UNIVERSIDAD NACIONAL EXPERIMENTAL DE GUAYANA

VICERRECTORADO ACADÉMICO

COORDINACIÓN GENERAL DE PREGRADO

PROYECTO DE CARRERA: INGENIERÍA EN INFORMÁTICA

CÁTEDRA: SISTEMAS DISTRIBUIDOS

SECCIÓN 1/SEMESTRE VIII

Proyecto:
Puente en una Vía

Profesor:

Virginia Padilla

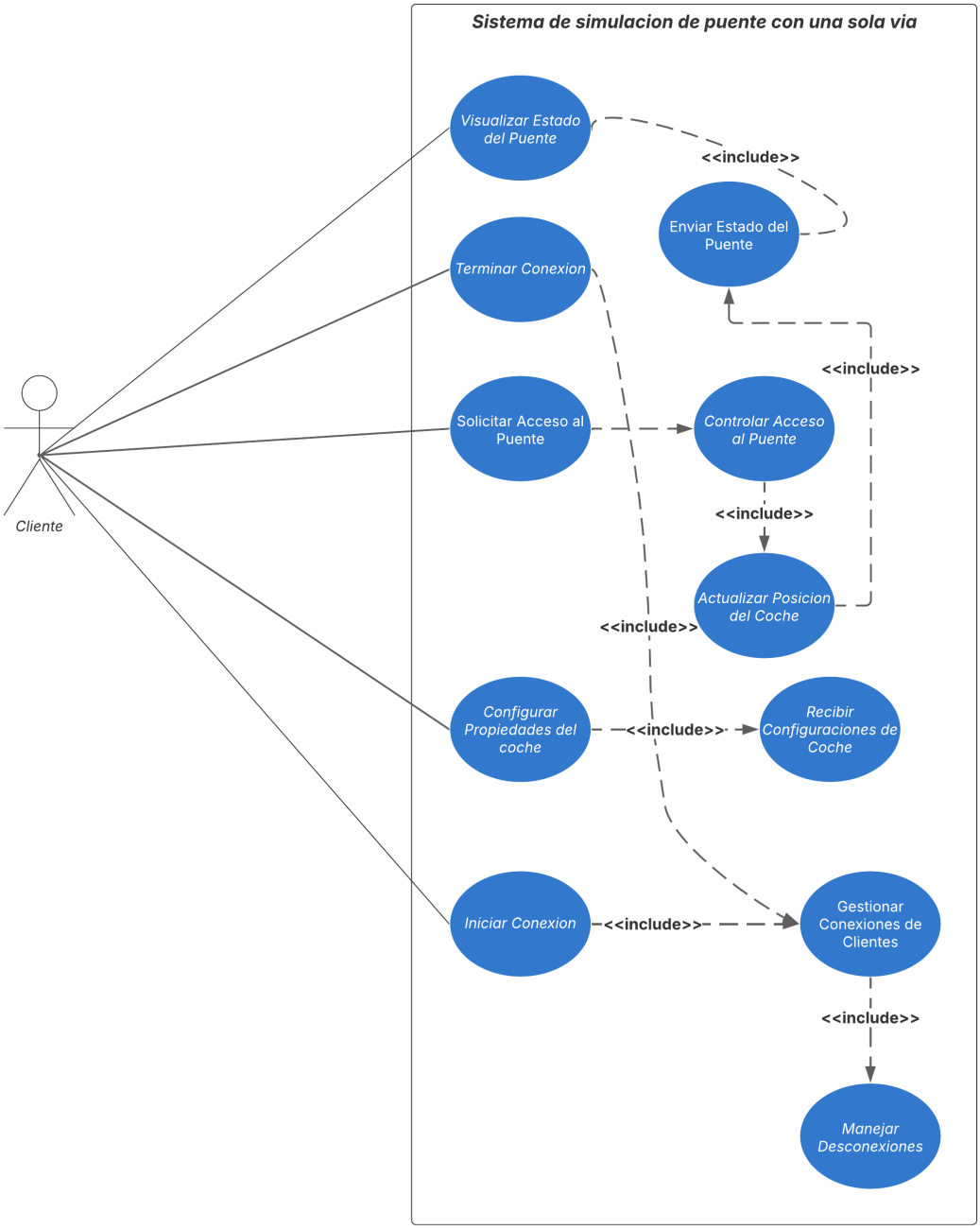
Alumnos:

Alanys Silva V- 30.857.778

Yorman Balan V- 31.782.211

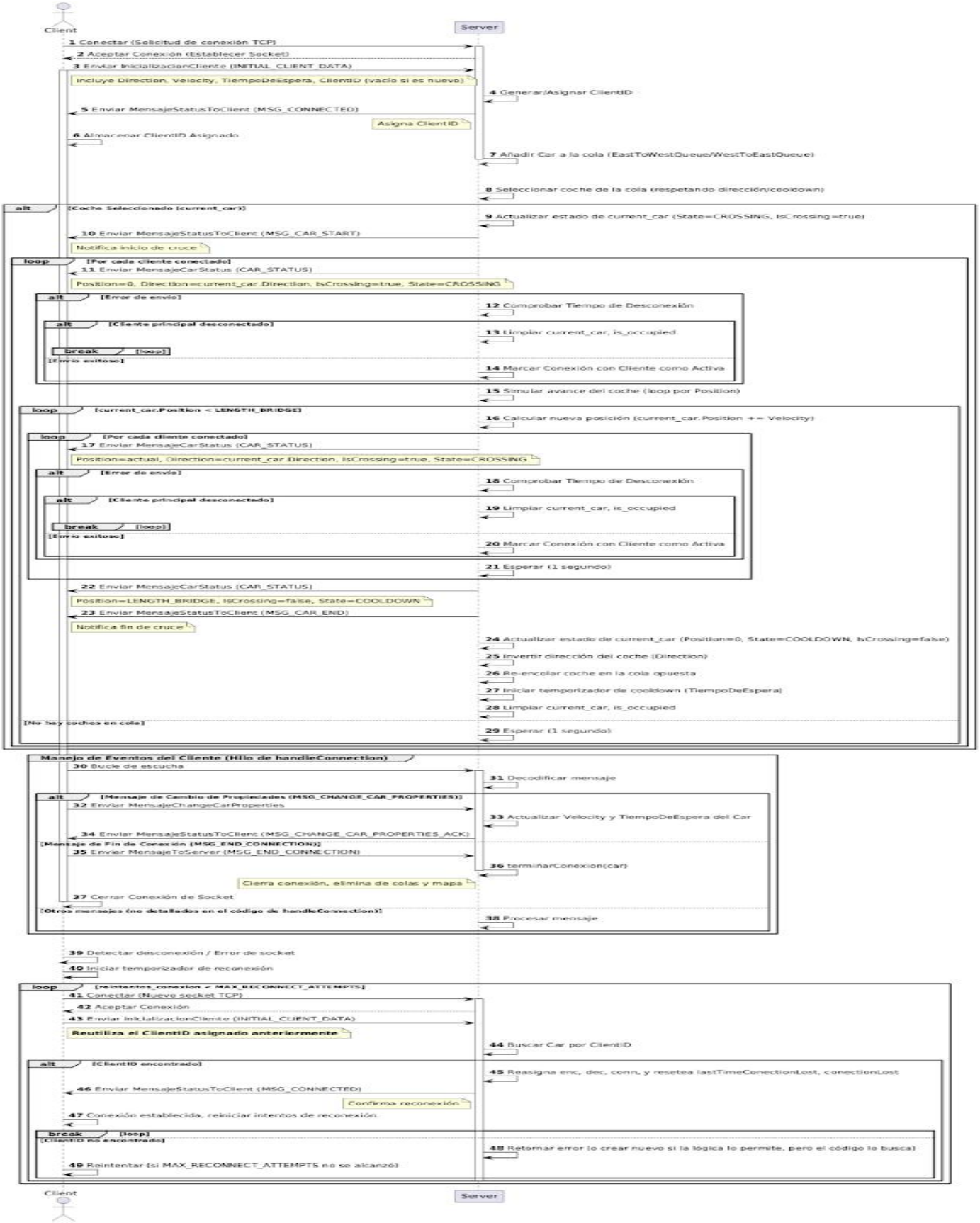
Puerto Ordaz, 15 de julio de 2025

Casos de Uso


Sistema de simulacion de puente con una sola via
Visualizar Estado del Puente
<<include>>
Enviar Estado del Puente
Terminar Conexion
<<include>>
Solicitar Acceso al Puente
Controlar Acceso al Puente
<<include>>
Actualizar Posicion del Coche
<<include>>
Configurar Propiedades del coche
<<include>>
Recibir Configuraciones de Coche
Iniciar Conexion
<<include>>
Gestionar Conexiones de Clientes
<<include>>
Manejar Desconexiones
Cliente

Diagramas de Interacción

Diagrama de secuencia de la aplicación completa



Componentes del diagrama:

- Cliente: La aplicación del usuario (coche).
- Servidor/Oyente: El punto de entrada inicial en el lado del servidor.
- Manejador de Conexión (por Cliente): Un objeto o proceso dedicado a gestionar la sesión de un cliente específico.
- Manejador del Puente: Lógica que controla el acceso y el movimiento de los coches a través de un "puente" o recurso compartido.
- Colas: Probablemente colas de espera para los coches (e.g., "Cola Este a Oeste", "Cola Oeste a Este").

Fases Probables:

1. **Inicio de Conexión (Parte superior):**
    - el cliente inicia una conexión.
    - Conectar TCP.
    - El cliente envía datos de inicialización.
    - El servidor asigna un ID al cliente y lo registra.

2. **Manejo del Bucle Principal - Cruce del Puente (Recuadros grandes en el medio):**

Lógica de selección/desencolado:

- El Manejador del Puente interactúa con las Colas para seleccionar el siguiente coche a cruzar.

Proceso de cruce:

- El Manejador del Puente notifica al Manejador de Conexión del coche seleccionado que puede empezar a cruzar.
- El Manejador de Conexión envía mensajes de estado (CAR_STATUS con posición) al Cliente mientras el coche avanza.
- bucle (loop) para enviar actualizaciones de posición hasta que el coche finaliza el cruce.
- Un mensaje final indicando el fin del cruce y quizás un estado de "cooldown".

Actualización de estado/Encolado post-cruce:

- El coche se encola de nuevo, posiblemente en la misma cola si es un bucle, o en la cola opuesta si cambia de dirección.

3. **Cambio de Propiedades (Recuadro más abajo, "Gestión de Eventos del Cliente"):**
    - El Cliente envía un mensaje para cambiar propiedades del coche (velocidad, tiempo de espera).

- El Manejador de Conexión actualiza estos datos internamente.
- Se envía un mensaje de confirmación al Cliente.

4. **Desconexión (Recuadro inferior):**
   - El Cliente envía un mensaje para desconectarse.
   - El Manejador de Conexión realiza tareas de limpieza:
     - ✓ Desencolar el coche de las colas.
     - ✓ Eliminar el coche de su registro interno.
     - ✓ Detener las rutinas de manejo de la conexión.
     - ✓ Cerrar la conexión TCP.
5. **Reconexión (Recuadro más inferior):**
   - El Cliente intenta reconectarse, abriendo un nuevo socket TCP.
   - El Cliente se re-inicializa, proporcionando su IDCliente previamente asignado.
   - El Manejador de Conexión busca el coche por su ID y restablece su estado.
   - Se envía una confirmación de reconexión al Cliente.

Diagrama de secuencia enfocado en la comunicación

Este diagrama describe la interacción de una Aplicación de Coche (Cliente) con varios componentes de un servidor para gestionar el cruce de coches por un puente.

Fases Principales:

1. **Conexión Inicial:**
    - El Cliente se conecta al Oyente del Servidor.
    - Se crea un Manejador de Conexión (por Cliente) dedicado.
    - El Cliente envía datos de inicialización (dirección, velocidad, etc.).
    - El Manejador genera un IDCliente, lo envía al Cliente, y encola el coche en la Cola Este a Oeste o Cola Oeste a Este según su dirección.
    - Se inicia una rutina concurrente (handleConnection) para gestionar la comunicación continua con este cliente.
2. **Cruce del Coche (Bucle del Manejador del Puente):**
    - El Manejador del Puente desencola un coche de la cola correspondiente (Este a Oeste o Oeste a Este).
    - Informa al Manejador de Conexión del cliente que el coche inicia el cruce.
    - El Manejador de Conexión envía actualizaciones de estado (posición y CRUZANDO) al Cliente.
    - Una vez que el coche llega al final, su estado cambia a EN_COOLDOWN.
    - El coche se vuelve a encolar, posiblemente para un nuevo cruce o con un cambio de dirección.
3. **Cambio de Propiedades del Cliente:**
    - El Cliente puede enviar un mensaje para cambiar propiedades como Velocidad o TiempoEspera.
    - El Manejador de Conexión actualiza estas propiedades internamente y envía una confirmación al Cliente.
4. **Desconexión del Cliente:**
    - El Cliente envía un mensaje para finalizar la conexión.
    - El Manejador de Conexión desencola el coche de la cola, elimina los datos del coche y termina la rutina de manejo de conexión.
    - El Cliente cierra su conexión TCP.
5. **Reconexión del Cliente:**
    - El Cliente establece una nueva conexión TCP.
    - Se re-inicializa enviando su IDCliente conocido.

- El Manejador de Conexión busca el coche por IDCliente para restablecer su estado.
- Confirma la reconexión exitosa al Cliente.

Arquitectura del Proyecto

El proyecto realizado tiene una arquitectura de **Cliente-Servidor**

- **Servidor (Go):** Actúa como la entidad central que gestiona el estado del puente y coordina las acciones de los coches. Es responsable de:
  - ✓ Gestionar las conexiones de los clientes entrantes.
  - ✓ Mantener el estado de todos los coches conectados (tablaClientes).
  - ✓ Implementar la lógica del puente (qué coche cruza, en qué dirección, cuándo, tiempos de espera/cooldown).
  - ✓ Distribuir actualizaciones de estado de los coches a todos los clientes.
  - ✓ Detectar y gestionar las desconexiones de los clientes, eliminando sus coches del sistema.
  - ✓ Generar clientes aleatorios para simulación.
- **Cliente (Python con Pygame):** Actúa como una interfaz gráfica de usuario para un solo coche y su interacción con el puente. Es responsable de:
  - ✓ Establecer y mantener una conexión con el servidor.
  - ✓ Enviar solicitudes de cambio de propiedades (velocidad, tiempo de espera).
  - ✓ Recibir y procesar las actualizaciones de estado de *todos* los coches del servidor.
  - ✓ Renderizar el estado del puente y todos los coches en una interfaz gráfica (Pygame).
  - ✓ Permitir la interacción del usuario para controlar su propio coche (velocidad, tiempo de espera, terminar conexión, simular caída).
  - ✓ Gestionar la lógica de reconexión si la conexión con el servidor se pierde.

La comunicación entre el cliente y el servidor es bidireccional ya que los clientes envían información de inicialización y solicitudes de cambio de propiedades al servidor. Y el servidor envía constantemente actualizaciones de estado a todos los clientes.

Protocolos Usados

El proyecto utiliza los siguientes protocolos:

1. **TCP (Transmission Control Protocol):**

Es el protocolo fundamental para la comunicación entre el cliente y el servidor. Proporciona una conexión orientada a la conexión, fiable y ordenada. Esto significa que los datos se garantizan para llegar en el orden correcto y sin pérdidas. Tanto el servidor como el cliente establecen y mantienen conexiones TCP.

2. **JSON:**

Se utilizo Capa de Aplicación/Formato de Datos para la serialización y deserialización de los mensajes intercambiados entre el cliente y el servidor.

Los mensajes son estructuras de datos (objetos) que se convierten a cadenas JSON antes de ser enviados por la red y se vuelven a convertir en objetos en el lado receptor.

Esto permitió una comunicación flexible y legible entre el cliente (Python) y el servidor (Go) a pesar de estar implementados en lenguajes diferentes.

Mensajes de interacción del sistema

- CAR_STATUS: Este mensaje es enviado por el **servidor al cliente** para actualizar el estado actual de un coche en la simulación. Incluye información como la posición del coche, su dirección, si está cruzando el puente y su estado general (esperando, cruzando, enfriamiento).
- CAR_END: Enviado por el **servidor al cliente** para notificar que un coche ha terminado de cruzar el puente y ha salido de él. El cliente utiliza esta información para eliminar el coche de su visualización.
- CAR_START: Este mensaje es enviado por el **servidor al cliente** para indicar que un coche ha comenzado su travesía por el puente. Esto permite al cliente iniciar la animación de cruce y resaltar el coche activo.
- CONNECTED: Enviado por el **servidor al cliente** inmediatamente después de establecer una conexión exitosa. Este mensaje confirma la conexión y a menudo incluye el clientId asignado al cliente.
- CHANGE_CAR_PROPERTIES: Este mensaje es enviado por el **cliente al servidor** para solicitar un cambio en las propiedades de su coche, como la velocidad o el tiempo de espera.
- CHANGE_CAR_PROPERTIES_ACK: Enviado por el **servidor al cliente** como confirmación de que la solicitud de cambio de propiedades del coche (MSG_CHANGE_CAR_PROPERTIES) ha sido recibida y procesada.

- END_CONNECTION: Enviado por el **cliente al servidor** para solicitar el cierre de la conexión de forma limpia. Esto permite al servidor liberar los recursos asociados a ese cliente.

Anexos

Video explicativo de la ejecución del proyecto:
https://drive.google.com/drive/folders/15qlrXXO7ggAnyt22MwHlAFruR-nJvttZ?usp=sharing

Video explicativo del código del proyecto:
https://drive.google.com/file/d/1cs1qpG2HIUkSmIxZrhiYsIC0LvO8DhJl/view?usp=sharing

Repositorio de github del proyecto:
https://github.com/yormanbalanD/proyecto_sistemas_distribuidos