

# Índice

ANALISIS INICIAL.....	2
Actores del Sistema:.....	2
Casos de uso:.....	2
C1: Registro de usuarios :.....	2
C2: <i>Consultar espacios disponibles</i> :.....	2
C3: Crear una nueva reserva :.....	2
C4: Modificación de una reserva :.....	3
C5:Cancelación de reserva :.....	3
C6: Finalización automática de reservas :.....	3
C7: Gestión de usuarios :.....	3
C8: Gestión de los espacios :.....	3
C9: Visualizar las reservas :.....	3
C10 : Notificaciones :.....	4
DISEÑO DE ARQUITECTURA.....	4
1. BACKEND → .NET.....	4
2. FRONTEND → ANGULAR.....	4
3. BBDD → SQL server.....	5
4. Eventos → Apache Kafka.....	5
5- Despliegue de la aplicación → Docker Compose.....	5
6- Testing → Xunit y jasmine/karma.....	5
FLUJO DE TRABAJO.....	6
REQUISITOS FUNCIONALES Y NO FUNCIONALES.....	8
FUNCIONALES.....	8
RF 1. Registro de usuarios.....	8
RF 2. Gestión de espacios.....	8
RF 3. Creación y gestión de las reservas.....	8
RF 5. Gestión por parte de la administración.....	9
RF 6. Filtrar reservas.....	9
RF 7. Finalización automática de reservas con kafka.....	9
NO FUNCIONALES.....	9
DEFINICIÓN DE MODELOS DE DATOS PARA LA BASE DE DATOS.....	10
Modelo conceptual.....	10
Relaciones.....	10
Modelo lógico de datos propuesto.....	11

# **ANALISIS INICIAL**

Se requiere una aplicación web donde los usuarios puedan reservar espacios, como salas, pistas deportivas, etc.

- Consultar disponibilidad de los espacios a reservar
- Crear, modificar y cancelar reservas
- Gestionar los espacios de forma óptima, evitando solapamiento de reservas
- Notificar los cambios a los usuarios

## **Actores del Sistema:**

- **Usuario**: Consulta la disponibilidad de las salas, crea/edita/modifica/cancela sus reservas, y recibe notificaciones
- **Administrador** : Supervisa todas las reservas de todas las salas, edita/modifica/cancela las reservas de cualquier usuario. Exporta los resultados para un posterior análisis.

## **Casos de uso:**

### **C1: Registro de usuarios**

- Un usuario** crea una cuenta para acceder al sistema y realizar reservas.
- El usuario introducirá un nombre, e-mail y contraseña.
- El sistema validará que no exista duplicidad en los datos.
- La cuenta se valida y se registra el usuario, finalmente se inicia sesión.

### **C2: Consultar espacios disponibles**

- Un usuario** selecciona una fecha y se comprueba qué salas están disponibles.
- Se muestran los espacios disponibles en esa fecha y hora seleccionadas.

### **C3: Crear una nueva reserva**

- El usuario reserva un espacio en una hora y fecha disponibles.
- El sistema comprueba que no existen reservas en ese preciso momento y se valida la reserva, mandando un evento de tipo “ReservaCreada” a kafka.
- La reserva se crea pasando al estado “Activa”.

## C4: Modificación de una reserva

- Un usuario modifica su reserva.
- Cambia la fecha y hora si es posible.
- El sistema comprueba que no existe solapamiento.
- Se registra el cambio y se notifican las actualizaciones

## C5: Cancelación de reserva

- Un usuario cancela su reserva
- Se verifica que no haya comenzado la reserva, pues no se puede cancelar una reserva iniciada
- Se hace click en cancelar y pasa al estado “Cancelada”
- Kafka recibe el evento de reserva cancelada
- Se notifica al usuario de la cancelación

## C6: Finalización automática de reservas

- El sistema marca como finalizadas todas aquellas reservas en estado activa una vez se ha completado el tiempo de la reserva.
- El sistema marca la reserva como “Finalizada”
- Se notifica al usuarios

## C7: Gestión de usuarios

- Un administrador crea, edita o elimina usuarios

## C8: Gestión de los espacios

- El administrador puede ver los espacios reservados, modificarlos, o cancelarlos
- El administrador puede crear un nuevo espacio reservable
- El administrador puede borrar un espacio existente

## C9: Visualizar las reservas

- El usuario y el administrador pueden ver que salas están reservadas y en qué fechas y horas concretas
- El usuario no debe ver a qué otro usuario pertenece una reserva, simplemente puede observar si un espacio está reservado o no
- El administrador puede ver la información de cualquier usuario asociado a una reservable

## C10 : Notificaciones

- El sistema avisará al usuario de cambios importantes en su reserva.

## DISEÑO DE ARQUITECTURA

Tecnología	Funciones
.NET	Backend
Angular	Frontend
SQL Server	BBDD
Apache kafka	Sistema de eventos
Docker-Docker Compose	Despliegue del sistema mediante contenerización
xUnit/Jasmine-Karma	Testing

### 1. BACKEND .NET

- Gestión de usuarios, espacios y reservas
- Controlar las reglas de negocio, como el solapamiento
- Controla la autenticación y permisos de usuarios
- Comunicación con base de datos mediante Entity framework
- Envía y recibe eventos Kafka
- Endpoints consumibles en el Frontend.

### 2. FRONTEND ANGULAR

- Mostrar las pantallas de login, lista de espacios, calendario, etc
- Recoge los datos introducidos por el usuario y los envía al backend mediante peticiones HTTP
- Guarda el token de sesión
- Diseño web responsive

### **3. BBDD POSTRESSQL**

- Persistencia de la información
- Filtrado de información
- Garantiza la integridad de los datos
- Comunicación con el backend
- Base de datos relacional

### **4. EVENTOS APACHE KAFKA**

- Gestión de eventos con Apache Kafka
- Guardar y enviar los eventos
- Comunicación con el worker

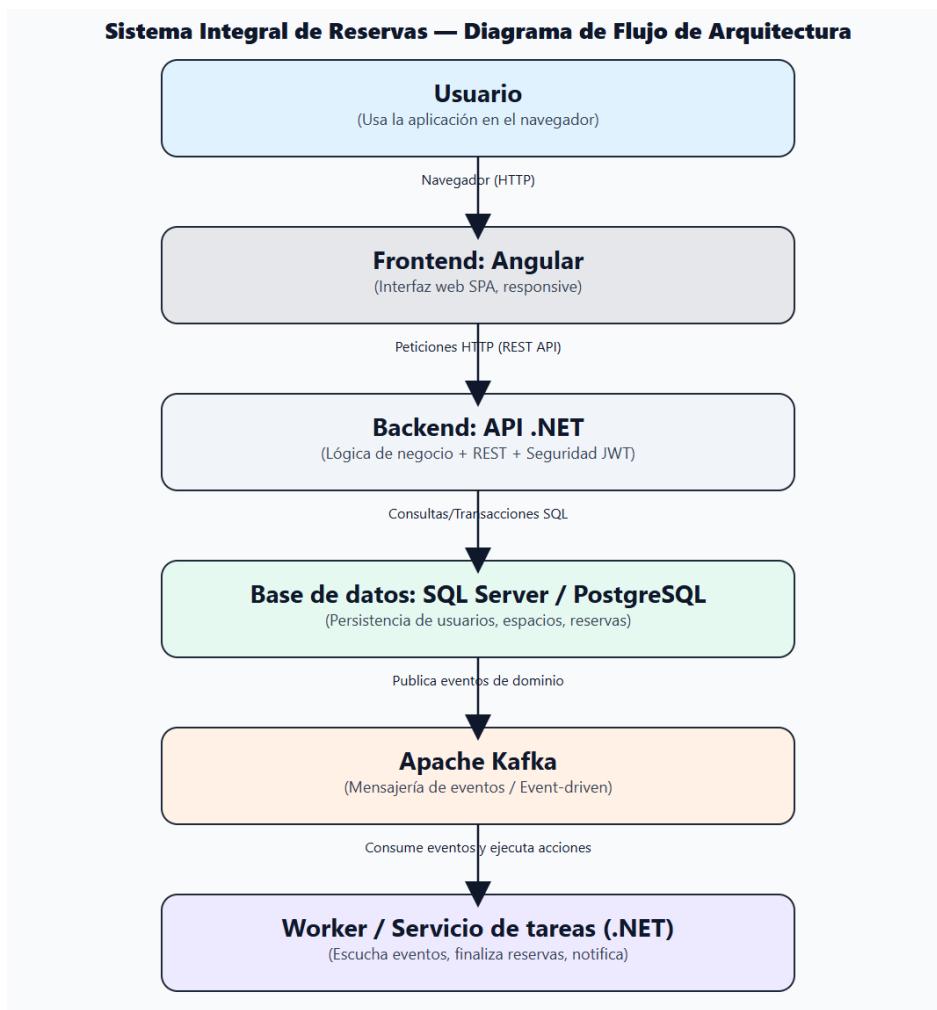
### **5- DESPLIEGUE DOCKER COMPOSE**

- Creación de contenedores para cada parte del sistema
- Conexión con todos mediante docker Compose
- Despliegue total del sistema

### **6- TESTING**

- Herramientas de testeo Xunit y jasmine/karma

# FLUJO DE TRABAJO



# **REQUISITOS FUNCIONALES Y NO FUNCIONALES**

## **FUNCIONALES**

### **RF 1. Registro de usuarios**

- RF 1.1 : El sistema debe permitir a los usuarios registrarse
- RF 1.2 : El sistema debe validar que el usuario sea único en el sistema
- RF 1.3 : El sistema debe almacenar la contraseña de forma segura
- RF 1.4 : El sistema debe validar el formato del dni y del correo electrónico
- RF 1.5 : El sistema debe comprobar que no existan usuarios registrados con el mismo dni o el mismo correo electrónico.

### **RF 2. Gestión de espacios**

- RF 2.1 : El sistema debe permitir reservar un espacio
- RF 2.2 : El sistema debe verificar que un espacio no esté reservado para poder realizar una reserva
- RF 2.3 : El sistema debe verificar la finalización de una reserva en un espacio antes de poder liberarlo para una nueva reserva
- RF 2.4 : El sistema debe verificar que un espacio está activo, existe y se puede reservar

### **RF 3. Creación y gestión de las reservas**

- RF 3.1 : El sistema debe evitar solapamiento a la hora de gestionar una reserva
- RF 3.2 : El sistema debe permitir reservar un espacio disponible
- RF 3.3 : El sistema debe permitir modificar o cancelar una reserva no activa
- RF 3.4 : El sistema debe verificar si ha finalizado la reserva de un espacio para lanzar el evento de finalización y liberar el espacio

## **RF 5. Gestión por parte de la administración**

RF 5.1 : Un usuario con el rol de administrador podrá gestionar las reservas de todos los demás usuarios

## **RF 6. Filtrar reservas**

RF 6.1 : El sistema permitirá filtrar por id de reserva para consultar su estado

## **RF 7. Finalización automática de reservas con kafka**

## **NO FUNCIONALES**

1. Seguridad
2. Rendimiento
3. Escalabilidad
4. Fiabilidad
5. Usabilidad
6. Testing
7. Contenerización

# DEFINICIÓN DE MODELOS DE DATOS PARA LA BASE DE DATOS

## Modelo conceptual

ENTIDAD	Descripción	Ejemplo
Usuario	Quien usa el sistema	Adrián Martinez
Espacio	Lugar que puede reservarse	Padel 1, Tenis 5,etc
Reserva	Reserva asociada a un espacio	Padel 1-- Lunes 10 10:00-12:00
Evento	Mensajes kafka	“Reserva creada con id :0011”

## Relaciones

Relacion	Tipo	Tipo
Usuario → Reserva	1 → N	Uno a muchos
Espacio → Reserva	1 → N	Uno a muchos
Reserva → Usuario	N → 1	Muchos a uno

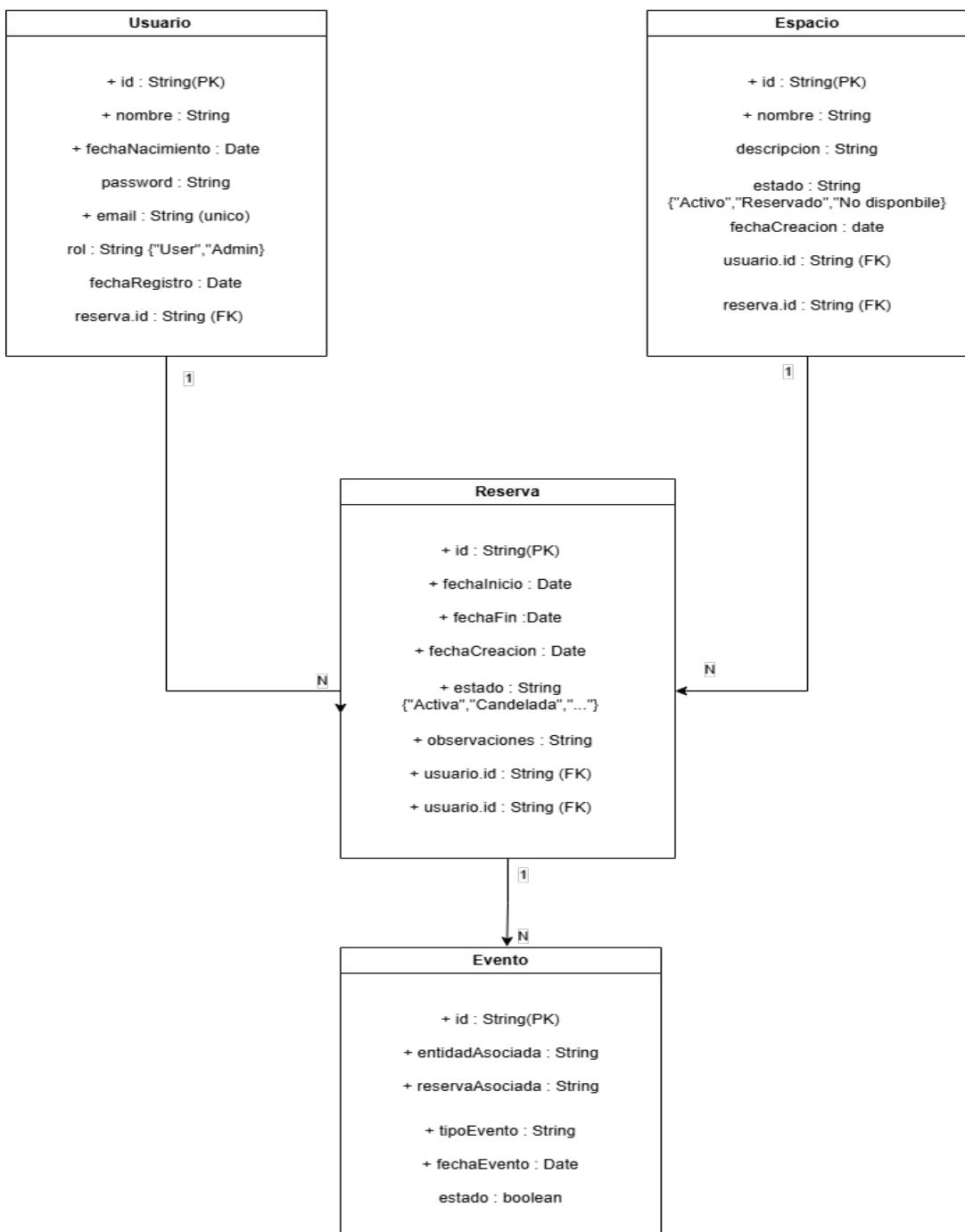
## Modelo lógico de datos propuesto

Usuario		
Campo	Tipo	Descripción
dni	VARCHAR	(PK) Id único de cada usuario
nombre	VARCHAR	Nombre completo
email	VARCHAR	Fecha de nacimiento del usuario
password_hash	VARCHAR	Contraseña cifrada
rol	VARCHAR	Admin o Usuario
activo	Boolean	Usuario activo en el sistema
fechaRegistro	TIMESTAMP	Fecha del registro en la aplicación
fechaNacimiento	TIMESTAMP	
reserva_id	VARCHAR	(FK) Reservas asociadas a un usuario

Reserva		
Campo	Tipo	Descripción
id	VARCHAR	(PK) Id de la reserva
fechaInicio	Date	Fecha de inicio de la reserva
fechaFin	Date	Fecha fin de la reserva
fechaCreacion	Date	Fecha de creación de la reserva
estado	boolean	Activa,Cancelada,Modificada, Finalizada
observaciones	VARCHAR	Notas del usuario
usuario_id	VARCHAR	(FK) Relación con la tabla Usuario
espacio_id	VARCHAR	(FK) Relación con la tabla Espacio

Espacio		
Campo	Tipo	Descripción
id	VARCHAR	(PK) ID unico de cada espacio registrado
nombre	VARCHAR	Nombre del espacio
descripcion	VARCHAR	Detalels
estado	VARCHAR	Activo,No Disponible, etc
fechaCreación	Date	Fecha de alta del espacio en el sistema
usuario_id	VARCHAR	(FK) Usuarios que han reservado un espacio
reserva_id	VARCHAR	(FK) Reservas que ha recibido el espacio

EVENTO		
Campo	Tipo	Descripcion
id	VARCHAR	Id único de cada evento
entidadAsociada	VARCHAR	Describe que entidad lanzó el eveno
reservaAsociada	VARCHAR	Asocia el Id de la reserva afectada
tipoEvento	VARCHAR	"ReservaCreada","ReservaModificada", etc
fechaEvento	Date	fecha del evento
estado	boolean	Si se ha enviado a kafka



## Script creación tablas BBDD

### Tabla usuarios

```
create table usuarios(
    dni VARCHAR(20) primary key,
    nombre VARCHAR(100) not null,
    email varchar(150) unique not null,
    fecha_nacimiento date not null,
    password_hash varchar(255) not null,
    rol varchar(50) not null default 'cliente',
    activo boolean not null default 'true',
    fecha_registro timestamp not null default now(),
    reserva_id int
);
```