Automatización para la Gestión Integral del Centro de Excelencia (CoE) en Digital Transformation and Agile

Franklin Garcés Usma

Informe de práctica para optar al título de Ingeniero de Sistemas

Tutor
Deisy Loaiza Berrío, ingeniera de sistemas

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín, Antioquia, Colombia
2024

Cita	(Garcés Usma, 2024)
Referencia **Estilo APA 7 (2020)**	Garcés Usma F. (2024). *Automatización para la Gestión Integral del Centro de Excelencia (CoE) en Digital Transformation and Agile* [Trabajo de grado profesional]. Universidad de Antioquia, Medellín, Colombia.

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Centro de Documentación de Ingeniería (CENDOI)

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes.

Decano/Director: Julio César Saldarriaga.

Jefe departamento: Diego José Botia Valderrama.

Tabla de contenido

Lista de figuras

Resumen

El Centro de Excelencia en Transformación Digital y Ágil enfrentaba desafíos críticos en la gestión eficiente de recursos humanos, operaciones financieras y solicitudes internas, lo que impactó negativamente en su eficiencia operativa. Este trabajo abordó dicha problemática mediante el desarrollo de un software personalizado utilizando ReactJS para el front-end y Spring Boot para el back-end. Aunque el front-end no pudo completarse debido a limitaciones de recursos, se implementó con éxito un sólido back-end siguiendo buenas prácticas y se desarrolló un sistema escalable con Java y Spring Boot. Este sistema fue integrado con PostgreSQL mediante Hibernate y Spring Security para la autenticación. Además, se aplicó una arquitectura orientada a servicios, separando responsabilidades en controladores, servicios y repositorios. Se implementó Logback para el registro de eventos, JUnit para pruebas unitarias y Gradle para la automatización. Además, se siguieron principios SOLID y se aplicaron patrones como la inyección de dependencias. Se generaron peticiones REST para gestionar colaboradores, solicitudes internas y transacciones financieras. También, se diseñaron diagramas de casos de uso, clases, entidad-relación y arquitectura. Esta solución sentó bases tecnológicas robustas para impulsar la transformación digital y la agilidad empresarial del CoE, optimizando procesos internos y fortaleciendo su posición como centro de innovación y eficiencia en el mercado.

Palabras clave: Transformación digital, Agilidad empresarial, Centro de Excelencia, Gestión de recursos humanos, Operaciones financieras, Solicitudes internas, Software personalizado, ReactJS, Spring Boot, Java, PostgreSQL, Arquitectura de servicios, API RESTful, Pruebas unitarias, Principios SOLID, Inyección de dependencias, Optimización de procesos, Innovación, Eficiencia.

Abstract

The Center of Excellence in Digital Transformation and Agile was facing critical challenges in efficiently managing human resources, financial operations, and internal requests, negatively impacting its operational efficiency. This work addressed this issue by developing custom software using ReactJS for the front-end and Spring Boot for the back-end. Although the front-end could not be completed due to resource limitations, a solid back-end was successfully implemented following best practices, and a scalable system was developed using Java and Spring Boot. This system was integrated with PostgreSQL using Hibernate and Spring Security for authentication. Additionally, a service-oriented architecture was applied, separating responsibilities into controllers, services, and repositories. Logback was implemented for event logging, JUnit for unit testing, and Gradle for automation. Furthermore, SOLID principles were followed, and patterns such as dependency injection were applied. REST requests were generated to manage collaborators, internal requests, and financial transactions. Use case, class, entity-relationship, and architecture diagrams were also designed. This solution laid robust technological foundations to drive digital transformation and business agility for the CoE, optimizing internal processes and strengthening its position as a center of innovation and efficiency in the market.

*Keywords***:** Digital transformation, Business agility, Center of Excellence, Human resources management, Financial operations, Internal requests, Custom software, ReactJS, Spring Boot, Java, PostgreSQL, Service architecture, RESTful API, Unit tests, documentation, SOLID principles, Dependency injection, Process optimization, Innovation, Efficiency.

Introducción

Este proyecto de prácticas académicas tenía como objetivo proporcionar a un estudiante de ingeniería de sistemas la oportunidad de contribuir activamente al desarrollo de un software personalizado para el Centro de Excelencia (CoE) en Transformación Digital y Ágil. El CoE es una entidad especializada dentro de la organización que lidera y promueve la adopción de prácticas y tecnologías innovadoras en la transformación digital y la agilidad empresarial. Es decir, el CoE es un lugar centralizado donde se busca formular mejores prácticas basándose en conocimientos y en la experiencia de la empresa (Claire, 2011). La aplicación se enfocaba en la administración integral de colaboradores, finanzas y novedades del CoE, utilizando tecnologías modernas como ReactJS para el desarrollo del front-end y Spring Boot para el back-end.

El entorno actual del Centro de Excelencia (CoE) demandaba una solución tecnológica integral que atendiera las necesidades específicas de gestión. La falta de un software dedicado limitaba la eficiencia en la gestión de recursos humanos, las operaciones financieras y la incorporación de novedades relevantes. Esta propuesta de prácticas se centraba en la creación de una aplicación que utilizara ReactJS para el front-end y Spring Boot para el back-end, permitiendo una implementación eficiente y sostenible.

El principal desafío radicaba en la ausencia de una solución tecnológica que se ajustara a las necesidades específicas del CoE. La propuesta de prácticas buscaba superar esta limitación mediante el desarrollo de un software personalizado que integrara las áreas de recursos humanos, finanzas y novedades, utilizando ReactJS para la interfaz de usuario y Spring Boot para la lógica de negocio y acceso a datos.

Planteamiento del problema

El Centro de Excelencia (CoE) en Transformación Digital y Ágil, enfrentó desafíos críticos en la gestión integral de sus recursos humanos, operaciones financieras y tramitación de solicitudes internas. La carencia de un sistema dedicado para estas funciones impactó negativamente en su eficiencia operativa y capacidad para alcanzar sus objetivos estratégicos.

1. Gestión de Recursos Humanos:

La administración de horarios laborales, el registro de horas trabajadas, la asignación de roles y responsabilidades, así como la gestión de salarios y cargos, eran tareas fundamentales dentro del CoE. Sin embargo, la falta de un sistema centralizado para estas funciones resultó en procesos manuales propensos a errores y pérdida de eficiencia.

2. Operaciones Financieras:

El control de gastos relacionados con recursos humanos, el seguimiento de ingresos y egresos, y la generación de informes financieros son cruciales para la salud financiera del CoE. La ausencia de una solución tecnológica dedicada dificultó la precisión y transparencia en estas operaciones, lo que pudo afectar la toma de decisiones estratégicas.

3. Gestión de Novedades y Solicitudes Internas:

La tramitación y seguimiento de solicitudes internas como PQRS, vacaciones, incapacidades y otras novedades del personal son procesos críticos que requerían una atención diligente. La falta de un sistema integrado dificultó la gestión eficiente de estas solicitudes, lo que pudo afectar la satisfacción y productividad de los colaboradores.

4. Impactos Negativos:

La ineficiencia operativa derivada de la gestión manual de recursos humanos y operaciones financieras, junto con el riesgo de errores y la falta de transparencia, comprometieron la capacidad del CoE para cumplir con sus objetivos estratégicos y fortalecer su posición como un centro de innovación y eficiencia en el mercado.

Solución Propuesta:

El desarrollo e implementación de un software personalizado que integrara la gestión de recursos humanos, operaciones financieras y tramitación de solicitudes internas, permitió al CoE optimizar sus procesos internos, mejorar la eficiencia operativa y fortalecer su competitividad en el mercado. Esta solución proporcionó una plataforma centralizada para la administración eficiente de recursos y la toma de decisiones estratégicas basadas en datos confiables y actualizados.

Justificación

Este estudio surgió como respuesta a la necesidad de mejorar la eficiencia operativa y la gestión integral del Centro de Excelencia (CoE) en Transformación Digital y Ágil. La selección de este tema se fundamentó en la importancia estratégica que tenía para el CoE contar con un sistema dedicado que optimizara la administración de recursos humanos, operaciones financieras y tramitación de solicitudes internas.

Entender el tema fue fundamental para abordar los desafíos y limitaciones que enfrentaba el CoE en aquel momento. La gestión eficiente de recursos humanos, finanzas y solicitudes internas era crucial para su funcionamiento y cumplimiento de sus objetivos estratégicos. La selección de este tema se basó en la necesidad de implementar soluciones tecnológicas que mejoraran los procesos internos del CoE y fortalecieran su posición como un centro de innovación y eficiencia.

Este trabajo contribuyó al campo de la gestión empresarial y la tecnología de la información al proporcionar un análisis detallado de los desafíos y oportunidades en la gestión de recursos humanos, operaciones financieras y tramitación de solicitudes internas en un contexto empresarial específico. Además, la propuesta de desarrollo e implementación de un software personalizado brindó una solución práctica y aplicable a otros centros de excelencia y organizaciones similares que enfrentaban desafíos similares en la gestión de sus recursos y operaciones internas. En este sentido, el texto ofreció recomendaciones y buenas prácticas para la implementación exitosa de sistemas de gestión integrada en entornos empresariales dinámicos y orientados a la innovación.

Objetivos

Objetivo general

Desarrollar un software personalizado utilizando ReactJS y Spring Boot para gestionar integralmente el Centro de Excelencia (CoE) en Transformación Digital y Ágil.

Objetivos específicos

- Analizar las necesidades del CoE en la gestión de colaboradores, finanzas y novedades.
- Diseñar una interfaz de usuario intuitiva y eficiente utilizando ReactJS.
- Desarrollar la lógica de negocio y acceso a datos siguiendo las mejores prácticas con Spring Boot.
- Implementar el software en un entorno de pruebas y evaluar su rendimiento.
- Integrar adecuadamente los aplicativos para que funcionen de una manera eficiente.
- Realizar las pruebas de funcionalidad y aceptación pertinentes para garantizar que se cumplen con los diferentes criterios.

Marco teórico

Según Claire (2011), Un centro de excelencia de fabricación, o CoE (del ingés Centre of Excellence) es un lugar centralizado donde se formulan las mejores prácticas basándose en conocimientos y en datos de la experiencia de una empresa. Se trata de un entorno para probar nuevos procesos emergentes y una herramienta para descubrir, supervisar y difundir mejoras de fabricación.

Figura 1
Centro de excelencia (COE).

Nota. Fuente: https://www.spheregen.com/determining-your-center-of-excellence-coe-strategy. (spheregen, 2024).

Con el fin de otorgar una solución moderna y confiable, se examinaron los principios y metodologías de desarrollo de software, como el ciclo de vida del desarrollo de software donde el software es la clave del éxito de muchas empresas y negocios, ya que sin él sería casi imposible el mantenimiento y crecimiento de los mismos. Lo que diferencia una compañía de otra es la suficiencia, exactitud y oportunidad de la información dada por el software (Maida & Paziencia, 2015); modelos de proceso de desarrollo (como el modelo en cascada, modelo en espiral, y en particular Scrum); y prácticas de ingeniería de software como la gestión de requisitos, diseño de software, pruebas y mantenimiento. Se enfatizo en el uso de metodologías ágiles como Scrum, que

promueve la entrega incremental, la colaboración cercana entre el equipo de desarrollo y el cliente, y la adaptabilidad a los cambios en los requisitos del proyecto.

Para el desarrollo de aplicaciones web, se exploraron los conceptos esenciales de arquitectura web, protocolos de comunicación cliente-servidor (como HTTP), diseño de interfaces de usuario (UI/UX), y las tecnologías y herramientas comunes utilizadas en el desarrollo web, como JavaScript que corresponde a un lenguaje de programación muy utilizado dentro de las empresas de desarrollo de software; HTML que es el componente más básico de la Web, define el significado y la estructura del contenido web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (CSS) (HTML: Lenguaje de etiquetas de hipertexto); y frameworks de desarrollo web como ReactJS, que es una biblioteca de JavaScript de código abierto que se utiliza para crear interfaces de usuario (UI) interactivas y de una sola página. Además, actualiza y renderiza los componentes adecuados cuando la información cambia, gracias al diseño de vistas simples para cada estado de la aplicación (Marín, 2018).

Figura 2
HTML, CSS Y JAVASCRIPT.

Nota. Fuente: https://www.p92.hu/techdetails/html-css-and-javascript. (P92, 2024).

En cuanto a ReactJS, se examinaron sus características principales, como el enfoque en componentes reutilizables, el estado del componente, el enlace de datos unidireccional, y su papel en la creación de interfaces de usuario dinámicas y responsivas en aplicaciones web.

Figura 3
Java API REST Microservicios.

Nota. Fuente https://escuela.it/cursos/curso-de-microservicios-con-java-y-spring-boot (Escuela IT, 2024).

Para Spring Boot, se abordaron su arquitectura, principios de diseño, características principales como la configuración automática, el desarrollo basado en convenciones y la facilidad de creación de aplicaciones web y servicios RESTful. Además, se analizó su integración con otras tecnologías y frameworks en el ecosistema de Java.

Se destaca la importancia de las buenas prácticas en el desarrollo de software, como el Clean Code, que promueve la escritura de código limpio, legible y mantenible. Se revisaron los principios de Clean Code, como la claridad, la simplicidad, la expresividad y la modularidad del código.

Figura 4
Principios SOLID.

Nota. Fuente https://kata-software.com/es/publicaciones/principios-solid-en-programacion (KATA Software, 2020).

Además, se abordaron los principios SOLID de diseño de software, que incluyen según Rodriguez (2022):

- Principio de Responsabilidad Única (Single Responsibility Principle):Indica que solo un cambio potencial (lógica de base de datos, lógica de registro, etc.) en la especificación del software debería poder afectar la especificación de la clase.

- Principio de Abierto/Cerrado (Open/Closed Principle): Exige que las clases deban estar abiertas a la extensión y cerradas a la modificación.

- Principio de Sustitución de Liskov (Liskov Substitution Principle): Establece que las subclases deben ser sustituibles por sus clases base.

- Principio de Segregación de Interfaces (Interface Segregation Principle): Este principio trata de separar las interfaces, establece que muchas interfaces específicas del cliente son mejores que una interfaz de propósito general.

- Principio de Inversión de Dependencias (Dependency Inversion Principle): Establece que las clases deben depender de interfaces o clases abstractas en lugar de clases y funciones concretas.

Estos cinco principios ayudaron a comprender la necesidad de ciertos patrones de diseño y arquitectura de software en general (Rodriguez, 2022). Además, estos principios guían el diseño de software orientado a objetos y promueven la modularidad, la reutilización y la extensibilidad del código.

Metodología

La metodología propuesta para el desarrollo del software se basó en un enfoque mixto que integre aspectos cualitativos y cuantitativos. A continuación, se describen en detalle las actividades específicas para cada objetivo específico:

1. Análisis de Requerimientos:

En esta fase inicial, se llevó a cabo una revisión exhaustiva de la documentación existente y los procesos actuales del Centro de Excelencia (CoE). Se organizaron entrevistas con los stakeholders clave, como el gerente del CoE y los colaboradores, para identificar sus necesidades y expectativas con respecto al software. La información recopilada se analizó cuidadosamente para definir los requisitos del software de manera clara y completa.

2. Desarrollo del Back-end (Spring Boot):

En esta fase, se diseñó la arquitectura del back-end de manera meticulosa, considerando los requisitos funcionales y no funcionales del software. Se implemento la lógica de negocio y el acceso a datos utilizando Spring Boot, siguiendo las mejores prácticas de desarrollo de software. Se presto especial atención a la eficiencia y la seguridad del back-end, garantizando un rendimiento óptimo y una gestión eficiente de la información.

3. Integración y Pruebas:

Una vez completado el back-end por separado, se procedió a integrar en un entorno de prueba. Se realizaron pruebas exhaustivas, incluyendo pruebas de integración, pruebas unitarias y pruebas funcionales, para garantizar que el software funcione correctamente en su conjunto. Se identificaron y corrigieron los errores encontrados durante esta fase, asegurando la calidad del producto final.

4. Implementación y Capacitación:

Con el software probado y validado, se procedió a su implementación en el entorno operativo del CoE. Se brindó capacitación a los usuarios finales para familiarizarlos con el uso del software, asegurando que puedan aprovechar al máximo todas sus funcionalidades. Se proporciono soporte técnico y se resolvieron dudas durante la fase inicial de implementación para garantizar una transición sin problemas.

5. Evaluación y Seguimiento:

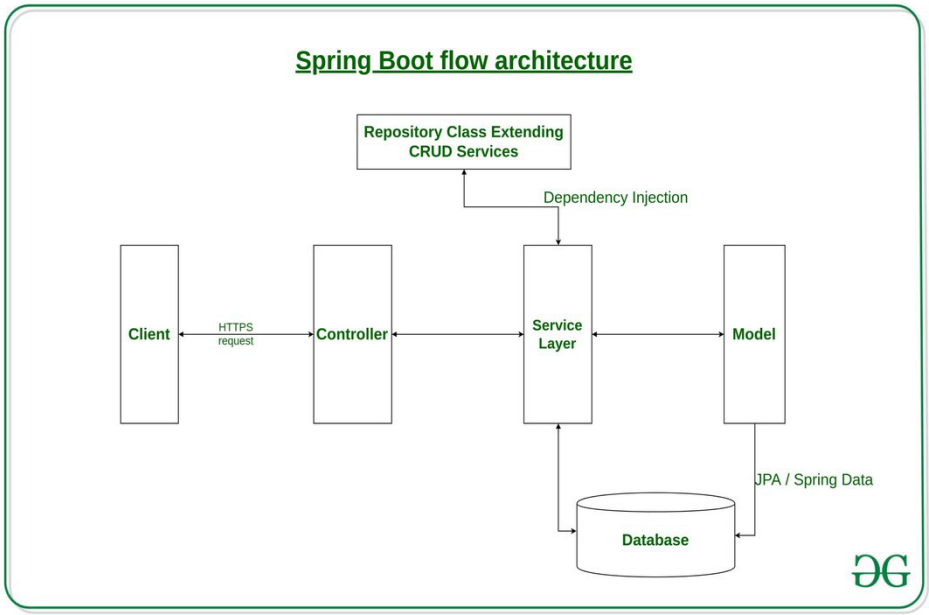
Finalmente, se evaluó el rendimiento del software y la satisfacción de los usuarios mediante encuestas, métricas de uso y retroalimentación directa. Se recopilaron comentarios y sugerencias para la mejora continua del software, y se implementaron las mejoras y correcciones necesarias para mantener la calidad y la satisfacción del usuario a lo largo del tiempo.

Resultados

Los resultados esperados del estudio se centran en comunicar los hallazgos y descubrimientos derivados del desarrollo e implementación del aplicativo para el Centro de Excelencia (CoE) en Transformación Digital y Ágil. El foco principal en el desarrollo fue poner los primeros esfuerzos en el back-end, el cual se formó con una arquitectura orientada a servicios.

El Centro de Excelencia (CoE) en Transformación Digital y Ágil enfrentaba el desafío de gestionar de manera eficiente los datos y operaciones relacionados con sus proyectos e iniciativas. Para resolverlo, se implementó una solución de back-end que sentó una base tecnológica sólida.

Figura 5
Spring Boot architecture.



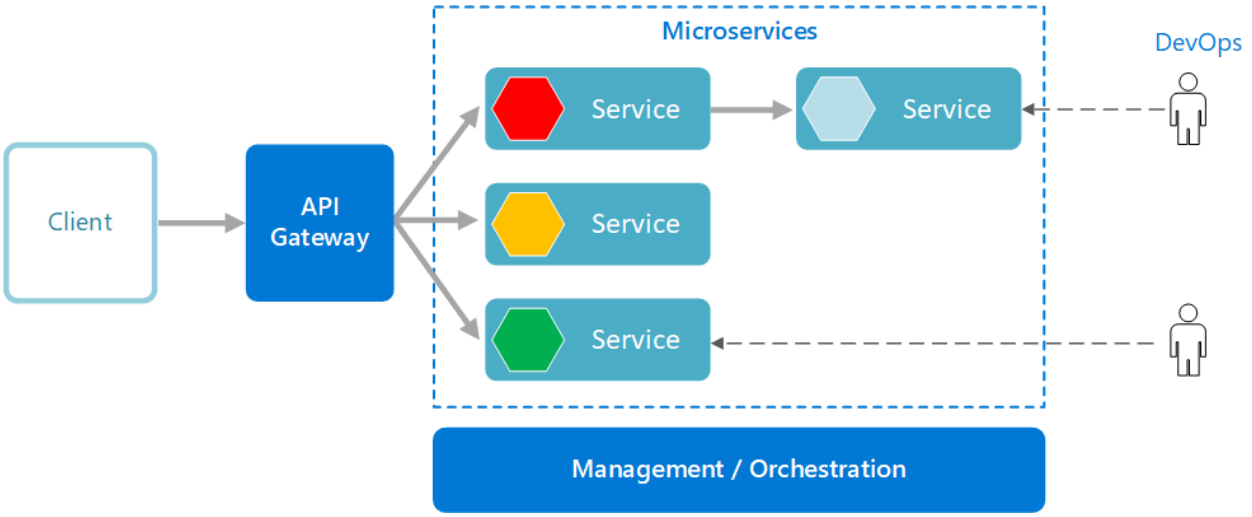
Nota. Fuente http:/www.geeksforgeeks.org

En primer lugar, se enfocaron los esfuerzos en el desarrollo de un sistema robusto y escalable utilizando Java y el framework Spring Boot. Esto permitió gestionar los datos y operaciones del CoE de manera óptima. Además, la integración con la base de datos relacional

PostgreSQL, junto con el mapeo objeto-relacional proporcionado por Hibernate, garantizó un almacenamiento y acceso eficientes a la información.

Figura 6

Diagrama Arquitectura de microservicios.



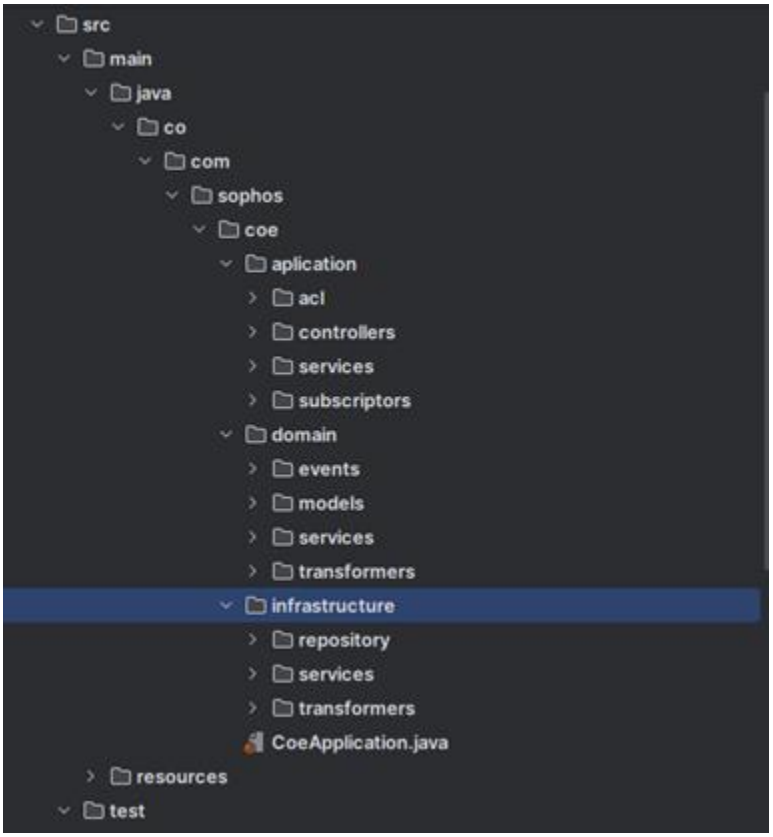
Nota. Fuente https://www.edrawsoft.com/es/article/microservices-architecture-diagram.html

Asimismo, se incorporaron herramientas de registro y monitoreo como Logback, permitiendo un seguimiento detallado de eventos para facilitar la depuración y el mantenimiento del sistema. En el aspecto de pruebas, se aplicaron prácticas sólidas de pruebas unitarias con JUnit, asegurando la calidad y el correcto funcionamiento del código.

Además, para una gestión eficiente del proyecto, se utilizó Gradle como herramienta de automatización de construcción, garantizando un desarrollo y despliegue fluidos. En cuanto a la arquitectura, se siguieron los principios de diseño SOLID y patrones como Inyección de Dependencias, promoviendo un código modular, mantenible y escalable.

En la parte de la arquitectura del proyecto se optó por adoptar la arquitectura hexagonal debido a la necesidad de desacoplar la lógica de negocio de los detalles de implementación y facilitar la prueba y evolución del sistema. Decidimos organizar nuestro código en tres principales capas: aplicación, dominio e infraestructura.

Figura 7
Estructura proyecto arquitectura hexagonal.



Nota. Fuente: imagen propia.

En la capa de aplicación, implementamos adaptadores de controlador de límites (ACL) para interactuar con los puntos de entrada de la aplicación, como controladores Spring MVC. Estos adaptadores traducían las solicitudes y respuestas HTTP a objetos de dominio y viceversa. Además, desarrollamos controladores que gestionaban las solicitudes entrantes y delegaban la lógica de negocio a los servicios de aplicación.

En la capa de dominio, definimos eventos de dominio que representaban cambios significativos en el estado de la aplicación. También creamos modelos que encapsulaban la lógica y el comportamiento relacionados con el dominio, así como servicios de dominio para la lógica de

negocio específica del dominio. Los transformadores se utilizaron para transformar datos entre diferentes representaciones, como la conversión de objetos de dominio en DTOs para su envío a través de una API REST.

Por último, en la capa de infraestructura, creamos adaptadores de repositorio para acceder y persistir datos en la base de datos. Estos adaptadores implementaban las interfaces definidas en la capa de dominio para permitir que la lógica de la aplicación interactuara con el almacenamiento de datos de manera independiente. También desarrollamos servicios de infraestructura para integrar sistemas externos y gestionar aspectos relacionados con la infraestructura.

En general, la implementación de la arquitectura hexagonal nos permitió lograr una clara separación de responsabilidades y facilitar la modularidad y la escalabilidad del sistema. Aunque requirió un esfuerzo adicional en la fase de diseño e implementación, encontramos que los beneficios a largo plazo en términos de mantenibilidad y extensibilidad justificaron la elección de esta arquitectura

Del desarrollo anterior, se ha generado una lista de peticiones REST, las cuales son utilizadas por el sistema para cumplir con los requerimientos establecidos:

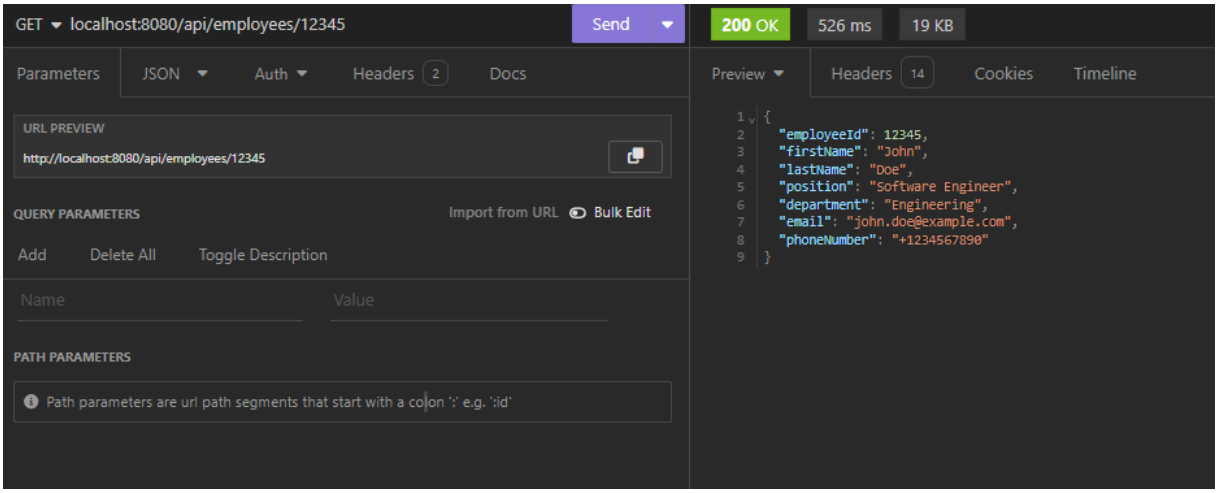
Peticiones GET:

- Obtener la lista de todos los colaboradores: Esta petición permitió obtener una lista de todos los colaboradores registrados en el sistema del Centro de Excelencia (CoE). La respuesta incluyo detalles como el nombre, cargo, departamento, y cualquier otra información relevante de cada colaborador.
- Obtener los detalles de un colaborador específico por su ID: Esta petición permitió obtener los detalles completos de un colaborador específico identificado por su ID único. La respuesta incluyo información detallada como nombre, cargo, departamento, contacto, horario laboral, entre otros.

- Obtener la lista de todas las solicitudes internas: Esta petición permitió obtener una lista de todas las solicitudes internas registradas en el sistema del CoE. La respuesta incluyo detalles como el tipo de solicitud, estado actual, fecha de creación, y cualquier otra información relevante de cada solicitud.
- Obtener los detalles de una solicitud interna específica por su ID: Esta petición permitió obtener los detalles completos de una solicitud interna específica identificada por su ID único. La respuesta incluyo información detallada como el tipo de solicitud, estado actual, fecha de creación, detalles adicionales proporcionados por el solicitante, entre otros.
- Obtener la lista de transacciones financieras: Esta petición permitió obtener una lista de todas las transacciones financieras registradas en el sistema del CoE. La respuesta incluyo detalles como el tipo de transacción, monto, fecha, descripción, y cualquier otra información relevante de cada transacción.
- Obtener los detalles de una transacción financiera específica por su ID: Esta petición permitió obtener los detalles completos de una transacción financiera específica identificada por su ID único. La respuesta incluyo información detallada como el tipo de transacción, monto, fecha, descripción, y cualquier otra información relevante asociada a la transacción.
- Obtener informes financieros: Esta petición permitió generar y obtener informes financieros específicos según ciertos criterios como período de tiempo, tipo de transacción, o departamento. La respuesta incluyo datos resumidos y estadísticas relevantes sobre las transacciones financieras realizadas en el CoE.
- Obtener métricas y estadísticas relevantes: Esta petición permitió obtener métricas y estadísticas relevantes sobre el rendimiento y la eficiencia del CoE en términos de recursos humanos, operaciones financieras, y gestión de solicitudes internas. La respuesta incluyo datos analíticos y gráficos que permitieron evaluar el desempeño del CoE en diferentes áreas.

Figura 8

Prueba petición GET información básica empleados.



Nota. Fuente elaboración propia.

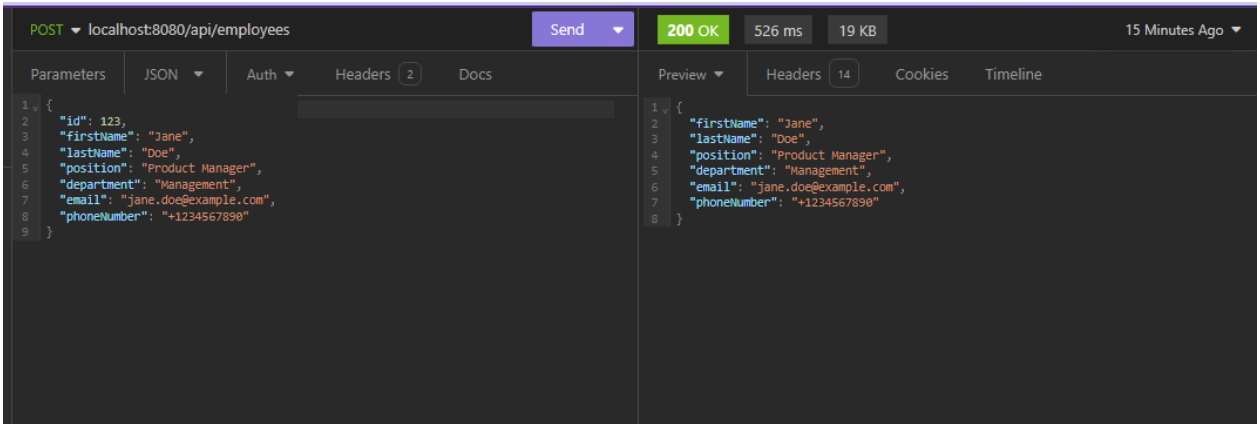
Peticiones POST:

- Crear un nuevo colaborador: Esta petición permitió agregar un nuevo colaborador al sistema del CoE proporcionando los detalles relevantes como nombre, cargo, departamento, contacto, horario laboral, entre otros.
- Crear una nueva solicitud interna: Esta petición permitió crear una nueva solicitud interna en el sistema del CoE proporcionando los detalles relevantes como tipo de solicitud, descripción, fecha de creación, entre otros.
- Crear una nueva transacción financiera: Esta petición permitió registrar una nueva transacción financiera en el sistema del CoE proporcionando los detalles relevantes como tipo de transacción, monto, descripción, fecha, entre otros.
- Generar informes financieros específicos: Esta petición permitió generar informes financieros específicos según ciertos criterios proporcionados como período de tiempo, tipo de transacción, o departamento.
- Actualizar los detalles de un colaborador existente: Esta petición permitió actualizar los detalles de un colaborador existente en el sistema del CoE proporcionando los datos actualizados.

- Actualizar el estado de una solicitud interna: Esta petición permitió actualizar el estado de una solicitud interna existente en el sistema del CoE (por ejemplo, aprobar, rechazar, en proceso) proporcionando el estado actualizado.
- Actualizar los detalles de una transacción financiera existente: Esta petición permitió actualizar los detalles de una transacción financiera existente en el sistema del CoE proporcionando los datos actualizados.
- Eliminar un colaborador: Esta petición permitió eliminar un colaborador existente del sistema del CoE según su ID único.
- Eliminar una solicitud interna: Esta petición permitió eliminar una solicitud interna existente del sistema del CoE según su ID único.
- Eliminar una transacción financiera: Esta petición permitió eliminar una transacción financiera existente del sistema del CoE según su ID único.

Figura 9

Prueba petición POST Creación de empleado.

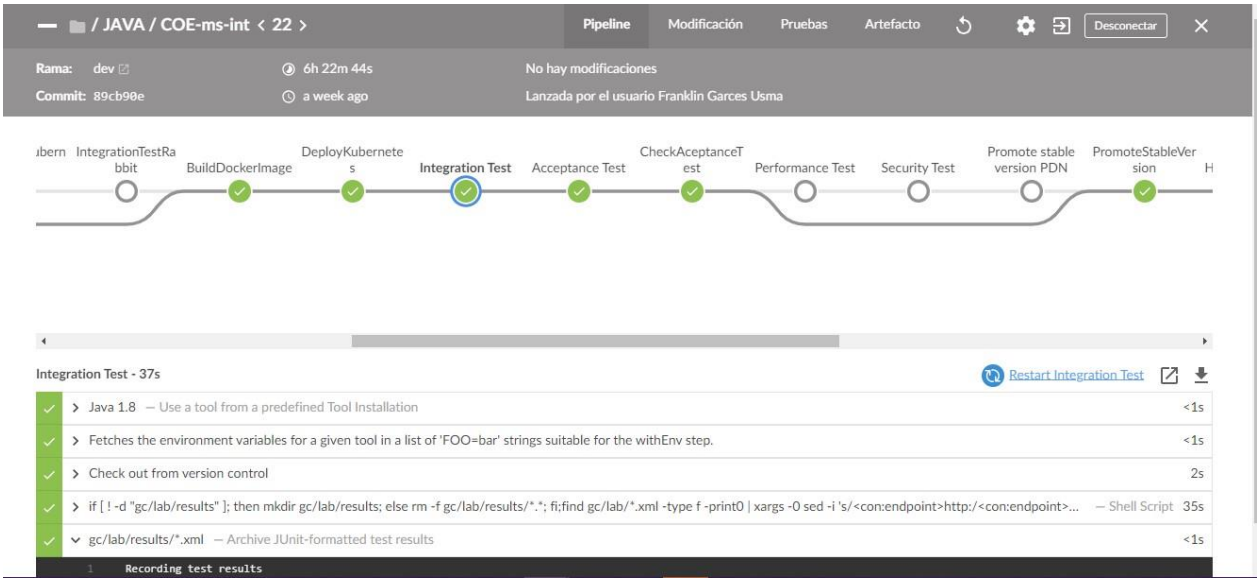


Nota. Fuente elaboración propia.

Luego de probar diferentes funcionalidades del back end, se realizaron pruebas de integración para validar y corroborar que el comportamiento en los ambientes sea adecuado, para esto se utilizaron herramientas de pruebas de integración como JUnit y Mockito para simular solicitudes HTTP, inyectar dependencias simuladas y verificar el comportamiento esperado del sistema. Se prestó especial atención a las pruebas de flujos de datos complejos, transacciones, manejo de errores y condiciones de borde.

Figura 10

Ejecución de pruebas de integración.



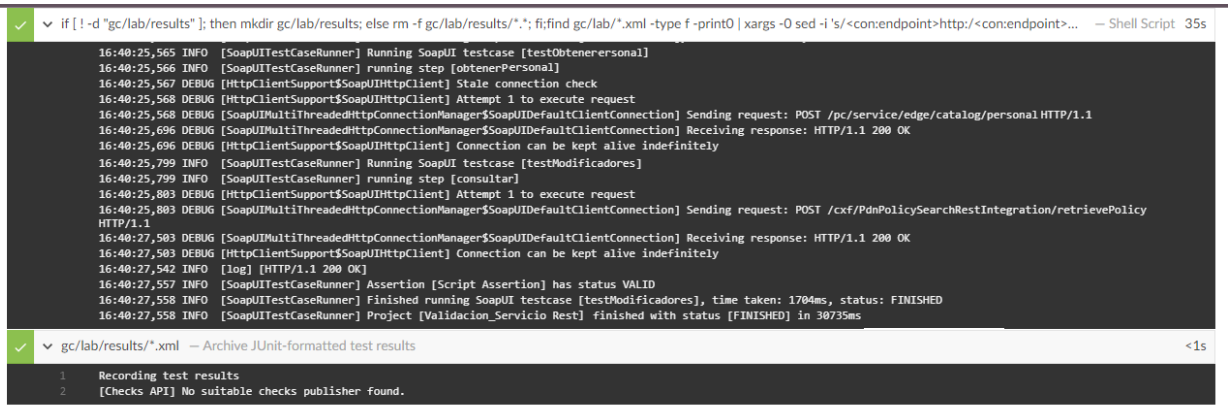
Nota. Fuente elaboración propia.

Durante el proceso de pruebas, se identificaron y corrigieron varios problemas de integración, como inconsistencias en el mapeo de datos, errores de lógica de negocio y problemas de rendimiento. Estas correcciones se aplicaron de manera iterativa, y las pruebas se repitieron hasta que se alcanzó un nivel de confianza satisfactorio.

Además, se realizaron pruebas de integración con servicios y sistemas externos simulados, como servicios de autenticación, servicios de correo electrónico y otras dependencias relevantes para el funcionamiento completo del back-end.

Figura 11

Resultados pruebas integración.



Nota. Fuente elaboración propia.

Si bien estaba contemplado inicialmente desarrollar el front-end utilizando ReactJS, lamentablemente no se logró completar esta tarea debido a limitaciones en los recursos humanos y el personal disponible. La prioridad se centró en asegurar una implementación sólida y robusta del back-end, siguiendo las mejores prácticas y tecnologías más adecuadas. No obstante, se reconoce la importancia crítica de contar con una interfaz de usuario moderna e intuitiva para aprovechar al máximo las capacidades del back-end desarrollado. Por lo tanto, se recomienda continuar con el desarrollo del front-end en ReactJS en una etapa posterior, una vez se cuente con el personal y los recursos necesarios. Esto permitirá brindar una experiencia integral y de alta calidad a los usuarios finales del sistema.

Conclusiones

1. El desarrollo del back-end del sistema para el Centro de Excelencia (CoE) en Transformación Digital y Ágil se llevó a cabo de manera exitosa. Se implementó una solución sólida y escalable utilizando Java y el framework Spring Boot, lo cual permitió una gestión eficiente de los datos y operaciones del CoE. La integración con PostgreSQL y Hibernate, junto con la implementación de Spring Security, aseguró un almacenamiento seguro y un acceso óptimo a la información. Además, esta experiencia me permitió comprender la importancia de seguir las mejores prácticas y estándares de la industria para garantizar la calidad y escalabilidad del software.
2. Durante el desarrollo del back-end, se aplicaron buenas prácticas y estándares de la industria. Se siguieron los principios de diseño SOLID, se implementaron patrones como la inyección de dependencias, y se utilizaron herramientas como Logback y JUnit para mejorar la documentación, el registro de eventos y las pruebas unitarias, respectivamente, garantizando un código modular, mantenible y de alta calidad.
3. El back-end se diseñó con una arquitectura orientada a servicios, separando las responsabilidades en diferentes capas: controladores, servicios y repositorios. Esta separación de concerns facilito la escalabilidad, el mantenimiento y la reutilización del código, además de promover una mejor organización y modularidad del sistema.
4. La implementación de PostgreSQL y el diseño de la base de datos utilizando el modelo Entidad-Relación me proporcionaron una comprensión más profunda de cómo gestionar eficientemente la información en un sistema. Esta experiencia me permitió familiarizarme con las mejores prácticas en el diseño de bases de datos relacionales ya que este modelo permite almacenar y gestionar de manera eficiente la información de colaboradores, solicitudes internas y transacciones financieras del CoE.
5. Si bien se priorizó el desarrollo del back-end debido a limitaciones de recursos, se reconoce la importancia crítica de contar con una interfaz de usuario moderna e intuitiva utilizando ReactJS. Se recomienda continuar con el desarrollo del front-end en una etapa posterior para brindar una experiencia completa y de alta calidad a los usuarios finales del sistema.

En futuros proyectos, priorizaré el equilibrio entre el desarrollo del back-end y el front-end para ofrecer una solución completa y satisfactoria para los usuarios finales.

6. En general, la solución de back-end desarrollada sienta las bases tecnológicas sólidas para impulsar la transformación digital y la agilidad empresarial en el Centro de Excelencia. Al combinar esta solución con el desarrollo futuro del front-end en ReactJS, el CoE contará con un sistema integral y robusto que optimizará sus procesos internos y fortalecerá su posición como un centro de innovación y eficiencia en el mercado. Finalmente quiero resaltar que adquirí los conocimientos sobre cómo la tecnología puede ser un habilitador poderoso para mejorar los procesos internos y fortalecer la posición competitiva de una organización en el mercado.

Referencias

Claire, D. (2011). Centros de excelencia de fabricación ¿Qué son y por qué son necesarios? Nexus Integra. https://nexusintegra.io/es/centros-de-excelencia-CoE-empresas/ (consultado el 14 de noviembre de 2023).

HTML: Lenguaje de etiquetas de hipertexto. (s/f). MDN Web Docs. Recuperado el 20 de marzo de 2024, de https://developer.mozilla.org/es/docs/Web/HTML

Maida, EG, Pacienzia, J. Metodologías de desarrollo de software [en línea]. Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería “Fray Rogelio Bacon”. Universidad Católica Argentina, 2015. Disponible en: http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf

Marín, P. (2018). Desarrollo de un frontend en ReactJS. [Trabajo Final de Grado]. Universitat Jaumei.

Rodriguez, D. S. (2022). Los principios SOLID de programación orientada a objetos explicados en Español sencillo. freecodecamp.org. https://www.freecodecamp.org/espanol/news/los-principios-solid-explicados-en-espanol/