

Proyecto Tincar		
Código: 001	Versión: 1.0	Fecha:1/07/2025




Profesora:  
Ing. Mary Luz Rubiano

---

## Documento de Especificación de Arquitectura


---

Realizado por:  
**Edgar Humberto Mojica**  
**Karen Michel Palacios López**  
**Diego Andrés Hernández Altamiranda**  
**Cristian David Rincón Urrea**  
**David Santiago Becerra**

 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha:1/07/2025</b>


## HISTORIAL DE REVISIONES

<b>Fecha</b>	<b>Versión</b>	<b>Autor</b>	<b>Descripción</b>	<b>Revisado Por</b>
2025-08-30	1.0	Equipo de desarrollo	Creación inicial del Documento de Especificación de Arquitectura del proyecto TinCar.	ING. Mary Luz Rubiano
AAAA-MM-DD	1.1	Equipo de desarrollo	Ajustes según revisión técnica.	ING. Mary Luz Rubiano
AAAA-MM-DD	2.0	Equipo de desarrollo	Versión final para entrega oficial.	ING. Mary Luz Rubiano

 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

## Contenido

1.	Documento de Arquitectura de Software .....	4
1.1.	Introducción .....	4
1.2.	Propósito .....	4
1.3.	Alcance .....	5
1.4.	Referencias .....	5
1.5.	Definiciones acrónimos y abreviaciones .....	6
2.	Generalidades del Proyecto .....	7
2.1.	Problema a Resolver .....	7
2.2.	Descripción General del Sistema a Desarrollar .....	7
2.3.	Identificación de los Stakeholders y sus responsabilidades .....	7
3.	Vistas de la arquitectura .....	10
3.1.	Vista de Casos de Uso .....	11
3.2.	Vista de Procesos .....	12
3.2.1.	Diagrama de Actividades .....	14
3.3.	Vista Lógica .....	14
3.3.1.	Diagramas – Clases .....	17
3.4.	Vista de Implementación .....	18
3.4.1.	Diagrama de Paquetes .....	20
3.5.	Vista de Despliegue .....	20
3.5.1.	Diagrama de despliegue .....	22
4.	Arquitectura en capas .....	22
5.	Vista de Datos .....	25
5.1.	Modelo Relacional .....	25
6.	Definición de Interfaces de Usuario .....	26
7.	Características Generales de Calidad .....	27
8.	Stack Tecnológico – TinCar .....	29
8.1.1.	Lenguajes de Programación .....	29
8.1.2.	Frameworks y Librerías .....	29
8.1.3.	Back-End (Negocio): .....	29
8.1.4.	Base de Datos y Persistencia .....	29
8.1.5.	Infraestructura y DevOps .....	29
8.1.6.	Otras Herramientas .....	30

		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>


## **1. Documento de Arquitectura de Software**

### **1.1. Introducción**

El presente Documento de Especificación de Arquitectura (DEA) describe los aspectos técnicos y de diseño del sistema **TinCar**, una plataforma orientada a mitigar la congestión vehicular en Bogotá mediante la gestión eficiente de parqueaderos. El sistema conecta a conductores con arrendadores de espacios disponibles, facilitando un tránsito más fluido y sostenible. En este documento se presentan la arquitectura en capas, los patrones de diseño seleccionados, el stack tecnológico, el modelo relacional y los diagramas de despliegue. Con ello, se establecen las bases técnicas que guiarán el desarrollo, implementación y mantenimiento del sistema, asegurando calidad, seguridad y usabilidad.

### **1.2. Propósito**

El propósito de este documento es describir la arquitectura de software del proyecto **TinCar**, el cual busca abordar y mitigar el impacto negativo de la congestión vehicular que afecta de manera constante a la ciudad de Bogotá. Este documento servirá como guía para los desarrolladores, diseñadores y demás partes interesadas, proporcionando una visión clara y estructurada de la arquitectura del sistema, los componentes que lo conforman y la manera en que estos interactúan. Asimismo, permitirá alinear las

 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>


decisiones técnicas con los objetivos del proyecto, asegurando que el producto final cumpla con los requisitos funcionales y no funcionales establecidos.

### **1.3. Alcance**

El presente documento abarca la descripción de la arquitectura del sistema **TinCar**, incluyendo sus vistas lógicas, físicas y de procesos. Contempla la definición de los principales componentes del software, la interacción entre módulos, la infraestructura de despliegue y las decisiones arquitectónicas que orientan el desarrollo. Este documento se centra en la fase de diseño arquitectónico y no aborda en detalle la implementación ni las pruebas específicas de cada módulo.

### **1.4. Referencias**

1. Documento de Especificación de Requerimientos no funcionales.
2. Documento de Visión del Proyecto.
3. Plan de Proyecto del Sistema
4. IEEE 42010-2011: Systems and Software Engineering — Architecture Description.

		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

### 1.5. Definiciones acrónimos y abreviaciones

**ARQUITECTURA DE SOFTWARE:** conjunto de elementos estáticos, propios del diseño intelectual del sistema, que definen y dan forma tanto al código fuente, como al


Comportamiento del software en tiempo de ejecución. Naturalmente este diseño Arquitectónico ha de ajustarse a las necesidades y requisitos del proyecto.

**DESCRIPCION DE ARQUITECTURA:** colección de productos de documentación.

**VISTAS:** es una representación de un área de interés o perspectiva del sistema en alto nivel.

**TIPOS DE VISTAS:** especificación de una convención de cómo construir y usar una vista. Deben satisfacer la capacidad de creación y análisis de una vista.

**STAKEHOLDER:** Individuo, equipo u organización con intereses relativos al sistema.

		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

## **2. Generalidades del Proyecto**


### **2.1. Problema a Resolver**

La congestión vehicular en Bogotá representa una de las problemáticas más críticas de movilidad urbana, afectando la calidad de vida de los ciudadanos, la productividad y el medio ambiente. A pesar de múltiples intentos de mitigación mediante políticas públicas y proyectos de infraestructura, los resultados han sido insuficientes.

### **2.2. Descripción General del Sistema a Desarrollar**


**TinCar** es un sistema digital diseñado para contribuir a la reducción de la congestión vehicular mediante el uso de tecnologías innovadoras que promuevan una movilidad más eficiente y sostenible. El sistema permitirá la interacción entre conductores y arrendadores de parqueaderos, facilitando la gestión de reservas y optimizando el uso de espacios de estacionamiento en la ciudad. La solución contempla una aplicación web y móvil que ofrecerá funcionalidades como registro de usuarios, búsqueda de parqueaderos, gestión de reservas, pagos en línea y administración del sistema.

### **2.3. Identificación de los Stakeholders y sus responsabilidades**


 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha:1/07/2025</b>

STAKEHOLDER	DESCRIPCIÓN	ESCENARIO	Caso de Uso
<b>Equipo de Desarrollo</b>	Ingenieros de software, arquitectos y diseñadores encargados de implementar el sistema.	Se encuentran en fase de construcción del sistema y requieren lineamientos claros de la arquitectura.	<ul style="list-style-type: none"> <li>• Diseñar arquitectura</li> <li>• codificar módulos, integrar funcionalidades</li> <li>• documentar software.</li> </ul>
<b>Usuario Conductor</b>	Ciudadano que utiliza la aplicación para encontrar, reservar y pagar un parqueadero.	Necesita estacionar su vehículo en una zona específica de la ciudad.	<ul style="list-style-type: none"> <li>• Buscar parqueadero.</li> <li>• reservar espacio.</li> <li>• realizar pago en línea.</li> <li>• cancelar reserva.</li> </ul>
<b>Arrendador de Parqueadero</b>	Propietario o administrador de un espacio de estacionamiento disponible para arriendo.	Desea poner a disposición su parqueadero en el sistema y gestionar reservas.	<ul style="list-style-type: none"> <li>• Registrar parqueadero.</li> <li>• Actualizar disponibilidad.</li> <li>• Aceptar o rechazar reservas.</li> </ul>



 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

Administrador del Sistema	Responsable de la operación y mantenimiento general de la plataforma.	Requiere garantizar la disponibilidad, seguridad y correcto funcionamiento del sistema.	<ul style="list-style-type: none"> <li>• Gestionar usuarios.</li> <li>• Monitorear transacciones.</li> <li>• Generar reportes, resolver incidencias.</li> </ul>
Equipo de Pruebas (QA)	Encargados de validar la calidad del sistema y el cumplimiento de los requisitos.	Realizan pruebas sobre el sistema para detectar errores o inconsistencias.	<ul style="list-style-type: none"> <li>• Diseñar casos de prueba.</li> <li>• Ejecutar pruebas funcionales y no funcionales.</li> <li>• Reportar errores.</li> </ul>
Proveedor de Pasarela de Pagos	Entidad que gestiona las transacciones electrónicas entre usuarios y arrendadores.	Un conductor desea pagar la reserva a través de la aplicación.	<ul style="list-style-type: none"> <li>• Procesar pagos electrónicos</li> <li>• validar transacciones</li> <li>• confirmar recibos de pago.</li> </ul>
Soporte Técnico	Equipo encargado de dar asistencia a usuarios y arrendadores.	Un usuario tiene problemas para iniciar sesión o realizar un pago.	<ul style="list-style-type: none"> <li>• Atender solicitudes de soporte</li> <li>• Gestionar incidentes</li> </ul>


 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha:1/07/2025</b>

			<ul style="list-style-type: none"> <li>• Restablecer cuentas.</li> </ul>
--	--	--	--

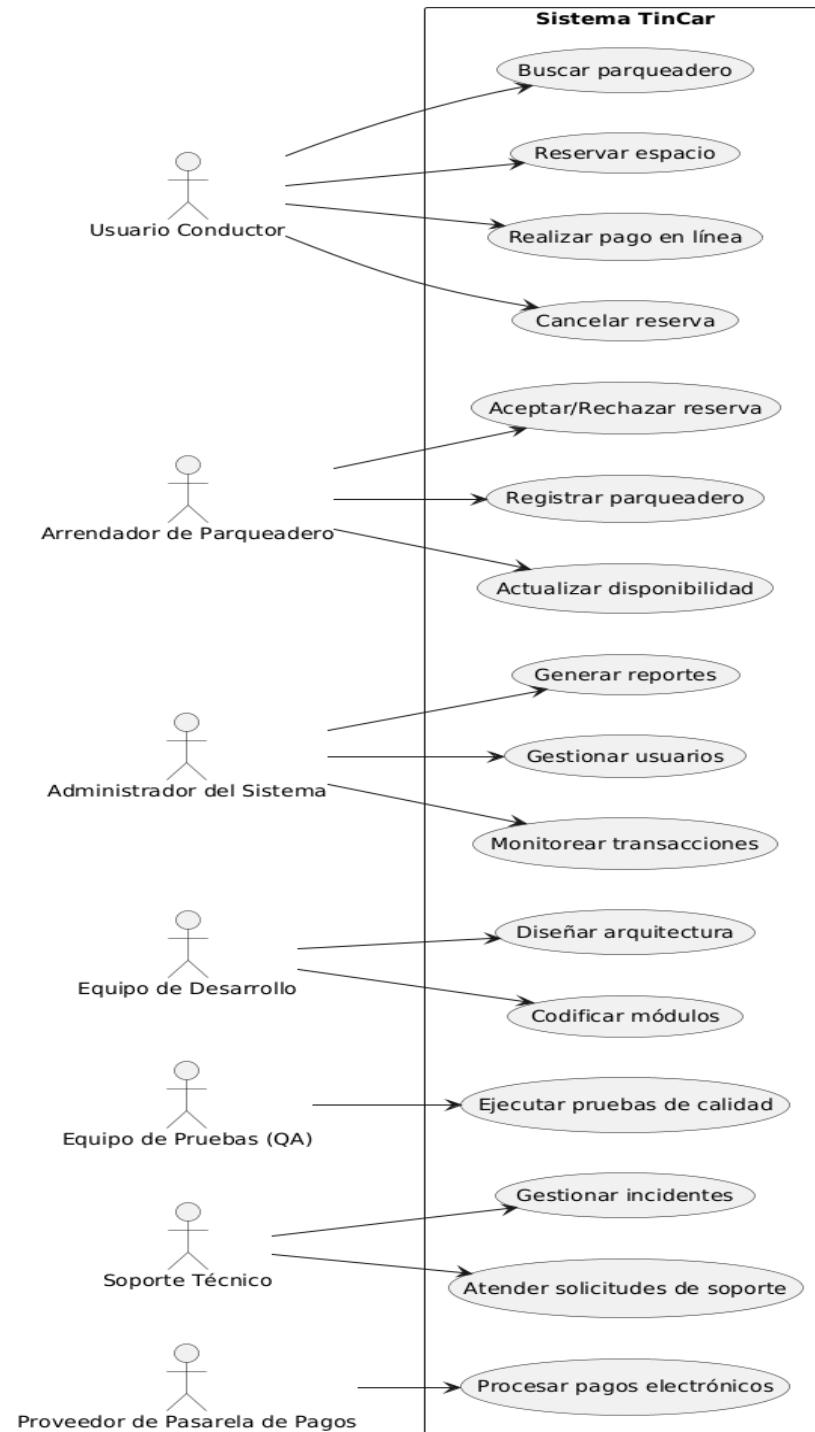
### 3. Vistas de la arquitectura


Este capítulo presenta distintas perspectivas del sistema para entender cómo se relacionan los componentes, procesos y actores del proyecto **TinCar**.

Las vistas ayudan a documentar las decisiones de diseño y facilitan la comunicación con los stakeholders. Para este proyecto se han definido las siguientes:

		
<h2>Proyecto Tincar</h2>		
Código: 001	Versión: 1.0	Fecha:1/07/2025

### 3.1. Vista de Casos de Uso



 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

### 3.2. Vista de Procesos

#### a. Procesos Principales del Sistema


Se identifican los procesos de negocio y técnicos que soportan la operación:

- Gestión de Usuarios
  - Registro, autenticación y autorización.
  - Roles: Conductor, Arrendador, Administrador.
- Gestión de Parqueaderos
  - Registro de parqueaderos (arrendadores).
  - Actualización de disponibilidad.
  - Validación de ubicación y espacios.
- Gestión de Reservas
  - Búsqueda de parqueaderos disponibles.
  - Creación y cancelación de reservas.
  - Aceptación/rechazo de solicitudes.
- Procesamiento de Pagos
  - Conexión con pasarela de pagos.
  - Validación de transacciones.
  - Confirmación y notificación.
- Monitoreo y Reportes
  - Generación de reportes de uso.
  - Monitoreo de transacciones y actividad del sistema.
- Soporte y Mantenimiento
  - Gestión de incidentes.
  - Atención a solicitudes de soporte.
  - Ejecución de pruebas de calidad (QA).

#### b. Flujo General de Procesos

Podemos representarlo en etapas dinámicas (alto nivel):

- Autenticación
  - El usuario (conductor o arrendador) accede al sistema → Servicio de autenticación valida credenciales.
- Reserva de Parqueadero
  - El conductor busca parqueaderos disponibles.

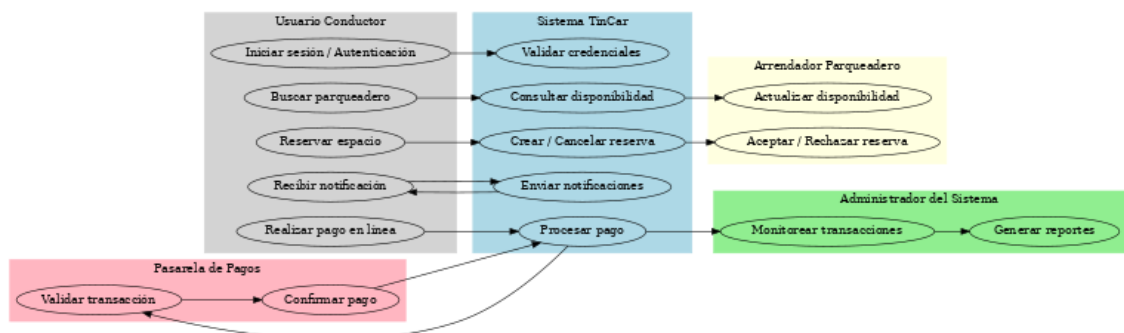
 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>


- El sistema consulta disponibilidad actualizada (proceso concurrente con arrendadores).
- El conductor realiza la reserva → El arrendador la acepta o rechaza.
- **Procesamiento de Pago**
  - Si se aprueba la reserva, el usuario procede al pago.
  - El sistema interactúa con la pasarela de pagos en un proceso independiente y seguro.
- **Confirmación y Notificación**
  - El sistema notifica al conductor y al arrendador el resultado (confirmación de reserva y pago).
- **Monitoreo y Soporte**
  - Administrador del sistema monitorea operaciones en paralelo.
  - Equipo de QA y Soporte Técnico gestionan incidencias y pruebas continuas.

### c. Diagrama de Procesos (Propuesta)

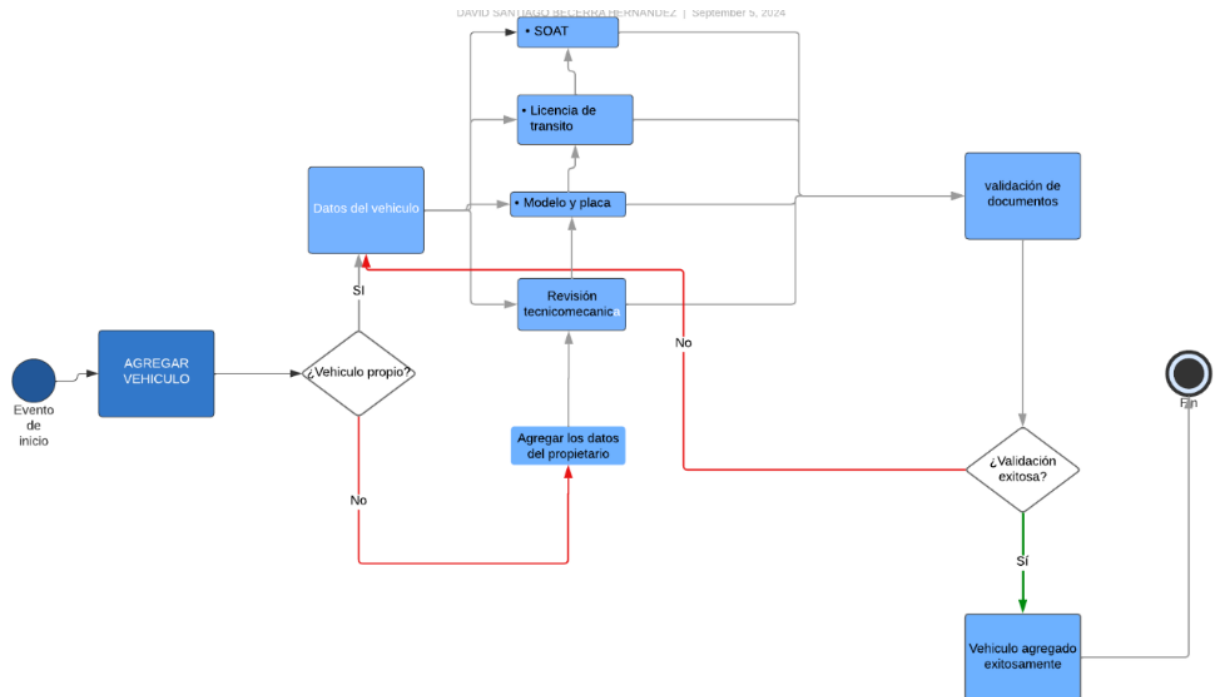
El diagrama de procesos debería mostrar los flujos concurrentes. Te lo describo para que lo visualices:

- **Conductor:** Solicita búsqueda de parqueadero.
- **Sistema TinCar:** Invoca proceso de disponibilidad (Arrendador).
- **Arrendador:** Actualiza disponibilidad en paralelo.
- **Sistema TinCar:** Gestiona reservas + pagos (en coordinación con pasarela).
- **Administrador:** Monitorea y genera reportes en un proceso concurrente.
- **Soporte y QA:** Intervienen de forma asincrónica en caso de fallos.



 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
Código: 001	Versión: 1.0	Fecha: 1/07/2025


### 3.2.1. Diagrama de Actividades



### 3.3. Vista Lógica

En **TinCar**, las clases principales son:

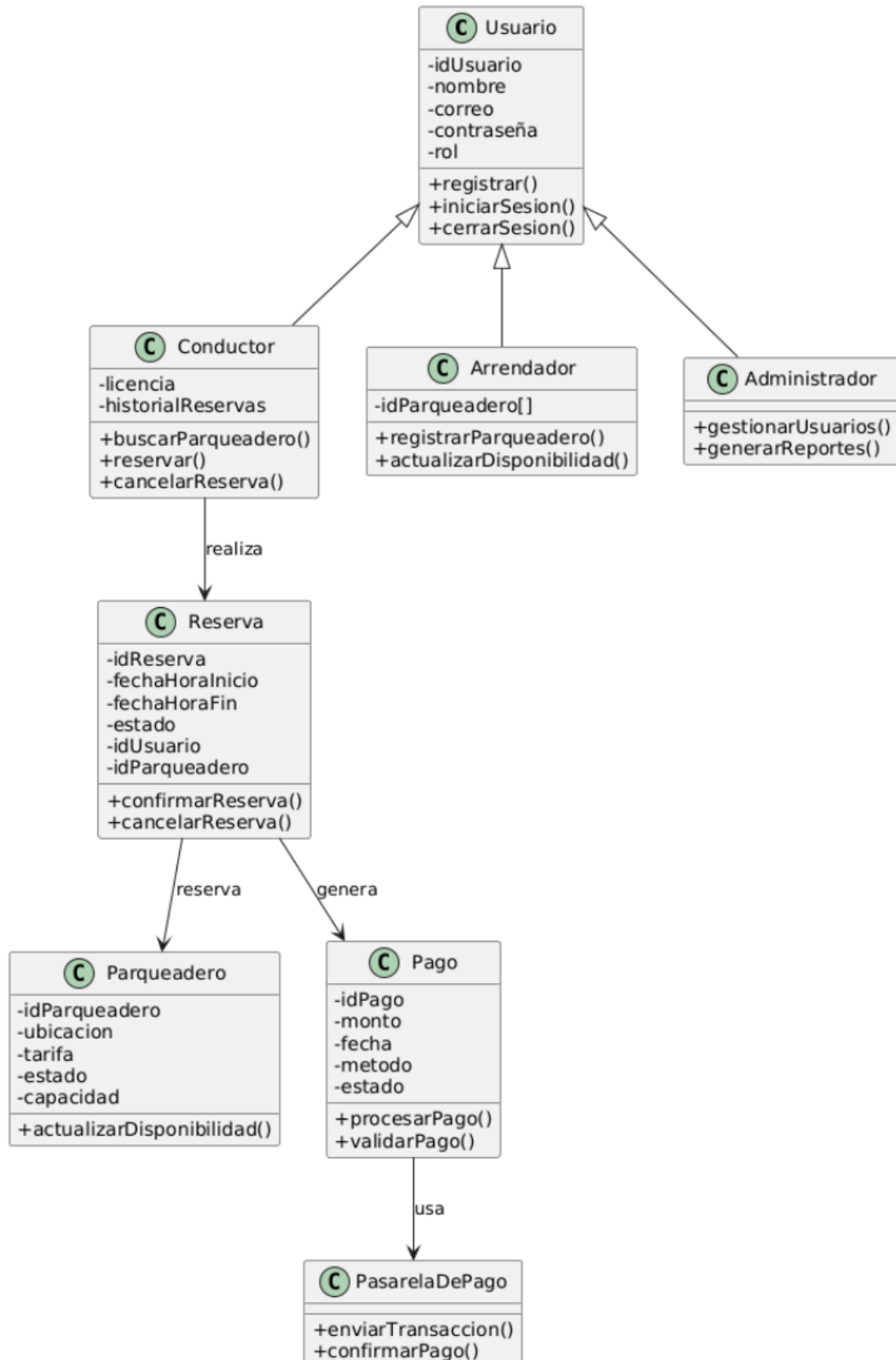
- **Usuario**
  - Atributos: idUsuario, nombre, correo, contraseña, rol
  - Métodos: registrar(), iniciarSesion(), cerrarSesion()
- **Conductor** (hereda de Usuario)
  - Atributos: licencia, historialReservas
  - Métodos: buscarParqueadero(), reservar(), cancelarReserva()
- **Arrendador** (hereda de Usuario)
  - Atributos: idParqueadero[]
  - Métodos: registrarParqueadero(), actualizarDisponibilidad()
- **Administrador** (hereda de Usuario)

 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>


- Métodos: gestionarUsuarios(), generarReportes()
- **Parqueadero**
  - Atributos: idParqueadero, ubicacion, tarifa, estado, capacidad
  - Métodos: actualizarDisponibilidad()
- **Reserva**
  - Atributos: idReserva, fechaHoraInicio, fechaHoraFin, estado, idUsuario, idParqueadero
  - Métodos: confirmarReserva(), cancelarReserva()
- **Pago**
  - Atributos: idPago, monto, fecha, metodo, estado
  - Métodos: procesarPago(), validarPago()
- **PasarelaDePago**
  - Métodos: enviarTransaccion(), confirmarPago()

Estas clases deben representarse en un **diagrama UML de clases**, con relaciones como:

- Asociación entre **Conductor** y **Reserva**.
- Asociación entre **Reserva** y **Parqueadero**.
- Asociación entre **Pago** y **Reserva**.
- Dependencia entre **Pago** y **PasarelaDePago**.

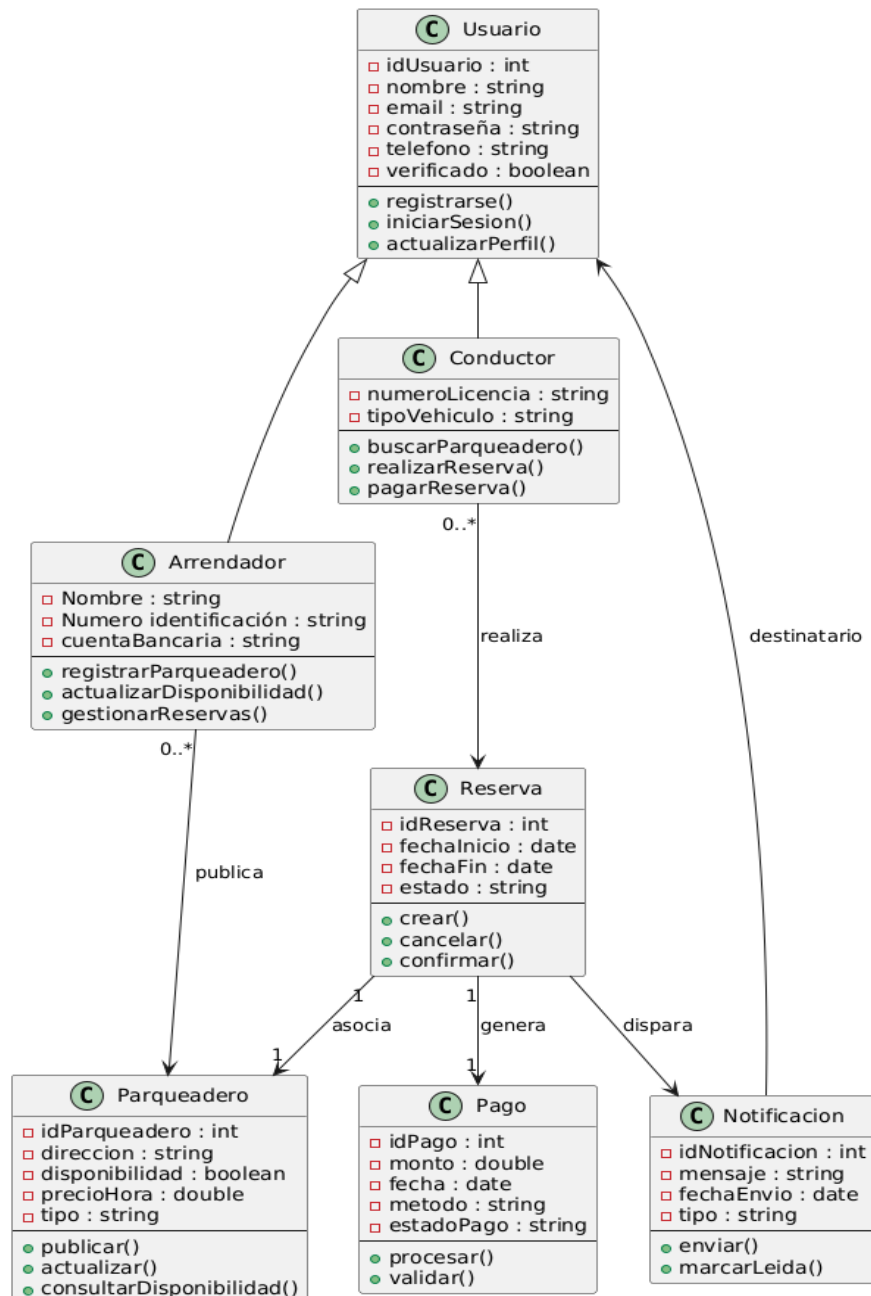





		
Proyecto Tincar		
Código: 001	Versión: 1.0	Fecha: 1/07/2025

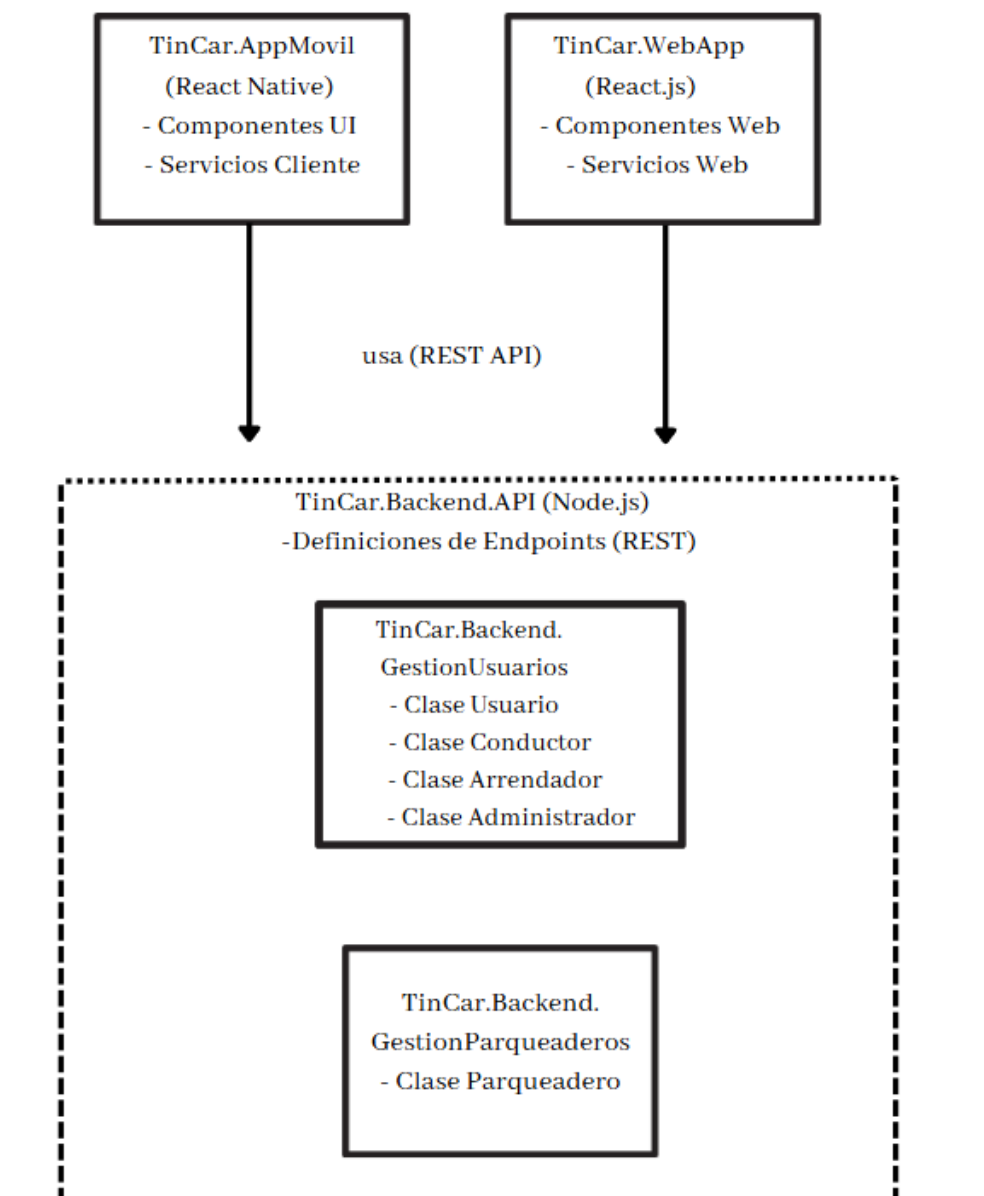
### 3.3.1. Diagramas – Clases


Diagrama de Clases - TinCar

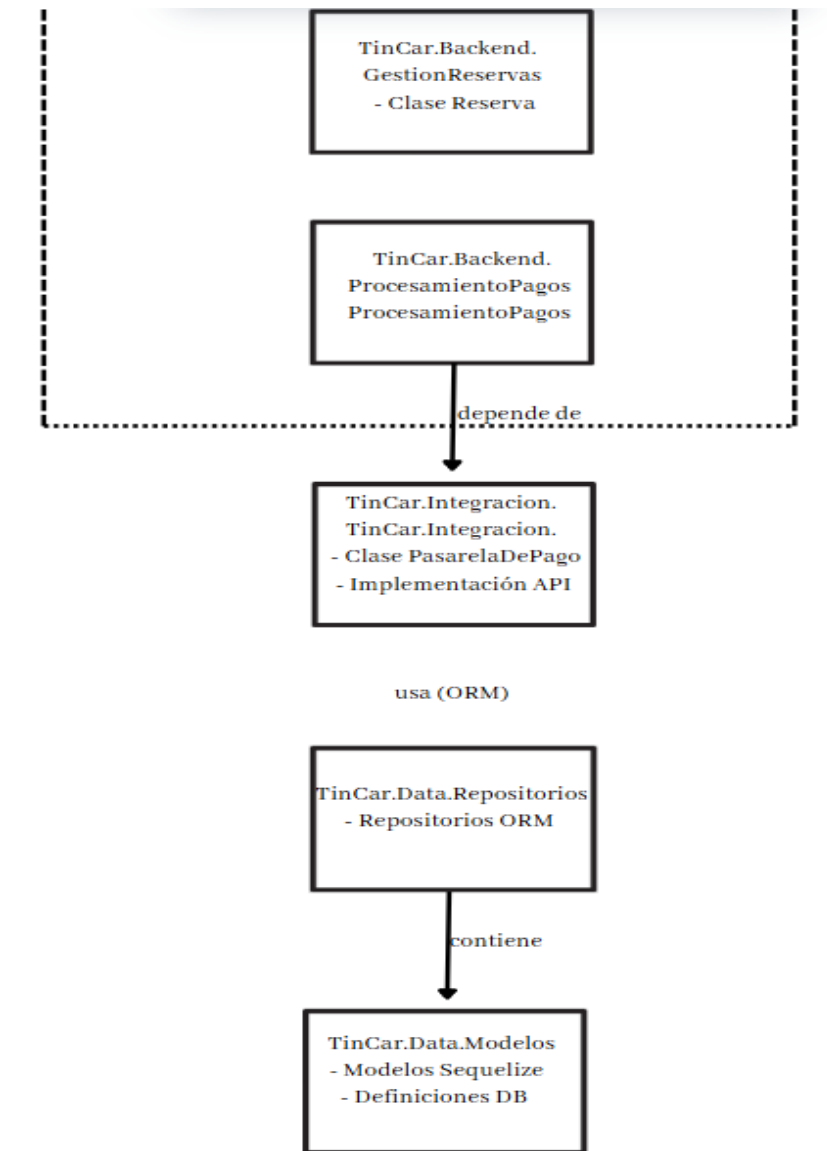



		
Proyecto Tincar		
Código: 001	Versión: 1.0	Fecha: 1/07/2025

### 3.4. Vista de Implementación

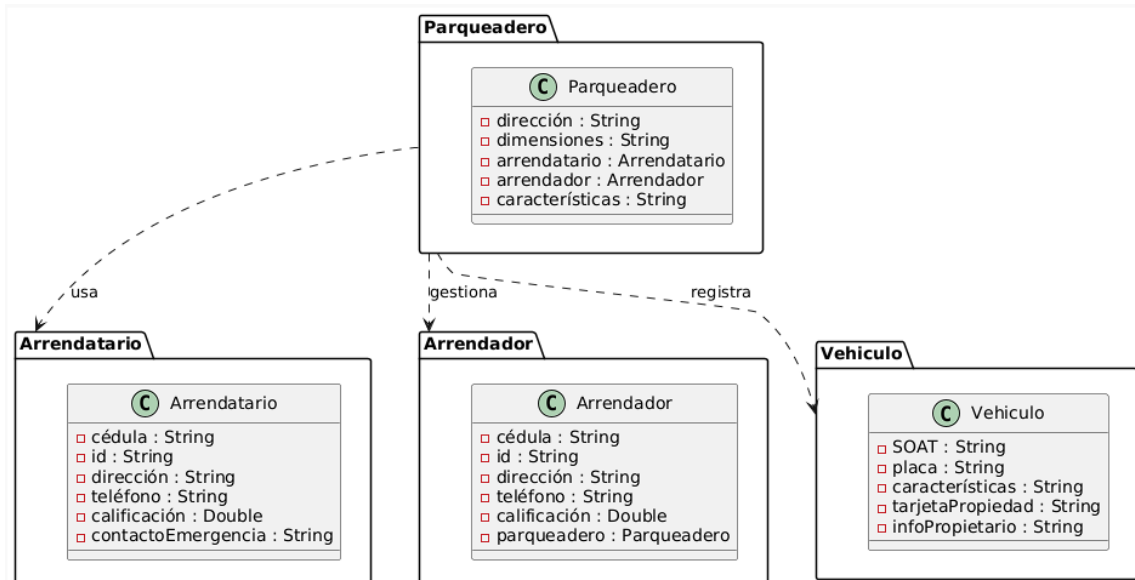


 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
Código: 001	Versión: 1.0	Fecha:1/07/2025



 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
Código: 001	Versión: 1.0	Fecha: 1/07/2025


### 3.4.1. Diagrama de Paquetes



### 3.5. Vista de Despliegue

#### a. Nodos Principales

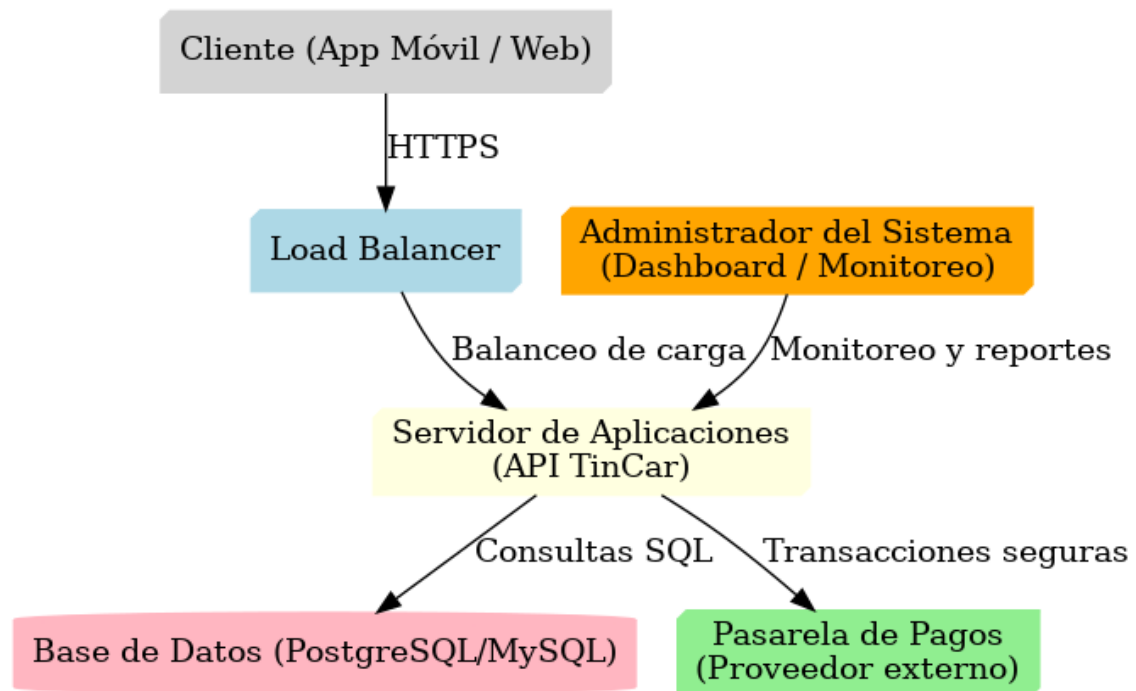
- Cliente (Front-end)
  - Aplicación móvil (Android / iOS).
  - Aplicación web (navegadores).
- Servidor de Aplicaciones (Back-end TinCar)
  - Servicios REST API (Flask, Django o Spring Boot, según tecnología elegida).
  - Módulos de negocio: gestión de usuarios, reservas, pagos, reportes.
- Base de Datos
  - Motor SQL (ejemplo: PostgreSQL / MySQL).
  - Tablas: usuarios, parqueaderos, reservas, transacciones, reportes.
- Pasarela de Pagos (Proveedor externo)
  - Servicio en la nube (PayU, Stripe, MercadoPago, etc.).
  - Comunicación segura vía HTTPS.
- Servicios de Monitoreo y Administración
  - Dashboard para administrador.
  - Logs y auditoría.


		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

- Infraestructura en la Nube (ejemplo AWS/Azure/GCP)
  - Load Balancer.
  - Servidores de aplicación escalables.
  - CDN para entregar contenido estático.

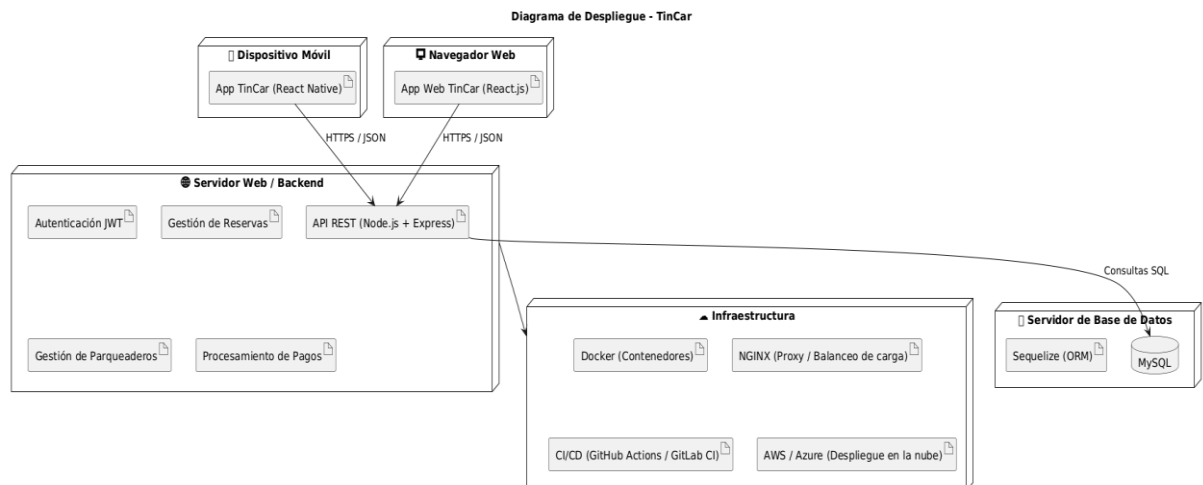
**b. Flujo de Despliegue**

- Usuarios (móvil/web) se conectan al Load Balancer.
- El balanceador distribuye las solicitudes a los Servidores de Aplicación TinCar.
- Los servidores consultan la Base de Datos y gestionan reservas.
- Para pagos, el back-end se comunica con la Pasarela de Pagos.
- El Administrador del Sistema accede al módulo de monitoreo/reportes vía interfaz web.



		
Proyecto TinCar		
Código: 001	Versión: 1.0	Fecha:1/07/2025

### 3.5.1. Diagrama de despliegue




## 4. Arquitectura en capas

Capa de Presentación (Front-End):

- Propósito: Encargada de la interfaz de usuario y de la interacción directa con los conductores y arrendadores. Es donde los usuarios buscarían parqueaderos, gestionarían reservas, realizarían pagos, etc.
- Componentes en TinCar: Incluiría la aplicación móvil (Android / iOS) y la aplicación web (navegadores)
- Tecnologías en TinCar: Utilizaría React Native para la aplicación móvil multiplataforma y React.js para la aplicación web
- Interacción: Se conectaría con la capa de lógica de negocio/aplicación a través de las APIs


Capa de Lógica de Negocio / Aplicación (Back-End):

		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

- **Propósito:** Contiene la lógica central del sistema, las reglas de negocio, la gestión de procesos y la orquestación de las operaciones. Aquí se manejarían los procesos principales del sistema, como la Gestión de Usuarios, Gestión de Parqueaderos, Gestión de Reservas, Procesamiento de Pagos, Monitoreo y Reportes, y Soporte y Mantenimiento
- **Componentes en TinCar:** Sería el Servidor de Aplicaciones (Back-end TinCar), que expondría Servicios REST API. Incluiría los módulos de negocio para gestionar usuarios, reservas, pagos y reportes
- **Tecnologías en TinCar:** El entorno de ejecución del servidor sería Node.js
- **Interacción:** Recibiría solicitudes de la capa de presentación, interactuaría con la capa de acceso a datos y con servicios externos como la pasarela de pagos

#### Capa de Acceso a Datos (Persistencia):

- **Propósito:** Responsable de la comunicación con la base de datos, manejando las operaciones de lectura y escritura. Abstrae la lógica de la base de datos de la lógica de negocio.
- **Componentes en TinCar:** Las clases de la Vista Lógica como Usuario, Parqueadero, Reserva, Pago tendrían sus métodos de persistencia implementados en esta capa.
- **Tecnologías en TinCar:** Se usaría Sequelize (ORM) para la abstracción de la base de datos y la gestión de entidades y relaciones mediante código
- **Interacción:** Recibiría comandos de la capa de lógica de negocio para interactuar con la capa de datos.

 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

#### Capa de Datos:

- **Propósito:** Almacenar de forma persistente toda la información del sistema.
- **Componentes en TinCar:** Sería la Base de Datos con un motor SQL, conteniendo tablas para usuarios, parqueaderos, reservas, transacciones, reportes, entre otros. El Modelo Relacional del proyecto sería clave en esta capa
- **Tecnologías en TinCar:** Se ha definido MySQL como la base de datos relacional

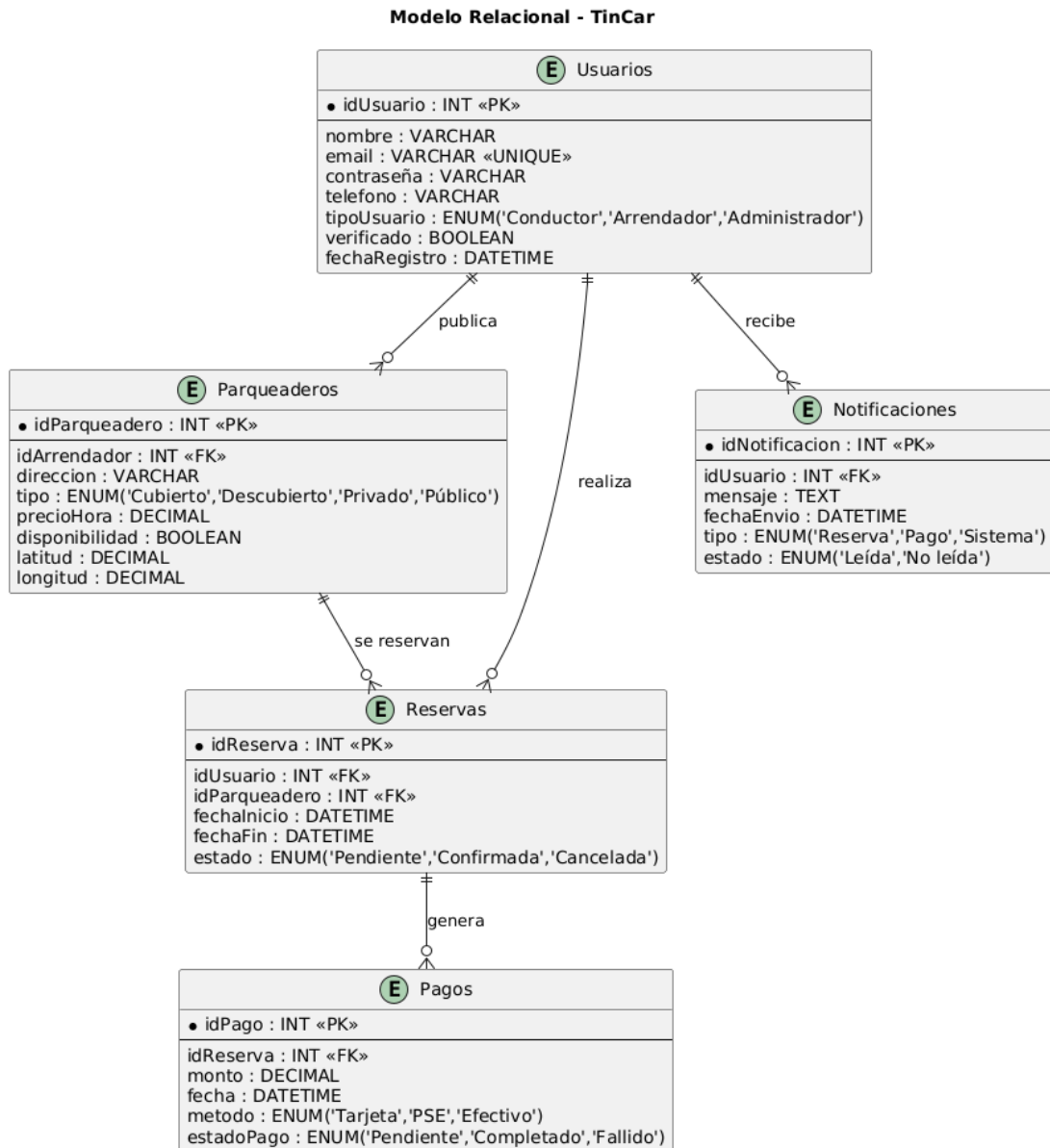
#### Capa de Servicios Externos / Integración:


- **Propósito:** Gestionar la comunicación con sistemas de terceros que ofrecen funcionalidades específicas.
- **Componentes en TinCar:** La Pasarela de Pagos (Proveedor externo) es el ejemplo principal, encargada de procesar pagos electrónicos y validar transacciones
- **Tecnologías en TinCar:** Podrían ser servicios como PayU, Stripe o MercadoPago, con comunicación segura vía HTTPS. La clase PasarelaDePago representaría la interfaz de interacción con esta capa.
- **Interacción:** La capa de lógica de negocio se comunicaría con esta capa para realizar transacciones de pago



## 5. Vista de Datos

### 5.1. Modelo Relacional




		
<div> <div>TinCar</div> <div>Proyecto Tincar</div> </div>		
Código: 001	Versión: 1.0	Fecha:1/07/2025

## 6. Definición de Interfaces de Usuario



### Características principales del diseño:

1. **Header con branding:** logo con los colores dorados característicos
2. **Barra de búsqueda intuitiva:** Para encontrar parqueaderos cercanos
3. **Filtros rápidos:** Chips para filtrar por proximidad, precio, horario, etc.
4. **Acciones rápidas:** Botones para reservar o rentar parqueaderos

 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

5. **Lista de parqueaderos:** Cards con información esencial como:

- Nombre del lugar
- Precio por hora
- Distancia
- Rating con estrellas
- Disponibilidad y características especiales

6. **Navegación inferior:** Con 5 secciones principales (Inicio, Buscar, Reservas, Rentar, Perfil)

## 7. Características Generales de Calidad

En el desarrollo de esta aplicación, se han considerado diversos atributos de calidad que garantizan su eficiencia, sostenibilidad y usabilidad. A continuación se describen y justifican los principales:

### Accesibilidad


La aplicación está diseñada para ser utilizada por personas con diferentes niveles de habilidad tecnológica, incluyendo opciones de navegación intuitiva, contraste adecuado y compatibilidad con lectores de pantalla. Esto permite que tanto arrendadores como arrendatarios puedan interactuar con la plataforma sin barreras técnicas o cognitivas.

### Responsabilidad

El sistema gestiona de forma segura los datos personales y financieros de los usuarios, cumpliendo con principios éticos y legales. Se implementan mecanismos de autenticación, trazabilidad de acciones y políticas de uso responsable para garantizar la integridad de la información.

### Precisión

La información mostrada en la app —como disponibilidad de parqueaderos, tarifas, horarios y datos de usuarios— se actualiza en tiempo real y se valida para evitar errores.

 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

Esto asegura que las decisiones tomadas por los usuarios estén basadas en datos confiables.

#### Adaptabilidad

La arquitectura modular permite que el sistema se ajuste fácilmente a nuevas condiciones, como cambios en políticas de arrendamiento, integración con pasarelas de pago o expansión a otras ciudades. Esto facilita su evolución sin necesidad de rediseños complejos.

#### Administrabilidad

Se han incorporado paneles de control para administradores del sistema, donde pueden gestionar usuarios, parqueaderos, reportes y métricas. Esto permite una supervisión eficiente y una toma de decisiones informada.

#### Mantenibilidad

El código está estructurado siguiendo buenas prácticas de desarrollo (como separación de responsabilidades y documentación interna), lo que facilita la corrección de errores, la incorporación de nuevas funcionalidades y la actualización tecnológica.

#### Manejabilidad

La app incluye herramientas para monitorear el rendimiento, detectar fallos y generar reportes de uso. Esto permite a los administradores anticiparse a problemas y optimizar la experiencia del usuario.

#### Movilidad


La solución está disponible en dispositivos móviles y de escritorio, permitiendo a los usuarios acceder desde cualquier lugar. Esto es clave para arrendatarios que buscan parqueaderos en tiempo real mientras se desplazan.

#### Modificabilidad

Gracias al uso de patrones de diseño y una arquitectura desacoplada, es posible modificar componentes individuales (como el módulo de pagos o el sistema de calificaciones) sin afectar el resto del sistema.

#### Modularidad

El sistema está dividido en módulos funcionales: Arrendador, Arrendatario, Parqueadero y Vehículo. Esta organización permite desarrollar, probar y mantener cada

 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha: 1/07/2025</b>

módulo de forma independiente, mejorando la escalabilidad y la colaboración entre equipos.

## 8. Stack Tecnológico – TinCar

### 8.1.1. Lenguajes de Programación

- **JavaScript (ES6+)** → Lenguaje principal para el desarrollo de la aplicación web y móvil.
- **SQL** → Definición y gestión de consultas en la base de datos relacional (MySQL).

### 8.1.2. Frameworks y Librerías

- **Front-End (Presentación):**
  - **React Native** → Desarrollo de la aplicación móvil multiplataforma (Android/iOS).
  - **React.js** → Desarrollo de la aplicación web.

### 8.1.3. Back-End (Negocio):


- **Node.js** → Entorno de ejecución del servidor.

### 8.1.4. Base de Datos y Persistencia

- **MySQL** → Base de datos relacional para la gestión de usuarios, parqueaderos, reservas y pagos.
- **Sequelize (ORM)** → Abstracción de la base de datos para manejar entidades y relaciones mediante código.

### 8.1.5. Infraestructura y DevOps

- **Git y GitHub/GitLab** → Control de versiones y gestión del código fuente.
- **CI/CD (GitHub Actions o GitLab CI)** → Integración y despliegue continuo del sistema.

 <b>TinCar</b>		
<b>Proyecto Tincar</b>		
<b>Código: 001</b>	<b>Versión: 1.0</b>	<b>Fecha:1/07/2025</b>

- **AWS / Azure (futuro)** → Despliegue en la nube con servicios de escalabilidad automática.

#### 8.1.6. Otras Herramientas

- **Postman** → Pruebas y documentación de las APIs.
- **Figma** → Diseño de prototipos de la interfaz de usuario.
- **Trello / Jira** → Gestión ágil de tareas y sprints.