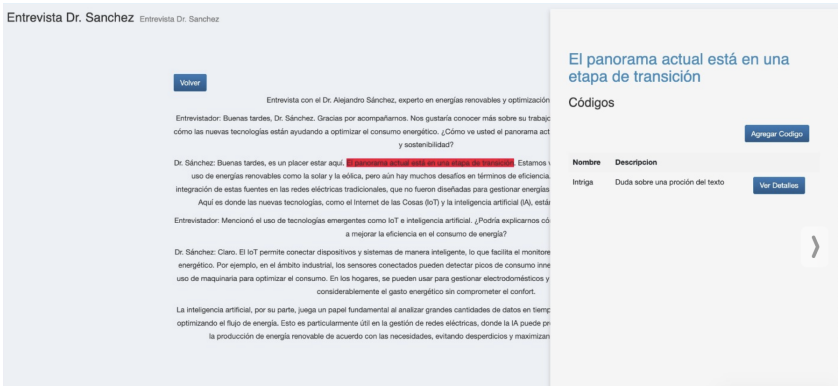

Figura 6.20: Agregar código a registro

6.7.7. Participantes

En la pestaña *Participantes*, el sistema muestra un listado con todos los participantes del proyecto, incluyendo información básica como su nombre, correo electrónico, rol y estado. En la parte superior, se incluye un buscador para filtrar a los participantes por nombre o correo electrónico. Además, se presentan filtros por fase y por rol, lo que facilita la gestión de los participantes. Se incluye un botón *Agregar investigador* que abre un modal para agregar nuevos participantes, como se muestra en la Figura 6.21RF2.10.

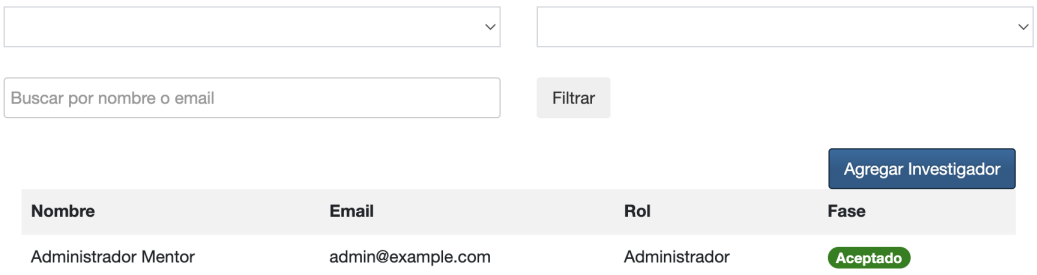

Figura 6.21: Listado de participantes de un proyecto

Agregar investigador El caso de uso *Agregar investigador* permite al usuario agregar un nuevo investigador al proyecto. Se presenta un formulario donde el

usuario debe especificar el rol del investigador y su correo electrónico. Una vez que el investigador es agregado, se le enviará una invitación para unirse al proyecto, como se muestra en la Figura 6.22RF2.8.

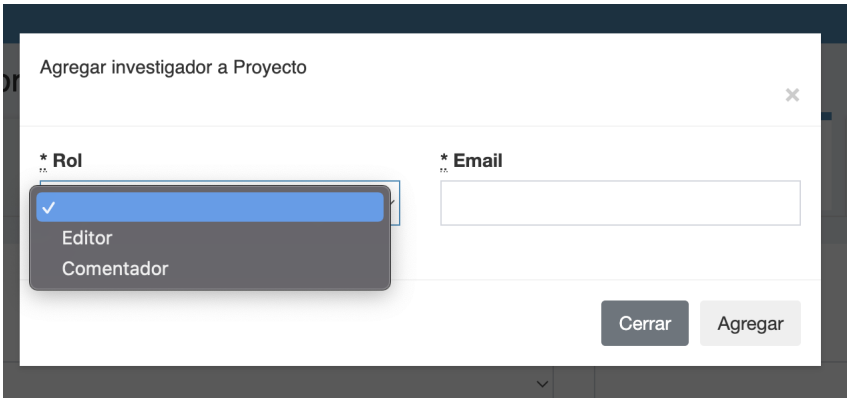

Figura 6.22: Formulario para agregar un nuevo investigador

6.7.8. Buscador de códigos

En la pestaña *Buscador de códigos*, se despliega un listado de todos los códigos aplicados en el proyecto, mostrando información como el nombre del código, el texto al que está asociado, el tipo de código y el marco teórico o registro correspondiente. También incluye un buscador que permite filtrar los códigos por nombre, lo que facilita la navegación en proyectos con muchos códigos, como se ilustra en la Figura 6.23RF4.4.

Figura 6.23: Buscador de códigos en el proyecto

6.7.9. Configuración

En la pestaña *Configuración* muestra las configuraciones de las fases y los códigos del proyecto. En la sección de fases, se presenta un listado de todas las fases creadas, con la posibilidad de eliminar fases mediante el ícono de papelera. También incluye un botón *Agregar fase*, que abre un modal para agregar una nueva fase al proyecto. En la sección de códigos, se listan todos los códigos creados, junto con acciones como *Importar código*, *Crear código*, *Unir códigos*, *Hacer subcódigos* y *Eliminar código*. Además, se incluye un buscador para encontrar códigos por nombre, como se muestra en la Figura 6.24.

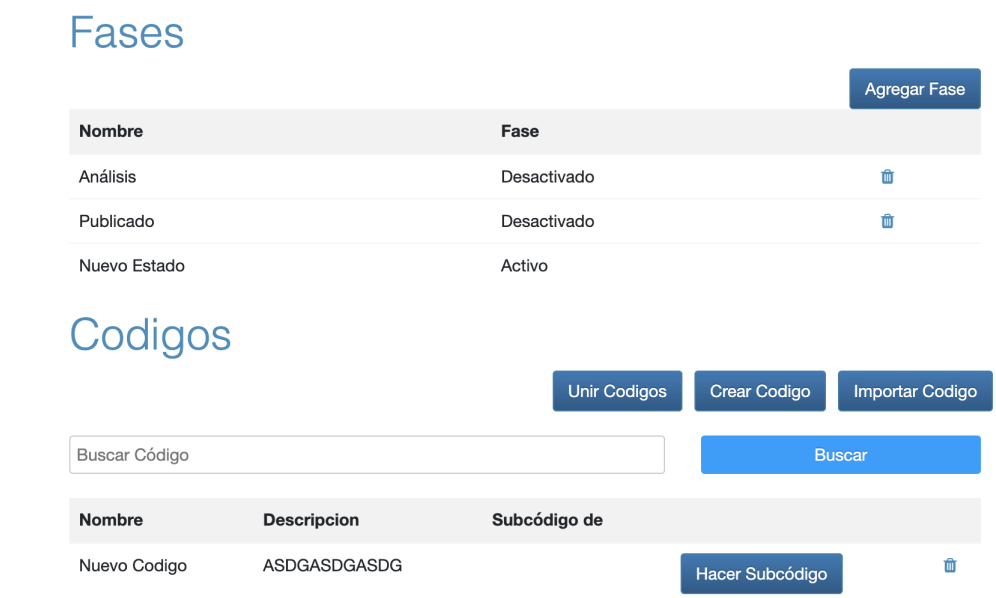

Figura 6.24: Configuración del proyecto

Agregar fase En el caso de uso *Agregar fase*, se presenta un formulario que permite seleccionar una de las fases creadas en el proyecto y cambiar su estado a activo. Al presionar el botón *Cambiar fase*, la fase seleccionada se establece como activa, como se muestra en la Figura 6.25.

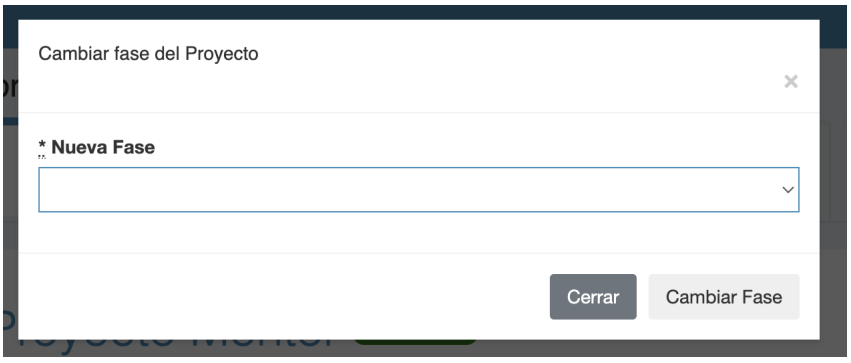

Figura 6.25: Formulario para cambiar la fase del proyecto

Crear código En el caso de uso *Crear código*, el usuario puede agregar un nuevo código al proyecto proporcionando un nombre y una descripción. Este proceso está ilustrado en la Figura 6.26RF4.1.

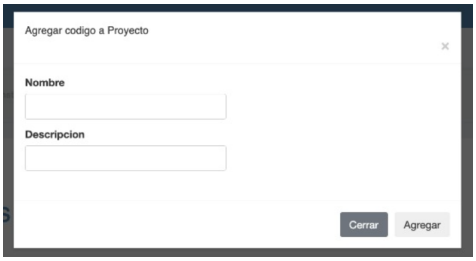

Figura 6.26: Formulario para crear un código en el proyecto

Unir códigos El caso de uso *Unir códigos* permite al usuario combinar dos códigos. Como muestra la Figura 6.27RF4.3.

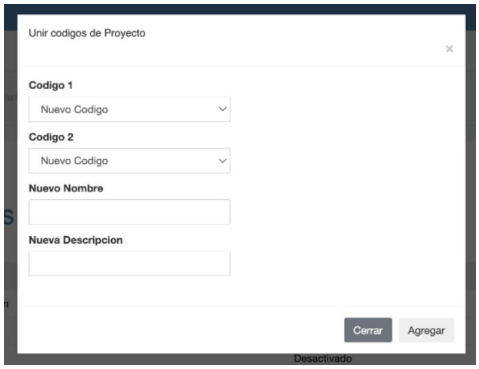

Figura 6.27: Formulario para unir códigos en el proyecto

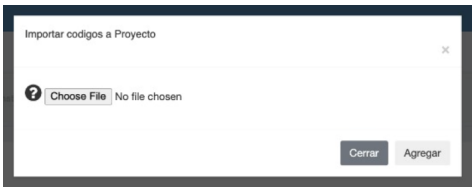

Figura 6.28: Formulario para importar códigos

Importar codigo

6.7.10. Aceptar invitación a proyecto

En esta pantalla observamos un listado de los proyectos de los cuales el investigador recibió invitación, con su nombre, administrador del proyecto, y el rol con el cual nos uniríamos al proyecto, por último en la parte derecha tenemos dos iconos, el primero que al ser presionado corresponde a aceptar la invitación y la papelera que corresponde a rechazar la invitación. Como se muestra en la Figura 6.29

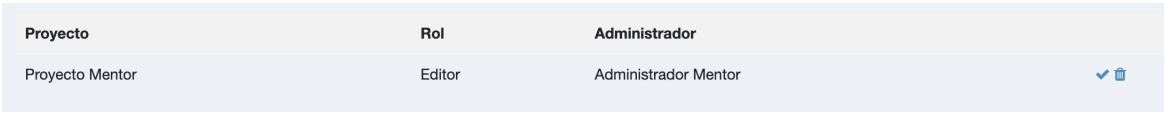

Figura 6.29: Aceptar invitación a proyecto

6.8. Escenarios de prueba ejecutados

Al finalizar la implementación se ejecutaron escenarios de extremo a extremo, documentando objetivo, datos, pasos, resultado esperado y observado.

Escenario 1: Proyecto con múltiples investigadores y entrevistas **Objetivo:** validar invitaciones, roles, carga de entrevistas y codificación.
Datos/Setup: 1 proyecto, 3 usuarios (admin, editor, comentador), entrevistas en PDF.
Pasos: crear proyecto; invitar editor y comentador; aceptar invitaciones; cargar entrevistas; aplicar códigos a fragmentos.
Esperado: roles efectivos; entrevistas visibles; códigos listados en buscador y vinculados a registros.
Observado: PASS. Invitaciones y roles OK; entrevistas visibles; códigos indexados y consultables en buscador; la bitácora registra las acciones realizadas (creación, invitaciones, cargas y codificaciones).
Observaciones: se ajustó mensaje de estado al aceptar invitación para mayor claridad.

Escenario 2: Gestión de instrumentos y registros **Objetivo:** validar creación de instrumentos (entrevista/encuesta) y registros asociados.
Datos/Setup: instrumentos de tipo entrevista (con secciones) y encuesta (con preguntas), y registros asociados.
Pasos: crear instrumentos; crear registros de entrevista (texto/PDF) y respuestas de encuesta; navegar y visualizar.
Esperado: estructura de secciones/preguntas visible; registros listados y accesibles.
Observado: PASS. Todos los registros creados se visualizan; preguntas de encuesta renderizadas con respuestas; navegación estable.
Observaciones: se mejoró el orden por fecha en listados para facilitar la revisión.

Escenario 3: Codificación y cruce de códigos **Objetivo:** validar aplicación de códigos en entrevistas y encuestas y su cruce.
Datos/Setup: conjunto de códigos en marco teórico; aplicaciones en entrevistas y encuestas.
Pasos: etiquetar fragmentos y respuestas; consultar buscador; cruzar por códi-

go y tipo de recurso.
Esperado: coincidencias por código en ambos tipos; conteos coherentes; acceso a cada recurso.
Observado: PASS. El buscador retorna coincidencias coherentes; el cruce por códigos (p. ej., "privacidad") muestra ocurrencias en entrevistas y encuestas con acceso a cada fuente.
Observaciones: se añadió badge de tipo de recurso en resultados del buscador para distinguir origen.

Escenario 4: Configuración y fases **Objetivo:** validar creación/cambio de fases y trazabilidad.
Datos/Setup: varias fases; 1 proyecto activo.
Pasos: crear fases; cambiar fase activa; revisar bitácora/estado.
Esperado: fase activa coherente; eventos de bitácora completos.
Observado: PASS. La fase activa se actualiza y persiste; eventos registrados correctamente.
Observaciones: sin incidencias.

Escenario 5: Permisos y visibilidad **Objetivo:** verificar restricciones por rol.
Datos/Setup: 3 roles en el mismo proyecto.
Pasos: intentar crear/editar/eliminar como cada rol; acceder a participantes.
Esperado: admin y editor pueden editar; comentador solo lectura; visibilidad de participantes.
Observado: PASS. Se impiden acciones no permitidas; vistas de solo lectura operativas; listado de participantes visible para roles autorizados.
Observaciones: se clarificaron tooltips en botones deshabilitados.

Escenario 6: Rendimiento básico **Objetivo:** validar tiempos de respuesta en listados/buscador.
Datos/Setup: conjunto de códigos, registros y aplicaciones suficiente para validar paginación y búsquedas.
Pasos: abrir listados (proyectos, registros, códigos); ejecutar búsquedas por nombre de código.
Esperado: tiempos de respuesta aceptables en entorno de pruebas.
Observado: PASS. Tiempos de respuesta aceptables en el entorno de prueba local.

Observaciones: recomendable paginar registros ¿100 para entornos con menor hardware.

Escenario	**Estado**	**Notas**
Múltiples investigadores y entrevistas	PASS	Bitácora registra acciones esperadas
Instrumentos y registros	PASS	Registros visibles y navegables
Codificación y cruce de códigos	PASS	Cruces y conteos coherentes
Configuración y fases	PASS	Fase activa consistente
Permisos y visibilidad	PASS	Restricciones por rol correctas
Rendimiento básico	PASS	Tiempos de respuesta aceptables

Tabla 6.1: Resumen de resultados de escenarios de prueba.

Capítulo 7

Conclusiones

El desarrollo de este proyecto permitió la creación de una plataforma destinada a la gestión y análisis de proyectos de investigación que integran metodologías cualitativas y cuantitativas. A través de la aplicación, se buscó satisfacer la necesidad de un entorno centralizado y colaborativo que permita tanto la gestión y almacenamiento como el análisis cruzado de datos de diversa naturaleza; por ejemplo, cruzar códigos aplicados en entrevistas semiestructuradas (p. ej., .ªcceso a internet", "privacidad") con variables de encuestas (edad, género, nivel educativo) y con metadatos del proyecto (fase, ubicación), para identificar patrones concretos como diferencias por tramo etario o por rol del investigador. Este enfoque proporciona una base sólida para proyectos de investigación complejos, ya que permite obtener todos los recursos con determinado código, generar nuevos subcódigos y analizar el conjunto total de recursos según un código específico.

7.1. Adecuación de la Propuesta

La solución desarrollada es adecuada debido a que, al estar específicamente diseñada para la integración de análisis cualitativos, logra satisfacer los requerimientos de un ambiente de datos que facilita la gestión de información en investigaciones, permitiendo declarar todas las caracterisiticas importantes, nombre, fecha de creacion, investigador responsable, las fases importantes del proyecto, participantes del proyecto, una bitácora donde se puede ver todo el historial. A diferencia de otras plataformas, como OSF, que también permite la gestión

y almacenamiento de datos de investigación, la aplicación desarrollada permite gestionar los datos correspondientes a entrevistas y encuestas, así como la posibilidad de generar instrumentos customizables. También permite asignarles códigos y poder filtrar y buscar por estos códigos en todos los documentos del proyecto. Esto otorga a los investigadores una mayor flexibilidad, permitiéndoles trabajar con datos cualitativos y cuantitativos sin necesidad de cambiar de software ni realizar exportaciones adicionales, como en el caso de OSF, que carece de funciones avanzadas para el análisis de datos en su versión básica.

A pesar de estas ventajas, *MENTOR* presenta limitaciones, especialmente en lo que respecta a su falta de interoperabilidad con herramientas ampliamente utilizadas como las de la suite de Google y el propio OSF, que facilita la conexión y sincronización con diversas plataformas. Para competir con soluciones más establecidas y de uso extendido, será necesario en futuras iteraciones mejorar esta capacidad de integración, logrando que la adopción de la herramienta sea más fluida para investigadores que ya emplean otros sistemas en su flujo de trabajo.

7.2. Limitaciones de la Solución

Si bien *MENTOR* ofrece una solución eficaz en la gestión de datos cualitativos y cuantitativos de manera integrada, existen varias limitaciones:

- **Interoperabilidad**: La capacidad de integrarse con otras herramientas como OSF o Google Drive es limitada en esta primera versión. Esta restricción se debe principalmente a que el enfoque inicial del desarrollo estuvo centrado en implementar las funcionalidades esenciales del sistema y asegurar su correcto funcionamiento en un entorno autónomo. Además, la integración con plataformas externas requiere la implementación de APIs específicas, protocolos de autenticación (como OAuth), y el manejo de formatos de datos estandarizados para garantizar una sincronización confiable.

  Para mejorar la interoperabilidad en futuras versiones, sería necesario dedicar esfuerzos al desarrollo de módulos de integración que permitan la comunicación fluida con servicios externos, incluyendo mecanismos de importación/exportación de datos, autenticación segura y mantenimiento de sincronía entre plataformas. También se recomienda evaluar el uso

de estándares abiertos para el intercambio de datos, lo cual facilitaría la compatibilidad con un mayor número de herramientas académicas.

- **Escalabilidad**: Dado que la aplicación fue diseñada con un enfoque específico en análisis principalmente cualitativo, su escalabilidad en proyectos de investigación donde predominan exclusivamente los datos cuantitativos podría no ser tan robusta como otras plataformas más generalistas. Ya que no es posible realizar un análisis macro de los datos, se estudian las instancias de instrumentos como entes individuales.

- **Capacidades de Visualización Avanzada**: La plataforma carece de herramientas avanzadas de visualización de datos, lo cual limita la capacidad de los investigadores para identificar patrones complejos a partir de grandes volúmenes de información, algo que puede ser vital en ciertas áreas de investigación social. Un ejemplo de esto, es la incapacidad de poder visualizar los resultados obtenidos mediante gráficas o tablas.

Estas limitaciones ofrecen oportunidades para mejorar la herramienta en futuras versiones, de manera que pueda expandirse para cubrir más ampliamente las necesidades de los investigadores y se convierta en una alternativa viable a plataformas de mayor alcance, como OSF.

7.3. Aprendizaje y Reflexión del Equipo de Trabajo

Al desarrollar este proyecto, el equipo adquirió experiencia valiosa en el diseño y construcción de una herramienta de gestión de datos de investigación híbridos. La experiencia nos permitió comprender la importancia de definir y revisar continuamente los requerimientos del sistema, además de adquirir habilidades para adaptarse a cambios y ajustes necesarios en el proceso de desarrollo.

Entre los aprendizajes clave están:

- La importancia de un desarrollo iterativo, donde la retroalimentación constante de las tutoras y las demostraciones periódicas contribuyeron a la validación y ajuste de las funcionalidades.

- La comprensión de los retos y beneficios de trabajar con datos cualitativos, donde aprendimos sobre la complejidad que conlleva lograr un entorno

flexible y adecuado para este tipo de análisis. Así como lograr entender como se trabaja en una investigación para poder dar un producto que se adapte lo mejor posible.

- La necesidad de contemplar aspectos de interoperabilidad con algunas de las herramientas ya existentes y escalabilidad desde las primeras etapas de diseño, dado que estas características impactan fuertemente en la adopción de la herramienta por parte de los usuarios finales.

7.4. Comparación con el Proyecto Propuesto

El desarrollo de un ambiente de datos para metodologías híbridas no parte de cero. Existen herramientas y plataformas que cubren algunos aspectos del problema, pero ninguna aborda completamente las necesidades específicas de este enfoque. A continuación, se presentan las características de las herramientas más utilizadas y su comparación con el sistema propuesto:

Herramienta	**Cualitativo**	**Cuantitativo**	**Colaboración**	**Integración**	**Costo**	**Flexi**
Atlas.ti	Excelente	Limitado	No	No	Alto	M
NVivo	Excelente	Moderado	No	Parcial	Alto	I
OSF	Moderado	Moderado	Sí	No	Bajo	A
Google Drive	Bajo	Bajo	Sí	No	Bajo	A
Propuesta	Excelente	Limitado	Sí	Sí	Moderado	A

Tabla 7.1: Comparación de herramientas existentes y la propuesta.

El ambiente de datos propuesto busca superar estas limitaciones, proporcionando una solución integrada que combine las funcionalidades avanzadas de análisis cualitativo como el etiquetado de códigos en documentos y análisis del mismo, asi como poder cruzar estos códigos en los distintos documentos de la investigación; Con capacidades de colaboración, gestión y almacenamiento accesible, optimizado para investigaciones interdisciplinarias.

7.5. Conclusión Final

En conclusión, el proyecto representa un paso importante hacia la creación de un ambiente de datos flexible y eficiente para investigaciones con metodologías híbridas. Aunque aún queda pendiente la validación de la herramienta en un entorno de producción más amplio, confiamos en que las funcionalidades desarrolladas hasta el momento constituyen una base sólida. A medida que más investigadores adopten la plataforma y aporten retroalimentación, se podrán realizar mejoras en la usabilidad de la aplicación, para lograr que sea utilizada como una herramienta clave para la investigación académica y social.

Capítulo 8

Trabajo futuro

El desarrollo de este proyecto ha permitido establecer las bases para futuras mejoras e investigaciones. A continuación, se presentan algunas áreas clave en las que se podría expandir el trabajo realizado.

8.1. Generación de una API Pública

Para optimizar la experiencia del cliente, se decidió exponer una API pública que ofrezca las mismas funcionalidades de la aplicación web. En primera instancia, esta idea surgió como una integración a otro proyecto *EDIGA* (Alberti, Toledo, Nocetti, y Font, 2025), llevado a cabo por otro grupo de tesis de grado de la facultad, cuyo objetivo es gestionar un proyecto de investigación en particular, con funcionalidades especializadas para ese proyecto. Debido a que el *proyecto MENTOR* (?, ?) tiene como objetivo abarcar todos los proyectos de investigación, y ante la necesidad de que el proyecto *EDIGA* también sea gestionado dentro de *MENTOR*, surgió la idea de evitar retrabajos para los investigadores de *EDIGA*, quienes de otro modo tendrían que cargar los datos en ambas plataformas. Para solucionar esto, se podria generar una API pública que permitirá a *EDIGA* (Alberti y cols., 2025) y otros proyectos replicar los datos y las acciones que correspondan.

De esta manera, el proyecto *EDIGA*, a medida que ingrese nuevos datos y análisis, podrá compartir estos y estimular la retroalimentación con *MENTOR*. *MENTOR* tendrá una réplica de los datos pertinentes, permitiendo que el proyecto *EDIGA* se beneficie de ambas plataformas.

Si bien este es el caso específico que motivó la creación de la API pública en un futuro, consideramos que su utilidad puede extenderse a futuros desarrollos. La API pública proporcionará acceso a las funcionalidades del sistema, permitiendo que otros proyectos o investigadores interactúen con los datos generados y generen asociaciones entre códigos, registros y marcos teóricos, optimizando la interoperabilidad y colaboración en investigaciones futuras.

8.2. Utilización de la Gema Neo4j para la Gestión de Bases de Datos de Grafos

Para mejorar la capacidad de análisis y representación de las relaciones entre las entidades principales del proyecto, como códigos, registros y marcos teóricos, es fundamental utilizar una base de datos de grafo. Este tipo de base de datos es ideal para modelar redes complejas de relaciones, permitiendo representar y explorar de manera más natural las conexiones entre diferentes unidades de información. A diferencia de las bases de datos relacionales tradicionales, las bases de datos de grafos están optimizadas para manejar datos interconectados, lo cual facilita la identificación de patrones, conexiones ocultas y estructuras de red dentro de los datos del proyecto.

Entre las diversas opciones disponibles, Neo4j (*Neo4j*, s.f.-b) destaca como una implementación potente y versátil de base de datos de grafo. Neo4j permite crear asociaciones complejas entre las entidades, como los códigos y los registros, y ofrece herramientas avanzadas para la visualización de estas relaciones. Esto enriquece el análisis al permitir consultas más específicas y detalladas, como la exploración de relaciones directas e indirectas entre códigos y marcos teóricos. Además, la utilización de Neo4j posibilita la incorporación de nuevas funcionalidades en el sistema, mejorando así las capacidades de investigación y análisis.

8.3. Generación de Métricas

Otro aspecto importante para el futuro desarrollo de este proyecto es la generación de métricas que midan el impacto y la relevancia de las asociaciones entre códigos, registros y marcos teóricos. Estas métricas permitirán evaluar la calidad de las relaciones generadas, así como el rendimiento del sistema en términos de precisión, relevancia y eficiencia.

Entre las métricas específicas que podrían implementarse se incluyen:

- **Grado de centralidad**: Mide cuán central es un nodo (código o registro) en la red, ayudando a identificar los elementos más influyentes o cruciales en el proyecto.
- **Coeficiente de agrupamiento**: Evalúa la tendencia de los nodos a formar grupos cerrados o clústeres, lo cual puede revelar la existencia de subtemas o categorías estrechamente relacionadas.
- **Distancia promedio entre nodos**: Determina la distancia media entre los nodos en el grafo, proporcionando una medida de la interconectividad general del sistema.
- **Precisión de relaciones**: Calcula la proporción de relaciones relevantes correctamente identificadas en comparación con el total de relaciones sugeridas por el sistema.
- **Relevancia temática**: Evalúa la pertinencia de las asociaciones basadas en el contexto temático, asegurando que las conexiones entre códigos, registros y marcos teóricos sean significativas y útiles para la investigación.
- **Eficiencia de consulta**: Mide el tiempo y los recursos necesarios para realizar consultas complejas en la base de datos, asegurando que el sistema opere de manera óptima incluso con grandes volúmenes de datos.

Estas métricas proporcionarán un marco integral para evaluar y mejorar continuamente el sistema, asegurando que las asociaciones generadas sean no solo precisas, sino también útiles para el avance del proyecto de investigación.

También se puede tomar en cuenta la creación de métricas relacionadas con la gestión del proyecto de investigación, como:

- Tiempos de entrega de resultados clave del proyecto. Por ejemplo, comparar el promedio estimado en proyectos anteriores con el que se obtiene usando la aplicación.
- Nivel de cumplimiento con los objetivos establecidos (para modelar la métrica es necesario tener los objetivos).
- Participación y colaboración de los investigadores en las actividades gestionadas dentro de la plataforma.

La implementación de estas métricas permitirá una evaluación constante y objetiva del sistema, facilitando la toma de decisiones para su mejora y optimización en futuros desarrollos.

Glosario

ACID ACID es un acrónimo que define las propiedades fundamentales que garantizan la fiabilidad de las transacciones en bases de datos: Atomicidad (garantiza que todas las operaciones de una transacción se completen con éxito o ninguna se completa), Consistencia (mantiene la integridad de los datos antes y después de cada transacción), Aislamiento (garantiza que el resultado de una transacción no se ve afectado por otras transacciones concurrentes) y Durabilidad (asegura que los cambios realizados por una transacción persistan incluso en caso de fallo del sistema)..

backend El backend se refiere a la parte de un sistema informático o aplicación que no es directamente accesible por los usuarios finales. Incluye la lógica de la aplicación, la base de datos y otros componentes internos..

backlog El backlog (o lista de pendientes) en Scrum es una lista priorizada de requisitos o funcionalidades que se deben implementar en un producto. Estos elementos pueden ser características, mejoras, correcciones de errores o cualquier otro trabajo que aporte valor al producto..

base de datos Una base de datos es un sistema organizado para almacenar y gestionar conjuntos de datos. Permite el almacenamiento, actualización y consulta eficiente de la información, proporcionando un entorno para la gestión de datos seguros y estructurados..

cache El cache es una técnica de almacenamiento temporal de datos en una ubicación accesible rápidamente, para mejorar el rendimiento al reducir el tiempo de acceso a los datos. Se utiliza para almacenar copias de datos frecuentemente accesados o resultados de operaciones costosas..

csv CSV (Comma-Separated Values) es un formato de archivo que se utiliza para almacenar datos tabulares en forma de texto plano. Cada línea del archivo representa una fila de datos, y los valores de cada columna están separados por comas u otros delimitadores. Es ampliamente utilizado para importar y exportar datos entre diferentes aplicaciones y sistemas de gestión de bases de datos..

doc Formato de archivo utilizado por Microsoft Word para documentos de texto. Los archivos .doc contienen texto formateado, imágenes, tablas y otros elementos que conforman un documento. Es uno de los formatos más utilizados para compartir documentos escritos..

framework Una estructura conceptual y tecnológica utilizada para desarrollar y organizar software. Los frameworks proporcionan una base de código estándar, incluyendo bibliotecas y mejores prácticas, que ayudan a los desarrolladores a construir aplicaciones de manera más eficiente y consistente..

frontend El frontend es la parte de un sistema informático o aplicación con la cual interactúan los usuarios finales. Incluye la interfaz gráfica de usuario y otros componentes visibles para el usuario..

grafos En el contexto de estructuras de datos, un grafo es un conjunto de vértices (nodos) conectados por aristas (arcos). Los grafos son utilizados para representar relaciones entre entidades y permiten modelar problemas en diversas áreas como redes, rutas de navegación y estructuras de datos..

SCRUM Metodología ágil de desarrollo de software.

sprints En Scrum, sprints son periodos cortos de tiempo (generalmente de una a cuatro semanas) en los cuales se desarrolla y entrega un conjunto de funcionalidades de un producto. Cada sprint tiene un objetivo específico y al final se produce un incremento potencialmente entregable..

SQL SQL (Structured Query Language) es un lenguaje de programación utilizado para gestionar y manipular bases de datos relacionales. Permite realizar consultas, actualizaciones, inserciones y eliminaciones de datos en una base de datos, siguiendo un conjunto de normas y convenciones..

TDD Desarrollo Dirigido por Pruebas.

Referencias

Active admin. (s.f.). https://activeadmin.info/.

Alberti, S., Toledo, C., Nocetti, L., y Font, M. (2025). *Proyecto de grado "ambiente de datos para el proyecto ediga".* (Disponible en: URL o repositorio del proyecto, si aplica.)

Atlas ti. (s.f.). https://atlasti.com/es.

Cd. (s.f.). https://aws.amazon.com/es/devops/continuous-delivery/.

Ci. (s.f.). https://books.google.com.uy/books?id=hYvcDwAAQBAJ&pg=PT26&dq=desarrollo+dirigido+por+tests&hl=es&sa=X&ved=2ahUKEwjC5_qg9or7AhUOrZUCHWJmCKYQ6AF6BAgIEAI#v=onepage&q=desarrollo%20dirigido%20por%20tests&f=false.

Circle ci. (s.f.). https://circleci.com/.

Creswell, J. W., y Plano Clark, V. L. (2017). *Designing and conducting mixed methods research* (3rd ed.). Sage Publications.

Css. (s.f.). https://developer.mozilla.org/es/docs/Web/CSS.

Devise. (s.f.). https://github.com/heartcombo/devise.

Docker. (s.f.). https://aws.amazon.com/es/docker/.

Github. (s.f.). https://github.com/.

Github flow. (s.f.). https://docs.github.com/en/get-started/using-github/github-flow.

Google drive. (s.f.). https://www.google.com/intl/es/drive/.

Html. (s.f.). https://developer.mozilla.org/es/docs/Web/HTML.

Iso 25010. (s.f.). https://iso25000.com/index.php/normas-iso-25000/iso-25010.

Java script. (s.f.). https://developer.mozilla.org/es/docs/Web/JavaScript.

Neo 4j. (s.f.-a). https://neo4j.com/.

Neo4j. (s.f.-b). https://neo4j.com/use-cases/.

Osf. (s.f.). https://osf.io/.

Postgres sql. (s.f.). https://www.postgresql.org/.

Postgres sql - rails. (s.f.). https://rubygems.org/gems/pg/versions/0.18.4?locale=es.

Pruebas. (s.f.). https://books.google.com.uy/books?id=hYvcDwAAQBAJ&pg=PT26&dq=desarrollo+dirigido+por+tests&hl=es&sa=X&ved=2ahUKEwjC5_qg9or7AhUOrZUCHWJmCKYQ6AF6BAgIEAI#v=onepage&q=desarrollo%20dirigido%20por%20tests&f=false.

Redis. (s.f.). https://redis.io/es/.

Rspec. (s.f.). https://github.com/rspec/rspec-rails.

Ruby on rails. (s.f.). https://rubyonrails.org/.

Scrum. (s.f.-a). https://www.atlassian.com/es/agile/scrum.

Scrum. (s.f.-b). https://aws.amazon.com/es/what-is/scrum/.

Trello. (s.f.). https://trello.com/guide/trello-101.