Desarrollo de la Aplicación Móvil ReservaFácil

Fabián Guillermo Salazar Sarmiento

Facultad de Ingeniería, Universidad UNIMINUTO

NRC: 40-61831: Desarrollo de Software Seguro

Edwin Albeiro Ramos Villamil

31 de enero de 2026

Contents

Capítulo 1: Introducción 3

1.1. Contexto 3

1.2. Motivaciones 4

1.3. Objetivo General 4

1.4. Objetivos Específicos 4

Capítulo 2: Marco Técnico y Metodológico 5

2.1. Metodología de Desarrollo: Scrum 5

2.2. Arquitectura y Stack Tecnológico 5

2.3. Herramientas de Desarrollo y Costo de Software 6

2.4. Requisitos y Costo de Hardware 6

Capítulo 3: Análisis del Sistema 7

3.1. Definición del Sistema ReservaFácil 7

3.1.1 Análisis de Seguridad del Sistema 7

3.2. Historias de Usuario (Product Backlog) 9

3.3. Casos de Uso 14

3.4: Descomposición de Tareas para Cronograma 15

Capítulo 4: Propuesta Técnica y Desarrollo 21

Capítulo 4.1: Propuesta Técnica y Desarrollo 21

4.1.1 Consideraciones de Seguridad en la Implementación 23

4.2. Estimación (Planning Poker Simulado) 26

4.3. Sprints 27

Capítulo 5: Planificación y Presupuesto 29

5.2. Matriz de Riesgos 29

5.3. Presupuesto Resumido 30

5.4 Consideraciones de Seguridad en el Despliegue y Operación 30

6. Pruebas de Aceptación del Usuario (UAT) 32

6.1 Pruebas de Seguridad del Sistema 33

Capítulo 7: Reflexión 35

7.1 Seguridad y Mejora Continua del Sistema 35

Referencias 36

Capítulo 1: Introducción

1.1. Contexto

El presente proyecto se enfoca en la aplicación de la metodología ágil Scrum para el desarrollo de un sistema de información móvil para la gestión de reservas, diseñado para un pequeño restaurante. Un sistema de este tipo es fundamental para organizar el flujo de clientes, optimizar la ocupación de mesas y mejorar la experiencia del comensal al ofrecer un canal digital para garantizar su visita.

1.2. Motivaciones

La motivación principal es aplicar los conocimientos teóricos sobre metodologías ágiles, arquitecturas de software como lo es MVVM en un caso práctico que abarque el ciclo de vida completo de un proyecto de software móvil. La elección de un sistema de reservas responde a una necesidad de negocio clara y fácil de entender, pero con la suficiente interacción de datos para justificar una gestión de proyecto iterativa e incremental.

1.3. Objetivo General

Desarrollar un prototipo funcional de una aplicación móvil para la gestión de reservas de un pequeño restaurante, aplicando la metodología Scrum y la arquitectura MVVM en todas las fases del ciclo de vida del desarrollo.

1.4. Objetivos Específicos

- Aplicar una metodología ágil: Utilizar los roles, artefactos y eventos de Scrum para gestionar el proyecto de manera flexible y adaptativa.
- Desarrollar una aplicación utilizable: Diseñar una interfaz de usuario móvil intuitiva y con diseño responsivo que se adapte a diferentes tamaños de pantalla.
- Asegurar una arquitectura modular: Implementar el patrón de diseño MVVM para separar la lógica de la presentación, facilitando el mantenimiento y la escalabilidad futura.
- Garantizar la persistencia de la información: Almacenar de forma segura y consistente todos los datos en una base de datos MySQL y utilizar persistencia local en el dispositivo para recordar la sesión del usuario.

Capítulo 2: Marco Técnico y Metodológico

2.1. Metodología de Desarrollo: Scrum

Para la gestión del proyecto se ha elegido Scrum, un marco de trabajo ágil que divide el desarrollo en ciclos cortos de tiempo fijo llamados Sprints. Al final de cada Sprint, se entrega un incremento de producto funcional, lo que permite una constante inspección y adaptación a los cambios.

2.2. Arquitectura y Stack Tecnológico

Arquitectura General (Cliente-Servidor): Se implementará un modelo Cliente-Servidor. El cliente será la aplicación móvil.

React Native que se comunica vía peticiones HTTP a un servidor (backend) que alojará la lógica de negocio y la conexión a la base de datos.

Arquitectura de la Aplicación: La aplicación se estructurará bajo el patrón Modelo-Vista-Modelo de Vista (MVVM), que organiza el código en tres capas:

- Vista: La interfaz de usuario creada con componentes de React Native. Es responsable de mostrar los datos y capturar las interacciones del usuario.
- Modelo de Vista: Contiene la lógica de presentación y el estado de la Vista. Actúa como puente entre la Vista y el Modelo, exponiendo datos y comandos.
- Modelo: Representa los datos y la lógica de negocio. Incluye las interacciones con la API y la base de datos.

2.3. Herramientas de Desarrollo y Costo de Software

Se ha optado por un stack tecnológico de código abierto para minimizar los costos de licenciamiento.

Elemento	Descripción	Costo Estimado (COP)
React Native / Expo	Framework para el desarrollo de la aplicación móvil (frontend).	$0 (Open Source)
Node.js / Express	Entorno y framework para el servidor (backend/API).	$0 (Open Source)
MySQL	Sistema Gestor de Base de Datos.	$0 (Open Source)
Visual Studio Code	Entorno de Desarrollo Integrado (IDE).	$0 (Gratuito)
Figma	Herramienta para el diseño de Mockups e interfaces.	$0 (Plan Gratuito)
Git / GitHub	Sistema de control de versiones y repositorio.	$0 (Plan Gratuito)
Total Costo de Software		$0

2.4. Requisitos y Costo de Hardware

Se asume el uso de equipos personales. El costo imputable se calcula basado en la depreciación durante los 1.25 meses del proyecto (5 semanas).

Elemento	Costo Estimado (COP)	% Uso en Proyecto	Dedicación Periodo (Meses)	Depreciación (Meses)	Costo Imputable (COP)
Laptop de Desarrollo	$3.500.000	100%	1.25	36	$121.528
Teléfono Móvil (Pruebas)	$1.200.000	50%	1.25	24	$31.250
Total Costo de Hardware					$152.778

Capítulo 3: Análisis del Sistema

3.1. Definición del Sistema ReservaFácil

ReservaFácil es una aplicación móvil destinada a la gestión de reservas de un pequeño restaurante. Su objetivo es permitir a los clientes registrarse, ver la disponibilidad y crear reservas de manera sencilla, mientras que el restaurante puede llevar un control centralizado de las mismas.

3.1.1 Análisis de Seguridad del Sistema

En la fase de análisis del sistema ReservaFácil se identifican los activos críticos de información, los actores que interactúan con el sistema y las principales amenazas de seguridad, con el objetivo de definir los requisitos de protección que deberán ser considerados.

Activos de información a proteger

- **Base de datos del sistema:**
    I. Información personal de los usuarios (nombre, correo electrónico, teléfono).
    II. Credenciales de acceso almacenadas en la base de datos.
    III. Información de las reservas (fecha, hora, número de personas y estado).
- **API del servidor:**
    I. Endpoints de autenticación, gestión de usuarios y reservas.
- **Sesiones de usuario:**
    I. Tokens de autenticación utilizados para mantener la sesión activa.

Actores del sistema

I. Usuario Cliente: Accede únicamente a sus datos personales y a sus propias reservas.

II. Usuario Administrador: Accede a la gestión completa de usuarios y reservas.

III. Atacante externo: Persona sin autorización que intenta vulnerar el sistema para acceder, modificar o eliminar información.

Amenazas de seguridad identificadas

I. Acceso no autorizado a la base de datos, mediante explotación de vulnerabilidades en la API.

II. Inyección SQL, orientada a obtener, modificar o eliminar información sensible.

III. Fuerza bruta sobre credenciales, intentando adivinar contraseñas de usuarios.

IV. Escalada de privilegios, donde un usuario cliente intenta ejecutar acciones exclusivas del administrador.

V. Suplantación de identidad, utilizando tokens robados o sesiones mal gestionadas.

Riesgos asociados

I. Pérdida de confidencialidad de los datos personales de los clientes.

II. Alteración o eliminación no autorizada de reservas.

III. Uso indebido del panel administrativo del sistema.

IV. Afectación a la integridad y confiabilidad del sistema de reservas.

Requisitos de seguridad derivados del análisis

I. El sistema debe proteger la base de datos contra accesos no autorizados.

II. Todas las operaciones deben ejecutarse únicamente a través de la API, evitando el acceso directo a los datos.

III. El acceso a funcionalidades debe estar estrictamente controlado por roles de usuario.

IV. Las credenciales y sesiones deben manejarse de forma segura para evitar suplantación de identidad.

3.2. Historias de Usuario (Product Backlog)

El Product Backlog se ha refinado para diferenciar las funcionalidades por rol de usuario (Cliente y Administrador), cubriendo todos los puntos solicitados.

ID	Nombre	Rol	Prioridad	Descripción	Criterios de Validación
HU01	Autenticación de Clientes	Cliente	Alta	Como cliente, quiero poder registrarme e iniciar sesión para acceder a las funciones de la app.	- El sistema permite el autoregistro con nombre, email y contraseña. - Un usuario registrado puede iniciar sesión. - El sistema muestra un error si las credenciales son incorrectas.
HU02	Gestión de Perfil Propio	Cliente	Media	Como cliente, quiero poder ver y actualizar la información de mi perfil para mantener mis datos al día.	- El usuario puede ver sus propios datos (nombre, email, teléfono).

					- El usuario puede modificar sus datos y guardarlos.
HU03	Gestión de Reservas por Cliente	Cliente	Alta	Como cliente, quiero poder crear, ver, actualizar y cancelar mis propias reservas para organizar mis visitas.	- El cliente puede registrar una nueva reserva. - El cliente puede ver una lista de sus reservas. - El cliente puede modificar (actualizar) una reserva existente. - El cliente puede eliminar (cancelar) una reserva.
HU04	Gestión de Usuarios por Admin	Admin	Alta	Como administrador, quiero poder crear, ver, buscar, actualizar y	- El admin puede registrar un nuevo usuario.

				eliminar las cuentas de usuario para gestionar la base de datos de clientes.	- El admin puede ver la lista de todos los usuarios. - El admin puede consultar el detalle de un usuario específico. - El admin puede modificar los datos de cualquier usuario. - El admin puede eliminar un usuario del sistema.
HU05	Gestión de Reservas por Admin	Admin	Alta	Como administrador, quiero poder consultar, buscar, actualizar y eliminar cualquier	- El admin puede ver todas las reservas (ej.

				reserva para gestionar la ocupación del restaurante.	filtrando por fecha). - El admin puede consultar el detalle de una reserva específica. - El admin puede modificar cualquier reserva. - El admin puede eliminar cualquier reserva.
HU06	Persistencia de Sesión	Cliente / Admin	Media	Como usuario, quiero que la app recuerde mi sesión para no tener que iniciarla cada vez que la abro.	- Al cerrar y reabrir la aplicación, la sesión del usuario permanece activa.

3.3. Casos de Uso

Basado en las Historias de Usuario, se definen los siguientes casos de uso:

Módulo Usuarios

CU01 - Autoregistro de Cliente: Un visitante abre la app, selecciona la opción Registrarse, completa el formulario y crea su cuenta de cliente.

CU02 - Inicio de Sesión: Un usuario (Cliente o Administrador) ingresa su correo y contraseña para acceder a las funcionalidades correspondientes a su rol.

CU03 - Registro de Usuario por Administrador: El administrador, desde su panel, completa un formulario para crear una nueva cuenta de usuario en el sistema.

CU04 - Consulta de todos los usuarios: El administrador accede a una sección que muestra una lista con todos los usuarios registrados.

CU05 - Consulta de un usuario: El administrador selecciona un usuario de la lista (o lo busca) para ver su información detallada. Un cliente solo puede consultar su propio perfil.

CU06 - Actualización de usuario: El administrador modifica los datos de un usuario seleccionado. Un cliente solo puede modificar los datos de su propio perfil.

CU07 - Eliminación de usuario: El administrador elimina la cuenta de un usuario del sistema.

Módulo Reservas

CU08 Registro de Reserva: Un cliente autenticado selecciona fecha, hora y número de personas para crear una nueva reserva.

CU09 - Consulta de todas las reservas: El administrador accede a un panel donde puede ver todas las reservas del restaurante, con opción de filtrar por fecha.

CU10 - Consulta de una reserva: Un cliente consulta los detalles de una de sus reservas. Un administrador puede consultar los detalles de cualquier reserva.

CU11 - Actualización de una reserva: Un cliente modifica la hora o el número de personas de una de sus reservas. Un administrador puede modificar cualquier reserva.

CU12 - Eliminación de una reserva: Un cliente cancela una de sus reservas. Un administrador puede cancelar cualquier reserva del sistema.

3.4: Descomposición de Tareas para Cronograma

A continuación, se presenta el desglose completo de tareas técnicas necesarias para el desarrollo del proyecto ReservaFácil. Las tareas están asociadas a las Historias de Usuario y distribuidas a lo largo de los Sprints planificados.

Fase 0: Preparación del Entorno (Tareas Previas al Sprint 1)

ID Tarea	Historia Asociada	Descripción de la Tarea
T01	Todas	Instalación de Software: Instalar Node.js, Visual Studio Code, MySQL, Expo CLI, Git y Figma.
T02	Todas	Configuración del Entorno: Configurar las extensiones de VS Code y crear el usuario/base de datos en MySQL.

T03	Todas	Inicialización de Proyectos: Crear el repositorio en GitHub y clonarlo. Iniciar los proyectos de React Native (frontend) y Node.js (backend).

Sprint 1 (02/02/2026 - 06/02/2026): Configuración y Cuentas de Usuario

Objetivo del Sprint: Un usuario puede registrarse e iniciar sesión. El administrador puede ver la lista de usuarios.

ID Tarea	Historia Asociada	Descripción de la Tarea
T04	HU01, HU04	Mockups: Diseñar en Figma las pantallas de Inicio de Sesión, Registro de Cliente y Panel de Administrador - Lista de Usuarios.
T05	HU01, HU04	Base de Datos: Crear la tabla Usuarios en MySQL con los campos: id, nombre, email, password, telefono y rol (cliente/admin).
T06	HU01	API: Desarrollar el endpoint para el autoregistro de clientes (POST /api/users/register).
T07	HU01	API: Desarrollar el endpoint para iniciar sesión (POST /api/users/login) que devuelva un Token (JWT).
T08	HU04	API: Desarrollar el endpoint para que el admin consulte todos los usuarios (GET /api/users).

T09	HU01	Interfaces: Construir el componente de la pantalla de Inicio de Sesión en React Native.
T10	HU01	Interfaces: Construir el componente de la pantalla de Registro de Cliente en React Native.
T11	HU01	Interfaces: Conectar las pantallas de Inicio de Sesión y Registro con sus respectivos endpoints de la API.
T12	HU04	Interfaces: Construir la pantalla Lista de Usuarios para el rol de Administrador.

Sprint 2 (09/02/2026 - 13/02/2026) Creación y Visualización de Reservas

Objetivo del Sprint: Un cliente puede crear una reserva y un administrador puede ver todas las reservas del sistema.

ID Tarea	Historia Asociada	Descripción de la Tarea
T13	HU03, HU05	Mockups: Diseñar en Figma las pantallas de Crear Nueva Reserva y Panel de Administrador - Lista de Reservas.
T14	HU03, HU05	Base de Datos: Crear la tabla Reservas en MySQL y establecer la relación con la tabla Usuarios.
T15	HU03	API: Desarrollar el endpoint para registrar una nueva reserva (POST /api/reservations/create), asociándola al cliente autenticado.

T16	HU05	API: Desarrollar el endpoint para que el admin consulte todas las reservas (GET /api/reservations), permitiendo filtrar por fecha.
T17	HU03	Interfaces: Construir el componente de la pantalla Crear Reserva, incluyendo un selector de fecha, hora y número de personas.
T18	HU03	Interfaces: Conectar el formulario de creación de reserva con el endpoint de la API.
T19	HU05	Interfaces: Construir la pantalla Lista de Reservas para el rol de Administrador.

Sprint 3 (16/02/2026 - 20/02/2026): Gestión Completa (Actualizar y Eliminar)

Objetivo del Sprint: El cliente puede gestionar su perfil y sus reservas. El administrador puede gestionar cualquier usuario y reserva.

ID Tarea	Historia Asociada	Descripción de la Tarea
T20	HU02, HU03	Mockups: Diseñar las pantallas de Mis Reservas (lista para cliente) y Mi Perfil.
T21	HU02, HU04	API: Desarrollar los endpoints para actualizar (PUT /api/users/:id) y eliminar (DELETE /api/users/:id) usuarios (con lógica de roles).

T22	HU03, HU05	API: Desarrollar los endpoints para actualizar (PUT /api/reservations/:id) y eliminar (DELETE /api/reservations/:id) reservas.
T23	HU02, HU04	API: Desarrollar endpoint para consultar un usuario específico (GET /api/users/:id).
T24	HU03	Interfaces: Construir la pantalla Mis Reservas para que el cliente vea y pueda cancelar/modificar sus reservas.
T25	HU02	Interfaces: Construir la pantalla Mi Perfil para que el cliente pueda consultar y actualizar sus datos.
T26	HU04	Interfaces: Implementar la funcionalidad de actualizar y eliminar en el panel de Lista de Usuarios del administrador.

Sprint 4 (23/02/2026 - 27/02/2026): Persistencia y Refinamiento de Diseño

Objetivo del Sprint: Implementar la persistencia de sesión y asegurar que el diseño sea responsivo.

ID Tarea	Historia Asociada	Descripción de la Tarea
T27	HU06	Interfaces: Implementar persistencia local (AsyncStorage) para guardar el token de sesión del usuario.

T28	HU06	Interfaces: Añadir lógica al inicio de la app para verificar si existe un token válido y redirigir automáticamente al usuario.
T29	HU06	Interfaces: Implementar la funcionalidad de Cerrar Sesión, que elimina el token del almacenamiento local.
T30	Todas	Interfaces: Realizar una revisión completa del diseño responsivo en todas las pantallas para asegurar una buena visualización en distintos dispositivos.

Sprint 5 (02/03/2026 - 06/03/2026): Pruebas Finales y Entrega

Objetivo del Sprint: Corregir errores, validar el cumplimiento de todos los requisitos y preparar el paquete final.

ID Tarea	Historia Asociada	Descripción de la Tarea
T31	Todas	Pruebas: Ejecutar todas las Pruebas de Aceptación del Usuario (UAT) para validar cada HU.
T32	Todas	Pruebas: Realizar pruebas de extremo a extremo (E2E) de los flujos principales (ej: registro, login, reserva, cancelación).
T33	Todas	Corrección: Identificar y solucionar los bugs encontrados durante la fase de pruebas.

T34	Todas	Documentación: Finalizar el documento README.md del proyecto con las instrucciones de instalación y despliegue.
T35	Todas	Entrega: Preparar el paquete final del proyecto para su entrega.

Capítulo 4: Propuesta Técnica y Desarrollo

Capítulo 4.1: Propuesta Técnica y Desarrollo

En la fase de diseño del sistema ReservaFácil se definen los principios y mecanismos de seguridad que orientarán la implementación del sistema, asegurando la protección de la información y el control de acceso a las funcionalidades, sin comprometer la usabilidad de la aplicación.

Diseño de seguridad en la arquitectura Cliente-Servidor

El sistema adopta una arquitectura Cliente-Servidor, lo cual permite centralizar la lógica de negocio y los controles de seguridad en el servidor. Bajo este enfoque:

I. La aplicación móvil actúa únicamente como consumidor de la API.

II. La base de datos no es accesible directamente desde el cliente.

III. Todas las validaciones críticas y decisiones de autorización se realizan en el servidor.

Este diseño reduce el riesgo de manipulación de datos desde el cliente y protege la integridad de la información almacenada.

Diseño de seguridad por capas

El sistema se diseña bajo un enfoque de defensa en profundidad, aplicando controles de seguridad en múltiples capas:

- Capa de presentación (Aplicación móvil): control de acceso a pantallas según el estado de autenticación del usuario.
- Capa de lógica de negocio (API): validación de permisos y reglas de negocio.
- Capa de datos (Base de datos): restricción de acceso únicamente desde el servidor.

Este enfoque permite mitigar riesgos incluso si una de las capas es comprometida.

Control de acceso basado en roles

Desde el diseño del sistema se define un modelo de control de acceso basado en roles diferenciando claramente los permisos del usuario cliente y del usuario administrador. Cada funcionalidad del sistema se asocia a un rol específico, evitando el acceso indebido a operaciones sensibles.

Diseño seguro de la gestión de sesiones

El diseño contempla el uso de mecanismos de autenticación basados en tokens para la gestión de sesiones. La sesión del usuario se mantiene activa únicamente mientras el token sea válido, evitando el uso de sesiones persistentes inseguras.

Protección de datos sensibles

En el diseño del sistema se establece que los datos sensibles, como las credenciales de usuario, no deben ser almacenados ni transmitidos en texto plano. Asimismo, se define la necesidad de proteger la comunicación entre el cliente y el servidor para evitar la interceptación de información.

Este diseño de seguridad garantiza que los aspectos críticos de protección de la información estén contemplados antes de iniciar la fase de implementación.

4.1.1 Consideraciones de Seguridad en la Implementación

Durante la fase de implementación del sistema ReservaFácil, se deberán aplicar controles de seguridad alineados con el análisis de riesgos y el diseño arquitectónico definido, con el objetivo de proteger la información y prevenir vulnerabilidades comunes en aplicaciones móviles y servicios web.

Autenticación segura de usuarios

La implementación deberá garantizar que todos los usuarios se autentiquen antes de acceder a las funcionalidades del sistema. Las credenciales de acceso deberán gestionarse de forma segura, evitando el almacenamiento o transmisión de contraseñas en texto plano.

Autorización y validación de roles

Todas las funcionalidades del sistema deberán validar el rol del usuario autenticado antes de ejecutar cualquier operación. Las acciones administrativas estarán restringidas exclusivamente a usuarios con rol administrador, evitando la escalada de privilegios.

Validación de entradas y protección contra ataques

El sistema deberá validar todos los datos recibidos desde la aplicación móvil antes de ser procesados o almacenados. Esta validación permitirá reducir el riesgo de ataques como inyección SQL, manipulación de parámetros o envío de datos maliciosos.

Seguridad en el acceso a la base de datos

La implementación deberá asegurar que el acceso a la base de datos se realice únicamente a través de la capa de servidor, aplicando principios de mínimo privilegio y evitando consultas directas desde el cliente.

Gestión segura de sesiones

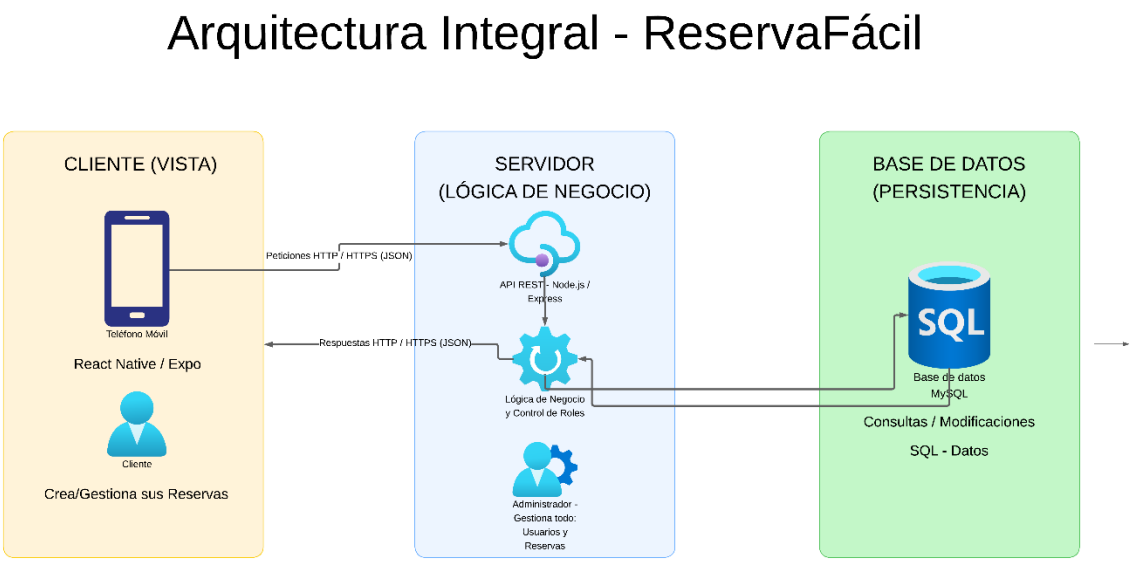
La sesión del usuario deberá gestionarse mediante mecanismos que permitan identificar de forma confiable al usuario autenticado y controlar la vigencia de la sesión, reduciendo el riesgo de suplantación de identidad o reutilización de sesiones.

Manejo seguro de errores

Los mensajes de error generados por el sistema no deberán exponer información sensible sobre la estructura interna del sistema, la base de datos o la lógica de negocio, limitándose a mensajes controlados y comprensibles para el usuario.

- **Diagrama de Arquitectura General (Cliente-Servidor)**

La arquitectura sigue un flujo claro: la aplicación móvil, desarrollada en React Native, interactúa con el usuario y envía peticiones HTTP a la API REST. Esta API, construida con Node.js y Express, procesa toda la lógica de negocio (validaciones, permisos por rol) y se comunica con la base de datos MySQL para almacenar y recuperar la información de manera persistente.



- **Diagrama de Arquitectura de la Aplicación (MVVM)**

El patrón MVVM organiza el código de la aplicación móvil para facilitar su mantenimiento y escalabilidad. La Vista captura las acciones del usuario y las comunica al ViewModel. Este actúa como un intermediario que solicita los datos al Modelo. El ViewModel luego expone los datos a la Vista, que se actualiza automáticamente para reflejar cualquier cambio de estado.

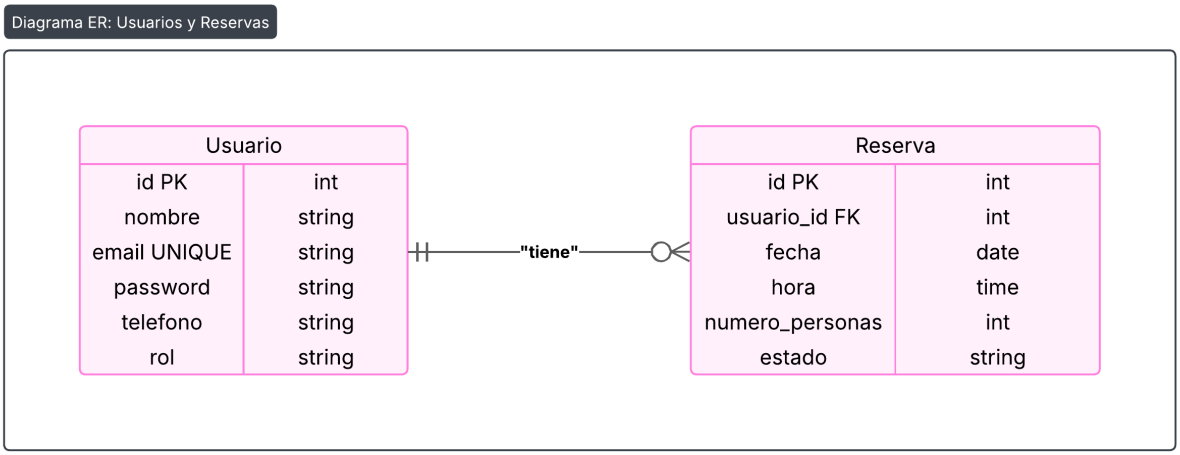
- **Diagrama Entidad-Relación (ERD) de la Base de Datos (MySQL)**

El diseño de la base de datos es fundamental para soportar los roles de usuario.

  - Tabla Usuarios: id (PK), nombre (VARCHAR), email (VARCHAR, UNIQUE), password (VARCHAR), telefono (VARCHAR), rol (VARCHAR).

El campo rol es clave, ya que almacenará si el usuario es un 'cliente' o un 'administrador'.

- Tabla Reservas: id (PK), usuario_id (FK a Usuarios), fecha (DATE), hora (TIME), numero_personas (INT), estado (VARCHAR).



4.2. Estimación (Planning Poker Simulado)

Con las Historias de Usuario ya definidas, se estima el esfuerzo necesario para cada una.

ID Historia de Usuario	Estimación (horas)	Prioridad
HU01: Autenticación de Clientes	14	Alta
HU02: Gestión de Perfil Propio	8	Media
HU03: Gestión de Reservas por Cliente	18	Alta
HU04: Gestión de Usuarios por Admin	16	Alta

HU05: Gestión de Reservas por Admin	12	Alta
HU06: Persistencia de Sesión	6	Media
Total Estimado	74 horas	

4.3. Sprints

El proyecto se ejecutará en **5 Sprints semanales**, abordando las tareas técnicas previamente desglosadas.

- Sprint 1 (02/02/2026 - 06/02/2026): Cimientos y Autenticación
    i. Objetivo: Implementar el sistema de autenticación para clientes y la consulta de usuarios para el administrador.
    ii. Historias: HU01, HU04 (parcialmente: consulta).
    iii. Tareas Clave: T04 a T12 (Diseño inicial, creación de BD, endpoints de registro/login, y construcción de las interfaces correspondientes).
- Sprint 2 (09/02/2026 - 13/02/2026): Funcionalidad Central de Reservas
    i. Objetivo: Permitir que un cliente cree una reserva y que el administrador pueda visualizarlas todas.
    ii. Historias: HU03 (parcialmente: registro), HU05 (parcialmente: consulta).
    iii. Tareas Clave: T13 a T19 (Diseño de pantallas de reserva, creación de tabla Reservas, endpoints para crear/consultar reservas, e interfaces).
- Sprint 3 (16/02/2026 - 20/02/2026): Gestión Completa (CRUD)

   i. Objetivo: Implementar todas las funciones de actualización y eliminación para ambos roles.

   ii. Historias: HU02, HU03 (resto), HU04 (resto), HU05 (resto).

   iii. Tareas Clave: T20 a T26 (Diseño de perfiles, desarrollo de endpoints de actualización/eliminación para usuarios y reservas, e integración en las interfaces).

- Sprint 4 (23/02/2026 - 27/02/2026): Usabilidad y Pulido

   i. Objetivo: Mejorar la experiencia del usuario con la persistencia de sesión y refinar el diseño.

   ii. Historias: HU06.

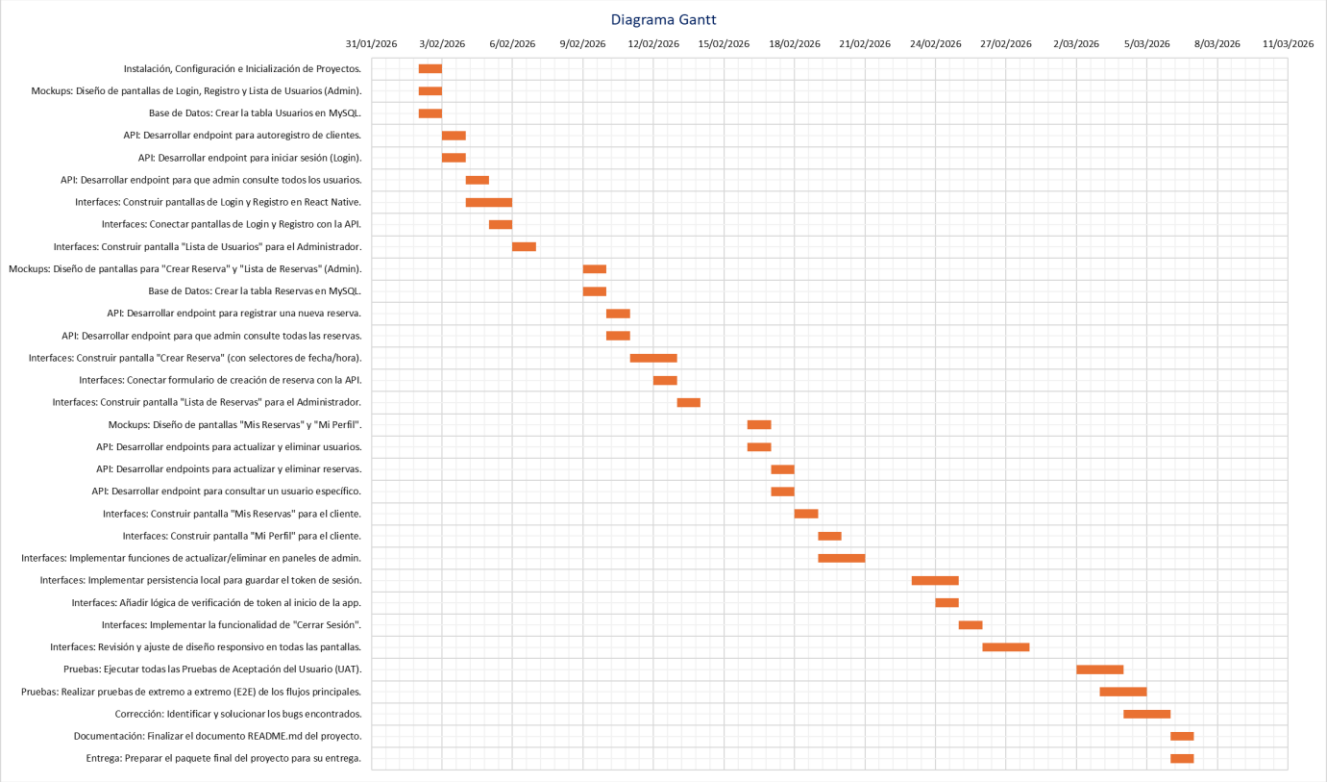
   iii. Tareas Clave: T27 a T30 (Implementación de AsyncStorage para el token, lógica de auto-login, botón de logout y revisión de diseño responsivo).

- Sprint 5 (02/03/2026 - 06/03/2026): Pruebas Finales y Entrega

   i. Objetivo: Asegurar la calidad del producto, corregir errores y preparar la entrega final.

   ii. Tareas Clave: T31 a T35 (Ejecución de todas las UAT, pruebas de flujos completos, corrección de bugs.

Capítulo 5: Planificación y Presupuesto



5.2. Matriz de Riesgos

ID	Riesgo	Probabilidad (1-5)	Impacto (1-5)	Estrategia de Mitigación
R01	Dificultades con la configuración de React Native/Node.js	3	4	Seguir detalladamente el tutorial base. Realizar una prueba de concepto en el primer día del Sprint 1.
R02	Errores en la estimación de las tareas	4	3	Dejar un margen de tiempo en cada Sprint. Re-estimar tareas al inicio de cada Sprint.

R03	Fallos en el equipo de desarrollo	2	5	Mantener copias de seguridad del código actualizadas en GitHub.

5.3. Presupuesto Resumido

Concepto	Costo (COP)
Personal (5 Sprints x 25h x $30.000/h)	$3.750.000
Hardware	$152.778
Software	$0
Subtotal	$3.902.778
Costos Indirectos (10%)	$390.278
Total Estimado del Proyecto	$4.293.056

5.4 Consideraciones de Seguridad en el Despliegue y Operación

Una vez finalizada la fase de desarrollo, el sistema ReservaFácil deberá desplegarse y operar bajo condiciones que garanticen la seguridad de la información y la estabilidad del servicio. Por ello, desde la planeación del proyecto se contemplan las siguientes consideraciones de seguridad.

Protección del entorno de servidor

El sistema deberá ejecutarse en un entorno controlado, restringiendo el acceso al servidor únicamente al personal autorizado. La configuración del entorno de ejecución deberá minimizar la exposición de servicios innecesarios que puedan representar un riesgo de seguridad.

Gestión segura de configuraciones

Las configuraciones sensibles del sistema, como credenciales de acceso a la base de datos o claves de autenticación, no deberán almacenarse directamente en el código fuente. Estas deberán gestionarse de forma separada para reducir el riesgo de exposición.

Seguridad en la comunicación

Durante la operación del sistema, la comunicación entre la aplicación móvil y el servidor deberá protegerse para evitar la interceptación o modificación de la información transmitida, especialmente en operaciones relacionadas con autenticación y gestión de reservas.

Registro y monitoreo de eventos

El sistema deberá contemplar mecanismos de registro de eventos relevantes, como intentos de acceso fallidos o acciones administrativas, con el fin de facilitar la detección de comportamientos anómalos y apoyar tareas de auditoría.

Respaldo y continuidad del servicio

Desde la planeación del despliegue se establece la necesidad de contar con mecanismos de respaldo periódico de la información, que permitan la recuperación de datos ante fallos del sistema o incidentes de seguridad.

Estas consideraciones permiten que el sistema mantenga un nivel adecuado de seguridad durante su operación y reduzca los riesgos asociados a su uso en un entorno real.

6. Pruebas de Aceptación del Usuario (UAT)

ID HU	Caso de Prueba	Pasos a Realizar	Resultado Esperado
HU01	UAT-01: Autenticación de Cliente	Un nuevo usuario se registra y luego inicia sesión con sus credenciales.	La cuenta se crea exitosamente y el login redirige a la pantalla principal del cliente.
HU02	UAT-02: Actualizar Perfil Propio	Un cliente entra a su perfil, cambia su número de teléfono y guarda los cambios.	El número de teléfono se actualiza correctamente y se muestra en el perfil.
HU03	UAT-03: CRUD de Reservas (Cliente)	Un cliente crea una reserva, luego la modifica (cambia la hora) y finalmente la cancela.	La reserva aparece, se actualiza y se elimina correctamente de su lista "Mis Reservas".
HU04	UAT-04: CRUD de Usuarios (Admin)	Un administrador crea un usuario, lo busca en la lista, actualiza su nombre y luego lo elimina.	Todas las operaciones se reflejan correctamente en la lista de usuarios del panel de administración.

HU05	UAT-05: Gestión de Reservas (Admin)	Un administrador visualiza todas las reservas para una fecha específica y cancela una de ellas.	La lista se filtra correctamente y la reserva seleccionada es eliminada del sistema.
HU06	UAT-06: Sesión Persistente	Un usuario con sesión iniciada cierra por completo la aplicación y la vuelve a abrir.	La aplicación carga directamente en la pantalla principal del usuario, sin solicitar credenciales de nuevo.

6.1 Pruebas de Seguridad del Sistema

Además de las pruebas funcionales de aceptación del usuario, el sistema ReservaFácil contempla la ejecución de pruebas de seguridad orientadas a validar la correcta aplicación de los controles definidos en las fases de análisis, diseño e implementación.

Estas pruebas permiten identificar comportamientos inseguros del sistema ante intentos de acceso no autorizado, manipulación de datos o uso indebido de las funcionalidades.

Objetivos de las pruebas de seguridad

- Verificar que solo los usuarios autenticados puedan acceder al sistema.
- Validar que las funcionalidades estén correctamente restringidas según el rol del usuario.
- Comprobar que el sistema maneje adecuadamente entradas inválidas o maliciosas.
- Confirmar que la información sensible no sea expuesta a usuarios no autorizados.

ID	Caso de Prueba	Descripción	Resultado Esperado
TS-01	Acceso sin autenticación	Intentar acceder a una funcionalidad sin haber iniciado sesión	El sistema deniega el acceso y solicita autenticación
TS-02	Acceso con rol incorrecto	Un usuario cliente intenta acceder a funciones de administrador	El sistema bloquea la acción y muestra un mensaje de error
TS-03	Manipulación de parámetros	Modificar manualmente el identificador de una reserva	El sistema impide el acceso a datos que no pertenecen al usuario
TS-04	Entrada de datos inválidos	Enviar datos incorrectos o incompletos en formularios	El sistema valida y rechaza la información

TS-05	Sesión inválida o expirada	Intentar usar el sistema con una sesión no válida	El sistema solicita nuevamente autenticación

Capítulo 7: Reflexión

La metodología Scrum es ideal para este proyecto, ya que permite construir y validar las funcionalidades más importantes (como el registro y la creación de reservas) en las primeras semanas. Este enfoque iterativo reduce el riesgo de encontrar problemas graves al final del desarrollo. La flexibilidad de Scrum permitirá ajustar las prioridades si surgen nuevas ideas o dificultades técnicas, asegurando que al final del cronograma se entregue un producto valioso y funcional.

7.1 Seguridad y Mejora Continua del Sistema

La seguridad del sistema ReservaFácil no se concibe como una actividad puntual, sino como un proceso continuo que acompaña todo el ciclo de vida del software. Una vez el sistema entra en operación, es necesario mantener y mejorar sus mecanismos de protección frente a nuevas amenazas y cambios en el entorno tecnológico.

Mantenimiento de la seguridad

Durante la fase de mantenimiento, se deberán realizar revisiones periódicas del sistema con el fin de identificar posibles vulnerabilidades, corregir fallos de seguridad y actualizar componentes que puedan quedar obsoletos. Estas actividades contribuyen a preservar la confidencialidad, integridad y disponibilidad de la información.

Gestión de incidentes de seguridad

El sistema deberá contar con procedimientos definidos para la atención de incidentes de seguridad, tales como accesos no autorizados o fallos en la protección de datos. La identificación temprana y la respuesta oportuna permiten minimizar el impacto de este tipo de eventos.

Mejora continua basada en Scrum

La metodología Scrum facilita la incorporación de mejoras de seguridad de forma iterativa. Los hallazgos derivados de las pruebas, el monitoreo del sistema o la retroalimentación de los usuarios pueden convertirse en nuevas historias de usuario o tareas técnicas, priorizadas en el Product Backlog para futuros Sprints.

Concientización y buenas prácticas

Finalmente, se resalta la importancia de mantener buenas prácticas de desarrollo seguro y una cultura de seguridad en el equipo de trabajo, asegurando que las futuras modificaciones al sistema mantengan los estándares de protección definidos.

Referencias

- Pérez A., O. A. (2011). Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP - SCRUM. *INVENTUM*, *6*(10), 64-78. https://doi.org/10.26620/uniminuto.inventum.6.10.2011.64-78
- Monte Galiano, J. (2016). *Implantar scrum con éxito:* (ed.). Editorial UOC. https://elibro.net/es/lc/uniminuto/titulos/58575
- Heras del Dedo, R. D. L. & Álvarez García, A. (2017). *Métodos ágiles: Scrum, Kanban, Lean:* (ed.). Difusora Larousse - Anaya Multimedia. https://elibro.net/es/lc/uniminuto/titulos/122933

- Amaro Soriano, J. E. (2019). *Android: programación de dispositivos móviles a través de ejemplos:* (1 ed.). Marcombo. https://elibro.net/es/lc/uniminuto/titulos/281429
- Domínguez Mateos, F. & Santacruz Valencia, L. P. (2015). *Programación multimedia y dispositivos móviles:* (ed.). RA-MA Editorial. https://elibro.net/es/lc/uniminuto/titulos/62496
- MoureDev by Brais Moure. (17 de enero de 2020). ANDROID STUDIO: COMO Crear una APP (para Principiantes) [Tutorial] [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=BQaxPwZWboA