Actividad 6 - Entrega final del proyecto

Presentado por: Miguel Ángel Herrera Vargas
Presentado a: Tatiana Cabrera

CORPORACIÓN UNIVERSITARIA IBEROAMERICANA
Proyecto de Software
2025

Contenido

Contextualización de la necesidad

En muchas comunidades urbanas, las canchas sintéticas han dejado de ser simples espacios deportivos para convertirse en escenarios de encuentro social, recreación y construcción de lazos vecinales. Sin embargo, el uso de estos espacios suele estar marcado por la falta de organización y reglas claras, lo que genera tensiones entre los habitantes. La necesidad de una herramienta que permita administrar las reservas de manera justa no responde únicamente a un tema logístico, sino también a un aspecto social: garantizar que el deporte y la convivencia se desarrollen en condiciones de equidad, inclusión y respeto. Esta situación abre la puerta a la creación de un prototipo tecnológico pensado desde las realidades comunitarias, que permita mejorar el acceso a un recurso compartido de gran valor.

Planteamiento del problema

La existencia de canchas sintéticas en los barrios no garantiza que estas se utilicen de forma adecuada o equitativa. Por el contrario, la ausencia de un sistema transparente para gestionar reservas provoca conflictos frecuentes, favoritismo hacia ciertos grupos y exclusión de sectores como mujeres, adultos mayores o jóvenes sin afiliación a clubes. Esto no solo limita el acceso al deporte, sino que afecta directamente la convivencia comunitaria. Aunque en contextos privados existen sistemas digitales de gestión, estas soluciones no responden a las condiciones de las comunidades, donde predominan la autogestión y la organización local. Ante esta situación surge la necesidad de diseñar una herramienta accesible, simple y participativa que permita organizar de manera justa el uso de las canchas, fortaleciendo la cohesión social y evitando conflictos.

Objetivos

El propósito general de este proyecto es diseñar un prototipo funcional que permita gestionar las reservas de canchas sintéticas en comunidades urbanas de manera equitativa y sostenible. La idea es que la tecnología no solo resuelva un problema de organización, sino que también aporte a la construcción de confianza entre los vecinos y al fortalecimiento del sentido de pertenencia hacia los espacios públicos.

Objetivos específicos

1. Analizar, junto con los actores locales, las necesidades y dinámicas que surgen en torno al uso de las canchas comunitarias.
2. Formular los requerimientos funcionales y no funcionales de un sistema de reservas, garantizando que sean coherentes con la realidad de los usuarios.
3. Diseñar un prototipo navegable, accesible y adaptable que sirva como base para la validación comunitaria y como insumo para futuros desarrollos tecnológicos.

Alcance

El proyecto se centrará en el diseño de un prototipo funcional navegable que permita gestionar las reservas de canchas sintéticas en un entorno comunitario específico. El alcance no contempla un producto final en producción, sino una versión inicial que evidencie los flujos principales de reserva, consulta y validación de horarios. Su desarrollo estará condicionado por las limitaciones de tiempo propias del semestre académico y los recursos tecnológicos disponibles, lo cual restringirá la implementación a un piloto único dentro de una comunidad determinada. A pesar de estas limitaciones, el prototipo debe cumplir con criterios de aceptación claros: ofrecer una interfaz simple y accesible que permita a los usuarios organizar las reservas de manera transparente, evitar conflictos de dobles reservas y validar su funcionamiento mediante retroalimentación directa de los actores comunitarios involucrados.

Metodología

Para el desarrollo del proyecto se empleará la metodología ágil Scrum, ya que esta ofrece un marco flexible y colaborativo que se ajusta a la necesidad de trabajar con entregas parciales y validaciones constantes con los usuarios finales. La estructura de Scrum permitirá dividir el trabajo en sprints cortos, con revisiones periódicas que aseguren que el prototipo responda a las expectativas de la comunidad. En este esquema, el docente asesor asumirá el rol de Scrum Master, el equipo de estudiantes actuará como el grupo de desarrollo, y un representante comunitario podrá cumplir el rol de Product Owner, garantizando que las decisiones técnicas estén alineadas con las necesidades sociales. Además, se gestionará el progreso y las historias de usuario en un tablero digital (Trello o Jira), lo que facilitará la transparencia y la organización de las tareas del equipo.

Matriz de riesgo

No	Riesgo identificado	Probabilidad	impacto	Nivel de riesgo	Estrategia de mitigación
1	Baja participación comunitaria en las entrevistas y grupos focales.	Media	Alta	Alto	Generar confianza a través de líderes barriales, realizar convocatorias inclusivas y ofrecer horarios flexibles para la participación.
2	Resistencia al cambio por parte de ciertos grupos acostumbrados a un manejo informal de las canchas.	Alta	Media	Alto	Involucrar a estos actores desde las primeras fases del diseño, destacando los beneficios colectivos de la solución.
3	Limitaciones técnicas (falta de internet o acceso a dispositivos móviles).	Media	Media	Medio	Diseñar un prototipo ligero, multiplataforma y accesible desde celulares básicos, con posibilidad de uso offline parcial.
4	Retrasos en la validación comunitaria del prototipo.	Media	Alta	Alto	Definir un cronograma claro con espacios de retroalimentación y mantener comunicación constante con los participantes.
5	Conflictos internos en la comunidad que afecten la implementación del proyecto.	Baja	Alta	Medio	Garantizar un enfoque neutral, mediado por la Junta de Acción Comunal o líderes reconocidos para reducir tensiones.
6	Dificultades en la recolección y análisis de la	Media	Media	Medio	Usar diferentes técnicas de triangulación

	información cualitativa.				(entrevistas, grupos focales, observación) para complementar los hallazgos.
7	Limitaciones de tiempo académico para cumplir todas las fases del proyecto.	Alta	Alta	Critico	Priorización de actividades esenciales, ajuste del cronograma y asignación clara de responsabilidades dentro del equipo.
8	Escasa apropiación de la solución por parte de la comunidad después del diseño del prototipo.	Baja	Alta	Medio	Asegurar que la herramienta se construya con participación activa, validación continua y adaptaciones según la retroalimentación.

Levantamiento de información

El levantamiento de información se realizará a través de metodologías participativas que permitan recoger las voces y experiencias de los usuarios de la cancha comunitaria. Para ello, se aplicarán entrevistas semiestructuradas a líderes locales y usuarios frecuentes, complementadas con grupos focales que faciliten la identificación de problemáticas comunes y expectativas colectivas. Además, se emplearán herramientas propias del diseño centrado en el usuario, como mapas de experiencia y perfiles de usuario, que ayudarán a estructurar los hallazgos de forma visual y estratégica. Estas técnicas permitirán comprender de manera profunda cómo se percibe actualmente la gestión de reservas, qué tensiones surgen entre los distintos actores y cuáles son las oportunidades de mejora. De manera preliminar, se ha identificado que la comunidad percibe injusticias en la asignación de horarios, manifiesta la necesidad de mayor transparencia en la administración de los espacios y espera una herramienta tecnológica sencilla, rápida y accesible desde dispositivos móviles.

Stakeholders

Stakeholders	Rol del proyecto	interés	poder	clasificación
Comunidad local	Uso del sistema	alto	medio	Clave
Junta de acción comunal	administradores	Alto	Alto	Decisiones
Estudiantes	Desarrollo	Alto	Alto	Facilitador

Requerimientos funcionales

código	Requisito funcional
RQF001	Nombre: Registro de usuario
	Descripción: el sistema permitirá al usuario realizar la acción de registrarse
	Usuarios: usuario

código	Requisito funcional
RQF002	Nombre: registro del administrador
	Descripción: el sistema permitirá al usuario realizar la acción de registrase por medio de CLI
	Usuarios administrador

código	Requisito funcional
RQF003	Nombre: inicio de sesión
	Descripción: el sistema permitirá al usuario realizar la acción de iniciar sesión
	Usuarios: usuario , administrador

código	Requisito funcional
RQF004	Nombre: Cerrar sesión
	Descripción: el sistema permitirá al usuario realizar la acción de cerrar sesión
	Usuarios: usuario , administrador

código	Requisito funcional
RQF005	Nombre: actualizar perfil
	Descripción: el sistema permitirá al usuario realizar la acción de ver sus datos personales
	Usuarios: usuario , administrador

código	Requisito funcional
RQF006	Nombre: actualizar perfil
	Descripción: el sistema permitirá al usuario realizar la acción de actualizar sus datos personales
	Usuarios: usuario , administrador

código	Requisito funcional
RQF006	Nombre: crear espacios deportivos
	Descripción: el sistema permitirá al usuario realizar la acción de registrar los espacios deportivos
	Usuarios: administrador

código	Requisito funcional
RQF006	Nombre: listar espacios deportivos
	Descripción: el sistema permitirá al usuario realizar la acción de listar los espacios deportivos
	Usuarios: administrador

código	Requisito funcional
RQF006	Nombre: listar espacios deportivos
	Descripción: el sistema permitirá al usuario realizar la acción de listar los espacios deportivos habilitados
	Usuarios: administrador, usuario

código	Requisito funcional
RQF006	Nombre: obtener espacio deportivo
	Descripción: el sistema permitirá al usuario realizar la acción de obtener el espacio deportivo
	Usuarios: administrador

código	Requisito funcional
RQF006	Nombre: obtener espacio deportivo
	Descripción: el sistema permitirá al usuario realizar la acción de obtener el espacio deportivo habilitado
	Usuarios: administrador, usuario

código	Requisito funcional
RQF006	Nombre: crear reserva
	Descripción: el sistema permitirá al usuario realizar la acción de reservar el espacio deportivo
	Usuarios: usuario

código	Requisito funcional
RQF006	Nombre: actualizar reserva
	Descripción: el sistema permitirá al usuario realizar la acción de actualizar la reserva
	Usuarios: administrador

código	Requisito funcional
RQF006	Nombre: eliminar reserva
	Descripción: el sistema permitirá al usuario realizar la acción de actualizar (soft delete) la reserva
	Usuarios: usuario

código	Requisito funcional
RQF006	Nombre: eliminar reserva
	Descripción: el sistema permitirá al usuario realizar la acción de actualizar (soft delete) la reserva
	Usuarios: usuario

código	Requisito funcional
RQF006	Nombre: eliminar reserva
	Descripción: el sistema permitirá al usuario realizar la acción de eliminar la reserva
	Usuarios: administrador

Código	Requisito no funcional
RQNF001	Nombre: Especificaciones mínimas de hardware y software
	Descripción: Tener acceso a una computadora o mas Mínimo de RAM 4gb Sistema operativo Windows 8 o superior, Ubuntu 20.04 o superior Disco duro 256gb Conectividad a internet wifi, ethernet Que tenga las herramientas y programas para el desarrollo de la aplicación

Presupuesto

1. Recursos humanos

Rol	Integrante	Horas estimadas	Valor Hora ref	Subtotal
Analista de requerimientos	1	60h	$ 20.000	$ 1.200.000
Diseñador ux/ui	1	50h	$ 20.000	$ 1.000.000
Desarrollador backend	1	80h	$ 20.000	$ 1.600.000
Desarrollador frontend	1	80h	$ 20.000	$ 1.600.000
Tester QA	1	40h	$20.000	$ 800.000
Total				$ 6.200.000

2. Recursos tecnológicos

Recurso	Descripción	Costo estimado
Dominio web	Registro por 1 año	$ 50.000
Hosting web	Plan básico (6 meses)	$ 600.000
Licencias de diseño(Figma)	Suscripción mensual (6 meses)	$ 300.000
Herramientas de gestión (Trello - jira)	Versión gratuita	$ 0
Total		$ 950.000

3. Actividades de levantamiento de información

Recurso	Descripción	Costo estimado
materiales	Impresiones, formularios	$ 100.000
Logística	Reuniones comunitarias (refrigerios)	$ 200.000
Total		$ 300.000

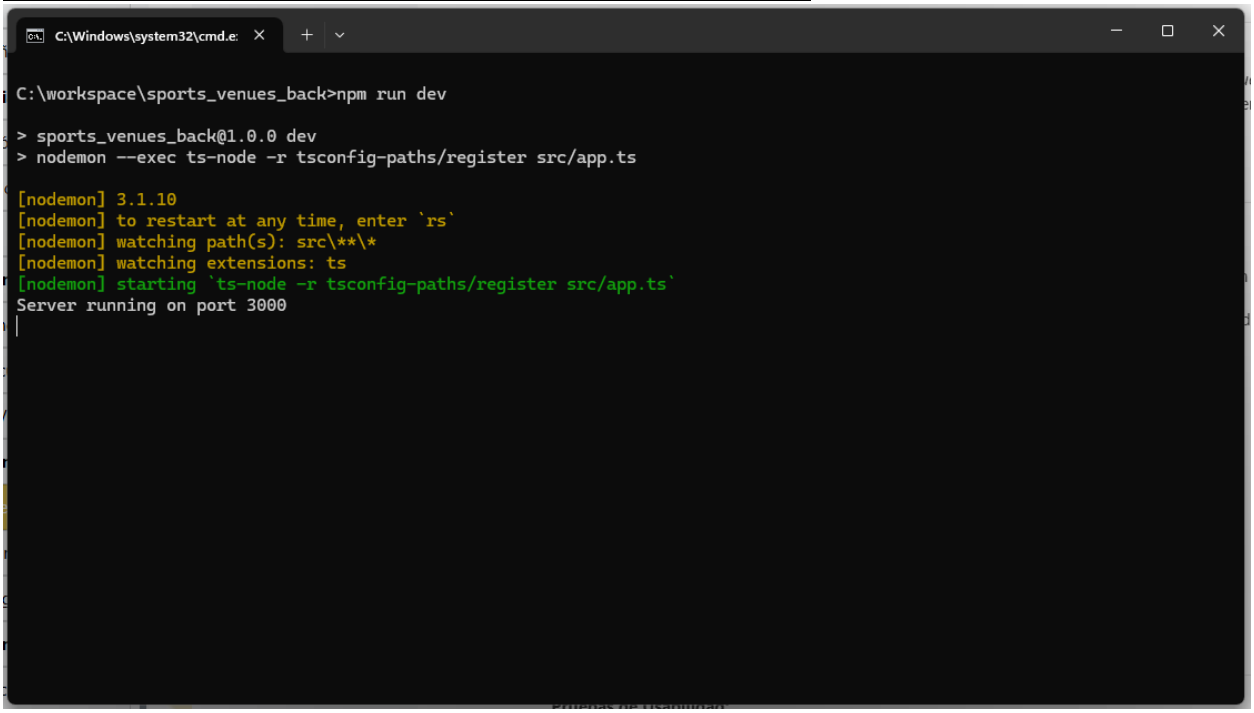
Para un total estimado para el desarrollo del proyecto es de $8.195.000

Historia de usuario
https://trello.com/invite/b/68df1fa5ee4e9fd39fa5bfc9/ATTI51e3c75a2f3107edb1131739784e93b3D5ACD9A9/analisis-y-diseno

Prototipo funcional

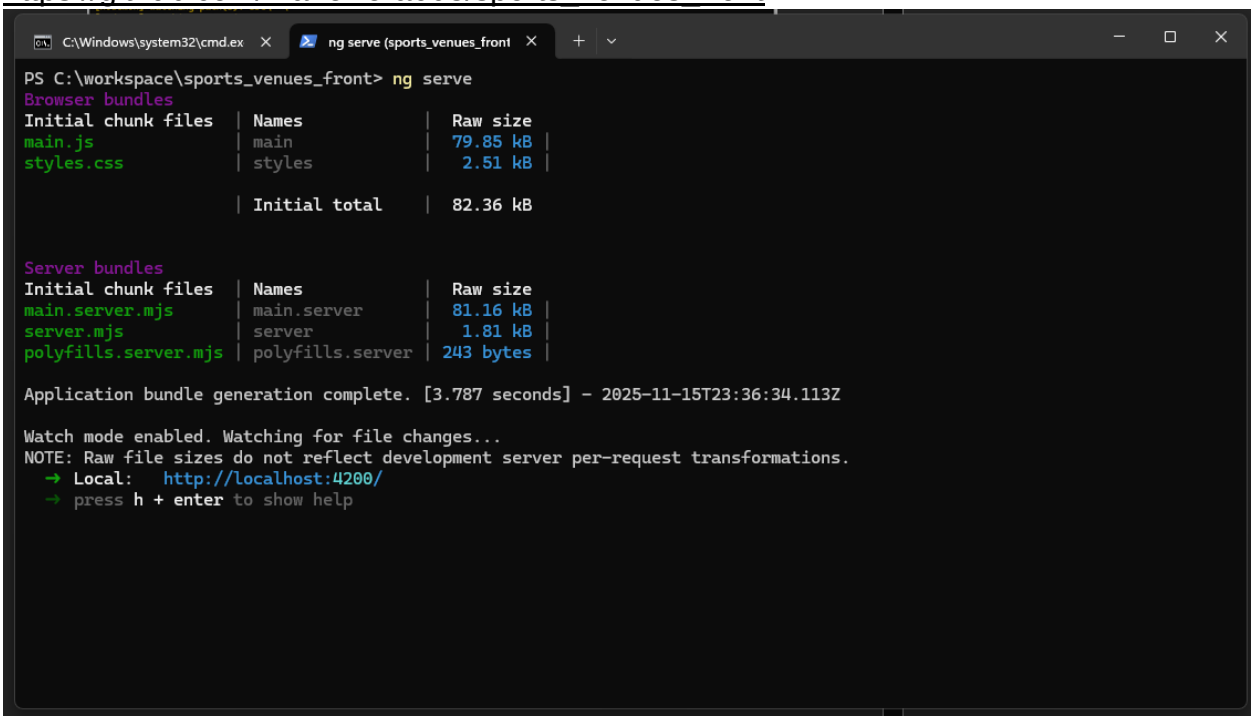
Link – repositorio backend

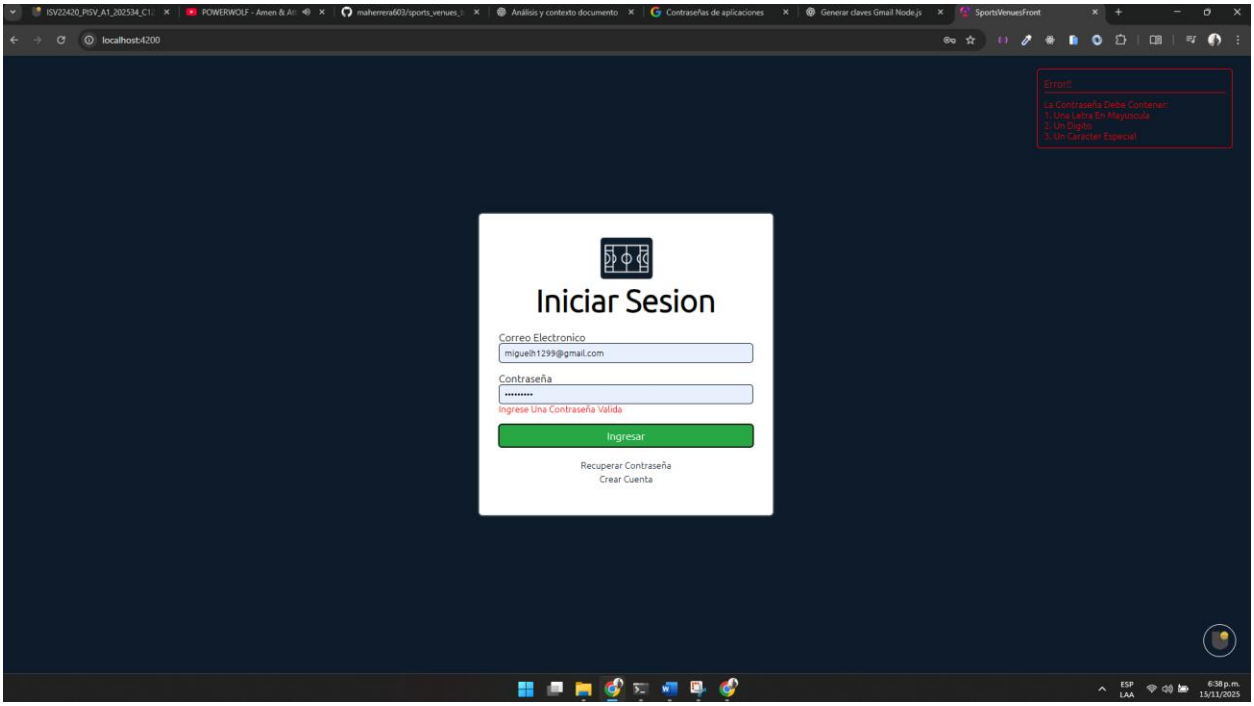
https://github.com/maherrera603/sports_venues_back.git

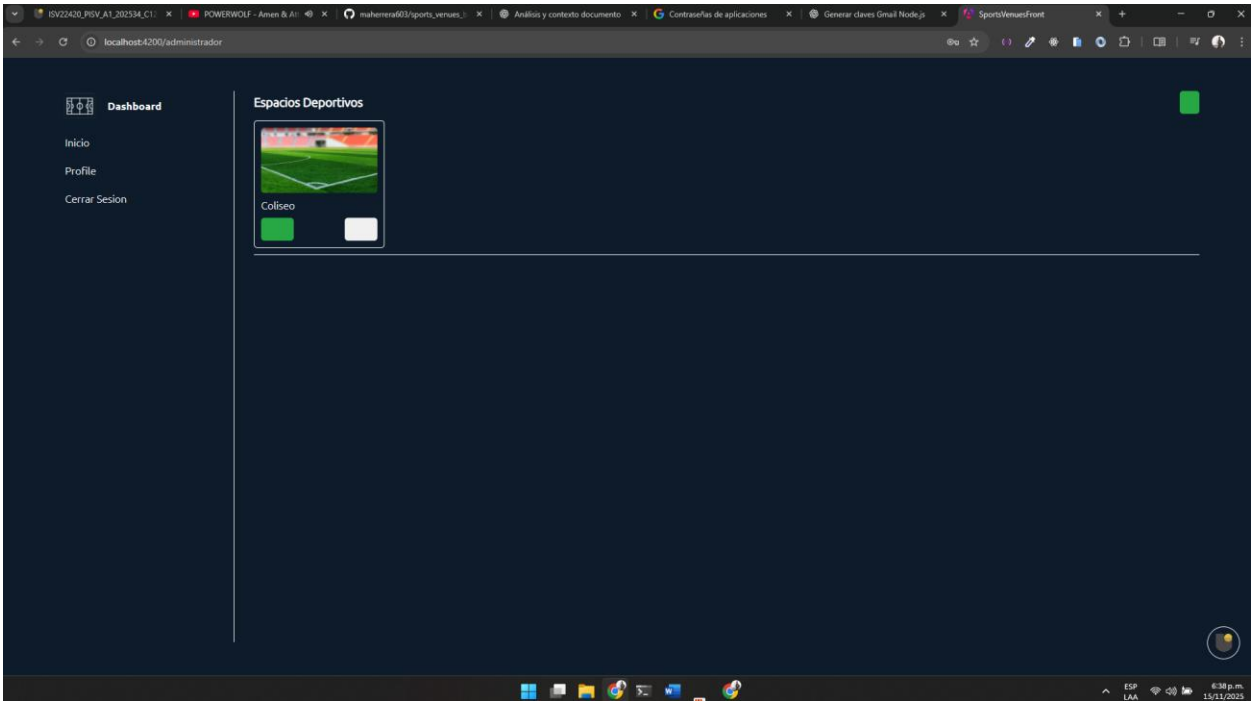


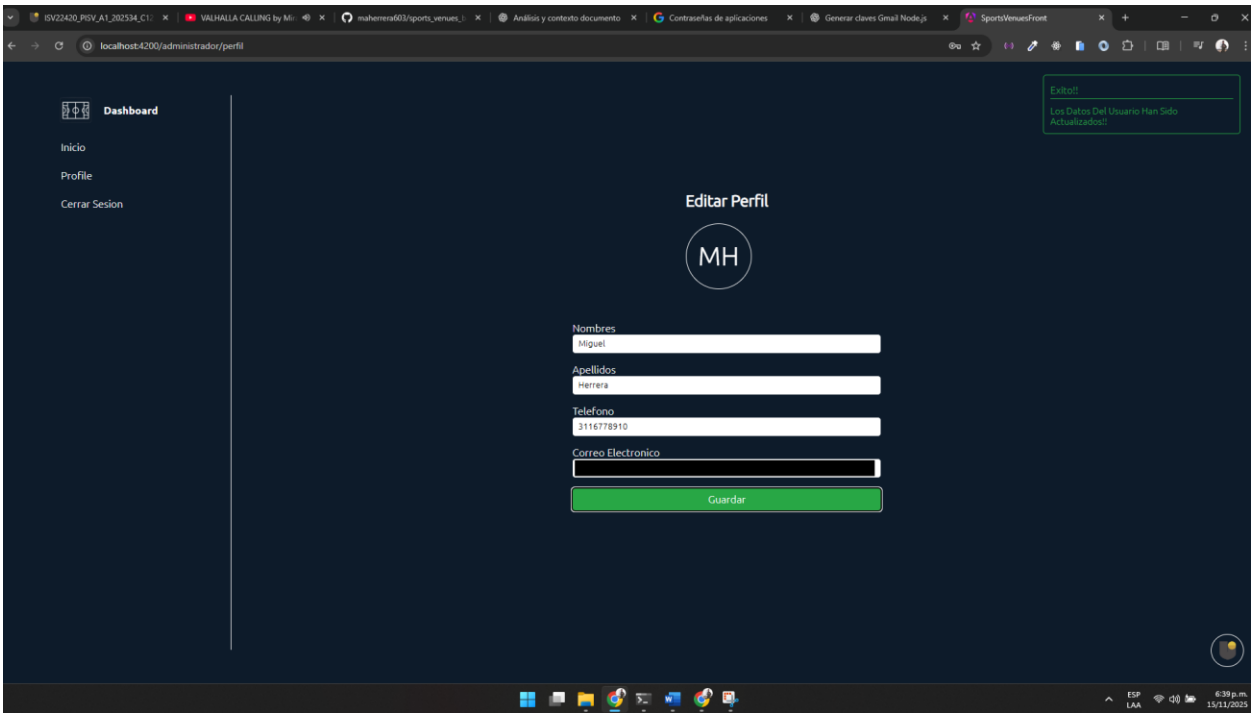
Link – repositorio frontend – avances

https://github.com/maherrera603/sports_venues_front




localhost:4200
Error!!!
La Contraseña Debe Contener:
1. Una Letra En Mayuscula
2. Un Digito
3. Un Caracter Especial
Iniciar Sesion
Correo Electronico
migueih1299@gmail.com
Contraseña
Ingrese Una Contraseña Valida
Ingresar
Recuperar Contraseña
Crear Cuenta


localhost:4200/administrador
Dashboard
Inicio
Profile
Cerrar Sesion
Espacios Deportivos
Coliseo


localhost:4200/administrador/perfil
Dashboard
Inicio
Profile
Cerrar Sesion
Exito!!
Los Datos Del Usuario Han Sido Actualizados!!
Editar Perfil
MH
Nombres
Miguel
Apellidos
Herrera
Telefono
3116778910
Correo Electronico
Guardar

Modelos de comportamiento y estructura

Diagrama de casos de uso

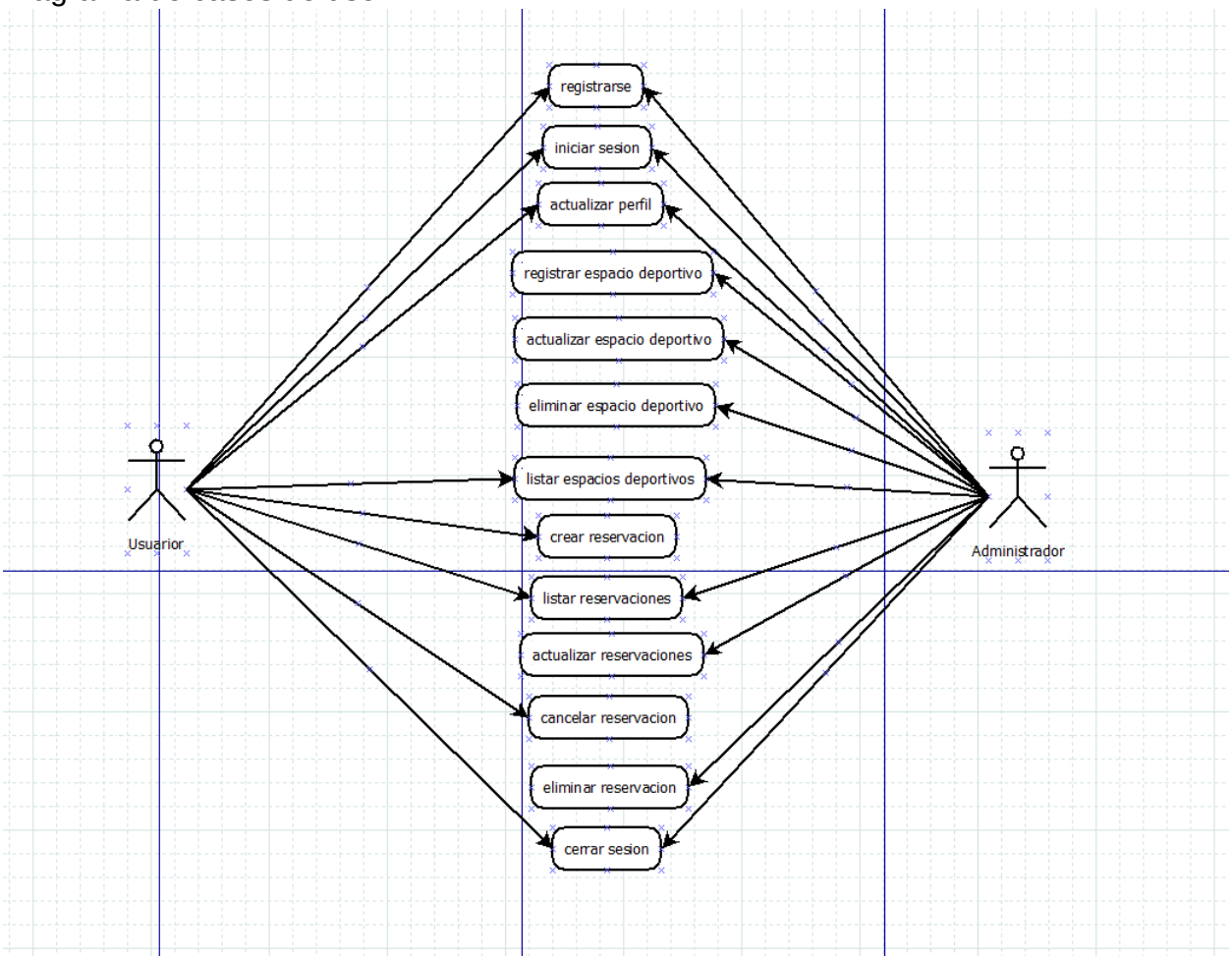


Diagrama de secuencia

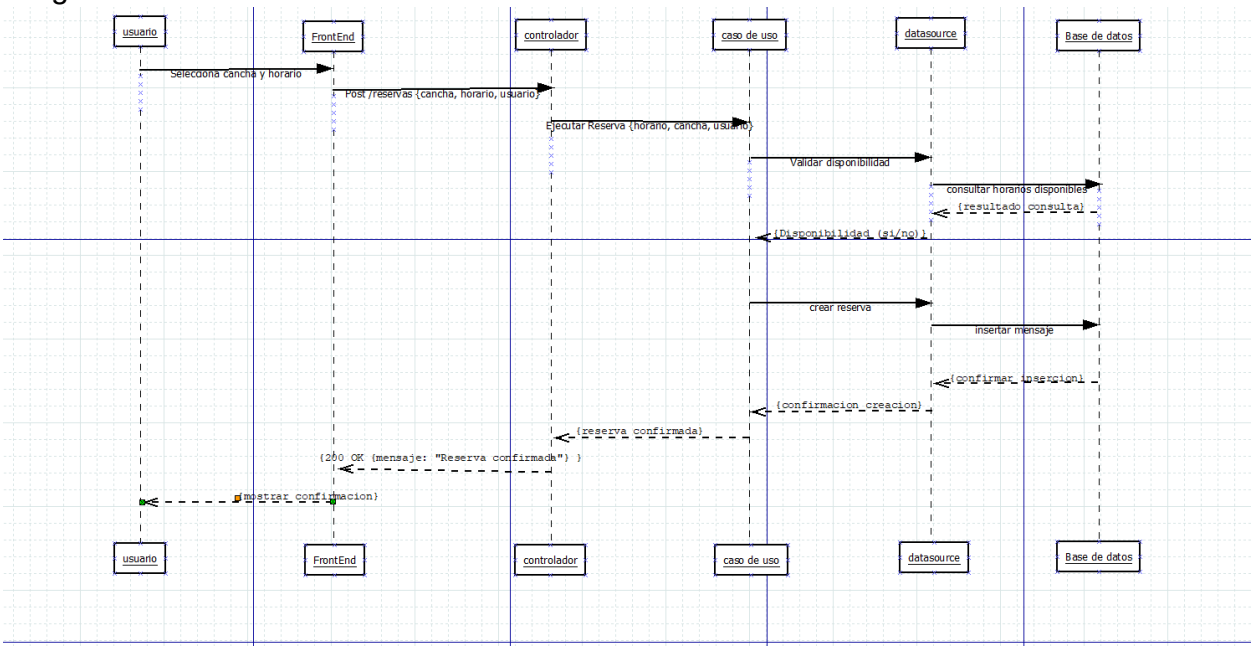


Diagrama de clases

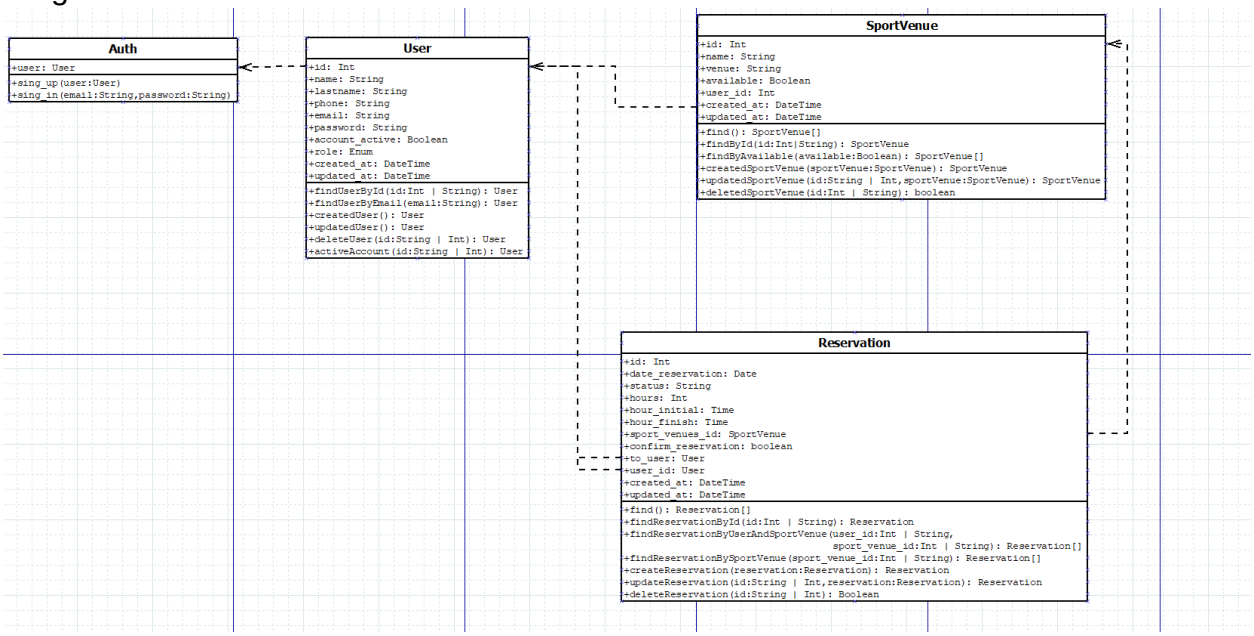


Diagrama de componentes

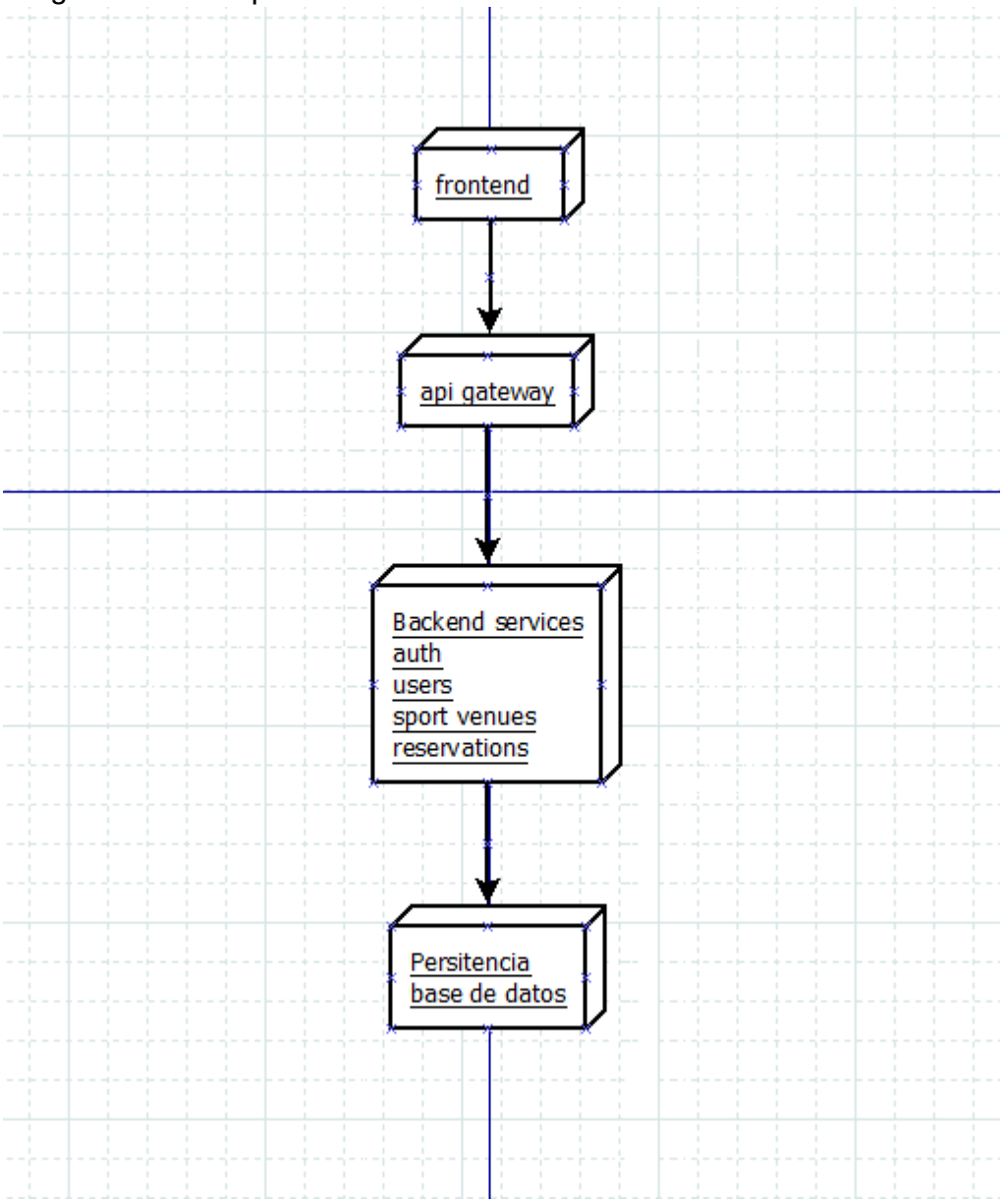
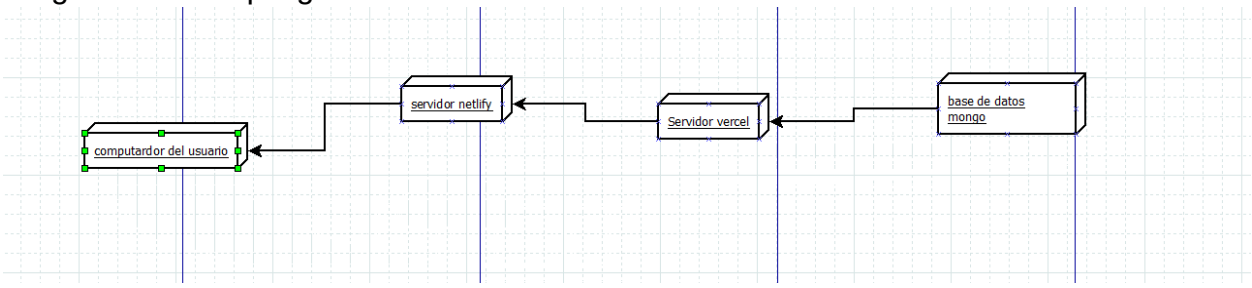
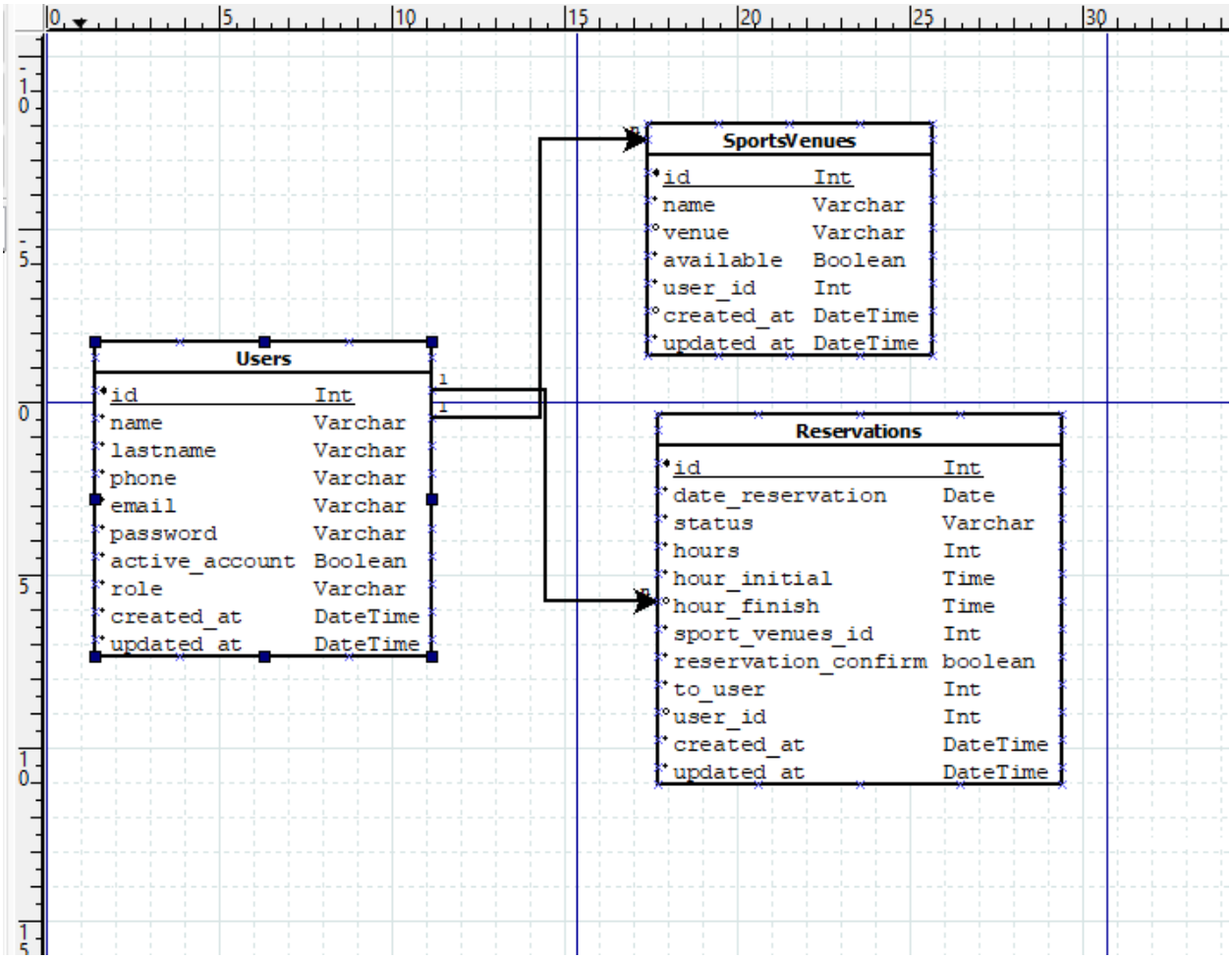


Diagrama de despliegue



Modelo de base de datos



Prototipos de interfaz

Link prototipo baja fidelidad
https://www.figma.com/proto/BAbEMgFACx6loAk8Y0dp1V/Untitled?page-id=0%3A1&node-id=1-43&p=f&viewport=173%2C-181%2C0.98&t=1Ou7SSUiwfnAdixO-1&scaling=min-zoom&content-scaling=fixed&starting-point-node-id=1%3A43&show-proto-sidebar=1

Link prototipo alta fidelidad
https://www.figma.com/proto/BAbEMgFACx6loAk8Y0dp1V/Untitled?page-id=1%3A2&node-id=25-1517&viewport=-550%2C307%2C0.98&t=1LSXE1Lrgm03kdOy-1&scaling=min-zoom&content-scaling=fixed&starting-point-node-id=25%3A1517

Link maze
https://t.maze.co/469439679

Patrones arquitectónicos

Clean Architecture (Arquitectura Limpia)

El backend se diseñó bajo el patrón Clean Architecture, que separa el sistema en capas independientes, permitiendo que la lógica de negocio no dependa del framework, la base de datos ni servicios externos.

Capas implementadas:

Domain: entidades, reglas de negocio y modelos esenciales.

Use Cases / Application: casos de uso (crear reserva, validar disponibilidad, cancelar reserva, CRUD de espacios).

Interfaces / Adapters: controladores HTTP, DTOs, validaciones, repositorios como interfaces.

Infrastructure: implementación real de la base de datos, correo, cache, almacenamiento y API.

Justificacion
La arquitectura se seleccionó en función de los siguientes criterios:

- Escalabilidad
  La división en módulos (auth, reservas, espacios, usuarios) permite escalar horizontalmente cada parte cuando el sistema crezca.

- Robustez y mantenibilidad
  - Clean Architecture garantiza:
    i. Código fácil de extender
    ii. Dependencias bien controladas
    iii. Reglas de negocio inalterables ante cambios tecnológicos

- Separación clara de responsabilidades
  - Cada capa cumple una función específica:
    - Domain → Reglas puras
    - Use Cases → Orquestación
    - Controllers → Entrada/salida HTTP
    - Infra → conexión con servicios externos

- Preparación para microservicios
  Aunque el sistema es monolítico para reducir complejidad inicial, la estructura modular permite migrar cada servicio a microservicio sin reescribir la lógica.

- Facilita pruebas unitarias
  Los casos de uso pueden probarse sin necesidad de base de datos real.

GitHub
https://github.com/maherrera603/proyecto-software

Tecnologías utilizadas

El sistema fue implementado utilizando un enfoque de aplicación web moderna basada en arquitectura cliente-servidor. Para el backend se empleó Node.js con TypeScript, utilizando un framework orientado a APIs REST, lo que permitió estructurar el proyecto de forma modular y escalable. En el frontend se utilizó el framework de angular para el desarrollo web basado en componentes, lo que facilitó la reutilización de código y la construcción de interfaces dinámicas.

La persistencia de la información se gestionó mediante una base de datos relacional/no relacional (según configuración del entorno), accedida a través de un ORM que abstrae las operaciones CRUD y mejora la mantenibilidad del código.

Implementación de la arquitectura del software

La aplicación fue desarrollada siguiendo los principios de Clean Architecture, lo que garantiza una separación clara de responsabilidades y un bajo acoplamiento entre componentes.

Las capas implementadas son:

- **Domain:** Contiene las entidades principales del sistema como Usuario, EspacioDeportivo y Reserva, junto con las reglas de negocio fundamentales.
- **Application (Use Cases):** Incluye los casos de uso que orquestan la lógica del sistema, tales como crear reserva, validar disponibilidad, cancelar reserva y gestionar usuarios.
- **Interfaces / Adapters:** Implementa los controladores HTTP, validaciones de entrada y DTOs utilizados para la comunicación entre capas.
- **Infrastructure:** Contiene las implementaciones concretas de acceso a base de datos, servicios externos y configuración del servidor.

Este enfoque permitió que la lógica de negocio se mantuviera independiente de frameworks, bases de datos o servicios externos.

Patrones de diseño implementados

Durante la implementación se aplicaron varios patrones de diseño orientados a mejorar la calidad del software:

- **Repository Pattern:** Se utilizó para desacoplar la lógica de negocio del acceso a datos, permitiendo cambiar la fuente de persistencia sin afectar los casos de uso.
- **DTO (Data Transfer Object):** Se emplearon objetos de transferencia para estructurar los datos que viajan entre el cliente y el servidor, evitando la exposición directa de las entidades del dominio.
- **Singleton:** Aplicado en la configuración de la conexión a la base de datos para garantizar una única instancia activa durante la ejecución.
- **Factory:** Utilizado para la creación controlada de entidades del dominio, asegurando la validación de reglas de negocio desde su origen.

Aplicación de los principios SOLID

El desarrollo del sistema tuvo en cuenta los principios SOLID:

- **Responsabilidad Única (SRP):** Cada clase y módulo cumple una única función específica.
- **Abierto/Cerrado (OCP):** El sistema permite extender funcionalidades sin modificar código existente.
- **Sustitución de Liskov (LSP):** Las implementaciones respetan los contratos definidos por sus interfaces.
- **Segregación de Interfaces (ISP):** Las interfaces son pequeñas y específicas, evitando dependencias innecesarias.
- **Inversión de Dependencias (DIP):** Los casos de uso dependen de abstracciones y no de implementaciones concretas.

Casos de prueba

ID	Descripción	Precondición	Resultado esperado	Resultado obtenido
CP-01	Registro de usuario	Datos válidos	Usuario creado	OK
CP-02	Login válido	Usuario registrado	Acceso concedido	OK
CP-03	Crear reserva	Usuario autenticado	Reserva creada	OK
CP-04	Reserva duplicada	Horario ocupado	Reserva rechazada	OK
CP-05	Cancelar reserva	Reserva existente	Reserva cancelada	OK
CP-06	Listar espacios	Sistema activo	Lista mostrada	OK
CP-07	Acceso sin token	Usuario no autenticado	Acceso denegado	OK
CP-08	Rol administrador	Usuario admin	Acceso permitido	OK

Proceso de Despliegue

Plataforma de despliegue

El sistema fue desplegado utilizando servicios de alojamiento en la nube. El frontend se publicó en una plataforma de despliegue continuo para aplicaciones web, mientras que el backend se alojó en un servicio render compatible con Node.js. La base de datos se configuró en un servicio administrado.

Proceso paso a paso

1. Preparación de los repositorios en GitHub.
2. Configuración de variables de entorno.
3. Construcción del proyecto (build).

4. Despliegue automático desde el repositorio.
5. Validación del funcionamiento en entorno productivo.

Evidencias del Desarrollo

Como evidencia del desarrollo se incluyen los repositorios públicos del proyecto, los resultados documentados de las pruebas de software y la validación funcional de los principales casos de uso. Estas evidencias respaldan la correcta implementación del sistema y su disponibilidad para uso público.

Pruebas de Software y Aseguramiento de la Calidad

Estrategia de pruebas

Las pruebas se realizaron con el objetivo de validar el correcto funcionamiento del sistema, asegurar la calidad del software y verificar que los requerimientos funcionales y no funcionales fueran cumplidos.

Pruebas según el nivel

Pruebas unitarias

Se realizaron pruebas unitarias sobre los casos de uso del backend, validando reglas de negocio como la creación de reservas, validación de horarios y control de duplicidad.

Pruebas de integración

Estas pruebas permitieron verificar la correcta interacción entre módulos, como la comunicación entre controladores, servicios y repositorios.

Pruebas de sistema

Se evaluó el sistema completo en un entorno de ejecución, comprobando los flujos principales desde el inicio de sesión hasta la creación y gestión de reservas.

Pruebas de aceptación

Se validaron los escenarios principales desde la perspectiva del usuario final, confirmando que el sistema responde a las necesidades planteadas.

Pruebas según la técnica

- **Caja Blanca:** Se analizaron las rutas lógicas internas del código.
- **Caja Negra:** Se validaron entradas y salidas sin considerar la implementación interna.
- **Caja Gris:** Se combinó el conocimiento parcial del sistema con pruebas funcionales.

Pruebas funcionales

- Registro de usuario
- Inicio y cierre de sesión
- Creación, modificación y cancelación de reservas
- Gestión de espacios deportivos

Pruebas no funcionales

- **Rendimiento básico:** Validación de tiempos de respuesta aceptables.
- **Seguridad:** Uso de autenticación basada en tokens y control de roles.

Bibliografia

Cabrera, A., & López, D. (2020). Diseño e implementación de un sistema de reservas para instalaciones deportivas comunitarias. Universidad Politécnica de Madrid.

Mata, A., & Sánchez, R. (2019). Plataformas web para la gestión de recursos comunitarios: modelos, requisitos y mejores prácticas. Revista Iberoamericana de Sistemas, 15(2), 45–60.

Villalobos, J., & Gómez, P. (2021). Arquitectura y desarrollo de aplicaciones web basadas en API REST para reservas y control de disponibilidad de recursos. Revista Colombiana de Ingeniería de Software, 13(1), 22–34.

Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall.

García, M., & Hernández, T. (2022). Modelado UML y despliegue de aplicaciones para sistemas de información comunitaria. Revista Tecnología y Sociedad, 10(3), 75–89.