S.I.S. - Documentación 3.0

Documentación del proyecto propuesto como resolución del Trabajo Práctico Final de la asignatura Desarrollo de Software.

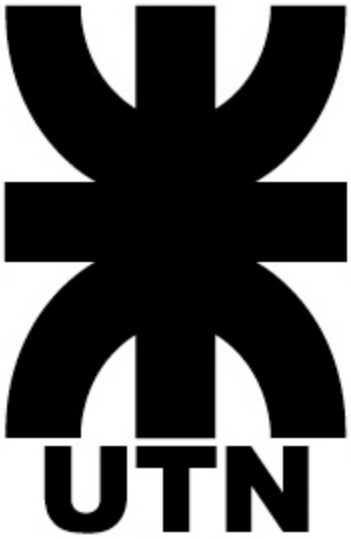
Grupo 8

Aita Jerónimo
Fernández Nicolás
Mactavish Tomás
Rugiero Fausto

Profesores

Badino Marcelo
Leibouski Ariel
Viera Sergio

Universidad Tecnológica Nacional, Facultad Regional Delta



Proyecto: Sistema Integrado de Socios (S.I.S.)

Objetivo del proyecto

Desarrollar un sistema que permita la gestión de socios y eventos de un club deportivo.

Alcance

El sistema será un sistema distribuído, siendo el servidor y el sistema de base de datos desplegado localmente con posibilidad de migración a servicio en la nube. Será controlado por un administrador dentro del club, y proveerá a él una interfaz web accesible vía navegador en computadoras de escritorio. También proveerá una interfaz web específica para socios y otra para miembros de staff.

Análisis

Realizamos un relevamiento para comprender las necesidades de información de la organización.

El mismo se documenta con los requisitos funcionales/no-funcionales y los casos de uso.

Funcionamiento del Sistema

Administración

El usuario administrador del sistema posee acceso completo al mismo. El administrador puede:

- Ver y modificar el listado de socios y los datos personales, historiales y pagos realizados de un socio.
- Ver y modificar el listado de miembros de staff y los pagos efectuados a ellos.
- Ver y modificar las actividades disponibles en el club (**clases** deportivas como básquet o fútbol, **eventos** como festivales o charlas, y **servicios** como acceso a pileta o a gimnasio).

- Configurar parámetros del sistema, como valor base de cuota, porcentaje de retención.

Registro y acceso de socios

Para ser registrado como socio, el interesado debe presentarse en el club y completar un formulario de registro. Con esta información, el administrador crea la cuenta del usuario en el sistema.
Una vez registrado, el socio puede acceder remotamente a la plataforma con sus credenciales.

Frente al sistema, el usuario socio puede:

- Ver y modificar su información personal y credenciales.
- Ver su historial de actividades y pagos realizados, y su estado actual (al día o atrasado).
- Gestionar sus inscripciones a las distintas actividades disponibles.
- Ver el valor de su cuota mensual (calculado como la suma del valor base de cuota más cargo de las actividades a las que está inscripto).
- Subir al sistema un comprobante de pago de su cuota mensual.

Con respecto al pago de la cuota, un socio puede pagar mediante transferencia bancaria o en efectivo presenciamente en el club. Por este motivo es el administrador del sistema el responsable de verificar y acreditar el pago de la cuota mensual de un socio.

Registro y acceso de miembros del staff

Se define como "miembro de staff" a un usuario que dirige una actividad dentro del club. Por ejemplo, un profesor de fútbol o un director de festival.

Para ser registrado como miembro del staff del club, el interesado debe presentarse en el club, completar un formulario con la actividad que propone dirigir y presentar su CV.

Frente al sistema, el usuario staff puede:

- Ver los listados de socios inscriptos a las actividades a su cargo.
- Ver el valor de la compensación económica mensual asociada a cada actividad a su cargo (calculada en base al número de inscriptos y el porcentaje que retiene

el club).
- Ver el historial de compensaciones recibidas.

El administrador es responsable de registrar los pagos efectuados a los miembros de staff.

Requisitos Funcionales

Definimos las capacidades y el comportamiento que el sistema desarrollado deberá cumplir.

"El sistema debe permitir..."

Al administrador:

RF1: Registrar un nuevo socio o miembro de staff, con información personal básica (nombre, apellido, DNI, correo, teléfono).
RF2: Modificar información de un socio o miembro de staff.
RF3: Dar de baja socios o miembros de staff (manteniendo su historial).
RF4: Consultar el historial de un socio (info. personal, pagos realizados, actividades a las que estuvo inscripto, valor de cuota mensual, estado de pago de cuota).
RF5: Registrar el pago de la cuota mensual de un socio.
RF6: Consultar el estado de pago de cuota de un socio (al día o atrasado; para definir rápido si puede o no acceder al club).
RF7: Publicar una actividad (descripción, cargo por inscripción, fechas/horarios y miembro del staff a cargo)
RF8: Modificar actividad publicada, generando un aviso automático a sus inscriptos (email a socios inscriptos).
RF9: Archivar actividad finalizada.
RF10: Validar comprobante de pago subido por un socio.
RF11: Registrar el pago efectuado a un miembro de staff.

Al socio:

RF12: Consultar su historial (información personal, pagos realizados, actividades a las que está inscripto, valor de cuota mensual, estado de pago de cuota).
RF13: Modificar su información personal.
RF14: Subir comprobante de pago.
RF15: Inscribirse a una actividad entre las disponibles.

Al miembro de staff:

RF16: Consultar listado de socios inscriptos a las actividades a su cargo.
RF17: Consultar monto a cobrar por los eventos a su cargo.
RF18: Consultar las actividades a su cargo iniciadas, finalizadas, archivadas o

eliminadas.
RF19: Consultar historial de compensaciones.

Tareas del sistema:

RF20: El sistema debe generar recordatorios de pago a los socios (email automático al socio).

Requisitos no-funcionales

"El sistema debe cumplir con..."

Rendimiento:

RNF1: Tiempos de respuesta menores a 2s para el 95% de las solicitudes bajo carga normal (consultas).
RNF2: Tiempos de respuesta menores a 30s para el 100% de las solicitudes bajo carga alta (subida/bajada de archivos).
RNF3: Consultas indexadas y formato `webp` en imágenes con el fin de optimizar comunicaciones.

Disponibilidad:

RNF4: Disponibilidad (*uptime*) del 99.9% (máximo de 8,76 horas de inactividad por año).
RNF5: Copias de respaldo (*backups*) diarias de la base de datos.

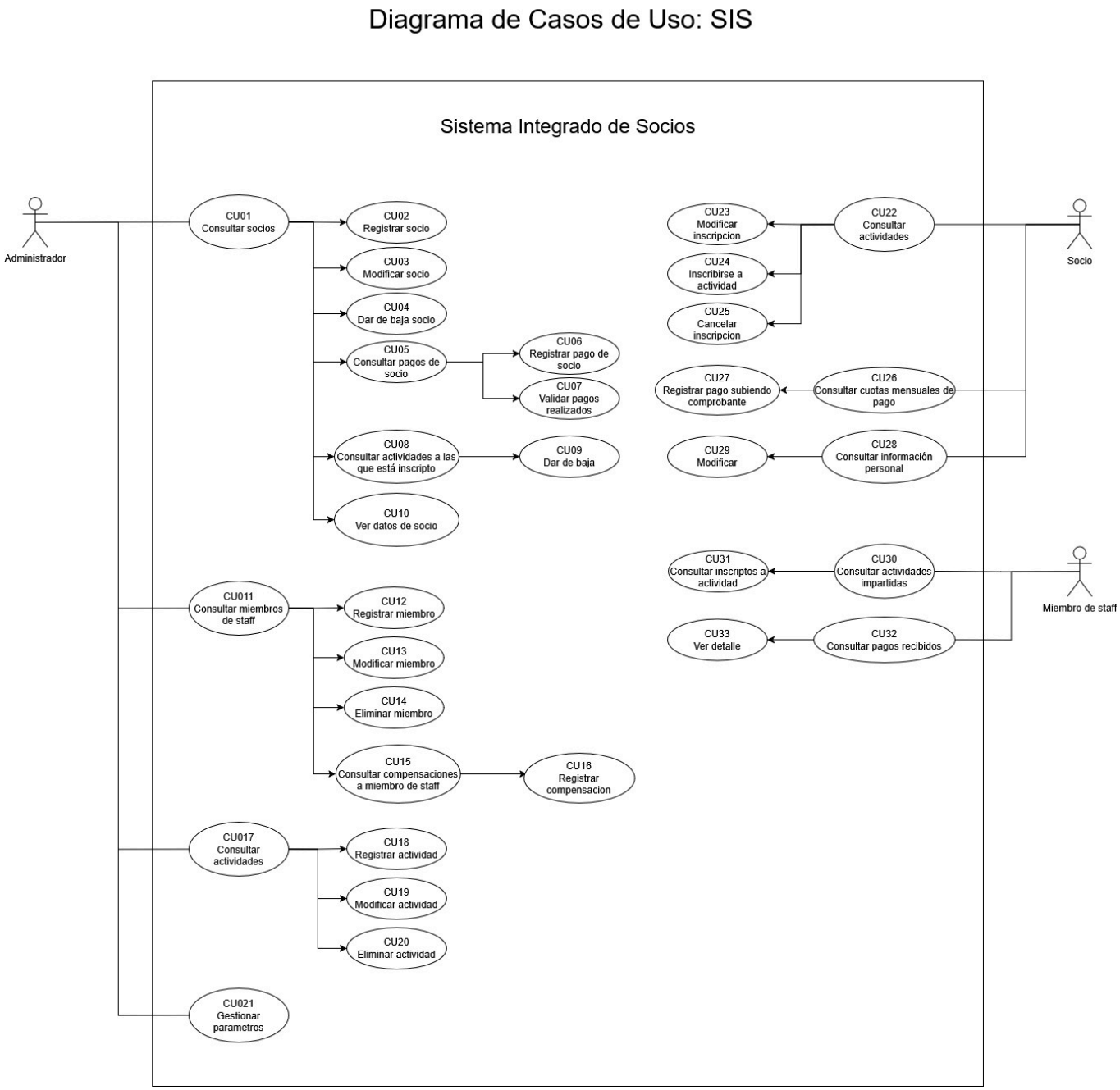
Seguridad:

RNF6: Implementación de autenticación JWT y perfiles de usuario.
RNF7: Implementación de comunicación HTTPS.
RNF8: Implementación de cifrado de contraseñas Bcrypt.
RNF9: Inclución de campos de auditoría en la base de datos.

Usabilidad:

RNF10: Interfaces web simples e intuitivas.

Diagrama de Casos de Uso



Principales Casos de Uso

Algunos casos de uso fundamentales:

Caso de uso:	Consultar socios - CU001	
Actor	Administrador	
Descripción	Consultar listado de socios	
Flujo normal	Usuario: Click en buscar	El sistema muestra en pantalla el listado de socios

Caso de uso:	Registrar socio - CU002	
Actor	Administrador	
Descripción	Permite al administrador del sistema dar de alta nuevos socios ingresando información del mismo como nombre y apellido , direccion , telefono ,correo electronico , DNI, entre otros.	
Flujo normal	Usuario: Presiona el botón de Registrar socio	Se muestra en pantalla un formulario con los campos en blanco requeridos para dar de alta un socio.

Caso de uso:	Registrar nuevo miembro - CU012	
Actor	Administrador	
Descripción	Permite al administrador del sistema dar de alta nuevos Profesores/Directores ingresando informacion del mismo como nombre y apellido , direccion , telefono ,correo electronico , DNI, entre otros.	
Flujo normal	Usuario: Presiona el botón de Registrar Profesor/Director	Se muestra en pantalla un formulario con los campos en blanco requeridos para dar de alta un Profesor/Director.

Diseño

Una vez hecho el análisis, aplicamos conceptos de diseño para modelar la solución. La misma se documenta con diagramas:

- Diagrama de dominio
- Diagrama entidad-relación
- Diagrama de arquitectura
- Diagrama de despliegue
- Diagrama de componentes

También se documenta el diseño con prototipos para las pantallas y con diagramas de flujo/diagramas de estados para solución de funcionalidades específicas.

Diagrama de dominio

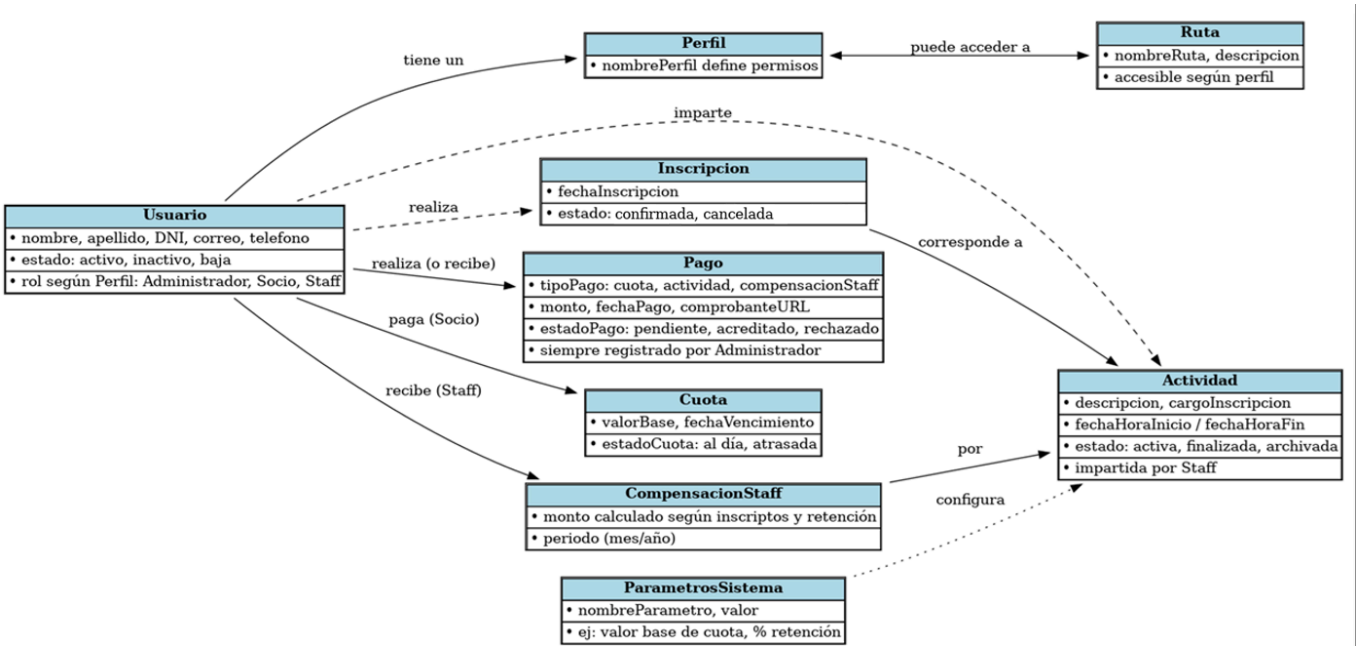
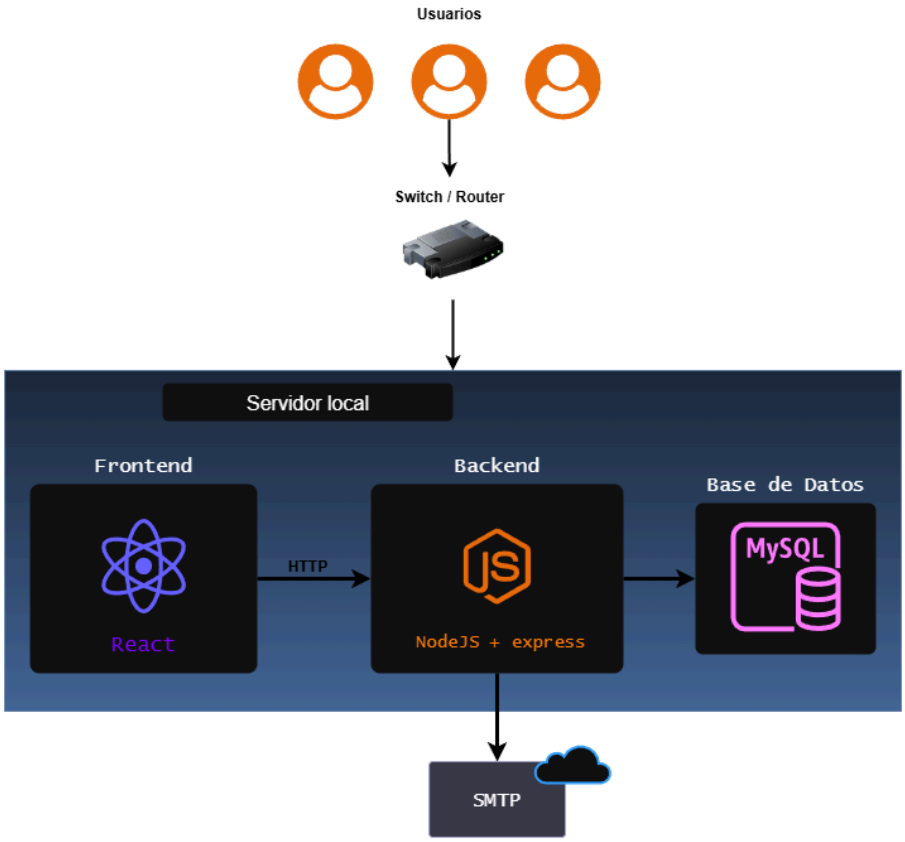


Diagrama de arquitectura

La arquitectura del software estaría basada en una SPA, con una API REST como back-end.

Nuestra primer aproximación fue utilizar el sistema gestor de bases de datos MySQL, el framework Express.JS para programar el back-end, y el framework React para programar el front-end:



Pero luego de la etapa de diseño del sistema decidimos cambiar Express.JS por Django, al conocer más en detalle las herramientas que provee y descubrir que es un framework más robusto.
Volvimos a la etapa de diseño y cambiamos la arquitectura:

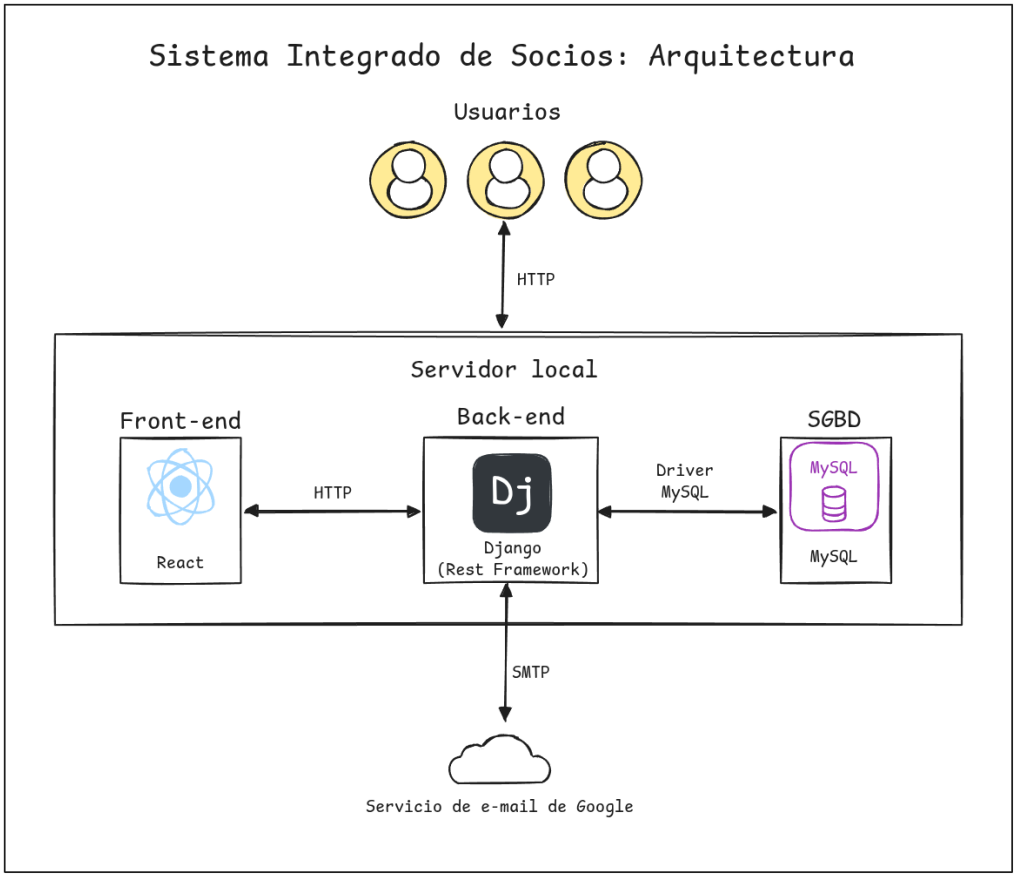
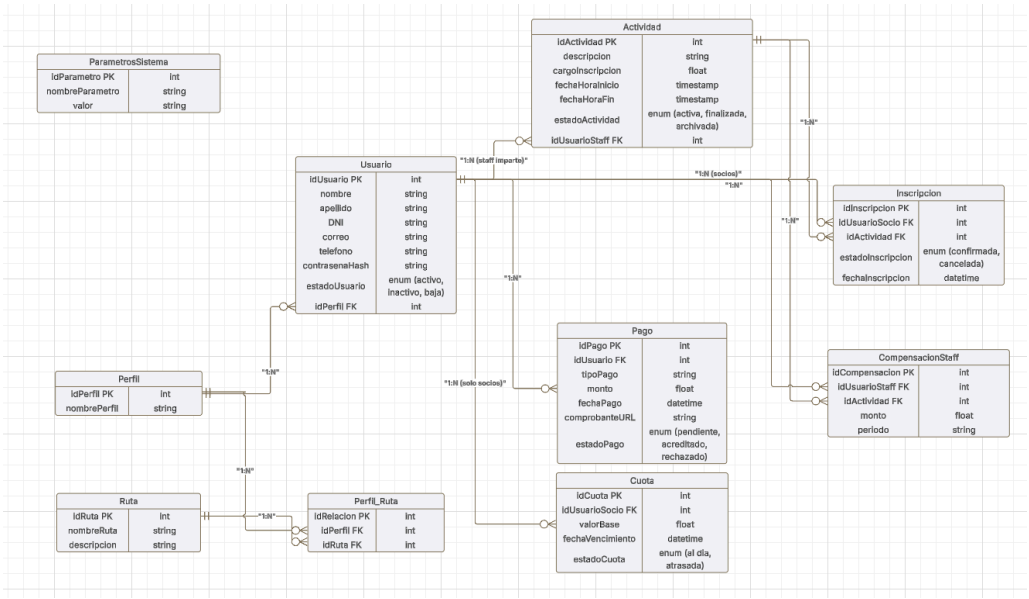

Sistema Integrado de Socios: Arquitectura
Usuarios
HTTP
Servidor local
Front-end
React
HTTP
Back-end
Dj
Django
(Rest Framework)
Driver
MySQL
SGBD
MySQL
MySQL
SMTP
Servicio de e-mail de Google

Diagrama entidad-relación

La primer aproximación al diagrama de entidad-relación:



Luego de cambiar Express.JS por Django, y considerando que Django sigue los principios DRY, re-escribimos el diagrama de forma que refleje uno a uno cómo se definen los modelos en Django.
Además, evitamos modelar perfiles de usuario, al ser que Django provee "grupos de usuario" que cumplen el mismo propósito:

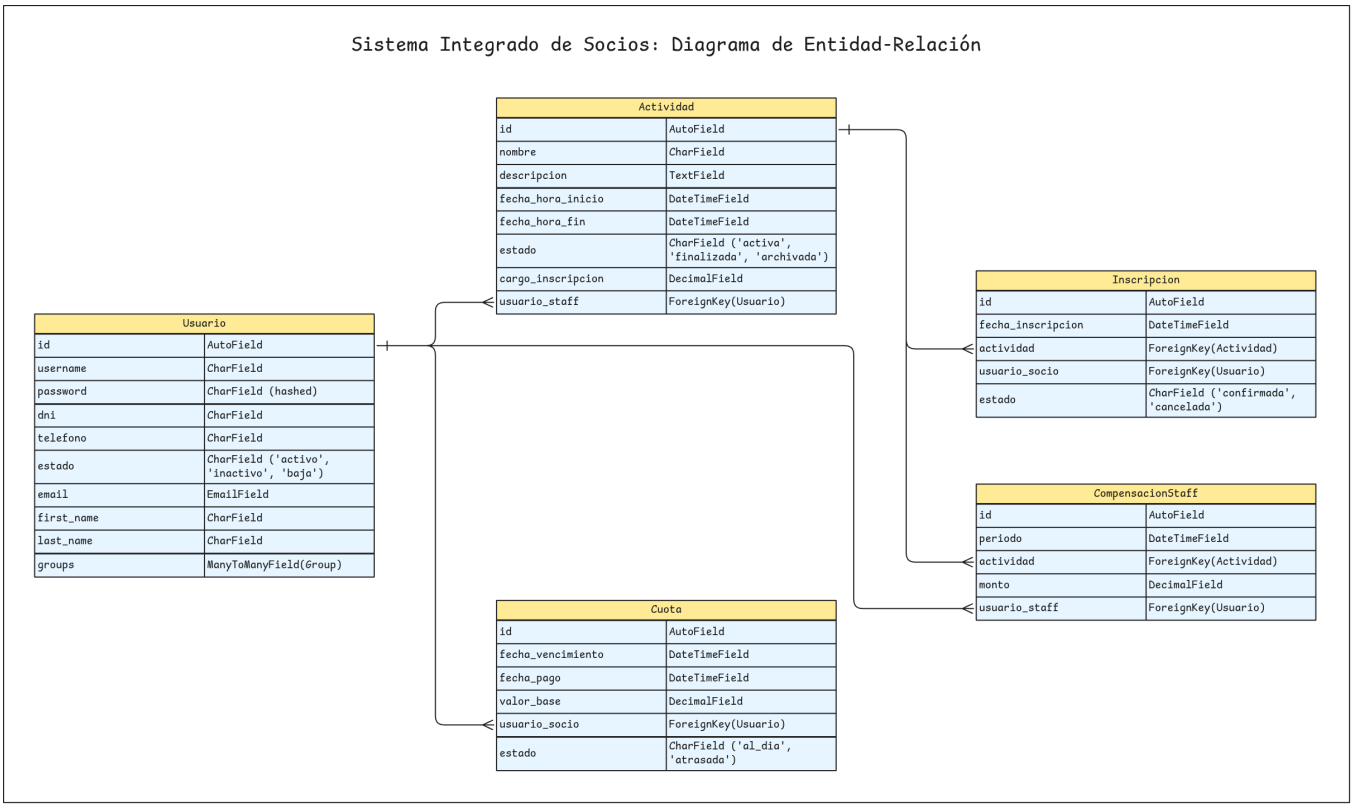
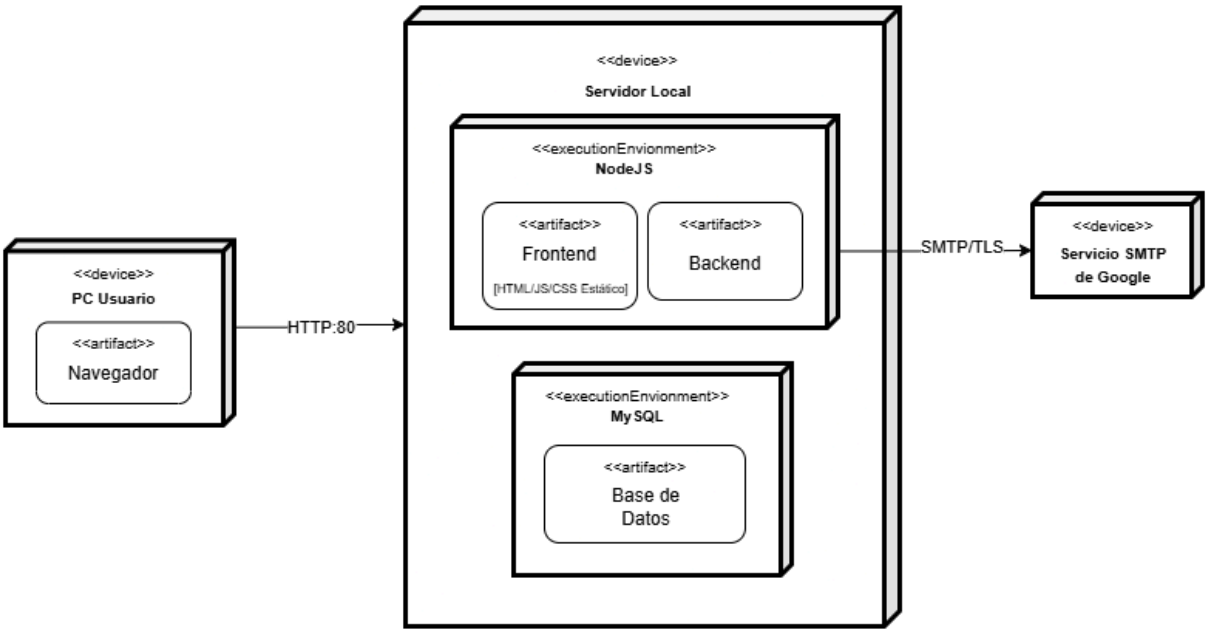


Diagrama de despliegue

Primer aproximación:



Ajustado a Django: