OBJETIVOS

Poner en práctica los conocimientos adquiridos en la primera parte del curso.

- Manejo de paralelismo, concurrencia y sincronización de Threads.
- Manejo de Sockets.
- Manejo de Streams.
- Implementación y diseño de un protocolo propietario.

DESCRIPCIÓN DEL SISTEMA: SISTEMA TRIPORTUNITY

Se debe construir un sistema que permita compartir viajes en auto. Los usuarios pueden publicar los viajes que van a realizar y otros usuarios pueden unirse a los viajes publicados. El sistema debe contar con dos aplicaciones: servidor y cliente.

REQUERIMIENTOS

- Aplicación **Cliente**.
    - CR1. **Autenticación.** Se deberá poder iniciar sesión y cerrar sesión del sistema.
    - CR2. **Publicación de viaje.** Se deberá poder agregar un viaje al sistema. El usuario debe introducir todos los datos relacionados: Origen, destino, fecha y hora de salida, cantidad de asientos disponibles, precio por persona, si se permiten mascotas y una foto de su auto.
    - CR3. **Unirse a un viaje**. Se deberá poder unirse a un viaje que esté publicado y que tenga asientos disponibles. Una vez que un usuario se une a un viaje, la cantidad de asientos disponibles disminuye.
    - CR4. **Modificación de viaje.** El usuario que publicó el viaje debe poder modificar los siguientes datos del mismo: Origen, destino, fecha y hora de salida, precio, si se permiten mascota y la foto.
    - CR5. **Baja de viaje.** El usuario que publicó el viaje debe poder eliminarlo del sistema.
    - CR6. **Búsqueda de viaje.** Se deberá poder ver los viajes publicados en el sistema y además se podrá filtrar por al menos un criterio (ej. por destino, precio etc.). Criterios a definir por los alumnos.
    - CR7. **Consultar información de un viaje en específico.** El sistema deberá mostrar todos los datos de un viaje en específico: Origen, destino, fecha y hora de salida, cantidad de asientos disponibles, precio, si se permiten mascotas y permitir descargar la imagen asociada.

- CR8. **Calificar conductor.** Los usuarios que se unieron a un viaje pueden dejar una calificación del conductor, agregando un breve comentario acerca de la opinión del mismo y una puntuación que debe estar comprendida en un rango a definir por los alumnos.

- CR8. **Ver calificaciones de un conductor.** Al momento de ver la información de un viaje, los usuarios pueden ver las calificaciones del conductor.

- Aplicación **Servidor**.

  - SR1. El servidor debe contar con todas las funcionalidades para poder permitir a los clientes realizar los requerimientos anteriores.

  - SR2. **Aceptar pedidos de conexión de un usuario**. El servidor debe ser capaz de aceptar pedidos de conexión de varios clientes a la vez. Las acciones de un cliente no deben afectar considerablemente el rendimiento o la funcionalidad de los otros.

  - SR3. **Borrado**. Cuando los usuarios den de baja sus viajes los archivos asociados a las fotos deben ser borrados del servidor.

  - SR4. **Cierre**. El servidor debe tener una opción que permita cerrarlo, cerrando previamente las conexiones activas con los clientes.

- Requerimientos en común

  - **Configuración.** Las aplicaciones deberán poder correr en cualquier máquina sin necesidad de recompilar la solución, incluso en máquinas distintas. Cualquier valor que se necesite cambiar entre máquinas se debe poder cambiar sin necesidad de recompilar. Dichos valores no deben estar “hardcodeados” en el código.

  - **Desconexión**. Cualquier desconexión de clientes no debe generar fallas en el servidor u otros clientes ni errores en los datos.

  - **Manejo de errores:** Si sucede un error al momento de procesar una petición del cliente, el servidor debe informarle al cliente y seguir su funcionamiento normal.

Datos de prueba (opcional)
Pueden entregar el sistema con una fuente de datos de prueba propia o datos precargados para facilitar la corrección.

Compilados
Para facilitar la ejecución de la aplicación, deben incluir en el repositorio carpetas con las aplicaciones compiladas en *Release* y todo lo necesario para poder ejecutarlas (archivos de configuración, etc.), con ejecutables para Windows (*.exe*).

Opcionalmente pueden incluir los archivos Dockerfile y docker-compose.yml, junto a una guía, para ejecutar las aplicaciones utilizando Docker.

Especificación de Protocolo

1. El siguiente protocolo es una sugerencia, los estudiantes pueden (**y deberían**) hacer cambios al mismo.
2. Protocolo orientado a caracteres.
3. Implementado sobre TCP/IP.
4. Los valores deberán ir alineados a la derecha, los bytes de relleno deberán tener el valor 0.
5. Los campos HEADER, CMD y LARGO tendrán largo fijo. El campo DATOS tendrá largo variable, según el valor indicado en LARGO.
6. Formato general de la trama.

Nombre Del Campo	**HEADER**	**CMD**	**LARGO**	**DATOS**
Valores	RES/REQ	0-99	Entero	Variable
Largo	3	2	4	Variable

Documentación

Como documentación se espera que se incluya el alcance de la aplicación, errores conocidos (si los hay), supuestos de la letra que hayan realizado. una descripción de la arquitectura, así como el diseño detallado de cada uno de sus componentes.

Es muy importante (tanto como la aplicación) que la documentación explique las principales decisiones tomadas para construirla. Esas decisiones dependen de lo que cada trabajo persiga, pero a modo de ejemplo, sería bueno documentar por qué se tomaron las diferentes decisiones a nivel de protocolo, cómo se manejan los errores que se pueden presentar, cuáles son los mecanismos de concurrencia utilizados, etc.

Se debe documentar además el funcionamiento de la aplicación. **No es necesario** un manual de usuario, pero si aquellas aclaraciones que consideren necesarias para poder probar y utilizar la aplicación, así como para comprender su estructura.

No se espera ningún formato particular para la entrega. El orden, las técnicas y nomenclaturas a utilizar serán decisión de los alumnos. Se evaluará la correcta selección y uso de los mismos.

Se espera el adecuado uso de diagramas que complementen las explicaciones.

CÓDIGO

La gestión del código debe realizarse utilizando UN ÚNICO repositorio de **GitHub** y es obligatorio cumplir los siguientes 6 puntos:

1. El Repositorio debe **pertenecer a la organización de GitHub https://github.com/ArqSis-PDR-2024-1.**
2. El **nombre del repositorio** debe ser "GRUPO_CARRERA_#Estudiante1_#Estudiante2_#Estudiante3"
   Ej. : N6A_Licenciatura_123456_987654_123321
3. El repositorio debe ser **Privado**
4. Todos los miembros del equipo deben estar en el repositorio
5. El **link al repositorio** debe estar en la primera página de la **documentación**
6. **Al realizar la entrega se debe realizar un release en el repositorio y la rama master (o main) no debe ser afectada luego del hito que corresponde a la entrega del obligatorio. El release debe incluir el código fuente, los compilados y la documentación.**

ENTREGA

Se debe entregar por Gestión un archivo PDF que contenga la **documentación** y en la primera página el link al repositorio que cumpla los 6 puntos de la sección anterior.

El tamaño máximo del archivo a subir es de 40mb.

Importante: Solo se aceptará lo entregado en gestión y el código en el repositorio de la organización de GitHub, por lo que es de suma importancia revisar los medios y la documentación que se entrega antes de hacerlo.

EVALUACIÓN

El criterio para la corrección se basará en los siguientes aspectos:

- Deberán utilizar explícitamente la clase Thread y **no Task** para el desarrollo del obligatorio. Tampoco estará permitido el uso de binary formatter para el pasaje de información, se espera que se utilicen streams para hacer esto.
- En esta entrega NO se permitirá que se utilicen parsers existentes (como puede ser JSON) para serializar objetos.
- En la defensa se correrá un plan de testing, donde se evaluará cuantos puntos pasa el sistema.
- Criterios de Diseño.
- Porcentaje de cumplimiento con los requerimientos presentados.
- Funcionamiento de la aplicación.
- El cumplimiento de buenas prácticas de los temas trabajados en clase (Multi-threading, sockets, etc.)
- Calidad de diseño e implementación.
- Completitud y calidad de la documentación

DEFENSA

La defensa del trabajo intenta:

- Evaluar el conocimiento general de los integrantes del grupo sobre la solución propuesta. Todos los integrantes deben conocer toda la solución.
- Evaluar el aporte individual al trabajo por parte de cada uno de los integrantes del equipo. Se espera que como mínimo cada uno de los integrantes haya participado en la codificación de al menos alguna parte significativa del obligatorio.
- La defensa podrá ser presencial y que requiera ejecutar la aplicación en varias máquinas del laboratorio
- Las máquinas de los laboratorios utilizan Windows 10 por lo que es importante probar la aplicación antes de la defensa y tener los compilados en la entrega.
- Se coordinará el lugar y fecha de las defensas por los grupos de Teams de cada grupo.

Información importante

Defensas: a coordinar.

Plazo máximo de entrega: 25/4/2024 Puntaje mínimo/máximo: 5/25 puntos.

Los obligatorios se forman como máximo por grupos de hasta 3 (tres) estudiantes.

Todas las entregas se realizan en gestión (gestión.ort.edu.uy) y hasta las **21.00** hs del día de entrega.