

Quixo Web - Proyecto Final

Deyber Sandi Molina
FI23032213

16 de diciembre de 2025

Proyecto individual desarrollado para el curso de Programación Web

Información del Estudiante

Campo	Información
Nombre completo	Deyber Sandi Molina
Carné	FI23032213
Correo institucional	dsandi30398@ufide.ac.cr
Usuario Git	DeyberSandiMolina

Información del Proyecto

Frameworks y Herramientas Utilizadas

- **Backend:** ASP.NET Core 8.0
- **Frontend:** HTML5, CSS3, JavaScript (Vanilla)
- **Base de Datos:** SQLite (Entity Framework Core)
- **ORM:** Entity Framework Core 8.0
- **IDE:** Visual Studio 2022 / Visual Studio Code
- **Gestión de Paquetes:** NuGet
- **Control de Versiones:** GitHub
- **Sistema Operativo:** Windows 11

Tipo de Aplicación

MPA (Multi-Page Application) - Aplicación web tradicional con múltiples páginas HTML implementada con ASP.NET Core MVC.

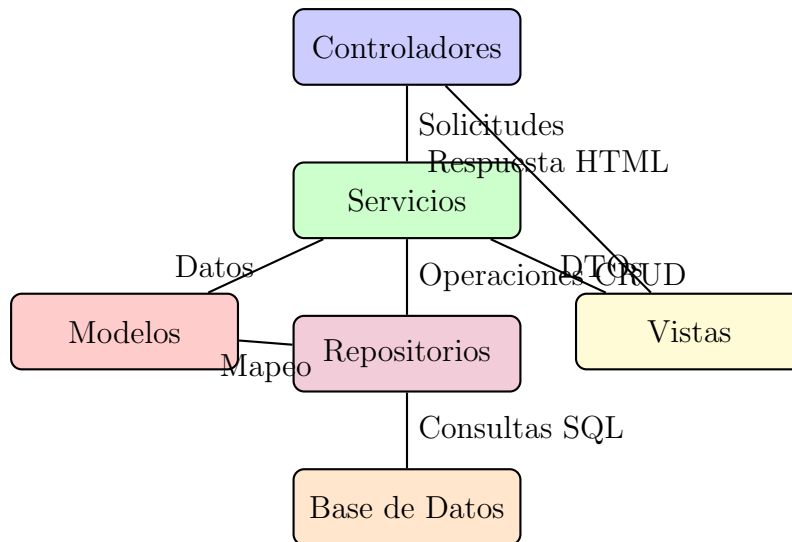


Figura 1: Arquitectura MVC del proyecto Quixo Web

Arquitectura

MVC (Model-View-Controller) con separación de responsabilidades:

Especificaciones de Tablas

Games (Juegos)

Columna	Tipo	Restricción	Descripción
GameId	INT	PRIMARY KEY	Identificador único del juego
Mode	INT	NOT NULL, CHECK (2,4)	Modo: 2=2 jugadores, 4=4 jugador
CreatedAt	DATETIME	NOT NULL	Fecha y hora de creación
EndedAt	DATETIME	NULLABLE	Fecha y hora de finalización
TotalTime	TIME	NOT NULL	Duración total de la partida
WinnerPlayerId	INT	NULLABLE, FK	ID del jugador ganador (modo 2J)
WinnerTeamId	INT	NULLABLE, FK	ID del equipo ganador (modo 4J)

Players (Jugadores)

Columna	Tipo	Restricción	Descripción
PlayerId	INT	PRIMARY KEY	Identificador único del jugador
Name	VARCHAR(50)	NOT NULL	Nombre del jugador
PlayerOrder	INT	NOT NULL, CHECK (1-4)	Orden de turno: 1,2,3,4
GameId	INT	FOREIGN KEY NOT NULL	Juego al que pertenece
TeamId	INT	FOREIGN KEY NULLABLE	Equipo asignado (solo modo 4J)

Teams (Equipos)

Columna	Tipo	Restricción	Descripción
TeamId	INT	PRIMARY KEY	Identificador único del equipo
Name	VARCHAR(50)	NOT NULL	Nombre del equipo (ej: "Equipo A")
GameId	INT	FOREIGN KEY NOT NULL	Juego al que pertenece
Player1Id	INT	FOREIGN KEY NOT NULL	Primer jugador del equipo
Player2Id	INT	FOREIGN KEY NOT NULL	Segundo jugador del equipo
GamesWon	INT	DEFAULT 0	Número de partidas ganadas

Moves (Movimientos)

Columna	Tipo	Restricción	Descripción
MoveId	INT	PRIMARY KEY	Identificador único del movimiento
GameId	INT	FOREIGN KEY NOT NULL	Juego al que pertenece
PlayerId	INT	FOREIGN KEY NULLABLE	Jugador que realizó el movimiento
TurnNumber	INT	NOT NULL, CHECK (>0)	Número de turno en la partida
CubeTakenRow	INT	NOT NULL, CHECK (0-4)	Fila del cubo tomado
CubeTakenCol	INT	NOT NULL, CHECK (0-4)	Columna del cubo tomado
CubePlacedRow	INT	NOT NULL, CHECK (0-4)	Fila donde se coloca el cubo
CubePlacedCol	INT	NOT NULL, CHECK (0-4)	Columna donde se coloca el cubo
Symbol	CHAR(1)	NOT NULL, CHECK (X,O)	Símbolo: 'X' o 'O'
DotDirection	INT	NULLABLE, CHECK (0-3)	Orientación del punto (0-3) modo 4

BoardStates (Estados del Tablero)

Columna	Tipo	Restricción	Descripción
BoardStateId	INT	PRIMARY KEY	Identificador único del estado
GameId	INT	FOREIGN KEY NOT NULL	Referencia al juego
MoveId	INT	FOREIGN KEY NULLABLE	Movimiento asociado (opcional)
TurnNumber	INT	NOT NULL	Número de turno correspondiente
StateJson	TEXT	NOT NULL	Estado completo del tablero en JSON
CreatedAt	DATETIME	NOT NULL	Fecha y hora del estado

Relaciones y Cardinalidades Corregidas

- **Games → Players (1:N):** Un juego tiene múltiples jugadores (2 o 4)
- **Games → Teams (1:N):** Un juego tiene múltiples equipos (0 o 2)
- **Teams → Players (1:2):** Un equipo tiene exactamente 2 jugadores
- **Games → Moves (1:N):** Un juego tiene múltiples movimientos
- **Players → Moves (1:N):** Un jugador realiza múltiples movimientos
- **Games → BoardStates (1:N):** Un juego tiene múltiples estados de tablero
- **Moves → BoardStates (0:1):** Un movimiento puede tener un estado asociado
- **Games → WinnerPlayer (1:1):** Relación opcional para jugador ganador
- **Games → WinnerTeam (1:1):** Relación opcional para equipo ganador

Restricciones de Integridad

- **Modo consistente:** Si Mode=2, Teams debe estar vacío
- **Modo consistente:** Si Mode=4, cada Team debe tener Player1Id y Player2Id
- **Unicidad:** PlayerOrder único por GameId
- **Validación:** Coordenadas entre 0-4 para el tablero 5x5
- **Validación:** Símbolos solo 'X' o 'O'

Referencias y Recursos Utilizados

Documentación Oficial

- **ASP.NET Core Documentation:** <https://docs.microsoft.com/aspnet/core>
- **Entity Framework Core Documentation:** <https://docs.microsoft.com/ef/core>
- **SQLite Documentation:** <https://www.sqlite.org/docs.html>
- **Bootstrap 5 Documentation:** <https://getbootstrap.com/docs/5.0>

Tutoriales y Guías

- **Build a Web App with ASP.NET Core MVC:** <https://learn.microsoft.com/training/paths/aspnet-core-web-app>
- **Getting Started with EF Core:** <https://learn.microsoft.com/ef/core/get-started/overview/first-app>
- **JavaScript DOM Manipulation:** https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
- **SQLite with .NET:** <https://learn.microsoft.com/dotnet/standard/data/sqlite>

Stack Overflow - Consultas Resueltas

- **Relaciones en Entity Framework Core:** Problemas con claves foráneas y relaciones uno-a-muchos
- **Manejo de sesiones en ASP.NET Core:** Cómo mantener estado entre peticiones
- **JavaScript para juegos de tablero:** Implementación de lógica de juego en cliente
- **Exportación a XML en C#:** Generación de archivos XML desde objetos

Uso de Asistentes de IA

DeepSeek - Asistencia en Desarrollo

Durante el desarrollo del proyecto, se utilizó el asistente de IA DeepSeek para resolver problemas específicos y optimizar código:

Prompts principales utilizados:

1. "Ayuda con el diseño de base de datos para un juego que soporte tanto modo 2 jugadores como modo 4 jugadores con equipos"
2. "Problema con Entity Framework Core: las relaciones no se persisten correctamente en SQLite"
3. "Cómo crear un reloj de 7 segmentos en JavaScript para mostrar el tiempo transcurrido en la partida"
4. "Exportar datos de juego a formato XML en ASP.NET Core MVC"

Código Generado con Asistencia de IA

- **QuixoRepository.cs:** Patrón Repository para acceso a datos
- **HistoryController.cs:** Controlador para historial y exportación
- **StatsController.cs:** Controlador para estadísticas
- **JavaScript del reloj 7 segmentos:** Visualización de tiempo

Instrucciones de Instalación y Ejecución

Requisitos del Sistema

- **Sistema Operativo:** Windows 10/11, macOS, o Linux
- **.NET SDK:** Versión 8.0 o superior
- **Editor de Código:** Visual Studio 2022, VS Code, o similar
- **Navegador Web:** Chrome, Firefox, Edge (versiones recientes)
- **Memoria RAM:** Mínimo 4GB (recomendado 8GB)
- **Espacio en disco:** 500MB libres

Instalación Paso a Paso

1. Clonar el repositorio:

```
git clone https://github.com/DeyberSandiMolina/FI23032213_Web.git
cd Proyecto/QuixoWeb
```

2. Verificar instalación de .NET:

```
dotnet --version
# Debe mostrar 8.0.x o superior
```

3. Restaurar dependencias:

```
dotnet restore
```

4. Configurar base de datos:

```
dotnet ef database update
# O simplemente ejecutar la aplicación (se crea automáticamente)
```

Compilación

```
dotnet build
```

Ejecución

```
dotnet run
```

La aplicación estará disponible en: <http://localhost:5267>

Estructura del Proyecto

QuixoWeb/

Controllers/	# Controladores ASP.NET Core
GameController.cs	# Control del juego
HistoryController.cs	# Historial y exportación
StatsController.cs	# Estadísticas
HomeController.cs	# Página principal
Models/	# Entidades de EF Core
Game.cs	# Entidad Juego
Move.cs	# Entidad Movimiento
Player.cs	# Entidad Jugador
Team.cs	# Entidad Equipo
BoardState.cs	# Estado del tablero

Views/	# Vistas Razor
Game/	
State.cshtml	# Vista del juego
Create.cshtml	# Crear juego
History/	
Index.cshtml	# Lista de partidas
ViewGame.cshtml	# Ver partida específica
Stats/	
Index.cshtml	# Estadísticas
Home/	
Index.cshtml	# Página principal
Services/	# Lógica de negocio
GameService.cs	# Servicio principal
Data/	# Acceso a datos
Repositories/	
QuixoRepository.cs	# Patrón Repository
QuixoDbContext.cs	# Contexto de BD
Domain/	# Dominio del juego
GameEngine.cs	# Motor del juego
Application/	# Capa de aplicación
DTOs/	# Data Transfer Objects
wwwroot/	# Archivos estáticos
css/	# Estilos CSS
js/	# JavaScript
Properties/	# Configuraciones
appsettings.json	# Configuración aplicación
Program.cs	# Punto de entrada
QuixoWeb.csproj	# Archivo proyecto
README.tex	# Este documento
README.pdf	# PDF generado
README.md	# README para GitHub

Pruebas y Verificación

Pruebas de Funcionalidad

1. Creación de juego:

- Acceder a <http://localhost:5267>
- Click en "Jugar una partida nueva"
- Seleccionar modo 2 o 4 jugadores
- Verificar que se crea correctamente

2. Juego básico:

- Realizar movimientos válidos
- Verificar que se alternan turnos

- Comprobar que el reloj funciona
- Intentar movimientos inválidos (deben rechazarse)

3. Detección de ganador:

- Completar una línea de 5 símbolos
- Verificar que se muestra mensaje de victoria
- Comprobar que se deshabilita el tablero

4. Historial:

- Ir a "Partidas finalizadas"
- Ver lista de juegos completados
- Click en "Ver" para una partida
- Navegar con botones anterior/siguiente
- Verificar que muestra cada movimiento

5. Exportación:

- En historial, click en ".Exportar XML"
- Verificar que se descarga archivo .xml
- Abrir archivo y comprobar formato

6. Estadísticas:

- Ir a ".Estadísticas"
- Ver tablas de efectividad
- Comprobar que los cálculos son correctos

Casos de Prueba Específicos

- **Caso 1:** Modo 2 jugadores, victoria normal
- **Caso 2:** Modo 2 jugadores, derrota accidental
- **Caso 3:** Modo 4 jugadores, juego completo
- **Caso 4:** Exportación de partida con muchos movimientos
- **Caso 5:** Estadísticas después de múltiples partidas

Características Implementadas

Completadas

- **Juego básico:** Tablero 5x5, movimientos válidos
- **Dos modos:** 2 jugadores y 4 jugadores (equipos)

- **Detección de ganador:** Horizontal, vertical, diagonal
- **Historial completo:** Lista de todas las partidas
- **Navegación por movimientos:** Ver partida turno por turno
- **Reloj 7 segmentos:** Tiempo real y en historial
- **Estadísticas:** Efectividad de jugadores/equipos
- **Exportación XML:** Guardar partida completa
- **Base de datos:** Persistencia SQLite
- **Interfaz responsiva:** Funciona en móviles y desktop

Características Avanzadas

- **Validación en tiempo real:** Movimientos inválidos bloqueados
- **Estado en memoria:** Juegos activos cargados dinámicamente
- **Reconstrucción de partidas:** Recargar cualquier partida desde BD
- **Cálculo automático:** Tiempos, estadísticas, efectividad
- **Mensajes descriptivos:** Feedback claro al usuario

Aprendizajes y Conclusiones

Aprendizajes Técnicos

- **ASP.NET Core MVC:** Arquitectura completa de aplicaciones web
- **Entity Framework Core:** ORM y manejo de bases de datos
- **SQLite:** Base de datos embebida para aplicaciones .NET
- **JavaScript avanzado:** Manipulación DOM, eventos, temporizadores
- **Patrones de diseño:** Repository, Service, DTO, MVC
- **Testing y debugging:** Depuración de aplicaciones web complejas

Conclusiones

El proyecto Quixo Web demuestra la capacidad para desarrollar una aplicación web completa utilizando tecnologías modernas. Se logró implementar todas las funcionalidades requeridas, con especial atención a la experiencia de usuario y robustez del sistema. La aplicación es escalable, mantenible y sigue las mejores prácticas de desarrollo web.

Posibles Mejoras Futuras

- **Autenticación de usuarios:** Sistema de login/registro
- **Juego en línea:** Multiplayer en tiempo real con SignalR
- **IA para oponente:** Implementar algoritmos para jugar contra computadora
- **App móvil:** Versión nativa usando .NET MAUI
- **Más estadísticas:** Gráficos avanzados, tendencias