Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Programación de Redes 1

Obligatorio 1

Renzo José Nro. Est. 240272

Nicolas Duarte Nro. Est. 268772

Repositorio: https://github.com/ArqSis-PDR-2024-2/obl-n6a-licenciatura-268772-165010-240272/tree/main

Grupo: N6A

Docentes: Andrés Soria y Guillermo Echichure

TABLA DE CONTENIDO

DESCRIPCIÓN DEL PROYECTO:

Presentamos "Sistema Vapor" o, en inglés, "Steam System", una aplicación diseñada para la compra y publicación de videojuegos. La aplicación está diseñada bajo una arquitectura cliente-servidor, brindando las funcionalidades esenciales que una plataforma de este tipo requiere. En su estado actual, "Sistema Vapor" permite que múltiples clientes se conecten simultáneamente al servidor, asegurando una gestión eficiente y estable de las conexiones, incluso a medida que aumenta la carga de usuarios.

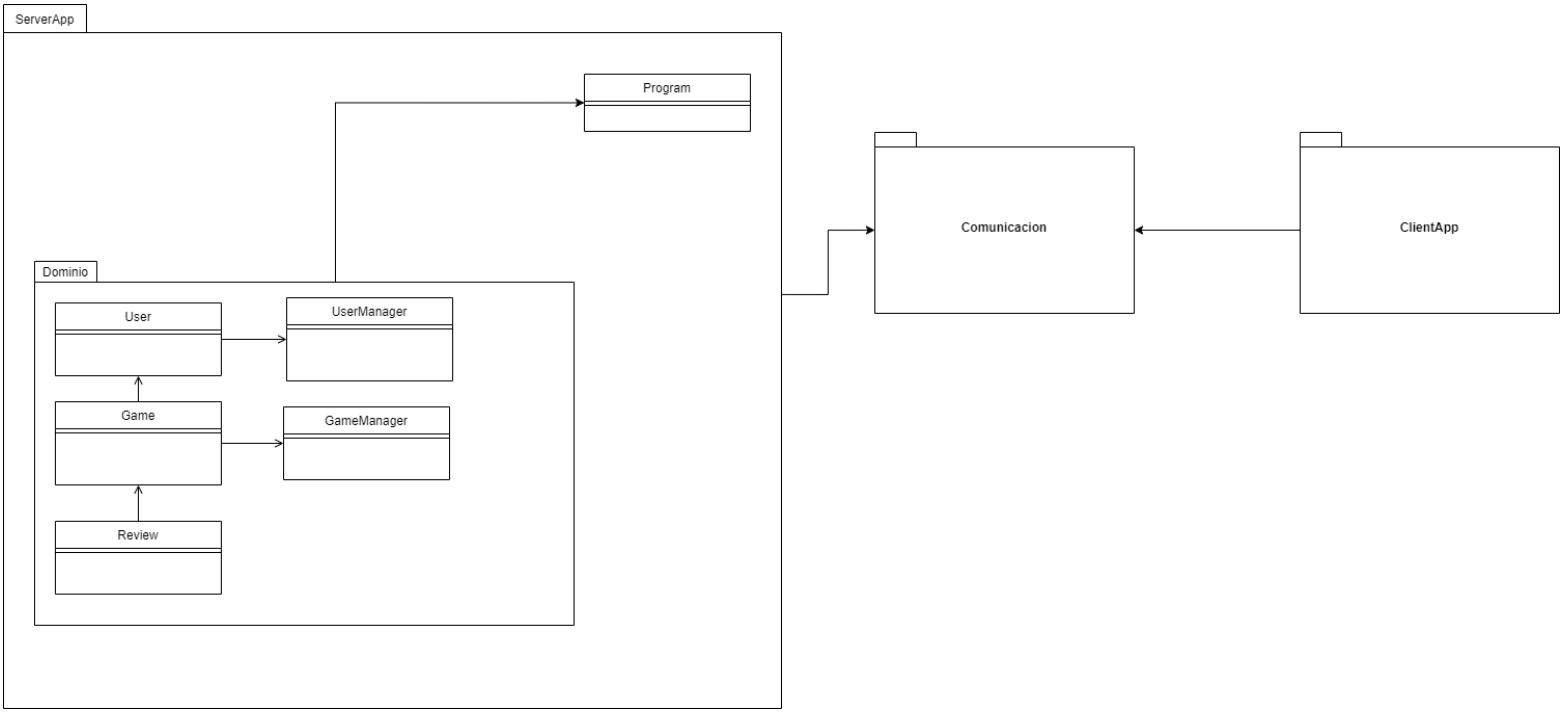
Para lograr esto, se ha implementado un sistema de manejo de sockets, permitiendo que cada nuevo cliente que se conecte al servidor disponga de un canal de comunicación dedicado. Esta arquitectura permite una comunicación sin interrupciones y asegura que el servidor pueda gestionar múltiples solicitudes de forma concurrente sin comprometer el rendimiento o la estabilidad del sistema.

Actualmente, la interfaz de usuario está basada en una línea de comandos, pero el diseño del sistema está planificado para facilitar futuras expansiones, como la incorporación de interfaces gráficas u otros métodos de interacción. La modularidad y flexibilidad de la arquitectura que diseñamos permitiría que estos cambios se implementen de manera sencilla y eficiente.

Entre las funcionalidades clave del sistema se encuentran la posibilidad de publicar, comprar, modificar, eliminar y buscar videojuegos dentro del catálogo. El servidor está diseñado para gestionar estas operaciones de manera concurrente, manteniendo altos estándares de seguridad, eficiencia y escalabilidad.

Decisiones de diseño:

Para facilitar el entendimiento del proyecto proporcionaremos los siguientes diagramas para visualizar las distintas funcionalidades bla.



El sistema implementa un dominio simple que incluye las clases User y Game, cada una con su correspondiente gestor (Manager). Estos gestores son responsables de administrar los métodos necesarios para interactuar con los datos, permitiendo operaciones como la creación, modificación y eliminación de usuarios y juegos. Actualmente, todos los datos se almacenan en memoria, pero el diseño está orientado hacia una futura persistencia en una base de datos. Este enfoque facilita la transición a un sistema de almacenamiento persistente sin necesidad de realizar cambios significativos en la lógica de la aplicación.

Dentro de las clases mencionadas, cada operación crítica se gestiona mediante el uso de mecanismos de bloqueo (Lock). Esto asegura que el acceso a las listas y la adición o modificación de elementos sean seguros en un entorno de ejecución multi-hilo. Gracias a este sistema de control de concurrencia, se garantiza que si dos usuarios intentan modificar simultáneamente el mismo recurso —por ejemplo, un videojuego en particular— no se generen inconsistencias ni errores en el sistema.

Por otro lado, van a haber dos aplicaciones, ClientApp y ServerAPP.

ClientAPP:

La aplicación cliente está diseñada para ser liviana y eficiente, con una lógica encapsulada en métodos privados que manejan la interacción con la interfaz de usuario y la comunicación con el servidor. El cliente no tiene acceso directo a las clases o la lógica interna del servidor, y la única forma de interacción se realiza mediante el envío de texto y archivos (actualmente, solo imágenes). Este diseño asegura que la aplicación mantenga un enfoque simple y rápido, con una comunicación clara y eficiente con el servidor.

Autenticación de Cliente:
Los usuarios podrán registrarse mediante la creación de una cuenta, proporcionando un nombre de usuario y contraseña. Asimismo, podrán iniciar sesión para acceder a su cuenta personal.

Publicación de Juegos:
Los usuarios tendrán la capacidad de agregar nuevos juegos al sistema. Cada publicación incluirá detalles como el título del juego, género, año de lanzamiento, plataforma, editor, unidades disponibles, precio y, opcionalmente, una imagen de la carátula del juego.

Compra de Juegos:
La aplicación permite a los usuarios adquirir juegos, siempre y cuando haya unidades disponibles en el inventario. Cada vez que se realiza una compra, el sistema reduce la cantidad de unidades disponibles de ese juego de manera automática.

Modificación de Juegos:
Los usuarios podrán modificar la información de los juegos que hayan publicado. Esto incluye la posibilidad de cambiar el título, género, año de lanzamiento, plataforma, editor, unidades disponibles, precio e imagen de carátula.

Eliminación de Juegos:
Los usuarios que hayan publicado un juego tendrán la opción de eliminarlo del sistema, removiendo tanto su información como la posibilidad de compra.

Búsqueda de Juegos:
El sistema cuenta con un motor de búsqueda que permite a los usuarios explorar los juegos disponibles. Los resultados pueden filtrarse según criterios específicos como género, plataforma o valoración por parte de otros usuarios.

Consulta de Juegos Específicos:
Los usuarios pueden acceder a la información detallada de un juego específico, incluyendo la opción de descargar la imagen de la carátula y revisar las calificaciones recibidas por parte de otros usuarios.

Calificación de Juegos:
Los usuarios que hayan adquirido un juego pueden dejar una calificación, proporcionando una breve reseña y asignando una puntuación en una escala del 1 al 10. Este sistema de calificación permite compartir opiniones y mejorar la experiencia de otros compradores.

ServerApp:

La aplicación del servidor se encargará de hacer toda la gestión de manejar a los clientes en distintos hilos permitiendo así de esta manera aprovechar el multi-hilo y gestionar solicitudes de conexión de múltiples clientes simultáneamente, asegurando que las acciones de un cliente no impacten significativamente el rendimiento de los demás.

Cada nuevo usuario es interpretado como un nuevo Socket que este mismo enviara, se delegara a una nueva Task y de esta manera generaremos el multihilo.

En si esta aplicación es muy similar a la de los clientes en cuanto a diseño y manejabilidad, se encarga de proporcionar todas las funcionalidades requeridas para que los clientes puedan llevar a cabo las acciones previamente descritas y al ser la encargada de todo está relacionada a las clases auxiliares para manejar bien los hilos.

Cuando los usuarios dan de baja sus juegos, las imágenes asociadas a las carátulas son borradas del servidor.

El servidor es capaz de apagarse de manera controlada, cerrando previamente las conexiones activas con los clientes.

Supuestos:

Usuario:
- Los username y password deben ser de al menos 4 caracteres.

Juegos:
- Los juegos tienen un único título en todo el sistema y es la clave que lo diferencian de los demás.
- A la hora de publicar un juego se setea por default con una valoración en “0”.
- Se agrega a la hora de publicar un juego el precio de este. Este se ingresa únicamente en unidades enteras, por lo que no puede contener comas ni decimales. Es decir, el precio debe ser considerado como si estuviera en pesos uruguayos y no en dólares, por ejemplo.
- Los usuarios pueden hacer varias valorizaciones de un mismo juego.
- Las imágenes se guardarán con el mismo nombre del juego independientemente del nombre del archivo original y todas como jpg.
- Las reviews que tengan menos de 5 caracteres o sean vacías se guardaran con "No review".

Aunque actualmente los datos se encuentran persistidos en memoria, durante la definición de la solución creamos clases de acceso a datos (DataAccess), como GameRepository y UserRepository, así como sus correspondientes servicios (services). Esto se hizo pensando en una futura transición hacia una solución que persista los datos en una base de datos, lo cual facilitará la escalabilidad y adaptabilidad del sistema.

Mecanismos de Concurrencia:

- Manejo de multithreading para atender múltiples clientes: del lado del servidor se expone un único puerto y cada vez que se acepta una nueva conexión, se inicia un nuevo hilo para manejar las solicitudes del cliente específico. Para esto utilizamos la clase Thread.

- Para la entrada de cada usuario se utiliza un Thread adicional y dedicado para la escucha de los comandos. Esto lo hacemos para no bloquear el hilo principal que aceptan los clientes.

- Locks en recursos identificados como compartidos: Esto nos previene ciertas condiciones de carrera que podrían llegar a presentarse en un contexto de concurrencia.

Protocolo:

El protocolo de comunicación entre el cliente y el servidor ha sido diseñado para asegurar que los mensajes enviados se interpreten correctamente y de manera eficiente. La estructura del protocolo consta de dos campos principales: **Largo del mensaje** y **Mensaje**.

Nombre Del Campo	**Largo**	**Mensaje**
Valores	Entero	String
Largo	4	Variable
Descripción	Se envía primero el largo del mensaje que se va a enviar a continuación para poder parsear correctamente.	Contiene el mensaje a enviar, cuyo tamaño está definido por el campo anterior.

El largo del mensaje es un entero de 4 bytes que define el tamaño, en bytes, del mensaje que será enviado posteriormente. Este enfoque permite al receptor saber exactamente cuántos bytes debe leer para interpretar el mensaje correctamente. Actúa como un prefijo obligatorio en cada transacción de datos, garantizando que el servidor o el cliente pueda gestionar mensajes de longitud variable de manera controlada.

Tras el envío del "Largo", se transmite el contenido del mensaje en sí. Este mensaje es una cadena de texto (o cualquier otro tipo de dato, dependiendo del caso), cuya longitud ha sido especificada en el campo anterior.

El protocolo de mensajería de esta manera mantiene un simplicidad y flexibilidad a la hora de utilizarlo muy amplia ya que se pueden agregar modificaciones del código sin necesidad de estar viendo el protocolo de comunicación. A su vez tanto el cliente como el servidor utilizan el mismo formato de mensajes, lo que asegura una comunicación fluida y eficiente.

Por otro lado, para el envío de archivos se define otro mecanismo utilizando la clase System.IO.FileStream. Esta nos permite ir leyendo un archivo por partes de un tamaño definido, enviarlas a través de la red en un flujo continuo de bytes e ir escribiendo en disco en el destino por medio de un buffer. El protocolo de envío de archivos sigue el siguiente formato:

Nombre del campo	**Largo**	**Nombre de archivo**	**Tamaño del archivo**	**Archivo o parte de archivo**
Valores	Entero	String	Entero	Bytes
Largo	4	Variable	8	Hasta 32 KB
Descripción	Representa el largo del nombre del archivo a enviar.	Nombre del archive.	Tamaño total del archivo en Bytes.	Bytes del archivo que se envían por fracciones de hasta 32 KB.

Errores conocidos:

En cuanto a la funcionalidad de modificar un juego específico, al intentar cambiar la imagen de carátula, el sistema arrojaba una excepción en la clase NetworkDataHelper, relacionada con un valor negativo en el tamaño del archivo subido. La lógica para esta operación ya está implementada, y para evitar eliminarla por completo, decidimos dejarla comentada temporalmente, de modo que podamos evitar esta excepción sin perder el trabajo desarrollado.

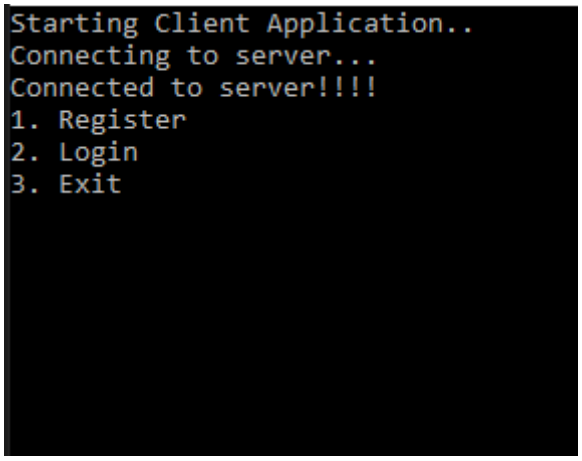
Funcionamiento de la aplicación:

Tanto en la aplicación cliente como servidor se definieron archivos de configuración (App.config) los cuales son leídos en el inicio de la aplicación.

Para iniciar la aplicación simplemente se deben configurar los parámetros de configuración dentro del App.config, ejecutar en primer lugar la aplicación servidor y luego el cliente debe loguearse con un usuario válido.

A continuación, proporcionamos un ejemplo de inicio de sesión de usuario, seguido de un flujo simple para la publicación de un juego y la compra de este:

Una vez que iniciamos la aplicación obtendremos un menú de inicio como el siguiente:



En este caso tendremos que escribir el número dos por consola y darle al enter para acceder al “login”, introduciendo como credenciales los datos de un user ya precargado en el sistema con los datos: username= ‘admin’ y password=’admin’

```
Starting Client Application..
Connecting to server...
Connected to server!!!!
1. Register
2. Login
3. Exit
2
Server says: Option '2' received successfully
Enter username: admin
Enter password: admin
Server says: Login successful
```

Una vez dentro ya obtendremos el menú desplegable que se muestra cuando el usuario ya tiene la sesión iniciada y podremos escribir la opción que deseemos, en este caso iremos por la opción cinco la cual nos permite publicar un juego.

```
1. Register
2. Login
3. Exit
2
Server says: Option '2' received successfully
Enter username: admin
Enter password: admin
Server says: Login successful
1. Search Games
2. More information about a game
3. Buy GAMES
4. Review Game
5. Publish a game
6. Modify Published Game
7. Delete Published Game
8. Logout
5
Server says: Option '5' received successfully
Enter the game title:Mario
Server says: Succesful New Game Name
Enter the game's genre:aventura
Enter the release date (dd/mm/yyyy):15/10/2000
Enter the platform:Pc
Enter the number of units available:50
Enter the price:900
Do you want to upload a cover image? (yes/no)no
Server says: no

Game published successfully.
```

Una vez ingresados los datos de forma correcta, el juego es publicado exitosamente.

Ingresamos la opción tres ahora para poder comprar el juego una vez que fue publicado.

```
1. Search Games
2. More information about a game
3. Buy GAMES
4. Review Game
5. Publish a game
6. Modify Published Game
7. Delete Published Game
8. Logout
3
Server says: Option '3' received successfully
Titulo del juego a comprar:
Mario
Server says: Game purchased successfully
```