# **Plantilla para la Documentacion del proceso de Despliegue (Spring Boot + Angular)**

## 🏫 **Datos Generales del Proyecto**

| Campo | Descripción |
| --- | --- |
| **Nombre del Proyecto:** | [Plataforma-ticketing-interno](https://github.com/Johan-Hernandez-80/Plataforma-ticketing-interno) |
| **Integrantes del Equipo:** | Juan Jose Molano Franco  Simon David Cruz Suazo  Johan Steven Hernandez |
| **Fecha de Entrega:** | 18/11/2025 |
| **Docente:** | Rafael Roberto Pérez Grisales |
| **Versión del Proyecto:** | # |
| **Tecnologías Utilizadas:** | Spring Boot, Angular, Docker, MariaDB y MySql |

## ⚙️ **1. Despliegue Local con Docker**

### 🧩 1.1 Requisitos Previos

| Requisito | Verificación (✓ / ✗) |
| --- | --- |
| Docker instalado | ✓ |
| Backend (Spring Boot) empaquetado en .jar | ✓ |
| Frontend (Angular) compilado con ng build | ✓ |
| Archivo Dockerfile creado para cada proyecto | ✓ |
| Archivo docker-compose.yml configurado | ✓ |

### ⚙️ 1.2 Configuración del Backend

**Ruta del proyecto backend para desplegar:**

**Ejemplo** ./Backend

**Codigo del archivo Dockerfile del Backend:**

# --- Etapa de construcción ---

FROM gradle:8.8-jdk21 AS build

WORKDIR /app

# Copiamos TODOS los archivos necesarios para el build

COPY build.gradle settings.gradle ./

COPY gradle ./gradle

COPY src ./src

# Construimos el .jar

RUN gradle clean bootJar --no-daemon

# --- Etapa final ---

FROM eclipse-temurin:21-jre

WORKDIR /app

# Copiamos el jar generado

COPY --from=build /app/build/libs/\*.jar app.jar

# Copiamos el .env (asegúrate de que está junto al Dockerfile)

COPY .env .env

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar"]

**Comando de construcción de imagen:**

docker build -t mi-backend .

**Comando de ejecución del contenedor:**

docker run \

--name backend \

--network backend-net \

-p 8082:8080 \

-e DB\_HOST=mysql \

-e DB\_PORT=3306 \

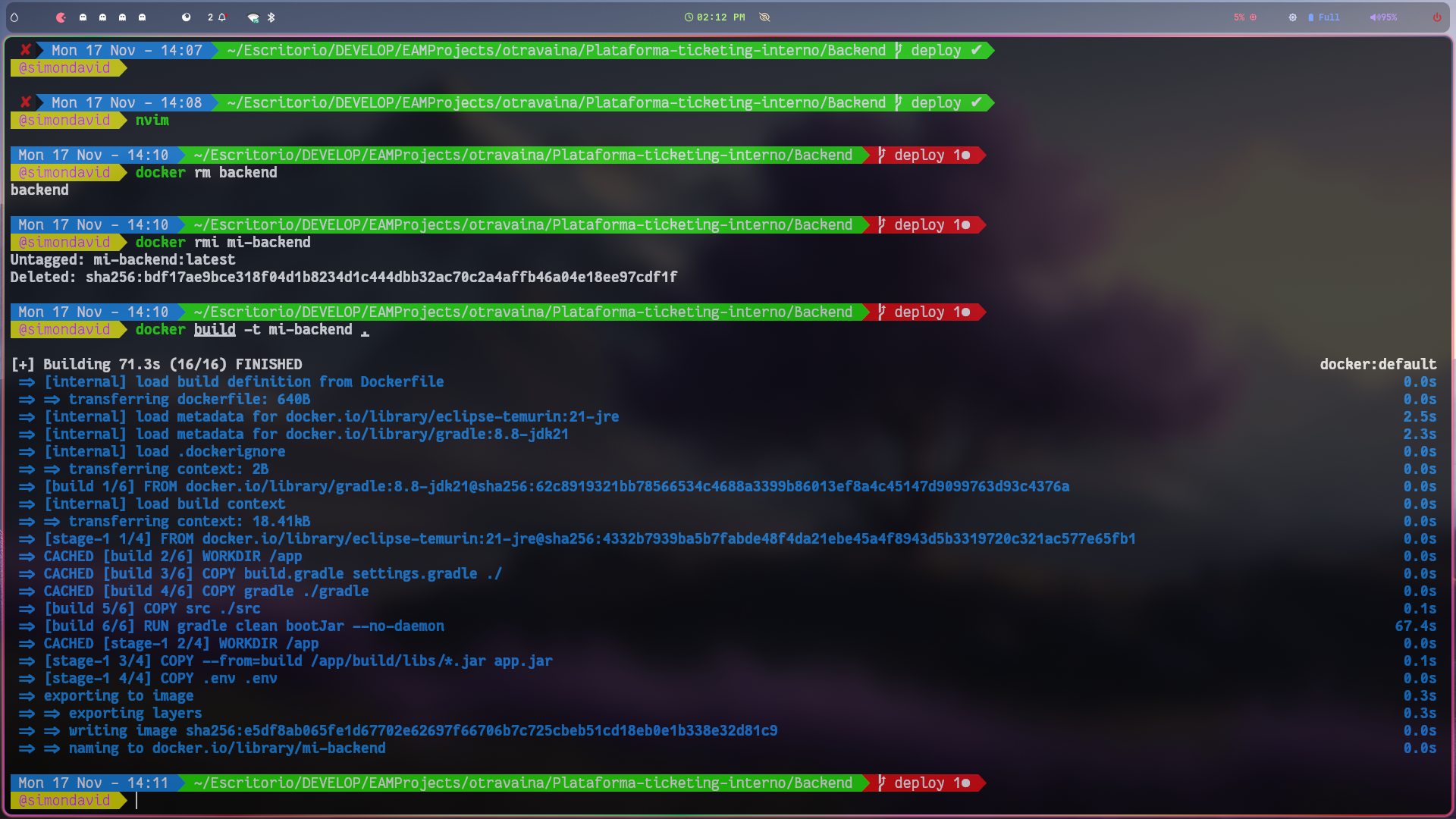
-e DB\_NAME\_DEV=ticketing \

-e DB\_USER\_DEV=root \

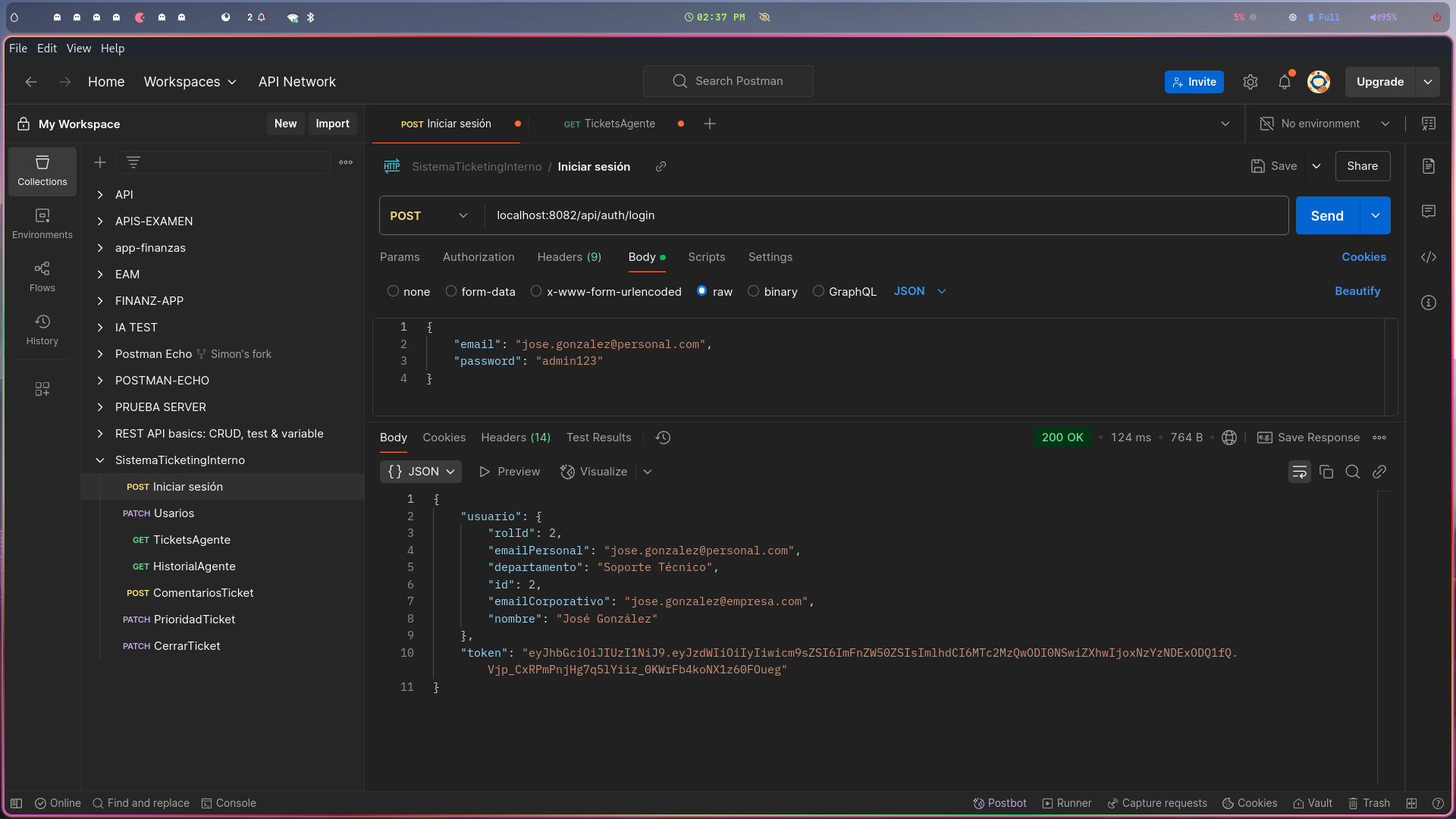
-e DB\_PASSWORD\_DEV=123 \

mi-backend

📸 **Captura 1:** Compilación exitosa del backend



📸 **Captura 2:** Contenedor ejecutándose y prueba de un endpoint funcionando



### 🖥️ 1.3 Configuración del Frontend

**Ruta del proyecto frontend:**

**Ejemplo:** ./Frontend

**Codigo del archivo Dockerfile del Frontend:**

# --- Etapa de build ---

FROM node:20-alpine AS build

# Crear carpeta de trabajo

WORKDIR /app

# Copiar solo package.json y package-lock.json para aprovechar cache

COPY package\*.json ./

# Instalar dependencias (Docker cacheará esta capa si package.json no cambia)

RUN npm ci

# Copiar el resto del código

COPY . .

# Build de producción

RUN npx ng build --configuration production

# --- Etapa de producción con nginx ---

FROM nginx:alpine

# Copiar archivos construidos al directorio de nginx

COPY --from=build /app/dist/frontend /usr/share/nginx/html

# Configurar nginx para Angular Routing

COPY nginx.conf /etc/nginx/conf.d/default.conf

# Exponer puerto 80

EXPOSE 80

# Comando para iniciar nginx

CMD ["nginx", "-g", "daemon off;"]

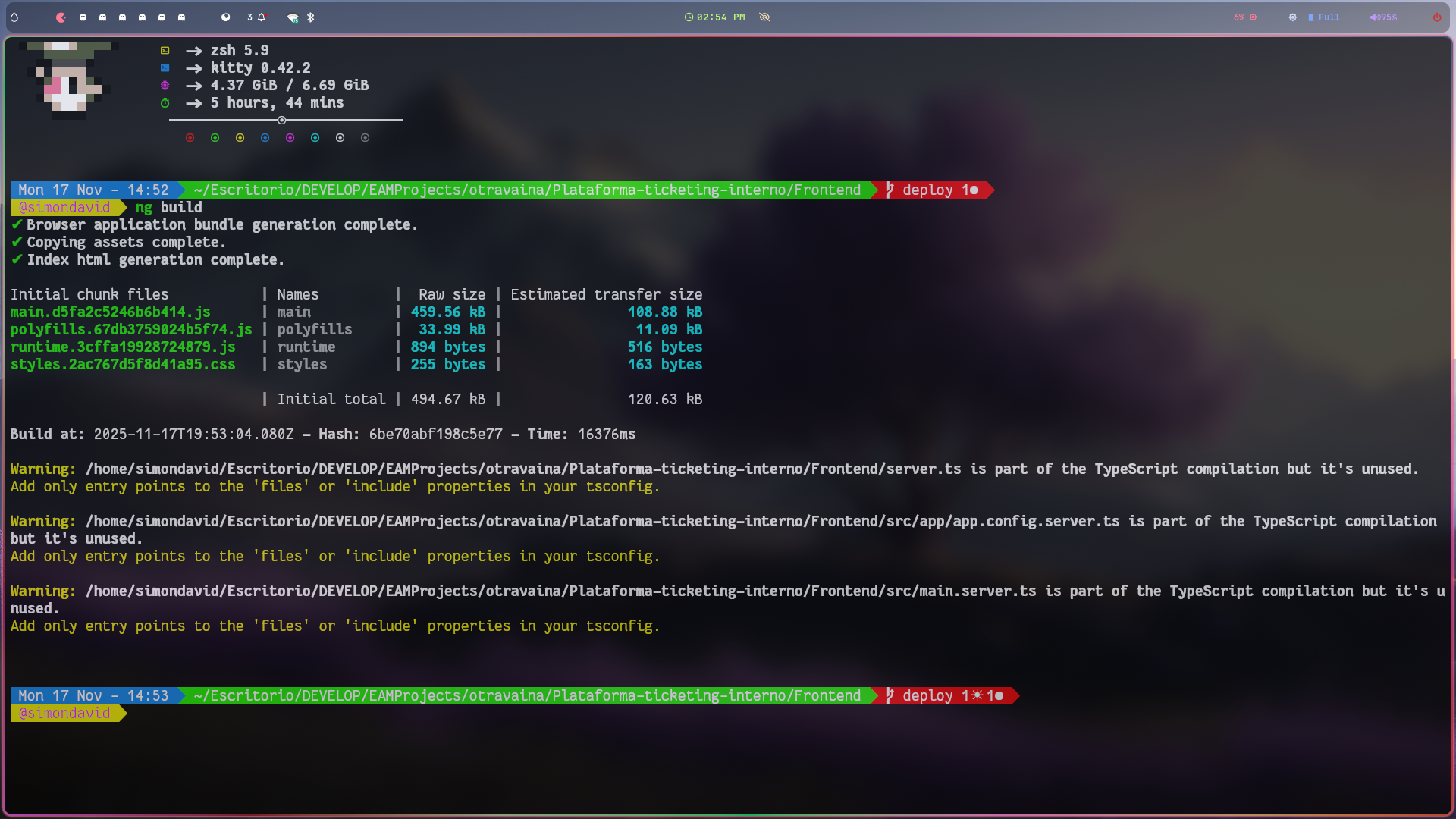
**Comando de construcción de imagen:**

docker build -t frontend-app .

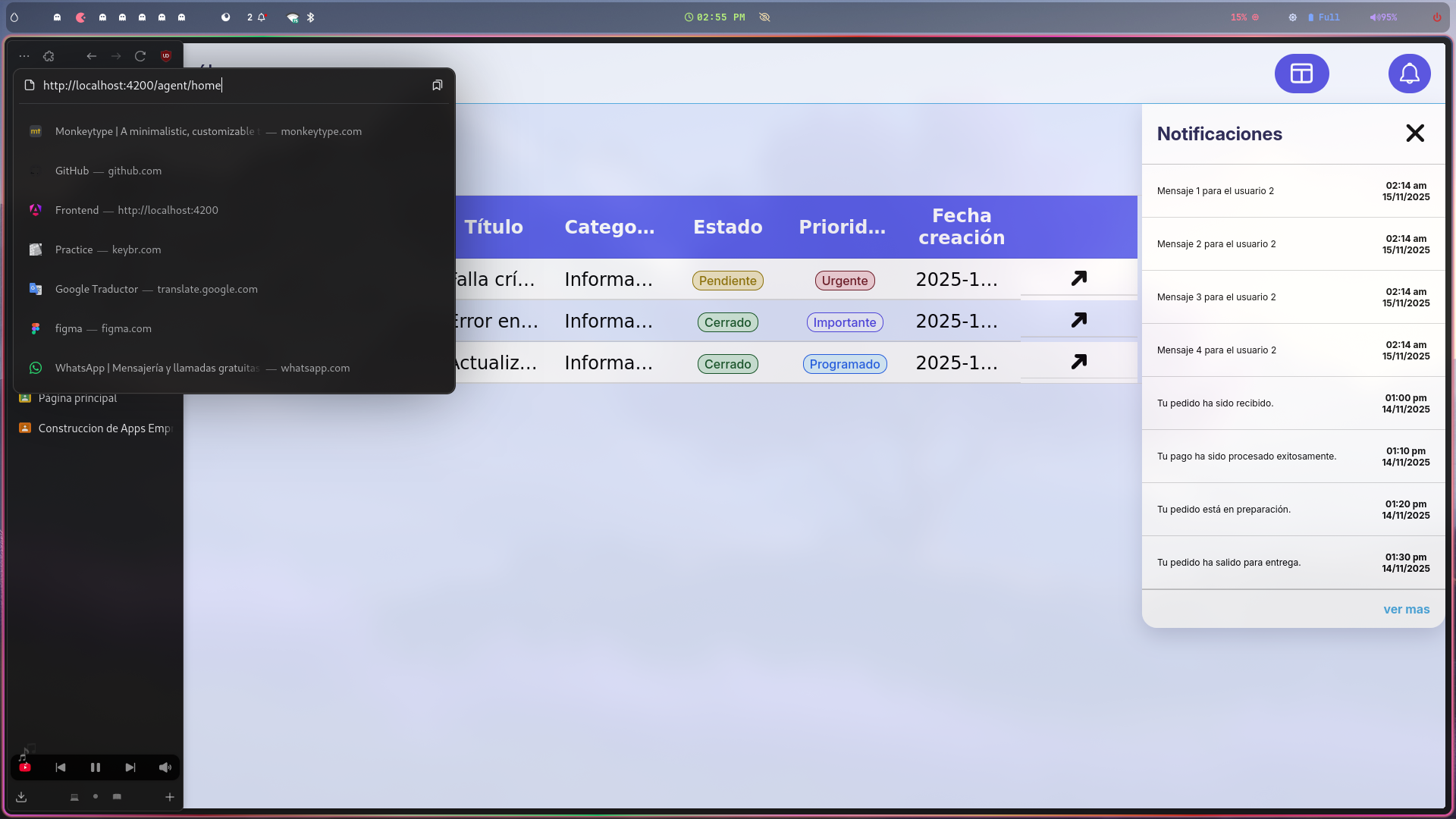
**Comando de ejecución del contenedor:**

docker run -p 4200:80 frontend-app

📸 **Captura 3:** Resultado de ng build



📸 **Captura 4:** Frontend funcionando en navegador local



### 🧩 1.4 Integración Local con Docker Compose

**Codigo del archivo docker-compose.yml:**

version: "3.8"

services:

# --- Backend Spring Boot ---

backend:

image: mi-backend

build:

context: ./Backend

dockerfile: Dockerfile

ports:

- "8082:8082"

depends\_on:

- db

env\_file:

- ./Backend/.env

environment:

SPRING\_PROFILES\_ACTIVE: prod

DB\_HOST: db

DB\_PORT: 3306

DB\_NAME\_PROD: ticketing

DB\_USER\_PROD: user

DB\_PASSWORD\_PROD: pass

JWT\_SECRET: b42e9139f99e8823d89e2178ab63edfe

# --- Base de datos MySQL ---

db:

image: mysql:8.0

environment:

MYSQL\_ROOT\_PASSWORD: rootpass

MYSQL\_DATABASE: ticketing

MYSQL\_USER: user

MYSQL\_PASSWORD: pass

ports:

- "3307:3306"

volumes:

- db\_data:/var/lib/mysql

command: --default-authentication-plugin=mysql\_native\_password

# --- Frontend Angular ---

frontend:

build:

context: ./Frontend

dockerfile: Dockerfile

ports:

- "4200:80"

depends\_on:

- backend

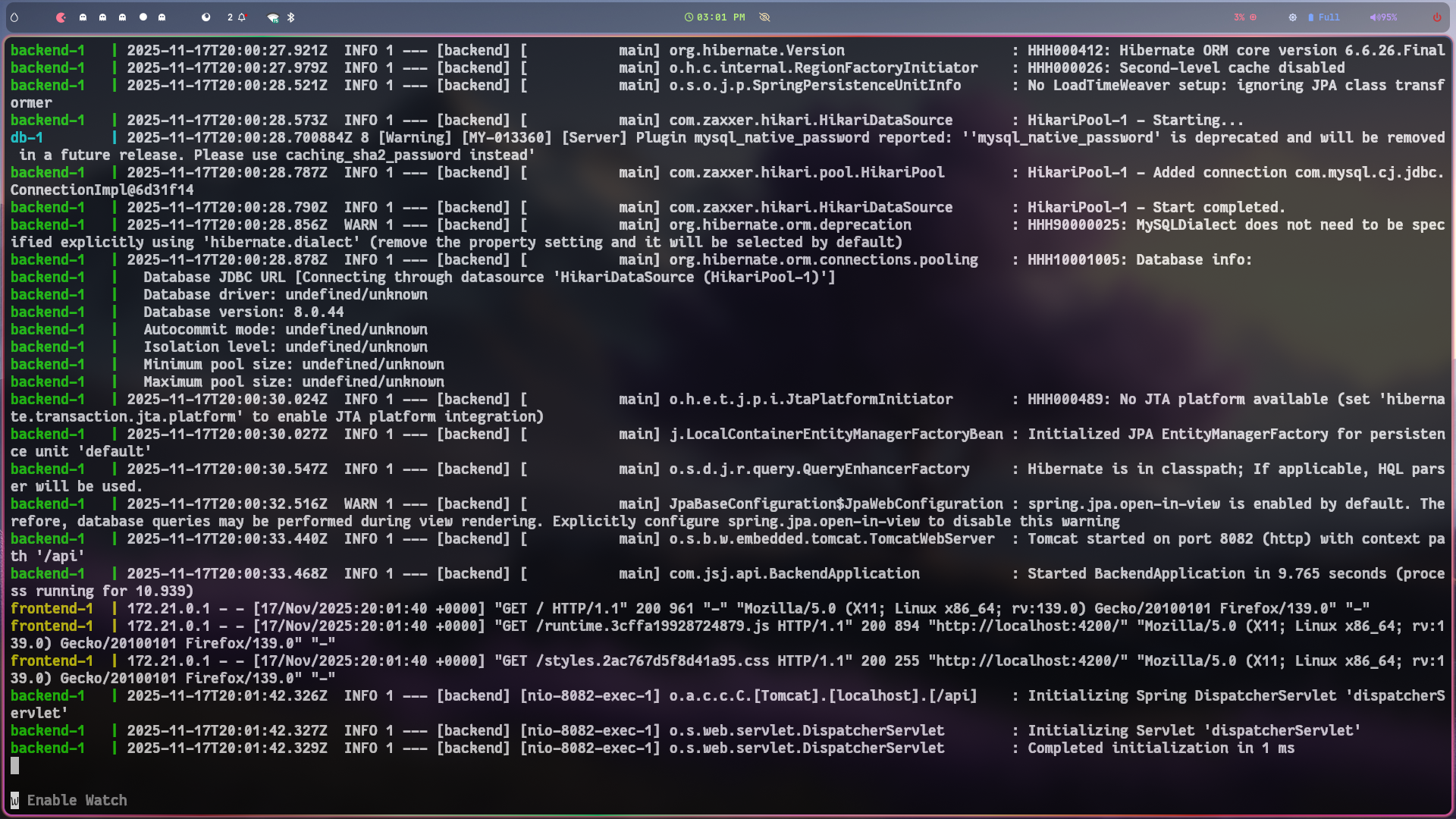
volumes:

db\_data:

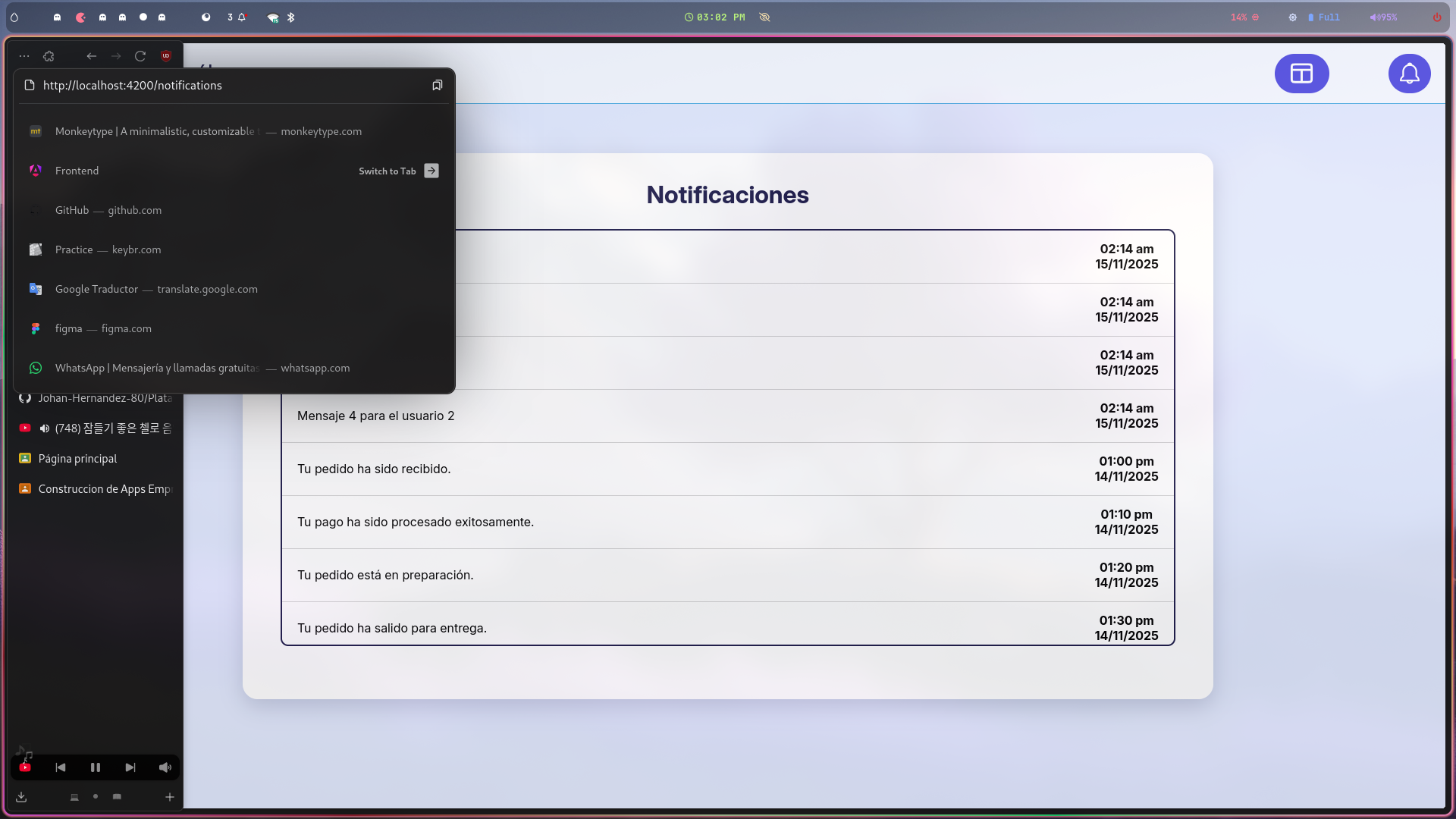
**Comando para levantar ambos servicios:**

docker-compose up --build

📸 **Captura 5:** Ejecución de docker-compose up



📸 **Captura 6:** Frontend consumiendo la API local



## ☁️ **2. Despliegue en la Nube**

### 🌐 2.1 Despliegue del Backend

**Plataforma utilizada:** (Ejemplo: Render / Railway / AWS / otro)

**Repositorio GitHub:**

**Ruta del backend:**

📸 **Captura 7:** Configuración del servicio de base de datos en la nube

**Ejemplo** Configuración del servicio de base de datos en la nube**:**

📸 **Captura 8:** Configuración del servicio del backend en la nube

**Ejemplo** Configuración del servicio del backend en la nube**:**

📸 **Captura 9:** Backend desplegado (prueba de un endpoint desde navegador)

**URL pública del backend:**

**Ejemplo:**

https://nombre-proyecto.onrender.com/api/hello

### 🌍 2.2 Despliegue del Frontend

**Plataforma utilizada:** (Ejemplo: GitHub Pages / Netlify / Vercel / otro)

**Repositorio GitHub:**

**Configuración base-href:**

**Comando de despliegue:**

📸 **Captura 10:** Configuración del despliegue del frontend

📸 **Captura 11:** Sitio web desplegado correctamente

**URL pública del frontend:**

**Ejemplo:**

https://usuario.github.io/nombre-repositorio/

## 🔗 **3. Integración Frontend - Backend (en la nube)**

| Elemento | Descripción |
| --- | --- |
| **Archivo de configuración en Angular** |  |
| **URL del backend configurada en Angular:** |  |
| **Endpoint consumido desde Angular:** |  |

📸 **Captura 12:** Solicitud HTTP desde Angular funcionando correctamente

📸 **Captura 13:** Resultado visible en la interfaz del frontend

## ✅ **4. Validación del Despliegue**

| Criterio | Cumple (✓ / ✗) | Evidencia (captura o URL) |
| --- | --- | --- |
| Backend desplegado localmente con Docker |  |  |
| Frontend desplegado localmente con Docker |  |  |
| Comunicación local entre frontend y backend |  |  |
| Backend desplegado en la nube |  |  |
| Frontend desplegado en la nube |  |  |
| Comunicación en la nube entre frontend y backend |  |  |

## 🧠 **5. Observaciones y Conclusiones**

**Principales dificultades encontradas:**

**Soluciones aplicadas:**

**Recomendaciones para futuros despliegues:**