Reporte Final - Reto de Desarrollo Módulo de Recetas

Juan Antonio González

April 26, 2025

Índice

1. Introducción

Este documento presenta el reporte final del reto de desarrollo enfocado en la creación de un módulo de gestión de recetas de cocina. El objetivo es describir el alcance del proyecto, la arquitectura implementada, las tecnologías utilizadas y las funcionalidades desarrolladas, reflejando el estado final de la aplicación web.

2. Descripción del Reto

El desafío consistió en desarrollar una aplicación web completa para la gestión de recetas de cocina. El sistema debía permitir a los usuarios realizar operaciones **CRUD** (Crear, Leer, Actualizar, Eliminar) sobre las recetas y sus ingredientes asociados, además de funcionalidades adicionales como búsqueda y marcado de favoritos.

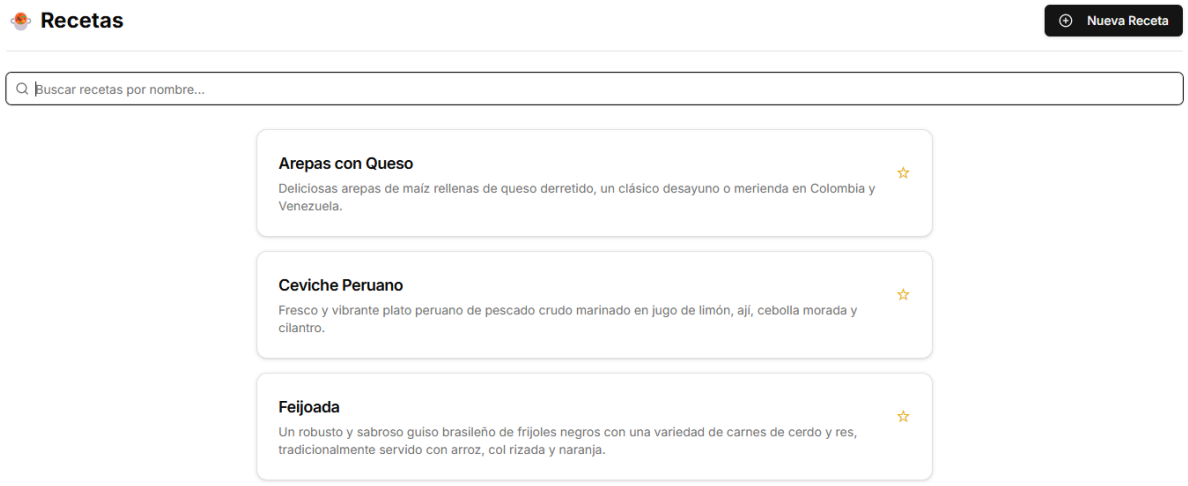

Figura 1: Captura de pantalla de la interfaz de usuario

3. Tecnologías Utilizadas

Las tecnologías seleccionadas y empleadas en el desarrollo del proyecto son:

- **Backend: API** Web con **ASP.NET Core** 8 (utilizando **C#** 8.0.408 LTS)
- **Base de Datos: SQL Server** 2022, gestionado mediante **Entity Framework Core**
- **Frontend: Next.js** (v15.3.1 con runtime de **Node.js** v20)
- **Estilo de API: RESTful**
- **Despliegue:** Contenerización mediante **Docker**

4. Arquitectura de la Aplicación

La aplicación sigue una arquitectura de 3 niveles con una clara separación de responsabilidades:

1. **Frontend (**Next.js**):** Interfaz de usuario responsable de la interacción con el usuario y la comunicación con el backend a través de la **API RESTful**.
2. **Backend (**ASP.NET Core**):** Expone la **API RESTful** y maneja la lógica de negocio y el acceso a datos. Internamente, se organiza en capas:
   - **Controladores:** Gestionan las peticiones **HTTP** entrantes y orquestan las respuestas
   - **Servicios:** Contienen la lógica de negocio principal y coordinan las operaciones de datos
   - **Acceso a Datos:** Interactúa con la base de datos utilizando **Entity Framework Core** y el **DbContext**
   - **Modelos:** Definen las entidades de datos (`Recipe` , `RecipeIngredient`)
3. **Base de Datos (**SQL Server**):** Almacenamiento persistente de los datos de la aplicación.

Se utilizó **Inyección de Dependencias** en el backend para gestionar las relaciones entre las capas.

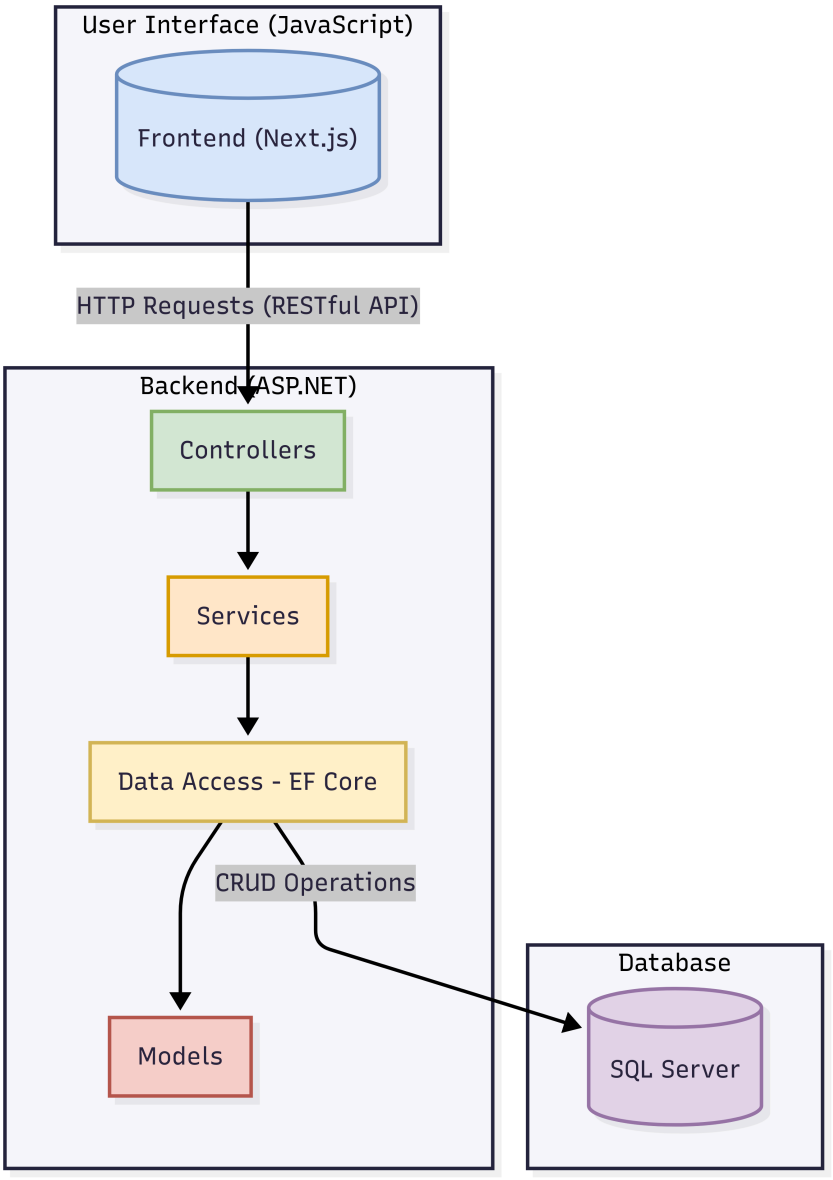

Figura 2: Diagrama de la arquitectura de la aplicación

5. Funcionalidades Implementadas

Basado en la planificación y el seguimiento de tareas, se implementaron las siguientes funcionalidades principales:

5.1. Gestión de Recetas

- Creación de nuevas recetas (nombre, descripción, ingredientes)
- Listado de todas las recetas existentes
- Visualización detallada de una receta específica, incluyendo sus ingredientes
- Edición de la información de una receta existente
- Eliminación de recetas

5.2. Gestión de Ingredientes

- Asociación de ingredientes (nombre, cantidad) a recetas específicas
- Adición de nuevos ingredientes a una receta existente
- Eliminación de ingredientes específicos de una receta

5.3. Búsqueda

- Implementación de funcionalidad de búsqueda para localizar recetas por nombre o descripción tanto en el backend como en el frontend.

5.4. Favoritos

- Posibilidad de marcar/desmarcar recetas como favoritas
- Indicador visual en la interfaz para recetas favoritas
- Ordenamiento del listado de recetas para mostrar las favoritas primero

5.5. API Endpoints

El backend expone los siguientes endpoints **RESTful** bajo la ruta base `/api/Recipes`:

- `GET /api/Recipes/`
  - Recupera una lista de todas las recetas, incluyendo sus ingredientes
- `GET /api/Recipes/{id}`
  - Recupera una receta específica por su ID, incluyendo sus ingredientes
- `POST /api/Recipes/`
  - Crea una nueva receta (requiere nombre y descripción)
- `PUT /api/Recipes/{id}`
  - Actualiza una receta existente por su ID (requiere el objeto completo)
- `DELETE /api/Recipes/{id}`
  - Elimina una receta específica por su ID
- `POST /api/Recipes/{id}/ingredients`
  - Añade un nuevo ingrediente a la receta especificada (requiere detalles del ingrediente)
- `DELETE /api/Recipes/{recipeId}/ingredients/{ingredientId}`
  - Elimina un ingrediente específico de una receta específica
- `PATCH /api/Recipes/{id}/favorite`
  - Cambia el estado de favorito de una receta específica por su ID

6. Despliegue

La aplicación está configurada para ser desplegada utilizando **Docker**. Se proporciona un archivo `docker-compose.yml` que orquesta los contenedores necesarios para el backend (**ASP.NET Core**) y el frontend (**Next.js**), facilitando la configuración y ejecución del entorno.

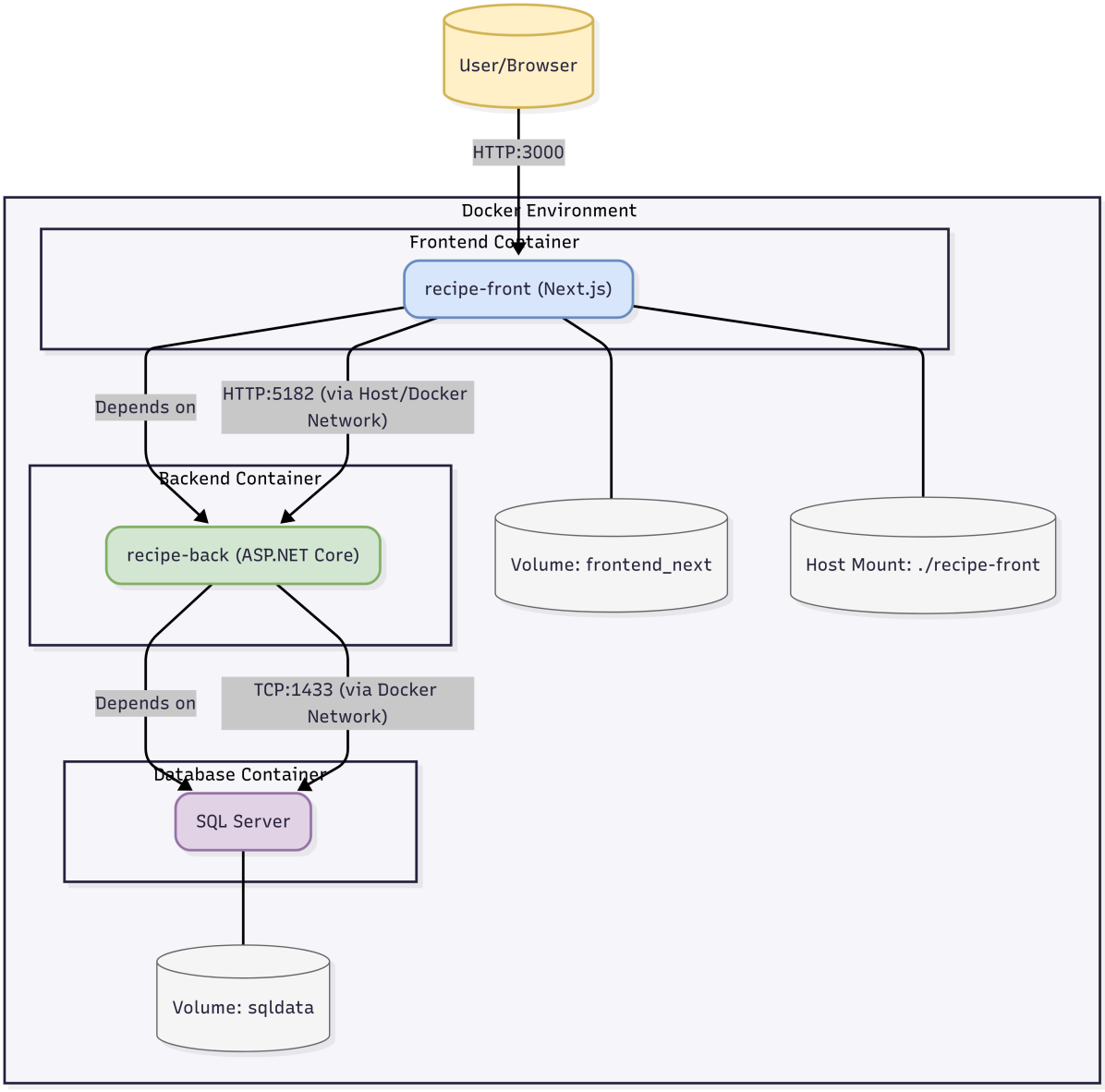

Figura 3: Diagrama de arquitectura de despliegue con Docker

7. Anexos

7.1. Repositorio de Código

El código fuente del proyecto está disponible en el siguiente repositorio de **GitHub**:

- https://github.com/anntnzrb/recipe-app-dotnet°