Documentación Técnica: Sistema de Gestión de Parqueadero

1. Descripción del Software

- **Nombre del Software:** Sistema de Gestión de Parqueadero
- **Propósito:** Proveer una solución sencilla y robusta para la gestión de vehículos en un parqueadero, enfocándose en la automatización de los procesos de entrada, salida, cálculo de tarifas y gestión de capacidad.
- **Alcance:** El sistema gestiona el registro de vehículos (placa), calcula el tiempo de permanencia y el monto a pagar, y controla la capacidad máxima.

Funcionalidades Clave:

- **Registro de Entrada:** Asigna un vehículo a un puesto con un registro de hora.
- **Cálculo de Tarifa y Salida:** Calcula el costo total basado en el tiempo de parqueo (tarifa por hora) y libera el puesto.
- **Control de Capacidad:** Permite configurar y verificar la capacidad máxima del parqueadero.
- **Reporte de Ocupación:** Muestra una lista detallada de todos los vehículos actualmente en el parqueadero.

2. Arquitectura

Tecnologías Utilizadas:

- **Lenguaje de Programación:** JavaScript.
- **Framework de Pruebas:** Mocha.
- **Biblioteca de Aserciones:** Chai.

Justificación de Elección:

- **JavaScript/Node.js:** Ofrece una sintaxis moderna y una vasta comunidad, ideal para prototipos rápidos y desarrollo backend. Es altamente accesible y permite una implementación ágil.
- **Mocha y Chai:** Son herramientas estándar y maduras en el ecosistema de Node.js para TDD. Mocha proporciona la estructura para las suites de prueba (describe, it), y Chai ofrece una sintaxis de aserciones clara y legible (expect, assert). Su simplicidad garantiza que el enfoque TDD se mantenga ágil y no se complique con configuraciones excesivas.
- **Módulo Principal (Parqueadero):** Una clase que encapsula toda la lógica de negocio (registro, cálculo, reportes) y el estado interno (la lista de vehículos parqueados).
- **Módulo de Pruebas (Parqueadero.test.js):** Utiliza Mocha y Chai para verificar que cada método del módulo principal se comporte según las especificaciones, guiando así el desarrollo (TDD).

3. Metodología: Test-Driven Development (TDD)

- **Elección:** TDD (Desarrollo Guiado por Pruebas).
- **Implementación (Red-Green-Refactor):**
    - **Red:** Antes de escribir cualquier código de la funcionalidad, se escribe una prueba fallida para la nueva característica. Esta prueba debe ser atómica y cubrir un caso de uso específico (p. ej., "El parqueadero debe registrar una placa").
    - **Green:** Se escribe la cantidad mínima de código en el módulo principal (Parqueadero.js) necesaria para que la prueba pase. No se buscan soluciones perfectas, solo funcionales.
    - **Refactor:** Una vez que todas las pruebas son verdes, se mejora y optimiza el código existente sin cambiar su comportamiento. Las pruebas aseguran que el refactor no introduzca errores.

Herramientas para TDD:

Mocha: Utilizado para ejecutar los tests y estructurar las pruebas en suites.
Chai: Utilizado para escribir las aserciones (las condiciones que deben ser ciertas) en un formato legible.

Casos de prueba

ID	Descripción	Entrada	Resultados esperados	Fecha de ejecución
F1-01	Registrar un vehículo con una placa única.	Placa: "XYZ001"	Lista de vehículos aumenta en 1. El vehículo "XYZ001" está presente en el registro con hora de entrada.	25/10/2025 - 9:00 AM
F1-02	Intentar registrar una placa que ya está en el parqueadero.	Placa: "XYZ001" (ya registrada)	Debe lanzar una excepción indicando que la placa ya está registrada.	25/10/2025 - 10:00 AM
F2-01	Calcular el pago para un vehículo que estuvo 30 minutos (cobro mínimo de 1h).	Placa: "MIN789". Duración: 0.5 horas. Tarifa: $10,000/h.	Total a pagar: $10,000. El vehículo es removido del registro.	25/10/2025 - 2:00 PM
F2-02	Calcular el pago para un vehículo que estuvo 2 horas y 30 minutos (2.5h).	Placa: "LONG123". Duración: 2.5 horas. Tarifa: $10,000/h.	Total a pagar: $25,000. El vehículo es removido del registro.	25/10/2025 - 3:00 PM
F2-03	Intentar retirar un vehículo con una placa no registrada.	Placa: "ABC234"	Debe lanzar una excepción indicando que la placa no fue encontrada.	25/10/2025 - 4:00 PM
F3-01	Consultar la capacidad cuando el parqueadero está vacío (Capacidad Máx: 3).	Ninguna	Debe retornar el número 3 (puestos disponibles).	26/10/2025 - 10:00 AM
F3-02	Intentar registrar una entrada cuando la capacidad máxima ha sido alcanzada (3/3).	Placa: "FLL456"	Debe lanzar una excepción indicando "Parqueadero lleno."	26/10/2025 - 11:00 AM

ID	Descripción	Entrada	Resultados esperados	Fecha de ejecución
F4-01	Obtener el reporte con 3 vehículos registrados.	3 vehículos parqueados con diferentes horas de entrada.	Debe retornar un array de 3 objetos, cada uno con placa, hora de entrada y duración del parqueo.	26/10/2025 - 2:00 PM