Pontificia Universidad Javeriana Cali
Facultad de Ingeniería.
Maestría en Ingeniería de Software.
Proyecto de Grado.

QuickMoto: Diseño de arquitectura para aplicación móvil de transporte de mercancía en Motocicletas

Edison Antonio Vasquez Burbano

Director: Dr. Gerardo Mauricio Sarria Montemiranda

24 de Julio de 2024

Santiago de Cali, 24 de Julio de 2024.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Por medio de la presente hago constar que en mi calidad de director de trabajo de grado he revisado el proyecto titulado "QuickMoto: Diseño de arquitectura para aplicación móvil de transporte de mercancía en Motocicletas" realizado por el estudiante de Magister en Ingeniería de Software Edison Antonio Vasquez Burbano (cod: 8955310), el cual se encuentra terminado y considero que cumple con los requisitos para ser sustentado.
Atentamente,

Dr. Gerardo Mauricio Sarria Montemiranda

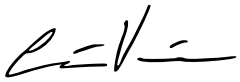
Santiago de Cali, 24 de Julio de 2024.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "QuickMoto: Diseño de arquitectura para aplicación móvil de transporte de mercancía en Motocicletas" con el fin de cumplir con los requisitos exigidos por la Universidad y para que sea sometido a revisión del jurado y cumpla su aprobación, para conseguir posteriormente el título de Magister en Ingeniería de Software.

Atentamente,



Edison Antonio Vasquez Burbano
Código: 8955310

FICHA RESUMEN
TRABAJO DE GRADO DE MAESTRÍA

Título: QuickMoto: Diseño de arquitectura para aplicación móvil de transporte de mercancía en Motocicletas

1. **Énfasis:** Ingeniería de Software
2. **Tipo de proyecto:** Aplicado
3. **Área de trabajo:** Arquitectura de software
4. **Estudiante:** Edison Antonio Vasquez Burbano
5. **Correo electrónico:** evasquez16@javerianacali.edu.co
6. **Dirección y teléfono:** Calle 70BN # 8A - 81, Popayán - Cauca, 3225132883
7. **Director:** Gerardo Mauricio Sarria Montemiranda
8. **Vinculación del director:** Planta
9. **Correo electrónico del director:** gsarria@javerianacali.edu.co
10. **Palabras clave:** Diseño, Arquitectura, Software, Transporte, Aplicación, Móvil
11. **ODS que aplica el proyecto (Agenda 2030):** Ciudades y comunidades sostenibles
12. **Fecha de inicio:** 01/02/2024
13. **Resumen:** Durante la ejecución del proyecto de diseño de arquitectura para la aplicación móvil de gestión de entregas en motocicletas, se avanzó significativamente en la definición y estructuración de los componentes esenciales. Se focalizó en la selección de requisitos, tecnologías adecuadas y la integración de servicios esenciales. Se dio especial atención a la seguridad de datos y al rendimiento óptimo en entornos urbanos congestionados.

    Se logró diseñar mecanismos para la gestión de errores por medio de la metodología ATAM. Aunque el enfoque fue en el diseño y no en el desarrollo directo, se sentaron bases sólidas para futuras fases de implementación. La colaboración interdisciplinaria fue fundamental, garantizando la viabilidad técnica y funcional.

    Este proyecto representa un avance significativo en la transformación de la industria de entregas en motocicletas, ofreciendo soluciones innovadoras y eficientes para los desafíos urbanos, sentando las bases para un desarrollo futuro exitoso.

Resumen

El servicio de transporte informal en motocicleta emergió como una opción ágil y eficiente para el transporte de encomiendas en entornos urbanos caracterizados por la congestión vehicular y la falta de movilidad. Sin embargo, esta modalidad enfrentaba desafíos significativos que afectaban la seguridad, la legalidad y la confiabilidad del servicio. Los usuarios a menudo carecían de información sobre los conductores y la legalidad de las motocicletas utilizadas, lo que generaba preocupaciones legítimas sobre su seguridad.

Este proyecto se propuso abordar estos desafíos mediante el diseño de una arquitectura de aplicación móvil, además, espera mejorar la calidad y seguridad del servicio de transporte en motocicleta, y contribuir al desarrollo sostenible del sistema de transporte en ciudades en constante crecimiento y urbanización.

Palabras Clave: Arquitectura de software; Ingeniería de software; Aplicación móvil.

Abstract

The informal motorcycle transport service has emerged as an agile and efficient option for parcel delivery in urban environments characterized by traffic congestion and a lack of mobility. However, this modality faces significant challenges that impact the safety, legality, and reliability of the service. Users often lack information about the drivers and the legality of the motorcycles used, raising legitimate concerns about their safety.

This project aimed to address these challenges by designing a secure mobile application architecture. In addition, this app will enhance the quality and safety of motorcycle transportation services, it aimed to contribute to the sustainable development of the transportation system in rapidly growing and urbanizing cities.

Keywords: Software architecture; Software Engineering; Mobile application.

Índice general

Capítulo 1

Introducción

1.1. Presentación del tema

Durante el auge del mercado de transporte de mercancías, surgió la necesidad imperante de diseñar una arquitectura robusta para una aplicación móvil dedicada a la gestión de entregas. Este proyecto se enfrentó a desafíos que trascendieron la mera selección de tecnologías y la definición de requisitos, ya que implicaba aspectos críticos como la congestión vehicular, la seguridad urbana y la eficiencia en las entregas. Se requería una comprensión profunda de los requisitos funcionales y no funcionales para abordar aspectos de escalabilidad, rendimiento y seguridad.

El proyecto se enfocó en la elección de tecnologías y la integración de servicios externos como mapas y métodos de pago, además de abordar la seguridad de datos y la privacidad de los usuarios. Estrategias sólidas fueron desarrolladas para enfrentar desafíos como la autenticación, el cifrado y la prevención de riesgos, considerando la lamentable propensión de las motocicletas a los robos en entornos urbanos.Londoño et al. (2021)

La detección y gestión de errores, junto con la tolerancia a fallos, se consideraron cruciales para mantener una disponibilidad de servicio ininterrumpida, aún así, el proyecto se enfrentó a desafíos más amplios, como la congestión vehicular y el caos en las vías, que impactaban la movilidad urbana. Aarón et al. (2019)

Para abordar los desafíos del diseño arquitectónico, se empleó la metodología de Arquitectura de Diseño Dirigido (ADD). Esta metodología proporcionó un marco estructurado para el desarrollo de la arquitectura, desde la identificación de los requisitos hasta la creación de vistas y la evaluación de la arquitectura resultante. A través de ADD, se pudo gestionar de manera efectiva la complejidad del diseño, asegurando la alineación con los objetivos del proyecto y la satisfacción de los stakeholders.

1.2. Objetivos del proyecto

1.2.1. Objetivo General

Diseñar una arquitectura de aplicación móvil segura y efectiva para el transporte en motocicleta de bienes y productos.

1.2.2. Objetivos específicos

- Identificar los requisitos funcionales y no funcionales de la aplicación.
- Establecer y documentar los requerimientos arquitectónicos de la aplicación.
- Definir estructura arquitectónica.
- Determinar los componentes específicos de la arquitectura.
- Realizar pruebas y validaciones para garantizar la estabilidad y la viabilidad de la arquitectura.

1.3. Metodología de la investigación

La metodología de investigación empleada en este proyecto se fundamentó en un enfoque mixto, combinando tanto métodos cualitativos como cuantitativos para obtener una comprensión completa y profunda del fenómeno estudiado. En primer lugar, se llevó a cabo una revisión exhaustiva de la literatura existente sobre el transporte de mercancías en motocicletas, la gestión de entregas y la arquitectura de aplicaciones móviles.

Además, se realizaron entrevistas semiestructuradas con expertos en el campo del transporte urbano, la logística y potenciales usuarios finales, con el fin de recabar información cualitativa sobre las necesidades, desafíos y oportunidades en el ámbito de estudio. Estas entrevistas permitieron obtener perspectivas enriquecedoras y contextualizar los hallazgos obtenidos a través de la revisión bibliográfica.

Una vez recopilados y analizados los datos, se procedió a la fase de diseño arquitectónico, basada en los hallazgos obtenidos durante la investigación. Se utilizaron técnicas de diseño centrado en el usuario y se involucró activamente a los stakeholders en el proceso de toma de decisiones, asegurando así que la arquitectura propuesta fuera relevante, efectiva y aceptada por los usuarios finales.

Posteriormente, se empleó la metodología ADD (Arquitectura de Diseño Dirigido) para el diseño de la arquitectura de la aplicación móvil. Este enfoque proporcionó una estructura sistemática y orientada a objetivos para el desarrollo de la arquitectura, desde la identificación de requisitos hasta la definición de componentes y la validación de la solución propuesta.

Además, se realizó una evaluación detallada de la arquitectura propuesta utilizando la metodología ATAM (Metodología de Evaluación de la Arquitectura Técnica). Esta metodología se centró en analizar los compromisos arquitectónicos y evaluar cómo las decisiones de diseño impactaban en los atributos de calidad esenciales, como la escalabilidad, la seguridad y la mantenibilidad. Se

definieron y ejecutaron escenarios de prueba representativos para garantizar que la arquitectura cumpliera con los requisitos técnicos y se alineara con las necesidades del negocio.

En resumen, la metodología de investigación empleada en este proyecto combinó métodos cualitativos y cuantitativos para obtener una comprensión integral del fenómeno estudiado y fundamentar el diseño de la arquitectura. Esta aproximación metodológica rigurosa y multidisciplinaria garantizó la validez y fiabilidad de los resultados obtenidos, así como la relevancia y utilidad práctica de la solución propuesta.

1.4. Relevancia e impacto

La relevancia e impacto de este proyecto de diseño de arquitectura es significativa en varios aspectos:

- **Mejora de la seguridad y legalidad**: Al proporcionar una plataforma que garantiza la autenticación de conductores, el seguimiento de la legalidad de las motocicletas utilizadas y la implementación de medidas de seguridad de datos, se contribuye a mejorar la confianza y la seguridad tanto de los usuarios como de los operadores del servicio. Beecroft (2019)

- **Optimización del transporte urbano**: En un contexto de creciente congestión vehicular y caos en las vías, esta aplicación busca optimizar el transporte urbano al ofrecer una alternativa eficiente y ágil para la entrega de mercancías. Al mejorar la gestión de entregas, se puede reducir la congestión en las calles y mejorar la movilidad en entornos urbanos.

- **Contribución al desarrollo sostenible**: Al fomentar un sistema de transporte más eficiente y seguro, el proyecto contribuye al desarrollo sostenible de las ciudades en constante crecimiento y urbanización. La reducción de la congestión vehicular y la optimización de los recursos de transporte tienen un impacto positivo en la calidad del aire y en la calidad de vida de los residentes urbanos.

- **Innovación tecnológica**: La aplicación propuesta a raíz de la arquitectura diseñada implica la integración de tecnologías avanzadas para garantizar la seguridad, la eficiencia y la escalabilidad del servicio. La selección y aplicación de tecnologías adecuadas no solo mejora la experiencia del usuario, sino que también establece un estándar para futuras innovaciones en el sector del transporte urbano. Butler et al. (2020)

En resumen, el proyecto no solo aborda desafíos críticos en el transporte urbano y la seguridad del servicio de entrega en motocicletas, sino que también establece un precedente para la adopción de soluciones tecnológicas innovadoras y sostenibles en entornos urbanos en todo el mundo. Su impacto se extiende más allá de la esfera tecnológica, contribuyendo a la mejora del bienestar y la calidad de vida de la comunidad urbana.

1.5. Estructura de la tesis

En esta sección, se proporciona una visión general de la estructura y organización de la presente tesis. Se detalla la secuencia de capítulos y secciones que conforman el documento, brindando al lector una guía clara sobre cómo está organizada la investigación y qué temas se abordan en cada parte. Esta estructura facilita la navegación por el contenido y permite una comprensión sistemática de los objetivos, el desarrollo y las conclusiones del estudio realizado.

1. **Introducción:** En este capítulo se presenta el tema de investigación, destacando su relevancia e impacto en el contexto del transporte de mercancías en motocicletas. Se establecen los objetivos del proyecto, tanto el objetivo general como los objetivos específicos. Se describe la metodología de la investigación utilizada y se justifica la importancia de este estudio. Además, se proporciona una visión general de la estructura de la tesis y se delimita el alcance del proyecto.

2. **Marco de referencia:** Este capítulo se divide en varias secciones. En primer lugar, se presenta el contexto específico del problema de investigación, abordando aspectos relevantes del transporte de mercancías en motocicletas. Luego, se desarrolla el marco teórico, que incluye conceptos y teorías relacionadas con el tema de estudio. Posteriormente, se realiza una revisión de la literatura existente, identificando brechas y áreas de oportunidad para la investigación.

3. **Desarrollo del proyecto:** Este capítulo se centra en el proceso de desarrollo del proyecto. Se inicia con la identificación de los requisitos funcionales y no funcionales de la aplicación móvil, incluyendo un análisis detallado de los procesos de transporte. Luego, se establecen los requerimientos arquitectónicos de la aplicación, definiendo estándares, directrices y restricciones. Se describe el diseño de la estructura general en capas y se detallan las tecnologías utilizadas, así como la estructura de la base de datos. Se introduce el modelo C4 para la arquitectura y se detallan los componentes principales de la aplicación.

4. **Evaluación:** En este capítulo se lleva a cabo una evaluación de la solución propuesta, analizando su desempeño y cumplimiento de los objetivos establecidos. Con ayuda el método ATAM se incluyen pruebas de rendimiento, seguridad, escalabilidad, disponibilidad y adaptabilidad para validar la efectividad de la arquitectura diseñada.

5. **Conclusiones:** En este último capítulo, se presentan las conclusiones obtenidas a partir del desarrollo del proyecto. Se resumen los hallazgos principales, se discuten las implicaciones prácticas de la investigación y se proponen recomendaciones para futuros trabajos en el área. Se destaca la contribución del estudio al conocimiento existente y se reflexiona sobre los desafíos y aprendizajes durante el proceso de investigación y desarrollo.

6. **Bibliografía:** Se incluye una lista de todas las fuentes utilizadas durante la investigación, siguiendo el formato de citación correspondiente. Esto permite a los lectores acceder a las fuentes consultadas y profundizar en el tema de estudio si así lo desean.

1.6. Alcance

Durante la ejecución de nuestro proyecto, nos enfrentamos al desafío de diseñar la arquitectura de una aplicación móvil dirigida al servicio de transporte en motocicletas, en un contexto marcado por el crecimiento continuo del mercado y las complejidades urbanas asociadas. Conscientes de la importancia crítica de este proyecto en la optimización de la logística de entregas y la seguridad vial en entornos urbanos, nos propusimos no solo desarrollar una arquitectura sólida desde el punto de vista técnico, sino también una que abordara de manera integral los desafíos sociales y ambientales que enfrentan nuestras ciudades.

Desde el inicio del proyecto, establecimos objetivos claros que guiaron nuestra labor hacia la creación de una arquitectura que garantizara la seguridad, eficiencia y confiabilidad del servicio. Para ello, nos sumergimos en la definición de medidas avanzadas de seguridad, como la autenticación multifactor y la encriptación de datos, conscientes de la sensibilidad de la información manejada en una aplicación de transporte. Estas medidas no solo buscaban proteger la privacidad de los usuarios, sino también salvaguardar la integridad de los datos y prevenir posibles incidentes de seguridad que pudieran comprometer la confianza en nuestro servicio.

Además de la seguridad, nos enfocamos en mejorar la movilidad urbana mediante la implementación de tecnologías y estrategias innovadoras. Exploramos soluciones como algoritmos de enrutamiento para optimizar las rutas de entrega, especialmente en períodos de alta congestión vehicular, contribuyendo así a reducir los tiempos de traslado y minimizar el impacto ambiental asociado al transporte de mercancías. Esta visión holística de la arquitectura nos llevó a considerar no solo aspectos técnicos, sino también sociales y ambientales, reconociendo el papel fundamental que desempeñan las soluciones tecnológicas en la construcción de ciudades más sostenibles y habitables.

Durante el proceso de diseño, no solo nos centramos en la definición de la arquitectura técnica, sino también en su validación y evaluación a través del método ATAM. Este enfoque nos permitió analizar en profundidad los compromisos arquitectónicos y garantizar que nuestras decisiones de diseño estuvieran alineadas con los requisitos del negocio y las expectativas de los stakeholders. La colaboración activa con los usuarios y las partes interesadas fue fundamental para comprender sus necesidades y expectativas, asegurando así que la arquitectura propuesta no solo fuera sólida desde el punto de vista técnico, sino también viable y deseable desde una perspectiva empresarial y de usuario.

1.7. Justificación

Nuestra iniciativa surgió de la percepción de un potencial aún no explotado en el ámbito de la movilidad urbana: las motocicletas como una opción viable para el transporte compartido en entornos urbanos densos. Durante la ejecución del proyecto, nos dimos cuenta de las ventajas significativas que estas ofrecen en términos de agilidad, velocidad y eficiencia en el consumo de combustible, características que las convierten en una alternativa idónea para desplazamientos rápidos en áreas urbanas congestionadas. Law et al. (2023)

Al fusionar la versatilidad de las motocicletas con la conveniencia de las aplicaciones móviles, nuestro objetivo era proporcionar una solución de movilidad adaptable a las cambiantes necesidades de los habitantes urbanos. La concepción de una arquitectura innovadora para la plataforma fue fundamental en nuestro planteamiento, ya que esperábamos que su implementación transformara la gestión del transporte de mercancías dentro de la ciudad, tanto para empresas como para individuos.

A pesar de los avances en el mercado de aplicaciones de transporte compartido, notamos que el enfoque se centraba principalmente en automóviles y vehículos más grandes, ignorando el potencial de las motocicletas. Observamos una oportunidad en este vacío, ya que las motocicletas ofrecen la capacidad de navegar eficazmente el tráfico y proporcionar experiencias de viaje ágiles en áreas urbanas densamente pobladas.

Durante la ejecución del proyecto, nos enfrentamos al desafío de desarrollar una arquitectura sólida que respaldara la creación y operación de una aplicación de transporte en motocicletas para mercancías. Nos basamos en el análisis previo que señalaba la evolución arquitectónica en otros sectores, como la industria del automóvil, para establecer una base técnica sólida.

Aunque la idea de negocio era atractiva y respondía a una necesidad evidente en el mercado, nos encontramos con obstáculos significativos debido a la falta de una estructura tecnológica sólida. La arquitectura que buscábamos desarrollar tenía que abordar diversas cuestiones cruciales, como la asignación eficiente de solicitudes a conductores disponibles, la seguridad de los usuarios durante los viajes, la protección de datos personales y la escalabilidad para manejar un crecimiento constante en usuarios y transacciones. Puliafito et al. (2021)

En resumen, nuestra misión durante la ejecución del proyecto fue convertir una visión ambiciosa en una realidad tecnológica funcional. Como se plantea por parte de Andrade et al. (2020), a través del diseño de una arquitectura sólida, buscamos establecer una plataforma robusta y confiable que redefiniera la movilidad urbana, ofreciendo a los usuarios una alternativa ágil y eficiente para sus desplazamientos diarios y el envío de pequeñas encomiendas.

Durante la investigación y de acuerdo con el estudio llevado a cabo por Aarón et al. (2019), identificamos la congestión vehicular como un problema crítico en nuestras ciudades, especialmente causada por vehículos más grandes y pesados. Nuestro enfoque en la optimización de rutas y la planificación inteligente se destacó como una estrategia clave para mejorar la circulación vehicular y reducir la congestión en entornos urbanos densamente poblados.

Capítulo 2

Marco de referencia

2.1. Contexto específico

Se enmarca en un entorno urbano caracterizado por una serie de factores relevantes:

- **Creciente demanda de servicios de entrega**: En un mundo cada vez más orientado hacia la conveniencia y la rapidez, la demanda de servicios de entrega de mercancías ha experimentado un notable aumento. Esta demanda se ve especialmente marcada en entornos urbanos densamente poblados, donde la movilidad y la logística son fundamentales para la vida diaria y el funcionamiento de los negocios. Samora Marchele Obudho (2020)
- **Desafíos de movilidad urbana**: Las ciudades enfrentan desafíos significativos en términos de congestión vehicular, limitaciones de infraestructura y crecimiento urbano desordenado. Estos desafíos afectan no solo la eficiencia del transporte de personas y mercancías, sino también la calidad de vida de los residentes y el impacto ambiental de las actividades urbanas.
- **Proliferación del transporte en motocicleta**: En muchos entornos urbanos, las motocicletas han surgido como una alternativa popular y ágil para el transporte de mercancías. Su capacidad para navegar por el tráfico denso y acceder a áreas de difícil acceso las convierte en una opción atractiva para la entrega de paquetes y encomiendas.
- **Preocupaciones de seguridad y legalidad**: A pesar de su popularidad, el transporte en motocicleta enfrenta desafíos en términos de seguridad y legalidad. Los usuarios y los operadores del servicio a menudo carecen de información sobre la legalidad de las motocicletas y la confiabilidad de los conductores, lo que genera preocupaciones legítimas sobre la seguridad de las entregas y la integridad de los datos del cliente.

2.2. Marco teórico

Se exploraron diversos aspectos que contextualizan la problemática y las soluciones relacionadas con la movilidad urbana y la tecnología aplicada al transporte de mercancías en motocicletas:

- **Movilidad urbana y desafíos actuales**: Se analizó cómo la movilidad urbana ha emergido como uno de los principales desafíos contemporáneos, con proyecciones que indican un aumento sustancial en el número de viajes en vehículos a nivel mundial (Law et al. (2023)). Este contexto reveló un crecimiento constante en la dependencia de los automóviles, planteando

desafíos significativos en términos de congestión del tráfico y emisiones de gases de efecto invernadero. Además, se destacó la accesibilidad económica y la eficiencia de las motocicletas como medio de transporte (Martin et al. (2023)).

- **Tecnología y movilidad**: Se exploró cómo la industria de la movilidad está experimentando una transformación impulsada por avances en la Inteligencia Artificial (IA) y el Big Data (Hoffmann Souza et al. (2021)). Estos avances permiten analizar datos en tiempo real, predecir patrones de tráfico y personalizar la experiencia del usuario, lo que tiene el potencial de revolucionar la movilidad urbana.

- **Diseño de aplicaciones móviles**: Se subrayó la importancia del diseño de aplicaciones móviles en la era tecnológica actual, enfatizando la necesidad de una experiencia de usuario excepcional (Feng and Wei (2019)). Se resaltó la diversidad de plataformas móviles y los desafíos asociados con el diseño adaptativo para garantizar una experiencia uniforme en diferentes dispositivos.

- **Mejoras en la seguridad**: Se abordaron las preocupaciones sobre la seguridad de los datos en la era digital y se exploraron las métricas de privacidad y técnicas de ofuscación como métodos para proteger la información sensible de los usuarios (di Vimercati et al. (2023)). Se destacó la importancia de evaluar la efectividad de las técnicas de anonimización para garantizar la protección adecuada de los datos. Estos aspectos proporcionaron el contexto necesario para comprender la complejidad de la movilidad urbana y la importancia de desarrollar soluciones tecnológicas innovadoras y seguras para abordar los desafíos relacionados con el transporte de mercancías en motocicletas en entornos urbanos.

2.3. Revisión de la literatura

La gestión eficiente del transporte de mercancías urbanas en motocicletas es un área de creciente interés en el ámbito de la ingeniería de software y la planificación urbana. Para comprender mejor este contexto y diseñar una arquitectura adecuada, se ha realizado una revisión de la literatura pertinente, abarcando teorías, modelos, estudios empíricos y otras fuentes académicas relevantes.

- Law et al. (2023) investigan la propiedad de motocicletas en relación con la movilidad urbana y la desigualdad de ingresos. Su estudio proporciona una perspectiva valiosa sobre la dinámica del transporte en entornos urbanos, destacando la importancia de considerar factores socioeconómicos en el diseño de soluciones de gestión de transporte.

- Martin et al. (2023) examinan las tendencias de digitalización y electrificación en ciudades africanas, centrándose en los servicios de taxi en motocicletas. Su investigación ofrece información relevante sobre las tecnologías emergentes y las necesidades específicas de transporte en contextos urbanos en desarrollo, lo que puede guiar la implementación de una arquitectura adaptable y eficiente.

- Puliafito et al. (2021) exploran el concepto de ciudades inteligentes como sistemas ciberfísicos y discuten los desafíos y tecnologías habilitadoras. Esta perspectiva proporciona una base teórica sólida para el diseño de una arquitectura de gestión de transporte que integre eficazmente sistemas de información y comunicación avanzados para mejorar la eficiencia y la sostenibilidad del transporte urbano.

- Reis et al. (2022) se enfocan en el desarrollo de especificaciones para Docker y Docker Compose, lo que puede ser relevante para la implementación práctica de la arquitectura propuesta, especialmente en lo que respecta a la gestión de contenedores y la orquestación de servicios en entornos de transporte de mercancías.

- Hoffmann Souza et al. (2021) presentan un método de identificación de características para explicar anomalías en el monitoreo de condiciones. Aunque su enfoque difiere del transporte de mercancías en motocicletas, su metodología puede ser aplicable para identificar y abordar problemas en la gestión de transporte urbano.

- Hoffmann Souza et al. (2021) presentan un modelo espacio-temporal sobre el comportamiento criminal en Medellín durante la crisis de la pandemia. Aunque este estudio ofrece una visión interesante sobre la dinámica urbana, su enfoque en la criminalidad no está directamente relacionado con la gestión de transporte de mercancías en motocicletas.

- Love and Vyas (2022) discuten sobre Kubernetes, un sistema de orquestación de contenedores, que si bien es relevante en el ámbito de la tecnología, no aborda directamente el diseño de una arquitectura para la gestión de transporte de mercancías en motocicletas.

- Maratkar P (2021) examina React JS, una biblioteca de JavaScript para construir interfaces de usuario, pero su enfoque en el desarrollo web no está directamente relacionado con la gestión de transporte urbano.

- R (2011) presenta paradigmas y benchmarks de Node.js, lo cual es relevante en el ámbito de la tecnología de desarrollo de software, pero no está directamente relacionado con la gestión de transporte de mercancías en motocicletas.

- Sai K (2017) Realiza un estudio comparativo entre datos estructurados y no estructurados utilizando MongoDB. Aunque el enfoque del estudio no está directamente relacionado con la gestión de transporte urbano, su relevancia radica en la aplicación potencial de los hallazgos en nuestro proyecto. La comprensión de las diferencias entre estos tipos de datos y su almacenamiento en MongoDB proporcionó información valiosa para optimizar la gestión de datos.

- Samora Marchele Obudho (2020) discuten sobre el uso de motocicletas en el transporte rural, lo cual proporciona información valiosa sobre la movilidad en áreas menos desarrolladas, pero su enfoque en el transporte rural difiere del contexto urbano que se aborda en el proyecto.

2.4. Identificación de brechas

La revisión de la literatura revela varias brechas significativas que esta investigación busca abordar. A pesar de la amplia gama de estudios relacionados con la movilidad urbana, la gestión de transporte de mercancías en motocicletas ha recibido una atención limitada en la literatura existente. A continuación, se identifican las principales brechas:

- **Escasez de enfoques específicos:** Aunque existen numerosos estudios sobre tecnologías de transporte urbano y gestión de flotas, hay una falta de enfoques específicos para la gestión eficiente de la entrega de mercancías utilizando motocicletas en entornos urbanos. La literatura existente tiende a centrarse más en los vehículos de carga más grandes o en soluciones generales de logística, dejando un vacío en términos de estrategias específicas para las motocicletas.
- **Desafíos tecnológicos y logísticos no abordados:** Si bien hay investigaciones sobre tecnologías emergentes y soluciones de gestión de transporte, pocas abordan los desafíos tecnológicos y logísticos únicos asociados con el transporte de mercancías en motocicletas. Estos desafíos incluyen la optimización de rutas, la seguridad del envío y la integración eficiente con sistemas de gestión de flotas existentes.
- **Impacto socioeconómico y ambiental subestimado:** La literatura revisada proporciona una visión limitada del impacto socioeconómico y ambiental de la gestión de transporte de mercancías en motocicletas en entornos urbanos. Si bien algunos estudios abordan aspectos específicos, como la accesibilidad y la equidad en la distribución de bienes, hay una falta de análisis integrados que aborden el impacto general en la calidad de vida urbana y la sostenibilidad ambiental.
- **Falta de investigación empírica en contextos específicos:** Muchos de los estudios revisados son teóricos o se basan en casos de estudio limitados. Existe una necesidad de investigación empírica más extensa en contextos urbanos específicos, considerando factores como la densidad de tráfico, la infraestructura vial y las regulaciones locales, para informar el diseño de una arquitectura de gestión de transporte de mercancías efectiva y adaptable.

Al abordar estas brechas en la literatura existente, esta investigación aspira a contribuir al desarrollo de soluciones más completas y efectivas para la gestión de transporte de mercancías urbanas en motocicletas, con el objetivo final de mejorar la eficiencia, la sostenibilidad y la accesibilidad en las ciudades modernas.

Capítulo 3

Desarrollo del Proyecto

En el desarrollo del proyecto de diseño de arquitectura, nos sumergimos en un proceso meticuloso y colaborativo destinado a abordar los desafíos clave que enfrenta el sector del transporte de mercancías en entornos urbanos. Conscientes del impacto significativo que esta solución podría tener en la eficiencia operativa, la seguridad vial y la calidad de vida en las ciudades, nuestro enfoque se centró en la creación de una arquitectura sólida y adaptable que no solo satisficiera las necesidades técnicas del proyecto, sino que también respondiera de manera efectiva a las demandas cambiantes de un entorno urbano dinámico y en constante evolución. En este contexto, nuestro equipo se embarcó en un viaje de exploración, innovación y colaboración, donde cada paso nos acercó más a la materialización de una visión compartida: la de proporcionar un servicio de transporte eficiente, seguro y sostenible que contribuya al desarrollo de ciudades más inteligentes y habitables.

3.1. Identificación de requisitos funcionales y no funcionales de la aplicación

Sumergidos en el proceso fundamental de identificar los requisitos funcionales y no funcionales de nuestra arquitectura, nos basamos en los resultados de la encuesta realizada. Los requisitos funcionales delinean las funciones y características específicas que nuestra aplicación debe cumplir para satisfacer las necesidades de nuestros usuarios y alcanzar los objetivos del negocio. Por otro lado, los requisitos no funcionales establecen los criterios que son esenciales para garantizar la eficacia y calidad del sistema, todo ello fundamentado en la retroalimentación obtenida de los usuarios mediante la encuesta realizada.

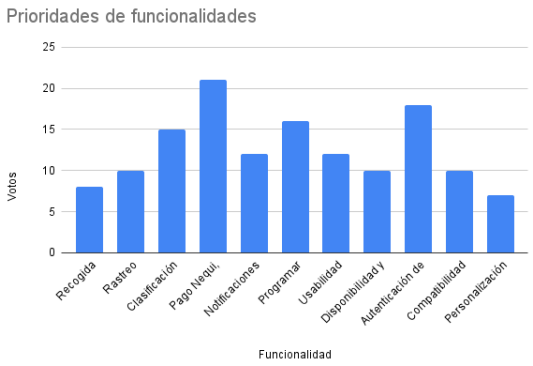


Figura 1: Resumen resultado encuesta de prioridades para usuarios finales

Esta versión enfatiza que los requisitos identificados se basan en los datos concretos recopilados a través de la encuesta, lo que refuerza la relevancia y pertinencia de los mismos para la aplicación.

3.1.1. Análisis de los procesos de transporte

Llevamos a cabo un exhaustivo análisis de los procesos involucrados en el transporte de mercancías en motocicletas. Este análisis incluyó la identificación de los diferentes pasos, desde la recogida de la mercancía hasta la entrega final, así como los posibles desafíos y requisitos específicos asociados con cada etapa del proceso.

3.1.2. Requisitos funcionales

Documentamos detalladamente los requisitos funcionales de la aplicación, centrándonos en las funciones específicas que la aplicación debe cumplir para satisfacer las necesidades de los usuarios. Esto implicó la identificación de características clave, como la gestión de pedidos, la asignación de conductores, el seguimiento en tiempo real de las entregas, entre otros aspectos relevantes para el proceso de transporte.

- Para documentar los requisitos funcionales de nuestra arquitectura, nos apoyamos principalmente en la serie de entrevistas con los diferentes actores involucrados en el proceso de transporte (Solicitantes de servicio de recogida, Motociclistas y Usuarios finales receptores de mercancia). Estas entrevistas nos permitieron comprender a fondo las necesidades y expectativas de los usuarios finales, así como identificar las funciones específicas que la aplicación debería ofrecer como son: **Registro de usuarios, Gestión de pedidos, Asignación de conductores, Seguimiento en tiempo real, Gestión de pagos e Historial de pedidos y estadísticas**. La documentación detallada correspondiente se encuentra disponible en los archivos adjuntos al proyecto. Estos documentos complementan y respaldan la información recopilada durante las entrevistas, al igual que una visión completa de los requisitos funcionales de nuestra arquitectura

- Una vez recopilada esta información, procedimos a elaborar documentos detallados que describían cada uno de los requisitos funcionales identificados. Estos requisitos abarcaron una amplia gama de funcionalidades, desde la capacidad de los usuarios para solicitar y programar entregas, hasta la gestión de la flota de motocicletas.

- Una vez completada la documentación inicial, compartimos el documento con los principales stakeholders para su revisión y retroalimentación. Se llevaron a cabo reuniones de seguimiento para discutir cualquier comentario o sugerencia y asegurar que los requisitos funcionales reflejaran con precisión las necesidades del negocio y los usuarios.

- Finalmente, tras incorporar las modificaciones sugeridas, finalizamos la documentación de los requisitos funcionales, que sirvió como guía fundamental para continuar con el diseño de nuestra arquitectura.

3.1.3. Requisitos no funcionales

Junto con la documentación de los requisitos funcionales, también hemos detallado los requisitos no funcionales en los archivos adjuntos al proyecto. La información contenida en estos documentos complementarios es esencial para garantizar que nuestra arquitectura cumpla con altos estándares de calidad y satisfaga plenamente las expectativas tanto de los usuarios como de las partes interesadas. Los archivos adjuntos proporcionan un análisis y especificaciones técnicas detalladas que respaldan nuestra atención a estos requisitos no funcionales clave como son:

- **Rendimiento del sistema:** Debe ser capaz de manejar un volumen 30 solicitudes simultáneas de manera eficiente, asegurando tiempos de respuesta rápidos incluso en momentos de alta demanda, para garantizar una experiencia fluida para los usuarios y evitar retrasos en la entrega de mercancías.
- **Seguridad de los datos:** Se deben implementar medidas de seguridad robustas para proteger la información confidencial de los usuarios, como datos personales, detalles de pago y detalles de envío. Esto incluye el cifrado de datos, la autenticación de usuarios y la gestión adecuada de accesos y permisos.
- **Escalabilidad:** La arquitectura de la aplicación debe ser escalable para adaptarse al crecimiento futuro del negocio y a un aumento en el número de usuarios y solicitudes de transporte. Esto implica diseñar una infraestructura que pueda expandirse fácilmente para soportar la carga adicional sin comprometer el rendimiento o la calidad del servicio.
- **Disponibilidad:** Debe estar disponible y operativa 24 horas los 7 dìas de la semana, con un tiempo de inactividad mínimo de 2h a la semana planificadas para mantenimiento y actualizaciones. Se deben implementar medidas de redundancia y respaldo para garantizar la continuidad del servicio en caso de fallas o interrupciones inesperadas.
- **Adaptabilidad:** Debe ser adaptable para integrarse con otros sistemas y servicios externos, como sistemas de pago o herramientas de seguimiento de flotas.

3.1.4. Priorización y validación de requisitos con los stakeholders

Llevamos a cabo un proceso de priorización y validación de los requisitos con los stakeholders clave incluyendo a los usuarios finales. Esto nos permitió asegurar que los requisitos identificados estuvieran alineados con las necesidades y expectativas del cliente, así como con los objetivos del proyecto.

- **Requisitos funcionales:**

  1. **Gestión de pedidos:** Es fundamental para la funcionalidad básica de la aplicación, ya que permite a los usuarios solicitar y programar sus entregas. Sin este requisito, la aplicación carecería de su propósito principal y no podría satisfacer las necesidades de los usuarios ni generar ingresos para el negocio.

2. **Seguimiento en tiempo real:** Proporciona a los usuarios visibilidad sobre el progreso de sus entregas y les permite planificar en consecuencia. Esta funcionalidad es crucial para mejorar la experiencia del cliente, ya que aumenta la transparencia y la confianza en el servicio. Además, el seguimiento en tiempo real ayuda a optimizar la eficiencia operativa al permitir una gestión proactiva de las entregas y la resolución rápida de problemas.
3. **Asignación de conductores:** Asegura que los recursos de la flota de motocicletas se utilicen de manera óptima y se minimicen los tiempos de espera para los usuarios. Además, una asignación eficaz de conductores contribuye a la mejora de la productividad y la satisfacción del cliente.
4. **Gestión de pagos:** Es crucial para facilitar transacciones seguras y sin problemas entre los usuarios y el servicio de transporte. Al priorizar este requisito, se garantiza la viabilidad financiera del negocio al facilitar la captura y el procesamiento de pagos de manera eficiente. Además, una gestión efectiva de pagos contribuye a mejorar la experiencia del usuario al ofrecer opciones de pago convenientes y seguras.
5. **Historial de pedidos y estadísticas:** Proporcionan a los usuarios información valiosa sobre sus actividades pasadas y el rendimiento del servicio. Al priorizar este requisito, se ofrece a los usuarios la capacidad de realizar un seguimiento de sus actividades anteriores, lo que les permite planificar futuras entregas de manera más informada. Además, las estadísticas sobre el rendimiento del servicio son fundamentales para la toma de decisiones estratégicas y la mejora continua del negocio.

- **Requisitos no funcionales:**

1. **Seguridad de los datos:** Debe tener la máxima prioridad debido a la sensibilidad de la información manejada por la aplicación, que incluye datos personales, detalles de pago y detalles de envío. La seguridad de los datos es fundamental para proteger la privacidad y la confidencialidad de los usuarios y para garantizar la integridad y la disponibilidad de la información frente a posibles amenazas y ataques cibernéticos.
2. **Disponibilidad:** Es esencial para garantizar que la aplicación esté siempre accesible y operativa para los usuarios. Sin una alta disponibilidad, los usuarios podrían experimentar interrupciones en el servicio, lo que podría afectar negativamente la confianza y la satisfacción del cliente. Por lo tanto, este requisito debe ser una prioridad para garantizar una experiencia de usuario continua y sin interrupciones.
3. **Adaptabilidad:** La capacidad del sistema para adaptarse a cambios y evolucionar con el tiempo es crucial para mantener la relevancia y la competitividad en un entorno empresarial en constante cambio. Si el sistema no puede adaptarse fácilmente a nuevas tecnologías, requisitos del negocio y necesidades de los usuarios, corre el riesgo de volverse obsoleto y perder su utilidad a largo plazo. Por lo tanto, este requisito debe recibir una alta prioridad para garantizar la flexibilidad y la capacidad de respuesta del sistema.

4. **Escalabilidad:** Aunque la escalabilidad es importante para permitir el crecimiento y la expansión del sistema en el futuro, puede ser abordada de manera más efectiva una vez que se haya establecido una base sólida de seguridad, disponibilidad y adaptabilidad. Si bien la escalabilidad es crucial para satisfacer las demandas cambiantes del negocio y el aumento en el número de usuarios, no es tan crítica como la seguridad y la disponibilidad en las etapas iniciales del desarrollo de la aplicación.

3.2. Establecer los requerimientos arquitectónicos de la aplicación

Para llevar a cabo la actividad de establecer los requerimientos arquitectónicos de nuestra aplicación de transporte de mercancías en motocicletas, primero nos reunimos como equipo para identificar de manera exhaustiva los principales requisitos arquitectónicos que guiarán nuestro diseño. Durante esta sesión de lluvia de ideas, discutimos aspectos clave como la escalabilidad del sistema, la seguridad de los datos, la interoperabilidad con otros sistemas y la capacidad de respuesta en tiempo real. Una vez identificados estos requisitos, procedimos a definir los estándares y directrices arquitectónicas que debíamos seguir para garantizar la coherencia y la eficacia en el diseño. Establecimos patrones de diseño, principios de modularidad y reglas de diseño limpio que servirían como base para el desarrollo. Además, documentamos cuidadosamente todas las restricciones y limitaciones que podrían influir en la arquitectura, como los recursos limitados de hardware, las regulaciones de seguridad y las restricciones presupuestarias. Esta documentación detallada nos proporcionó un marco sólido sobre el cual construir y evaluar nuestra arquitectura, asegurándonos de que cumpliéramos con los requisitos del proyecto de manera eficiente y efectiva.

3.2.1. Requisitos arquitectónicos

Hemos identificado y priorizado una serie de requisitos arquitectónicos clave para nuestra aplicación de transporte de mercancías en motocicletas. En esta lista, destacamos los requisitos que consideramos fundamentales para el diseño de nuestra arquitectura.

- **Escalabilidad:** Hemos seleccionado este requisito arquitectónico porque anticipamos un crecimiento significativo en el número de usuarios y transacciones a medida que nuestra aplicación de transporte de mercancías en motocicletas gane popularidad. Es crucial que nuestra arquitectura pueda escalar de manera eficiente para manejar esta demanda creciente sin comprometer el rendimiento del sistema.

- **Interoperabilidad:** Dada la naturaleza interconectada de nuestro ecosistema tecnológico, es fundamental que nuestra aplicación pueda integrarse sin problemas con otros sistemas, como servicios de mapas para la navegación, sistemas de pago para transacciones financieras y sistemas de gestión de inventario para el seguimiento de mercancías. La interoperabilidad garantizará una experiencia completa y sin interrupciones para nuestros usuarios y facilitará la colaboración con socios comerciales.

- **Mantenibilidad:** Hemos priorizado la mantenibilidad en el diseño de nuestra arquitectura porque reconocemos la importancia de poder realizar actualizaciones, correcciones de errores y mejoras de manera eficiente a lo largo del tiempo. Siguiendo principios de diseño limpio y modularidad, aseguramos que nuestro código sea fácil de entender, mantener y actualizar, lo que reducirá los costos operativos y mejorará la estabilidad a largo plazo de nuestro sistema.

- **Fiabilidad:** Dado que nuestra aplicación es fundamental para la logística y el transporte de mercancías, hemos enfatizado la fiabilidad en nuestra arquitectura. Es esencial que el sistema sea altamente disponible y resistente a fallos para garantizar que los usuarios puedan acceder y utilizar la aplicación en todo momento. La fiabilidad también contribuirá a generar confianza entre nuestros usuarios y promover la lealtad a la marca.

- **Eficiencia de recursos:** Consideramos crucial optimizar el uso de recursos de hardware y software en nuestra arquitectura para maximizar el rendimiento y minimizar los costos operativos. Al seleccionar tecnologías y técnicas que sean eficientes en términos de consumo de energía y utilización de servidores, podemos garantizar una operación rentable y sostenible de nuestra aplicación de transporte de mercancías en motocicletas.

3.2.2. Estándares y directrices arquitectónicas

Como parte del proceso de diseño de nuestra arquitectura, hemos establecido estándares y directrices arquitectónicas que servirán como marco de referencia para nuestro equipo de desarrollo. En esta etapa, nos hemos enfocado en definir los principios y prácticas que guiarán la construcción de nuestra arquitectura, asegurando coherencia, consistencia y calidad en todo el proceso.

- **Modularidad y separación de responsabilidades:** Identificamos la necesidad de independencia de funciones. Esto garantiza una mejor organización del código y facilita la escalabilidad y mantenibilidad.

- **API Restful:** Hemos establecido el uso de API RESTful para permitir la comunicación entre los distintos componentes de la aplicación de manera eficiente y estandarizada. Esto facilitará la integración con otros sistemas en el futuro y permitirá un desarrollo ágil y escalable.

- **Microservicios:** Optamos por una arquitectura de microservicios para descomponer la aplicación en servicios independientes que se encargan de tareas específicas. Esto facilita la escalabilidad horizontal, permitiendo escalar cada servicio de manera individual según sea necesario, y mejora la disponibilidad y la tolerancia a fallos.

- **Seguridad en capas:** Hemos definido un enfoque de seguridad en capas, donde establecimos medidas de seguridad en cada nivel de la aplicación, desde la capa de presentación hasta la capa de datos. Utilizamos técnicas como autenticación y autorización, cifrado de datos y protección contra ataques de denegación de servicio para garantizar la seguridad de los datos sensibles y la integridad del sistema.

- **Arquitectura multicapa:** Definimos el uso de una arquitectura multicapa, donde se separa claramente la lógica de negocio, la presentación y el acceso a datos. Esto permite una mayor flexibilidad y facilita la evolución y mantenimiento del sistema a lo largo del tiempo.

Al seguir estos estándares y directrices arquitectónicas, aseguramos que la arquitectura cumpla con los requisitos funcionales y no funcionales establecidos, garantizando un sistema robusto, seguro, escalable y eficiente para la gestión de transporte de mercancías en motocicletas.

3.2.3. Restricciones y limitaciones

Hemos identificado una serie de restricciones y limitaciones que afectan directamente a la arquitectura que estamos construyendo. Estas consideraciones son cruciales para garantizar que nuestra solución sea eficiente, segura y viable en el contexto específico en el que operamos.

- **Cobertura de red:** Se refiere a las áreas donde la disponibilidad de redes móviles es limitada o inexistente, como zonas remotas o con baja cobertura. Esto puede afectar la capacidad de los conductores y usuarios de acceder a la aplicación y utilizar sus funciones, lo que puede resultar en retrasos en las entregas, dificultades de comunicación y una experiencia de usuario deficiente.

- **Disponibilidad de GPS:** La dependencia del sistema de posicionamiento global (GPS) para el seguimiento de las mercancías y la navegación implica que cualquier limitación en la disponibilidad o precisión de los datos GPS puede afectar la eficiencia y precisión del servicio. En áreas donde la señal GPS es débil o inestable, la capacidad de rastrear envíos y dirigir a los conductores puede verse comprometida, lo que afecta la calidad del servicio y la satisfacción del cliente.

- **Disponibilidad de datos:** Esta restricción se refiere a las limitaciones en la disponibilidad y precisión de los datos relacionados con rutas, tráfico y condiciones climáticas. La falta de datos actualizados y precisos puede conducir a decisiones erróneas en la planificación de rutas, lo que resulta en retrasos en las entregas, pérdida de eficiencia operativa y una experiencia deficiente para los usuarios.

- **Costo de implementación:** Las restricciones presupuestarias limitan la capacidad de adquirir hardware o contratar servicios adicionales necesarios para la implementación y operación de la aplicación. Esto puede afectar la calidad y el alcance de la infraestructura tecnológica utilizada, así como la capacidad de ofrecer características y servicios adicionales que mejoren la experiencia del usuario y la eficiencia operativa.

- **Cumplimiento legal:** Esta restricción se refiere a los requisitos legales y normativos que deben cumplirse en todas las operaciones y transacciones relacionadas con el transporte de mercancías en motocicletas. El incumplimiento de estas regulaciones puede resultar en multas, sanciones legales y pérdida de reputación para la empresa, lo que resalta la importancia de garantizar el cumplimiento normativo en todas las etapas del servicio.

- **Mantenimiento y soporte:** Se refiere a los requisitos de mantenimiento y soporte a largo plazo necesarios para garantizar el funcionamiento continuo de la arquitectura implementada. Esto incluye actividades como actualizaciones de software, resolución de problemas técnicos y soporte al cliente, que son fundamentales para mantener la calidad y la fiabilidad del servicio a lo largo del tiempo.
- **Requisitos de seguridad vial:** Estos requisitos se refieren al cumplimiento de normativas de seguridad vial para conductores y usuarios. Esto implica asegurarse de que la aplicación promueva prácticas seguras de conducción y cumpla con las regulaciones relacionadas con el uso de motocicletas para el transporte de mercancías, lo que contribuye a la seguridad y protección de todas las partes involucradas en el servicio.
- **Privacidad del usuario:** Esta restricción implica garantizar la privacidad y seguridad de la información personal de los usuarios almacenada y procesada por la aplicación. Es fundamental implementar medidas de seguridad robustas para proteger los datos confidenciales de los usuarios contra accesos no autorizados, pérdidas o filtraciones, lo que ayuda a generar confianza y fomentar la adopción y retención de usuarios en la plataforma.

3.3. Diseño de estructura general en capas.

Durante la definición de la arquitectura de nuestra aplicación, llevamos a cabo un proceso detallado para identificar las diferentes capas que serían necesarias para construir una solución robusta y escalable.

- **Presentación:** Elegimos incluir esta capa para separar claramente la lógica de presentación de la lógica de negocio y los datos subyacentes. Esto nos permitirá crear una interfaz de usuario intuitiva y amigable para los usuarios finales, motociclistas y administradores, facilitando así la interacción con la aplicación y mejorando la experiencia del usuario en general. Optamos por seleccionar React para la capa de presentación debido a su flexibilidad y capacidad para construir interfaces de usuario interactivas y receptivas. Esta elección nos permite desarrollar una interfaz de usuario moderna y altamente modular que mejora la experiencia del usuario final.
- **Lógica de negocio:** Esta capa es fundamental para encapsular toda la lógica empresarial de nuestra aplicación. Decidimos separarla de la capa de presentación para garantizar un diseño modular y facilitar futuras modificaciones o mejoras en la lógica de negocio sin afectar la interfaz de usuario. Aquí reside la esencia de nuestra aplicación, donde se gestionan los procesos clave como la asignación de conductores, el seguimiento de entregas y la gestión de pedidos. Elegimos Node.js para la capa de lógica de negocio debido a su eficiencia y escalabilidad para manejar la lógica empresarial de nuestra aplicación.
- **Acceso a datos:** Optamos por incluir una capa de acceso a datos dedicada para gestionar todas las operaciones de lectura y escritura en la base de datos. Separar esta funcionalidad en

una capa independiente nos permitirá mantener un código más limpio y modular, y facilitará la gestión y el mantenimiento de la base de datos a medida que la aplicación crezca en complejidad y volumen de datos. Decidimos seleccionar MongoDB como nuestra base de datos debido a su flexibilidad y capacidad para manejar datos no estructurados y semi-estructurados.

- **Servicio:** Esta capa proporciona una abstracción adicional entre la lógica de negocio y las operaciones subyacentes, lo que nos permite encapsular funcionalidades compartidas y reutilizables en forma de servicios. Decidimos incluir esta capa para promover la reutilización de código y mejorar la cohesión y la modularidad de nuestra arquitectura, lo que nos facilitará la implementación de nuevas características y la corrección de errores en el futuro.

- **Integración externa:** Finalmente, consideramos esencial incluir una capa dedicada a la integración con sistemas externos o servicios de terceros. Esta capa nos permite separar claramente las preocupaciones relacionadas con la integración externa de la lógica interna de la aplicación, lo que facilita el mantenimiento y la escalabilidad de la aplicación a medida que se agregan o modifican las integraciones externas.

3.4. Tecnologías

3.4.1. Aplicación:

- **ReactJS:** En nuestra arquitectura, hemos decidido utilizar React como framework principal para el desarrollo del frontend de la aplicación. Nos sentimos atraídos por la eficiencia y la flexibilidad que ofrece React, así como por su gran comunidad de desarrollo. Con React, planeamos estructurar la interfaz de usuario de manera modular utilizando componentes reutilizables, lo que nos permitirá crear una experiencia de usuario fluida e interactiva. Estamos seguros de que React nos ayudará a construir un frontend robusto y altamente adaptable para nuestra futura aplicación. Maratkar P (2021)

- **Node.js:** Para el desarrollo del backend de nuestra aplicación, hemos elegido Node.js como nuestra plataforma principal. Estamos convencidos de que Node.js es una excelente opción debido a su capacidad para manejar operaciones de entrada/salida de manera eficiente y su escalabilidad inherente. Utilizaremos Node.js para construir la lógica de negocio de nuestra aplicación, gestionar las solicitudes HTTP y proporcionar servicios web RESTful. Creemos que la modularidad y la vasta cantidad de paquetes disponibles en npm nos permitirán construir un backend flexible y altamente funcional para nuestra futura aplicación. R (2011)

- **MongoDB:** En cuanto a la base de datos, hemos optado por MongoDB debido a su flexibilidad y escalabilidad. Creemos que MongoDB es una excelente opción para nuestra aplicación, ya que nos permite almacenar y recuperar datos de manera eficiente en formato JSON. Utilizaremos MongoDB para almacenar todos los datos relacionados con usuarios, pedidos, conductores, vehículos y otras entidades relevantes para nuestra aplicación. Estamos seguros

de que la integración de MongoDB con Node.js será fluida y nos permitirá construir una base de datos robusta y altamente adaptable para nuestra aplicación. Sai K (2017)

3.4.2. Despliegue:

- **Docker y Docker compose:** Hemos decidido utilizar Docker y Docker Compose para contenerizar nuestra aplicación debido a su capacidad para proporcionar un entorno de desarrollo y despliegue consistente y portátil. El costo de adoptar Docker se ve compensado por los beneficios de tener una infraestructura estandarizada y reproducible, lo que simplifica enormemente el proceso de despliegue y aumenta la consistencia entre los entornos de desarrollo, prueba y producción. Además, Docker Compose facilita la gestión de múltiples contenedores como un solo servicio, lo que reduce la complejidad y los costos operativos asociados con la administración de la infraestructura de contenedores. Reis et al. (2022)

- **AWS (Amazon Web Services):** Elegimos AWS como nuestro proveedor de servicios en la nube debido a su amplia gama de servicios gestionados y su escalabilidad inherente. Aunque AWS puede tener tarifas variables según el uso, los beneficios de utilizar servicios gestionados como Amazon EC2, Amazon RDS y Amazon ECS o Amazon EKS superan los costos asociados. AWS ofrece una infraestructura globalmente distribuida con una alta disponibilidad y durabilidad, así como una amplia variedad de servicios que nos permiten escalar y gestionar nuestra aplicación de manera eficiente. Además, AWS ofrece una facturación basada en el uso, lo que nos permite pagar solo por los recursos que consumimos, reduciendo así los costos operativos y mejorando la rentabilidad de nuestra aplicación. Bundela et al. (2022)

- **GitHub:** Para implementar una estrategia de CI/CD, hemos elegido GitHub debido a su integración nativa y su capacidad para automatizar el proceso de construcción, prueba y despliegue de nuestra aplicación de manera eficiente. Aunque puede requerir cierta configuración inicial, los beneficios de tener un flujo de trabajo de desarrollo automatizado y una entrega continua de nuevas características y correcciones de errores superan con creces estos costos. GitHub nos permite mejorar la calidad del software, acelerar el tiempo de comercialización y reducir los riesgos asociados con los despliegues manuales, lo que nos ayuda a maximizar el valor entregado a nuestros usuarios finales y optimizar la rentabilidad de nuestra aplicación. Cowell et al. (2023)

- **Prometheus y Grafana:** Para supervisar y monitorear el rendimiento y la salud de nuestra aplicación en producción, hemos seleccionado Prometheus y Grafana debido a su capacidad para recopilar, almacenar y visualizar métricas y registros de manera eficiente y escalable. Aunque puede requerir cierta configuración y mantenimiento, los beneficios de tener una solución de monitoreo completa y personalizable superan estos costos. Prometheus y Grafana nos permiten detectar y resolver problemas de rendimiento y disponibilidad de manera proactiva, lo que reduce el tiempo de inactividad y mejora la experiencia del usuario. Además, la integración entre Prometheus y Grafana nos proporciona una visibilidad completa sobre el

estado de nuestra aplicación, lo que nos ayuda a optimizar su rendimiento y rentabilidad a largo plazo. Abirami et al. (2023)

- **AWS Elastic Load Balancer:** Para garantizar la disponibilidad y la escalabilidad de nuestra aplicación en producción, hemos decidido utilizar AWS Elastic Load Balancer (ELB) como nuestro balanceador de carga principal. Aunque puede tener costos asociados con el uso de instancias EC2 y el tráfico de red, los beneficios de tener un balanceador de carga altamente disponible y escalable superan estos costos. AWS ELB nos permite distribuir el tráfico de manera uniforme entre múltiples instancias de nuestra aplicación, lo que mejora la fiabilidad y el rendimiento de nuestra aplicación. Además, AWS ELB ofrece características avanzadas, como la detección automática de instancias saludables y el escalado automático, que nos permiten gestionar eficientemente la carga de trabajo y garantizar una experiencia de usuario óptima en todo momento. Bundela et al. (2022)

- **HashiCorp Vault:** Para gestionar de forma segura y centralizada las credenciales y otros datos sensibles de nuestra aplicación en producción, hemos seleccionado HashiCorp Vault como nuestra solución de gestión de secretos. Aunque puede requerir cierta configuración y administración, los beneficios de tener un almacén de datos altamente seguro y encriptado superan con creces estos costos. HashiCorp Vault nos permite almacenar y distribuir secretos de forma segura, lo que reduce el riesgo de exposición de datos confidenciales y mejora la seguridad de nuestra aplicación. Además, HashiCorp Vault ofrece características avanzadas, como la rotación automática de secretos y la integración con sistemas de autenticación externos, que nos ayudan a garantizar el cumplimiento de los requisitos de seguridad y la integridad de nuestros datos en todo momento. Block and Soto (2022)

3.4.3. Plataforma de desarrollo

En el desarrollo del proyecto, se estableció una plataforma de desarrollo robusta y eficiente, basada en cuatro pilares principales, cada uno respaldado por tecnologías específicas que permitieron alcanzar los objetivos del proyecto de manera efectiva.

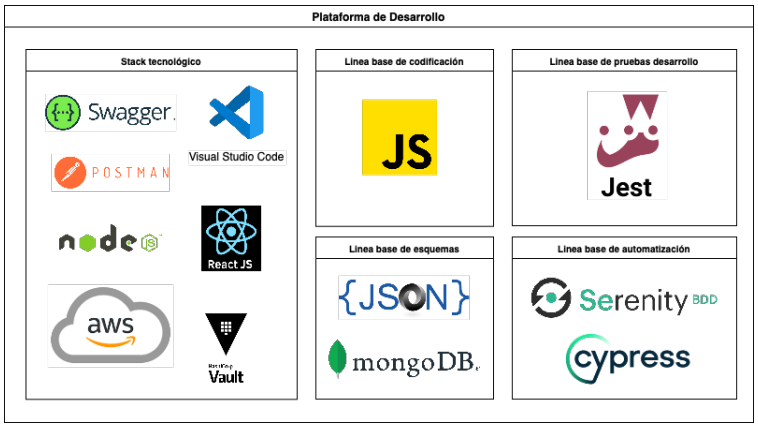


Figura 2: Plataforma de desarrollo seleccionada

- **Stack tecnológico**

  - Para la documentación y definición clara de las API, se optó por Swagger, una herramienta altamente reconocida que proporciona una especificación estándar para describir servicios RESTful.
  - La elección de Visual Studio Code como el entorno de desarrollo integrado (IDE) principal se basó en su amplia aceptación en la comunidad de desarrolladores y su robusta funcionalidad para la escritura de código, la depuración y la gestión de proyectos.
  - Postman fue seleccionado para las pruebas de las API debido a su versatilidad y facilidad de uso, lo que permitió realizar pruebas automatizadas de manera eficiente y colaborativa.
  - La utilización de Node.js como plataforma de backend se justificó por su capacidad para construir aplicaciones escalables y de alto rendimiento, así como por su amplio ecosistema de módulos y bibliotecas disponibles.
  - React.js se eligió como el framework de frontend debido a su enfoque en la creación de interfaces de usuario interactivas y dinámicas, lo que lo convierte en una opción ideal para el desarrollo de aplicaciones web modernas.
  - AWS se seleccionó como proveedor de servicios en la nube debido a su escalabilidad, fiabilidad y flexibilidad, así como a la amplia gama de servicios y herramientas que ofrece para el despliegue y la gestión de aplicaciones.
  - La adopción de Vault como herramienta de gestión de secretos y credenciales se basó en su capacidad para proporcionar un almacenamiento centralizado y seguro para secretos de aplicaciones, lo que garantiza la confidencialidad y la integridad de la información sensible.

- **Línea base de codificación**

  La elección de JavaScript como lenguaje de programación principal se basó en su amplia adopción en la comunidad de desarrollo web, así como en su capacidad para construir aplicaciones escalables y de alto rendimiento.

- **Línea base de esquemas**

  - El uso de JSON como formato de representación de datos se justificó por su simplicidad, legibilidad y amplia compatibilidad con una variedad de lenguajes y plataformas, lo que facilita el intercambio de datos entre el cliente y el servidor.
  - MongoDB fue elegido por su flexibilidad y escalabilidad, siendo ideal para manejar grandes volúmenes de datos de manera eficiente. Su modelo de documentos y capacidad de escalado horizontal garantizan un rendimiento óptimo, mientras que su amplia comunidad y soporte aseguran continuidad en el desarrollo.

- **Línea base de pruebas de desarrollo**

  La elección de Jest como framework de pruebas se basó en su popularidad y su capacidad para proporcionar una suite completa de herramientas para escribir, ejecutar y depurar pruebas de JavaScript de manera efectiva.

- **Línea base de automatización de pruebas**

  - La selección de SerenityRest para la automatización de pruebas de API se justificó por su facilidad de uso y su capacidad para simplificar la escritura y ejecución de pruebas automatizadas de servicios RESTful.
  - Cypress se eligió como framework de pruebas de extremo a extremo debido a su enfoque moderno y su capacidad para proporcionar una experiencia de usuario ágil y confiable durante el proceso de desarrollo y validación.

3.5. Estructura de base de datos.

3.5.1. Tablas principales

- **Users:** Almacena información sobre los usuarios de la aplicación, incluyendo administradores, motociclistas y usuarios finales. Cada documento en esta colección contiene detalles como el nombre completo, correo electrónico, tipo de usuario y otros datos relevantes según el tipo de usuario.

  - **_id:** ObjectId (Clave primaria)
  - **full_name:** String. Nombre completo del usuario.
  - **email:** String. Correo electrónico del usuario.
  - **password:** String. Contraseña del usuario (se recomienda almacenarla de forma segura mediante hash).
  - **user_type:** String. Tipo de usuario (Administrador, Motociclista, Usuario Final).
  - **address:** String. Dirección del usuario.
  - **city:** String. Ciudad del usuario.
  - **state:** String. Estado o Departamento del usuario.
  - **postal_code:** String. Código postal del usuario.
  - **phone_number:** String. Número de teléfono del usuario.
  - **registration_date:** Date. Fecha de registro del usuario.
  - **account_status:** String. Estado de la cuenta del usuario (activo, inactivo, suspendido).

- **Orders:** Registra los detalles de los pedidos realizados por los usuarios finales. Cada pedido está asociado a un usuario y puede contener información como la ubicación de recogida y entrega, el estado del pedido y el total del pedido.

- **_id:** ObjectId (Clave primaria)
- **user_id:** ObjectId. ID del usuario que realizó el pedido (referencia a la colección de usuarios).
- **order_date_time:** Date. Fecha y hora del pedido.
- **pickup_address:** String. Dirección de recogida del pedido.
- **delivery_address:** String. Dirección de entrega del pedido.
- **order_status:** String. Estado del pedido (pendiente, en progreso, completado, cancelado).
- **additional_comments:** String. Comentarios adicionales proporcionados por el usuario al realizar el pedido.

- **Drivers:** Almacena información sobre los conductores asignados a los pedidos. Cada documento en esta colección incluye detalles como el nombre del conductor, la licencia de conducir y la disponibilidad.
  - **_id:** ObjectId (Clave primaria)
  - **user_id:** ObjectId. ID del usuario que realizó el pedido (referencia a la colección de usuarios).
  - **identification_documents:** String. Documentos de identificación del conductor (licencia de conducir, número de identificación, etc.).
  - **availability:** String. Disponibilidad del conductor (ocupado, disponible, en descanso).
  - **last_known_location:** [Double, Double]. Última ubicación conocida del conductor (formato GeoJSON para almacenar latitud y longitud).
  - **average_rating:** Number. Calificación promedio del conductor.
  - **completed_orders_count:** Number. Cantidad de pedidos completados por el conductor.

- **Vehicles:** Registra los detalles de los vehículos utilizados para el transporte de mercancías. Cada documento puede incluir información como la marca, modelo, placa y capacidad de carga de la motocicleta.
  - **_id:** ObjectId (Clave primaria)
  - **driver_id:** ObjectId. ID del conductor propietario del vehículo (referencia a la colección de drivers).
  - **make:** String. Marca del vehículo.
  - **model:** String. Modelo del vehículo.
  - **year:** Number. Año de fabricación del vehículo.
  - **plate_number:** String. Número de placa del vehículo.

- **color:** String. Color del vehículo.
- **registration_date:** Date. Fecha de registro del vehículo.
- **vehicle_type:** String. Tipo de vehículo (moto, scooter, etc.).
- **insurance_company:** String. Compañía de seguros del vehículo.
- **insurance_policy_number:** String. Número de póliza de seguro del vehículo.
- **active:** Boolean. Indica si el vehículo está activo o inactivo en el sistema.

3.5.2. Tablas auxiliares

- **OrderDetails:** Contiene información detallada sobre los productos incluidos en cada pedido. Cada documento en esta colección incluye detalles como el nombre del producto, cantidad, precio unitario y subtotal.
  - **_id:** ObjectId (Clave primaria)
  - **order_id:** ObjectId. ID del pedido al que pertenece el detalle (referencia a la colección de pedidos).
  - **item_description:** String. Descripción del artículo.
  - **quantity:** Number. Cantidad del artículo.
  - **unit_price:** Number. Precio unitario del artículo.
  - **subtotal:** Number. Subtotal del artículo (cantidad * precio unitario).
- **Payments:** Registra los pagos realizados por los usuarios finales. Cada documento contiene información sobre el monto del pago, el método de pago utilizado y el pedido asociado.
  - **_id:** ObjectId (Clave primaria)
  - **order_id:** ObjectId. ID del pedido asociado al pago (referencia a la colección de pedidos).
  - **payment_method:** String. Método de pago utilizado (tarjeta de crédito, efectivo, transferencia bancaria, etc.).
  - **amount_paid:** Number. Monto pagado.
  - **currency:** String. Moneda en la que se realizó el pago.
  - **payment_status:** String. Estado del pago (pendiente, completado, rechazado).
  - **payment_date_time:** Date. Fecha y hora del pago.
  - **transaction_number:** String. Número de transacción del pago.
- **OrderHistory:** Registra el historial de cambios de estado de los pedidos. Cada documento contiene detalles como el estado anterior y nuevo del pedido, la fecha y hora del cambio y el usuario responsable del cambio.
  - **_id:** ObjectId (Clave primaria)

- **order_id:** ObjectId. ID del pedido asociado al evento (referencia a la colección de pedidos).
- **event_date_time:** Date. Fecha y hora del evento.
- **event_description:** String. Descripción del evento.
- **responsible_user_id:** ObjectId. ID del usuario responsable del evento (referencia a la colección de usuarios).

3.5.3. Tablas de registro

- **AccessLogs:** Registra los registros de acceso al sistema por parte de los usuarios y administradores. Cada documento contiene información como la fecha y hora del acceso, la dirección IP del usuario y el resultado de la operación.
  - **_id:** ObjectId (Clave primaria)
  - **user_id:** ObjectId. ID del usuario asociado al registro de acceso (referencia a la colección de usuarios).
  - **access_date_time:** Date. Fecha y hora del acceso.
  - **ip_address:** String. Dirección IP desde la cual se realizó el acceso.
  - **action_type:** String. Tipo de acción realizada (inicio de sesión, cierre de sesión, intento de acceso fallido, etc.).

3.6. Modelo C4

Durante la fase de diseño de nuestra arquitectura, utilizamos el modelo C4 como una guía estructurada para visualizar y comunicar la arquitectura en diferentes niveles de abstracción. El modelo C4 nos proporcionó un enfoque claro y escalable para describir la estructura del sistema, desde el contexto general hasta los detalles internos de los componentes. A través de esta metodología, pudimos establecer una arquitectura sólida y comprensible que abordaba los requisitos funcionales identificados y las necesidades de los usuarios.

3.6.1. Nivel1 - Contexto del sistema

En este nivel, identificamos los actores externos que interactúan con nuestro sistema de transporte de mercancías en motocicletas. Estos actores externos incluyen usuarios finales que solicitan entregas, administradores que supervisan el funcionamiento del sistema y motociclistas que realizan las entregas. El sistema actúa como el punto central de interacción entre estos actores externos, facilitando el intercambio de información y la ejecución de las diferentes funciones de la aplicación.

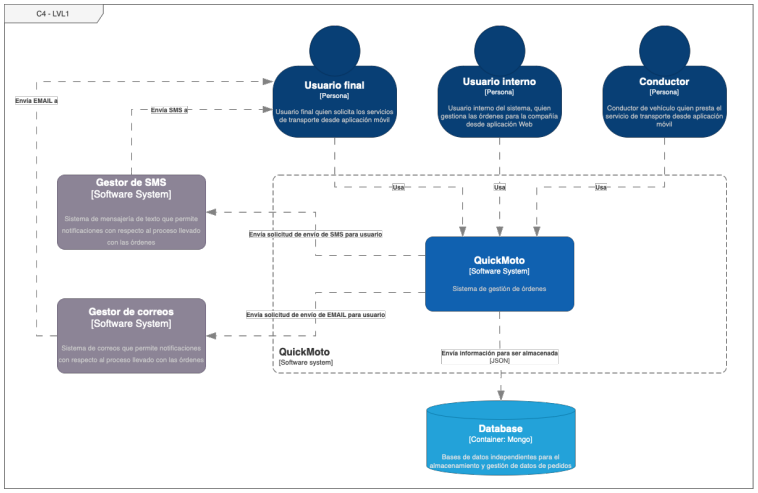


Figura 3: Diagrama C4 - Nivel1

3.6.2. Nivel2 - Contenedor

En este nivel, identificamos los contenedores internos del sistema y cómo se comunican entre sí para cumplir con los requisitos funcionales.

La aplicación sirve como la interfaz principal a través de la cual los usuarios finales, los administradores y los motociclistas interactúan con el sistema. Permite a los usuarios realizar pedidos, realizar seguimiento de las entregas en tiempo real, gestionar pagos y acceder a otras funcionalidades relacionadas con el transporte de mercancías. El servidor de aplicación dentro de este contenedor es responsable de manejar las solicitudes entrantes de los clientes, procesar la lógica empresarial y coordinar la comunicación con otros sistemas y servicios, como bases de datos y servicios externos de pagos.

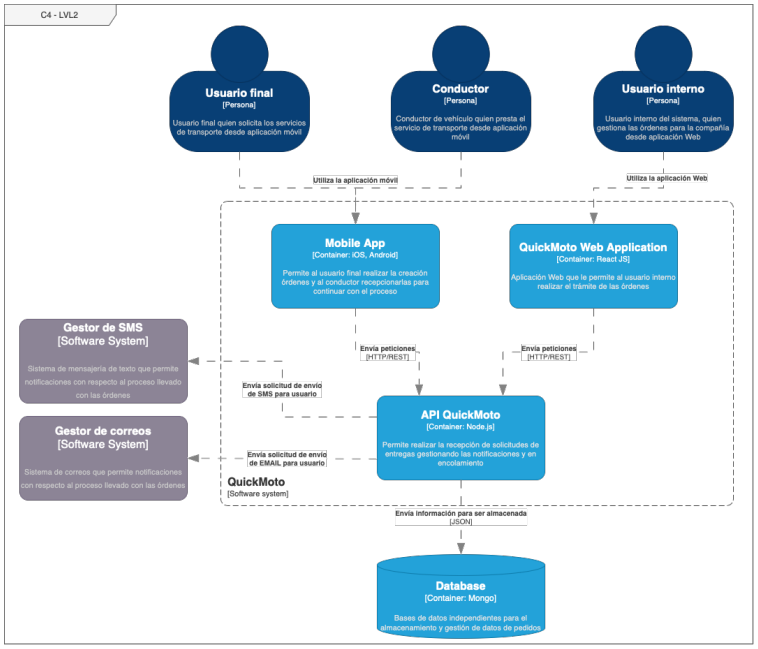


Figura 4: Diagrama C4 - Nivel2

3.6.3. Nivel3 - Componentes

Durante la fase de diseño del nivel 3, desarrollamos una serie de componentes clave para abordar los diversos requisitos funcionales identificados. Estos componentes, meticulosamente diseñados y desarrollados, desempeñan roles fundamentales en la funcionalidad y el rendimiento de nuestra aplicación. A continuación, presentamos una visión detallada de cada componente y cómo contribuye a nuestro sistema.

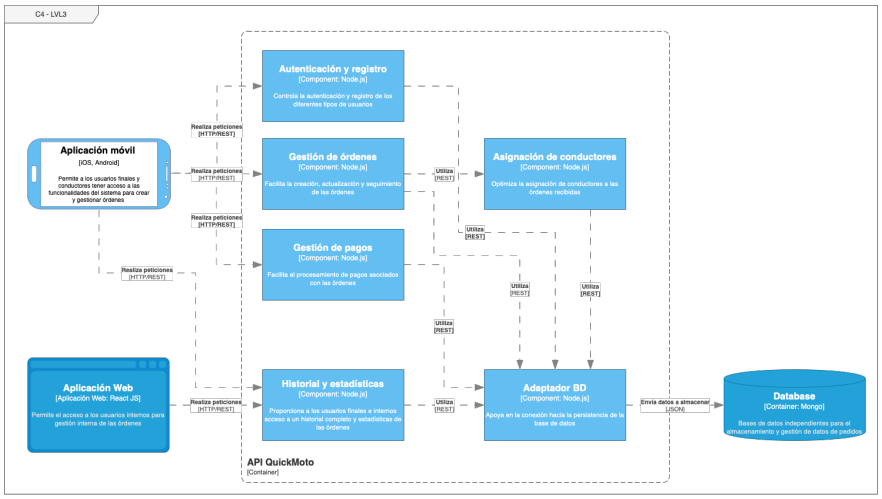


Figura 5: Diagrama C4 - Nivel3

3.7. Detallado de componentes

Durante el diseño de la arquitectura para nuestra aplicación, identificamos la necesidad de definir una serie de componentes específicos que trabajarían en conjunto para satisfacer los requisitos funcionales establecidos. Estos componentes se diseñaron para abordar diferentes aspectos del proceso, desde la autenticación y registro de usuarios hasta la gestión de pedidos, asignación de conductores, seguimiento en tiempo real, gestión de pagos y análisis de estadísticas. A continuación, presentamos una descripción detallada de cada componente y por qué se definió de esa manera para cumplir con los objetivos de nuestro proyecto.

3.7.1. Autenticación y registro de usuarios

Definimos un módulo de autenticación y registro de usuarios para gestionar eficientemente el proceso de registro de los diferentes tipos de usuarios en la aplicación. Esto nos permitió garantizar la seguridad de las credenciales de acceso y verificar la autenticidad de los datos proporcionados durante el registro. Además, al integrar este módulo con nuestro sistema de gestión de bases de datos, pudimos almacenar la información de los usuarios registrados de manera segura y confiable.

- **Autenticación:** Definimos un servicio de autenticación para verificar la identidad de los usuarios durante el proceso de inicio de sesión en la aplicación. Este servicio se diseñó para validar las credenciales de acceso proporcionadas por los usuarios, incluyendo el nombre de usuario y la contraseña. Al implementar este servicio, buscamos garantizar la seguridad de la plataforma al prevenir accesos no autorizados y proteger la información confidencial de los usuarios.

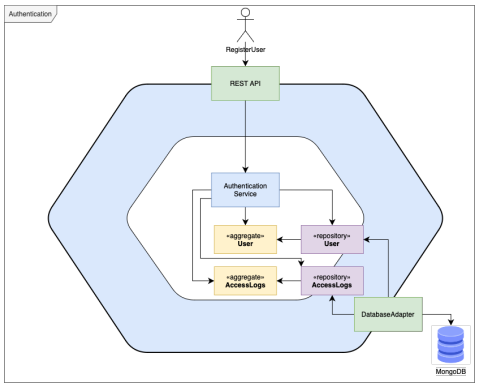


Figura 6: Diagrama hexagonal servicio de autenticación

- **Registro de usuarios:** Diseñamos un servicio de registro de usuarios para facilitar el proceso de registro de nuevos usuarios en la aplicación. Este servicio busca que los usuarios puedan crear una cuenta. Además, se definieron validaciones para garantizar la unicidad de la dirección de correo electrónico y verificar la autenticidad de los datos proporcionados.

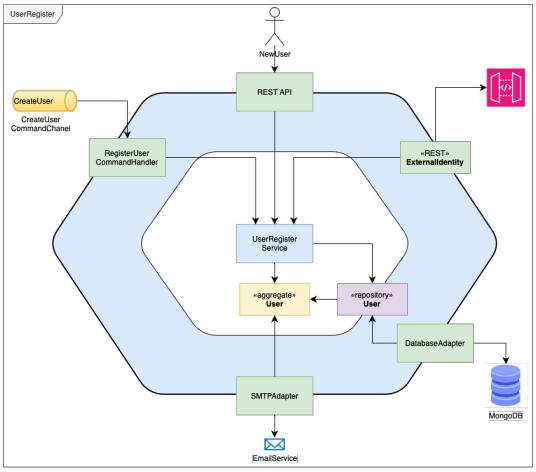


Figura 7: Diagrama hexagonal servicio de registro de usuarios

- **Gestión de sesiones:** Diseñamos un servicio de gestión de sesiones para mantener la sesión activa de los usuarios durante su interacción con la aplicación. Este servicio se encarga de administrar el inicio de sesión y la sesión de usuario, gestionando los tokens de acceso y actualizando la información de la sesión según sea necesario. Al implementar este servicio, buscamos proporcionar una experiencia de usuario continua y sin interrupciones, permitiendo a los usuarios acceder a las funcionalidades de la aplicación de manera segura y eficiente.

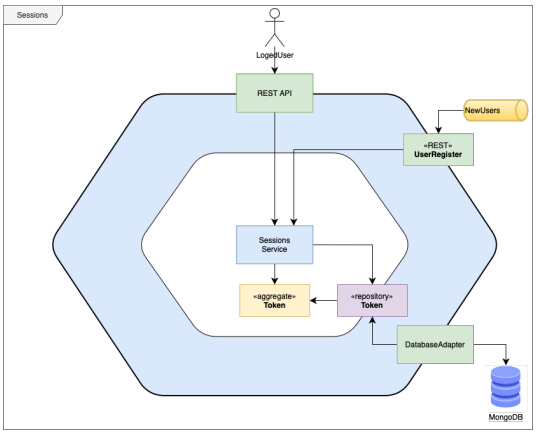


Figura 8: Diagrama hexagonal servicio de gestión de sesiones

- **Recuperación de contraseñas:** Definimos un servicio de recuperación de contraseña para permitir a los usuarios restablecer sus contraseñas en caso de olvido o pérdida. Este servicio proporcionaría a los usuarios un mecanismo seguro para restablecer sus contraseñas, como la generación de tokens de restablecimiento de contraseña y la validación de la identidad del usuario a través de métodos de autenticación secundarios.

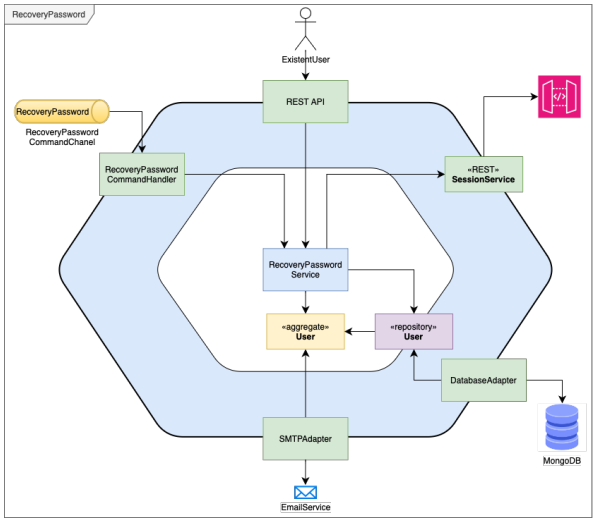


Figura 9: Diagrama hexagonal servicio de recuperar contraseña

- **Integración proveedores de identidad externos:** Especificamos una integración con proveedores de identidad externos para permitir a los usuarios registrarse e iniciar sesión utilizando cuentas de terceros, como Google, Facebook o Microsoft. Este servicio ofrece a los usuarios una opción conveniente de inicio de sesión sin necesidad de crear y recordar nuevas credenciales. Decidimos integrar este servicio para mejorar la accesibilidad y la experiencia del usuario, aprovechando la autenticación segura proporcionada por los proveedores externos.

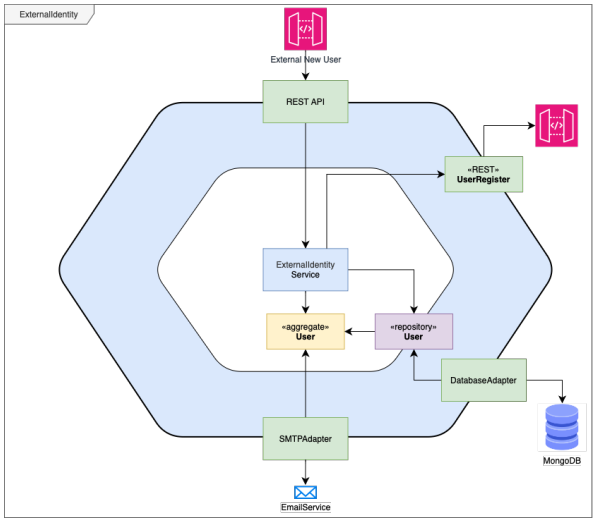


Figura 10: Diagrama hexagonal servicio de indentidad externa

3.7.2. Gestión de pedidos

Optamos por diseñar un módulo de gestión de pedidos para facilitar la creación, actualización y seguimiento de los pedidos de transporte de mercancías. Esto nos permitió garantizar que los usuarios finales tendrán experiencia fluida y eficiente al realizar pedidos a través de la aplicación. Además, al definir funcionalidades de seguimiento en tiempo real, podremos mejorar la visibilidad y transparencia del proceso de entrega para todos los involucrados.

- **Creación de pedidos:** Diseñamos un servicio para permitir a los usuarios finales crear nuevos pedidos de transporte de mercancías. Esto se definió para ofrecer una forma intuitiva y fácil para que los usuarios soliciten el transporte de sus productos a través de la aplicación. Al proporcionar este servicio, buscamos agilizar el proceso de creación y mejorar la experiencia del usuario al ofrecer una interfaz simple y amigable.

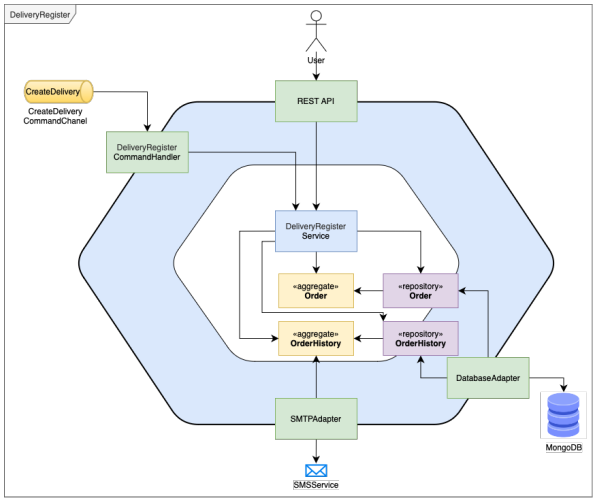


Figura 11: Diagrama hexagonal servicio de creación de pedidos

- **Actualización de pedidos:** Definimos un servicio para permitir la actualización de pedidos existentes. Esto se agregó con el objetivo de brindar flexibilidad a los usuarios finales y permitirles realizar cambios en sus pedidos, como agregar instrucciones especiales o realizar la cancelación de pedidos que ya no deseen completar al igual que actualizar que el pedido se culminó correctamente.

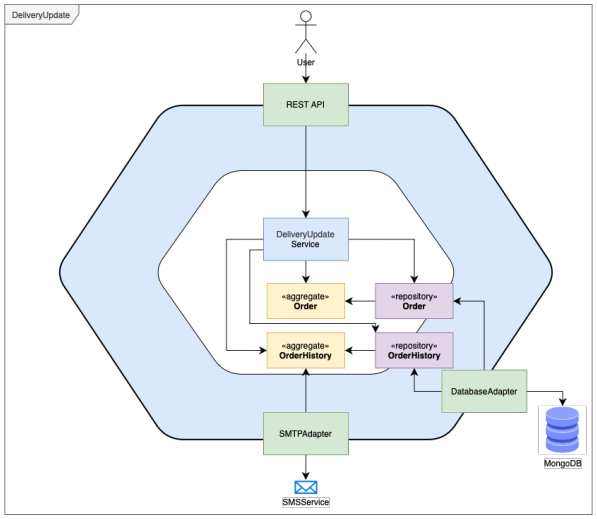


Figura 12: Diagrama hexagonal servicio de actualización de pedidos

- **Seguimiento de pedidos:** Diseñamos un servicio para permitir a los usuarios finales rastrear el estado de sus pedidos en tiempo real. Esto se definió para ofrecer transparencia y visibilidad en el proceso de entrega de los productos.

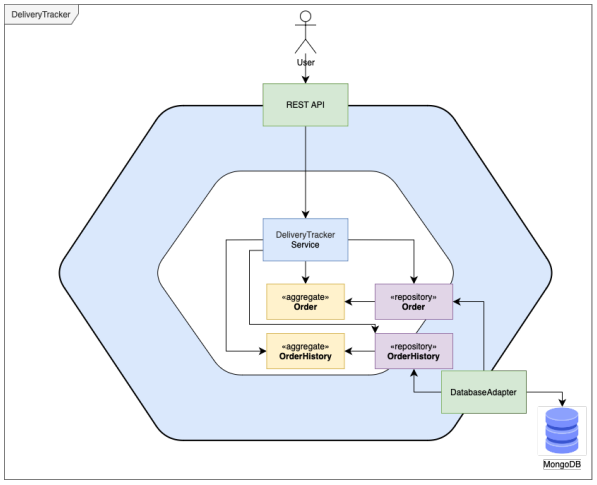


Figura 13: Diagrama hexagonal servicio de seguimiento de pedidos

3.7.3. Asignación de conductores

Definimos un módulo de asignación de conductores para optimizar la asignación a los pedidos recibidos. Esto se reflejará en las asignaciones automáticas y eficientes, considerando diversos criterios como la ubicación y disponibilidad de los conductores. Además, al proporcionar funcionalidades de reasignación, podremos adaptarnos rápidamente a cambios en la disponibilidad o en las circunstancias del pedido.

- **Asignación automática de conductores:** Diseñamos un servicio de asignación automática de conductores para optimizar la asignación de conductores a los pedidos recibidos. Este servicio utilizará algoritmos de asignación inteligente que consideran la ubicación, disponibilidad y capacidad de carga de cada conductor para realizar asignaciones eficientes.

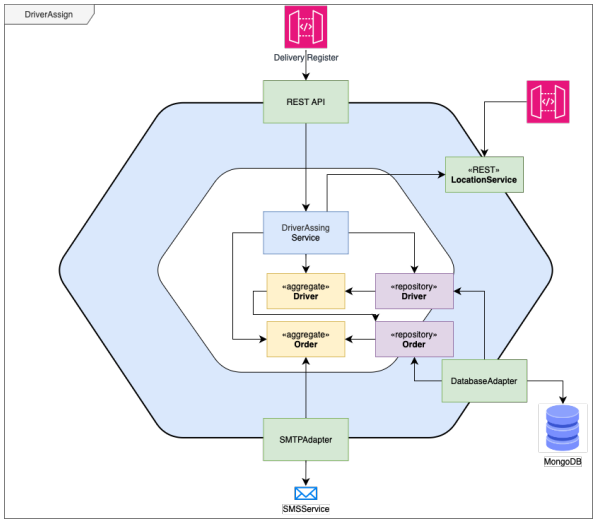


Figura 14: Diagrama hexagonal servicio de asignación de conductores

- **Re-asignación automática de conductores:** Definimos un servicio de reasignación dinámica de conductores para adaptarnos rápidamente a cambios en la disponibilidad o en las circunstancias del pedido. Este servicio permitirá reasignar conductores en tiempo real en caso de emergencias, retrasos o cancelaciones de pedidos.

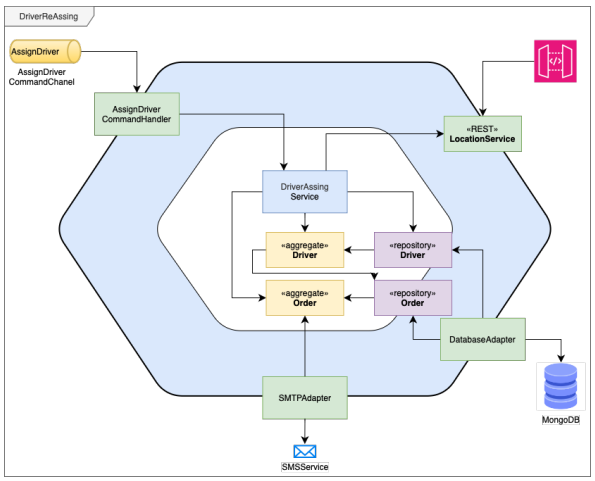


Figura 15: Diagrama hexagonal servicio de re-asignación de conductores

3.7.4. Gestión de pagos

Diseñamos un módulo de gestión de pagos para facilitar el procesamiento de pagos asociados con los pedidos de transporte de mercancías. Esto nos permitirá admitir una variedad de métodos de pago y garantizar transacciones seguras y confiables. Además, al integrarnos con servicios de pago externos, podremos ofrecer a los usuarios una experiencia de pago conveniente y sin problemas.

- **Procesamiento de pagos:** Decidimos diseñar un servicio de procesamiento de pagos para permitir a los usuarios realizar transacciones seguras y confiables a través de la aplicación. Este servicio se diseñó para integrarse con diferentes proveedores de servicios de pago, lo que nos permitiría admitir una variedad de métodos de pago, como tarjetas de crédito, transferencias bancarias y pagos en efectivo.

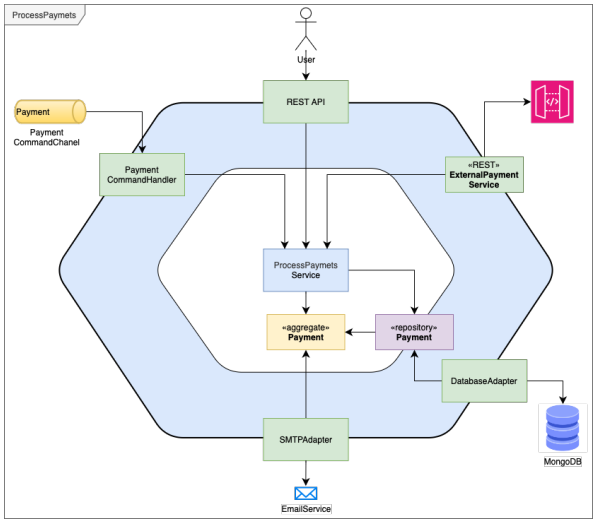


Figura 16: Diagrama hexagonal servicio de procesamiento de pagos

- **Gestión de suscripciones y descuentos:** Decidimos desarrollar un servicio de gestión de suscripciones y descuentos para permitir a los usuarios finales beneficiarse de ofertas especiales y promociones. Este servicio se diseñó para gestionar suscripciones recurrentes y aplicar descuentos automáticos según las condiciones establecidas.

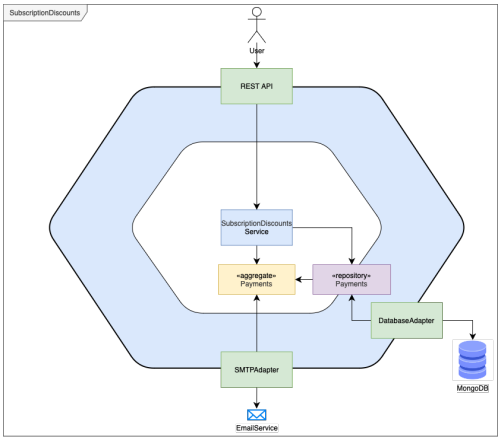


Figura 17: Diagrama hexagonal servicio de subscripciones y descuentos

- **Reembolsos y devoluciones:** Incluimos un servicio de gestión de reembolsos y devoluciones para manejar de manera eficiente las solicitudes de reembolso y devolución de los usuarios finales. Este servicio se diseñó para seguir políticas claras y transparentes en cuanto a los casos elegibles para reembolso y devolución, lo que nos permitiría garantizar una experiencia de usuario satisfactoria.

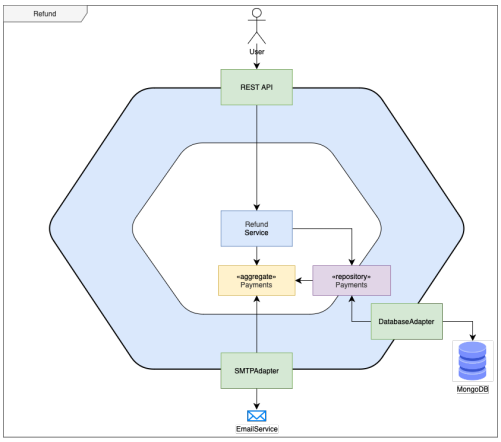


Figura 18: Diagrama hexagonal servicio de reembolso y devoluciones

3.7.5. Historial de pedidos y estadísticas

Definimos un módulo de historial de pedidos y estadísticas para proporcionar a los usuarios finales e internos acceso a un historial completo de pedidos, así como a estadísticas y análisis sobre el rendimiento del servicio. Esto nos permitirá mejorar la experiencia del usuario al proporcionar información útil y relevante sobre sus actividades pasadas. Además, al generar informes detallados, podremos obtener información valiosa para mejorar continuamente nuestro servicio.

- **Consulta historial de pedidos:** Definimos un servicio de consulta de historial de pedidos para permitir a los usuarios finales acceder a su historial completo de pedidos anteriores. Este servicio proporcionará una interfaz de consulta intuitiva que permitirá a los usuarios buscar y filtrar sus pedidos por diferentes criterios, como fecha, estado o destino.

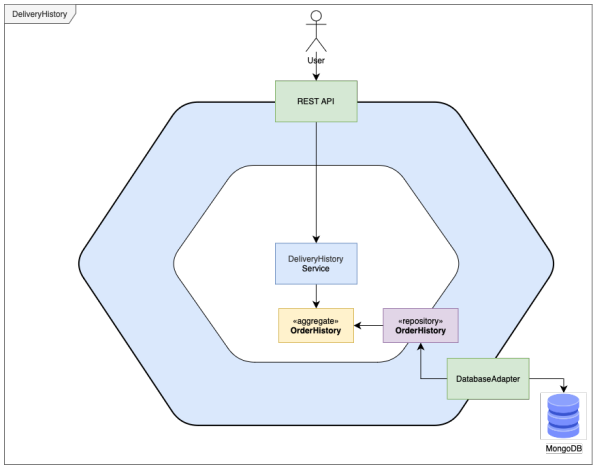


Figura 19: Diagrama hexagonal servicio de historial de pedidos

- **Generación de estadísticas:** Definimos un servicio de generación de estadísticas para analizar y visualizar el rendimiento del servicio en función de diferentes métricas, como el tiempo de entrega promedio, la satisfacción del cliente y la eficiencia de los conductores. Este servicio procesará los datos almacenados en el servicio de almacenamiento de datos y generará informes detallados y gráficos visuales para proporcionar una visión completa del rendimiento del servicio.

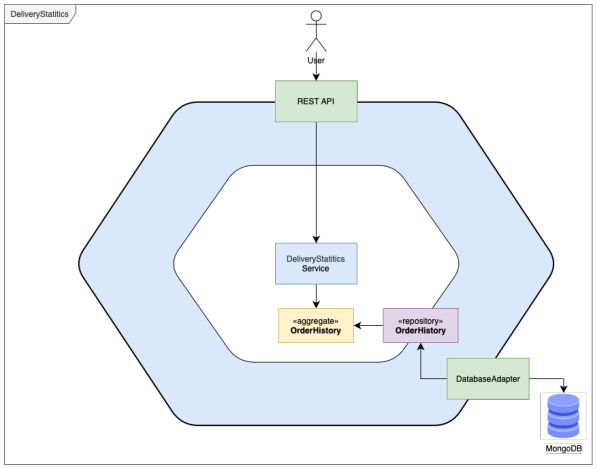


Figura 20: Diagrama hexagonal servicio de generación de estadísticas

3.8. Plataforma de despliegue

En esta sección, exploraremos la plataforma de despliegue diseñada para nuestra aplicación, concebida bajo el entorno confiable y escalable de Amazon Web Services (AWS). Como equipo, hemos dedicado tiempo y esfuerzo a modelar nuestra arquitectura apoyados en los anteriores diagramas C4 y los diagramas de los servicios hexagonales que constituyen el núcleo de nuestra aplicación. Ahora, damos un paso adelante para abordar la infraestructura que soportará y facilitará el despliegue eficiente y seguro de nuestra solución.

3.8.1. Diagrama de despliegue

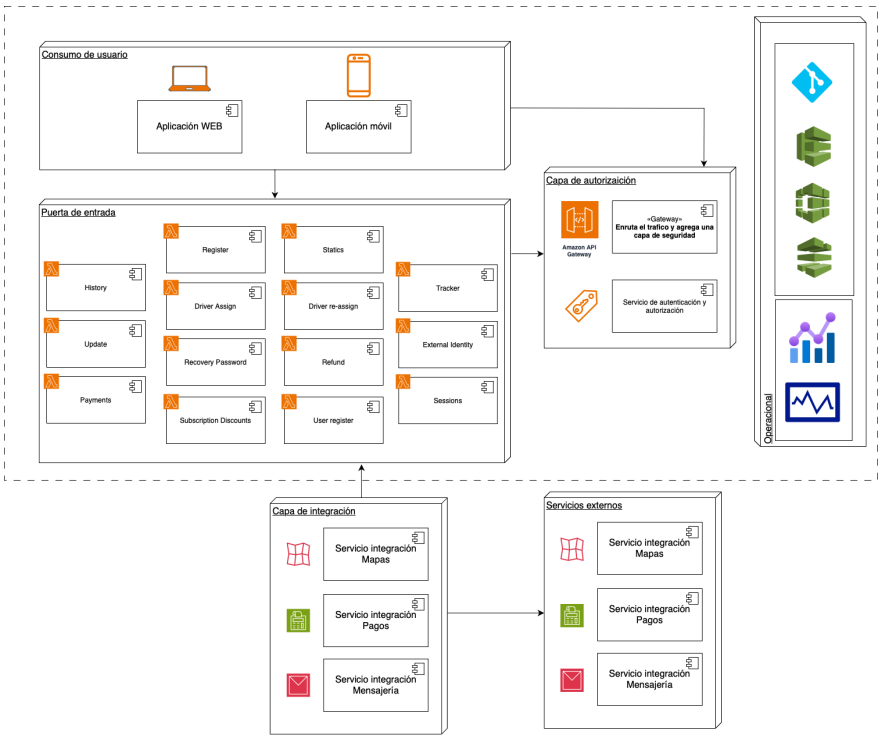


Figura 21: Diagrama de despliegue

Profundizaremos en las distintas capas que componen nuestra plataforma de despliegue, que puede verse desde la entrada de usuario hasta la gestión operativa, cada capa desempeña un papel fundamental en la entrega eficiente y segura de nuestro servicio. Exploraremos en detalle cómo cada componente se integra y contribuye al funcionamiento global de la plataforma, brindando una visión completa de su arquitectura y funcionalidad.

- **Consumo de usuario:** Esta capa sirve como el punto de entrada para los usuarios finales de nuestra plataforma de transporte de mercancías en motocicletas. La Aplicación Web ofrece una experiencia de usuario intuitiva y accesible a través de navegadores de escritorio, mientras que la Aplicación Móvil proporciona una interfaz optimizada para dispositivos

móviles, permitiendo a los usuarios acceder a los servicios desde cualquier lugar y en cualquier momento.

- **Puerta de entrada:** Aquí se encuentran los servicios encargados de gestionar las solicitudes de los usuarios y dirigirlas hacia los componentes adecuados de la plataforma. Los Servicios de Gestión de Usuarios facilitan el registro, autenticación y gestión de cuentas de usuario. El Servicio de Gestión de Pedidos se encarga de recibir y procesar las solicitudes de transporte de mercancías, mientras que el Servicio de Estadísticas proporciona información útil sobre el rendimiento y la actividad de la plataforma para la toma de decisiones informadas.

- **Capa de integración:** En esta capa se encuentran los servicios que permiten la integración fluida de nuestra plataforma con sistemas externos y servicios de terceros. El Servicio de Integración con Mapas ofrece capacidades de geolocalización y enrutamiento para mejorar la precisión y eficiencia de las entregas. El Servicio de Integración con Servicios de Pago facilita transacciones seguras y convenientes, mientras que el Servicio de Integración con Servicios de Mensajería permite una comunicación efectiva entre conductores y usuarios.

- **Capa de autorización:** Esta capa garantiza la seguridad y la autorización adecuada de las solicitudes que ingresan a la plataforma. El API Gateway actúa como un punto de entrada único para todas las solicitudes externas, proporcionando funciones de enrutamiento y control de acceso. El Servicio de Autorización y Autenticación verifica la identidad de los usuarios y controla su acceso a los recursos de la plataforma, garantizando la confidencialidad y la integridad de los datos.

- **Servicios externos:** Aquí se encuentran los servicios de terceros que complementan y mejoran las capacidades de nuestra plataforma. El Servicio de Mapas proporciona datos geoespaciales precisos y actualizados para optimizar las rutas de entrega. El Servicio de Pagos facilita transacciones financieras seguras y convenientes entre conductores y usuarios, mientras que el Servicio de Mensajería permite una comunicación directa y eficiente entre ambas partes.

- **Operacional:** Esta capa se encarga de gestionar aspectos operativos clave de la plataforma para garantizar su eficiencia y confiabilidad a largo plazo. El Manejo de Repositorios y Código Fuente facilita el desarrollo, prueba e implementación continua de nuevas funcionalidades y mejoras en la plataforma. El Manejo de Métricas y Reportes proporciona información valiosa sobre el rendimiento y la salud del sistema, permitiendo una optimización proactiva y una toma de decisiones basada en datos.

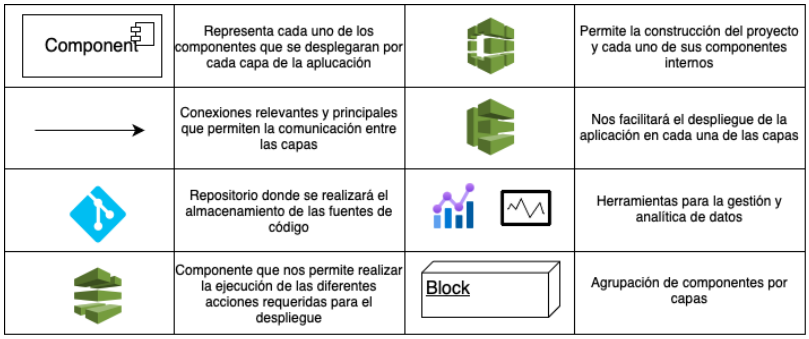


Figura 22: Iconografía despliegue

3.8.2. Diagrama de infraestructura

Nuestra arquitectura de infraestructura en AWS, que puede verse en la figura 23, se ha diseñado cuidadosamente desde cero, apoyado en la documentación proporcionada por AWS, para garantizar la seguridad, escalabilidad y disponibilidad de nuestra aplicación, atendiendo a múltiples usuarios que acceden a través de diversos dispositivos, incluyendo dispositivos móviles y páginas web. La distribución de los componentes en secciones principales responde a criterios específicos que maximizan la eficiencia operativa y la resistencia del sistema.

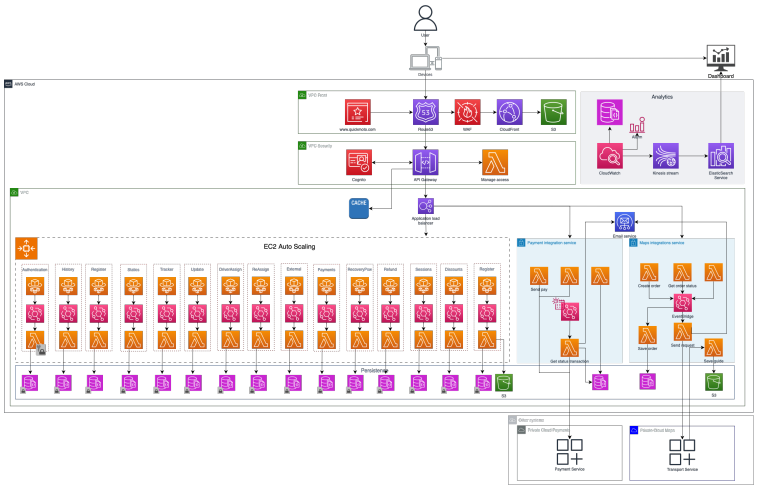


Figura 23: Diagrama de infraestructura

- **VPC front:** Esta sección alberga los componentes de la infraestructura relacionados con la interfaz de usuario y la experiencia del cliente. Aquí se encuentran los servicios y recursos necesarios para la gestión del tráfico web y la presentación de la interfaz de usuario en dispositivos móviles y páginas web.

- **VPC security:** La seguridad es una prioridad fundamental en nuestra arquitectura. En esta sección, se concentran los componentes encargados de proteger la infraestructura y los datos

sensibles de la aplicación. Esto incluye la implementación de firewalls, grupos de seguridad, servicios de autenticación y autorización, así como la gestión de claves de cifrado.

- **Elastic container:** La sección de Elastic Container es crucial para garantizar la escalabilidad y la disponibilidad de nuestros servicios. Aquí utilizamos contenedores para encapsular y desplegar nuestras aplicaciones de forma eficiente, permitiendo una gestión ágil de recursos y una rápida escalabilidad en respuesta a la demanda del usuario.

- **Services integrations:** Esta sección se enfoca en la integración de servicios externos necesarios para el funcionamiento de nuestra aplicación. Aquí gestionamos la comunicación con otros sistemas y servicios, como pasarelas de pago, servicios de geolocalización, y sistemas de mensajería, asegurando una integración fluida y confiable.

- **Persistence:** La persistencia de datos es esencial para nuestra aplicación. En esta sección, gestionamos las bases de datos y sistemas de almacenamiento necesarios para almacenar y gestionar la información de manera segura y eficiente. Esto incluye bases de datos relacionales y NoSQL, así como sistemas de almacenamiento de archivos.

- **OtherSystems:** Esta sección alberga los servicios externos clave necesarios para nuestra aplicación, como proveedores de mapas, servicios de pago en línea y sistemas de mensajería. Estos servicios se integran de manera estratégica para mejorar la funcionalidad y la experiencia del usuario en nuestra plataforma.

Figura 24: Iconografía infraestructura

La figura 24 presenta la iconografía de AWS que hemos empleado para visualizar y comunicar la estructura y componentes de nuestra solución representada en la figura 23. Esta representación gráfica nos permite una comprensión clara y concisa de cómo cada servicio de AWS contribuye a la funcionalidad general de nuestra arquitectura.

Capítulo 4

Evaluación

En esta sección exploraremos cómo hemos aplicado el método ATAM para evaluar y mejorar la arquitectura de nuestra aplicación. A través de workshops con expertos y stakeholders, identificamos riesgos, debilidades y áreas de mejora en nuestra arquitectura, desarrollando recomendaciones para fortalecerla. Este proceso nos ha permitido garantizar que nuestra arquitectura esté alineada con los objetivos del proyecto y las necesidades de los usuarios.

4.1. Identificación de stakeholders y escenarios de calidad

- **Stakeholders relevantes**

  - **Usuarios de la aplicación:** Representa a los clientes que utilizan la aplicación móvil para solicitar y rastrear entregas de mercancías. Su experiencia de usuario y satisfacción son cruciales para el éxito y la adopción continua de la aplicación.
  - **Conductores de motocicletas:** Este grupo comprende a los conductores que utilizan la aplicación para aceptar y completar entregas. Su eficiencia, seguridad y facilidad de uso de la aplicación son aspectos críticos a considerar en el diseño de la arquitectura.
  - **Administradores de la plataforma:** Responsables de administrar y mantener la plataforma de la aplicación, incluyendo la gestión de usuarios, pagos y soporte técnico. La capacidad de la arquitectura para admitir funciones administrativas eficientes y seguras es esencial para el funcionamiento fluido de la plataforma.
  - **Desarrolladores del software:** Equipo encargado del desarrollo y mantenimiento de la aplicación móvil y su infraestructura subyacente. La arquitectura debe ser modular, escalable y fácil de mantener para facilitar el trabajo de los desarrolladores y garantizar la evolución continua de la aplicación.
  - **Proveedores de servicios de mapas y navegación:** Stakeholders externos que proporcionan servicios de mapas y navegación utilizados por la aplicación. La integración fluida con estos servicios es fundamental para proporcionar a los usuarios rutas precisas y actualizadas.
  - **Proveedores de métodos de pago:** Entidades encargadas de procesar los pagos realizados a través de la aplicación. La seguridad y confiabilidad en la integración con estos proveedores son críticas para garantizar transacciones seguras y sin problemas para los usuarios.

- **Escenarios de calidad identificados**

  - **Rendimiento:** El sistema debe ser capaz de proporcionar una ruta óptima en un tiempo razonable cuando un usuario solicita la búsqueda de rutas para la entrega de mercancías. Esto implica la optimización de algoritmos de enrutamiento y la gestión eficiente de recursos computacionales para garantizar tiempos de respuesta rápidos incluso en condiciones de carga elevada.
  - **Seguridad:** El sistema debe contar con mecanismos sólidos para detectar y prevenir accesos no autorizados a datos sensibles, incluyendo información de usuarios y detalles de las entregas. Se deben implementar protocolos de cifrado robustos para proteger la privacidad y la seguridad de los datos, así como sistemas de autenticación multifactor para garantizar la legitimidad de las transacciones.
  - **Disponibilidad:** El sistema debe ser capaz de manejar la pérdida de conexión a la red durante la entrega, cambiando de manera efectiva al modo de operación fuera de línea para permitir que el conductor complete la entrega sin interrupciones. Esto implica el diseño de una arquitectura tolerante a fallos que pueda mantener la funcionalidad básica incluso en condiciones adversas de conectividad.
  - **Experiencia de usuario:** La interfaz de usuario debe ser intuitiva y fácil de usar para que los usuarios puedan solicitar entregas de mercancías de manera eficiente. Se deben incorporar principios de diseño centrados en el usuario para garantizar una experiencia satisfactoria desde el primer uso de la aplicación, incluyendo elementos como una navegación clara y una retroalimentación visual adecuada.
  - **Integración:** El sistema debe ser capaz de comunicarse de manera confiable con servicios externos, como el procesamiento de pagos, para garantizar que las transacciones se realicen de manera segura y sin problemas. Se deben implementar estándares y protocolos de comunicación robustos para facilitar la integración con sistemas externos y garantizar la interoperabilidad sin problemas.
  - **Escalabilidad:** El sistema debe ser capaz de escalar dinámicamente para manejar un aumento repentino en la demanda de entregas, especialmente durante eventos especiales u horas pico. Esto implica el diseño de una arquitectura modular y elástica que pueda asignar recursos de manera eficiente según sea necesario para garantizar un rendimiento óptimo incluso bajo cargas máximas.

  Es importante destacar que, como parte del proceso de diseño de la arquitectura, se han trabajado diversos escenarios que abordan aspectos críticos de calidad, así como la definición de técnicas y patrones arquitectónicos. Estos documentos han sido desarrollados de manera detallada para garantizar una comprensión completa de los requisitos y consideraciones arquitectónicas del proyecto. Para facilitar su revisión y consulta, los documentos que contienen los escenarios y las definiciones de técnicas y patrones serán agregados como archivos adjuntos al presente documento.

4.2. Aplicación de ATAM

En esta sesión, emplearemos la metodología ATAM para evaluar la arquitectura de nuestra aplicación móvil de transporte de mercancías en motocicletas. Nos centraremos en identificar escenarios críticos de calidad y analizar cómo la arquitectura propuesta aborda estos desafíos. Con la participación de expertos y stakeholders clave, buscamos identificar riesgos y oportunidades de mejora para garantizar el éxito del proyecto.

1. **Preparación:** Para iniciar la evaluación utilizando el método ATAM, primero seleccionaremos un equipo multidisciplinario compuesto por expertos en arquitectura de software y stakeholders clave del proyecto. A continuación, revisaremos exhaustivamente la documentación pertinente, como diagramas arquitectónicos y documentos de diseño, para comprender la arquitectura actual del sistema y establecer claramente el alcance de la evaluación. Luego, programaremos un workshop inicial en el que discutiremos los escenarios de calidad previamente definidos y cualquier otro aspecto relevante para la evaluación. Durante este workshop, utilizaremos herramientas de colaboración para facilitar la discusión y la recopilación de información. Finalmente, nos aseguraremos de contar con todas las herramientas y materiales necesarios para llevar a cabo la evaluación de manera efectiva, incluyendo pizarras, documentos de trabajo y cualquier otra cosa que pueda ser útil. Una vez completados estos pasos, estaremos listos para comenzar el proceso de evaluación utilizando el método ATAM.

2. **Brainstorming de escenarios de calidad:** Comenzamos reuniendo al equipo de evaluación y a los stakeholders clave en un workshop inicial. Durante esta sesión, discutimos y revisamos los escenarios de calidad previamente definidos, además de identificar cualquier otro escenario relevante para la evaluación. Utilizando herramientas de colaboración, como pizarras o software de trabajo en equipo, facilitamos la discusión y la generación de ideas. Cada participante aportó su conocimiento y perspectiva para garantizar una amplia cobertura de escenarios importantes. Al final de la sesión, habíamos recopilado una lista completa de escenarios de calidad que serían fundamentales para la evaluación de la arquitectura de la aplicación móvil de transporte de mercancías en motocicletas.

3. **Identificación de riesgos y trade-offs:** Tras haber revisado exhaustivamente la documentación existente y discutido los escenarios de calidad con el equipo de evaluación y los stakeholders, procedimos a identificar los riesgos y trade-offs asociados con la arquitectura. Durante esta fase, se destacaron varios riesgos potenciales, incluyendo posibles problemas de rendimiento debido al aumento del volumen de entregas durante períodos de alta demanda, así como preocupaciones sobre la seguridad de los datos del usuario, especialmente en lo que respecta a la protección contra accesos no autorizados. Además, se reconocieron trade-offs importantes, como la necesidad de equilibrar la usabilidad de la aplicación con la seguridad de los datos del usuario, y la necesidad de optimizar el rendimiento sin comprometer la eficiencia energética del dispositivo móvil. Estos riesgos y trade-offs fueron documentados y discutidos con el equipo de evaluación y los stakeholders para su consideración en la fase de generación de recomendaciones.

4. **Generación de recomendaciones:** Después de completar la evaluación utilizando el método ATAM, nuestro equipo multidisciplinario generó una serie de recomendaciones para mejorar la arquitectura de la aplicación móvil de transporte de mercancías en motocicletas. Estas recomendaciones se basaron en los hallazgos de la evaluación y se centraron en abordar los riesgos identificados y mejorar los escenarios de calidad. Se desarrollaron estrategias específicas para mejorar el rendimiento, la seguridad, la escalabilidad y la usabilidad del sistema. Además, se propusieron cambios en el diseño arquitectónico para optimizar la integración de servicios externos, como mapas de navegación y métodos de pago. Todas estas recomendaciones fueron presentadas a los stakeholders relevantes en una sesión dedicada, donde se discutieron posibles acciones a seguir y se establecieron planes para implementar las mejoras sugeridas.

4.3. Resultados de la evaluación

Una vez completada la evaluación utilizando el método ATAM, los resultados se encuentran reflejados en las técnicas y patrones arquitectónicos identificados durante el proceso. Estos hallazgos están documentados en detalle en los archivos adjuntos, proporcionando una visión clara de los puntos fuertes y áreas de mejora de la arquitectura actual.

Los patrones y técnicas de diseño definidos como resultado de la evaluación incluyen:

- **Replicas y balanceo de carga**: Esta técnica fue seleccionada para asegurar la alta disponibilidad y escalabilidad del sistema. Al replicar los servicios en múltiples nodos y distribuir las solicitudes entrantes de manera equitativa, se mejora la tolerancia a fallos y se garantiza un mejor rendimiento bajo cargas variables.

- **Circuit breaker**: Elegido para prevenir fallos cascada en la arquitectura, este patrón permite que los servicios se recuperen de fallos temporales sin afectar el sistema completo.

- **Detección y recuperación de fallos**: Implementar mecanismos de detección y recuperación rápida de fallos es crucial para minimizar el tiempo de inactividad y asegurar la continuidad del servicio. Esta técnica es esencial para mantener la confianza del usuario y la eficiencia operativa.

- **Escalado automático**: La capacidad de ajustar automáticamente los recursos del sistema en respuesta a la demanda asegura que el sistema pueda manejar picos de tráfico sin intervención manual.

- **Desacoplamiento de componentes**: Al diseñar el sistema de manera modular, se facilita el mantenimiento y la actualización de componentes individuales sin afectar el sistema completo. Este enfoque reduce el riesgo de fallos sistémicos y mejora la flexibilidad para futuras expansiones.

- **Bridge**: Este patrón facilita la extensión y variabilidad en las implementaciones sin modificar las interfaces. Es ideal para escenarios donde se anticipan cambios en los requisitos o en las tecnologías utilizadas, permitiendo una evolución más controlada de la arquitectura.

- **Caching**: Utilizar almacenamiento en caché reduce la latencia y mejora el rendimiento al minimizar el acceso a recursos lentos o costosos.
- **MFA (Autenticación Multifactor)**: La implementación de múltiples capas de autenticación mejora significativamente la seguridad del sistema, protegiendo contra accesos no autorizados y asegurando que solo usuarios legítimos puedan acceder a los recursos críticos.
- **Diseño responsivo**: Asegurar que la interfaz de usuario sea adaptable y funcione correctamente en una variedad de dispositivos y tamaños de pantalla es esencial para proporcionar una experiencia de usuario consistente y accesible, independientemente del dispositivo utilizado.

Estas técnicas y patrones no solo abordan los riesgos identificados, sino que también fortalecen la arquitectura del sistema, alineándola mejor con los objetivos del proyecto y las necesidades de los stakeholders. La documentación resultante proporciona una guía clara para futuras iteraciones de diseño y desarrollo, asegurando que la arquitectura evolucione de manera continua para satisfacer las demandas cambiantes del entorno operativo y los usuarios finales.

4.4. Recomendaciones

Después de completar la evaluación utilizando el método ATAM, se generaron recomendaciones específicas que están documentadas en detalle en los archivos adjuntos del ATAM para abordar los riesgos identificados y mejorar la calidad general de la arquitectura del sistema. Estas recomendaciones se basaron en las mejores prácticas y estrategias arquitectónicas discutidas durante los workshops de evaluación, y se diseñaron para alinear la arquitectura con los objetivos del proyecto y las necesidades de los stakeholders. El plan de acción incluyó pasos concretos para implementar las recomendaciones, asignando responsabilidades y plazos claros para cada tarea.

Se establecieron criterios de evaluación para medir el éxito de las acciones tomadas y garantizar la mejora continua de la arquitectura a lo largo del tiempo. Con este enfoque estructurado y centrado en la acción, el equipo pudo avanzar de manera efectiva hacia la implementación de cambios significativos en la arquitectura del sistema, garantizando su alineación con los requisitos del proyecto y las expectativas de los stakeholders.

4.5. Observaciones

Tras completar el proceso de evaluación utilizando el método ATAM, se derivaron varias observaciones fundamentales que arrojaron luz sobre la arquitectura propuesta y su implementación.

Se identificaron y documentaron diversos patrones y técnicas arquitectónicas que demostraron ser efectivos en la satisfacción de los escenarios de calidad definidos. Estos hallazgos no solo destacaron los aspectos positivos del diseño, sino que también señalaron oportunidades para optimizar y fortalecer aún más la arquitectura.

Es esencial destacar que la alineación con los objetivos del proyecto y las necesidades de los stakeholders se reconoció como un factor crucial en todo el proceso de diseño y desarrollo. Las

recomendaciones específicas generadas a partir de estas observaciones tienen como objetivo abordar los riesgos identificados y mejorar la calidad general de la arquitectura, manteniendo esta alineación como prioridad.

En resumen, la evaluación utilizando el método ATAM proporcionó una visión detallada y bien fundamentada de la arquitectura del sistema. Estas observaciones actúan como una guía invaluable para futuras iteraciones y mejoras, asegurando que la arquitectura evolucione de manera coherente y eficiente con las necesidades del proyecto y los stakeholders.

Capítulo 5

Conclusiones

- **Resumen de los hallazgos:**

  Durante el desarrollo de este proyecto, se buscó diseñar una arquitectura de aplicación móvil segura y efectiva para el transporte en motocicleta. Para lograr este objetivo, se establecieron objetivos específicos que guiaron el proceso de investigación y desarrollo.

  Cumplimiento de los objetivos:

  1. **Identificación de requisitos**: Se llevó a cabo una exhaustiva investigación para identificar los requisitos tanto funcionales como no funcionales de la aplicación móvil. Esto implicó un análisis detallado de las necesidades de los usuarios, así como de los requisitos de seguridad y rendimiento.
  2. **Documentación de requerimientos arquitectónicos**: Se establecieron y documentaron los requisitos arquitectónicos de la aplicación, incluyendo consideraciones de escalabilidad, seguridad y modularidad.
  3. **Definición de estructura arquitectónica**: Se diseñó una estructura arquitectónica sólida que cumple con los requisitos identificados. Esto incluyó la definición de capas de aplicación, la separación de preocupaciones y la elección de patrones arquitectónicos adecuados para el contexto del proyecto.
  4. **Determinación de componentes específicos**: Se identificaron y definieron los componentes específicos necesarios para implementar la arquitectura, como la capa de presentación, la capa de lógica de negocio y la capa de acceso a datos.
  5. **Pruebas y validaciones**: Se realizaron pruebas para garantizar la estabilidad y viabilidad de la arquitectura. Esto se incluyó dentro del proceso ATAM.

- **Relación con la literatura existente**

  El diseño de la arquitectura para nuestra aplicación de transporte de mercancía en motocicletas se ha enriquecido significativamente gracias a una exploración detallada de la literatura relevante. Al sumergirnos en estudios previos, hemos obtenido una perspectiva amplia y diversa que ha influido en nuestra comprensión y enfoque del proyecto.

  Nos hemos inspirado en la investigación de Law et al. (2023), que resalta la importancia de considerar la propiedad de motocicletas en el contexto de la movilidad urbana y la desigualdad de ingresos. Esta mirada hacia las dinámicas sociales y económicas nos ha llevado a priorizar la accesibilidad y equidad en nuestra arquitectura.

Además, el trabajo de Martin et al. (2023) nos ha guiado al explorar las tendencias de digitalización y electrificación en ciudades en desarrollo, con un enfoque particular en los servicios de transporte en motocicletas. Esta investigación nos ha ayudado a comprender las tecnologías emergentes que podrían integrarse en nuestra arquitectura para adaptarse a las necesidades específicas de estos contextos.

También hemos tomado nota de la perspectiva presentada por Puliafito et al. (2021) sobre las ciudades inteligentes como sistemas ciberfísicos, y cómo estas pueden influir en el diseño de soluciones de gestión de transporte más eficientes y sostenibles. Integrar sistemas avanzados de información y comunicación se ha convertido en una piedra angular de nuestra arquitectura.

En términos de implementación práctica, la investigación de Reis et al. (2022) sobre Docker y Docker Compose ha sido invaluable. Esta exploración nos ha proporcionado una comprensión más profunda de la gestión de contenedores y la orquestación de servicios, aspectos cruciales para el desarrollo de nuestra infraestructura tecnológica.

Además, aunque no directamente relacionados con nuestro enfoque principal, los estudios de Hoffmann Souza et al. (2021), Hoffmann Souza et al. (2021) y Love and Vyas (2022) han ampliado nuestra comprensión del panorama general de la gestión de transporte urbano. Estos aportes nos han ayudado a contextualizar nuestro proyecto dentro de un marco más amplio, considerando los desafíos y oportunidades del entorno urbano en general.

En resumen, nuestra exploración de la literatura existente ha sido fundamental para informar y enriquecer el diseño de nuestra arquitectura de transporte de mercancías en motocicletas. Al integrar conocimientos y perspectivas de una variedad de fuentes, hemos podido desarrollar una solución sólida y adaptable que aborda las complejidades del transporte urbano contemporáneo.

- **Implicaciones prácticas**

  Durante el desarrollol proyecto se identificaron varios hallazgos que tienen importantes implicaciones prácticas para la implementación de políticas en el desarrollo de software. A continuación, se discuten algunas recomendaciones y sugerencias derivadas de estos hallazgos:

  - Optimización del rendimiento del sistema: Uno de los principales hallazgos de esta investigación fue la identificación de ciertos patrones arquitectónicos que contribuyen significativamente a la mejora del rendimiento del sistema. Se recomienda encarecidamente a los profesionales del desarrollo de software considerar la adopción de estos patrones en sus proyectos para garantizar un rendimiento óptimo del sistema, especialmente en entornos de alta demanda.
  - Seguridad y robustez del sistema: La investigación también reveló la importancia de ciertas prácticas arquitectónicas en la mejora de la seguridad y la robustez del sistema. Se sugiere que las organizaciones y los equipos de desarrollo de software integren de manera proactiva estas prácticas en sus procesos de desarrollo para mitigar riesgos de seguridad y mejorar la fiabilidad del sistema.

- Escalabilidad y mantenibilidad del código: Otro hallazgo destacado fue la influencia de ciertas decisiones arquitectónicas en la escalabilidad y mantenibilidad del código. Se insta a los profesionales del desarrollo de software a considerar cuidadosamente estas implicaciones al diseñar la arquitectura de sus sistemas, priorizando la escalabilidad y la capacidad de mantenimiento a largo plazo.
- Adopción de tecnologías emergentes: La investigación también resaltó la importancia de mantenerse al tanto de las tendencias y tecnologías emergentes en el campo de la arquitectura de software. Se recomienda a las organizaciones y a los profesionales del desarrollo de software estar abiertos a la adopción de nuevas tecnologías que puedan ofrecer ventajas significativas en términos de eficiencia, rendimiento y seguridad.
- Colaboración interdisciplinaria y buenas prácticas de desarrollo: Finalmente, se enfatiza la necesidad de fomentar la colaboración interdisciplinaria y la adopción de buenas prácticas de desarrollo en todos los aspectos del ciclo de vida del software. Esto incluye la comunicación efectiva entre equipos, la documentación adecuada y la realización de pruebas exhaustivas para garantizar la calidad del software entregado.

- **Limitaciones de la investigación**

  A pesar de los esfuerzos realizados durante el desarrollo de este proyecto de investigación en arquitectura de software, es importante reconocer y discutir las limitaciones que pueden haber influido en la validez y generalización de los hallazgos. Las principales limitaciones identificadas incluyen:

  - Restricciones de tiempo: Una de las limitaciones principales de este estudio fue el tiempo disponible para la ejecución del proyecto. La duración limitada del proyecto pudo haber afectado la profundidad y el alcance de la investigación, así como la capacidad para realizar análisis más exhaustivos y realizar experimentos adicionales.
  - Disponibilidad de recursos: La disponibilidad limitada de recursos, tanto humanos como financieros, también representó una limitación para este estudio. Esto podría haber influido en la selección de las técnicas de investigación y en la amplitud de las actividades realizadas durante el proyecto.
  - Complejidad del dominio: La naturaleza compleja del dominio de la arquitectura de software también presentó desafíos durante el desarrollo de la investigación. La comprensión completa de todos los aspectos del dominio puede haber sido difícil de lograr en el marco de este proyecto, lo que podría haber limitado la profundidad de los análisis realizados.
  - Limitaciones en la metodología: Si bien se seleccionó cuidadosamente una metodología de investigación apropiada, como los estudios de casos o ATAM, es importante reconocer que ninguna metodología es perfecta. Las limitaciones inherentes a la metodología utilizada, como posibles sesgos o limitaciones en la generalización de los resultados, deben ser tenidas en cuenta al interpretar los hallazgos.

- Limitaciones en la recopilación de datos: La recopilación de datos puede haber estado sujeta a ciertas limitaciones, como la disponibilidad de datos históricos o la calidad de los datos recopilados. Estas limitaciones podrían haber influido en la precisión y la exhaustividad de los análisis realizados durante el estudio.

- **Sugerencias para futuras investigaciones**

  Se pueden hacer varias sugerencias para futuras investigaciones que podrían ampliar y mejorar el conocimiento en este campo. Algunas áreas que podrían beneficiarse de investigaciones adicionales incluyen:

  - Análisis de la evolución del software a largo plazo: Sería interesante realizar estudios longitudinales para analizar la evolución de la arquitectura de software a lo largo del tiempo, especialmente en sistemas de larga duración y en entornos empresariales complejos. Esto permitiría comprender mejor cómo cambian y se adaptan las arquitecturas de software en respuesta a los cambios en los requisitos comerciales y tecnológicos.
  - Investigación sobre prácticas de desarrollo colaborativo: Dada la importancia de la colaboración y la comunicación efectiva en el desarrollo de software, se sugiere investigar más a fondo las prácticas de desarrollo colaborativo y su impacto en la calidad y el rendimiento del software. Esto podría incluir el estudio de metodologías ágiles, herramientas de colaboración y técnicas de gestión de proyectos.
  - Análisis de la seguridad y la privacidad en arquitecturas distribuidas: Con el creciente uso de arquitecturas distribuidas y sistemas en la nube, sería relevante investigar más sobre cómo garantizar la seguridad y la privacidad en estos entornos. Esto podría implicar la evaluación de técnicas de autenticación, autorización y cifrado en arquitecturas distribuidas, así como el estudio de posibles vulnerabilidades y amenazas.

- **Anexos**

  Presentamos los anexos que complementan el desarrollo y diseño de nuestra arquitectura de software propuesta. Estos documentos proporcionan detalles adicionales sobre los requisitos, el análisis de entrevistas y encuestas, así como otros aspectos relevantes para la comprensión y evaluación de nuestro proyecto.

  Los anexos incluidos abarcan una variedad de materiales que respaldan nuestro proceso de diseño y toma de decisiones, incluyendo: documentos técnicos, diagramas arquitectónicos, y enlaces relacionados con recursos adicionales. Además, para facilitar el acceso a estos recursos, los lectores pueden seleccionar el texto correspondiente para acceder directamente a los enlaces relacionados:

  - Requisitos funcionales: En este documento detallamos las funciones y capacidades que nuestro sistema debe proporcionar para satisfacer las necesidades del usuario y del negocio.

- Requisitos no funcionales: Aquí describimos los aspectos de calidad de nuestro sistema, como la usabilidad, el rendimiento y la seguridad, que influyen en su diseño y desarrollo.
- Encuestas y entrevistas: Presentamos los resultados y análisis de las encuestas y entrevistas que realizamos durante la fase de investigación, proporcionando información crucial para la definición de requisitos y la toma de decisiones de diseño.
- Documentos relacionados con la aplicación de ATAM: Este conjunto de documentos incluye las evaluaciones y resultados obtenidos mediante la aplicación del método ATAM, que nos ayudaron a identificar y abordar los riesgos arquitectónicos del sistema propuesto.

Los anexos se presentan con el objetivo de brindar una visión más completa y detallada de nuestro proceso de diseño de la arquitectura de software, así como para respaldar la validez y la robustez de las decisiones tomadas a lo largo del proyecto.

Bibliografía

Aarón, M. A., Gómez, C. A., Fontalvo, J., and Gómez, A. J. (2019). Análisis de la movilidad vehicular en el departamento de la guajira usando simulación. el caso de riohacha y maicao. *Informacion Tecnologica*, 30:321–332.

Abirami, T., Mapari, S., Jayadharshini, P., Krishnasamy, L., and Vigneshwaran, R. R. (2023). 2023 international conference on advances in computation, communication and information technology (icaiccit). In *Streamlined Deployment and Monitoring of Cloud-Native Applications on AWS with Kubernetes Prometheus Grafana*, pages 1149–1155.

Andrade, H., Berger, C., Crnkovic, I., and Bosch, J. (2020). Principles for re-architecting software for heterogeneous platforms. *Proceedings - Asia-Pacific Software Engineering Conference*, 2020-December:1.

Beecroft, M. (2019). The future security of travel by public transport: A review of evidence. *Research in Transportation Business Management*, 32:100388. The future of public transport.

Block, A. and Soto, A. (2022). *Kubernetes Secrets Management*. Maning.

Bundela, R., Dhanda, N., and Verma, R. (2022). Load balanced web server on aws cloud. In *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 114–118. IEEE.

Butler, L., Yigitcanlar, T., and Paz, A. (2020). Smart urban mobility innovations: A comprehensive review and evaluation. *IEEE Access*, 8:196034–196049.

Cowell, C., Lotz, N., and Timberlake, C. (2023). *Automating DevOps with GitLab CI/CD Pipelines: Build efficient CI/CD pipelines to verify, secure, and deploy your code using real-life examples*. Packt Publishing.

di Vimercati, S. D. C., Facchinetti, D., Foresti, S., Livraga, G., Oldani, G., Paraboschi, S., Rossi, M., and Samarati, P. (2023). Scalable distributed data anonymization for large datasets. *IEEE Transactions on Big Data*, 9 (3):818–831.

Feng, L. and Wei, W. (2019). An empirical study on user experience evaluation and identification of critical ux issues. *Sustainability (Switzerland)*, 2432:11.

Hoffmann Souza, M. L., da Costa, C. A., de Oliveira Ramos, G., and da Rosa Righi, R. (2021). A feature identification method to explain anomalies in condition monitoring. *Computers in Industry*, 133:103528.

Law, T. H., Ng, C. P., Poi, A. W. H., Guo, B., and Hu, Q. (2023). Motorcycle-to-car ownership at different urbanization levels: Road mobility, accessibility, and income inequality. *Transportation Research Part D: Transport and Environment*, 122:103879.

Londoño, D., Palacio, E., Preciado, A., Henao, E., Salazar, J., Balbín, D., Restrepo, D., Mira, J., and Gutiérrez, F. (2021). Criminal behavior in medellin in times of the pandemic crisis, a spatio-temporal model. *Estudios Gerenciales*, 37 (159):188–199.

Love, C. and Vyas, J. (2022). *Core Kubernetes*. Manning.

Maratkar P, A. P. (2021). React js - an emerging frontend javascript library. *Revistas IRE*, 12:99–102.

Martin, E., Courtright, T., Nkurunziza, A., and Lah, O. (2023). Motorcycle taxis in transition? review of digitalization and electrification trends in selected east african capital cities. *Case Studies on Transport Policy*, 13:101057.

Puliafito, A., Tricomi, G., Zafeiropoulos, A., and Papavassiliou, S. (2021). Smart cities of the future as cyber physical systems: Challenges and enabling technologies. *Sensors*, 21 (10).

R, M. (2011). Node. js paradigms and benchmarks. *Revistas IRE*, 11:1–6.

Reis, D., Piedade, B., Correia, F. F., Dias, J. P., and Aguiar, A. (2022). Developing docker and docker-compose specifications: A developers' survey. *IEEE Access*, 10:2318–2329.

Sai K, V. A. (2017). Comparative study of structured and unstructured data using mongo db. *Advanced Research in Dynamical and Control Systems*, 9:Special issue 17.

Samora Marchele Obudho, Wilson A.P. Otengah, T. I. S. (2020). Motorcycle taxi in addressing the rural transport conundrum. *TRANSPORT CONUNDRUM*, 7:73–84.