Primer parcial

Universidad del Cauca

Vigilada Mineducación

Ingeniería de Software II

Presentado por:

Jose David Arteaga Fernández

Juan Camilo Benavides Salazar

Juan Esteban Chavez Collazos

Jhoan Sebastian Garcia Camacho

Juan Diego Perez Martinez

Profesor:

Wilson Libardo Pantoja

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Ingeniería de Sistemas

Popayán, Febrero de 2025

CONTENIDO

Resumen

El Sistema de Gestión de Proyectos Académicos para Empresas es una plataforma web diseñada para fortalecer la vinculación entre la universidad y el sector empresarial. Su objetivo es facilitar la gestión y asignación de proyectos empresariales a estudiantes de últimos semestres, permitiéndoles desarrollar soluciones de software reales como parte de sus prácticas profesionales, pasantías o proyectos académicos.

La plataforma permite a las empresas registrarse, publicar sus necesidades de software y seguir el estado de sus solicitudes. Por su parte, el coordinador del programa evalúa y asigna los proyectos a estudiantes, asegurando que estos se alineen con el currículo académico. Los estudiantes, a su vez, pueden postularse a proyectos de su interés.

Este sistema no solo optimiza el proceso de asignación de proyectos, sino que también garantiza que los estudiantes adquieran experiencia práctica en el desarrollo de soluciones tecnológicas con impacto real en el sector empresarial.

1. Requisitos funcionales

Los requisitos funcionales están descritos en la siguiente hoja de cálculo que muestra las épicas, las historias de usuario y los criterios de aceptación.

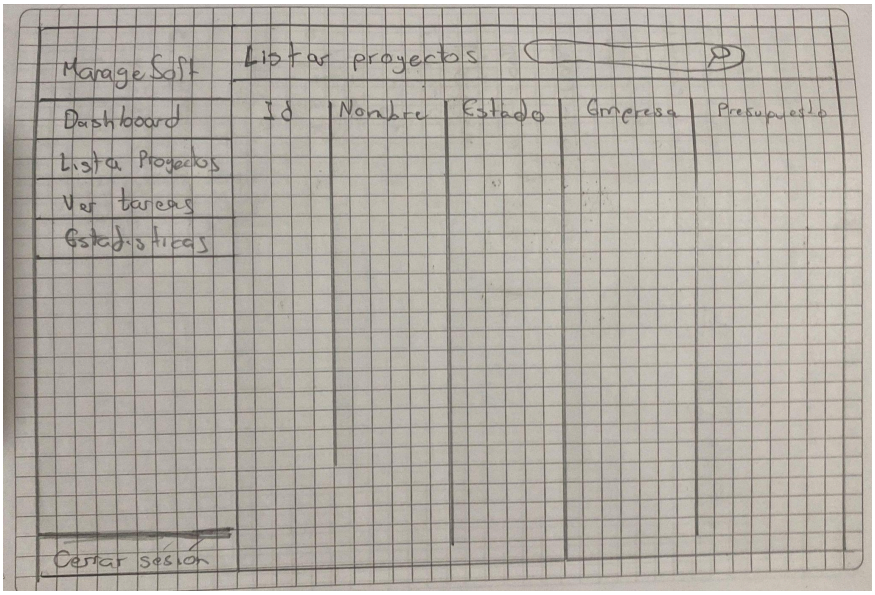
Link: Especificando requisitos funcionales

2. Prototipos y test de usabilidad

2.1. Prototipos

Los prototipos fueron realizados de dos formas: un prototipo de baja fidelidad que consiste en un boceto realizado de forma rápida para crear una primera idea del planteamiento ante la resolución del problema; también un prototipo de alta fidelidad realizado en la herramienta Figma donde se plasmó de mejor forma la estructura, el flujo de la plataforma y la disposición de los elementos de la interfaz (botones, menús, formularios, etc.),

2.2.1. Prototipo de baja fidelidad



2.2.1. Prototipo de alta fidelidad

Link: Figma

2.2. Test de usabilidad

3. Requisitos no funcionales

3.1. Atributos de calidad

3.1.1. Desempeño

Contexto: El Sistema de Gestión de Proyectos Académicos para Empresas es una aplicación de escritorio que permite a empresas, estudiantes y coordinadores gestionar y dar seguimiento a proyectos en tiempo real. A medida que crece la cantidad de usuarios y proyectos activos, es crucial garantizar que el sistema continúe respondiendo de manera eficiente sin afectar la experiencia del usuario.

Estímulo: Se produce un aumento significativo en el número de usuarios simultáneos debido a la publicación masiva de proyectos por parte de diversas empresas. Como resultado, los tiempos de respuesta del sistema se ven afectados, ralentizando la ejecución de procesos y la consulta de información en la base de datos.

Respuesta: El sistema debe optar por un enfoque basado en optimización del rendimiento mediante técnicas como el almacenamiento en caché, optimización de consultas en base de datos relacional (SQL) y una eficiente gestión de recursos de la aplicación de escritorio. Además, de emplear una estrategia de monitoreo del rendimiento con herramientas especializadas para detectar cuellos de botella optimizando el consumo de memoria y procesamiento.

Medición de Calidad: El criterio para la evaluación de calidad está dado en el tiempo de respuesta de la aplicación bajo carga, asegurando que las operaciones críticas se ejecuten en menos de 2 segundos para, al menos, el 90% de las solicitudes. Además, de analizar el consumo de recursos del sistema y la eficiencia en la gestión de la base de datos.

Resultado Esperado: La aplicación debe mantener un tiempo de respuesta óptimo incluso en momentos de alta demanda, asegurando una experiencia

de usuario fluida y eficiente sin afectar la persistencia de los datos o inducir a errores en los demás módulos del sistema.

3.1.2. Escalabilidad

Contexto: Inicialmente, la aplicación está diseñada para ser utilizada exclusivamente por estudiantes de una universidad. Sin embargo, el sistema empezó a ser adoptado por diferentes universidades del país, generando un incremento significativo en la cantidad de usuarios y en el volumen de datos manejados por el sistema.

Estímulo: El crecimiento del software implica un aumento sustancial en la cantidad de estudiantes y empresas que utilizarán la aplicación, lo que representa un desafío en términos de almacenamiento de datos y procesamiento eficiente de la información.

Respuesta: Para garantizar la escalabilidad, se opta por implementar una base de datos con escalabilidad vertical, permitiendo el almacenamiento eficiente de grandes volúmenes de información y la gestión flexible de datos estructurados y no estructurados. Además, para evitar comprometer el rendimiento se decide diseñar la arquitectura del sistema de manera modular.

Medición de Calidad: La capacidad de la aplicación para manejar el crecimiento del número de usuarios se evaluará mediante pruebas de volumen de datos. La base de datos deberá poder almacenar y gestionar de manera eficiente al menos 3000 registros de proyectos y usuarios sin afectar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) críticas.

Resultado Esperado: La aplicación podrá adaptarse a la expansión nacional sin comprometer su estabilidad ni su rendimiento, permitiendo su adopción progresiva en diferentes instituciones educativas.

3.1.3. Modificabilidad

Contexto: La aplicación debe permitir la incorporación de nuevas funcionalidades sin afectar la estabilidad ni la estructura de los módulos existentes. Es necesario que el sistema sea flexible para soportar mejoras y evoluciones a lo largo del tiempo.

Estímulo: La universidad solicita la integración de una nueva función que permita la evaluación de los proyectos académicos por parte de empresas, añadiendo un sistema de retroalimentación y calificaciones. Esto implica

cambios en la interfaz de usuario, la base de datos (SQL) y la lógica de negocio.

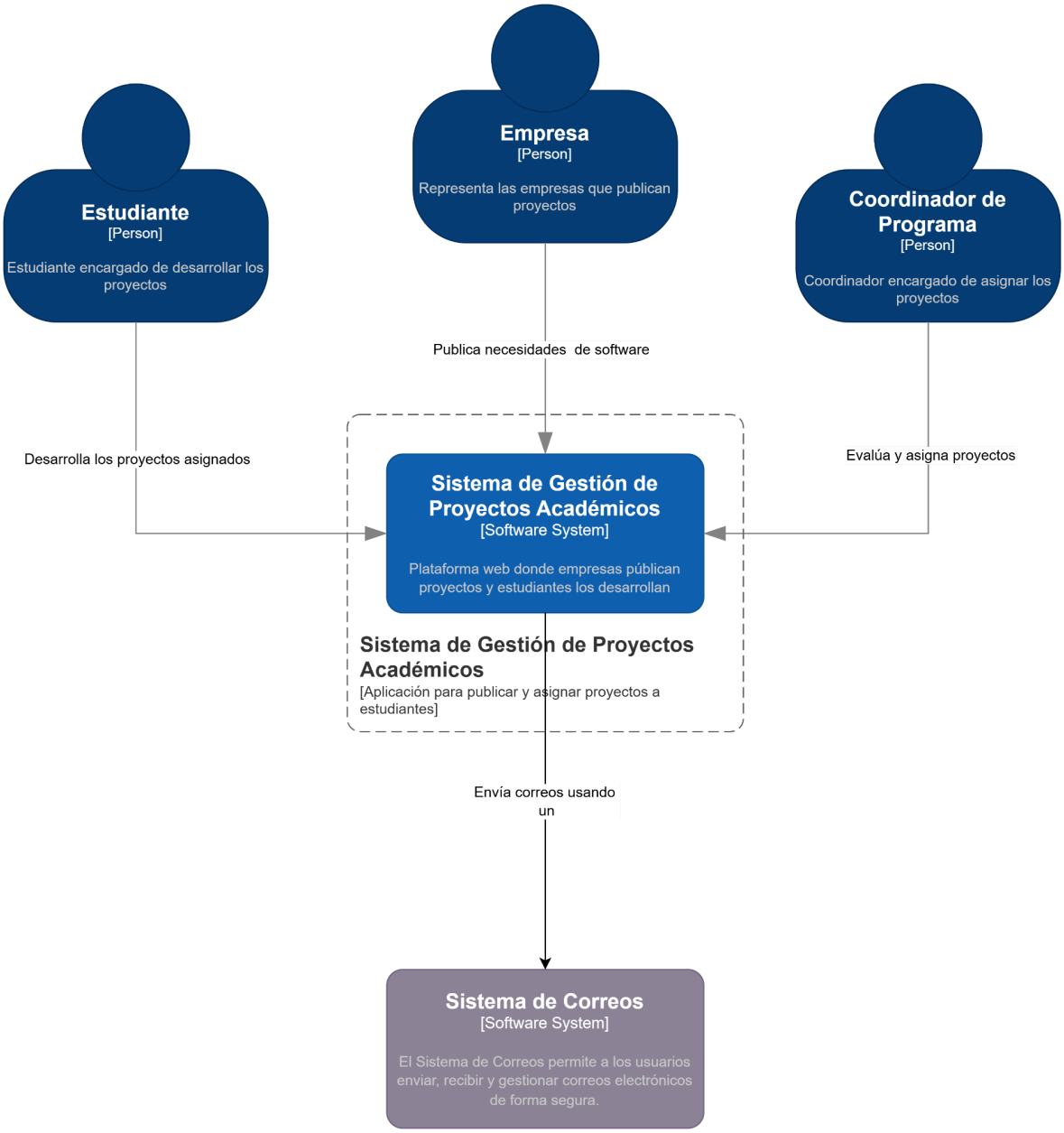
Respuesta: Se diseñará un nuevo módulo desacoplado que maneje la evaluación y retroalimentación de proyectos. Se seguirá una arquitectura modular basada en principios SOLID para garantizar que la nueva funcionalidad pueda integrarse sin afectar los módulos existentes. La base de datos no relacional permitirá almacenar y recuperar evaluaciones de manera flexible sin alterar la estructura de datos original.

Medición de Calidad: Se medirá el impacto del cambio en la base de código existente, asegurando que menos del 10% de los archivos actuales necesiten modificaciones. Además, las pruebas unitarias deben continuar pasando sin errores tras la incorporación del nuevo módulo.

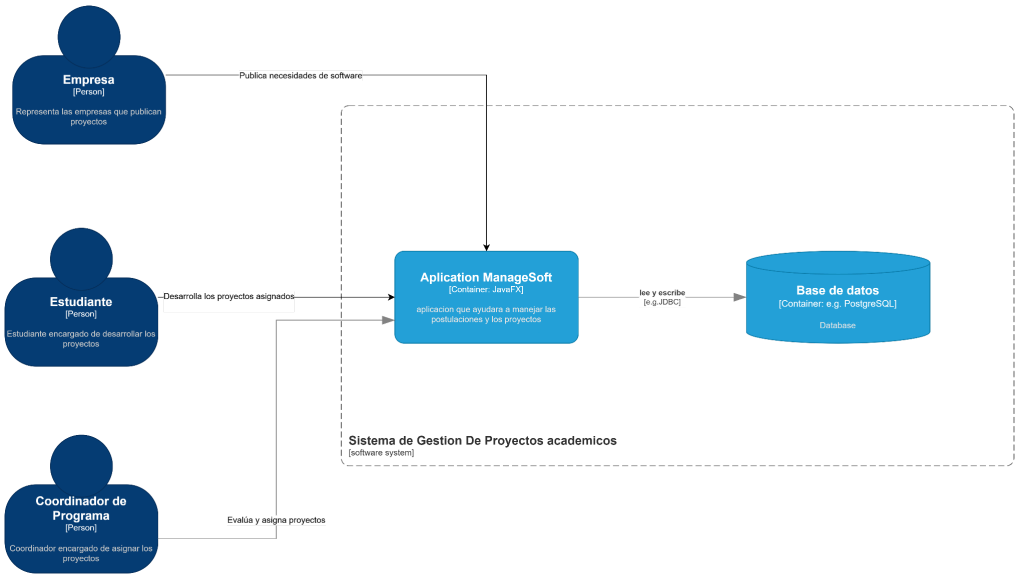
Resultado Esperado: La nueva funcionalidad se integrará sin afectar la operación actual de la aplicación, permitiendo futuras modificaciones de manera sencilla y eficiente.

4. Arquitectura del sistema (Modelos C4)

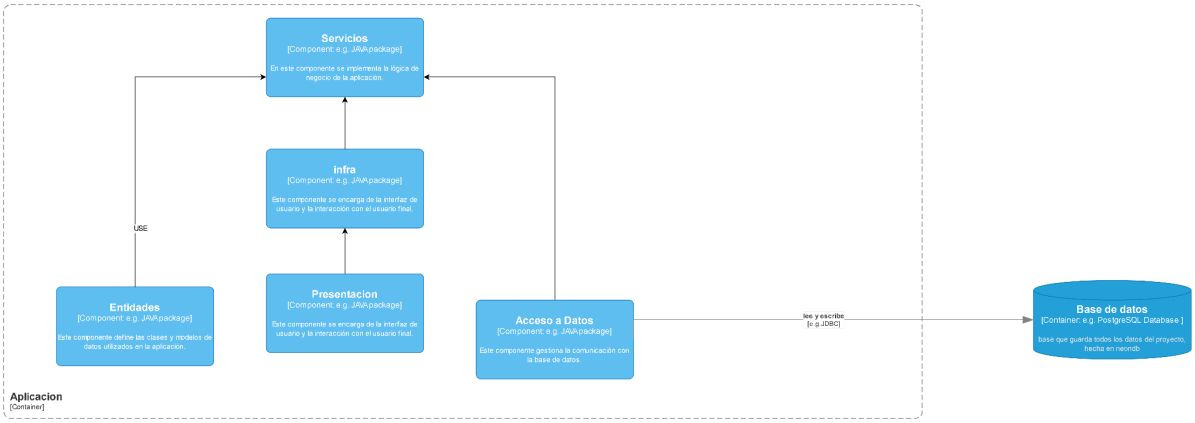
4.1. Diagrama de Contexto



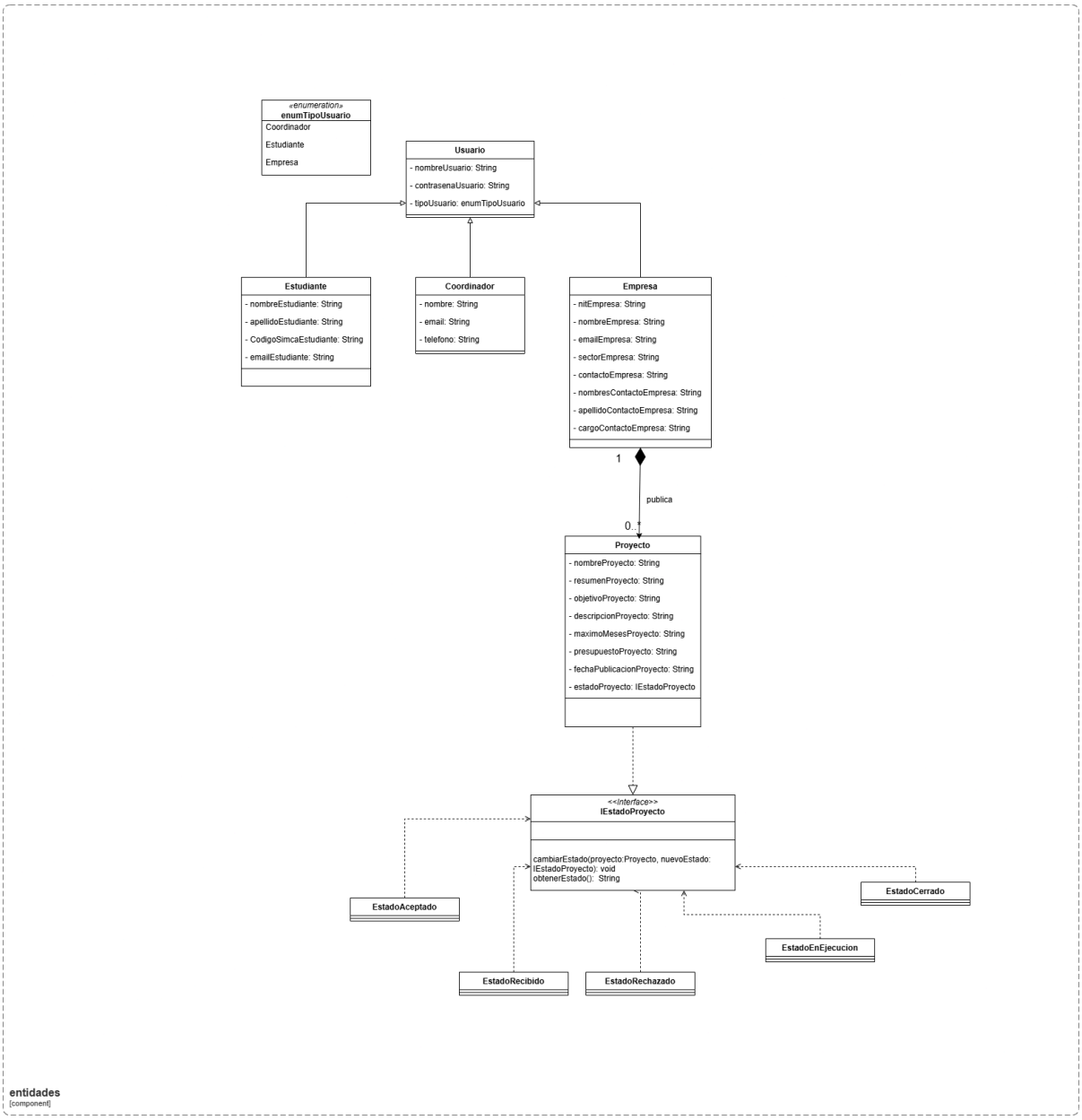
4.2. Diagrama de Contenedores



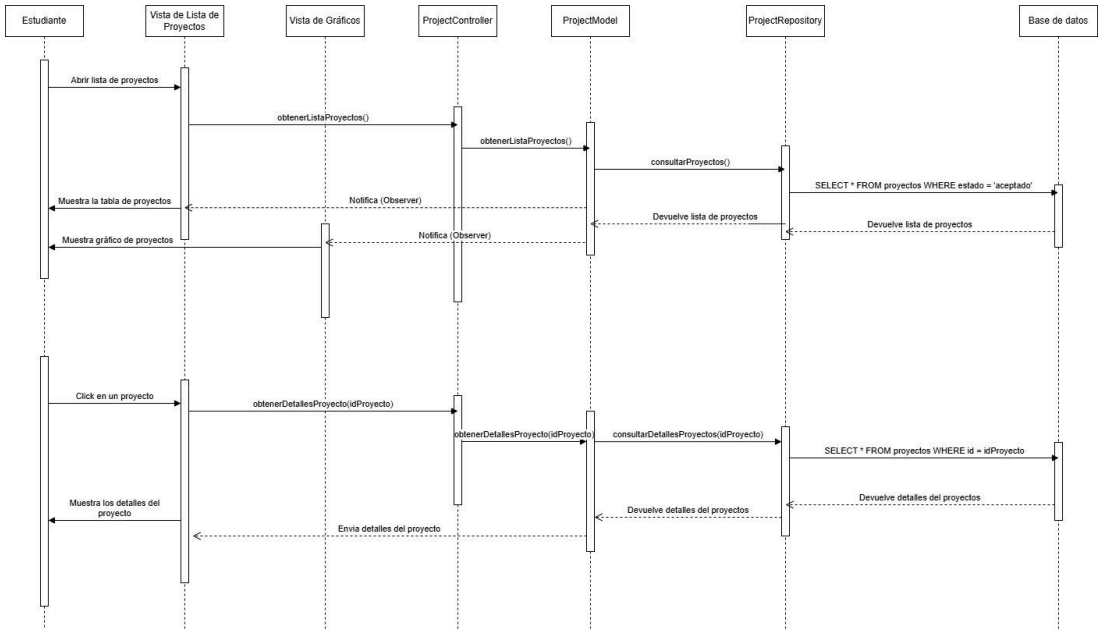
4.3. Diagrama de Componentes



4.4. Diagrama de Clases



5. Diagrama de secuencias



6. Decisiones de arquitectura

6.1 ¿Por qué escogimos trabajar con Java?:

Optamos por Java como lenguaje de desarrollo debido a su enfoque orientado a objetos, lo que nos permite estructurar nuestro código de manera organizada y reutilizable a través de clases y objetos. Además, su robusto sistema de modularización facilita la división del proyecto en componentes independientes, mejorando la mantenibilidad y escalabilidad del software. Java también cuenta con una amplia comunidad de desarrolladores y una extensa biblioteca de herramientas que optimizan el proceso de desarrollo, permitiéndonos implementar soluciones eficientes y seguras.

6.2 ¿Por qué escogimos JavaFx?:

Elegimos JavaFX como entorno para el desarrollo de la interfaz gráfica de usuario debido a sus amplias capacidades y su diseño moderno, que ofrece una apariencia más atractiva y agradable en comparación con Java Swing. JavaFX permite la creación de interfaces dinámicas y visualmente enriquecidas mediante el uso de CSS, animaciones y efectos gráficos avanzados, lo que mejora la experiencia del usuario.

Si bien esta elección implicó una curva de aprendizaje más pronunciada y un mayor tiempo de estudio en comparación con otras alternativas, consideramos que los beneficios a largo plazo justifican el esfuerzo adicional. La flexibilidad de JavaFX, su compatibilidad con arquitecturas como MVC y

su integración con herramientas como Scene Builder nos han permitido desarrollar una interfaz más intuitiva y funcional, optimizando la usabilidad del sistema.

6.3 ¿Por qué lo hicimos a través de capas MVC?:

Optamos por la arquitectura en capas debido a su accesibilidad y facilidad de implementación, ya que permite una modularización clara y estructurada del proyecto. Este patrón organiza el código en diferentes niveles, asignando responsabilidades específicas a cada capa, lo que ayuda a mantener una separación lógica entre la presentación, la lógica de negocio y el acceso a datos.

Al distribuir adecuadamente las funcionalidades, se evita la acumulación excesiva de código en un solo archivo, lo que reduce la complejidad y facilita el mantenimiento del software. Además, esta arquitectura mejora la escalabilidad del sistema, ya que permite realizar modificaciones en una capa sin afectar directamente a las demás, promoviendo así un desarrollo más flexible y ordenado.

6.4 ¿Por qué escogimos NeonDB/PostgreSQL?:

Optamos por **NeonDB/PostgreSQL** como nuestra solución de base de datos debido a su combinación de almacenamiento en la nube con las ventajas de PostgreSQL como sistema de gestión relacional. **NeonDB** nos permite trabajar de manera remota y colaborativa, facilitando el acceso y la sincronización de la base de datos entre todos los miembros del equipo sin necesidad de configuraciones locales complejas. Esto mejora la eficiencia del desarrollo y garantiza una administración centralizada de los datos en tiempo real.

Por otro lado, **PostgreSQL** es una base de datos **open-source**, lo que elimina costos de licencias y ofrece una solución robusta, escalable y segura para nuestro sistema. Su arquitectura bien estructurada se adapta perfectamente a nuestro enfoque en **capas**, permitiendo un manejo eficiente de la información y asegurando una separación clara entre las distintas partes del sistema. Además, su soporte para **transacciones ACID**, consultas SQL avanzadas y extensibilidad mediante índices y tipos de datos adicionales lo convierten en una opción ideal para garantizar la integridad, rendimiento y flexibilidad de nuestra aplicación.

7. **URL del repositorio en GitHub**

   Link: https://github.com/StreetRogue/ManageSoft

8. **URL del video**

   Link: https://www.youtube.com/watch?v=3FVVOjaK8HQ

Referencias

Anaya, R. (2006). Una visión de la enseñanza de la Ingeniería de Software como apoyo al mejoramiento de las empresas de software. *REVISTA Universidad EAFIT*, *42*(141).

CASTAÑO, J. F. (2021). Métricas en la evaluación de la calidad del software: una revisión conceptual. *Computer and Electronic Sciences: Theory and Applications*, *2*(2). https://doi.org/10.17981/cesta.02.02.2021.03

Gordón Graell, R. D. (2023). INGENIERÍA DE SOFTWARE: *Revista FAECO Sapiens*, *6*(2). https://doi.org/10.48204/j.faeco.v6n2.a4014

Krishna Madasu, V., Venkata Swamy Naidu Venna, T., & Eltaeib, T. (2015). SOLID Principles in Software Architecture and Introduction to RESM Concept in OOP. *Journal of Multidisciplinary Engineering Science and Technology (JMEST)*, *2*(2).

Morales Vargas, A., & Codina, L. (2019). Atributos de calidad web para repositorios de datos de investigación en universidades. *Hipertext.Net*, *19*. https://doi.org/10.31009/hipertext.net.2019.i19.04