Universidad ORT Uruguay

Facultad de Ingeniería

1er Obligatorio Programación de Redes

25 de Abril de 2024

Índice.

Alcance de la Aplicación

El sistema "Triportunity" está diseñado para permitir a los usuarios compartir y acceder a viajes en automóviles. Los usuarios pueden publicar detalles sobre viajes disponibles,

incluyendo la información crucial como punto de salida, destino, precio, cantidad de asientos disponibles, si el viaje es pet friendly o no, fecha y hora de salida e imagen del vehículo.

Las funcionalidades del sistema se dividen entre dos aplicaciones principales: una aplicación cliente y una aplicación servidor, complementadas por un programa auxiliar, Helpers, que facilita la comunicación entre ambas aplicaciones del lado del servidor. El cliente permite a los usuarios interactuar con el sistema para gestionar sus viajes y cuenta, mientras que el servidor procesa estas solicitudes, maneja la lógica de negocio, y mantiene la persistencia de usuarios y viajes. La integración de estas aplicaciones asegura un funcionamiento fluido y eficiente.

Errores Conocidos y Mejoras

Errores:

Cuando el cliente pregunta por los comentarios/rating de un conductor. Si el conductor no existe, el cliente falla y se queda trancada la aplicación. Se debería validar y comunicar al cliente cuando un conductor con ese username no existe, como si se validan las otra acciones (unirse a un viaje, editar un viaje, etc).

Para reproducir: Loguearse, luego ir a la acción 7, y apretar 9 luego ingresar un usuario que no pertenece a un usuario.

Mejoras:

Actualmente las imágenes de los viajes se guardan en la ruta "C:\FotosPr". Una mejora posible es permitirle al cliente elegir la ruta donde se guardan sus imágenes en un archivo de configuración.

Supuestos

1. Usuario entra mal la imagen, se crea un viaje sin imagen hasta que el usuario lo modifica:
    a. Si un usuario intenta subir una imagen que no cumple con los requisitos del sistema (por ejemplo, formato incorrecto, tamaño de archivo demasiado grande, o un error de carga), el sistema permitirá que el viaje se cree sin una imagen asociada. Esto significa que el viaje se puede registrar sin una imagen, evitando así que el error en la carga de la imagen impida la creación del viaje.
2. El numeral (#) es un caracter no válido:
    a. En el contexto de la aplicación, ciertos caracteres especiales, como el numeral (#), estan restringidos para su uso en ciertos campos. Esto se debe a que el carácter tiene un uso específico en el sistema, este es utilizado como separador a la hora de enviar los datos del cliente al servidor.

Descripción Arquitectónica

La arquitectura del **SISTEMA TRIPORTUNITY**, que permite compartir viajes en auto, se divide principalmente en dos aplicaciones: el servidor y el cliente, que interactúan entre sí a través de una red utilizando protocolos de comunicación estándar. A continuación se describe detalladamente cada componente y su interacción

Componentes del Sistema:

Aplicación Servidor

- **Funcionalidad**: El servidor es el núcleo del sistema, responsable de manejar las solicitudes de los clientes, procesarlas y enviar las respuestas adecuadas. Gestiona toda la lógica relacionada con usuarios y viajes.
- **Estructura:**
- **Socket Server**: Inicia un socket en la dirección IP "127.0.0.1" y puerto "20000", esperando conexiones de clientes. Utiliza TCP como protocolo de transporte, lo cual garantiza la entrega y el orden correcto de los paquetes.
- **Manejadores y Controladores**:
- **UserController**: Gestiona todas las operaciones relacionadas con los usuarios, como registro (signup), inicio de sesión (login), y otras funcionalidades de gestión de perfil.
- **TripControlle**r: Maneja las operaciones relacionadas con los viajes, incluyendo crear, actualizar, eliminar viajes, y permitir a los usuarios unirse a viajes existentes.
- **Concurrencia**: Utiliza hilos para manejar múltiples clientes simultáneamente. Cada conexión cliente se maneja en un nuevo hilo, permitiendo así procesos concurrentes sin bloquear la ejecución del servidor.

Aplicación Cliente

- **Funcionalidad**: La aplicación cliente permite a los usuarios interactuar con el sistema, enviando solicitudes al servidor y recibiendo respuestas. Provee una interfaz para registrar y gestionar viajes, así como para la interacción con otros usuarios.
- **Estructura**:
- **Socket Client**: Se conecta al servidor a través del mismo IP y puerto, estableciendo un canal de comunicación. La conexión se realiza de manera explícita mediante la función Connect.
- **UserController**: A través de esta clase, el cliente puede navegar por los menús iniciales y realizar acciones como registrarse o iniciar sesión.

Helpers

Connections

Se encarga principalmente de la comunicación entre Server y Client. Tiene dos principales funcionalidades: enviar y recibir mensajes.

- SendMessage: Este método toma un mensaje de texto y un socket cliente como parámetros. El texto es codificado a bytes utilizando UTF-8 y luego es enviado a través del socket. Primero se envía la longitud del mensaje codificado para que el receptor sepa cuántos bytes necesita leer para obtener el mensaje completo.
- ReceiveMessage: Recibe datos de un socket cliente. Inicialmente, lee la longitud del mensaje esperado y, basándose en esa longitud, lee el mensaje completo. El mensaje recibido en bytes es convertido de nuevo a texto utilizando UTF-8 para su uso posterior.

Estos métodos garantizan que los mensajes enviados y recibidos entre el cliente y el servidor sean precisos y completos, manejando adecuadamente la codificación y decodificación de los datos para prevenir errores comunes de transmisión o interpretación de caracteres.

Otros

El resto de las clases dentro de Helpers son principalmente utilizadas para la conversión, envío y recepción de la imagen del vehículo.

- **SettingsManager**: Gestiona la lectura de configuraciones desde un archivo, permitiendo la recuperación de valores configurados mediante claves, lo que facilita la adaptabilidad del software a diferentes entornos sin necesidad de cambiar el código.
- **FileHandler**: Proporciona funcionalidades esenciales para interactuar con el sistema de archivos, como verificar la existencia de archivos, obtener su nombre y determinar su tamaño, funciones críticas para la preparación de archivos antes de realizar operaciones más complejas como la lectura o escritura.
- **FileCommsHandler**: Encargada de la comunicación de archivos sobre redes utilizando sockets, esta clase maneja el envío y recepción de archivos asegurando que la transferencia se realice de manera correcta y eficiente, integrando otras clases para la conversión de datos y la gestión de flujos.
- **ConversionHandler**: Facilita la conversión entre diferentes tipos de datos y su representación en bytes, esencial para la transmisión eficiente de datos a través de redes o su almacenamiento. Permite conversiones bidireccionales entre cadenas y bytes, así como entre enteros y bytes.
- **FileStreamHandler**: Mejora el manejo de archivos permitiendo leer y escribir datos con control preciso sobre su posición, lo que es útil para aplicaciones que requieren acceso aleatorio a los datos o necesitan anexar datos a archivos existentes de manera eficiente.
- **Protocol**: Define constantes y métodos para estandarizar el tamaño de los paquetes de datos y calcular la cantidad de partes necesarias para transmitir un archivo completo, asegurando una transmisión ordenada y completa de todos los componentes del archivo.

Comunicación

Protocolo TCP/IP

Se ha seleccionado TCP como el protocolo de transporte debido a su orientación a la conexión, lo que es esencial para garantizar la integridad y el orden correcto de los datos transmitidos.

Formato de Datos

La comunicación entre el cliente y el servidor se realiza mediante el intercambio de mensajes estructurados como cadenas de texto. A diferencia de utilizar formatos de serialización automática como JSON, que está prohibido en este proyecto, los mensajes son codificados manualmente en UTF-8 y enviados como bytes a través del socket. Cada mensaje incluye un prefijo con su longitud total, lo que facilita la recepción y reconstrucción del mensaje completo antes de procesarlo. Esto es crucial para manejar correctamente la fragmentación de mensajes que puede ocurrir en redes TCP.

Decisiones Críticas

Manejo de Errores

La gestión eficaz de errores es fundamental durante los procesos de autenticación como el login y el signin. A continuación se detallan los procedimientos para manejar errores de manera efectiva en estos procesos.

1. **Signin (Registro)**
    a. **Cliente**: Envía al servidor los datos del usuario, como el nombre de usuario y la contraseña deseada.
    b. **Servidor:** Utiliza controladores específicos para validar si el nombre de usuario está disponible.
        i. **Si está disponible**: Se procede con el registro y se envía una confirmación al cliente.
        ii. **Si no está disponible**: Se envía un mensaje de error al cliente, especificando que el nombre de usuario ya está en uso.
2. **Login (Inicio de sesión)**
    a. **Cliente**: Envía al servidor el nombre de usuario y la contraseña.
    b. **Servidor**: Verifica la existencia del usuario y la corrección de la contraseña utilizando controladores adecuados.
        i. **Si la verificación es exitosa**: Envía una confirmación al cliente para permitir el acceso a la aplicación.
        ii. **Si hay un error (usuario no existe o la contraseña es incorrecta)**: Se envía un mensaje de error al cliente, indicando que los datos de autenticación son incorrectos.

El correcto manejo de errores de formato en las entradas del usuario es crucial para asegurar la integridad de los datos en la aplicación. A continuación se describe cómo se validan diferentes tipos de datos para prevenir y manejar errores de formato:

1. **Fechas**
    - **Cliente**: Antes de enviar datos al servidor, el cliente verifica el formato de las fechas ingresadas por el usuario.
    - **Formato Requerido**: dd/MM/YYYY.
    - **Validación**: El cliente utiliza expresiones regulares o funciones de fecha para asegurarse de que el formato y los valores son correctos (por ejemplo, que el día y el mes son válidos).
    - **Errores Comunes**: Fechas como 35/03/2023 o 12/13/2023 serán rechazadas.
    - **Bucle de Corrección**: Si una fecha es inválida, el cliente notifica al usuario y solicita la reentrada de la fecha correcta. Este proceso se repite hasta que se ingresa un valor válido.
2. **Valoración del Conductor**
    - **Cliente**: Valida que la valoración ingresada por el usuario esté dentro del rango permitido.
    - **Rango Válido**: Entero entre 1 y 5.
    - **Bucle de Corrección**: Si el valor está fuera del rango, el cliente informa del error y solicita al usuario que ingrese un número válido dentro del rango especificado. Este proceso se repite hasta obtener una entrada correcta.
3. **Viaje Pet Friendly**
    - **Cliente**: Verifica que la entrada para la opción de viaje 'pet friendly' sea correcta.
    - **Valores Aceptados**: 'Sí' o 'No'.
    - **Bucle de Corrección**: Si se introduce un valor distinto, el cliente notifica el error y solicita al usuario que ingrese uno de los valores aceptados. Este proceso se repite hasta que la entrada sea correcta.

Implementación Técnica

- **Loops de Validación**: Para cada tipo de entrada, el cliente emplea bucles que continúan solicitando la información hasta que se reciben datos válidos, mejorando la experiencia del usuario y evitando el procesamiento de datos incorrectos en el servidor.

Mecanismos de Concurrencia

Uso de Hilos

El servidor utiliza hilos para manejar múltiples conexiones de clientes de manera concurrente. Cada vez que un cliente se conecta al servidor, el método **handleClients** acepta la conexión y luego crea un nuevo hilo para gestionar la comunicación con ese cliente específico.

- Implementación: En el método **handleClients**, se llama a **socketClient.Accept()** para aceptar conexiones entrantes. Por cada conexión aceptada, se crea un nuevo hilo que ejecuta **HandlerGestor.Call**, pasando el socket del cliente y controladores para usuarios y viajes.

Aislamiento de Estados

Para evitar condiciones de carrera y asegurar la integridad de los datos, el servidor maneja estados específicos del cliente, como el usuario conectado, de manera aislada en cada hilo.

- Implementación: Se mantiene una instancia separada del usuario conectado (**loggedUser**) para cada hilo, permitiendo que las operaciones sean específicas para cada sesión de usuario sin interferencia entre hilos.

Control de Excepciones

El manejo de excepciones dentro del bucle de conexión del cliente asegura que cualquier error en la comunicación no afecte al servidor ni a otros clientes.

- Implementación: En el ciclo **while**, si ocurre una excepción durante la recepción o el procesamiento de un mensaje, se captura la excepción y se finaliza el bucle, desconectando adecuadamente al cliente.

Otras Decisiones

- **Menú Serve**r: Un menú en el servidor en el cual si se desea salir, se cierra la conexión y se desconectan todos los clientes conectados.
- **Identificador del Viaje**: En una primera instancia habíamos tomado como identificador del viaje la unión de fecha y hora de salida, origen y destino y conductor. Finalmente dado que seria mas sencillo la identificación del viaje, utilizando un identificador propiamente dicho, decidimos incluir uno el cual aumenta según cada viaje que se crea

Funcionalidad de la Aplicación

A continuación se detallan flujos de como un usuario usará el sistema

1. Inicio de Sesión:
   - Usuario Selecciona Iniciar Sesión: Se elige la opción "1" desde el menú inicial.
   - Ingreso de Datos: El usuario ingresa su nombre de usuario y contraseña.
   - Envío de Datos: La combinación de nombre de usuario y contraseña se envía al servidor.
   - Validación: El servidor valida los detalles y devuelve una respuesta.
   - Acceso Concedido o Denegado:
   - Si es exitoso, se accede al menú de opciones (***OptionsMenu***).
   - Si falla, se muestra un mensaje de error y se retorna al menú inicial.
2. Registro:
   - Usuario Selecciona Registrarse: Se elige la opción "2" desde el menú inicial.
   - Ingreso de Datos: El usuario proporciona los datos necesarios para el registro, como nombre de usuario, contraseña y otros datos relevantes.
   - Envío de Datos: Los datos se envían al servidor para crear una nueva cuenta.
   - Confirmación de Registro: El servidor procesa la información y confirma la creación de la cuenta.

- Redirección al Menú Inicial: Posterior al registro, el usuario es dirigido de nuevo al menú inicial para iniciar sesión.

3. Salir:
   - Usuario Selecciona Salir: Se elige la opción "3" desde el menú inicial.
   - Cierre de Conexión: Se envían comandos al servidor para cerrar la sesión y desconectar.
   - Confirmación de Cierre: El sistema confirma el cierre y finaliza la ejecución del cliente.

Post-Inicio de Sesión: Opciones del Menú

Una vez autenticado, el usuario tiene acceso a las siguientes opciones a través del ***OptionsMenu***:

- Publicar un Viaje:
  - Selección: Opción "3".
  - Proceso: Ingreso y envío de detalles del viaje al servidor, y confirmación.
- Unirse a un Viaje:
  - Selección: Opción "4".
  - Proceso: Visualización de viajes disponibles, selección de un viaje, y envío de comentarios y calificación.
- Modificar un Viaje:
  - Selección: Opción "5".
  - Proceso: Identificación del viaje a modificar, cambio de detalles, y confirmación.
- Dar de Baja un Viaje:
  - Selección: Opción "6".
  - Proceso: Selección del viaje a cancelar y confirmación de la baja.
- Buscar Viajes Disponibles:
  - Selección: Opción "7".
  - Proceso: Solicitud y visualización de viajes disponibles.
- Consultar Información de un Viaje Específico:
  - Selección: Opción "8".
  - Proceso: Solicitud y visualización de detalles de un viaje específico.
- Salir del Menú de Opciones:
  - Selección: Comando de salida.
  - Proceso: Cierre de sesión y finalización de la conexión con el servidor.

Anexos

Crear Viaje con registro y autenticación