UNIVERSIDAD NACIONAL DE SAN JUAN
FACULTAD DE CIENCIAS EXACTAS FÍSICAS Y NATURALES
DEPARTAMENTO DE INFORMÁTICA
LICENCIATURA EN SISTEMAS DE INFORMACIÓN

TRABAJO FINAL

FARMADIP: PLATAFORMA DIGITAL PARA LA GEOLOCALIZACIÓN DE FARMACIAS Y SOLICITUD DE MEDICAMENTOS MEDIANTE RECETAS ELECTRÓNICAS

Autora: Zamira Nahir Alé García

Director: Mg. Lic. Juan A. Aranda Romera

San Juan, 2026

A mis padres,
por haber sembrado en mí el amor por el conocimiento,
por su esfuerzo silencioso y su apoyo incondicional a lo largo de toda mi vida.
Este logro es también fruto de su entrega y sacrificio.

AGRADECIEMIENTOS

En primer lugar, agradezco a la Universidad Nacional de San Juan por brindarme el espacio académico y humano que hizo posible mi formación profesional y la concreción de esta tesis.

A la Facultad de Ciencias Exactas, Físicas y Naturales, por su compromiso con la excelencia académica y por acompañar mi desarrollo a lo largo de la carrera.

Al Departamento de Informática, por su acompañamiento institucional y por formar parte fundamental de este recorrido académico.

Al Mg. Lic. Juan Aranda, profesor tutor de esta tesis, por su generosidad al brindarme la oportunidad de recurrir a su capacidad y experiencia científica en un marco de confianza, sinceridad y apoyo constante, fundamentales para la realización de este trabajo.

A mi esposo, Sergio Martínez, quien me sostuvo cuando me derrumbé, me levantó cuando caí y me impulsó cuando quise detenerme. Su amor, su fe en mí y su compañía incondicional fueron pilares esenciales en este proceso.

A mi hermana Zoe y a su familia, que desde la distancia me apoyaron siempre.

A mi compañero Hernán, que estuvo a mi lado hasta el final, en este camino de aprendizaje.

A mis profesores, quienes con dedicación y compromiso trazaron junto a mí el camino que me condujo hasta este logro.

Y, finalmente, agradezco a mis mascotas, Samir, Willy, Blacky, Lobo, Gohan y Ali por su compañía silenciosa en cada jornada de estudio, en cada desvelo y en cada momento de incertidumbre. Su presencia fue refugio, calma y amor incondicional durante todo este proceso.

INDICE DE FIGURAS

I: INTRODUCCIÓN

1.1 Resumen y Palabras Clave

El acceso a farmacias disponibles y el acceso a medicamentos presentan desafíos en la provincia de San Juan, especialmente en zonas alejadas. Este estudio propone el desarrollo de una plataforma que permite a los usuarios localizar farmacias disponibles según su ubicación y, de manera opcional, solicitar medicamentos a través de recetas electrónicas.

Mediante un diseño exploratorio-descriptivo, se analizan las necesidades de los usuarios y farmacias locales, integrando herramientas como sistemas de información geográfica (SIG) que permitan la geolocalización y visualización de farmacias. Además, se considera la normativa vigente, incluyendo la Ley 27.553 (Ver Anexo I - Ley 27553) y el Decreto 345/2024 (Ver Anexo II - Decreto 345/2024). La investigación se enmarca en una lógica cualitativa, enfocada en comprender la interacción entre usuarios y tecnología.

Como resultado, se espera la implementación de una plataforma que optimice el acceso a farmacias y el acceso a medicamentos recetados en la región.

Palabras Clave

FARMADIP – GEOLOCALIZACIÓN FARMACIAS – ARQUITECTURA SOFTWARE – MODELO C4

Abstract

Access to available pharmacies and medications presents significant challenges in the province of San Juan, particularly in remote areas. This study proposes the development of a platform that allows users to locate available pharmacies based on their location and, optionally, request medications using electronic prescriptions.

Using an exploratory-descriptive design, the study analyzes the needs of local users and pharmacies by integrating tools such as Geographic Information Systems (GIS) for geolocation and visualization. Furthermore, the research considers current

regulations, including Law 27,553 and Decree 345/2024. The study is framed within a qualitative logic, focusing on understanding the interaction between users and technology.

As a result, the project anticipates the implementation of a platform that optimizes access to pharmacies and prescription medications throughout the region.

KEYWORDS

FARMADIP – PHARMACY GEOLOCATION – SOFTWARE ARCHITECTURE – C4 MODEL

1.2. CONTEXTUALIZACIÓN

1.2.1 ANTECEDENTES

En esta sección se presentan los antecedentes más relevantes relacionados con las tecnologías de geolocalización, las soluciones existentes en el ámbito farmacéutico y los enfoques metodológicos empleados en su diseño y operación. Este análisis permite situar el problema en su marco conceptual y tecnológico, identificar brechas no resueltas y fundamentar la necesidad del desarrollo de la plataforma propuesta.

Herramienta cartográfica digital basada en XML para la ciudad de La Plata (Nadia Kreimer, Mauricio Gende - 2010).

Se elabora una aplicación específica basada en la implementación de la API de Google Maps y un lenguaje de alto nivel (Octave). Estas son combinadas con código original para presentar farmacias cercanas a un domicilio, como así también aquellas que se encuentran de turno. En este trabajo se enfatiza en la búsqueda de la solución numérica al problema de la determinación geográfica del domicilio del usuario de la aplicación, haciendo hincapié en el desarrollo de un SIG (conjunto de herramientas informáticas que permiten recopilar, almacenar, analizar y visualizar datos geográficos) que permita resolver el problema específico de hallar las tres farmacias más cercanas respecto de la ubicación arbitraria de una persona en La Plata.

Sin embargo, este análisis e implementación no especifica qué información se proporciona de cada farmacia además de su ubicación, como así también limita la cantidad solo a las 3 más cercanas, no especifica qué sucedería si el usuario desea recurrir a otras farmacias que se encuentren de turno. Así mismo este desarrollo está limitado al casco urbano de La Plata.

FarMap – Matías Ignacio Rojo.

Aplicación disponible para Android y IOS, posee una versión web. Ofrece una lista precargada desde donde se puede seleccionar una provincia y una localidad. Una vez seleccionados, muestra una lista de farmacias que se encuentran de turno, proporcionando nombre, dirección y hora límite de atención. Al hacer click en un item (farmacia) de la lista, el usuario es llevado al servicio externo de Google Maps con la ubicación de la farmacia seleccionada.

La aplicación carece de posibilidades de elaborar pedidos de medicamentos, no proporciona datos de las farmacias de turno (como número de teléfono, obras sociales que dicha farmacia recibe, horarios normales de atención, teléfono de contacto, etc). Tiene indicativos de colores para las farmacias, pero no posee referencias para el usuario. Por último, pero no menos importante, se encuentra restringido a algunas localidades de la provincia de Buenos Aires, no se encuentra disponible para la provincia en su totalidad, ni para otras provincias.

Farmacia de turno ahora – M.A.S.

Aplicación web, disponible como aplicación móvil. Proporciona un buscador donde, una vez seleccionada la provincia, partido y localidad, muestra una lista de farmacias de turno, mostrando nombre, teléfono, dirección, localidad y un contador que muestra cuánto tiempo falta para finalizar la guardia o turno de dicha farmacia, como así también un botón bajo la leyenda "Cómo Llegar", el cual externaliza el servicio a Google Maps, proporcionando la ubicación de dicha farmacia en este servicio externo.

Esta aplicación no muestra datos como obras sociales, no brinda la posibilidad de realizar compras de medicamentos o pedidos a domicilio de estos. Por último, solo está restringido a la provincia de Buenos Aires, no se encuentra disponible para otras provincias.

Farmacia Aperta - Federfarma Lombardia.

Disponible como página web o como app para Android y IOS. Ofrece una lista de farmacias, ubicadas visualmente en un mapa, pudiendo seleccionarlas desde el mismo. También ofrece una serie de filtros entre los que destacan: estudios realizados o servicios prestados, aquellas que se encuentran de turno, farmacias cerradas, etc. Al seleccionar una farmacia, se puede visualizar información de esta: nombre, teléfono, correo, dirección o ubicación, horario semanal, etc.

Sin embargo, esta aplicación se encuentra limitada bajo las leyes y normativas del país de Lombardía, no se encuentra disponible para otros servicios que no sean farmacéuticos, o para otras regiones diferentes a la de origen (Lombardía).

Vitau, farmacia digital - VITAU MEDICAL, S.A.P.I. DE C.V.

Farmacia digital de Mexico, proporciona venta online de insumos de diversas categorías para personas comunes, como así también venta de medicamentos desde consultorios clínicos. Brinda también una plataforma solo para médicos, orientado a emitir recetas digitales, informes médicos, y medicamentos a domicilio para sus pacientes.

Sin embargo, esta aplicación se rige bajo las leyes y normativas del país de México, como así también solo se restringe a aquellas farmacias que se encuentran en el mencionado país. Así mismo no proporciona aquellas farmacias que se encuentran de turno fuera del horario comercial que, debido al país, puede diferir al de Argentina.

1.2.2 Formulación del Problema y Justificación

Las farmacias han sido, a lo largo de la historia, un recurso fundamental en la vida de las personas, ya que proporcionan insumos y medicamentos esenciales para preservar la salud de la población. En Argentina, mediante la resolución N°139 de la Secretaría de Política y Regulación de Salud, sancionada el 8 de octubre de

1997 y actualizada mediante la resolución 192/98 de la Secretaría de Política y Regulación de Salud, sancionada el 19 de agosto de 1998, se creó una Comisión encargada de evaluar y proponer aquellas normativas que las farmacias deben cumplir para su habilitación y funcionamiento. Dentro de dichas normativas, se establece la declaración de turnos de atención: Según la Resolución 192/98, "Si cumplirá turnos voluntarios y/u horario extendido o de VEINTICUATRO (24) horas" (Secretaría de Política y Regulación de Salud, 1998, art. 1, inc. V).

A nivel nacional, el ejercicio profesional farmacéutico y el expendio de medicamentos se encuentran regulados por la Ley 17.565 (1967) (Ver Anexo IV - Ley 17565), que establece las condiciones de habilitación y funcionamiento de farmacias. En la Provincia de San Juan, el Código Sanitario (Ley 67-Q, 2014) dispone que las farmacias deberán cumplir turnos de atención conforme a lo establecido por la autoridad sanitaria provincial (Ver Anexo V - Ley Provincial 67-Q).

Sin embargo, a pesar de las exigencias bajo ley para garantizar la atención farmacéutica a la población, existen zonas alejadas en las que la atención en horarios nocturnos o la disponibilidad de medicamentos no se encuentra garantizada. Debido a esto, la población debe recurrir a farmacias que se encuentran a muchos kilómetros de distancia.

Como ejemplo, al sur del departamento de Pocito, en la Provincia de San Juan, Argentina, se encuentra una localidad llamada Carpintería. Dicha localidad se divide en dos sectores principales, donde se concentra la mayoría de sus habitantes: La "Villa Cabecera" de Carpintería, y la zona comprendida entre calle N°18 y Castro Padin, y barrios alejados como el Barrio José Ramírez o Barrio Ruta 40, contando con 5Km aproximadamente de un sector a otro. Estas zonas de la población cuentan con una única farmacia, ubicada en la "Villa Cabecera" de Carpintería; la misma posee atención en "horarios normales" o los denominados "horarios de comercio", sin embargo, no cuenta con turnos rotativos o atención 24h, según la Resolución 192/98, "La Autoridad Sanitaria podrá eximir del cumplimiento del turno obligatorio a las farmacias que así lo soliciten en razón de hallarse a menos de UN MIL METROS (1.000 m) de otras que funcionen las VEINTICUATRO (24) horas o cumplan turnos voluntarios. El incumplimiento del turno obligatorio declarado será considerado como falta grave desde el punto de vista

sanitario, aplicándose las sanciones previstas en el Decreto N° 341/92" (Secretaría de Política y Regulación de Salud, 1998, art. 9).

Considerando que la farmacia más próxima a la localidad de Carpintería, que cumple con turnos rotativos de 24h, se encuentra a 18Km aproximadamente de la "Villa Cabecera" y a 22Km aproximadamente del "Barrio Ruta 40", no existe una farmacia cercana que se encuentre en un turno de 24h ante una emergencia, como así mismo no se cuenta con otra opción cercana en caso de que la primera opción no cuente con el medicamento solicitado, no reciban la obra social del paciente o cliente, o no cuenten con un servicio como colocación de inyectables o vacunación.

Se analizan desarrollos de sistemas que cumplen únicamente con la necesidad de conocer la ubicación de farmacias que se encuentran de turno, brindando información acerca de la misma. Sin embargo, la información ofrecida es escasa y solo se limita a la provincia de Buenos Aires o incluso a países o continentes diferentes que no podrían ser considerados por los diversos marcos legales. Así mismo, los desarrollos nacionales no abarcan la provincia de San Juan ni la solicitud de medicamentos, por lo que resultan insuficientes para esta problemática.

Se observan en redes sociales publicaciones, en diarios digitales, de un cronograma que indica qué farmacias se encuentran de turno en el departamento Pocito, indicando solamente día y nombre de dicha Farmacia. También, bajo normativa, se pueden encontrar expuestas en la puerta de las farmacias, una lista con aquellas que estarán de turno durante los próximos días, indicando día, nombre y dirección. Sin embargo, esto implica que el paciente deba desplazarse hasta la farmacia disponible, consultar por factores como la obra social o medicamentos disponibles, y, de no cumplir con estas expectativas, debe desplazarse nuevamente hasta encontrar una farmacia disponible que cumpla con lo deseado.

Las farmacias contribuyen a un pilar esencial en la salud de los residentes de la provincia de San Juan. Sin embargo, la disponibilidad de su información como así también su ubicación, disponibilidad de medicamentos y la ausencia de

facilidad de acceso o llegada a las mismas desde departamentos alejados, dificulta con creces el acceso a los servicios farmacéuticos.

La presente investigación pretende abordar la problemática del usuario al momento de recurrir a una farmacia que cumpla con estas necesidades (que se encuentre abierta al público, medicación disponible, recepción de obra social, cercana a su ubicación geográfica o que ofrezca envío a domicilio). De esta manera, el usuario podrá recurrir a la farmacia que cubra con estos aspectos.

1.3 OBJETIVOS

Objetivo General

Desarrollar una plataforma digital para la geolocalización de farmacias y solicitud de medicamentos mediante recetas electrónicas.

Objetivos Específicos

- Relevar las farmacias en localidades seleccionadas de San Juan.
- Estudiar y analizar normativas legales referentes a recetas electrónicas.
- Investigar frameworks necesarios o relevantes para el desarrollo de la plataforma.
- Implementar aquellos frameworks seleccionados para el desarrollo de la plataforma.
- Diseñar una interfaz simple para Clientes y Farmacias que acompañe y facilite la usabilidad de la plataforma.
- Desarrollar un prototipo básico inicial que pueda representar el objetivo principal de la plataforma.

ESTRUCTURA DEL DOCUMENTO

El presente trabajo final se ha estructurado en cinco capítulos. El primero es una introducción sobre la temática que se aborda, incluyendo la problemática planteada, antecedentes relacionados y los objetivos: general, y específicos; del presente trabajo final.

En el Capítulo II – Marco Teórico, se realiza una recopilación teniendo en cuenta la gestión de información propia de la investigación en sí, como así también, las tecnologías de trabajo para el desarrollo de la solución propuesta. Los temas abordados son: arquitectura de sistemas y arquitectura software, estándares internacionales actuales relacionados con la arquitectura, arquitecturas de desarrollo, plataformas de implementación de sistemas, con el objeto de Diseñar la solución propuesta en esta tesis.

El Capítulo III – Problemática, describe la situación actual del problema bajo estudio, centrado en la comercialización y distribución de medicamentos en departamentos alejados, los cuales no cuentan con farmacias abiertas durante las 245 horas, como ocurre en el Departamento Capital y alrededores.

En el Capítulo IV, se aborda una propuesta de solución al problema tratado, aplicando una metodología de desarrollo de software basado en la arquitectura del sistema, lo que favorece la generación de un producto bajo ciertos estándares de calidad y considerando las principales preocupaciones (concerns) relacionados con los atributos de calidad y rendimiento.

En el Capítulo V, se enuncian las conclusiones del presente trabajo, como así también posibles líneas futuras de acción a seguir.

En el Capítulo VI, se incluyen las referencias bibliográficas citadas en la tesis, y bibliografía de consulta.

Finalmente, hay una sección denominada ANEXOS, en la cual se adjuntan una serie documentos que facilitan la comprensión del presente informe.

II – MARCO TEÓRICO

2 ARQUITECTURA DE SISTEMAS Y ARQUITECTURA DE SOFTWARE

2.1 DEFINICIÓN DE ARQUITECTURA

La Arquitectura de Software es la organización fundamental de un sistema software, expresada a través de sus componentes, sus relaciones (entre ellos y con su entorno), junto con los principios que guían su diseño y evolución a lo largo del ciclo de vida. (ISO/IEC/IEEE, 2023)

La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo las tareas o procesos computacionales, sus interfaces y la comunicación entre ellos. Toda arquitectura lógica debería poder ser implementada mediante una arquitectura física, que consiste en determinar qué dispositivos de procesamiento tendrán asignados cada tarea. (ISO/IEC/IEEE 42010:2022 - Software, 2022)

Características de la arquitectura de software:

- Describe los elementos estructurales que componen el sistema (como módulos, servicios, clases, etc.).
- Define las relaciones e interacciones entre esos elementos (como dependencias, comunicaciones, flujos de datos).
- Incluye principios y directrices de diseño (como patrones arquitectónicos, restricciones tecnológicas o decisiones de calidad).
- Considera Requisitos Funcionales, Requisitos No Funcionales y Restricciones.
- Considera la evolución y el mantenimiento del sistema en el tiempo.

Una arquitectura de software se selecciona y diseña con base en requisitos establecidos para el sistema de información. Se consideran tanto requisitos funcionales, como requisitos no funcionales y restricciones. (como rendimiento, escalabilidad, mantenimiento, seguridad, e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más

recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real.

2.2 Estándares de Arquitectura de Sistemas y de Software

2.2.1 ISO/IEC/IEEE 42010

Según la norma ISO/IEC/IEEE 42010:2022, que es el estándar internacional para la descripción de arquitecturas de sistemas y software, el concepto de arquitectura de software se enmarca dentro del concepto más general de arquitectura de sistema, aplicándose específicamente cuando el sistema bajo estudio es puramente el software. Esta norma, establece principios, requisitos y definiciones para la descripción de arquitecturas de sistemas y software. Define conceptos clave de la arquitectura de sistemas y de software. La norma, considera a la "arquitectura de software" como parte de la "arquitectura del sistema" cuando el sistema en cuestión es un sistema de software; y representa las decisiones estructurales y de comportamiento clave que influyen en el sistema completo. (ISO/IEC/IEEE 42010:2022 - Software, 2022)

2.2.2 ISO/IEC/IEEE 15288 e ISO/IEC/IEEE 12207

Las normas ISO/IEC/IEEE 12207 e ISO/IEC/IEEE 15288, son los estándares internacionales que definen los procesos adoptados en Ingeniería de Sistemas e Ingeniería de Software: (ISO/IEC/IEEE, 2023)

- ISO/IEC/IEEE 15288:2023 - Procesos del Ciclo de Vida de Sistemas
- ISO/IEC/IEEE 12207:2017 - Procesos del Ciclo de Vida del Software

Ambas normas (unificadas) adoptan la misma estructura general de 30 procesos, agrupados en 4 categorías:

1. Procesos de acuerdo (2)
2. Procesos organizacionales facilitadores de proyecto (6)
3. Procesos de gestión técnica (8)
4. Procesos técnicos (14)

Entre los 14 procesos técnicos, que son los procesos específicos de desarrollo de sistemas/software, proponen un proceso relacionado con la Arquitectura:

Proceso de definición de la Arquitectura

Proceso Técnico que busca establecer una solución estructurada que satisfaga los requisitos del sistema o software. Define la estructura organizativa de los elementos, sus funciones, interfaces y relaciones, considerando factores como desempeño, seguridad, reutilización, escalabilidad y mantenibilidad. La arquitectura guía el diseño posterior y permite la evaluación temprana de riesgos. (Sommerville, 2021)

2.3 CONCEPTOS PRINCIPALES ASOCIADOS A LA ARQUITECTURA (SEGÚN ISO/IEC/IEEE 42010:2022)

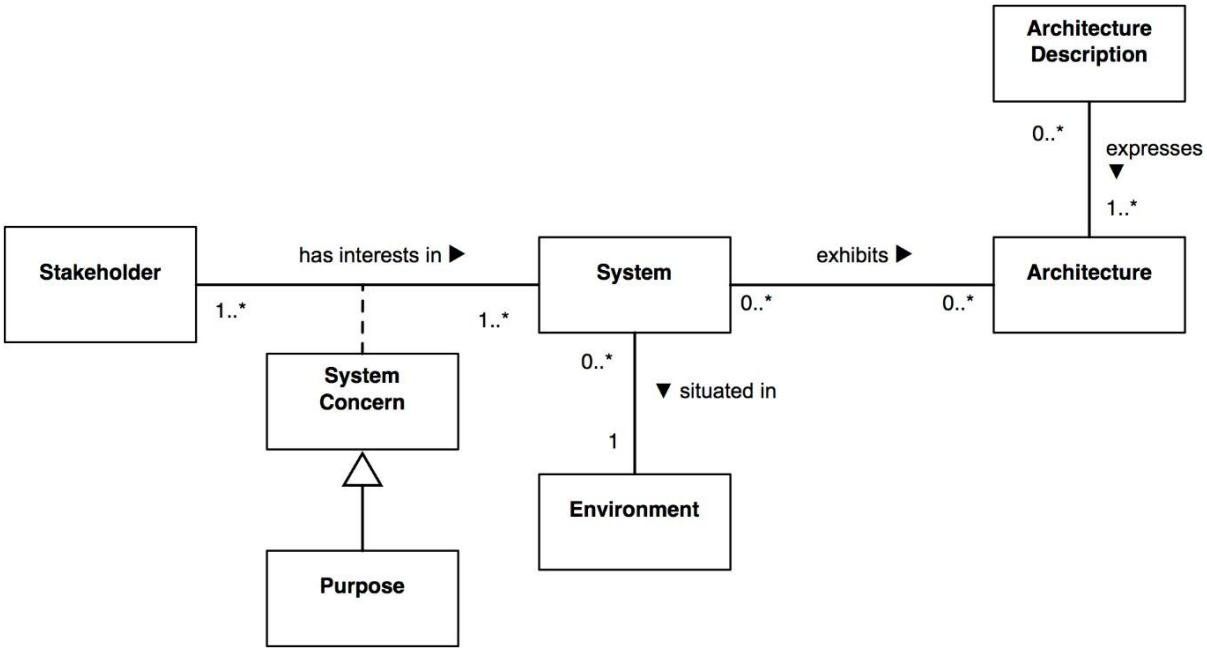

Figura 1: Arquitectura de Sistemas

1. **Parte interesada (Stakeholder):**

Toda persona, grupo u organización con interés legítimo en las decisiones arquitectónicas del sistema. Ejemplos: arquitectos, desarrolladores, usuarios, clientes, reguladores.

2. **Preocupación (Concern):**

   Aspecto del sistema que es relevante para uno o más stakeholders y que puede afectar o verse afectado por decisiones arquitectónicas. Ejemplos: seguridad, escalabilidad, interoperabilidad, mantenibilidad, rendimiento. Las preocupaciones o concerns se determinan a partir de los requisitos no funcionales.

3. **Punto de vista (Viewpoint):**

   Es un conjunto prescriptivo de convenciones para construir, interpretar y analizar una o más vistas. Define los interesados, preocupaciones, modelos a usar, técnicas de análisis y convenciones de presentación.

   Por ejemplo, un punto de vista lógico puede enfocarse en las relaciones entre componentes de software para abordar preocupaciones de modularidad y comprensión.

   Especificación que define cómo construir y analizar una vista, considerando preocupaciones específicas. Incluye convenciones de modelado, técnicas de análisis y normas de presentación. Un punto de vista define qué aspectos observar y cómo representarlos.

4. **Vista (View):**

   Es una representación concreta de la arquitectura basada en un conjunto específico de preocupaciones de uno o más interesados, conforme a un punto de vista. Describe cómo se aborda un conjunto determinado de preocupaciones para ciertos stakeholders. Las vistas pueden ser estructurales, de comportamiento, de despliegue, etc.

   Una vista concreta podría ser el diagrama de clases UML generado según el punto de vista lógico.

5. **Modelo (Model):**

   Es una representación de uno o más aspectos de un sistema, usada para construir una vista. En el contexto arquitectónico, un modelo describe alguna característica relevante del sistema, como su estructura, comportamiento o interacciones. Un modelo puede ser textual, gráfico, matemático o una combinación. Es la representación de uno o más aspectos del sistema. Ejemplo: un modelo de flujo de datos o un modelo de componentes y conectores. Ejemplo: un modelo UML de componentes.

6. **Relación (Relationship):**

   Es una conexión entre elementos de arquitectura, ya sea dentro de un modelo, entre modelos, o entre vistas. Ejemplo: una relación de dependencia entre módulos, o la trazabilidad entre una vista de requisitos y una vista de diseño. Puede ser estructural (como "contiene" o "depende de") o de trazabilidad (como "satisface un requisito").

7. **Lenguaje de descripción arquitectónica (Architecture Description Language – ADL):**

   Es un lenguaje formal, gráfico o textual, diseñado para describir arquitecturas. Define los elementos, relaciones, restricciones y reglas sintácticas y semánticas para describir y analizar la arquitectura. Puede ser formal o semiformal. Ejemplo: AADL (Architecture Analysis & Design Language) o SysML para sistemas, UML, ArchiMate.

8. **Framework de Descripción de Arquitectura (AD Framework):**

   Es una estructura que proporciona prácticas comunes, métodos, plantillas, puntos de vista, convenciones y metodologías organizadas que pueden usarse para desarrollar descripciones de arquitectura. Proporciona estructura y coherencia al proceso de descripción. Ejemplos incluyen TOGAF, DoDAF, Zachman, 4+1 View Model de Kruchten o el modelo C4.

9. **Descripción de Arquitectura (Architecture Description):**

   Conjunto coherente de artefactos (vistas, modelos, documentación) que describen la arquitectura de un sistema.

10. **Decisiones de arquitectura (Architecture Decisions):**

    Decisiones clave que impactan de forma significativa en la estructura, comportamiento y evolución del sistema. Documentar estas decisiones es fundamental para la trazabilidad y la gobernanza arquitectónica.

2.4 FRONT-END – BAC-KEND – API

Front end y back end son términos amplios que agrupan de forma lógica las diferentes tecnologías y capas de software de cualquier aplicación. El front end se centra en los aspectos que los usuarios pueden ver. Por el contrario, el back end es todo lo que hace que la aplicación funcione. (AWS, 2025)

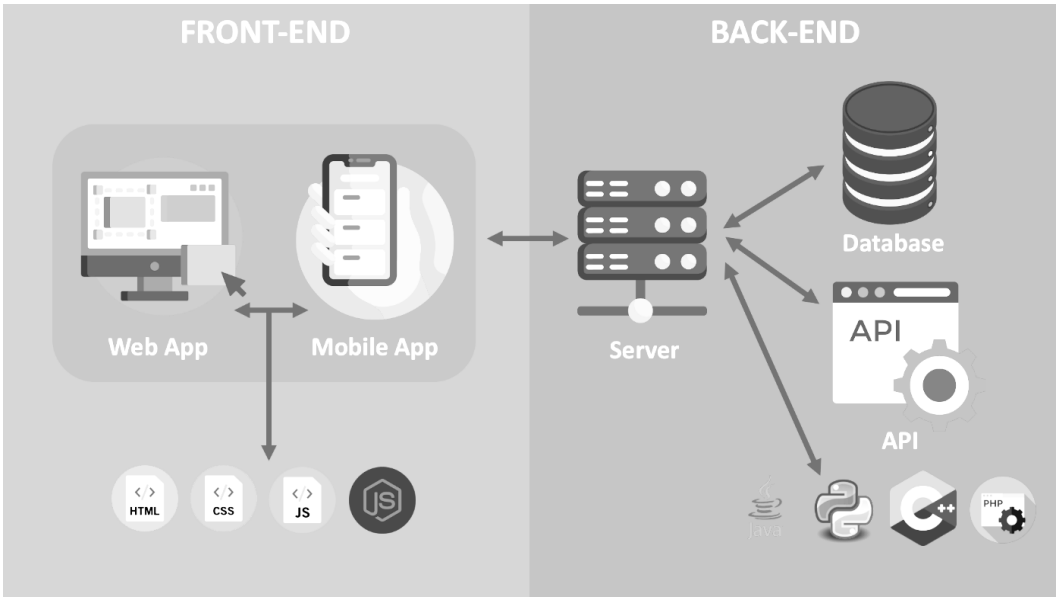

Figura 2: Front-End y Back-End

El frontend incluye todo lo que el usuario ve e interactúa directamente, como la interfaz gráfica, botones, menús, texto, imágenes, videos, etc. El backend se encarga de la lógica, la infraestructura y la gestión de datos que hacen que la aplicación funcione, incluyendo servidores, bases de datos y APIs, y es lo que el usuario no ve.

2.4.1 FRONT-END

El Frontend es la parte del sistema software que interactúa directamente con el usuario final. Incluye todo lo que el usuario puede ver, tocar e ingresar en una aplicación, y se ejecuta principalmente en el navegador web o dispositivo del usuario. Está compuesto por interfaces gráficas (UI), lógica de presentación y mecanismos para enviar o recibir información desde el Backend.

Frontend (Lado del cliente):

- Interfaz de usuario (UI): Es la parte de la aplicación que el usuario ve y con la que interactúa directamente, como botones, menús, formas, etc.

- Experiencia del usuario (UX): Se centra en cómo el usuario interactúa con la aplicación y cómo se siente al usarla.
- Desarrollo: Se utiliza HTML, CSS y JavaScript para crear y dar estilo a la interfaz de usuario. Frameworks como React, Angular o Vue.js.

Como ejemplo de este concepto, en un sistema de recetas electrónicas, el frontend incluye las pantallas que usan los médicos para cargar una receta, las vistas que usan los pacientes para consultarlas y los formularios que utilizan las farmacias para validarlas.

2.4.2 BACK-END

El Backend es la parte del sistema software que se encarga de la lógica de negocio, el procesamiento de datos y la interacción con bases de datos y otros sistemas o servicios. Funciona en servidores y no es directamente accesible ni visible para el usuario final. Su responsabilidad es procesar las solicitudes provenientes del frontend, ejecutar reglas de negocio y devolver respuestas o resultados.

Backend (Lado del servidor):

- Lógica de negocio: Implica las reglas y procesos que la aplicación sigue para funcionar, como procesar pedidos, gestionar usuarios, etc.
- Gestión de datos: Se ocupa de almacenar, recuperar y manipular los datos de la aplicación, utilizando bases de datos.
- Infraestructura: Incluye los servidores y las redes necesarias para que la aplicación funcione.
- API: Permite que el frontend y el backend se comuniquen entre sí.
- Tecnologías comunes del backend incluyen lenguajes y frameworks como Java, .NET, Python (Django/Flask), Node.js, y bases de datos como PostgreSQL, MySQL o MongoDB.

Ejemplo: En un sistema de recetas electrónicas, el backend incluye los servicios que almacenan las recetas, validan la identidad del médico, comunican la receta con las obras sociales, y actualizan el estado de entrega en las farmacias.

2.4.3 API

Una API (Application Programming Interface, o Interfaz de Programación de Aplicaciones) es un conjunto documentado de reglas, protocolos y herramientas que permiten que dos componentes de software se comuniquen entre sí. Una API define cómo se solicitan y entregan servicios o datos, sin exponer cómo se implementan internamente. Una API puede existir a distintos niveles (bibliotecas, sistemas operativos, servicios web) y facilita la interoperabilidad, la reutilización y la modularidad en el desarrollo de software.

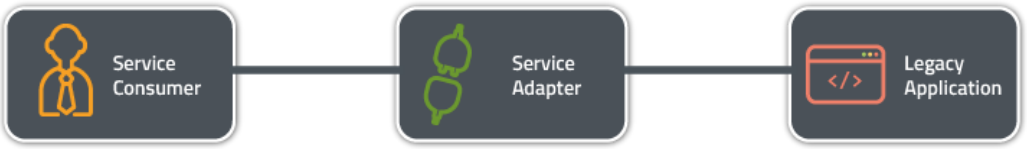

Figura 3: Itenrfaz de Programación de Aplicaciones

Una API es una interfaz para que diferentes componentes de software interactúen. Una API REST es una forma moderna, eficiente y ampliamente adoptada de ofrecer servicios a través de la web. Su correcto diseño y documentación es clave para la interoperabilidad y escalabilidad en arquitecturas modernas, incluyendo las basadas en microservicios o cliente-servidor.

API REST

Una API REST (Representational State Transfer) es un tipo particular de API que cumple con los principios del estilo arquitectónico REST, propuesto por Roy Fielding. Es ampliamente utilizada para construir servicios web ligeros y escalables, especialmente en aplicaciones web y móviles.

Características principales de una API REST:

- Usa el protocolo HTTP como canal de comunicación.
- Utiliza los métodos HTTP estándar (GET, POST, PUT, DELETE, PATCH) para representar operaciones sobre recursos.
- Los recursos (por ejemplo, pacientes, recetas, usuarios) se identifican mediante URLs (por ejemplo, /api/recetas/123).

- La comunicación es sin estado (stateless): cada solicitud contiene toda la información necesaria para procesarla.
- Las respuestas se devuelven normalmente en formato JSON o XML.

3 ESTILOS DE ARQUITECTURA

Un estilo de arquitectura de software es un patrón general y reutilizable de cómo estructurar un sistema de software, definiendo un conjunto de principios, patrones, restricciones y tipos de componentes que guían la organización e interacción de los elementos del sistema. (Sommerville, 2021)

Definición formal

Según el estándar ISO/IEC/IEEE 42010:2022 y la literatura académica, un estilo arquitectónico es un conjunto coherente de características arquitectónicas que describe una familia de sistemas que comparten una estructura y comportamiento común. Esto incluye:

- Componentes: tipos de unidades funcionales.
- Conectores: mecanismos de comunicación entre componentes.
- Configuración: reglas para organizar los componentes y conectores.
- Restricciones: principios que limitan el diseño para lograr propiedades deseadas (como escalabilidad o mantenibilidad).

Relación con arquitecturas reales

Los estilos arquitectónicos son modelos conceptuales, y en la práctica, un sistema puede combinar múltiples estilos para satisfacer distintos requisitos. Por ejemplo, una aplicación puede usar microservicios (estilo distribuido), cada uno con su propio diseño MVC (estilo estructural) y comunicación basada en eventos.

3.1 ARQUITECTURA EN CAPAS (N-TIER)

El estilo arquitectónico multicapa (también llamado n-tier) es un enfoque estructural para el diseño de sistemas de software, en el que las responsabilidades del sistema se separan en capas o niveles funcionales independientes, organizados

jerárquicamente. Cada capa tiene una responsabilidad específica y se comunica con la capa inmediata superior o inferior a través de interfaces bien definidas.

El estilo multicapa (n-tier) organiza el sistema en múltiples capas lógicas y/o físicas, donde cada capa encapsula un conjunto de responsabilidades relacionadas, como la presentación, la lógica de negocio o el acceso a datos. Este estilo busca separación de responsabilidades, modularidad y escalabilidad.

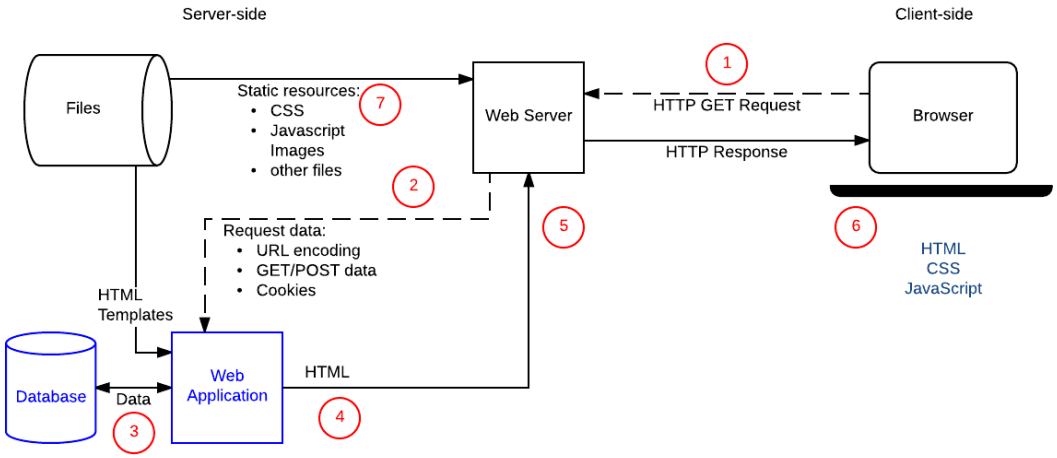

Figura 4: Implementación de una arquitectura en capas

La Figura 4 presenta un ejemplo de implementación basada en un patrón de arquitectura en capas. Este estilo de arquitectura de software, tiene como ventajas:

- Separación de preocupaciones (separation of concerns): Cada capa es responsable de una parte específica del sistema.
- Escalabilidad: Las capas pueden distribuirse en diferentes servidores (capas físicas).
- Mantenibilidad: Cambios en una capa pueden minimizar el impacto en las otras.
- Reusabilidad: Componentes de una capa pueden reutilizarse en diferentes contextos o interfaces.

A continuación, se mencionan las variantes mas importantes del estilo de arquitectura de software en capas.

3.1.1 ARQUITECTURA DE 2-CAPAS: CLIENTE-SERVIDOR

Esta variante de multicapas estructura el sistema en una arquitectura de dos capas, que se denomina también "Arquitectura Cliente-Servidor". El cliente se comunica directamente con el Servidor (y la base de datos). Es la forma más básica, con limitaciones de escalabilidad y seguridad.

3.1.2 ARQUITECTURA DE 3-CAPAS

Una arquitectura de 3 capas, se basa en la de dos capas, pero además introduce una capa intermedia denominada "lógica de negocio", entre la capa cliente y la capa Servidor de base de datos. Las capas resultantes son las siguientes:

Capa de Presentación (Cliente): UI UX. Maneja la interacción con el usuario. Muestra la información y captura acciones del usuario. Puede estar implementada en una aplicación web, móvil o de escritorio.

Capa de Lógica de Negocio (Servidor de Aplicaciones): Contiene las reglas del negocio, el procesamiento principal.

Capa de Datos (Servidor de Base de Datos): Gestiona la persistencia de los datos. Se encarga de acceder a la base de datos, ejecutar consultas y devolver los datos a la capa de lógica.

Este es el modelo predominante para la mayoría de las aplicaciones web modernas y ofrece mejor escalabilidad, seguridad y mantenibilidad que el modelo de 2 capas.

3.1.3 ARQUITECTURA DE 5-CAPAS

Una arquitectura de 5 capas en sistemas de software es una especialización del estilo arquitectónico multicapa (n-tier), donde cada capa tiene una responsabilidad claramente definida y se comunica con las capas adyacentes. Este modelo favorece la separación de preocupaciones, la escalabilidad, el mantenimiento y la reutilización del código. A continuación, se describen las cinco capas más comunes en este tipo de arquitectura:

Capa de presentación (User Interface o UI): Responsable de la interacción con el usuario final. Incluye elementos gráficos, formularios, navegación, validaciones de entrada y otras funciones centradas en la experiencia de usuario. Ejemplos: aplicaciones web en HTML/JavaScript, apps móviles, interfaces gráficas de escritorio.

Capa de servicios o interfaz de aplicación (API): Actúa como punto de entrada de las solicitudes externas (generalmente desde la capa de presentación) hacia la lógica interna del sistema. Expone funcionalidades a través de APIs, muchas veces siguiendo patrones como REST o GraphQL. Ejemplos: controladores de una API REST, servicios SOAP, endpoints HTTP.

Capa de lógica de negocio (Business Logic o Domain Layer): Contiene las reglas de negocio, políticas, cálculos y validaciones centrales del dominio del sistema. Es independiente de la presentación o del almacenamiento de datos. Ejemplos: servicios de dominio, clases de negocio, validadores, gestores de reglas.

Capa de acceso a datos (Data Access Layer o Persistence Layer): Se encarga de la interacción con los sistemas de almacenamiento persistente. Contiene los mecanismos para leer y escribir datos, usualmente desde una base de datos o repositorio externo. Ejemplos: repositorios, mapeadores ORM (como Hibernate o Entity Framework), controladores SQL.

Capa de infraestructura o alojamiento (Hosting / Infrastructure Layer): Proporciona el entorno donde se despliegan y ejecutan las demás capas. Incluye servidores, redes, servicios en la nube, bases de datos, sistemas de mensajería, contenedores, etc. Ejemplos: servidor web (Nginx, Apache), motor de bases de datos (PostgreSQL, MongoDB), servicios en la nube (AWS, Azure), Docker, Kubernetes.

Opcionalmente, pueden existir más capas (por ejemplo, una capa de servicios o de integración).

3.2 Arquitectura Orientada a Servicios SOA

SOA (Service-Oriented Architecture), o Arquitectura Orientada a Servicios es un estilo de arquitectura de software en el que las funcionalidades del sistema se estructuran como un conjunto de servicios interoperables, reutilizables, autónomos y loosely coupled (débilmente acoplados), que pueden ser orquestados y reutilizados en diferentes aplicaciones o procesos de negocio. Los servicios se comunican entre sí a través de interfaces bien definidas, generalmente mediante protocolos estándares como HTTP y formatos como XML o JSON.

Cada servicio ofrece una unidad funcional coherente, como por ejemplo verificar el stock, procesar pagos o registrar usuarios. Ejemplo: Un sistema bancario podría tener servicios independientes para:

- Verificar saldo de cuenta
- Transferir fondos
- Emitir comprobantes

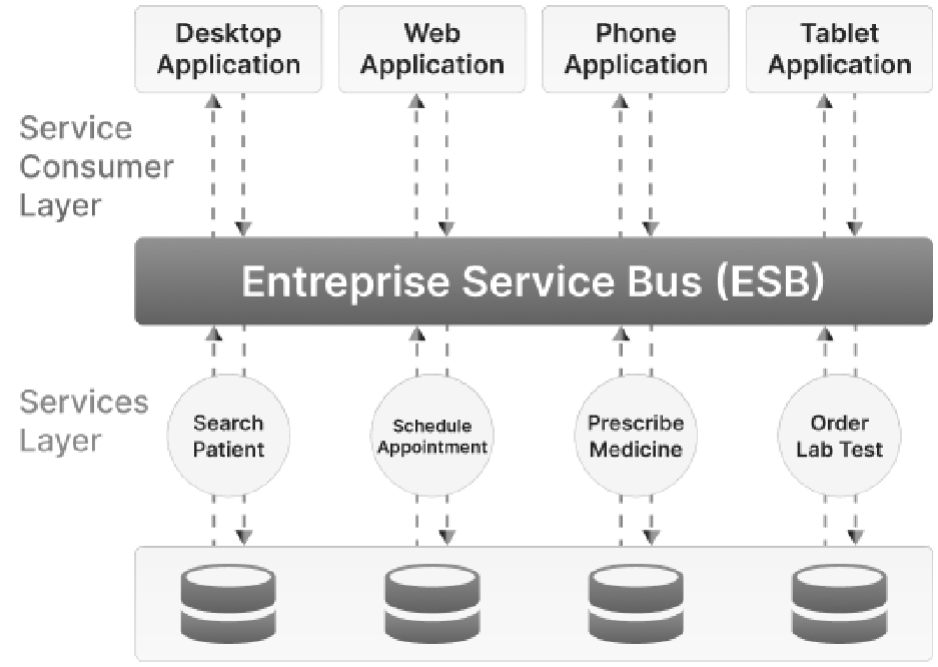

Figura 5: Arquitectura orientada a servicios

Entre las ventajas de la arquitectura orientada a Servicios, se pueden citar:

- Interoperabilidad: los servicios pueden comunicarse entre sí independientemente del lenguaje o plataforma en que estén implementados.
- Reutilización: un mismo servicio puede ser usado por múltiples aplicaciones o procesos.
- Acoplamiento débil: los servicios están diseñados para depender lo menos posible entre sí.
- Descubrimiento y composición: los servicios pueden descubrirse dinámicamente y combinarse para formar flujos de negocio complejos.
- Interfaces estándar: comúnmente implementados sobre protocolos como SOAP, WSDL, XML, aunque también pueden usar REST.

Estos servicios podrían ser usados por la aplicación móvil, el sitio web o incluso por terceros (por ejemplo, billeteras digitales) a través de una API de servicios.

3.3 ARQUITECTURA MICROSERVICIOS

Una arquitectura de microservicios consta de una colección de servicios autónomos y pequeños. Cada uno es un servicio es independiente y debe implementar una funcionalidad de negocio individual dentro de un contexto delimitado. Un contexto delimitado es una división natural en cuanto a procesos de negocio de una empresa y proporciona un límite explícito dentro del cual existe un modelo de dominio.

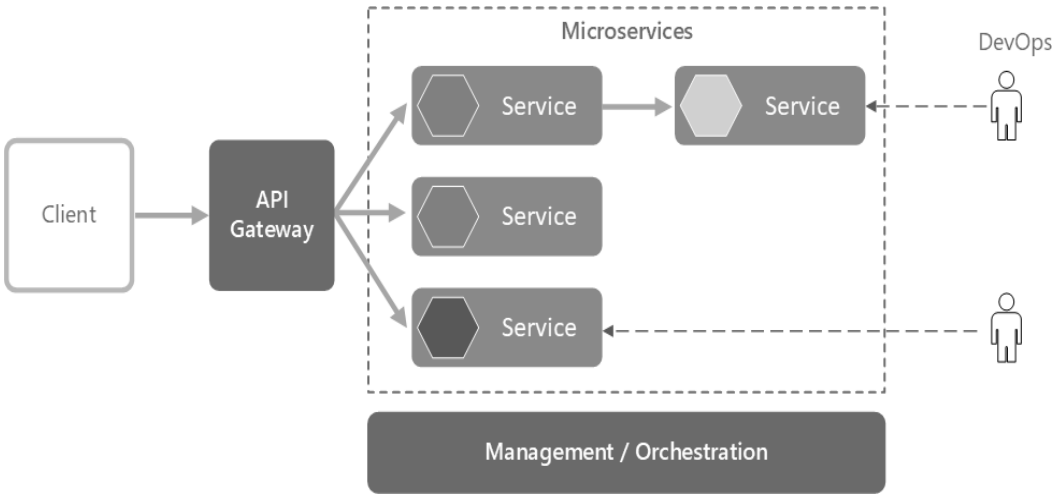

Figura 6: Arquitectura basada en Microservicios

Microservicios

Los microservicios son componentes de código que implementan funcionalidad clara precisa y reducida. Son pequeños en e independientes y presentan interfaces claras y bien definidas.

Cada servicio es un código base independiente, que puede administrarse por un equipo de desarrollo pequeño. Los servicios pueden implementarse de manera independiente lo que permite que un equipo de desarrollo pueda actualizar un servicio existente sin tener que volver a generar e implementar toda la aplicación. Los servicios son los responsables de conservar sus propios datos o estado externo. Esto difiere del modelo tradicional, donde una capa de datos independiente controla la persistencia de los datos.

Los servicios se comunican entre sí mediante API bien definidas. Los detalles de la implementación interna de cada servicio se ocultan frente a otros servicios. La arquitectura de microservicios, admite la programación políglota. Por ejemplo, no es necesario que los servicios compartan la misma pila de tecnología, las bibliotecas o los marcos.

Además de los propios servicios, hay otros componentes que aparecen en una arquitectura típica de microservicios:

Administración e implementación. Este componente es el responsable de la colocación de servicios en los nodos, la identificación de errores, el reequilibrio de servicios entre nodos, etc. Normalmente, para este componente se utiliza una herramienta estándar, como Kubernetes, en lugar de algo creado de forma personalizada.

Puerta de enlace de API La puerta de enlace de API es el punto de entrada para los clientes. En lugar de llamar a los servicios directamente, los clientes llaman a la puerta de enlace de API, que reenvía la llamada a los servicios apropiados en el back-end.

Entre las ventajas de usar una puerta de enlace de API se encuentran las siguientes:

- Desacoplan los clientes de los servicios. Los servicios pueden cambiar de versión o refactorizarse sin necesidad de actualizar todos los clientes.
- Los servicios pueden utilizar los protocolos de mensajería que no son fáciles de usar para un servicio web, como AMQP.
- La puerta de enlace de API puede realizar otras funciones transversales como la autenticación, el registro, la terminación SSL y el equilibrio de carga.
- Directivas estándar, como por ejemplo, para la limitación, el almacenamiento en caché, la transformación o la validación.

3.3.1 COMPARACIÓN ENTRE MICROSERVICIOS Y SOA

La arquitectura orientada a servicios (SOA) es un método de desarrollo de software que utiliza componentes de software llamados servicios para crear aplicaciones empresariales. Cada servicio proporciona una capacidad empresarial. También pueden comunicarse entre sí a través de diferentes plataformas y lenguajes. Los desarrolladores usan SOA para reutilizar servicios en diferentes sistemas o combinar varios servicios independientes para realizar tareas complejas. La arquitectura de microservicios es una evolución del estilo arquitectónico de SOA. Si bien cada servicio de SOA es una capacidad empresarial completa, cada microservicio es un componente de software mucho más pequeño que se especializa en una sola tarea. Los microservicios tratan los defectos

de SOA para hacer que el software sea más compatible con entornos empresariales modernos basados en la nube. (AWS, 2025)

Si bien la arquitectura orientada a servicios (SOA) puede funcionar bien para crear aplicaciones empresariales de gran tamaño, necesita más flexibilidad para escalar aplicaciones más pequeñas y específicas de la empresa. Estas son algunas de las limitaciones de la SOA:

- El bus de servicios empresariales (ESB) conecta varios servicios entre sí, lo que lo convierte en un único punto de error.
- Todos los servicios comparten un repositorio de datos común. Esto dificulta la gestión individual de los servicios.
- Cada servicio tiene un alcance amplio. Por lo tanto, si uno de los servicios falla, todo el flujo de trabajo empresarial se verá afectado.

El modelo de microservicios divide un servicio de SOA en servicios más pequeños. Cada microservicio funciona dentro de su contexto limitado y se ejecuta independientemente de otros servicios. En resumen, la arquitectura de microservicios tiene interdependencias limitadas o nulas entre los servicios individuales y reduce el riesgo de errores en todo el sistema. Por ejemplo, la gestión de inventario sería un servicio de SOA de un sistema de comercio electrónico. Sin embargo, el enfoque de microservicios dividiría la gestión del inventario en servicios más pequeños, como el comprobador de disponibilidad, el cumplimiento y la contabilidad. (AWS, 2025)

3.4 ARQUITECTURA SERVERLESS

El término "serverless" es un poco engañoso, ya que no significa que no haya servidores. Lo que realmente significa es que el desarrollador ya no tiene que preocuparse por aprovisionar, escalar o gestionar esos servidores. El proveedor de la nube (AWS Lambda, Azure Functions, Google Cloud Functions, etc.) es el que se encarga de toda la gestión de la infraestructura subyacente. (Amazon Web Services, 2025)

En este modelo, el código de la aplicación se ejecuta en funciones efímeras y elásticas que se activan en respuesta a eventos.

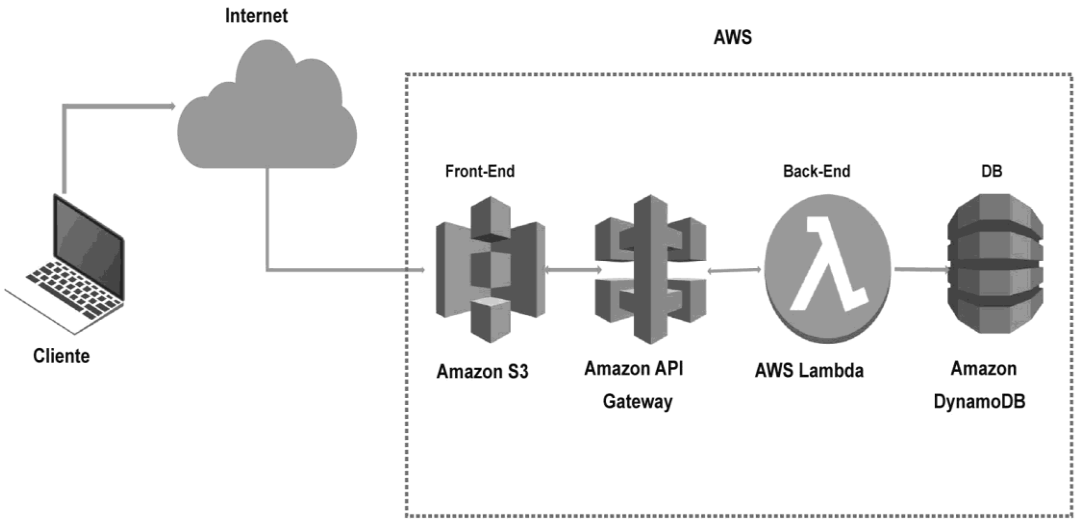

Figura 7: Arquitectura Serverless Amazon

Componentes

La arquitectura Serverless se basa en varios componentes interconectados, principalmente servicios gestionados por proveedores de nube:

1. **Funciones como Servicio (Function as a Service - FaaS):**

Es el corazón de Serverless. Los desarrolladores escriben funciones de código pequeñas y autocontenidas que realizan una única tarea (ej., procesar una imagen, manejar una solicitud HTTP, guardar un elemento en una base de datos). Estas funciones se ejecutan en contenedores efímeros que son aprovisionados y escalados automáticamente por el proveedor de la nube en respuesta a un evento. El modelo de pago es por uso real (tiempo de ejecución y memoria consumida), no por tiempo de servidor provisionado.

2. **Servicios Backend como Servicio (Backend as a Service - BaaS):**

Complementa a FaaS. Son servicios de terceros gestionados que la aplicación puede consumir directamente desde el cliente (frontend o FaaS),

eliminando la necesidad de desarrollar y mantener un backend propio para funcionalidades comunes.

Ejemplos: Bases de datos (DynamoDB, Firestore), autenticación (Auth0, AWS Cognito), almacenamiento de archivos (S3, Google Cloud Storage), pasarelas de pago (Stripe).

3. **Puertas de Enlace de API (API Gateways):**

Se utilizan para exponer las funciones FaaS a clientes externos (aplicaciones web, móviles). Gestionan el enrutamiento de solicitudes HTTP a las funciones apropiadas, la autenticación, la autorización y la limitación de tasas.

4. **Fuentes de Eventos (Event Sources):**

Son los disparadores que invocan las funciones FaaS. La arquitectura Serverless es inherentemente impulsada por eventos. Ejemplos: Solicitudes HTTP (API Gateway), cargas de archivos en almacenamiento en la nube, mensajes en una cola de mensajes (SQS, Pub/Sub), eventos de base de datos (DynamoDB Streams), eventos de servicios externos, cron jobs.

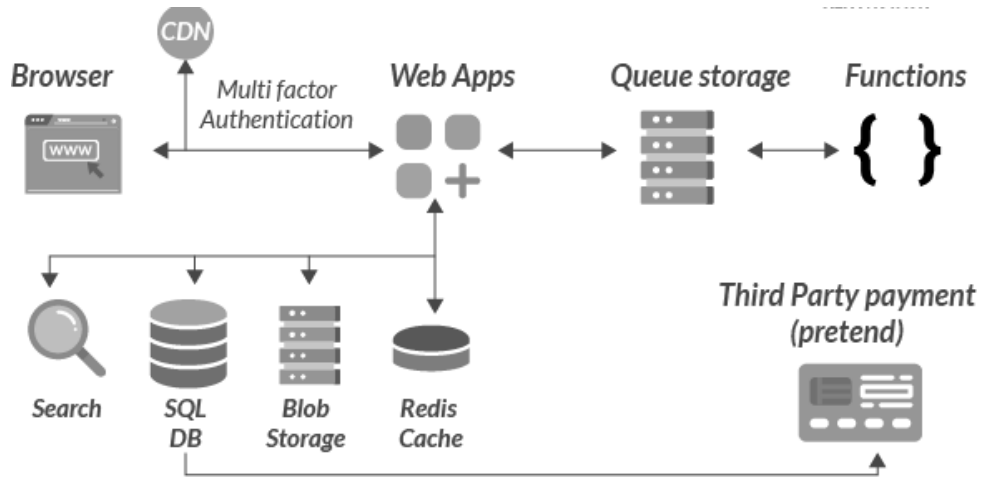

Figura 8: Arquitectura Serverless de Microsoft Azure

Ventajas de la Arquitectura Serverless

- Pago por uso (costo): Se paga solo cuando el código se está ejecutando, eliminando la necesidad de aprovisionar y pagar por servidores inactivos.

Esto puede llevar a ahorros significativos, especialmente para cargas de trabajo esporádicas o variables.

- Escalabilidad Automática y Elástica: Las funciones escalan automáticamente y casi ilimitadamente con la demanda, sin necesidad de configuración manual o gestión de clusters.
- Reducción de la Carga Operativa (Zero Ops): El proveedor de la nube gestiona toda la infraestructura (servidores, redes, SO, contenedores, actualizaciones de seguridad, parches, etc.). Los desarrolladores se centran únicamente en el código.
- Mayor Agilidad del Desarrollador: Los desarrolladores pueden implementar y desplegar funciones rápidamente, sin preocuparse por la infraestructura subyacente. Esto acelera el ciclo de desarrollo y la entrega de nuevas funcionalidades.
- Tolerancia a Fallos: La naturaleza distribuida y tolerante a fallos de las plataformas FaaS inherente de la nube, combinada con la ejecución efímera de funciones, puede mejorar la resiliencia general del sistema.

Desventajas

- "Cold Starts" (Arranques en Frío): La primera vez que una función inactiva es invocada, el proveedor de la nube necesita aprovisionar el entorno de ejecución, lo que puede introducir una latencia adicional ("cold start"). Esto es más crítico en aplicaciones con requisitos de latencia muy bajos.
- Bloqueo de Proveedor (Vendor Lock-in): Una aplicación serverless se integra profundamente con los servicios y APIs específicas de un proveedor de nube (AWS Lambda, Azure Functions, Google Cloud Functions). Migrar a otro proveedor puede ser costoso y complejo.
- Depuración y Monitorización Distribuidas: Depurar y monitorizar flujos de trabajo que abarcan múltiples funciones y servicios distribuidos puede ser más complejo que en un monolito o una aplicación con pocos servidores. Requiere herramientas de observabilidad específicas (tracing distribuido, logs agregados).
- Límites de Recursos: Las funciones FaaS suelen tener límites en el tiempo de ejecución, la memoria, el tamaño del paquete de código y la concurrencia.

Esto las hace inadecuadas para tareas de larga duración o intensivas en computación.

- Manejo del Estado: Las funciones son por naturaleza sin estado (stateless) entre invocaciones. Mantener el estado requiere el uso de servicios externos (bases de datos, caché distribuida, almacenamiento de objetos). Esto puede añadir complejidad al diseño.
- Pruebas Locales Complejas: Replicar el entorno serverless completo en un entorno de desarrollo local puede ser un desafío.
- Seguridad: Aunque el proveedor gestiona la seguridad de la infraestructura, los desarrolladores siguen siendo responsables de la seguridad del código de la función, la configuración de permisos (IAM roles) y la gestión de secretos.

A continuación se presentan algunos escenarios de aplicación idelaes para la arquitectura Serverless:

- APIs web y microservicios sin estado: Backends para aplicaciones móviles y web con picos de tráfico impredecibles.
- Procesamiento de datos en tiempo real: Funciones que se activan por eventos de streaming de datos, carga de archivos (ej., redimensionamiento de imágenes, procesamiento de videos).
- Procesamiento de formularios: Lógica de backend para formularios web.
- Chatbots y asistentes virtuales: Manejo de la lógica detrás de las interacciones.
- Automatización de tareas (Cron Jobs): Ejecución de tareas programadas (ej., generación de informes diarios).
- Validación y transformación de datos: En pipelines de datos.
- IoT (Internet of Things): Procesamiento de datos de dispositivos conectados.
- Backends para eventos: Servicios que reaccionan a eventos en la nube (ej.,

4 ARQUITECTURA FÍSICA

En cuanto a la elección de la infraestructura de hardware, redes y comunicaciones, existe una decisión básica entre dos tipos principales: On-Premise y Cloud. Los mismos representan dos modelos fundamentales para la implementación, gestión y entrega de software y recursos de TI. La elección entre uno y otro tiene implicaciones profundas en la inversión, la operación, la escalabilidad y la seguridad de una organización.

4.1.1 CLOUD COMPUTING

El modelo Cloud Computing (computación en la Nube) implica el uso de servicios informáticos (servidores, almacenamiento, bases de datos, redes, software, análisis, inteligencia artificial) a través de Internet ("la nube") desde un proveedor externo. En lugar de poseer y mantener la infraestructura, la organización alquila los recursos según sea necesario.

Modelos de Servicio Cloud ("XaaS")

La nube se ofrece en diferentes niveles de abstracción y gestión:

IaaS (Infrastructure as a Service - Infraestructura como Servicio): En este esquema, el proveedor es el responsable de gestionar la infraestructura física (servidores, redes, virtualización), y el usuario gestiona los sistemas operativos, middleware, aplicaciones y datos. Algunas tecnologías que implementan este modelo son: Máquinas virtuales (EC2 en Amazon AWS, VMs en Microsoft Azure), almacenamiento de bloques (EBS en AWS), redes virtuales. En cuanto al nivel de control, este esquema ofrece alto control, pero aún con responsabilidades de gestión del SO hacia las capas superiores.

PaaS (Platform as a Service - Plataforma como Servicio): Bajo este modelo,el proveedor gestiona la infraestructura y la plataforma (SO, middleware, bases de datos, tiempos de ejecución, herramientas de desarrollo). El usuario se enfoca en el desarrollo y despliegue de su aplicación. Ejemplos: Google App Engine, AWS Elastic

Beanstalk, Azure App Service, Heroku. En este modelo, hay menor control sobre la infraestructura subyacente, pero mayor agilidad en el desarrollo.

SaaS (Software as a Service - Software como Servicio): Aquí, el proveedor gestiona toda la pila (aplicación, plataforma, infraestructura). El usuario simplemente consume la aplicación a través de un navegador o una interfaz. Ejemplos: Gmail, Salesforce, Microsoft 365, Dropbox. El nivel de control es mínimo control, pero maximiza la facilidad de uso.

Además, existe **FaaS (Function as a Service)** como un subconjunto de PaaS/Serverless, donde el proveedor gestiona el tiempo de ejecución y el escalado de funciones individuales.

Según la responsabilidad en la forma de administración de los recursos, surgen diferentes tipos de implementación en la Nube:

- Nube Pública: Los recursos son propiedad y están operados por un proveedor de terceros (AWS, Azure, Google Cloud) y se comparten con múltiples organizaciones.
- Nube Privada: Los recursos son utilizados exclusivamente por una única organización. Puede estar ubicada On-Premise o ser gestionada por un tercero.
- Nube Híbrida: Combina una infraestructura On-Premise con recursos de nube pública, permitiendo que las cargas de trabajo se muevan entre ellas.
- Multicloud: Uso de múltiples proveedores de nube pública para diferentes servicios o redundancia.

Entre las ventajas asociadas a la selección de una arquitectura basada en cloud computing, se pueden mencionar:

- Costo-Efectividad (OPEX): Modelo de pago por uso (pay-as-you-go), convirtiendo grandes inversiones de capital en gastos operativos. Elimina la necesidad de comprar y mantener hardware.

- Escalabilidad y Elasticidad: Los recursos se pueden escalar hacia arriba o hacia abajo casi instantáneamente y de forma automática en respuesta a la demanda.
- Agilidad y Velocidad de Comercialización: Los desarrolladores pueden aprovisionar recursos en minutos, acelerando el desarrollo, las pruebas y el despliegue de aplicaciones.
- Menor Carga Operativa: El proveedor de la nube se encarga de la gestión de la infraestructura, mantenimiento, parches y actualizaciones.
- Alta Disponibilidad y Recuperación ante Desastres: Los proveedores de nube ofrecen servicios y arquitecturas inherentes de alta disponibilidad y recuperación ante desastres a escala global.
- Innovación: Acceso instantáneo a una amplia gama de servicios gestionados de vanguardia (IA/ML, IoT, Big Data, etc.) que serían costosos de implementar On-Premise.
- Accesibilidad: Acceso a los recursos desde cualquier lugar con conexión a Internet.

Como contraparte, algunas de las desventajas comunes asociadas a este modelo son:

- Dependencia del Proveedor (Vendor Lock-in): La migración de una nube a otra puede ser compleja debido a las APIs y servicios específicos del proveedor.
- Seguridad y Privacidad de Datos: Aunque los proveedores invierten masivamente en seguridad, la responsabilidad de la seguridad es compartida ("shared responsibility model"). La organización es responsable de la seguridad de sus datos y aplicaciones en la nube, mientras que el proveedor se encarga de la seguridad de la nube.
- Costos Impredecibles (a veces): Si no se gestiona bien el uso de recursos, los costos pueden salirse de control, especialmente con modelos de pago por uso muy granulares.

- Latencia de Red: La latencia puede ser un problema para aplicaciones que requieren respuestas en tiempo real o que están muy ligadas a datos locales.
- Menor Control: Menos control sobre la infraestructura subyacente y el software en comparación con el modelo On-Premise.
- Requisitos de Conectividad: La disponibilidad y el rendimiento dependen de la conexión a Internet.

En Ingeniería de Sistemas, la comprensión de estos modelos es fundamental para diseñar arquitecturas de sistemas robustas, escalables y eficientes, así como para asesorar en la estrategia de TI de una organización. La tendencia general es clara: la nube sigue ganando terreno por su agilidad, escalabilidad y reducción de la carga operativa, pero On-Premise sigue siendo relevante para casos de uso específicos que requieren control extremo, baja latencia o cumplimiento regulatorio muy estricto. Es común utilizar modelos híbridos que combinan la arquitectura On-Premise con la arquitectura Cloud.

5 MODELO C4

5.1 C4-MODEL

El Modelo C4 es un modelo jerárquico de vistas arquitectónicas que permite representar un sistema de software desde distintos niveles de abstracción, desde una visión general hasta los detalles técnicos de implementación. Es una alternativa moderna y pragmática a los enfoques tradicionales como los 4+1 de Kruchten o los modelos UML clásicos. La figura siguiente, presenta una visión global del modelado de la arquitectura de sistemas en Model c4:

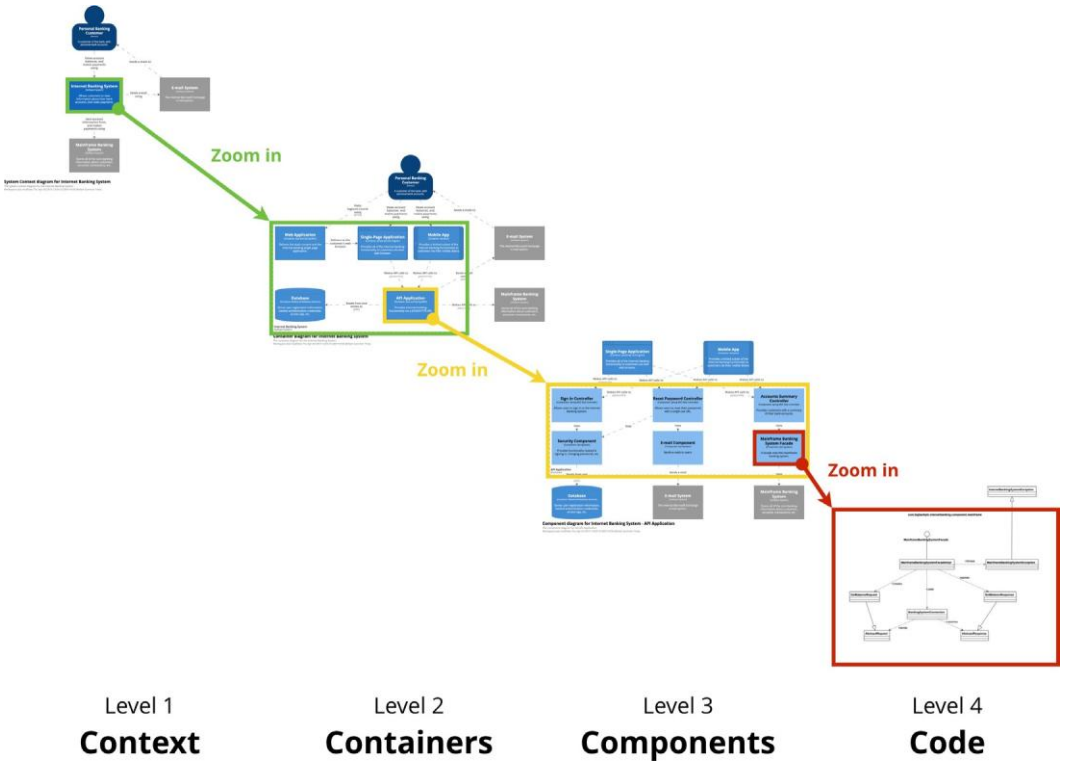

Figura 9: Modelado de Arquitectura de Software en Model C4

Su objetivo principal es ayudar a los equipos de desarrollo de software a describir y comunicar la arquitectura de software de manera efectiva y progresivamente detallada, centrándose en el significado semántico de los elementos.

El nombre "C4" proviene de los cuatro niveles jerárquicos de diagramas que se utilizan para describir la arquitectura, cada uno con un nivel de abstracción decreciente y de detalle creciente: Diagrama de Contexto (Context Diagram), Diagrama de Contenedores (Container Diagram), Diagrama de Componentes (Component Diagram), Diagrama de Clases (Class Diagram) o Código. En la figura siguiente, se presentan estos niveles:

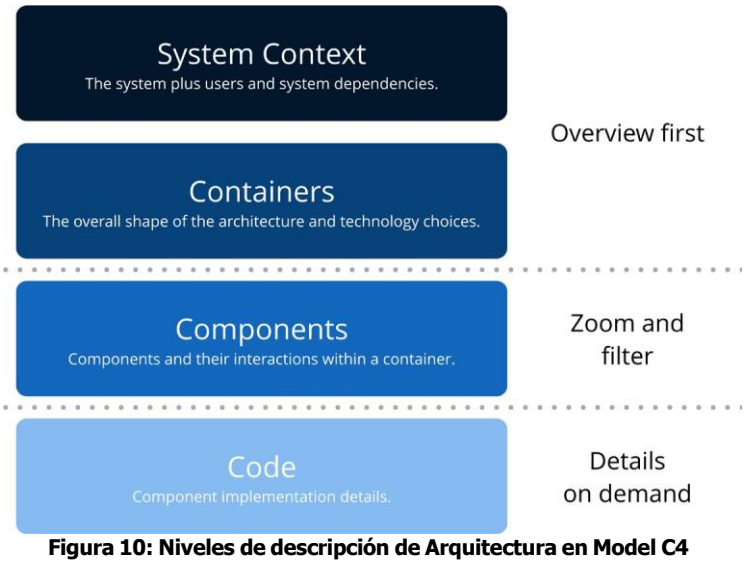

Figura 10: Niveles de descripción de Arquitectura en Model C4

5.2 Los cuatro niveles del modelo

5.2.1 Diagrama de Contexto (Context Diagram)

El Diagrama de Contexto del Sistema, representa el sistema de software como una caja negra dentro de su entorno operativo. Muestra usuarios, actores externos y otros sistemas con los que interactúa. Ejemplo: El sistema de recetas electrónicas interactúa con médicos, pacientes, farmacias, obras sociales y bases de datos gubernamentales.

La Audiencia Principal de este diagrama, es cualquier persona interesada en el sistema (stakeholder). Su propósito, es el de mostrar una vista global del sistema de software que se está construyendo y cómo se relaciona con sus usuarios y otros

sistemas de software. Es el "mapa del mundo" en el que vive el sistema. A continuación, se presenta un ejemplo de un diagrama de contexto de un sistema bancario:

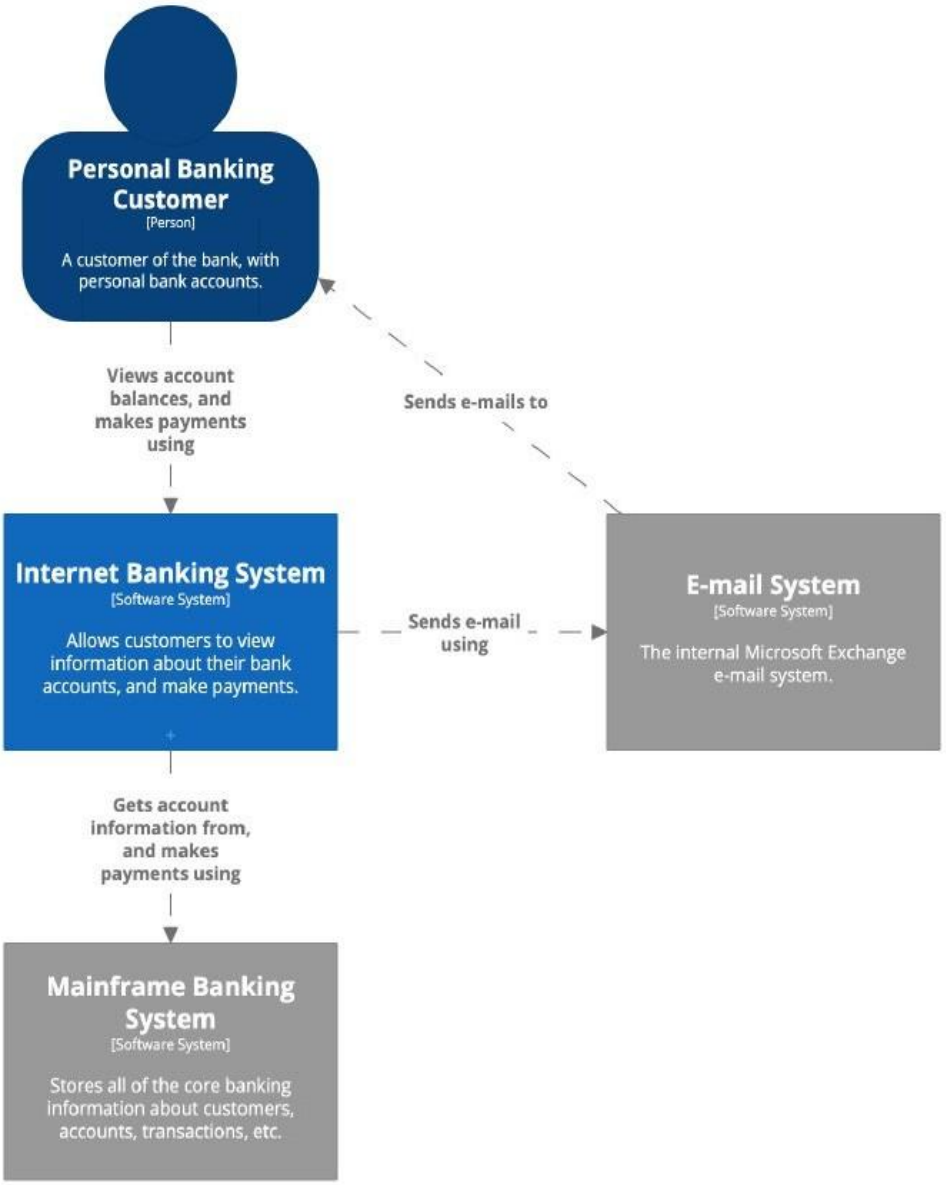

Figura 11: Ejemplo de diagrama de contexto

Este diagrama, incluye ciertos elementos típicos que facilitan la compresión del mismo y sus interacciones, como son:

- Sistema de Software (Software System): El sistema principal que se está describiendo.
- Usuarios (People): Las personas que interactúan con el sistema.
- Sistemas Externos (External Software Systems): Otros sistemas de software (propios o de terceros) con los que interactúa el sistema principal.
- Relaciones: Muestra las interacciones de alto nivel entre el sistema y su entorno.

El nivel de detalle del Diagrama de contexto, representa un alto nivel de abstracción. Ignora los detalles tecnológicos y de implementación. Responde a "¿Quién usa esto? ¿Con qué otros sistemas interactúa?" Por ejemplo, un sistema de banca online interactúa con los clientes, un sistema de gestión de cuentas bancarias y un sistema de procesamiento de pagos externo.

5.2.2 Diagrama de Contenedores (Container Diagram)

El Diagrama de Contenedores, permite descompone el sistema bajo análisis, en diferentes partes denominadas contenedores. Es decir, en aplicaciones o servicios ejecutables (como una API, una app móvil, una base de datos, etc.). Muestra cómo los contenedores se comunican y qué responsabilidades tienen. Son ejemplos de contenedores; un frontend web, una API REST, un microservicio de validación de recetas, una base de datos, un sistema de notificaciones, entre otros.

Este diagrama, esta destinado principalmente a los Arquitectos de Software e integrantes del equipo de desarrollo del sistema. El propósito del mismo, es el de descomponer el sistema de software principal en "contenedores". Un contenedor es una aplicación o un almacén de datos (servidor de aplicaciones web, base de datos, aplicación de escritorio, aplicación móvil, función serverless, microservicio).

Lo elementos típicos que integran un diagrama de contenedores, son:

- Sistema de Software (Software System): El sistema principal (del diagrama de contexto).

- Contenedores (Containers): Las aplicaciones y almacenes de datos que conforman el sistema.
- Usuarios (People): Los usuarios que interactúan con los contenedores.
- Sistemas Externos (External Software Systems): Otros sistemas externos con los que interactúan los contenedores.
- Relaciones: Muestra las comunicaciones entre los contenedores y los usuarios/sistemas externos, incluyendo las tecnologías de comunicación (ej., HTTPS, JMS).

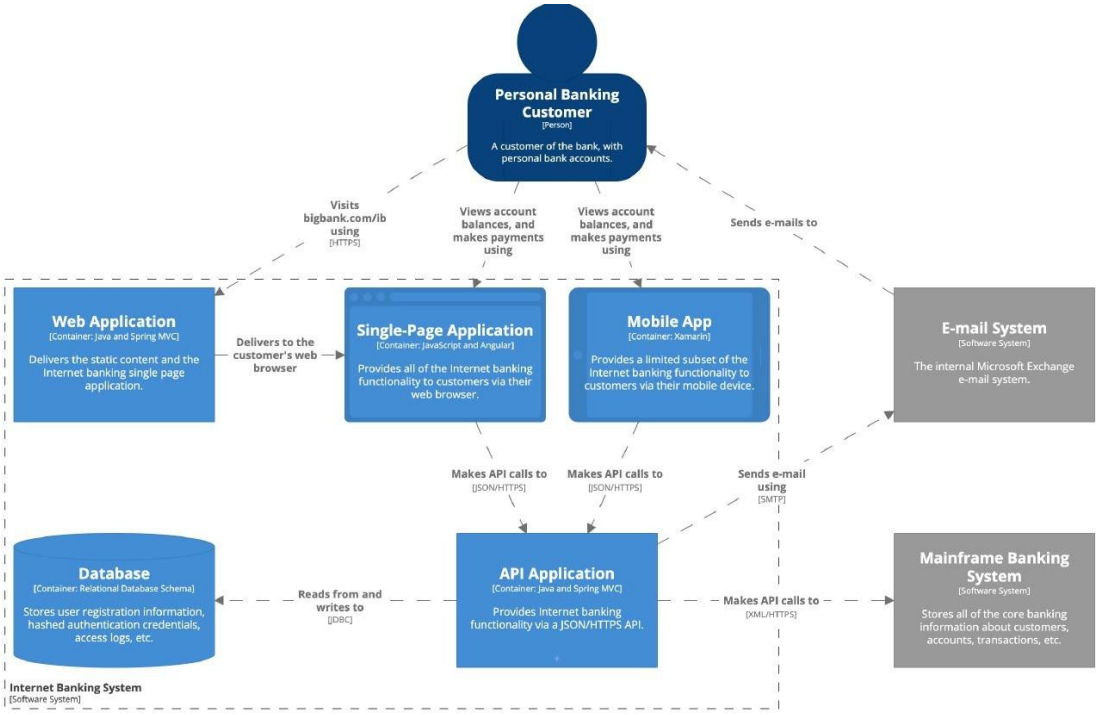

Figura 12: Ejemplo de diagrama de contenedores

La figura anterior, muestra un ejemplo de un diagrama de contenedores. El nivel de detalle en este tipo de diagramas es un mas bajo que el de los diagramas de contexto, pero aún de alto nivel. Muestra cómo se despliega el sistema. Responde a "¿Cómo se divide la responsabilidad en aplicaciones/almacenes de datos? ¿Qué tecnología usa cada uno?"

5.2.3 Diagrama de Componentes (Component Diagram)

Diagrama de Componentes: Descompone un contenedor en componentes internos que ejecutan roles específicos, como controladores, servicios, repositorios, etc. Muestra dependencias internas dentro del contenedor. Por ejemplo, dentro de la API, se puede identificar un componente de autenticación, otro de emisión de recetas, y otro para consultas de historial. En la figura siguiente se observa un ejemplo de este tipo de diagrama:

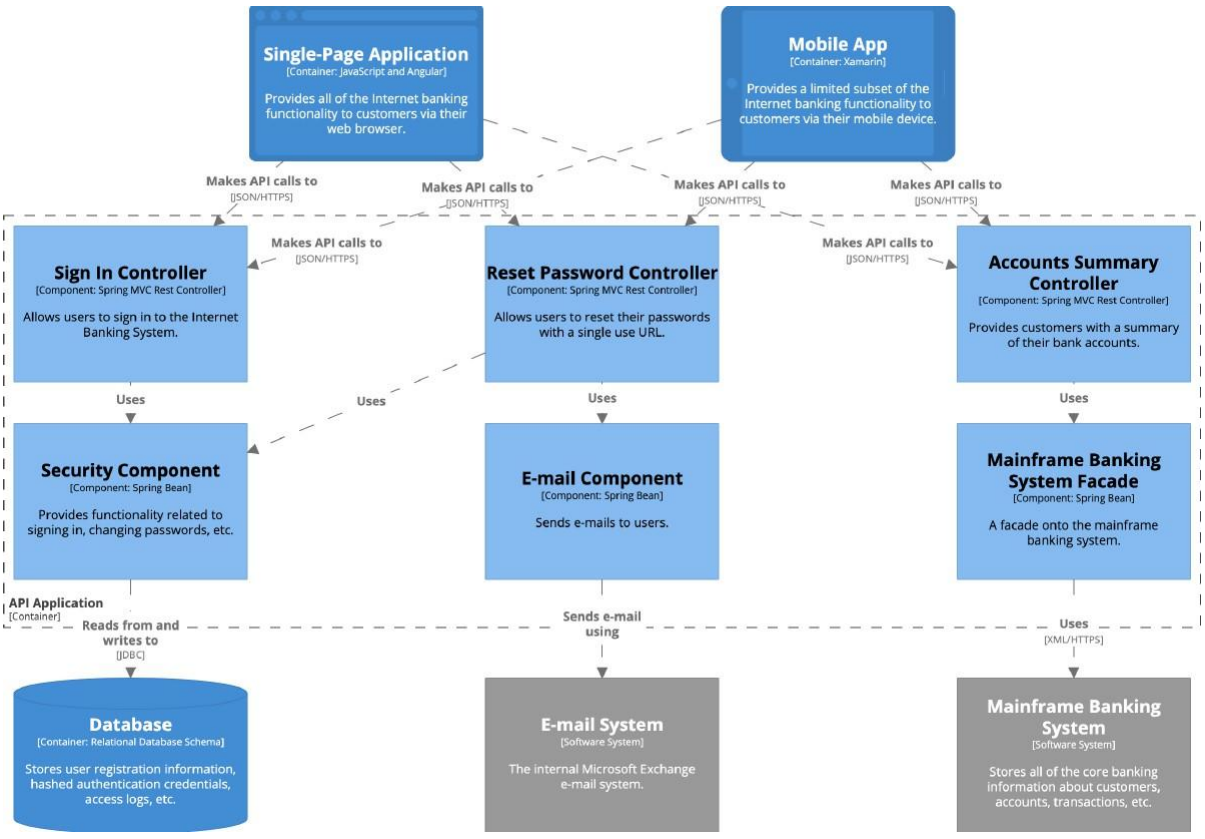

Figura 13: Diagrama de Componentes en C4

La principal audiencia de este tipo de diagramas son los desarrolladores y el propósito principal que tiene es el de descomponer un solo "contenedor" (del diagrama de contenedores) en sus "componentes" principales. Un componente es un grupo de clases cohesivo que se agrupa para ofrecer un conjunto bien definido

de funcionalidades a través de una interfaz bien definida. No es una clase, es una "unidad de implementación".

El nivel de detalle de estos diagramas es considerablemente mayor, ya que responde a los interrogantes de "¿Cómo se organiza la lógica dentro de una aplicación/contenedor? ¿Qué módulos existen?"

Los elementos que componen un diagrama de componentes son:

Contenedor (Container): El contenedor que se está descomponiendo.

Componentes (Components): Las unidades lógicas dentro del contenedor (ej., controladores, servicios, repositorios, fachadas, módulos).

Otros Contenedores/Sistemas: Contenedores o sistemas externos con los que interactúan los componentes.

Relaciones: Muestra las interacciones entre los componentes dentro del contenedor, y con los contenedores/sistemas externos. Se especifica cómo se comunican (ej., llamadas a métodos, inyección de dependencias, colas de mensajes internas).

5.2.4 Diagrama de Código (Code Diagram) / Clases

El diagrama de Código representa el detalle del código fuente, generalmente a nivel de clases, interfaces o módulos. Es útil cuando el nivel de granularidad lo requiere, por ejemplo para representar clases, interfaces y otros elementos. Esta destinado principalmente a los integrantes del equipo de desarrollo.

El propósito principal de estos diagramas es proporcionar una vista global agregada en un solo componente (del diagrama de componentes) para mostrar sus clases principales o las estructuras de código relevantes. Es el nivel de detalle más bajo y se acerca al código fuente real. Su nivel de detalle es más fino, a menudo es un diagrama de clases UML tradicional o se puede derivar

directamente del código. Responde a "¿Cómo se implementa un componente? ¿Qué clases lo forman?"

Son elementos típicos en este tipo de diagramas:

- Componente (Component): El componente que se está detallando.
- Clases/Interfaces/Funciones: Las clases y sus relaciones (asociación, herencia, composición) que componen el componente.
- Relaciones: Muestra las relaciones detalladas entre las clases, interfaces, etc.

En la figura siguiente se visualiza un ejemplo de diagrama de componentes:

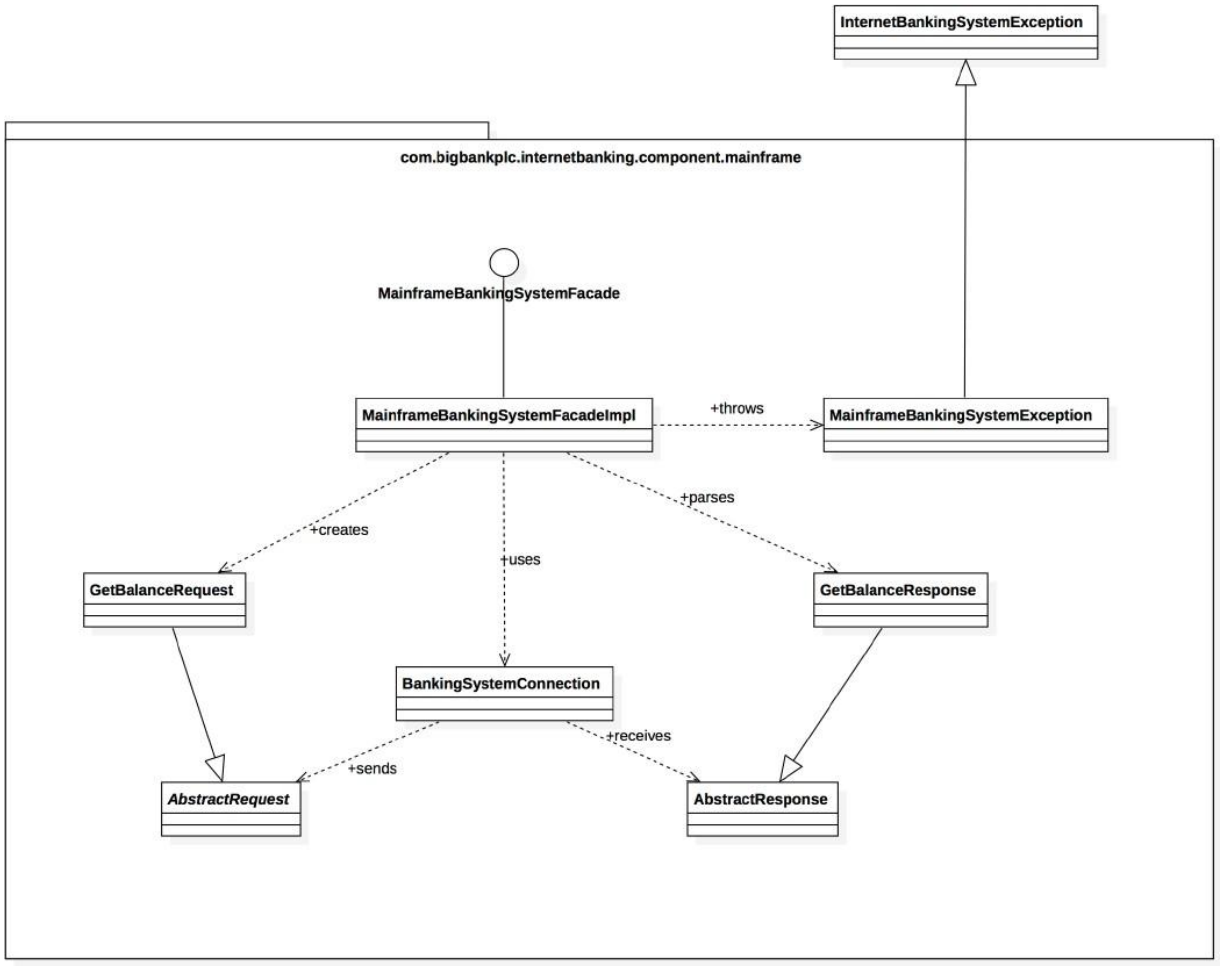

Figura 14: Diagrama de Código en C4

Este nivel a menudo se maneja mejor con herramientas de modelado de código o simplemente inspeccionando el código fuente.

Reglas Clave y Mejores Prácticas del Modelo C4

Abstracción Consistente: Cada nivel de diagrama debe ser consistente con los niveles superiores. Lo que se representa en un nivel superior debe ser desglosable en el siguiente nivel.

Leyenda y Explicación: Cada diagrama debe tener una leyenda clara que explique los símbolos y colores utilizados.

Títulos Significativos: Los títulos deben indicar claramente qué sistema/contenedor/componente se está describiendo.

Énfasis en el Significado Semántico: Utilizar nombres claros para los elementos (ej., "Servidor API" en lugar de "Capa de Lógica").

Herramientas Flexibles: No hay una herramienta específica. Se puede usar desde lápiz y papel, hasta herramientas de diagramación (Draw.io, Lucidchart, Miro), o herramientas de "diagramas como código" (Structurizr, PlantUML, Mermaid).

Enfoque en el Problema: Los diagramas deben crearse para responder a preguntas específicas de la audiencia. No se trata de documentar por documentar.

Evolución: Los diagramas deben evolucionar con la arquitectura.

Ventajas de utilizar Modelo C4

Claridad y Comunicación: Mejora drásticamente la comunicación de la arquitectura entre diferentes audiencias.

Abstracción Progresiva: Permite "hacer zoom" en el detalle sin abrumar al lector.

Fácil de Aprender: La notación es intuitiva y sencilla.

Flexible: Se adapta a diferentes tipos de arquitecturas (monolitos, microservicios, serverless) y tecnologías.

Fomenta la Modularidad: La necesidad de dibujar componentes y sus interfaces fomenta un mejor diseño modular.

Desventajas

No es una Notación Formal: A diferencia de UML, C4 no tiene una especificación formal estricta, lo que podría llevar a inconsistencias si no se establecen convenciones claras en el equipo.

Dependencia del Contexto: La comprensión total de un diagrama a menudo requiere el contexto de los diagramas de nivel superior.

Puede ser Verboso si se Abusa: Si no se limita el detalle en cada nivel, los diagramas pueden volverse complejos.

III - PROBLEMÁTICA

3.1 Descripción de Caso de Estudio

La provincia de San Juan presenta una distribución geográfica con numerosas zonas alejadas, las cuales, debido a su escasa población, no cuentan con servicios farmacéuticos cercanos; por lo cual, a la hora de recurrir a una farmacia, esta se puede encontrar muy alejada de dicha ubicación o inclusive puede significar un verdadero reto para la población debido a factores como el horario. La falta de disponibilidad inmediata de farmacias puede derivar en retrasos en el acceso a tratamientos médicos, afectando la calidad de vida de la población y favoreciendo prácticas como la automedicación.

El paradigma que encuadra esta investigación es el cognitivo, ya que los Clientes serán sujetos activos en el proceso informacional, el foco se encuentra en la necesidad de información, sobre las farmacias disponibles, del usuario, como así también de su interacción con la plataforma.

La digitalización de los servicios de salud permite mejorar la accesibilidad y eficiencia en la gestión de información médica. En este contexto, el desarrollo de una plataforma que facilite la geolocalización de farmacias y la gestión de recetas digitales en la provincia de San Juan representa una solución innovadora que responde a las necesidades de la población a la hora de ubicar una farmacia que cumpla con determinadas características.

La implementación de una plataforma que permita la rápida identificación de farmacias disponibles en tiempo real, que cumplan con ciertos factores de relevancia para el usuario (como es el caso de la obra social, servicios de vacunación o inyectables) y la gestión de recetas digitales, significa una solución clave para mejorar el acceso a la atención farmacéutica en zonas rurales y alejadas. Además, al proporcionar la ubicación exacta, horarios de atención, disponibilidad de medicamentos y la posibilidad del envío a domicilio, se optimiza, de manera considerable, la planificación del desplazamiento y se reduce la incertidumbre del usuario al momento de buscar una farmacia operativa.

Complementario a esto, la implementación de recetas digitales ha cobrado relevancia en los últimos años, sobre todo durante la Pandemia por Covid-19 del

año 2020, permitiendo la digitalización de prescripciones médicas, reduciendo así errores en la dispensación de medicamentos y alineándose con las tendencias globales en la digitalización de aspectos y recursos relacionados con la salud, promoviendo la interoperabilidad médica y mejorando la seguridad del manejo de información confidencial respecto a la salud del paciente.

En Argentina, la Ley 27.553 regula la prescripción de recetas digitales y electrónicas, estableciendo la validez de estas en todo el territorio nacional (Ver Anexo I - Ley 27553); así mismo el Decreto 345/2024 refuerza su aplicación y normativa, haciendo hincapié en garantizar la autenticidad de las recetas emitidas (Ver Anexo II - Decreto 345/2024). Ambas regulaciones son fundamentales para la implementación de una plataforma que permita la correcta gestión de un recurso tan importante como lo son las recetas médicas a la hora de solicitar una medicación.

Considerando el presente marco legal, y dado que para el acceso a bases de datos de recetas electrónicas es necesaria una autorización y aprobación por parte del Gobierno Nacional, y teniendo en cuenta que cada farmacia posee un sistema autorizado para el acceso a esta base de datos; se pretende implementar en la plataforma un chat bilateral entre el Cliente y la Farmacia, mediante la cual se podrá enviar el código de barras o número único identificatorio de la receta, de tal manera que la farmacia pueda acceder a ella para visualizar la medicación solicitada o autorizar esta receta. Si el Cliente no desea proporcionar dicho número o la receta no sea electrónica, se podrá proporcionar por escrito o foto la medicación solicitada o la receta papel a modo de consulta, y, si se desea adquirir la medicación, se hará necesaria la presencia del Cliente en la farmacia. El pago no es considerado en esta plataforma al menos de momento, será externalizado y delegado a la farmacia.

Otro elemento importante a tener en cuenta es la información que se proporciona acerca de la Farmacia que es consultada por el usuario, la información es la siguiente: nombre, dirección, teléfono de contacto (si posee), correo o dirección mail (si posee), obras sociales que recibe actualmente, servicios que ofrece (servicio de vacunación, colocación de inyectables, laboratorio), horarios de atención comercial, días y horarios en los que cumplirá turnos rotativos

(de tal manera que puedan ser registrados en la aplicación, el día correspondiente, la farmacia figurará como "abierta" o "disponible"). Esta información es extremadamente útil para simplificar y agilizar la búsqueda de una farmacia a través de diversos filtros.

Es de suma importancia, proporcionar una ruta GPS a través de un Sistema de Información Geográfica (SIG), "Un Sistema de Información Geográfica (SIG) es un sistema informático para la captura, almacenamiento, manipulación, análisis, gestión y presentación de datos espaciales o geográficos." (Longley, Goodchild, Maguire, & Rhind, 2015, traducción propia). A fines de simplificar la experiencia del usuario, como así también el desarrollo de la aplicación, se recurre a una API externa para situar geográficamente las farmacias disponibles: Google Maps Platforms.

Google Maps es una app que permite ir a cualquier lugar gracias a una navegación sencilla. Muestra instrucciones sobre cómo llegar, utiliza información del tráfico en tiempo real para encontrar la mejor ruta, proporciona alertas, señalizaciones, hitos, indicaciones por voz y alertas sobre el tráfico. Esta empresa proporciona, a su vez, diversas APIs diseñadas para integrar este SIG de manera sencilla a diversos desarrollos, con la capacidad de crear mapas interactivos y personalizados, proporcionando experiencias geoespaciales al usuario.

El desarrollo de una plataforma para el sector salud implica una constante evolución y adaptación, por lo que es propicio el uso de los recursos y herramientas anteriormente nombrados, en conjunto con el cumplimiento de normativas y estándares específicos que garanticen la protección de los datos personales de los usuarios, la Ley de Protección de Datos Personales (Ley 25.326) y las regulaciones del Ministerio de Salud que establecen los lineamientos para la gestión de la información médica y, adicionalmente, la Ley 27.553 (Ver Anexo I - Ley 27553) y el Decreto 345/2024 (Ver Anexo II - Decreto 345/2024) regulan el uso de recetas digitales, es posible desarrollar una plataforma que beneficie a los usuario en la búsqueda de una Farmacia que cumpla con la necesidad de recurrir a un tratamiento médico.

Complementario a esto, y a fin de potenciar su uso, es indispensable la incorporación de alianzas estratégicas. En relación con esto, la colaboración con el Colegio Farmacéutico de San Juan resulta un elemento o pilar esencial, ya que aporta respaldo institucional y facilita la adopción de esta plataforma por parte de las farmacias locales; esta alianza no solo refuerza la credibilidad, sino que, además de esto, permite establecer estándares de calidad, asegurando que la información y los servicios ofrecidos cumplan con los estándares del sector.

Además de esta alianza, surge la necesidad de incorporar empresas de delivery o entrega a domicilio como elemento crucial para optimizar la logística de entrega de medicamentos aquellas personas que no cuentan con medios propios o ajenos para desplazarse a la farmacia que se encuentra disponible. En este contexto, Uber se presenta como una opción destacada debido a su amplia infraestructura y gran experiencia en el transporte de personas y paquetes. Esta empresa demuestra su capacidad para ofrecer, además de su servicio de transporte de personas, servicios de entregas a través de su plataforma, aprovechando la tecnología móvil y de geolocalización en tiempo real. Esto permite ampliar la red de cobertura, llegando a zonas alejadas y reduciendo tiempos de espera, optimizar la logística posibilitando el seguimiento en tiempo real y garantizando transparencia, mejorar la experiencia de usuario al proporcionar comodidad al momento de solicitar su medicación, etc.

Esta alianza con Uber se complementa con la del Colegio Farmacéutico de San Juan, lo que contribuye a crear un ecosistema robusto y confiable. Así, se garantiza no solo la correcta localización de farmacias y gestión de recetas digitales, sino que, de ser necesario para el usuario, también se encontraría disponible entrega de los medicamentos solicitados a domicilio.

IV – SOLUCIÓN PROPUESTA

4.1 Descripción de la Solución

4.1 Descripción del Producto

Farmadip es una plataforma digital para la geolocalización de farmacias y solicitud de medicamentos mediante recetas electrónicas. Esta plataforma muestra en un mapa la posición de cada farmacia cercana al usuario, proporcionando no solo la ubicación, sino también datos esenciales para la solicitud de medicamentos de manera online, estos datos son: nombre, número de teléfono, obras sociales que se reciben, horarios de atención, estado (Abierto, Cerrado, De Turno). Adicional a esto, es opcional aplicar filtros de búsqueda que ajusten la cantidad de farmacias mostradas a las preferencias de la persona que desea comprar un medicamento (usuario comprador).

Haciendo uso de esta información, el usuario comprador podrá seleccionar la farmacia que se ajuste a sus necesidades o preferencias. Una vez hecha la elección y de la mano con un chat en tiempo real, el usuario comprador podrá contactar al usuario farmacia para tomar conocimiento acerca de la disponibilidad de los medicamentos que el usuario comprador desea adquirir. Esta solicitud se podrá realizar a través del chat de diferentes maneras: enviando el CUIR o Clave Única de Identificación de Recetas, de manera tal que el usuario farmacia pueda ingresarlo en un sistema autorizado para su lectura; podrá ingresar los medicamentos de forma escrita si no desea proporcionar el CUIR o enviar una imagen o documento pdf, el cual se prevé, sea la receta electrónica emitida por un profesional autorizado. Cabe destacar que este sistema no emite recetas electrónicas ni hará uso de bases de datos relacionadas a las mismas, solo se encargará del traspaso de información y consulta de disponibilidad de los medicamentos.

En caso de que el usuario comprador desee adquirir los medicamentos y no haya proporcionado el CUIR, deberá enviarlo al usuario farmacia para la autorización de la receta electrónica. El pago se hará a través de una plataforma externa y se validará el pedido del usuario comprador. Esta plataforma digital no

será un e-commerce, solo trabajará con la transacción del pedido, sin selección de productos dentro de la plataforma. En caso de añadir productos fuera de la receta o proporcionar medicamentos en reemplazo de otros, deberá ser aclarado y señalado en el pedido, esto figurará en el historial de pedidos del usuario comprador.

Es opcional retirar el pedido por la farmacia o solicitar a la misma el envío del pedido al domicilio del usuario comprador, el mismo formará parte de los datos de este usuario y podrá ser ingresado en plataformas de terceros como Uber. El momento de la preparación, disponibilidad para retirar o despacho del pedido y la recepción del mismo, deberá ser indicado también en la plataforma digital Farmadip, informándole al usuario comprador el estado de su pedido y la entrega al usuario farmacia. Con esto el ciclo de la solicitud y entrega de medicamentos estaría completo.

4.2 Características Técnicas del Producto

El desarrollo frontend se implementa mediante el uso del framework Flutter, un framework de código abierto, desarrollado por Google, ideal para este caso, ya que fue creado para el desarrollo multiplataforma, permitiendo la compilación a partir de un único código base. Al principio, en su lanzamiento en el año 2018, Flutter solo era compatible con desarrollos móviles; al momento del desarrollo del presente trabajo, Flutter es compatible con seis plataformas: iOS, Android, web, Windows, MacOS y Linux, por lo que este framework simplifica el proceso de creación de interfaces coherentes para aplicaciones en estas seis plataformas.

En el entorno de este marco de trabajo o framework, se selecciona como lenguaje de programación para realizar la codificación, el lenguaje Dart. Dart es un lenguaje de código abierto desarrollado por Google, orientado a objetos y con análisis estático de tipos. Se diseño con el objetivo de hacer que el desarrollo sea lo más cómo y rápido posible para los desarrolladores, lo que lo convierte en una gran opción para esta implementación. Posee un conjunto extenso de herramientas, entre ellos su propio gestor de paquetes, varios compiladores/transpiladores, un

analizador y formateador. Además, la máquina virtual de Dart y su compilación Just-In-Time hacen que los cambios realizados en el código puedan ser ejecutados inmediatamente.

Para la gestión de base de datos, y completando la arquitectura basada en componentes descripta en el marco teórico del presente trabajo, se opta por Supabase, una plataforma de BaS (Backend As Service) para el desarrollo de aplicaciones web y móviles que se encuentra en la nube. Se realiza esta elección debido a que esta plataforma simplifica el desarrollo presentando diversas funcionalidades, variadas y de fácil uso, las cuales contribuyen a que el desarrollo sea más rápido. Funcionalidades de interés:

Relacional y Estructurado: Utiliza PostgreSQL, uno de los sistemas de base de datos relacionales más populares del mundo, lo que permite manejar datos estructurados y relaciones complejas. Especialmente útil para manejar datos estructurados como la información de las farmacias o el manejo de recetas digitales, garantizando las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) en las operaciones, lo que es crucial cuando se manejan transacciones sensibles, como las relacionadas con información de salud.

Realtime: Supabase ofrece capacidades en tiempo real mediante subscripciones a websockets. Llevado a nuestro caso y con un ejemplo práctico, cuando se actualice la disponibilidad de una farmacia, este cambio se reflejará inmediatamente en la aplicación, sin necesidad de recargar la interfaz. Además, se podrán implementar notificaciones para el usuario en tiempo real, haciendo su experiencia más dinámica, mejorando su interacción con la aplicación.

Autenticación de Usuarios: Supabase ofrece un sistema de autenticación integrado, el cual permite gestionar usuarios, roles y permisos, lo cual permite un manejo minucioso del acceso a datos sensibles como lo son aquellos referidos a la salud.

Almacenamiento en la Nube: Supabase ofrece un módulo de almacenamiento en la nube, que permite guardar archivos y recursos multimedia, de forma escalable y segura. Esto resulta extremadamente útil al momento de almacenar recursos complementarios como imágenes de recetas digitales.

Supabase posee una estructura escalable, segura y rica en funcionalidades, proporcionando una solución completa para el desarrollo backend de este proyecto.

4.3 ARQUITECTURA DEL SISTEMA

La arquitectura en la que se enfoca el sistema propuesto será una arquitectura en capas, la misma será complementada con C4 Model, facilitando su representación conceptual y, por ende, su comprensión.

C4 Model es un modelo jerárquico que permite representar la arquitectura del sistema desde distintos niveles de abstracción. Esta representación permite desde una visualización global del sistema hasta los elementos más específicos del código.

C4 Model está basado en cuatro niveles jerárquicos de diagramas que permiten representar y describir la arquitectura, cada uno en nivel de abstracción decreciente y detalle creciente.

Nivel 1: Contexto

En este nivel se destaca la representación del contexto de sistema, es decir, cómo la plataforma de Farmadip se relaciona con los usuarios involucrados y con sistemas externos.

Este nivel está íntimamente relacionado con la Capa de Integración Externa, la cual incorpora aquellos servicios externos que amplían las funcionalidades de la plataforma, como lo son pasarelas de pago (como Mercado Pago) y sistemas de envíos de terceros (como UBER o DiDi), entre otros. Este nivel permite aclarar qué usuarios estarán relacionados y de qué manera interactuarán con el sistema. A su vez, las interacciones de los servicios contenidos en la Capa de Integración Externa son representados en relación con la plataforma de Farmadip. La representación de la relación entre la plataforma y estos servicios externos resulta extremadamente importante para entender el alcance de la funcionalidad y los servicios prestados por la misma. En este caso los usuarios compradores podrán

buscar, leer o revisar información y/o dirigirse a la farmacia más cercana, como así también solicitar medicamentos mediante un servicio de chat integrado; los usuarios farmacias podrán brindar datos esenciales para ser consultados y/o geográficamente localizados, también podrán responder consultas y tomar pedidos de medicamentos solicitados por los usuarios compradores.

A continuación, se presenta el diagrama de contexto modelado para Farmadip:

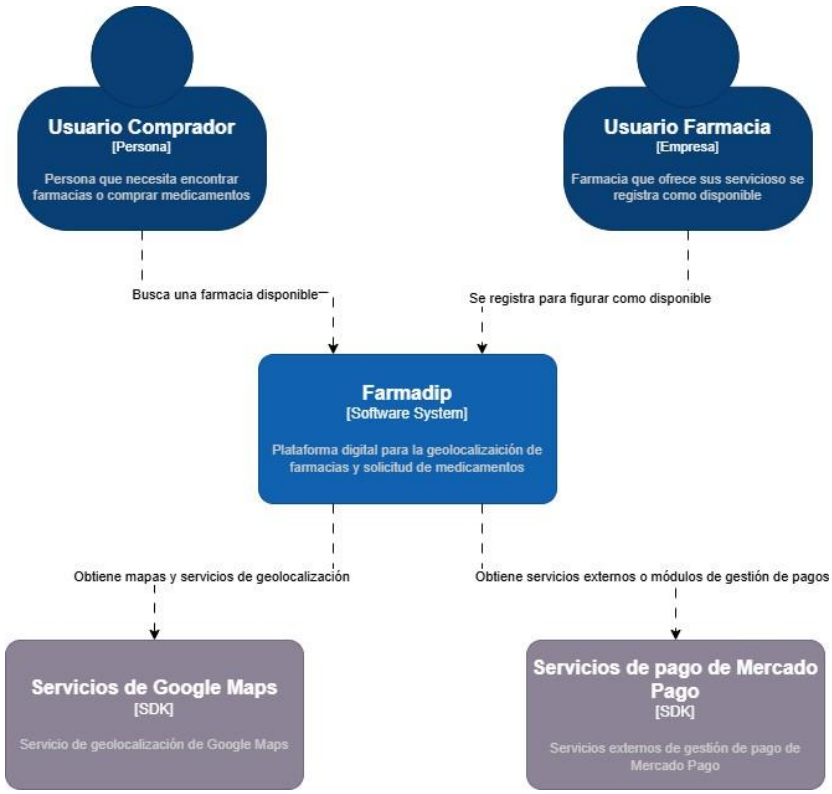

Figura 15: Diagrama de Contexto del Sistema

Nivel 2: Contenedores

En este nivel se representa la estructura interna de la plataforma, es decir, refleja cualquier entidad que ejecuta código o almacena datos. Cada uno de

estos contenedores cumple una función específica, se comunican entre sí y garantizan el funcionamiento integral de la plataforma.

Uno de los contenedores principales a considerar es el contenedor Fontend o Aplicación Móvil (puede extenderse a un desarrollo web), altamente relacionada con la capa de presentación y desarrollada con Flutter, este framework multiplataforma permite la implementación de interfaces en distintos dispositivos. Ese contenedor estará desarrollado en Flutter, brindando una interfaz sencilla, clara y óptima para el usuario. Permitirá el registro de usuarios tanto compradores como usuarios farmacias, inicio de sesión, solicitud de medicamentos mediante un chat en tiempo real, visualización de información referente a las farmacias, visualización de la ubicación de las farmacias, verificar el estado de las farmacias (si se encuentran abiertas, cerradas o de turno rotativo o atención 24h).

Otro contenedor principal es el contenedor Backend, se asocia estrechamente con la Capa de Servicios que se sustenta en Supabase como solución de tipo Backend as a Service (BaaS). La misma permite integrar diversos servicios que son esenciales para el desarrollo: gestión de la base de datos, autenticación de usuarios y gestión o control de estos a través de la implementación de roles, comunicación en tiempo real a través de WebSockets y almacenamiento en la nube, permitiendo a su vez la escalabilidad de la aplicación. La gestión de datos mediante PostgreSQL garantiza propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), posee control de concurrencia, el cual permite que múltiples usuarios accedan a la misma base de datos simultáneamente sin bloqueos entre sí, manteniendo la integridad de datos durante transacciones concurrentes; así mismo permite la escalabilidad y el procesamiento eficiente de grandes cargas de trabajo. Como podemos ver, Supabase es un contenedor en sí mismo, expone APIs, gestiona la autenticación y almacena datos en PostgreSQL.

En la figura siguiente se observa el diagrama de contenedores de Farmadip.

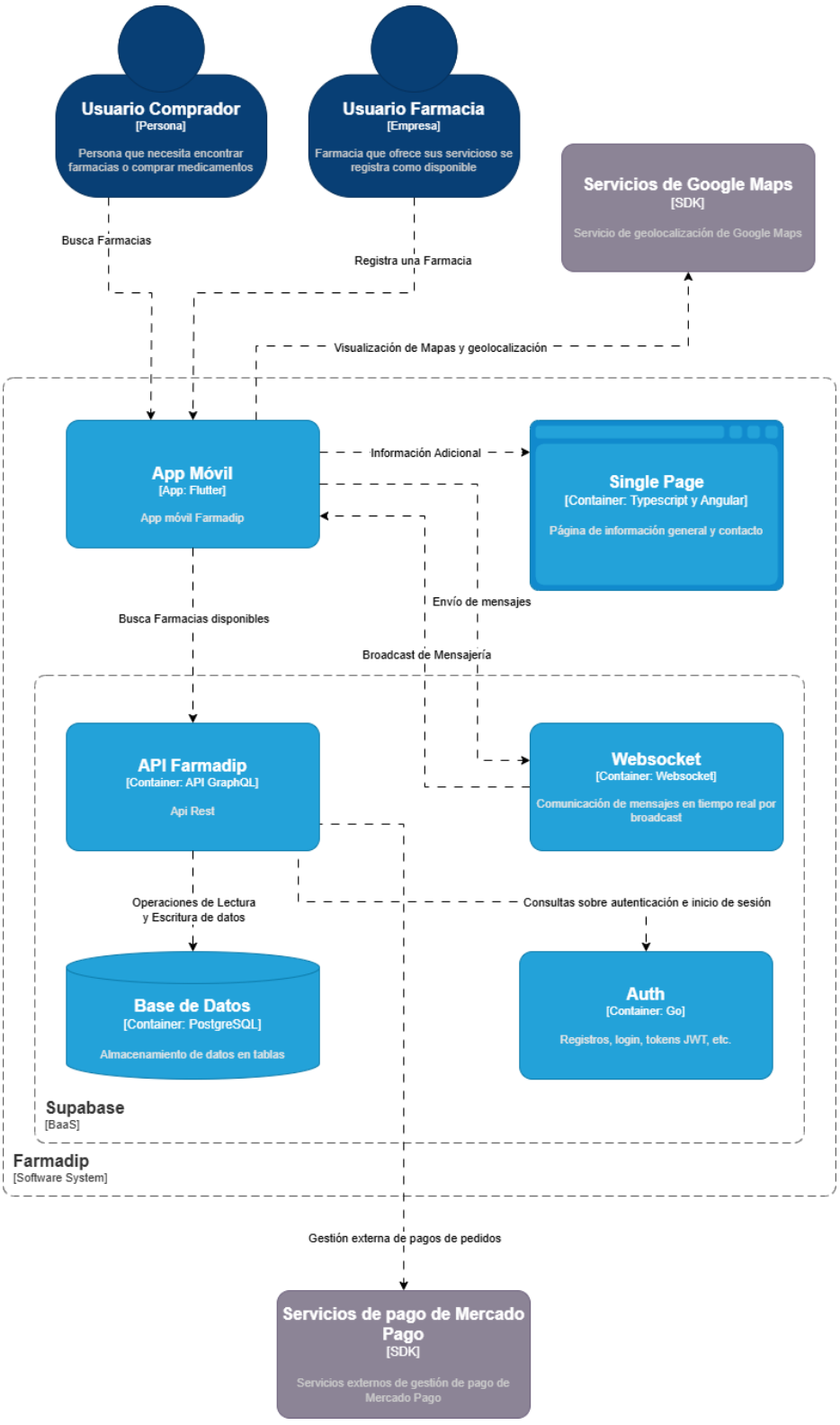

Figura 16: Diagrama de Contenedores del Sistema

Un contenedor relacionado con este es el Contenedor de Base de Datos, como se mencionó anteriormente, Supabase almacena datos en PostgreSQL y representa la Capa de Datos. Los datos se almacenarán en tablas, algunas de las principales tablas serán la de Farmacias, la tabla de Usuarios Compradores y la tabla de Pedidos. El sistema relacional implementado dentro de Supabase, permite garantizar integridad, disponibilidad y confidencialidad de los datos almacenados.

Por último, el contenedor que complementa las funcionalidades de Farmadip, es el contenedor de Integraciones Externas. El mismo incluye todos aquellos servicios de terceros de los que Farmadip hace uso, esto ayuda a la plataforma a aumentar su alcance sin incrementar la complejidad del código. Se implementan servicios externos como Mercado Pago para la integración de pagos directos de los pedidos sin necesidad de incorporar wallets o cuentas bancarias en Farmadip, el uso de Google Maps para visualizar la ubicación de cada farmacia y dirigirse a la más cercana, o el uso de UBER como posible agente de entregas de pedidos. Servicios como Mercado Pago no limitan la funcionalidad de la aplicación, el hecho que se pueda abonar de manera digital a través de esta wallet, no impide el pago en efectivo en la Farmacia, el usuario Farmacia tendrá a su disposición la selección del pago digital o el pago en efectivo, pudiendo quedar confirmado el pedido hasta que el usuario lo retire al dirigirse al establecimiento. El servicio de UBER tampoco es limitante, sino que ofrece una alternativa viable para aquellas empresas farmacéuticas que no poseen de una flota propia para dirigirse al domicilio del usuario, el usuario Farmacia puede utilizar un servicio propio o delegarlo a un tercero como lo es UBER, DiDi, etc. Los servicios de terceros de envíos mencionados poseen pagos de envíos contra entrega, por lo que pueden ser abonados en efectivo de ser necesario.

Nivel 3: Componentes

Este nivel contiene los componentes lógicos o módulos funcionales. Cada componente cumple un propósito específico y contribuye al comportamiento del sistema. Se destacan los siguientes componentes o módulos:

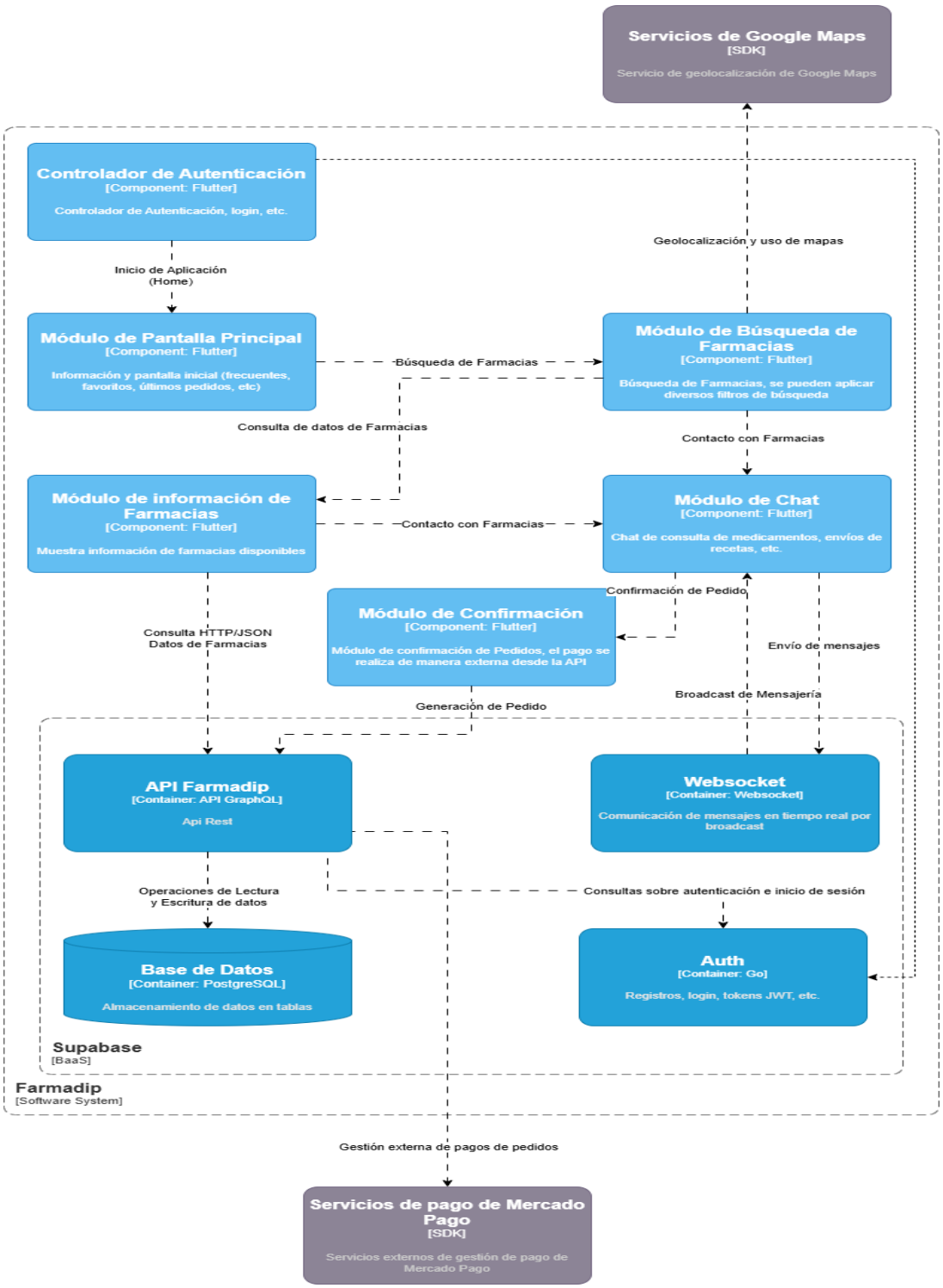

Figura 17: Diagrama de Componentes del Sistema

Usuarios: incluye todo lo relacionado a inicio y registro de sesión, roles y autenticación del usuario, datos personales como nombres, dirección, edad, dni, fecha de nacimiento.

Farmacias: incluye todos los datos relacionados a Farmacias, como horarios, dirección, turnos rotativos, obras sociales que recibe, otros servicios que presta.

Pedidos: datos como imagen de la receta, medicamentos, fecha de pedido, estado del pedido (confirmado, pendiente, cancelado, rechazado por Obra Social), observaciones en caso de que lo necesite, ya que en muchas ocasiones se reemplazan medicamentos por otros similares, pero de diferente laboratorio.

Mensajes: es necesaria la existencia de un módulo de mensajes que almacene remitente, destinatario, mensaje y fecha y hora. De esta manera se obtendrá un registro persistente de los mensajes enviados entre usuarios. Este registro es usado como historial, se podría consultar en caso de resultar algún inconveniente o de ser necesaria una consulta sobre los detalles de un pedido.

Geolocalización: módulo de integración con Google Maps.

Logística y reparto: integración con servicios de UBER, DiDi u otros servicios de logística, envíos o delivery.

Nivel 4: Código

Este nivel se centra en la descripción o especificación de la implementación de cada componente anteriormente nombrado. Dado que actualmente no se ha alcanzado el nivel de código, no es posible especificar el mismo a través del gráfico del presente nivel.

4.4 DISEÑO DEL PRODUCTO

El diseño inicial del producto, presentado en este apartado, se lleva a cabo con la herramienta Figma, la misma posee una interfaz sencilla, intuitiva y práctica para el diseño de aplicaciones.

Figura 18: Interfaz de inicio de la aplicación Farmadip

Es preciso el inicio de sesión por parte del usuario comprador, este puede usar su correo electrónico personal o iniciar sesión con Google o con Apple como se muestra en la Figura 19, facilitando el ingreso a la aplicación. La información proporcionada por esta acción es necesaria para realizar acciones dentro de la aplicación, como pedidos de medicamentos.

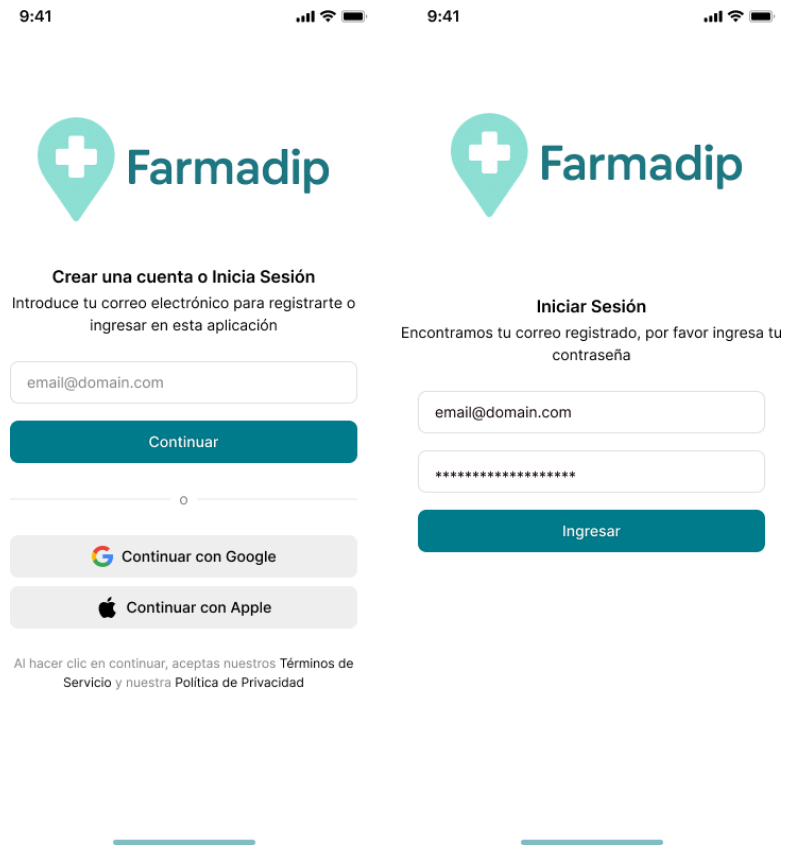

Figura 19: Interfaces de inicio de sesión

El usuario Farmacia también posee un inicio de sesión tradicional, como se muestra en la Figura 20, posteriormente debe añadir datos claves como obras sociales recibidas, turnos rotativos, horarios de atención comercial normal u otros servicios como vacunatorio.

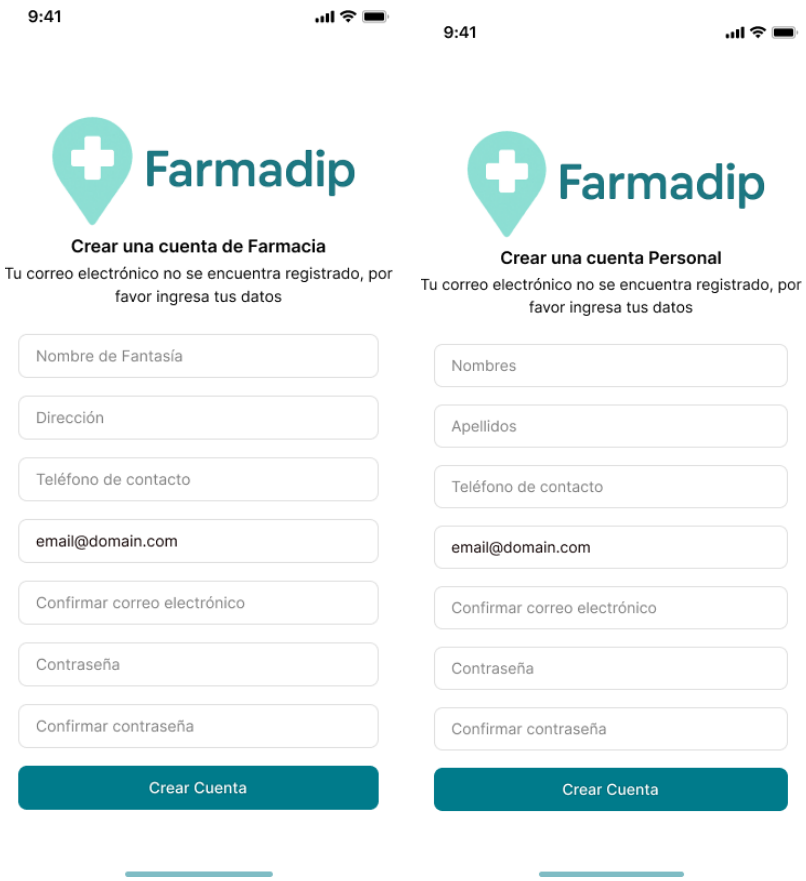

Figura 20: Interfaces de creación de cuentas de diferentes tipos de usuario

En ambos perfiles, una vez se ingresa en la aplicación, en la parte inferior, se visualizan íconos de acceso disponibles. Para el usuario comprador estos son, de izquierda a derecha, Home o Inicio, Mapa, Notificaciones, Mensajes, Favoritos, y Perfil, como se muestra en la Figura 21; mientras que para el usuario Farmacia estos son, de izquierda a derecha, Home o Inicio, Mapa, Notificaciones, Mensajes, y Perfil, representado en la Figura 22.

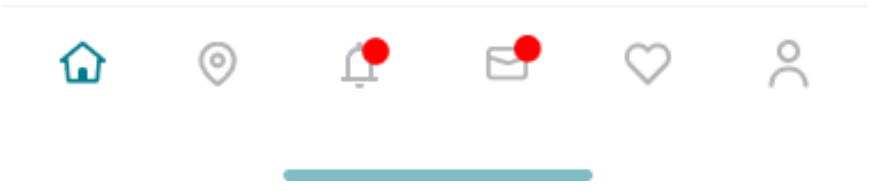

Figura 21: Accesos directos Comprador

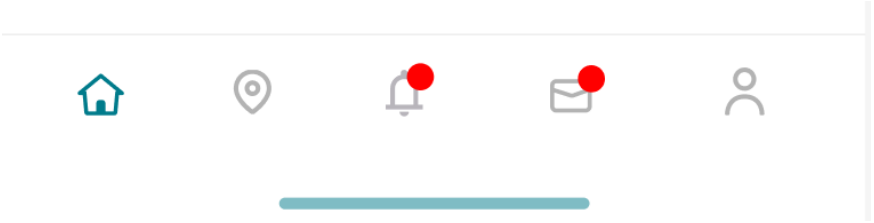

Figura 22: Accesos directos usuario Farmacia

Página de Inicio

Ambos usuarios tienen una pantalla de inicio, donde pueden ver sus últimos pedidos. Por parte del usuario comprador, se visualizan los últimos pedidos realizados y una lista como acceso rápido a aquellas farmacias que se encuentren de turno actualmente, en orden ascendente de distancias.

Figura 23: Interfaz de inicio para usuario Comprador

Por su parte, el usuario Farmacia, puede ver los últimos pedidos entregados; para este último, se puede considerar como sugerencia añadir un gráfico representativo de los pedidos en los últimos meses que facilite la lectura de la cantidad de solicitudes recibidos mediante la aplicación. La Figura 23 representa la pantalla de Inicio del usuario Comprador.

Página Mapa

El usuario comprador cuenta con un buscador, filtros configurables y un mapa interactivo. En el buscador donde ingresar el nombre de una farmacia conocida que desea encontrar, posterior a ello, se despliega una lista de coincidencias cercanas a su ubicación.

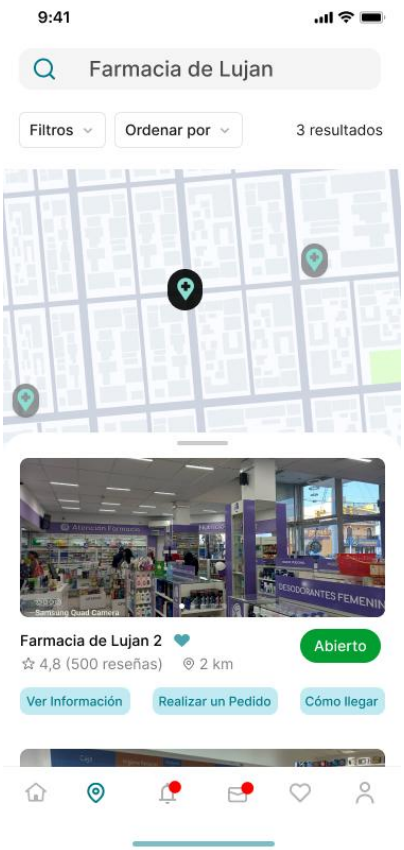

Figura 24: Interfaz de búsqueda de Farmacia basada en GIS

También se encuentra, bajo el buscador, un desplegable con filtros, aplicables a los resultados encontrados, estos filtros se aplican sobre la información

proporcionada por los usuarios Farmacia, como por ejemplo, filtrar por obra social recibida, por distancia desde el usuario comprador hasta la Farmacia o por servicio prestado. Así mismo se ordenan los resultados por distancia o por orden alfabético, ambos de manera ascendente o descendente.

Figura 25: Interfaz de especificación de detalle de Farmacia

Los resultados de la búsqueda pueden visualizarse como un ícono de punto de referencia con una cruz en el mapa. Así mismo, cada ítem de la lista de resultados coincidentes presenta en su información su estado actual (Abierto o Cerrado), un botón para ver datos detallados sobre la farmacia (teléfono, dirección, horarios, obras sociales, etc), un botón que redirige a un chat directo para realizar un pedido a la farmacia seleccionada, y un botón que redirige a Google Maps con la ruta más rápida desde la ubicación del usuario comprador hasta la farmacia.

Desde el usuario Farmacia, también pueden visualizarse los puntos referentes en el mapa que indican otras farmacias, se puede visualizar la información de otros usuarios farmacias, como así también tener una vista previa de sus propios datos. Se pueden enviar mensajes a otros usuarios Farmacia, está disponible el canal de contacto entre pares, proporcionando una red de apoyo para derivar compradores, o en caso de Farmacias con diversas sucursales, pueden tener contacto y derivar clientes.

Tanto en la Figura 24 como en la Figura 25 se puede observar el proceso de búsqueda completo de una farmacia. En la Figura 24 se realiza la búsqueda del establecimiento, mientras que en la Figura 25 se observa el detalle de la Farmacia seleccionada.

Página Notificaciones

Ambos usuarios tienen una pantalla donde se pueden visualizar las notificaciones recibidas. Aquellas notificaciones que no han sido leídas se visualizan con un puntito rojo en la parte izquierda de la notificación.

Figura 26: Interfaz de Notificaciones

Cada vez que una persona reciba un mensaje se puede visualizar en las notificaciones, también son visibles avisos del sistema relacionados al usuario, por ejemplo, completar datos requeridos, actualizaciones o promociones.

En la Figura 26: Interfaz de Notificaciones, es posible visualizar notificaciones "Leídas" y "No Leídas" marcadas con un punto rojo.

Página Mensajes

En la página de mensajes, los usuarios visualizan los chats entre usuarios, ordenados de manera ascendente según su antigüedad, mostrando en primera instancia aquellos más recientes.

Dentro de cada chat, los usuarios compradores pueden enviar archivos, facilitando la consulta sobre la disponibilidad de medicamentos, ya sea con fotos o archivos de la medicación solicitada, también pueden enviar recetas electrónicas que contengan especificados los medicamentos necesarios. Por su parte, el usuario Farmacia cuenta con un menú desplegable con la opción de generar un pedido de productos o medicamentos a raíz de dicho chat; elaborado el pedido, se puede remitir una confirmación al usuario comprador para que confirme el mismo.

Es importante destacar que esta aplicación no cuenta con la validación de la receta, ni con un almacenamiento de productos, tampoco con una tienda o carrito donde se añadan los productos a comprar; el chat solo es un nexo de conexión y comunicación entre el usuario comprador y el usuario Farmacia.

En la Figura 27: Bandeja de Mensajería, se representa la bandeja de mensajería del usuario Comprador.

Figura 27: Bandeja de Mensajería

Página Chat

Esta pantalla es visible para Farmacias y Compradores, y consta de burbujas de texto que engloban los mensajes enviados entre usuarios. Así mismo, en la parte inferior aparece un cuadro de texto que permite ingresar el mensaje a enviar. Junto a este figuran 3 botones, uno en forma de archivo para adjuntar archivos, uno en forma de emoticón y por último uno en forma de foto o imagen.

Como muestra la Figura 28, se entabla una conversación para solicitar medicamentos contenidos en una receta.

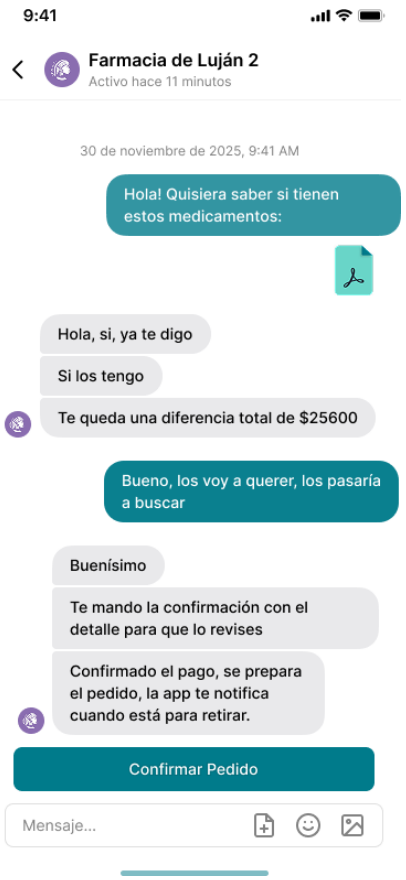

Figura 28: Interfaz de Chat Comprador - Farmacia

Pantalla Favoritos

Únicamente el usuario comprador tiene acceso a la pantalla de Favoritos, en ella cuenta con una lista de farmacias que el usuario marcó como favoritas, junto a las mismas se observa el estado de las farmacias (Abierto o Cerrado) para facilitar la búsqueda del usuario, tocar en una farmacia favorita despliega un menú con la opción de hacer un pedido o visualizar la información de la farmacia seleccionada. Si se decide realizar un pedido, se abre el chat con la farmacia, si se escoge visualizar la información, el usuario es redirigido al mapa, ubicando la farmacia seleccionada en el mapa y desplegando su información detallada.

En la figura siguiente se observa la lista de Farmacias favoritas marcadas por el Comprador.

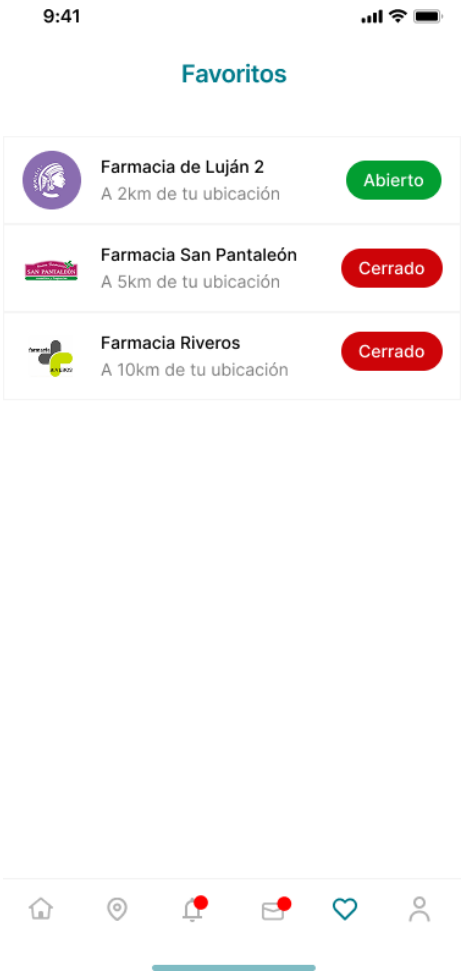

Figura 29: Interfaz de Farmacias Favoritas

Pantalla de elaboración de Pedido

La pantalla de Elaboración de Pedido es visible únicamente para los usuarios Farmacia, ya que son los que elaboran el pedido para el usuario comprador. La pantalla cuenta con el número de pedido, debajo se selecciona la modalidad de entrega, por ejemplo, retiro en sucursal o envío a domicilio. Por último, se encuentra el listado de artículos y un botón con el símbolo + para añadir artículos a la lista. Estos artículos se almacenan para otorgar una mayor facilidad al momento de generar el pedido. Se pueden importar listas generadas en documentos xls, que generalmente provienen de sistemas de stock, también pueden ingresarse nuevos productos. Se contempla a futuro una integración con

APIs o sistemas de control y gestión de stock, ampliando así las funcionalidades de la plataforma.

En la Figura 30se pueden observar los datos necesarios para registrar el pedido, como así también el botón de confirmación.

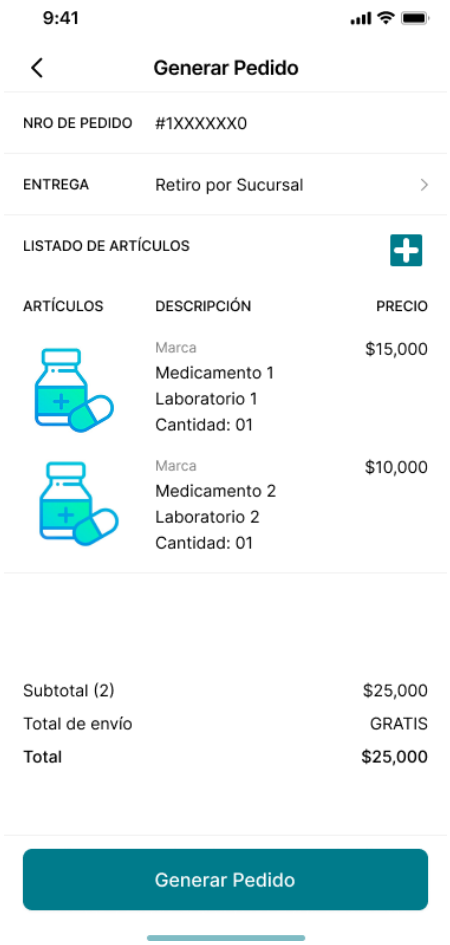

Figura 30: Interfaz de Generación de Pedido

Pantalla de Resumen de Pedido y Confirmación de Pedido

Una vez generado el pedido, el usuario comprador puede visualizar los detalles del pedido en una pantalla similar a la de la generación de pedido, con la diferencia de que cuenta con más información como el medio de pago, dirección de entrega, y posibilidad de aplicar algún cupón propio de la aplicación.

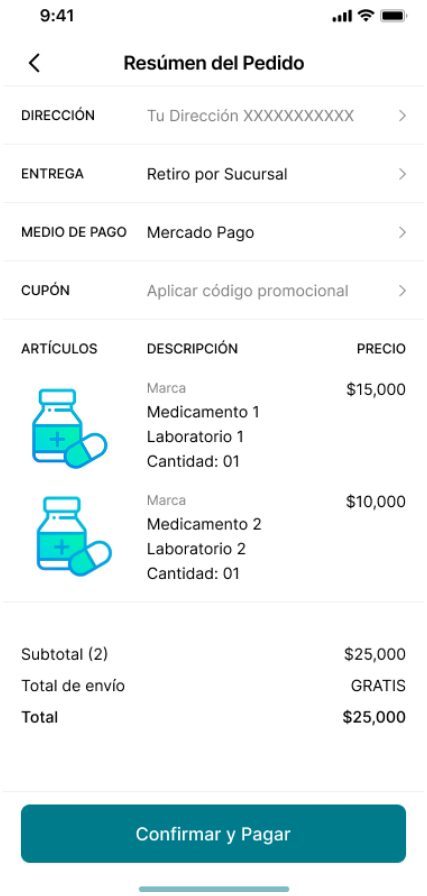

Figura 31: Interfaz de Resumen del Pedido

Si los datos del pago son correctos, el usuario procede a pagar. Si el usuario elige realizar el pago mediante Mercado Pago, es redirigido a la plataforma correspondiente para el pago; si selecciona efectivo, el pedido es confirmado automáticamente.

En la Figura 28: Interfaz de Chat Comprador - Farmacia, además del chat podemos observar un botón de confirmación del pedido, ese botón lleva a la pantalla representada en la Figura 31 y posteriormente la Figura 32.

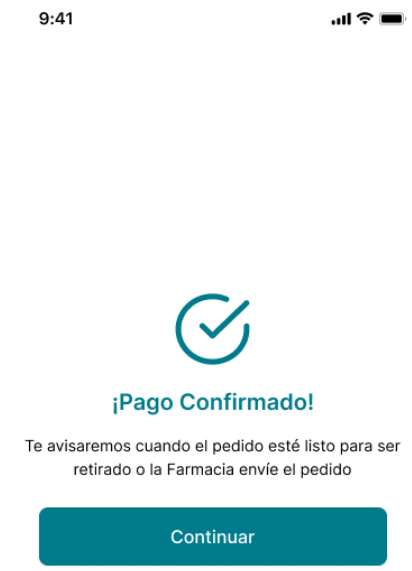

Figura 32: Interfaz de Confirmación del Pedido

Pantalla de Estado de Pedido

Esta pantalla es visible solamente para el usuario comprador. El detalle cuenta con la descripción del pedido realizado, el número de pedido, estado del pedido, línea de tiempo que indica el proceso de preparación y disponibilidad para retirar. Así mismo se puede contactar a la Farmacia que prepara el pedido, como así también contactar a soporte o ayuda al usuario.

El usuario Farmacia, tiene una pantalla para actualizar el estado del pedido que está preparando, de esta manera el usuario Comprador tiene un estado actualizado de su compra.

Una vez listo el pedido para ser entregado, el usuario Comprador es notificado, para que pueda retirar su pedido o estar pendiente de su entrega.

La Figura 33 muestra el estado del pedido del usuario Comprador, como así también contactar a la Farmacia a cargo del pedido.

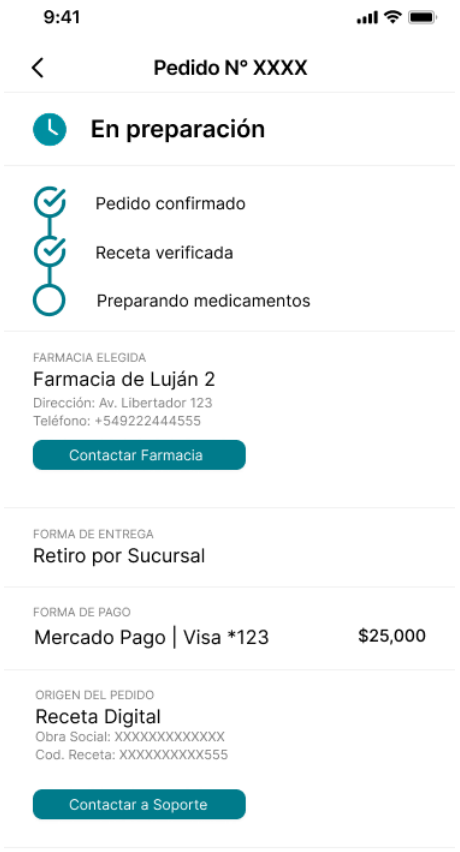

Figura 33: Interfaz de visualización de Estado de Pedido

Pantalla de Retiro

Una vez retirado el pedido, se visualiza un mensaje de confirmación para el usuario Comprador, y este podrá volver al inicio.

El usuario Farmacia también tiene un mensaje de confirmación, indicando que el pedido fue entregado con éxito. Si ocurre algún error, es notificado en la pantalla del detalle del pedido, indicando en específico si hubo un problema con la elaboración del pedido, con la entrega, o por otro motivo.

El estado de Retiro se visualiza en la Figura 34, mientras que la confirmación de Retiro se visualiza en la Figura 35

Figura 34: Interfaz de pedido listo para retirar

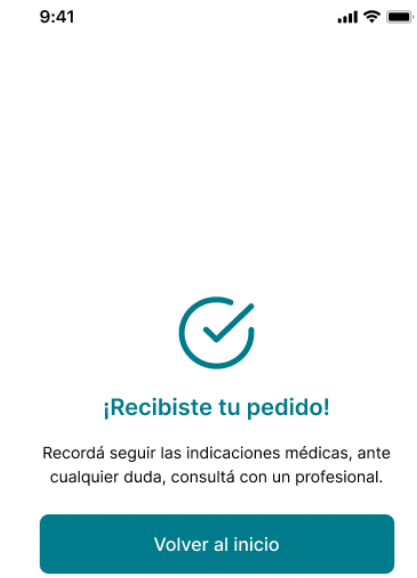

Figura 35: Confirmación de Retiro de Pedido

Esta propuesta de mejora tiene como objetivo acercar una solución a la problemática planteada en el capítulo anterior. Para la elaboración de la misma, se aplican lineamentos generales de Análisis y Diseño de Arquitectura de Sistemas.

.

V - CONCLUSIONES

5.1 CONCLUSIONES

En el presente trabajo se planteó desarrollar una plataforma digital para la geolocalización de farmacias y solicitud de medicamentos mediante recetas electrónicas. Para cumplir con este objetivo, se tomó como caso de estudio una zona alejada en la provincia de San Juan, con el fin de hacer un relevamiento de las farmacias de la zona, determinando que esta localidad no cuenta con la atención en turnos rotativos de 24hs de una farmacia en un radio menor a 18km, lo que permitió analizar la problemática en profundidad e investigar el marco legal referente al área Farmacéutica y el manejo de recetas electrónicas.

El problema principal que se destaca es la distancia desde la localidad seleccionada hasta la farmacia con atención en horarios rotativos de 24hs más cercana, como así también la dificultad de desplazamiento hasta la misma. Así mismo existe siempre la posibilidad de que la farmacia a la cual se recurría no contara con la medicación necesaria en stock, lo que obligaba a las personas a recorrer otras farmacias, a veces incluso en horario nocturno, en busca de aquellas que contaran con el medicamento y que, además en muchos casos, aceptaran una determinada obra social o prepaga.

Se propuso como solución al problema planteado, la implantación de una plataforma y se determinaron posibles funciones que debería implementar.

Una vez identificadas las funcionalidades concretas de la plataforma, se investigaron los frameworks, lenguajes y herramientas que mejor se adaptaran al desarrollo. Desde el punto de vista técnico, la plataforma fue diseñada bajo una arquitectura en capas, adecuada para sistemas sencillos, garantizando una correcta separación de responsabilidades entre la presentación, la lógica de negocio y el acceso a datos, lo que favorece la mantenibilidad, escalabilidad y evolución del sistema. El frontend fue pensado usando Flutter y Dart, lo que permite la construcción de una interfaz multiplataforma a través de un único código base. Para la implementación backend, se determinó el uso de Supabase como plataforma Backend as Service (BaaS), facilitando la autenticación de usuarios y la comunicación mediante APIs.

Así mismo, se investigaron servicios externos para la geolocalización de farmacias y se determinó que la más adecuada para el presente desarrollo son los servicios brindados por la API de Google Maps, la cual incorpora funcionalidades de geolocalización a través de un Sistema de Información Geográfica (SIG).

Por último, si bien no se desarrolló la implementación del sistema a nivel de código, se elaboró un prototipo navegable de la interfaz de usuario mediante Figma, que permite simular el flujo de navegación del sistema y las principales funcionalidades de la plataforma. Este prototipo navegable fue utilizado como una demostración conceptual del funcionamiento del sistema, permitiendo validar el diseño de las pantallas, la experiencia de usuario y la coherencia de los procesos definidos, aportando evidencia adicional sobre la viabilidad de la solución propuesta.

Se consideran cumplidos los objetivos planteados y se destaca que el presente trabajo realiza un aporte a las ciencias de la información, debido a que parte de un problema general que, para su abordaje, se acotó su alcance delimitándolo a una zona geográfica específica. La metodología que guió el proceso es una metodología basada en arquitectura de software. La solución resultante se concretó mediante una solución basada en un modelo de Cloud Computing utilizando Backend as Service, que representa una alternativa actualizada.

5.2 Líneas Futuras de Trabajo

Como primera línea de trabajo futuro, se propone considerar la implementación de un módulo de entregas, en el cual los propietarios de la farmacia puedan incorporar sus propias unidades de reparto, como así también compartir información en tiempo real de la ubicación del móvil repartidor, esto le brinda seguridad y accesibilidad a los clientes, como así también expande el área de cobertura de entregas para la farmacia.

Otro cambio considerable es la integración con los sistemas actuales que poseen las farmacias para la validación de recetas electrónicas, lo cual incurre en una investigación más a fondo acerca de la normativa legal de las mismas, ya que dichos sistemas se encuentras muy restringidos y aprobados únicamente por el Gobierno Nacional.

Una pasarela de pagos propia o la integración con múltiples pasarelas de pago como Google Pay, Apple Pay, entre otras, puede diversificar la propuesta, abriendo las puertas a pagos digitales para usuarios que no poseen Mercado Pago como billetera virtual.

Una funcionalidad que escapa al objetivo inicial, pero es una propuesta muy completa, es incluir una sección de e-commerce en la plataforma. Esta propuesta puede incurrir en un diseño y e investigación mucho más amplia, en la que se pueden tomar dos caminos: uno de ellos lleva a la investigación sobre la normativa legal para la venta de medicamentos online en Argentina, pudiendo incluirlos o no en un catálogo de compra; otra opción es incluir productos que pueden encontrarse en una farmacia pero que no son medicamentos como tal, como por ejemplo pastilleros, elementos de aseo personal, perfumería, etc. Como así también incluir la posibilidad de la reserva de turnos de servicios prestados por la farmacia, como lo son los vacunatorios.

Por último, este sistema es tan versátil que puede ser extendido a otros rubros donde se necesite, llevando el consumo de bienes o servicios a zonas donde, al

menos inicialmente, no se cuenta con cobertura o con la posibilidad de dirigirse a algún proveedor cercano.

REFERENCIAS BIBLIOGRÁFICAS

BIBLIOGRAFÍA

Amazon Web Services. (2025). *AWS*. Obtenido de What is Middleware?: https://aws.amazon.com/what-is/middleware/?nc1=h_ls

AWS. (2025). *Diferencia entre la arquitectura monolítica y la de microservicios*. Obtenido de Amazon Web Services: https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/

AWS. (2025). *Diferencia entre SOA y microservicios*. Obtenido de Amazon Web Services: https://aws.amazon.com/es/compare/the-difference-between-soa-microservices/

AWS. (2025). *Front end y back end en el desarrollo de aplicaciones*. Obtenido de Amazon Web Services: https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/

Brown, S. (2022). *The C4 model for visualising software.* Lean Publishing Process.

Cámara de Diputados de San Juan. (2014). *LEY Nº 67-Q – Código Sanitario de la Provincia de San Juan.* San Juan, Argentina: Digesto Jurídico de la Provincia de San Juan.

El Presidente de la Nación Argentina. (1967). *Ley 17.565.* Buenos Aires, Argentina: Boletín Oficial de la República Argentina.

El Presidente de la Nación Argentina. (2024, 19 de abril). *Decreto 345/2024.* Ciudad de Buenos Aires: Boletín Oficial de la República Argentina.

El Senado y Cámara de Diputados de la Nación Argentina. (2000, 02 de noviembre). *Ley 25.326, de 2000.* Boletín Oficial de la República Argentina.

El Senado y Cámara de Diputados de la Nación Argentina. (2020, 11 de agosto). *Ley 27553, de 2020.* Boletín Oficial de la República Argentina.

Federfarma Lombardia. (s.f.). *Farmacia Aperta*. Obtenido de Farmacia Aperta: https://www.farmacia-aperta.eu/

ISO/IEC/IEEE. (2023). *ISO-IEC-IEEE 15288-2023 - Systems and Software Engineering — System Life Cycle Processes.* IEEE.

ISO/IEC/IEEE 42010:2022 - Software, s. a.—A. (2022). *ISO/IEC/IEEE 42010:2022 - Software, systems and enterprise — Architecture description, 2nd ed. .* ISO/IEC/IEEE, 2022.

Jalón Arias, E. J., Albarracín Zambrano, L. O., Chillagana Orbes, A. I., & Herrera Chacón, W. A. (2024). Aplicación para determinar la ubicación Geográfica de las farmacias de turno en la Ciudad de Punyo. *Universidad y Sociedad, 16(5)*, 195-205.

Kreimer, N., & Gende , M. (2010). *Una herramienta cartográfica digital basada en XML para la ciudad de La Plata.* Obtenido de https://www.scielo.org.ar/: https://www.scielo.org.ar/scielo.php?script=sci_arttext&pid=S1852-77442010000100005&lang=es

Longley, P. A., Goodchild, M. F., & Rhind, D. W. (2015). *Geographic Information Systems and Science (4ª ed.).* Wiley.

M.A.S. (2023). *Farmacia De Turno Ahora*. Obtenido de Farmacia De Turno Ahora: https://farmaciadeturnoahora.com.ar/

Microsoft. (2025). *Diseño de una aplicación orientada a microservicios*. Obtenido de Microsoft Learn: https://learn.microsoft.com/es-es/dotnet/architecture/microservices/multi-container-microservice-net-applications/microservice-application-design

Rojo, M. I. (2022). *FarMap*. Obtenido de FarMap: https://farmap.com.ar/

Secretaría de Política y Regulación de Salud. (1998, 19 de agosto). *Resolución 192/98.* Buenos Aires, Argentina: Boletín Oficial de la República Argentina.

Sommerville, I. (2021). *SOFTWARE ENGINEERING 10th Ed.* Pearson.

Stair, R., & Reynolds, G. (2018). *Principles of Information Systems 13th.* ISBN 13: 9781305971776: Cengage Learning.

VITAU MEDICAL, S.A.P.I. DE C.V. (s.f.). *Vitau*. Obtenido de Vitau: https://vitau.mx/

ANEXOS

ANEXO I - LEY 27553 – RECETAS ELECTRÓNICAS O DIGITALES

Ley 27.553 – Prescripción y dispensación de recetas electrónicas o digitales
Publicada en el *Boletín Oficial de la República Argentina* el 11 de agosto de 2020.

Artículo 1º. Objeto

La presente ley tiene por objeto:

a) Establecer que la prescripción y dispensación de medicamentos, y toda otra prescripción, puedan ser redactadas y firmadas a través de firmas manuscritas, electrónicas o digitales, en recetas electrónicas o digitales, en todo el territorio nacional;

b) Establecer que puedan utilizarse plataformas de teleasistencia en salud, en todo el territorio nacional, de conformidad con la ley 25.326 de Protección de los Datos Personales y la ley 26.529 de Derechos del Paciente.

Toda prescripción electrónica o digital y plataforma de teleasistencia en salud que reúnan los requisitos técnicos y legales son válidas de acuerdo a la legislación vigente que no se encuentre modificada por la presente ley.

Artículo 2º. Ámbito de aplicación

La presente ley es de aplicación para toda receta o prescripción médica, odontológica o de otros profesionales sanitarios legalmente facultados a prescribir, en los respectivos ámbitos de asistencia sanitaria y atención farmacéutica pública y privada.

Los medicamentos prescriptos en recetas electrónicas o digitales deben ser dispensados en cualquier farmacia del territorio nacional, servicios de farmacia de establecimientos de salud y establecimientos del sector salud habilitados para tal fin, conforme a las disposiciones vigentes.

Artículo 3º. Autoridad de aplicación

La autoridad de aplicación de la presente ley será establecida por el Poder Ejecutivo nacional, coordinando su accionar con las autoridades jurisdiccionales competentes y los organismos con incumbencia en la materia que dichas autoridades determinen, quienes definirán por vía reglamentaria los plazos necesarios para:

- alcanzar la digitalización total en prescripción y dispensación de medicamentos,
- regular el uso de plataformas de teleasistencia en salud.

Artículo 4º. Implementación de sistemas electrónicos

Para la implementación de la presente ley se deben desarrollar y/o adecuar los sistemas electrónicos existentes y regular su implementación para utilizar recetas electrónicas o digitales, y plataformas de teleasistencia en salud, lo cual debe regular el organismo que el Poder Ejecutivo nacional oportunamente establezca y los organismos que cada jurisdicción determine.

Además, dichos organismos son responsables de:

- la fiscalización de los sistemas de recetas electrónicas o digitales,
- la custodia de las bases de datos de asistencia, prescripción, dispensación y archivo,
- establecer los criterios de autorización y control de acceso,
- garantizar el cumplimiento de la ley 25.326 de Protección de Datos Personales y la ley 26.529 de Derechos del Paciente, y demás normativas vigentes.

ANEXO II - DECRETO 345/2024

Decreto 345/2024 – Modificaciones a la Reglamentación de la Ley N° 27.553 (Recetas electrónicas o digitales)

Publicado en el *Boletín Oficial de la República Argentina* el 22 de abril de 2024.

Considerandos relevantes

El decreto tiene por finalidad adecuar la Reglamentación de la Ley 27.553, con el objeto de favorecer la informatización y digitalización de las recetas médicas, órdenes de estudios y prácticas, establecer la receta electrónica y/o digital como medio obligatorio para la prescripción en todo el territorio nacional (en la medida que las jurisdicciones locales adhieran) y promover la interoperabilidad, accesibilidad y calidad del sistema de prescripción digital.

Artículo 1° – Contenido de la receta electrónica o digital

Se sustituye el punto 2 del inciso A) del artículo 1° de la Reglamentación de la Ley N° 27.553, que quedará redactado así:

"El contenido de la receta electrónica y/o digital deberá cumplir con lo establecido por el inciso 7 del artículo 19 de la Ley N° 17.132 y su Reglamentación, favoreciendo la simplificación, accesibilidad, equidad y calidad de la atención sanitaria, conforme a la estructura que determine la Autoridad de Aplicación."

Artículo 2° – Identificación de medicamentos

Se incorpora como punto 7 del inciso A) del artículo 1° de la Reglamentación:

"La Autoridad de Aplicación implementará un instrumento de identificación y referencia para los medicamentos a prescribirse en recetas electrónicas y/o digitales, con el fin de favorecer el uso, acceso e interoperabilidad de estas."

Artículo 3° – Vigencia de implementaciones actuales

Se modifica el inciso C) del artículo 1° de la Reglamentación, estableciendo que:

- Las actuales implementaciones de receta electrónica y/o digital continuarán vigentes si cumplen con los requisitos previstos.
- La receta electrónica y/o digital será el medio obligatorio para prescripción médica, órdenes de estudios y prácticas, en todo el territorio nacional (en la medida que las jurisdicciones locales adhieran).

Artículo 4° – Adecuación tecnológica y legal

Se sustituye el artículo 2° de la Reglamentación para establecer:

"Toda prescripción de receta digital y/o electrónica y las plataformas de teleasistencia deberán adecuarse a los requerimientos legales que regulen su ejercicio…
Las autoridades jurisdiccionales podrán adherir a la utilización de la Licencia Sanitaria Federal con el fin de instrumentar la prescripción de la receta digital o electrónica."

Artículo 5° – Registro de recetarios electrónicos

Se redefine el inciso vi del artículo 3° de la Reglamentación:

"Crear el Registro de Recetarios Electrónicos al cual los responsables de las plataformas y/o sistemas con capacidad técnica para prescribir recetas electrónicas y/o digitales deberán informar el formato y modelo de las recetas a emitir, determinando los datos exigibles."

Artículo 13° – Autoridad de aplicación y procesos

El Ministerio de Salud, como Autoridad de Aplicación, deberá garantizar los procesos que permitan la implementación de la Ley 27.553 y promover la interoperabilidad de las plataformas y mecanismos de acceso a firma digital o electrónica.

Disposición final – Vigencia

Este decreto entró en vigencia a partir del 1° de julio de 2024, estableciendo un cronograma de adecuación progresiva para sistemas, plataformas y jurisdicciones.

ANEXO III - LEY 25326 – PROTECCIÓN DE DATOS PERSONALES

Ley 25.326 de Protección de los Datos Personales

Sancionada el 4 de octubre de 2000 y publicada en el *Boletín Oficial de la República Argentina*.

Artículo 1° – Objeto

La presente ley tiene por objeto la protección integral de los datos personales asentados en archivos, registros, bases de datos u otros medios técnicos de tratamiento de datos, sean públicos o privados, para garantizar el derecho al honor y a la intimidad de las personas, así como también el acceso a la información que sobre las mismas se registre.

Artículo 2° – Definiciones

A los fines de esta ley se entiende por:

- **Datos personales:** Información de cualquier tipo referida a personas físicas o jurídicas determinadas o determinables.
- **Datos sensibles:** Datos que revelan, entre otros, información referente a la salud o a la vida sexual.
- **Tratamiento de datos:** Operaciones que permiten la recolección, conservación, modificación, evaluación, borrado o acceso de datos personales por medios tecnológicos o no.

Artículo 5° – Consentimiento

El tratamiento de datos personales es ilícito cuando el titular no hubiere prestado su consentimiento libre, expreso e informado, el cual debe constar por escrito o por un medio equivalente.

Artículo 9° – Seguridad de los datos

El responsable del tratamiento debe adoptar medidas técnicas y organizativas necesarias para garantizar la seguridad y confidencialidad de los datos personales, evitando su adulteración, pérdida, acceso no autorizado o tratamiento no autorizado.

ANEXO IV - LEY 17565 – EJERCICIO DE LA PROFESIÓN FARMACÉUTICA

Ley 17.565 – Ejercicio de la actividad farmacéutica
Sancionada en 1967. Publicada en el Boletín Oficial de la República Argentina.

La Ley 17.565 regula el ejercicio de la actividad farmacéutica y el expendio de medicamentos en todo el territorio nacional, estableciendo el marco legal para la habilitación, funcionamiento y control de las farmacias.

Artículo 1° – Ámbito de aplicación

El artículo 1° establece que el ejercicio de la actividad farmacéutica y el expendio de medicamentos quedan sometidos a la presente ley y a su reglamentación. Esta disposición determina que toda actividad vinculada a la comercialización y dispensación de medicamentos debe ajustarse al régimen jurídico nacional vigente, constituyendo el marco normativo general que regula la profesión farmacéutica en Argentina.

Disposiciones sobre habilitación y dirección técnica

La ley establece que el expendio de medicamentos sólo puede realizarse en establecimientos habilitados por la autoridad sanitaria competente. Asimismo, dispone que las farmacias deben contar con dirección técnica ejercida por un profesional farmacéutico debidamente habilitado, quien asume la responsabilidad profesional por el funcionamiento del establecimiento. Estas previsiones garantizan que la dispensación de medicamentos se realice bajo supervisión profesional y dentro de un marco regulado por el Estado.

Facultades de la autoridad sanitaria

La norma otorga a la autoridad sanitaria facultades de fiscalización y control sobre los establecimientos farmacéuticos, así como la potestad de dictar normas reglamentarias y complementarias que regulen su funcionamiento. En este sentido, el Estado conserva la competencia para supervisar el cumplimiento de las disposiciones legales y adoptar medidas ante eventuales infracciones.

Las disposiciones precedentemente mencionadas resultan pertinentes para el presente trabajo, en tanto fundamentan que la plataforma propuesta debe operar conforme al marco regulatorio vigente y contemplar exclusivamente farmacias habilitadas por la autoridad competente.

Se incluyen en este anexo únicamente los artículos y disposiciones relevantes para el objeto de la investigación.

ANEXO V - LEY PROVINCIAL 67-Q – EJERCICIO DE LA PROFESIÓN FARMACÉUTICA EN SAN JUAN

Ley 67-Q – Código Sanitario de la Provincia de San Juan
Texto ordenado 2014. Provincia de San Juan.

La Ley 67-Q constituye el Código Sanitario de la Provincia de San Juan y regula el funcionamiento de los establecimientos de salud dentro del ámbito provincial, incluyendo las farmacias y su régimen de turnos.

Artículo 122 – Turnos de farmacias

El artículo 122 establece que las farmacias deberán realizar turnos conforme a la distribución que disponga la autoridad sanitaria en días domingos, feriados y en horarios extraordinarios, a fin de garantizar la atención al público. Asimismo, determina que los horarios de atención, tanto en días hábiles como en turnos, serán fijados y controlados por la autoridad sanitaria pública. Esta disposición asegura la continuidad del servicio farmacéutico como prestación esencial para la comunidad.

Disposiciones sobre habilitación y control sanitario

El Código Sanitario dispone que los establecimientos sanitarios requieren autorización de la autoridad provincial competente para su funcionamiento y que el expendio de medicamentos debe efectuarse en farmacias habilitadas. Asimismo, reconoce facultades de fiscalización y control a la autoridad sanitaria provincial, quien puede supervisar el cumplimiento de la normativa vigente.