Práctica 5: Deployment of a Web Information System with data persistence byusing Docker containers

Grado en Ingeniería Informática (Plan 439)

Sistemas de información
Curso: 2025-2026

Sergio Saura Oliva (838585)
José Secadura Del Olmo (815327)
Chloé Bolle-Reddat (961562)

Fecha de entrega: 27/11/2025

Universidad de Zaragoza - Escuela de Ingeniería y Arquitectura

Resumen de la Práctica 5

La Práctica 5 es la última parte del proyecto desarrollado a lo largo de la asignatura de Sistemas de Información. En ella se integran y actualizan los resultados obtenidos previamente en las Prácticas 1, 2, 3 y 4, y se completa el trabajo mediante el despliegue de la aplicación real utilizando contenedores Docker.

La aplicación desarrollada, denominada **Fylt**, es una plataforma web orientada al entretenimiento audiovisual. Permite consultar información actualizada sobre películas y series, gestionar listas personalizadas, escribir comentarios, participar en encuestas y acceder a un panel de administración con privilegios avanzados. La información proviene tanto de una base de datos interna como de la API externa TMDB y de YouTube para mostrar los tráilers.

Índice

1. Práctica 1 - Objetivo y alcance funcional final de la aplicación

La versión final de la aplicación cumple bastante lo desarrollado durante la Práctica 1 y la actualiza de acuerdo con el desarrollo real llevado a cabo. Fylt se presenta como una plataforma unificada para la consulta, organización y valoración de contenido audiovisual procedente de la API TMDB y almacenado de forma persistente en una base de datos PostgreSQL.

Objetivos del proyecto (actualizados)

Los objetivos principales de la aplicación son:

1. Facilitar el acceso a información actualizada sobre películas, incluyendo sinopsis, géneros, fecha de estreno y material promocional.
2. Permitir la gestión de contenidos favoritos y listas personalizadas para mejorar la organización individual del usuario.
3. Centralizar la información sobre películas, permitiendo que los usuarios interactuen mediante comentarios, y me gustas.
4. Integrar funcionalidades sobre películas

Funcionalidades del sistema (actualización desde P1)

Fylt proporciona las siguientes funcionalidades finales:

1. Registro e inicio de sesión de usuarios, incluyendo roles diferenciados (usuario y administrador).
2. Gestión de películas favoritas y listas personalizadas.
3. Consulta de información detallada sobre películas (título, valoración, géneros, actores, sinopsis, imagen).
4. Sistema de comentarios con moderación por parte del administrador.
5. Gestión de encuestas y votaciones.
6. Exploración mediante búsqueda avanzada con filtros (género, año, popularidad).
7. Panel de administración para supervisión, creación y edición de claves API.
8. Sistema de seguidores entre usuarios (perfil social básico).

2. Práctica 2 - Enfoque del sistema y storyboard de la aplicación

2.1. Metodología usada para el prototipado

El storyboard inicial se desarrolló utilizando la herramienta Moqups, seleccionada por su versatilidad y rapidez para crear prototipos funcionales. El equipo dividió el trabajo en dos bloques: diseño de interfaces de usuario y diseño del panel de administración.

2.2. Storyboard del administrador actualizado

El prototipo del administrador incluye:

- Una vista del control de comentarios, con paginación y controles para aprobar o rechazarlos.
- Un panel para la creación y gestión de encuestas, permitiendo recoger los datos y activando o desactivandolas.
- Una sección destinada a la edición de claves API externas.
- Gestión de las películas permitiendo editarlas, eliminarlas o importar nuevas.

2.3. Storyboard del usuario actualizado

El diseño para el usuario ha sido:

- Una página principal con películas recomendadas y otras de un género aleatorio, incluye también un tráiler de una película aleatorio para poder descubrir nuevo cine.
- Una sección de comunidad con los comentarios más populares y los de tus amigos, también las encuestas.
- Una vista de búsqueda avanzada con varios filtros.
- El perfil del usuario, incluyendo listas creadas, seguidores, seguidos y comentarios.

El diseño final de los storyboard han seguido lo planteado en la práctica 2 con cambios mínimos, como la inclusión de una página amigos para poder buscar nuevos y ver los que ya tienes, también se ha eliminado del prototipo final el ranking y actividades debido a que por tiempo no hha sido posible.

2.4. Storyboard de autenticación

El storyboard incluía también:

- Pantalla de inicio de sesión con opciones de autenticación y enlace al registro.
- Pantalla de registro con formulario completo.

2.5. Mapa de navegación general del Usuario y Administrador

En la aplicación Fylt el usuario puede acceder a un amplio conjunto de funcionalidades. El mapa de navegación completo del usuario se encuentra disponible en el **Anexo A**.

El administrador dispone de rutas adicionales para gestionar películas, revisar comentarios y crear encuestas. El mapa detallado de navegación del administrador puede consultarse también en el **Anexo A**.

3. Práctica 3 - Modelo de datos del sistema

El modelo de datos presentado originalmente en la Práctica 3 fue ampliado y adaptado durante el desarrollo final, debido a que mediante se iba codificando la aplicación nos surgoieron varios problemas. A continuación se describen los cambios más relevantes entre el modelo inicial y el modelo definitivo:

Cambios principales respecto al DDL original

- **La tabla usuario ahora incluye el campo** `email` como clave única del sistema. En el modelo inicial no existía este atributo.
- La tabla **encuesta** incorpora el campo `pregunta` y el campo booleano `activo`, permitiendo controlar encuestas activas/inactivas.
- La tabla **pelicula** incluye ahora el atributo `generos varchar` para almacenar los géneros y permitir las búsquedas.
- La tabla **comentario** añade dos atributos nuevos: `visible` y `aprobado`, necesarios para la moderación por parte del administrador.
- Se incorpora la tabla **comentario_like** para gestionar los me gusta en los comentarios.
- Se añade la tabla **encuesta_voto**, permitiendo registrar votaciones por usuario y encuesta.

- La tabla **api_key** se amplía añadiendo el campo `nombre` para identificar el servicio asociado.
- No se implementó finalmente el sistema completo de recomendaciones basado en **actividades**, aunque la tabla continúa presente.

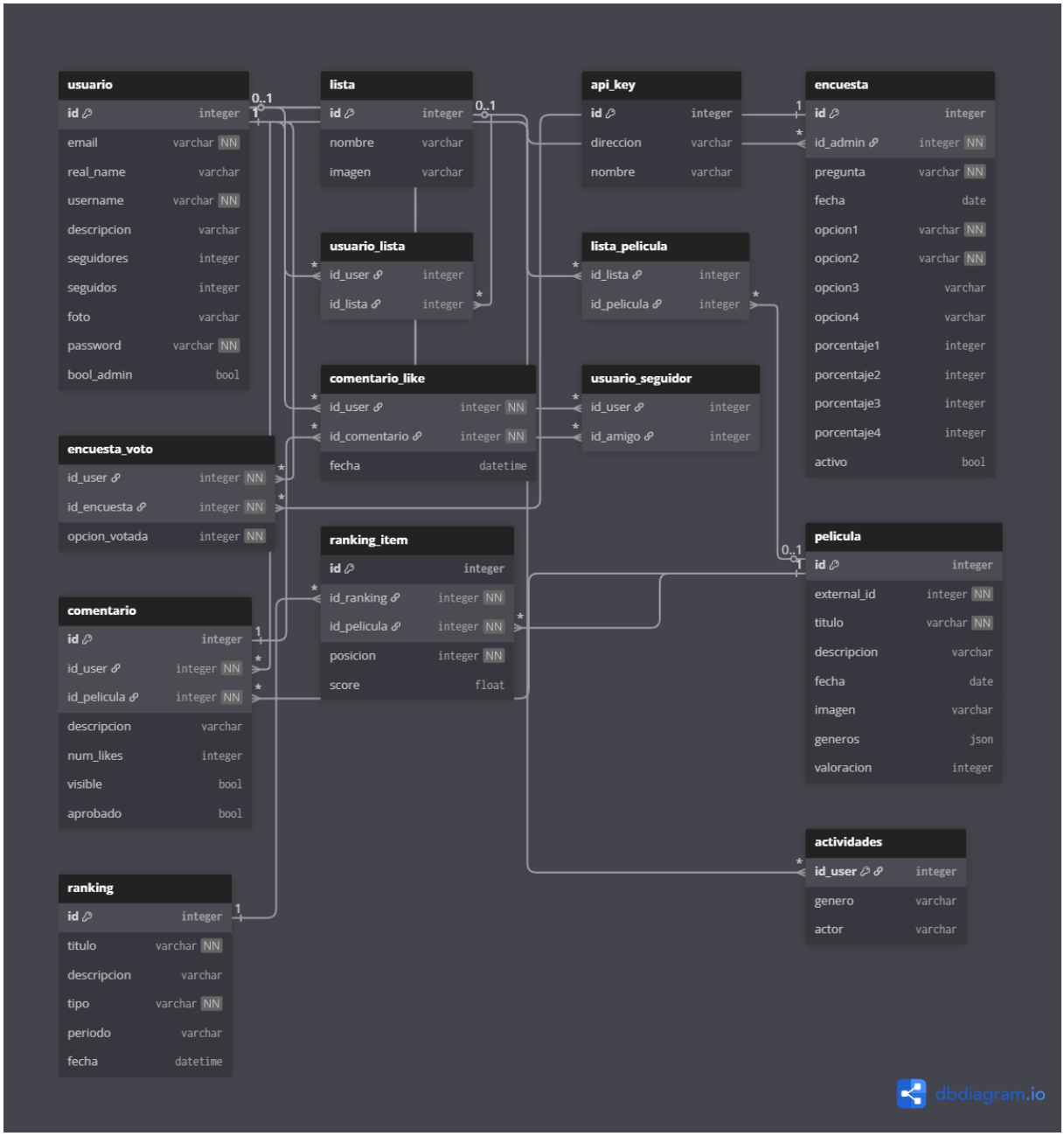

Figura 3.0.1: Diagrama de e/r

4. Diferencias entre la versión presentada en la Práctica 1 y la versión final de la aplicación

A lo largo del desarrollo del proyecto, la aplicación ha evolucionado significativamente respecto al diseño principal presentado en la Práctica 1. Aunque la idea general se ha mantenido, muchas funcionalidades fueron modificadas en función del tiempo disponible y la carga de trabajo, debido a que la creación de nuestra propia API y la conexión con APIS internas se nos dificulto un poco.

Resumen del proyecto (Práctica 1 vs versión final)

En la Práctica 1 se definió Fylt como una plataforma completa para acceder a información actualizada sobre películas y series, incluyendo estrenos, valoraciones de usuarios, listas personalizadas, recomendaciones y una comunidad activa. La versión final mantiene el enfoque del proyecto, pero se han realizado ajustes:

- El sistema final se centra únicamente en **películas**, descartando las series por limitaciones de tiempo y complejidad técnica.
- Las recomendaciones basadas en historial y gustos no se implementaron completamente, ya que dependían de una tabla de actividades que finalmente no se ha logrado implementar en la versión final.
- La interfaz mantiene la estructura base (home, comunidad, búsqueda, perfil), pero se amplió con el campo amigos, así como la integración de los perfiles de otros usuarios.

Necesidad a resolver (revisión final)

El problema central definido en la Práctica 1 se mantiene: ofrecer un punto centralizado para consultar información audiovisual y permitir interacción social mediante reseñas, favoritos y listas personalizadas. Sin embargo, la implementación final:

- Se apoya en la API TMDB para tener una fuente fiable de datos sobre películas en la actualidad.
- Simplifica las funciones avanzadas de comunidad (no hay mensajes directos, recomendaciones complejas ni sistema de actividad detallado).

Funcionalidades descartadas o modificadas

Comparando con el documento original:

- Se descartó el **historial de visualización**.
- Se eliminaron los **episodios y temporadas**, ya que la aplicación no trabaja finalmente con series.
- No se han implementado las **recomendaciones personalizadas** de películas.
- La funcionalidad de "plataformas donde ver el contenido" no se implementó.

Funcionalidades añadidas respecto a P1

La versión final incluye algunas características que no estaban contempladas originalmente:

- Página específica para **buscar y añadir amigos**.
- Sistema de **likes** en los comentarios.
- Página específica en el administrador para ver **comentarios aprobados y pendientes**.
- Sistema de moderación completo para administradores, incluyendo visibilidad y aprobación manual.

Conclusión de la evolución

En resumen, la aplicación final conserva la base planteada en la Práctica 1, pero se adapta a un enfoque más realista y viable dentro del tiempo disponible, centrándose en la gestión de películas, listas, comunidad básica y administración, mientras que algunas funcionalidades más complicadas se han tenido que descartar.

5. Procedimiento para el despliegue e instalación de la aplicación

El despliegue final de la aplicación Fylt se realiza utilizando contenedores Docker tanto para el frontend como para el backend. Para ello, se emplea Docker Desktop y un fichero `docker-compose.yml` que automatiza la construcción y la ejecución de ambos servicios.

La aplicación queda desplegada en dos contenedores independientes:

- **Frontend**: desarrollado en Next.js.
- **Backend**: implementado en .NET 9.

La base de datos utilizada por la aplicación es PostgreSQL alojada en la plataforma Neon, por lo que no es necesario desplegar ningún motor de base de datos adicional en Docker.

5.1. Requisitos previos

Para ejecutar el despliegue es necesario disponer de Docker en nuestro dispositivo.

No es necesario tener instalado ni Node.js ni .NET localmente, ya que la construcción se realiza dentro de las imágenes Docker.

5.2. Dockerfile del backend (.NET)

El backend se construye a través de un `Dockerfile` multietapa que compila la solución en modo *Release* y posteriormente publica la API lista para producción.

```
# Base runtime
FROM mcr.microsoft.com/dotnet/aspnet:9.0 AS base
WORKDIR /app
EXPOSE 7052

# Build stage
FROM mcr.microsoft.com/dotnet/sdk:9.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src

# Copiamos los csproj para aprovechar la cache
COPY ["Fylt.Api/Fylt.Api.csproj", "Fylt.Api/"]
COPY ["Fylt.Domain/Fylt.Domain.csproj", "Fylt.Domain/"]
COPY ["Fylt.Infrastructure/Fylt.Infrastructure.csproj", "Fylt.Infrastructure/"]

RUN dotnet restore "Fylt.Api/Fylt.Api.csproj"

# Copiamos todo el código
COPY . .

WORKDIR "/src/Fylt.Api"
RUN dotnet build "Fylt.Api.csproj" -c $BUILD_CONFIGURATION -o /app/build

# Publish stage
FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish "Fylt.Api.csproj" -c $BUILD_CONFIGURATION -o /app/publish
 ↪  /p:UseAppHost=false

# Final image
FROM base AS final
```

```
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "Fylt.Api.dll"]
```

5.3. Dockerfile del frontend (Next.js)

El frontend se construye mediante un `Dockerfile` multietapa que instala dependencias, compila el proyecto y crea un entorno de ejecución optimizado.

```
FROM node:20-alpine AS deps
RUN apk add --no-cache libc6-compat
WORKDIR /app

COPY package.json package-lock.json* ./
RUN npm ci --legacy-peer-deps

FROM node:20-alpine AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .

ENV NEXT_TELEMETRY_DISABLED=1
ENV NODE_ENV=production

RUN npm run build

FROM node:20-alpine AS runner
WORKDIR /app

ENV NODE_ENV=production
ENV NEXT_TELEMETRY_DISABLED=1

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public
COPY --from=builder /app/.next/standalone ./
COPY --from=builder /app/.next/static ./.next/static

RUN chown -R nextjs:nodejs /app
USER nextjs

EXPOSE 3000
ENV PORT=3000
ENV HOSTNAME="0.0.0.0"

CMD ["node", "server.js"]
```

5.4. Fichero `docker-compose.yml`

Para levantar ambos contenedores de forma simultánea se utiliza el siguiente fichero:

```yaml
version: '3.8'

services:
  frontend:
    container_name: fylt_web
    build:
      context: ./react_typescript
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "wget", "--no-verbose", "--tries=1", "--spider",
      ↪  "http://localhost:3000/"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s

  backend:
    container_name: fylt_api
    build:
      context: ./api_net
      dockerfile: Fylt.Api/Dockerfile
    ports:
      - "7052:8080"
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "wget", "--no-verbose", "--tries=1", "--spider",
      ↪  "http://localhost:7052/"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s
```

Ambos servicios se construyen automáticamente a partir de sus respectivos `Dockerfile` y quedan ejecutándose en segundo plano.

5.5. Script de despliegue

Para simplificar el proceso, se creó un script `bash` que lanza el despliegue completo:

```bash
#!/bin/bash
```

```
2
3 docker-compose up --build -d
```

Este script construye las imágenes y levanta los contenedores en modo *detached*.

5.6. Acceso a la aplicación desplegada

Una vez finalizado el proceso, la aplicación queda accesible de la siguiente forma:

- **Frontend**: `http://localhost:3000`
- **Backend**: `http://localhost:7052`

5.7. Conclusión

El uso de Docker y `docker-compose` permite desplegar la aplicación de forma reproducible y aislada, evitando tener dependencias locales y asegurando por lo tanto que el entorno es idéntico en todos los equipos así no surgen problemas debidos a dependencias o programas.

6. Información necesaria para usar la aplicación

Esta sección recoge los datos indispensables para que un corrector o usuario pueda acceder y probar la aplicación una vez desplegada.

Acceso al frontend y backend

Como se ha mencionado en el apartado anterior al desplegar en docker las direccones que nos quedan para el front y el back son:

- **Frontend**: `http://localhost:3000`
- **Backend**: `http://localhost:7052`

Cuentas incluidas para evaluación

Cuenta Administrador

- Email: `admin@fylt.com`

- Contraseña: `admin123`
- Permisos: acceso total al panel de administración, moderación de comentarios, gestión de encuestas y API keys.

Usuario normal

- Usuario 1:
  - Email: `user@fylt.com`
  - Contraseña: `user123`

Método de autenticación

La autenticación se basa en un sistema tradicional de:

- Email + contraseña.
- Encriptación segura de contraseñas en base de datos mediante BCript en .NET.
- Tokens JWT emitidos por el backend .NET para autorizar las peticiones.

Los roles se gestionan internamente mediante un atributo booleano en la tabla de Usuarios `bool_admin`.

Requisitos adicionales

La aplicación necesita:

- Una API Key válida de TMDB para poder rellenar la base de datos con información de películas.
- Una API Key válida de YouTUbe para poder buscar los tráilers de las películas.

Estas variables ya están registradas en la bd y se controlan mediante el Administrador del sistema.

7. Cronograma aproximado, dificultades y evaluación grupal

Cronograma aproximado

El desarrollo del proyecto se ha llevado a cabo a lo largo de todas las prácticas de la asignatura, siguiendo una progresión incremental tanto en complejidad como en volumen de trabajo. El esfuerzo total aproximado del equipo ha sido el siguiente:

- **Práctica 1 (Análisis y requisitos)**: 15 horas

  Estudio del problema, definición de usuarios, necesidades, alcance y objetivos del sistema.

- **Práctica 2 (Storyboard y diseño de interfaces)**: 32 horas

  Elaboración del prototipo visual, definición del flujo de navegación y diseño preliminar de las pantallas de usuario y administrador.

- **Práctica 3 (Modelo de datos y DDL)**: 25 horas

  Modelado E/R, normalización, creación del esquema, entidades y relaciones, índices y restricciones.

- **Práctica 4 (Implementación funcional)**: 98 horas Desarrollo del backend con .NET, integración con Angular, lógica de negocio, panel de administración, sistema de comentarios, encuestas y conectividad con la API TMDB.

- **Práctica 5 (Despliegue y memoria final)**: 36 horas Este apartado también incluye la finalización de implementaciones que quedaron pendientes en la práctica 4, además de la creación de imágenes Docker, despliegue completo, configuración de variables de entorno, pruebas y redacción de la memoria final.

Cuadro 1: Horas dedicadas por cada miembro del grupo en la práctica 5

Miembro	**Horas**
Chloé Bolle-Reddat	6 h
Sergio Saura Oliva	15 h
José Secadura Del Olmo	15 h
Total	**36 h**

En total, el esfuerzo conjunto del proyecto ronda las **206 horas de trabajo** sumando los tres integrantes del grupo.

Dificultades encontradas durante el desarrollo

En el proyecto hemos tenido varios problemas relacionados con .Net y Angular que son:

- **Integración Angular–.NET**: Configurar correctamente CORS y las rutas para los endpoints en ambos entornos generó numerosos problemas al inicio, sobretodo porque nadie del equipo había trabajado anteriormente con estas tecnologías.
- **Moderación de comentarios**: El modelo original no contemplaba visibilidad/revisión. Esto implicó añadir los campos `visible` y `aprobado` y modificar consultas y endpoints, además de incluir un diccionario con algunas palabras para que se rechazaran directamente.
- **Cambios del storyboard a Angular**: Algunas pantallas eran demasiado complejas para el tiempo disponible y debieron simplificarse respetando la esencia del diseño inicial.
- **Datos incompletos de TMDB**: Hubo que adaptar el modelo para almacenar bien los datos que nos devolvía la llamada a la API TMDB, además de una vez obtenidos los datos conseguir de la API de YouTUbe el enlace para el tráiler de esa película.
- **Problemas en el despliegue de docker**: El back fué más fácil de lanzar en docker aunque tuvimos problemas con los certtificados como el de http y en el puerto 7052, con el front no tuvimos problemas destacables.

Lecciones aprendidas

El proyecto ha permitido al equipo aprender diferentes y cosas como:

- Importancia de diseñar un **modelo de datos flexible** que permita cambios sin romper el sistema.
- Cómo integrar correctamente angular con .NET, además de diseñar nuestros propios endpoints.
- Uso de **Docker** para despliegar la aplicación.
- Trabajo colaborativo con control de versiones con github y organización mediante tareas con to do.
- Adaptación del diseño a las limitaciones reales de tiempo y tecnología.

Funcionalidades que nos habría gustado implementar

A pesar de que la aplicación final es estable y completa para nuestro diseño inicial, se identifican varias características planteadas que no se han podido implementar:

- **Sistema de recomendaciones personalizadas**: El modelo incluía una tabla de actividades (género favorito, actor favorito), pero no se llegó a desarrollar por tiempo.

- **Comunicación entre perfiles de amigos**: Se planteó permitir que amigos pudieran enviarse mensajes o interactuar directamente en tiempo real, pero se descartó por falta de tiempo y complejidad.
- **Gestión de series además de películas**: Aunque el planteamiento original incluía series, su integración duplicaba el volumen de datos, endpoints y lógica, por lo que finalmente se decidió centrarse solo en películas.
- **Encuestas creadas por usuarios**: Inicialmente se pensó que cualquier usuario pudiera proponer encuestas. Finalmente esta funcionalidad quedó exclusivamente en manos del administrador para simplificar la gestión de votos y control de las mismas.
- **Estadísticas avanzadas del perfil**: Gráficas de géneros más vistos, actividad semanal y evolución de gustos fueron ideas descartadas por sobrecarga de trabajo.

Conclusión general

El resultado final de Fylt refleja una versión funcional y razonablemente completa del proyecto planteado al inicio de la asignatura. Aunque algunas características previstas como recomendaciones avanzadas, integración de series o la comunicación entre usuarios no han podido implementarse por falta de tiempo, se ha logrado desarrollar un sistema estable con las funcionalidades esenciales: gestión de películas, listas, comentarios, encuestas y un panel de administración operativo.

El proceso ha supuesto un aprendizaje significativo en lenguajes como angular y .NET, además de la integración frontend–backend, uso de APIs externas y despliegue con Docker, permitiendo al equipo enfrentarse a problemas reales de desarrollo.

Bibliografía

- **NUnit Documentation**. NUnit. Disponible en: https://docs.nunit.org/
- **Unit Testing in .NET with NUnit**. Microsoft Learn. Disponible en: https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-csharp-with-nunit
- **The Movie Database (TMDB) API – Getting Started**. TMDB Developers. Disponible en: https://developer.themoviedb.org/reference/getting-started
- **Help with Movie Database API Genres**. Reddit /r/webdev. Disponible en: https://www.reddit.com/r/webdev/comments/16vbvni/help_with_the_movie_database_api_genres/
- **.NET Documentation**. Microsoft. Disponible en: https://learn.microsoft.com/es-es/dotnet/
- **.NET Tag**. StackOverflow. Disponible en: https://stackoverflow.com/questions/tagged/.net
- **Neon Database**. Neon. Disponible en: https://neon.com/
- **Neon Docs – Introduction**. Neon. Disponible en: https://neon.com/docs/introduction
- **TailwindCSS**. Disponible en: https://tailwindcss.com/
- **React + TypeScript Learn**. React Docs. Disponible en: https://react.dev/learn/typescript
- **TypeScript Handbook: React**. TypeScript. Disponible en: https://www.typescriptlang.org/docs/handbook/react.html
- **Introduction to Node.js**. Node.js Foundation. Disponible en: https://nodejs.org/es/learn/getting-started/introduction-to-nodejs
- **ChatGPT**. OpenAI. Disponible en: https://chatgpt.com/
- **RealWorld App Example**. GitHub – gothinkster. Disponible en: https://github.com/gothinkster/realworld

Anexos

A. Mapas de navegación de la aplicación

En este anexo se incluyen los mapas de navegación correspondientes al usuario normal y al administrador. Debido a su tamaño y orientación, se presentan en formato vertical para mejorar su legibilidad.

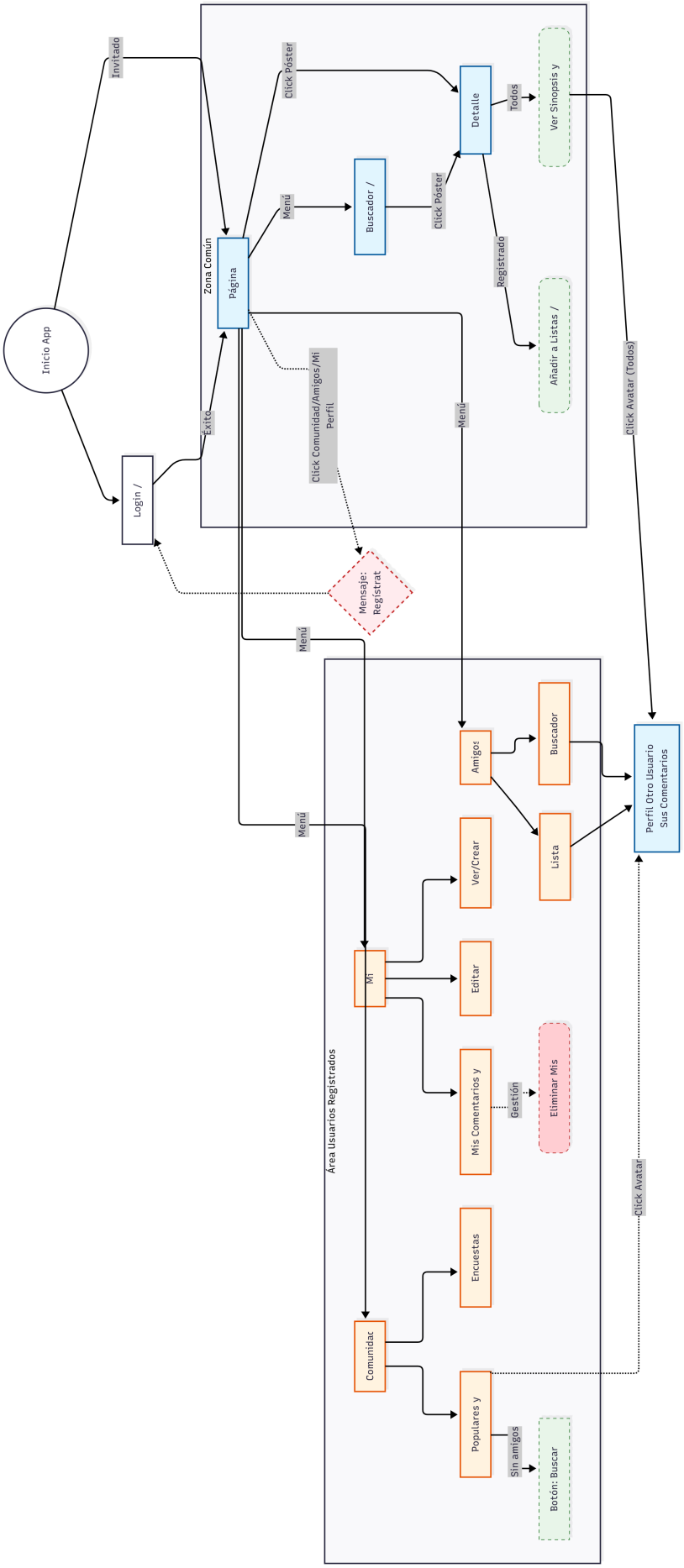

Figura A.0.1: Mapa de navegación completo del Usuario en Fylt

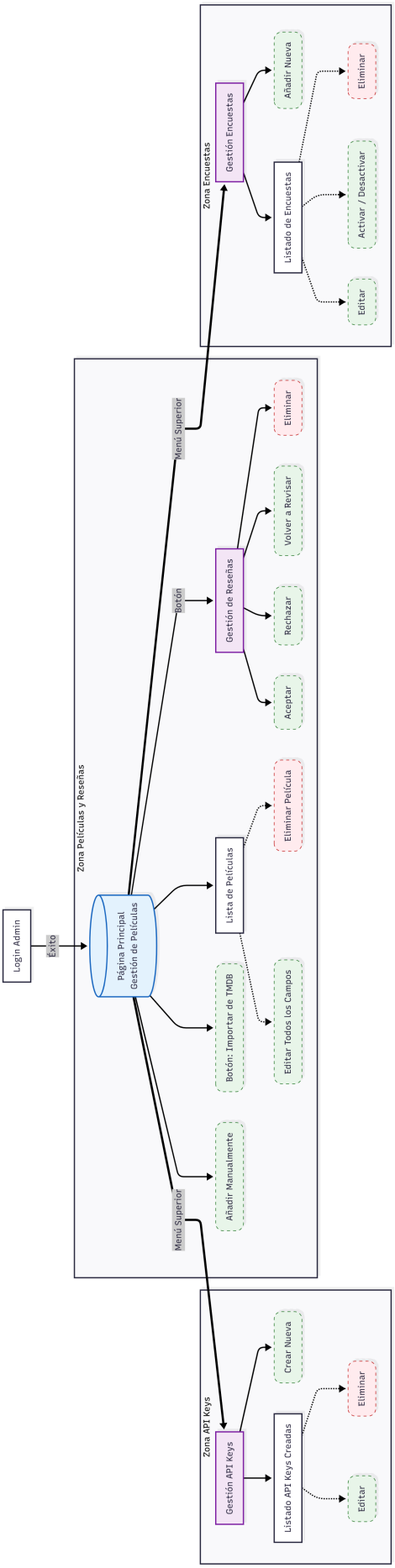

Figura A.0.2: Mapa de navegación del Administrador en Fylt