

Desarrollo Progresivo de Proyectos de software desde lo Básico hasta lo Avanzado basado en Soluciones Complejas

Daniel Loaiza Noreña

Informe de práctica para optar al título de Ingeniero de Sistemas

Asesor
Katerine Márceles Villalba

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

Loaiza Noreña, D. (2024). Desarrollo Progresivo de Proyectos de software desde lo Básico hasta lo Avanzado basado en Soluciones Complejas. Universidad de Antioquia, Medellín.

creative commons BY NC SA

Ux Technology, Grupo Reditos, Jose Edisson Castaño Castaño , Gana plus, Invictus:

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia
Sistema de Bibliotecas
Centro de Documentación 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: Danny Alexandro Múnera

El contenido de esta obra corresponde al derecho de expresión de los autores y no compromete el pensamiento institucional de la Universidad de Antioquia ni desata su responsabilidad frente a terceros. Los autores asumen la responsabilidad por los derechos de autor y conexos.

Tabla de contenido

Resumen **6**

Abstract **7**

Introducción **8**

1. PLANTEAMIENTO DEL PROBLEMA **9**

2. JUSTIFICACIÓN **10**

3. OBJETIVOS **11**

4. MARCO REFERENCIAL **12**

4.1. Marco teórico **12**

4.2 Marco conceptual **13**

5. METODOLOGÍA **16**

6. RESULTADOS **19**

7. ANÁLISIS **46**

8. CONCLUSIONES **47**

REFERENCIAS **48**

LISTA DE FIGURAS

Figura 1: Cronograma de actividades **18**

Figura 2: Encarpetado **21**

Figura 3: Microservicios **22**

Figura 4: Ejemplo MS **22**

Figura 5: Asistente **23**

Figura 6: Rest API **24**

Figura 7: Chat **25**

Figura 8: Commits **26**

Figura 9: Archivos prueba **29**

Figura 10: Resultado Pruebas **38**

Figura 11: Cobertura **39**

Figura 12: GLPI 1 **41**

Figura 13: GLPI 2 **42**

Figura 14: GLPI 3 **43**

Figura 15: GLPI 4 **43**

Figura 16: GLPI 5 **44**

Siglas, acrónimos y abreviaturas

APA: Asociación Psicológica Americana (American Psychological Association)

API: Interfaz de Programación de Aplicaciones (Application Programming Interface)

GLPI: Gestor Libre de Inventario de Parque Informático (Gestionnaire Libre de Parc Informatique)

HU: Historias de Usuario (User Stories)

MS: Microservicios (Microservices)

QA: Aseguramiento de la Calidad (Quality Assurance)

REST: Transferencia de Estado Representacional (Representational State Transfer)

APK: Paquete de Aplicación para Android (Android Package Kit)

Resumen

El informe final del proyecto "Desarrollo Progresivo de Proyectos de Software desde lo Básico hasta lo Avanzado basado en Soluciones Complejas" sintetiza los logros alcanzados mediante un enfoque práctico, estructurado y orientado al desarrollo de competencias clave. A lo largo de este proceso, el estudiante practicante participó en proyectos como Gana Plus, Gana, Loti y otras plataformas confidenciales, aplicando soluciones tecnológicas avanzadas y progresivamente desafiantes.

El uso de metodologías ágiles como Scrum, combinado con herramientas modernas como Git, Node.js, Java, Angular, SQL y Android, permitió integrar el aprendizaje técnico con la resolución efectiva de problemas en escenarios reales. Este enfoque fomentó no solo el dominio de habilidades técnicas, sino también el fortalecimiento de competencias transversales como la comunicación y la adaptación al entorno laboral dinámico.

Palabras clave: Android, Angular, Aprendizaje, Git, Java, Node, Pruebas, Scrum, SQL.

Abstract

The final report of the project **"Progressive Development of Software Projects from Basic to Advanced Levels Based on Complex Solutions"** summarizes the achievements reached through a practical, structured approach aimed at developing key competencies. Throughout this process, the student intern participated in projects such as **Gana Plus, Gana, Loti**, and other confidential platforms, applying advanced and increasingly challenging technological solutions.

The use of **agile methodologies** like Scrum, combined with modern tools such as **Git, Node.js, Java, Angular, SQL, and Android**, enabled the integration of technical learning with the effective resolution of real-world problems. This approach not only fostered mastery of technical skills but also strengthened transversal competencies such as communication and adaptability to a dynamic work environment.

Keywords: Android, Angular, Git, Java, Learnig, Node, Scrum, SQL, Testing.

Introducción

En el exigente y competitivo panorama actual del sector tecnológico, la adaptabilidad y la formación metodológica se han convertido en elementos esenciales para la preparación de los futuros profesionales. En este contexto, el proyecto "Desarrollo Progresivo de Proyectos de Software desde lo Básico hasta lo Avanzado basado en Soluciones Complejas" se presenta como una iniciativa innovadora que permite a los practicantes adentrarse en el desarrollo de tecnologías avanzadas, empleando herramientas y metodologías de vanguardia utilizadas por la empresa UX Technology.

El enfoque central del proyecto radica en la construcción de competencias técnicas y profesionales a través de una progresión cuidadosamente diseñada de proyectos con niveles crecientes de complejidad. Este proceso permite que los estudiantes adquieran experiencia práctica y tangible en la creación de soluciones tecnológicas específicas, orientadas a resolver desafíos reales. A lo largo de esta formación, los practicantes no solo desarrollan habilidades técnicas, sino que también fortalecen su capacidad para el análisis crítico, la resolución de problemas complejos y la toma de decisiones informadas en contextos dinámicos.

Entre los proyectos más destacados desarrollados dentro de este marco se encuentran Gana Plus, la plataforma web de Gana, y el sistema Loti, los cuales representan aplicaciones de alto perfil con gran impacto en sus respectivos ámbitos. Asimismo, los participantes tuvieron la oportunidad de trabajar en otras soluciones tecnológicas que, por razones de confidencialidad, no pueden ser divulgadas. Estas experiencias diversificadas ofrecen a los estudiantes una perspectiva integral sobre el desarrollo de software y la innovación tecnológica, acercándolos a la realidad del entorno empresarial moderno.

1. PLANTEAMIENTO DEL PROBLEMA

La empresa identificó la necesidad de incorporar un practicante en el área de desarrollo para asumir funciones específicas enfocadas en la implementación de ideas innovadoras y la ejecución de tareas clave. Este rol fue diseñado para contribuir significativamente al cumplimiento de los objetivos técnicos y operativos, al tiempo que alivianaba la carga de trabajo del equipo, mejoraba los procesos internos y fortalecía la capacidad de respuesta frente a los desafíos cotidianos del negocio.

2. JUSTIFICACIÓN

Con esta propuesta se busca aportar al desarrollo de soluciones innovadoras, proponiendo ideas que contribuyan a la optimización de procesos y la mejora de las herramientas existentes. El enfoque principal radica en apoyar el desarrollo de proyectos clave, demostrando versatilidad en la implementación de diversas tecnologías y adaptándose a las necesidades dinámicas del entorno empresarial. Asimismo, se promueve una colaboración estrecha entre los equipos, con el objetivo de identificar áreas de mejora, implementar soluciones efectivas y garantizar un flujo de trabajo ágil y eficiente, consolidando así un impacto significativo en el éxito y la competitividad de la organización.

3. OBJETIVOS

3.1. Objetivo general

Desarrollar soluciones tecnológicas de calidad mediante la participación activa en proyectos asignados por la empresa UX Technology, aplicando conocimientos teóricos y prácticos en desarrollo de software, para contribuir al cumplimiento de los objetivos empresariales y mejorar las competencias profesionales del estudiante.

3.1.2 Objetivos específicos

- Implementar funcionalidades nuevas y mejorar las existentes en las aplicaciones Gana Plus, la web de Gana y otras aplicaciones, siguiendo las directrices y estándares de la empresa UX Technology, para asegurar la calidad y eficiencia del software desarrollado. Realizar pruebas y depuración de código en los proyectos asignados, utilizando herramientas y técnicas de testing y debugging, para garantizar el correcto funcionamiento y estabilidad de las aplicaciones.
- Realizar pruebas y depuración de código en los proyectos asignados, utilizando herramientas y técnicas de testing y debugging, para garantizar el correcto funcionamiento y estabilidad de las aplicaciones.
- Planificar la entrega oportuna y efectiva de soluciones tecnológicas para contribuir a la finalización exitosa de los proyectos asignados, aplicando metodologías ágiles y principios de diseño.ar los procesos de desarrollo y los resultados obtenidos en cada proyecto, empleando formatos y herramientas de documentación estándar, para facilitar la transferencia de conocimiento y asegurar la continuidad de los proyectos.
- Documentar los procesos de desarrollo y los resultados obtenidos en cada proyecto, empleando formatos y herramientas de documentación estándar, para facilitar la transferencia de conocimiento y asegurar la continuidad de los proyectos.

4. MARCO REFERENCIAL

4.1. Marco teórico

4.1.1 Importancia de la mejora de productos en los desarrollos de software. La mejora continua de productos en el desarrollo de software es crucial porque asegura que el software se mantenga relevante, eficiente y útil para los usuarios. Esto implica corregir errores, agregar nuevas funcionalidades, mejorar el rendimiento y actualizar la seguridad. Además, escuchar y responder a las necesidades y comentarios de los usuarios permite crear un producto que realmente satisface sus expectativas y necesidades cambiantes. La mejora continua también ayuda a las empresas a mantenerse competitivas en el mercado y a adaptarse a las nuevas tecnologías y tendencias (Sommerville, 2011).

4.1.2 Inclusión de prácticas ágiles en equipos de trabajos. Incorporar prácticas ágiles en los equipos de trabajo es clave para aumentar la eficiencia y la adaptabilidad en los proyectos. Metodologías como Scrum y Kanban se enfocan en la colaboración continua, la entrega rápida de productos y la capacidad de ajustarse a cambios. Estas prácticas fomentan una comunicación abierta y constante entre los miembros del equipo y los clientes, permitiendo identificar y resolver problemas rápidamente. Al dividir el trabajo en ciclos cortos y manejables llamados "sprints", los equipos pueden entregar incrementos funcionales del producto de manera continua, garantizando que el proyecto avance de manera constante y se alinee con las necesidades del cliente (Beck et al., 2001; Schwaber & Sutherland, 2020).

4.2 Marco conceptual

4.2.1 Git. Es una herramienta de control de versiones ampliamente utilizada en el desarrollo de software, que permite gestionar los cambios en el código fuente durante el desarrollo colaborativo. Este sistema se implementará para controlar las versiones de los proyectos de la empresa, mejorando así el flujo de trabajo y la eficiencia general (Chacon & Straub, 2014).

4.2.2 Scrum. Es un marco de trabajo ágil utilizado en el desarrollo de software para gestionar proyectos complejos. Este enfoque se basa en iteraciones cortas conocidas como "sprints" y en roles claramente definidos, como el Scrum Master y el Product Owner, con el fin de optimizar la colaboración y la flexibilidad dentro de una empresa. Al utilizar Scrum, las empresas pueden mejorar la coordinación entre los equipos y adaptarse rápidamente a los cambios, asegurando que los proyectos se completen de manera eficiente y satisfactoria (Schwaber & Sutherland, 2017).

4.2.3 Node.js. Es un entorno de ejecución de JavaScript basado en el motor V8 de Chrome, que permite ejecutar código JavaScript fuera de un navegador web, facilitando el desarrollo de aplicaciones de red escalables. En una empresa, Node.js puede ser útil para crear aplicaciones rápidas y eficientes, manejar múltiples solicitudes al mismo tiempo y mejorar la capacidad de respuesta de los servicios web, lo que se traduce en una mejor experiencia para los usuarios y una mayor eficiencia operativa (Cantelon, Harter, & Rajlich, 2014).

4.2.4 Java. Es un lenguaje de programación orientado a objetos diseñado para ser altamente portátil e independiente de la plataforma. Es ampliamente utilizado en el desarrollo de aplicaciones empresariales y en dispositivos móviles. En una empresa, Java facilita la creación de aplicaciones robustas y seguras que pueden ejecutarse en diferentes sistemas operativos, lo que mejora la interoperabilidad y la eficiencia del desarrollo de software (Gosling, Joy, Steele, & Bracha, 2014).

4.2.5 Aprendizaje. El aprendizaje en el contexto de la tecnología implica adquirir conocimientos y habilidades en áreas como el desarrollo de software y otras disciplinas relacionadas, utilizando métodos tanto formales como informales. En una empresa, este tipo de aprendizaje es crucial para mantenerse al día con las nuevas tecnologías y técnicas, lo que puede mejorar la eficiencia, fomentar la innovación y mantener a los empleados actualizados con las últimas tendencias y herramientas (Siemens & Long, 2011).

4.2.6 Android. Es un sistema operativo móvil basado en Linux desarrollado por Google, especialmente diseñado para dispositivos táctiles como smartphones y tablets. En una empresa, Android permite desarrollar aplicaciones que pueden ser utilizadas en una amplia variedad de dispositivos móviles, lo que facilita la creación de soluciones accesibles para los usuarios en diferentes plataformas y mejora la presencia digital de la empresa (Deitel, Deitel, Wald, & Lin, 2016).

4.2.7 SQL (Structured Query Language). Es un lenguaje de programación diseñado para gestionar datos almacenados en sistemas de gestión de bases de datos relacionales (RDBMS). En el contexto empresarial, SQL permite a los usuarios crear, leer, actualizar y eliminar datos de manera eficiente, lo que facilita la gestión de grandes volúmenes de información y la toma de decisiones basada en datos precisos y actualizados (Kline, Kline, & Hunt, 2016).

4.2.8 Pruebas. Las pruebas en el desarrollo de software son actividades organizadas para verificar si un sistema cumple con los requisitos establecidos y para detectar errores antes de que el producto sea lanzado. En el ámbito empresarial, realizar pruebas exhaustivas ayuda a asegurar la calidad del software, reduce el riesgo de fallos en producción y mejora la satisfacción del cliente al ofrecer productos confiables y bien funcionando (Kaner, Falk, & Nguyen, 2017).

4.2.9 Angular. Es un framework de código abierto desarrollado por Google, utilizado principalmente para crear aplicaciones web de una sola página (SPA). Emplea TypeScript como lenguaje de programación y se basa en el patrón de diseño MVC (Modelo-Vista-

Controlador). En una empresa, Angular facilita el desarrollo de aplicaciones web dinámicas y escalables al permitir la reutilización de componentes y servicios que gestionan la lógica de negocio, mejorando así la eficiencia en el desarrollo y mantenimiento de aplicaciones (Angular, s.f, 2024).

5. METODOLOGÍA

El desarrollo del proyecto se realizó basado en el marco de trabajo ágil scrum, alineado con la metodología de desarrollo de software y plataforma empleada por la empresa, esto permitió que se articulara con las actividades previstas que se tienen para lograr el cumplimiento de los objetivos planteados para el desarrollo del producto.

A continuación, se definieron 4 fases que se alinearon con el cumplimiento de los objetivos:

Fase 1: Implementación de funcionalidades nuevas y mejorar las existentes en las aplicaciones Gana Plus, la web de Gana y otras aplicaciones.

Actividad 1.1: Diseñar y desarrollar módulos funcionales en las aplicaciones asignadas.

Actividad 1.2: Integrar nuevas características en las aplicaciones existentes.

Actividad 1.3: Realizar revisiones de código para asegurar la adherencia a los estándares de calidad.

Fase 2: pruebas y depuración de código en los proyectos asignados.

Actividad 2.1: Ejecutar pruebas unitarias y de integración en los módulos desarrollados.

Actividad 2.2: Identificar y corregir errores en el código.

Actividad 2.3: Documentar los resultados de las pruebas y las correcciones realizadas.

Fase 3: Planificación de entregas oportuna de soluciones tecnológicas.

Actividad 3.1: Participar activamente en las reuniones de planificación de sprints, aportando ideas y estimaciones de tiempo para las tareas asignadas.

Actividad 3.2: Colaborar con el equipo en la división y asignación de tareas, asegurando una carga de trabajo equilibrada y alcanzable.

Actividad 3.3: Realizar seguimientos periódicos del progreso de las tareas asignadas, ajustando los planes según sea necesario para cumplir con los plazos establecidos.

Actividad 3.4: Participar en las reuniones de revisión de sprints, presentando los resultados de las tareas completadas y discutiendo las mejoras necesarias.

Fase 4: Documentación de los procesos de desarrollo y los resultados obtenidos en cada proyecto.

Actividad 4.1: Elaborar documentación técnica detallada de los desarrollos realizados.

Actividad 4.2: Crear guías de usuario para las nuevas funcionalidades implementadas.

Actividad 4.3: Mantener un registro actualizado de los cambios y mejoras realizadas en cada proyecto.

No obstante, es importante anotar que, como desarrollador, desempeñé un papel fundamental en la actualización, modificación y mejora de los aplicativos de Gana Plus, la web de Gana, Loti y otras plataformas no mencionadas por temas de seguridad. Este proceso se llevó a cabo mediante un enfoque cualitativo centrado en historias de usuario, documentos de arquitectura y complementado con la metodología ágil Scrum.

El proyecto comenzó con un análisis detallado de las historias de usuario y los documentos de arquitectura pertinentes para cada plataforma en la cual estuve involucrado con su desarrollo. Estas historias y documentos guiaron la planificación y ejecución de las actualizaciones y modificaciones necesarias. Utilizando Scrum, se llevaron a cabo iteraciones cortas y regulares (sprints), durante las cuales se priorizaron y desarrollaron las funcionalidades más críticas según lo definido en las historias de usuario seleccionadas para cada sprint.

En mi rol como desarrollador, colaboré estrechamente con el equipo para implementar estas mejoras y nuevas funcionalidades. Durante los sprints, trabajé en la programación y en la resolución de desafíos técnicos, asegurando que las soluciones implementadas no solo satisfacieran las expectativas de los usuarios, sino que también se entregaran de manera eficiente y dentro de los plazos establecidos.

Este enfoque combinado de Scrum, desarrollo basado en historias de usuario y documentos de arquitectura garantiza que las decisiones de desarrollo estén alineadas estrechamente con las necesidades y deseos reales de los usuarios finales. Mi compromiso estará en contribuir activamente a la mejora continua de los sistemas de software de Gana Plus, la web de Gana, Loti y otras plataformas no mencionadas por temas de seguridad, asegurando así una experiencia mejorada y más efectiva para los usuarios.

A continuación, en la figura 1 se muestra el cronograma de actividades

Figura 1. Cronograma de actividades



6. RESULTADOS

El proyecto se desarrolló a través de diversas fases previamente definidas, cada una de ellas orientada a cumplir con objetivos específicos y alineada con las necesidades de la empresa UX Technology. Cada fase representó un hito en el progreso del proyecto, iniciando con la planificación y configuración del entorno de desarrollo, seguida por la implementación de funcionalidades básicas, y avanzando hacia la integración de componentes más complejos. A medida que se completaban las fases, se realizaron evaluaciones para garantizar que los resultados cumplieran con los estándares de calidad establecidos y los requerimientos técnicos planteados. Este enfoque permitió un avance estructurado y progresivo, fortaleciendo las competencias de los practicantes y asegurando el cumplimiento exitoso de las metas propuestas en cada etapa del proceso.

Implementación de Nuevas Funcionalidades y Mejoras en la Fase 1

Los resultados obtenidos corresponden a las actividades realizadas durante la Fase 1 del proyecto, centrada en la implementación de nuevas funcionalidades y la mejora de las existentes en aplicaciones clave como Gana Plus, la plataforma web de Gana, y otras relacionadas. Esta fase inicial representó un hito importante en el proyecto, sentando las bases para el desarrollo tecnológico alineado con los objetivos empresariales.

Desarrollo Basado en Requisitos

Durante esta etapa, se definieron y documentaron los requisitos específicos que guiaron el desarrollo. Estos requisitos aseguraron que cada funcionalidad implementada cumpliera con los estándares de calidad y respondiera a las necesidades de la empresa y sus usuarios. El seguimiento detallado de estos lineamientos permitió estructurar las tareas de manera eficiente, optimizando los recursos disponibles.

Historias de Usuario como Herramienta Clave

Una de las actividades fundamentales fue la elaboración de historias de usuario detalladas. Estas historias permitieron identificar y priorizar las funcionalidades necesarias, alineándolas con las expectativas de los usuarios finales y los objetivos estratégicos de la organización. Este enfoque metodológico garantizó que cada implementación estuviera orientada a resolver problemas

específicos y agregar valor a las aplicaciones desarrolladas. Ver los siguientes enlaces asociados a las historias de usuarios:

- Darle clic al enlace HU 1
- Darle clic al enlace HU 2
- Darle clic al enlace HU 3

Actividad 1.1: Diseño y Desarrollo de Módulos Funcionales

Como parte de los resultados alcanzados, se llevó a cabo el diseño y desarrollo de módulos funcionales en las aplicaciones asignadas, abordando de manera eficiente las necesidades identificadas en el sistema. Entre estas actividades, destaca el desarrollo de un asistente virtual para una aplicación emergente de la empresa, cuyo nombre se mantiene confidencial por motivos de seguridad.

Para el diseño del módulo, se elaboró una propuesta detallada que fue analizada y aprobada por el gerente general y el coordinador de la fábrica de software. Una vez aprobada, se procedió al desarrollo del asistente virtual siguiendo los estándares establecidos por la empresa. Estos estándares incluyeron prácticas de desarrollo seguro, el uso de una arquitectura basada en microservicios y la implementación de pruebas unitarias para garantizar la calidad del producto final.

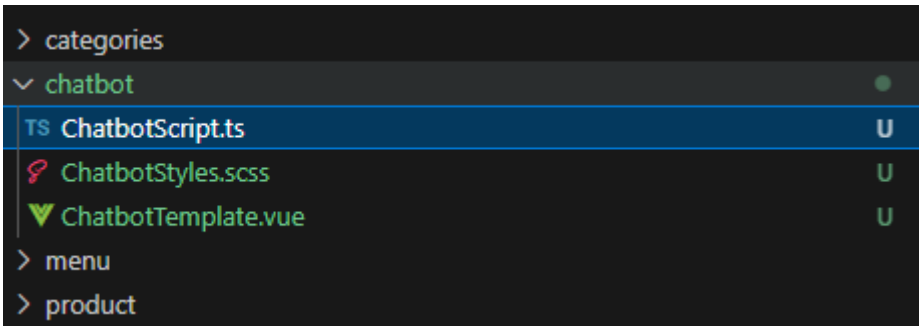
Este proceso demostró un enfoque estructurado y colaborativo, asegurando que las soluciones desarrolladas cumplieran con los lineamientos técnicos y estratégicos de la organización.

A continuación, se procedió al diseño del módulo y a su estructuración conforme a las especificaciones establecidas por la empresa. En este caso particular, se utilizó una arquitectura microfrontend, la cual permite una distribución eficiente de las tareas, asignando responsabilidades específicas a cada módulo para facilitar su desarrollo, mantenimiento y escalabilidad.

El módulo fue organizado de manera estructurada, siguiendo un esquema de encarpetado que optimiza la separación de responsabilidades y fomenta la claridad en el desarrollo. Este esquema fue implementado de acuerdo con los estándares definidos por la empresa, asegurando la alineación con las mejores prácticas en el desarrollo de microfrontends.

Además, el diseño y la implementación se llevaron a cabo utilizando Vue 3, cumpliendo estrictamente con los lineamientos proporcionados por la organización. Estos estándares garantizan no solo la coherencia técnica en el proyecto, sino también la calidad y robustez del módulo dentro del ecosistema tecnológico de la empresa. Este enfoque permitió establecer una base sólida para el desarrollo continuo del sistema, alineada con los principios de modularidad y reutilización.

Figura 2. Encarpetado



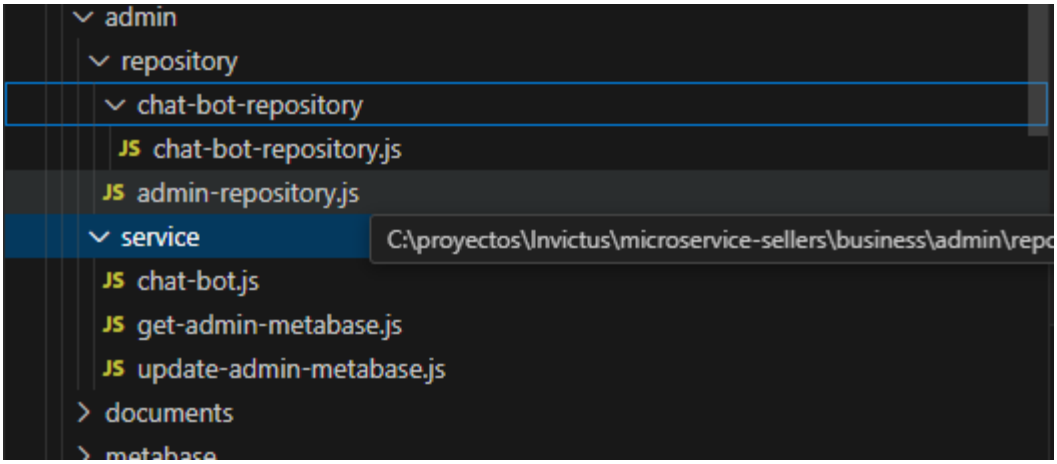
Fuente: Propia del autor

A continuación, se detalla el diseño del backend correspondiente al módulo mencionado, desarrollado como un microservicio para garantizar una integración eficiente y directa con el frontend. Este enfoque permite una comunicación ágil y escalable entre ambos componentes, asegurando una respuesta óptima a las solicitudes provenientes del microfrontend.

El microservicio fue diseñado para manejar tareas específicas, manteniendo la independencia funcional que caracteriza esta arquitectura. Se implementaron estándares de desarrollo seguro proporcionados por la empresa, garantizando la protección de los datos y la integridad de las operaciones realizadas a través del módulo. Además, se incorporaron pruebas unitarias y de integración para validar el correcto funcionamiento del servicio y su interacción con el sistema en general.

Esta implementación no solo fortalece la funcionalidad del módulo, sino que también respalda la modularidad y la capacidad de actualización del sistema, alineándose con los objetivos técnicos y estratégicos de la empresa.

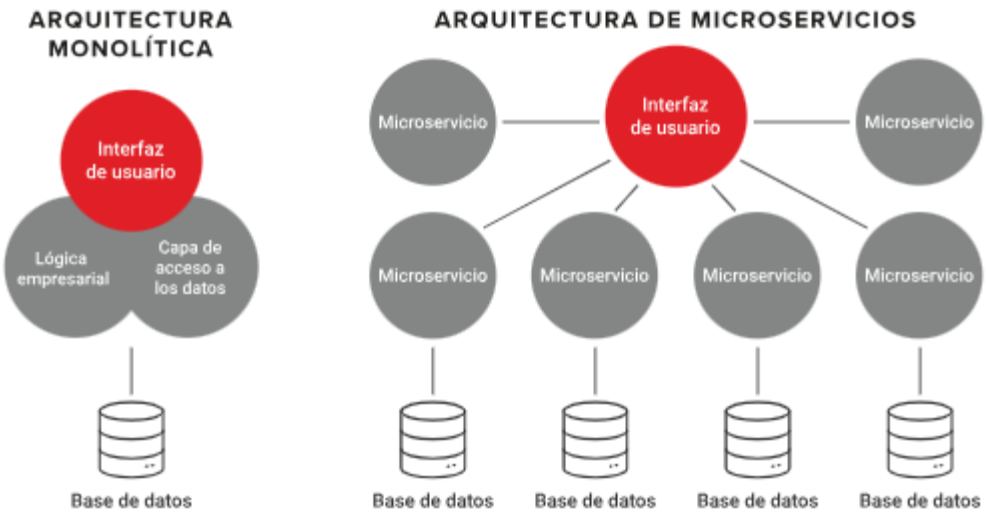
Figura 3. Microservicios



Fuente: Propia del autor

Para ejemplificar los microservicios se distribuyen con el fin de realizar un desarrollo de manera independiente como se muestra en la figura 4.

Figura 4. Ejemplo MS



Fuente: Chakray. (2021). Microservices diagram. Recuperado de https://www.chakray.com/wp-content/uploads/2021/05/microservices-spanish.svg

A continuación, se describe la integración del asistente virtual en la interfaz del nuevo aplicativo desarrollado por la empresa. El asistente fue ubicado estratégicamente en la parte inferior derecha de la pantalla, garantizando una experiencia de usuario intuitiva y no intrusiva, acorde con los estándares de diseño de la organización.

La implementación se llevó a cabo aplicando los aspectos técnicos previamente mencionados, incluyendo la arquitectura microfrontend, el desarrollo seguro, y el cumplimiento de los estándares definidos para Vue 3. Este diseño se sustentó en las Historias de Usuario (HU) elaboradas durante el proceso de planificación, las cuales sirvieron como base para priorizar funcionalidades y asegurar que la integración del asistente virtual cumpliera con los requerimientos tanto técnicos como funcionales.

Esta adición representa un avance significativo en la funcionalidad del aplicativo, proporcionando a los usuarios una herramienta dinámica que mejora la interacción y facilita el acceso a las soluciones ofrecidas por la empresa.

Figura 5. Asistente



Fuente: Propia del autor

Actividad 1.2: Integración de Nuevas Características en Aplicaciones Existentes

Como parte de los resultados alcanzados, se integraron nuevas características en las aplicaciones existentes, asegurando una actualización fluida y coherente con los requisitos establecidos. Estos requisitos, previamente documentados en las Historias de Usuario (HU),

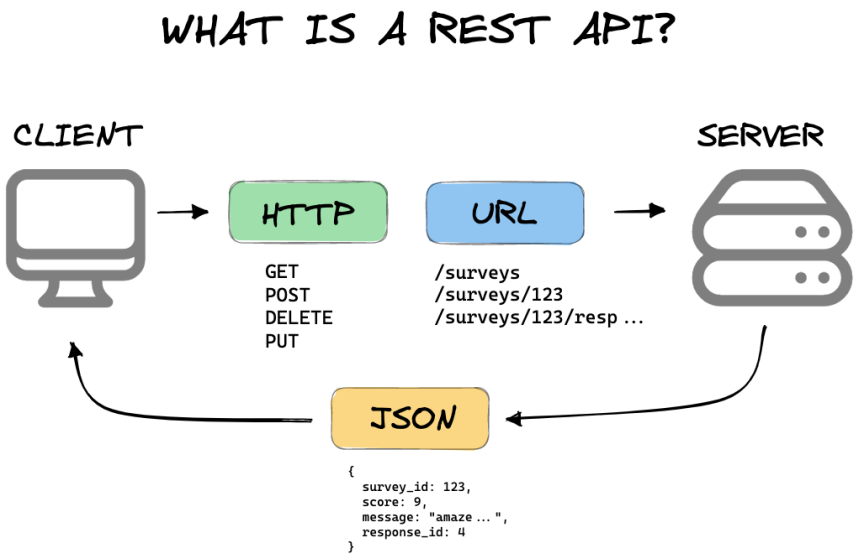
sirvieron como base para guiar el desarrollo y garantizar el cumplimiento de los objetivos del proyecto.

En este caso particular, la integración principal consistió en el consumo de una API REST para habilitar las funcionalidades del asistente virtual en los microservicios. El consumo de esta API REST fue diseñado e implementado siguiendo un enfoque estructurado que asegura la interoperabilidad y el cumplimiento de los estándares de la empresa.

Uso de Confluence API

La empresa utiliza Confluence, que cuenta con su propia API REST. Esta API fue empleada para obtener información y sugerencias necesarias para enriquecer las funcionalidades del asistente virtual. En la imagen adjunta, se muestra cómo se utilizó la API REST para consultar datos y proporcionar sugerencias en tiempo real, las cuales fueron integradas directamente en el nuevo módulo del asistente virtual.

Figura 6. Rest API



Fuente: Mann, H. (2023, marzo 24). Ejemplo de REST API [Imagen]. Recuperado de https://mannhowie.com/rest-api

Funcionalidades Adicionales

Además de consumir la API, estas nuevas características fueron incorporadas directamente en un nuevo aplicativo desarrollado por la empresa. En la imagen, se evidencia cómo el chat del asistente entrega una serie de sugerencias dinámicas obtenidas gracias a la nueva funcionalidad. Este piloto, si es aprobad, podrá ser reutilizado en otras aplicaciones futuras de la empresa. Por motivos de confidencialidad, solo se presenta el trabajo realizado por el estudiante, manteniendo borrosos los elementos relacionados con información interna de la empresa, ver figura 7.

Figura 7. Chat



Fuente: Propia del autor

Actividad 1.3: Revisión Exhaustiva de Código

Para garantizar la calidad del desarrollo, se realizaron revisiones exhaustivas de código, asegurando la adherencia a los estándares establecidos por la empresa. Estas revisiones tuvieron como objetivo mejorar la mantenibilidad y el rendimiento del software, así como asegurar la uniformidad en el estilo arquitectónico basado en microservicios.

Proceso de Revisión

Durante la implementación del asistente virtual, se revisó el código para alinearlo con la arquitectura estándar de microservicios de la empresa. Además, para garantizar un control continuo y documentado, se empleó un enfoque basado en commits incrementales. Estos commits registraron cada cambio realizado, permitiendo una revisión iterativa y asegurando que cada actualización cumpliera con los requisitos técnicos y de calidad.

En los repositorios trabajados, se aplicaron estas mismas prácticas para mantener la consistencia en todos los módulos. Las evidencias de estos procesos están disponibles y, en las imágenes adjuntas, se muestran ejemplos específicos de cómo se gestionaron y documentaron estos cambios. Ver figura 8.

Figura 8. Commits

Commit ID	Commit message	Commit date	Authored date	Author	Committer	Actions
43a086e7	feat: Se ajusta el codigo para evitar duplicaciones	14 hours ago	14 hours ago	UX\daniel.loaiza	UX\daniel.loaiza	Copy ID Browse
4a96f576	feat: Se agregan las entities necesarias y se realizan eliminaciones de logs inn...	17 hours ago	17 hours ago	UX\daniel.loaiza	UX\daniel.loaiza	Copy ID Browse
ee9fe36f	feat: Se realizan ajustes para el correcto desarrollo y arquitectura	18 hours ago	18 hours ago	UX\daniel.loaiza	UX\daniel.loaiza	Copy ID Browse
fda9a479	feat: Se realizan las preubas unitarias pra las nuevas funcionalidades implement...	1 day ago	1 day ago	UX\daniel.loaiza	UX\daniel.loaiza	Copy ID Browse
d8c694e3	feat: Se agrega modal y funcionamiento de todo el chat	1 day ago	1 day ago	UX\daniel.loaiza	UX\daniel.loaiza	Copy ID Browse

Fuente: Propia del autor

Cada commit tiene el objetivo de demostrar que trabajo se realizó y se envió.

En los commit se evidencia un desarrollo escalonado que paso por varios procesos y a modo de conclusión se evidencia el desarrollo y trabajo realizado de forma correcta y siguiendo los estándares propuestos

Pruebas y Depuración de Código en la Fase 2

La **Actividad 2** del proyecto se centró en las tareas de pruebas y depuración de código, con el objetivo de garantizar la calidad y funcionalidad de los módulos desarrollados. Estas actividades son fundamentales para asegurar que las implementaciones realizadas cumplan con los estándares técnicos y los requisitos establecidos por la empresa.

El proceso incluyó la ejecución de **pruebas unitarias** e **integraciones** utilizando herramientas como **Mocha**, lo que permitió validar el comportamiento esperado de funciones y módulos específicos. Uno de los enfoques principales fue el módulo **chat-bot-repository.js**, encargado de consumir una API REST y procesar datos para su uso en funcionalidades clave del sistema. Estas pruebas ayudaron a identificar áreas de mejora mediante análisis de cobertura de código, destacando líneas de código críticas para ajustes.

Actividad 2.1: Ejecutar Pruebas Unitarias y de Integración en los Módulos Desarrollados

Como parte de los resultados obtenidos, se llevó a cabo la ejecución de pruebas unitarias y de integración en los módulos desarrollados, utilizando **Mocha** como marco principal. Mocha permitió organizar y ejecutar pruebas de manera eficiente, validando que las funciones y módulos implementados funcionaran correctamente.

Contexto y Uso de Mocha

Mocha es un marco de pruebas unitarias para JavaScript que opera en entornos como **Node.js** y navegadores web. Su objetivo principal es facilitar la escritura y ejecución de pruebas mediante descripciones claras y flujos de trabajo basados en declaraciones como describe e it. Además, Mocha soporta bibliotecas como **Chai**, que permiten verificar si los resultados obtenidos cumplen con las expectativas predefinidas.

Ejemplo de Prueba

Para ilustrar este proceso, se desarrolló un ejemplo con la función suma, cuyo objetivo es retornar la suma de dos números. La función fue probada de la siguiente forma:

```
function suma(a, b) {
  return a + b;
}
module.exports = suma;
```

Ahora la prueba se realiza con el siguiente archivo suma.spec.js:

```
const assert = require('assert'); // Biblioteca de aserción nativa de Node.js
const suma = require('../suma'); // Importar la función a probar

describe('Pruebas de la función suma', function() {
  it('Debería devolver 5 cuando los números sean 2 y 3', function() {
    const resultado = suma(2, 3);
    assert.strictEqual(resultado, 5); // Verifica que el resultado sea igual a 5
  });

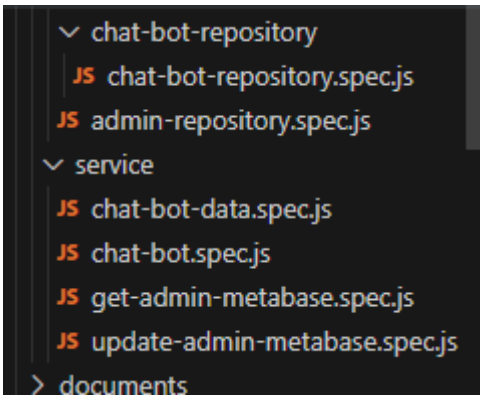
  it('Debería devolver -1 cuando los números sean -3 y 2', function() {
    const resultado = suma(-3, 2);
    assert.strictEqual(resultado, -1); // Verifica que el resultado sea igual a -1
  });
});
```

Si todo sale bien la prueba dará el siguiente resultado :

```
Pruebas de la función suma
✓ Debería devolver 5 cuando los números sean 2 y 3
✓ Debería devolver -1 cuando los números sean -3 y 2
2 passing (10ms)
```

Este ejemplo demuestra cómo se validaron los resultados esperados mediante pruebas unitarias, estableciendo la base para ejecutar pruebas en módulos más complejos. A partir de este contexto, se aplicaron estas pruebas en módulos desarrollados para aplicaciones de la empresa, garantizando su funcionalidad y alineación con los estándares establecidos.

Figura 9. Archivos prueba



Fuente: Propia del autor

Como se observa, en la imagen anterior se desarrollaron varios módulos de pruebas, destacando entre ellos el módulo **chat-bot-repository.js**, por su complejidad y alcance. Este módulo se centra en consumir una API REST y procesar los datos obtenidos para su uso en otras funcionalidades. Las pruebas unitarias e integraciones realizadas garantizaron su correcto funcionamiento, validando tanto la interacción con la API como el procesamiento de datos, asegurando así que cumpliera con los estándares y requisitos de la empresa.

su funcionamiento es el de consumir un Rest api y limpiar los datos, con el fin de que se pueda entender se proporciona el codigo de su funcionamiento

```
/* eslint-disable max-lines */
const { createClient } = require('@matrixtech/fw-http-client/service/http-client');
const { loadParameter } = require('@matrixtech/fw-system-parameters/business/parameter-configuration');
const cheerio = require('cheerio');
const { ID_CONFLUENCE }= require('../../../../business/constants/constants');
```

```
/**
 * @module private
 * @description Realiza una solicitud GET a la API de Confluence y procesa la respuesta para extraer y clasificar los datos
 */
const getConfluenceData = async (resourceId=null) => {
    const apiToken = await loadParameter('/SELLERS/CONFLUENCE_API_KEY');
    const confluenceUrl = await loadParameter('/SELLERS/CONFLUENCE_URL');
    const email = await loadParameter('/SELLERS/CONFLUENCE_EMAIL');
    const defaultId= ID_CONFLUENCE.ID;
    const authValue = `Basic ${Buffer.from(`${email}:${apiToken}`).toString('base64')}`;
    const headers = {
        Authorization: authValue,
        'Content-Type': 'application/json',
    };
    const options = {
        headers,
        method: 'GET',
    };
    const resourcePath = `/wiki/rest/api/content/${resourceId || defaultId}?expand=body.view`;
    return await consumeConfluenceService(options, confluenceUrl,resourcePath, defaultId);
};
/**
 * @description Llama a la API de Confluence, procesa el contenido HTML, y clasifica los recursos
 * @param {*} options
 * @param {*} confluenceUrl
 * @returns {Object} Datos clasificados y sugerencias
```

```
 */
const consumeConfluenceService = async (options, confluenceUrl, resourcePath, defaultId) => {
    let result = { success: false };
    try {
        const client = getClient(confluenceUrl, options.headers);
        const { data } = await client.get(resourcePath);
        let processedData;

        if (data.errors) {
            let errors = Object.values(data.errors);
            throw new Error(errors.join('\n'));
        }
        if (resourcePath.includes(defaultId)) {
            processedData = processConfluenceResponse(data);
        } else {
            processedData=data;
        }
        result = {
            success: true,
            response: processedData
        };
    } catch (error) {
        return { responses: {}, suggestions: { confp: [], formato: [] } };
    }
    return result;
};
/**
 * @description Procesa la respuesta de Confluence para extraer y clasificar los recursos
 * @param {*} responseData - Datos obtenidos de la API de Confluence
```

```
 * @returns {Object} Datos clasificados con respuestas y sugerencias
 */
const processConfluenceResponse = (responseData) => {
    const bodyView = responseData.body.view ? responseData.body.view.value : null;
    if (!bodyView) {
        return { responses: {}, suggestions: { confp: [], formato: [] } };
    }
    const $ = cheerio.load(bodyView);
    const recursos = [];
    let enFormatos = false;
    $('h1, h2, h3, a[data-linked-resource-id]').each((i, el) => {
        const tagName = $(el).get(0).tagName;
        if ($(el).text().trim() === 'FORMATOS DE IMPRESIÓN.') {
            enFormatos = true;
        } else if (tagName === 'h1' || tagName === 'h2' || tagName === 'h3') {
            if (enFormatos) enFormatos = false;
        } else if (tagName === 'a') {
            const resourceId = $(el).attr('data-linked-resource-id');
            const text = $(el).text().trim();
            recursos.push({
                id: resourceId,
                text: text,
                categoria: enFormatos ? 'Formatos de Impresión' : 'Configuración'
            });
        }
    });
    const confp = recursos.filter(r => r.categoria === 'Configuración').map(r => ({ text: r.text, url: r.id }));
    const formato = recursos.filter(r => r.categoria === 'Formatos de Impresión').map(r => ({ text: r.text, url: r.id }));
```

```
    return {
        responses: {
            hola: '¿Cómo puedo ayudarte?',
            confp: 'Aquí tienes las configuraciones.',
            formato: 'Aquí están los formatos disponibles.'
        },
        suggestions: {
            hola: [
                { text: 'Configuración de productos', next: 'confp' },
                { text: 'Formato de impresión', next: 'formato' }
            ],
            confp: confp,
            formato: formato
        }
    };
};
/**
 * @param {*} url
 * @param {*} headers
 * @returns
 */
const getClient = (url, headers) => {
    const TIMEOUT_SERVICE = 30000; // Timeout de 30 segundos
    const client = createClient(url, TIMEOUT_SERVICE);
    client.defaults.headers.common = {
        ...client.defaults.headers.common,
        ...headers,
    };
    return client;
};
```

```
module.exports = {
   getConfluenceData,
};
```

aunque sea un poco extenso el objetivo es demostrar como se efectuó la prueba unitaria para un módulo desarrollado y así denotar el trabajo tan extenso que se hizo en cada uno,

con el fin de no hacerlo tan extenso mostraremos solo la fundamentación de la prueba unitaria y el desarrollo de la misma

```
const chai = require('chai');
const chaiAsPromised = require('chai-as-promised');
chai.use(chaiAsPromised);
const { expect } = chai;
const sinon = require('sinon');
const { initSandbox, getComponent } = require('@matrixtech/fw-test/unit-utils/utils');


describe.only('business | admin | repository | chat-bot-repository', () => {
   context('chat-bot-repository', () => {
      const sandbox = initSandbox();
      let component = null;
      let getResponse= null;
      let response = {};
      let responses={};



      const errorResponse = {
         errors: {
```

```
        test1: 'error1',
        test2: 'error2',
    },
};



const clientGetSpy = sandbox.spy();
const loadParameterSpy = sandbox.spy();
const createClientSpy = sandbox.spy();

beforeEach(() => {
    response = {
        get: (endpoint) => {
            clientGetSpy(endpoint);
            return getResponse;
        },
        defaults: {
            headers: {
                common: {},
            }
        }
    };

    getResponse = {
        data: {
            body: {
                view: {
                    value: '<h1>Configuración</h1><a data-linked-resource-id="123">Link
to Resource</a><h2>FORMATOS DE IMPRESIÓN.</h2><a data-linked-resource-
```

```
id="456">Formato de Impresión 1</a><a data-linked-resource-id="789">Formato de Impresión 2</a>',
                        },
                    },
                },
            };


            component = getComponent('/business/admin/repository/chat-bot-repository/chat-bot-repository', sandbox, {
                '@matrixtech/fw-http-client/service/http-client': {
                    createClient: (url, timeout) => {
                        createClientSpy(url, timeout);
                        return response;
                    },
                },
                '@matrixtech/fw-system-parameters/business/parameter-configuration': {
                    loadParameter: (name) => {
                        loadParameterSpy(name);
                        return name; // Mocking the parameter loading
                    },
                },
            });
        });

        afterEach(() => {
            sandbox.reset();
        });

        after(() => {
```

```
    sandbox.restore();
  });

  it('should retrieve and process data from Confluence', async () => {
    const result = await component.getConfluenceData();

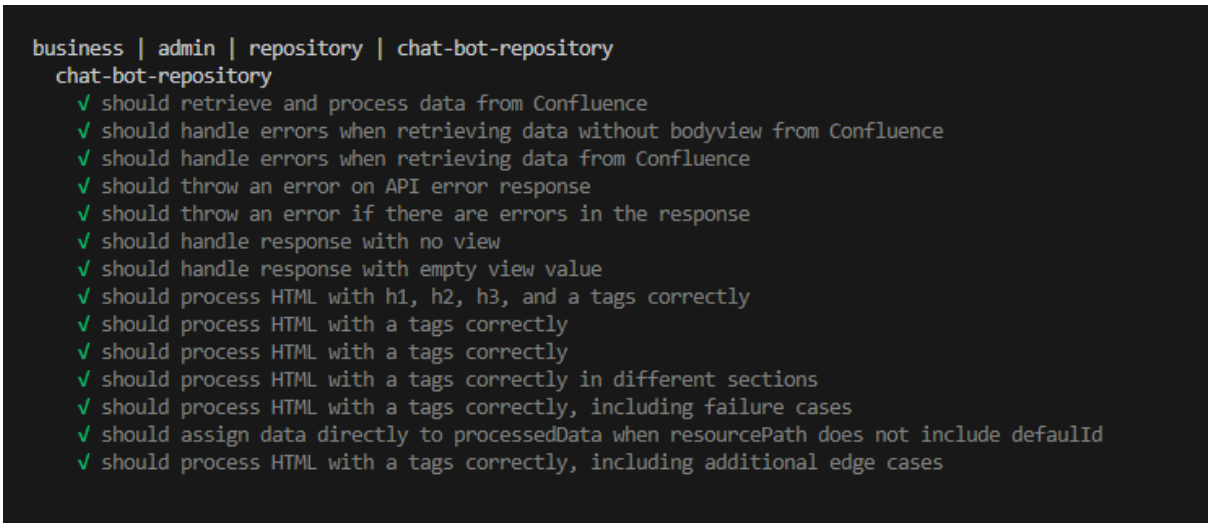
    expect(loadParameterSpy.callCount).to.equal(3); // Expect 3 parameters to be loaded
    /* expect(consumeConfluenceServiceSpy.calledOnce).to.be.true; */
    expect(createClientSpy.calledOnce).to.be.true;

    expect(result).to.deep.equal({
      success: true,
      response: {
        responses: {
          hola: '¿Cómo puedo ayudarte?',
          confp: 'Aquí tienes las configuraciones.',
          formato: 'Aquí están los formatos disponibles.'
        },
        suggestions: {
          hola: [
            { text: 'Configuración de productos', next: 'confp' },
            { text: 'Formato de impresión', next: 'formato' }
          ],
          confp: [{ text: 'Link to Resource', url: '123' }],
          formato: [{ text: 'Formato de Impresión 1', url: '456' },
            { text: 'Formato de Impresión 2', url: '789' },
          ],
        },
      },
```

```
    });
  });
```

Se presenta una única prueba unitaria por dos razones principales: en primer lugar, para evitar una visualización excesivamente extensa del código, y en segundo lugar, porque con una sola prueba es suficiente para demostrar correctamente el desarrollo realizado. Los resultados de la prueba pueden observarse en la figura 10.

Figura 10. Resultado Pruebas



Fuente: Propia del autor

Se observa que además de la primera prueba se pasaron otras 13 pruebas, que cada una busca probar los aspectos del código que anteriormente se proporcionó, además téngase en cuenta que esto fue solo para un módulo y se realizó el mismo proceso exacto para otros módulos como se aprecia en la figura 10.

Actividad 2.2: Identificar y corregir errores en el código.

para la identificación de los errores y corrección de los errores se utilizan las mismas pruebas unitarias, ya que si estas fallan pues significa que algo falta en el codigo y además de eso con el mismo desarrollo de dichas pruebas, podemos realizar algo que se llama cobertura, la cobertura es una forma de medir cuánto del código de un programa se ejecuta cuando corremos pruebas, como las pruebas unitarias. Básicamente, nos dice qué partes del código hemos probado

y cuáles no, ayudándonos a encontrar posibles errores o partes que no hemos revisado. Hay diferentes tipos de cobertura, como:

Cobertura de líneas: verifica cuántas líneas de código se ejecutaron.

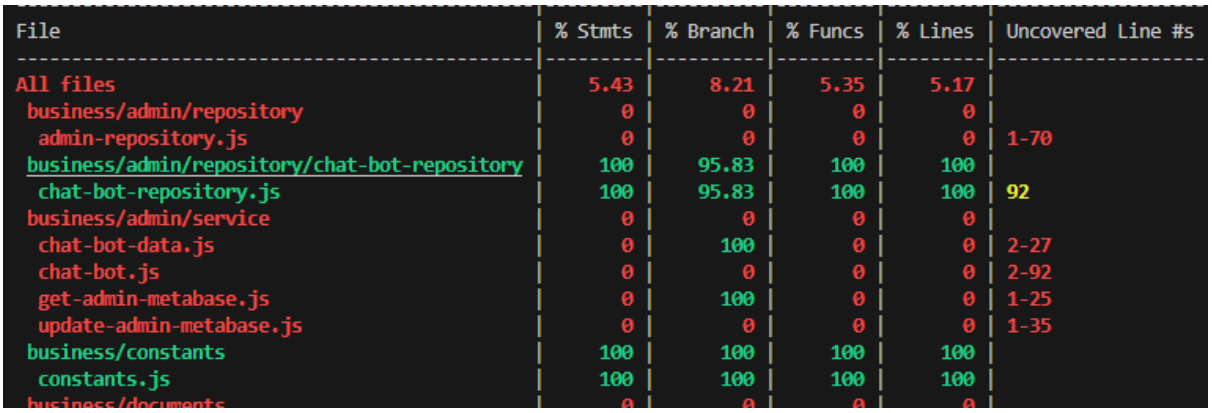
Cobertura de ramas: revisa si probamos todas las opciones de un if o un else.

Cobertura de funciones: muestra si todas las funciones fueron llamadas al menos una vez.

Las pruebas unitarias son muy útiles para mejorar la cobertura porque prueban pedazos pequeños del código, como funciones o métodos. Sin embargo, tener una cobertura alta no significa que el código esté 100% libre de errores; simplemente que lo hemos revisado más a fondo.

Ya teniendo ésto en claro pasemos a ver como maneja la cobertura el código que se ha mostrado anteriormente, ver figura 11:

Figura 11. Cobertura



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	5.43	8.21	5.35	5.17	
business/admin/repository	0	0	0	0	
admin-repository.js	0	0	0	0	1-70
business/admin/repository/chat-bot-repository	100	95.83	100	100	
chat-bot-repository.js	100	95.83	100	100	92
business/admin/service	0	0	0	0	
chat-bot-data.js	0	100	0	0	2-27
chat-bot.js	0	0	0	0	2-92
get-admin-metabase.js	0	100	0	0	1-25
update-admin-metabase.js	0	0	0	0	1-35
business/constants	100	100	100	100	
constants.js	100	100	100	100	
business/documents	0	0	0	0	

Fuente: Propia del autor

Ahora bien, se puede apreciar que se tiene buena cobertura en este caso nos enfocamos en el chat-bot-repository.js; sin embargo, esta no es perfecta en su totalidad ya que como podemos ver no llega al 100 por ciento lo que indica que hay que realizar algún cambio, en este caso la cobertura permite divisar donde está el problema el cual se evidencia en la línea 92 del código en cuestión, problema que se corregirá eficientemente.

Actividad 2.3: Documentar los resultados de las pruebas y las correcciones realizadas.

La documentación de dichos resultados y correcciones se evidencian en los commits, un **commit** en Git es un registro o punto de control que captura los cambios realizados en el código o en los archivos de un repositorio en un momento específico. Cada commit guarda información

sobre los cambios realizados, el autor, la fecha y un mensaje descriptivo que explica el propósito de esos cambios. Es como tomar una "foto" del estado actual del proyecto para poder volver a ella si es necesario, pero la pregunta seria cómo documenta los resultados de las pruebas, pues Cada vez que se realiza un commit, este ejecuta previamente las pruebas unitarias y verifica la cobertura para todo el proyecto, incluido el desarrollo propio. Una vez completado este proceso, los resultados se registran, como se muestra en la figura 10, y quedan almacenados directamente en el registro específico de Git correspondiente.

Fase 3: Planificación de entregas oportuna de soluciones tecnológicas.

La **Fase 3** se centró en la planificación estratégica para garantizar la entrega oportuna de soluciones tecnológicas. Este proceso incluyó la definición de cronogramas claros, la asignación eficiente de recursos y la colaboración activa del equipo, asegurando que cada entrega cumpliera con los estándares de calidad establecidos y los plazos acordados.

Actividad 3.1: Participar activamente en las reuniones de planificación de sprints

Se participó activamente en las reuniones de planificación de sprints, donde se realizaron aportes significativos relacionados con las estimaciones de tiempo y la definición de las tareas asignadas. Estas reuniones permitieron una mejor organización del trabajo, identificando prioridades y ajustando los recursos para garantizar el cumplimiento de los objetivos del sprint.

Actividad 3.2: Colaborar con el equipo en la división y asignación de tareas

Durante estas sesiones, se colaboró con el equipo en la división y asignación de tareas, asegurando una distribución equilibrada de la carga de trabajo. Este enfoque permitió que cada miembro pudiera gestionar sus responsabilidades de manera eficiente, contribuyendo al avance coordinado de las actividades del proyecto.

Actividad 3.3: Realizar seguimientos periódicos del progreso de las tareas asignadas

Para asegurar el cumplimiento de los plazos establecidos, se realizaron seguimientos periódicos de las tareas asignadas. Estos seguimientos permitieron ajustar los planes cuando fue necesario, abordando problemas imprevistos y optimizando los recursos para cumplir con las metas propuestas.

Actividad 3.4: Participar en las reuniones de revisión de sprints

En las reuniones de revisión de sprints, se presentaron los resultados de las tareas completadas, destacando los avances logrados y los desafíos superados. Además, se discutieron

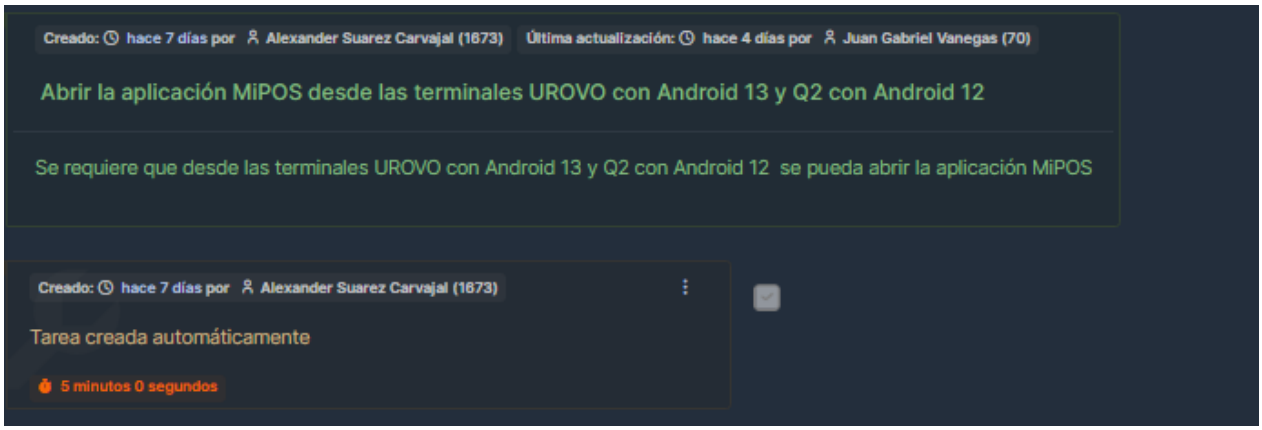
posibles mejoras y ajustes para los próximos ciclos, asegurando un proceso de desarrollo continuo y enfocado en la calidad.

Sistema de Gestión: GLPI

Para gestionar los casos y monitorear el progreso de las tareas, se utilizó el sistema GLPI. Este sistema facilitó la creación, el seguimiento y el cierre de casos asociados a las actividades del sprint, proporcionando una visión clara del estado de cada tarea y permitiendo una colaboración más efectiva entre los miembros del equipo. En las imágenes posteriores, se explicó en detalle el flujo de trabajo implementado dentro de GLPI, destacando cómo se gestionaron los casos y su proceso asociado.

A continuación, se muestra el funcionamiento del GLPI con un incidente de APK Android realizado.

Figura 12. GLPI 1



Fuente: Propia del autor

Se Asigna el Caso en la figura 12 se asigna un caso, en el cual se buscaba solucionar la apertura de una aplicación la cual fallaba en versiones superiores de sistemas Android, pasando a la siguiente figura se continúa con el caso.

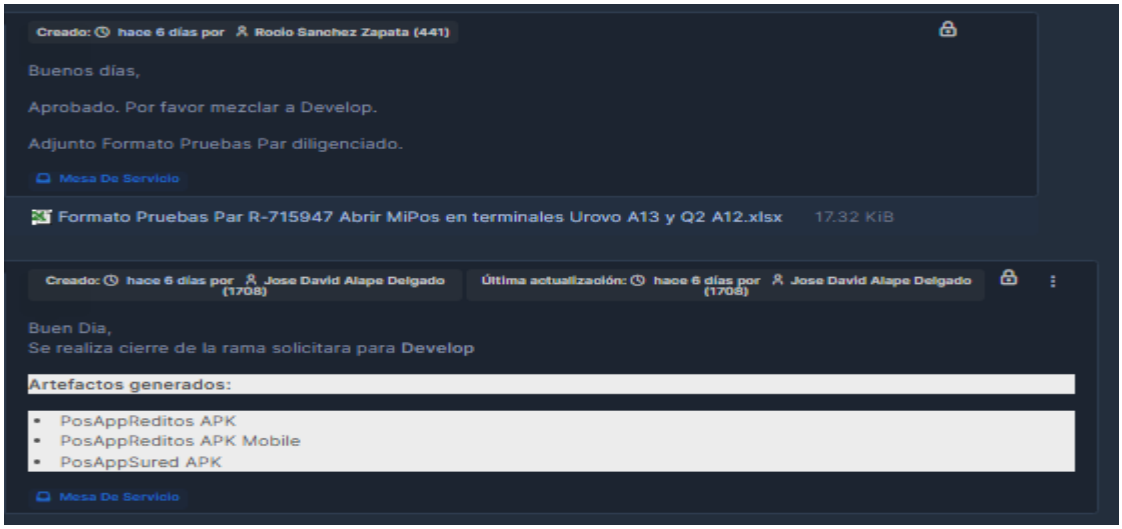
Figura 13. GLPI 2



Fuente: Propia del autor

Se entrega solución del error, en este momento se envió y desarrolló una solución de error mencionado al inicio del caso, por lo que con el caso se informó que ya estaba disponible la solución.

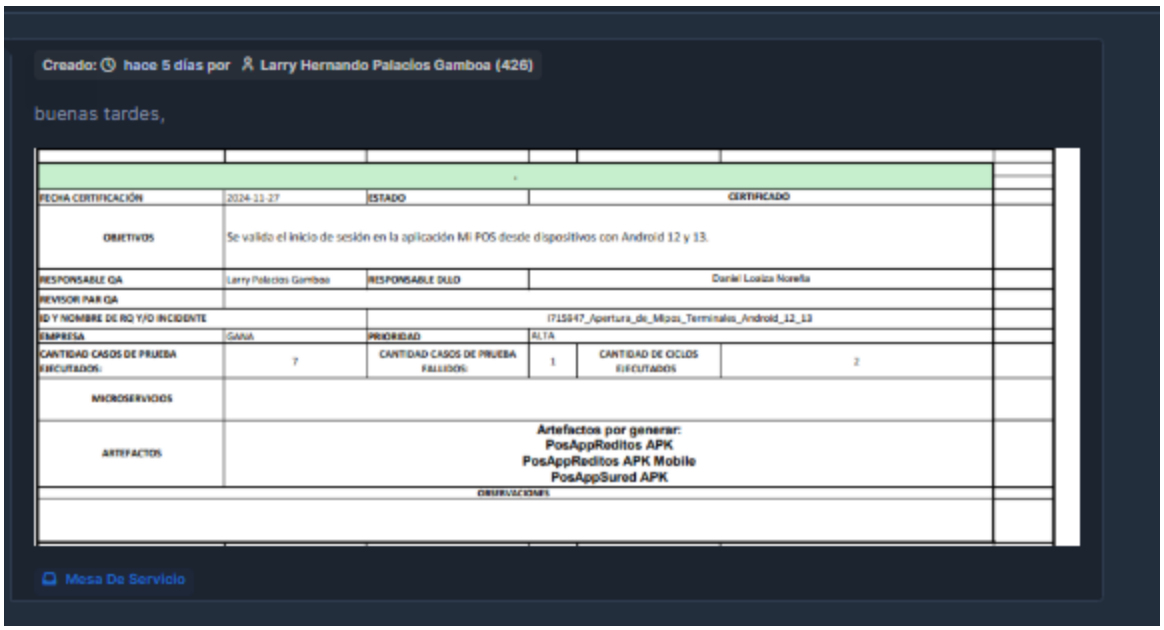
Figura 14. GLPI 3



Fuente: Propia del autor

Se realiza prueba par y posterior despliegue a QA área de pruebas, como se observa en la figura 14 se realiza vía caso el sistema de pruebas par en el cual un arquitecto valida la solución y además se envía para que el QA encargado realice la prueba correspondiente.

Figura 15. GLPI 4



Fuente: Propia del autor

Se certifica por QA, el qa correspondiente hizo sus pruebas y como se ve en la figura 15 envía su certificación.

Figura 16. GLPI 5



Fuente: Propia del autor

Se hace solicitud para producción, ya habiendo pasado por todos filtros mencionados en figuras anteriores se envía solicitud para que en producción se realice en despliegue se videncia en la figura 16.

Fase 4: Documentación de los Procesos de Desarrollo y Resultados Obtenidos

La **Fase 4** se centró en la creación y mantenimiento de documentación técnica y funcional para garantizar la continuidad y comprensión de los proyectos desarrollados. Esta fase es esencial para consolidar el conocimiento adquirido y facilitar futuras implementaciones.

Actividad 4.1: Se elaboró documentación técnica detallada de los desarrollos realizados, incluyendo descripciones de la arquitectura, los módulos implementados y las integraciones realizadas, siguiendo los estándares de la empresa.

Para obtener detalles sobre el tipo de documentación técnica utilizada, se recomienda consultar las Historias de Usuario (HU) mencionadas en la Fase 1.

Actividad 4.2: Se diseñaron guías de usuario claras y concisas para las nuevas funcionalidades, asegurando que los usuarios finales pudieran comprender y aprovechar las herramientas implementadas de manera eficiente.
En este caso para el desarrollo de un asistente virtual se crea y diseña un manual de usuario con ejemplos detallados

- Darle clic Manual de Usuario

Actividad 4.3: Se mantuvo un registro actualizado de los cambios y mejoras realizados en cada proyecto, utilizando sistemas de control de versiones y documentación colaborativa, lo que permitió un seguimiento transparente del progreso y resultados.

7. ANÁLISIS

El desarrollo de la propuesta de prácticas permitió identificar logros significativos y áreas de mejora clave dentro del proceso de implementación. A lo largo de las fases descritas, se evidenció un avance estructurado en la adquisición de habilidades técnicas y la aplicación de metodologías ágiles como Scrum. Las actividades realizadas no solo contribuyeron al desarrollo de soluciones funcionales para las aplicaciones asignadas, sino que también permitieron un entendimiento profundo de los estándares y requerimientos empresariales.

La integración de microservicios y microfrontends reflejó un enfoque moderno y escalable en la construcción de software, garantizando modularidad y facilidad de mantenimiento. El uso de pruebas unitarias e integraciones permitió identificar áreas críticas del código, optimizando su rendimiento y funcionalidad mediante una alta cobertura. Esto fortaleció la calidad del software desarrollado y la capacidad de respuesta ante posibles errores.

Por otro lado, la participación activa en sprints y la utilización de herramientas como GLPI demostraron un manejo eficiente de la planificación y la gestión de tareas, destacando la importancia del trabajo colaborativo y la comunicación constante en un entorno laboral. Este enfoque promovió la entrega puntual de soluciones alineadas con los objetivos empresariales.

En síntesis, el proyecto sirvió como una plataforma integral para consolidar competencias técnicas y prácticas en el desarrollo de software, proporcionando a los participantes una experiencia robusta y orientada al mundo laboral real. Los resultados obtenidos resaltan la importancia de adoptar metodologías modernas y enfoques estructurados en la ejecución de proyectos tecnológicos.

8. CONCLUSIONES

El desarrollo de este proyecto permitió cumplir con los objetivos establecidos, logrando implementar soluciones tecnológicas funcionales y escalables para las aplicaciones asignadas. A través de la integración de herramientas y metodologías modernas como Scrum, Node.js, y pruebas unitarias, se fortalecieron habilidades técnicas y profesionales, garantizando entregas de alta calidad alineadas con los estándares empresariales.

Entre los principales logros, se destaca la implementación de un asistente virtual utilizando una arquitectura basada en microservicios y microfrontends, lo cual optimizó la modularidad, la reutilización de componentes, y la escalabilidad del sistema. Adicionalmente, el uso de pruebas exhaustivas no solo permitió validar la funcionalidad de los módulos desarrollados, sino también detectar y corregir errores de manera eficiente, aumentando la confiabilidad del producto final.

El manejo de herramientas de gestión como GLPI y la participación en la planificación y revisión de sprints evidenciaron la importancia de la organización y el trabajo en equipo en un entorno ágil. Estos elementos fueron clave para asegurar el cumplimiento de los plazos y objetivos planteados, promoviendo una comunicación efectiva y un flujo de trabajo óptimo.

Finalmente, este proyecto consolidó una experiencia práctica valiosa, no solo en el desarrollo de software, sino también en la documentación técnica y la colaboración en equipos multidisciplinarios. Este aprendizaje ha preparado a los participantes para enfrentar los desafíos del entorno laboral actual, contribuyendo al desarrollo de soluciones tecnológicas innovadoras y de impacto.

REFERENCIAS

Agile Alliance. (2023). Microservices diagram [Imagen]. Recuperado de https://www.chakray.com/wp-content/uploads/2021/05/microservices-spanish.svg

Angular. (s.f., 2024). Angular Documentation. Recuperado de https://angular.io

Beck, K., et al. (2001). Manifesto for Agile Software Development. Agile Alliance.

Cantelon, M., Harter, M., & Rajlich, T. (2014). Node.js in Action (2nd ed.). Manning Publications.

Chacon, M., Harter, M., & Rajlich, T. (2014). Node.js in Action (2nd ed.). Manning Publications.

Deitel, P., Deitel, H., Wald, A., & Lin, T. (2016). Android™ 6 for Programmers: An App-Driven Approach (3rd ed.). Pearson.

Gosling, J., Joy, B., Steele, G., & Bracha, G. (2014). The Java Language Specification (Java SE 8 Edition). Addison-Wesley.

Kaner, C., Falk, J., & Nguyen, H. Q. (2017). Testing Computer Software (2nd ed.). Wiley.

Kline, K., Kline, B., & Hunt, D. (2016). SQL in a Nutshell: A Desktop Quick Reference (3rd ed.). O'Reilly Media.

Mann, H. (2023, marzo 24). Ejemplo de REST API [Imagen]. Recuperado de https://mannhowie.com/rest-api

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide. Scrum.org.

Siemens, G., & Long, P. (2011). Penetrating the fog: Analytics in learning and education. EDUCAUSE Review, 46(5), 30-32.

Sommerville, I. (2011). Software Engineering (9th ed.). Pearson.