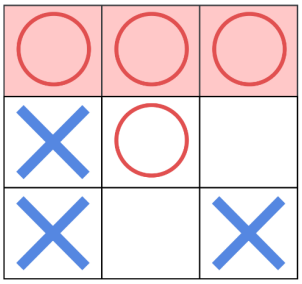
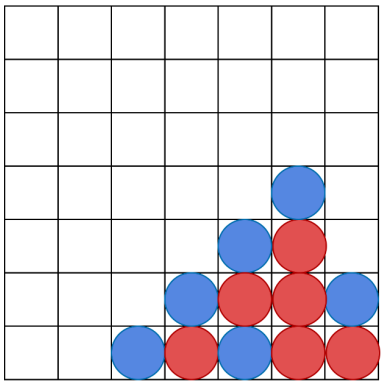
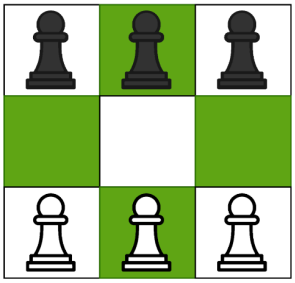
Capstone Project: Network Multiplayer Game



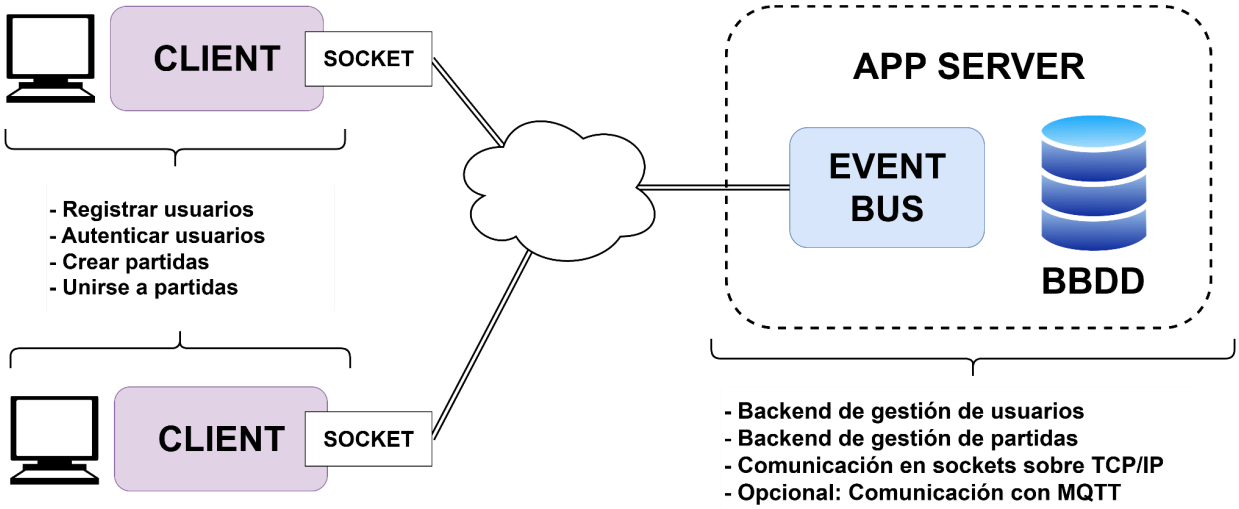




Objetivo del Proyecto

El objetivo de este proyecto es diseñar, desarrollar e implementar una aplicación de juego para dos jugadores (2P game), conectados simultáneamente por red. La aplicación debe garantizar el estado coherente del juego para ambos usuarios.

El alumno podrá elegir e implementar **1 (un)** juego a su gusto. Por ejemplo: tic-tac-toe, 4 en línea, hexapawn, etc. Si se te ocurre algún otro juego que te gustaría implementar, consultá con tu practitioner!



Requerimientos Base

1. La aplicación deberá permitir el registro de varios usuarios mediante *username* y *password*, persistiendo los datos de registro de manera segura.
2. La aplicación deberá autenticar a sus usuarios sin exponer los datos de registro.
3. La aplicación deberá ser capaz de mostrar los usuarios actualmente en línea.
4. Los usuarios deberán ser capaces de crear una nueva partida e invitar a cualquiera de los usuarios en línea.
5. Los usuarios deben poder aceptar o rechazar una invitación a una nueva partida.
6. Los usuarios deberán ser capaces de actualizar el estado del tablero según sea su turno (realizar una jugada). El tablero deberá ser actualizado para ambos jugadores.
7. La aplicación deberá mostrar el resultado final del juego (ganador, perdedor, empate) a ambos usuarios, cuando el mismo haya finalizado.
8. La aplicación deberá permitir a un jugador abandonar el juego y otorgar la victoria al contrincante.
9. La aplicación deberá monitorear los eventos de ejecución del servidor: logging de jugadores, partidas (comienzo/fin), errores, etc. Deseable: persistir esta información en la base de datos.

Requerimientos opcionales (nice-to-have)

10. Scoreboard: La aplicación podría tener la capacidad de mantener un ranking público de jugadores según cantidad de partidas ganadas y empates.
11. Bot player: La aplicación podría tener la capacidad de crear un jugador virtual (*player vs. computer*), gestionado desde el servidor.
12. Simulación de fallas y recuperación: diseñar la aplicación de tal modo que, en el evento de la desconexión de un jugador, el mismo pueda reconectarse a una partida ya comenzada.
13. Server dashboard: diseñar una interfaz de administrador que resuma los datos más importantes del estado del servidor en tiempo real: usuarios conectados, juegos activos, estadísticas de conexión, errores, etc.
14. Despliegue en la nube: desplegar tu aplicación en la nube y probarla con tus amigos!

Resultados Esperados

Al completar este proyecto, los estudiantes deberán haber logrado:

1. **Desarrollo de una Aplicación Completa:** Implementar una aplicación funcional de un juego en línea para dos usuarios simultáneos, que permita la actualización del estado del juego para ambos jugadores en tiempo real (RT). La aplicación debe ser testeada end-to-end.
2. **Implementación de Conceptos de Seguridad en Redes:** Identificar vulnerabilidades de comunicación del sistema e integrar protocolos de seguridad y encriptado (TLS, SSL, SHA-64, etc. deseable) para garantizar la confidencialidad de las partidas y la autenticación de los usuarios.
3. **Arquitectura Cliente-Servidor:** Comprender y aplicar la arquitectura cliente-servidor, manejando la gestión de usuarios, la creación de nuevas partidas de juego y el

enrutamiento de mensajes de acuerdo a cada partida, se pueden crear tantos usuarios como partidas de juego sean necesarias (no hay límite).

4. **Arquitectura orientada a eventos:** El servidor manejara la queue de acciones y jugadas por turno utilizando los conceptos de *event-bus* vistos durante el curso.
5. **Programación de Sockets:** Entender el uso de sockets en el contexto del modelo OSI y comunicar datos entre el cliente-servidor y cliente-cliente, asegurando la emisión y recepción de mensajes.
6. **Interfaz de Usuario Simple:** Diseñar una interfaz de usuario sencilla y efectiva para poder visualizar los clientes conectados, crear una partida, jugar una partida y ver el ranking de los jugadores.

Herramientas Necesarias

- Utilizaremos preferentemente .Net, C# o Java, seleccionando librerías bien documentadas, robustas y de amplio uso. Como alternativas se podrá utilizar Python o Javascript, si la elección es bien justificada.
- Se utilizará un sistema de control de versiones (GIT) para la gestión del código.
- Monitor de tráfico como Wireshark para monitorear el tráfico de red de la aplicación.

Pasos del Proyecto

1. **Fase de Diseño:**
    - Definir los requerimientos funcionales y no funcionales de la aplicación.
    - Diseñar la arquitectura cliente-servidor.
    - Planificar las políticas de seguridad y enrutamiento de los mensajes.
2. **Fase de Implementación:**
    - Desarrollar el servidor que maneje la autenticación de usuarios, la creación de nuevas partidas, y el enrutamiento de mensajes.
    - Implementar la aplicación cliente que permita a los usuarios registrarse, crear un nuevo juego o aceptar la invitación a unirse a una partida.
    - Integrar los protocolos de seguridad para asegurar la comunicación.
3. **Fase de Pruebas:**
    - Diseñar los test funcionales que se utilizarán para probar la aplicación.
    - Realizar pruebas funcionales de la aplicación jugando una partida con todos los posibles resultados.
    - Verificar la seguridad de la aplicación utilizando herramientas como Wireshark para asegurar que los mensajes son íntegros y confidenciales.
    - [Opcional] Realizar pruebas de intrusión/DDoS/etc. para validar la robustez de la aplicación contra ataques comunes (Man in the Middle).
4. **Documentación y Presentación:**
    - Documentar el código y el proceso de desarrollo.
    - Preparar una presentación final para mostrar el funcionamiento de la aplicación y los conceptos aprendidos.

Criterios de Evaluación

1. **Funcionalidad Completa de la Aplicación:** La aplicación debe permitir a los usuarios registrarse, crear nuevas partidas, unirse a una partida creada o aceptar invitación a unirse a una partida.
2. **Implementación Efectiva de Seguridad:** La aplicación debe demostrar una comunicación segura y estar protegida contra ataques comunes.
3. **Calidad del Código:** El código debe estar bien documentado, estructurado y seguir buenas prácticas de programación.
4. **Presentación Final:** Los estudiantes deben ser capaces de mostrar y explicar claramente el funcionamiento de la aplicación y los conceptos de seguridad implementados.