UNIVERSIDAD CENTRAL DE ECUADOR

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

SISTEMAS DE INFORMACIÓN

ARQUITECTURA DE SOFTWARE

ING. PAULO FERNANDO LLAGUNO CALLE

SISTEMA DE RESERVAS DE LABORATORIOS

INTEGRANTES:

BRYAN FABRICIO CHILENO AGUALONGO
DAVINSON MAYER DIAZ TAPIA
ABEL EDUARDO NAVARRETE GILER

MANUAL DE IMPLEMENTACIÓN

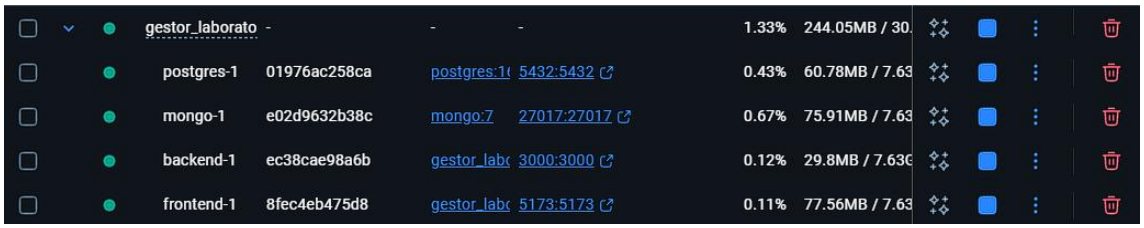
2025-2026

1. Arquitectura y justificación de dos orígenes de datos

El sistema utiliza dos bases de datos para demostrar el uso de múltiples orígenes de datos en un ambiente dockerizado, esto gracias a Docker.

• **PostgreSQL (Relacional):** almacenamiento de usuarios y autenticación. Se justifica por integridad, restricciones (UNIQUE), y consistencia transaccional para credenciales.

• **MongoDB (NoSQL):** almacenamiento de reservas como documentos. Se justifica por flexibilidad del esquema, rapidez para iterar atributos y consultas por usuario.



2. Patrones implementados

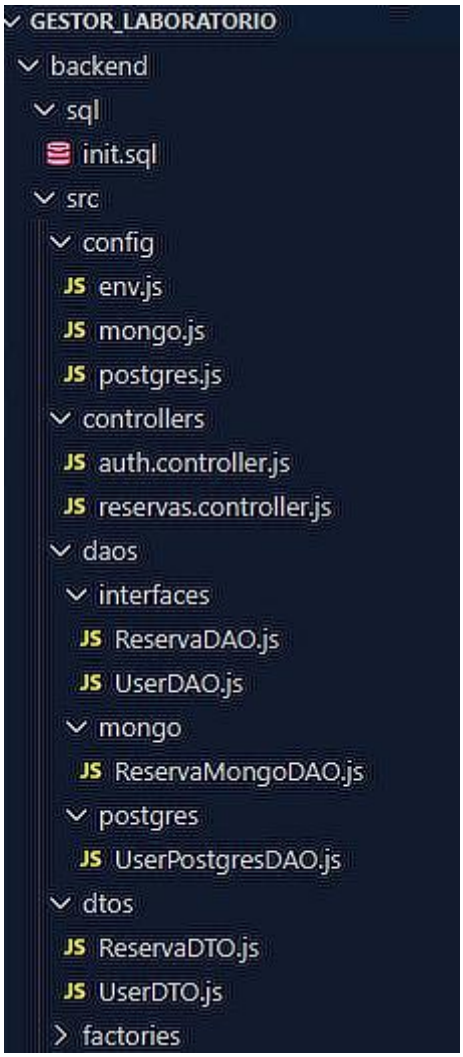
• **DTO: UserDTO y ReservaDTO** Para encapsular datos transferidos entre capas.

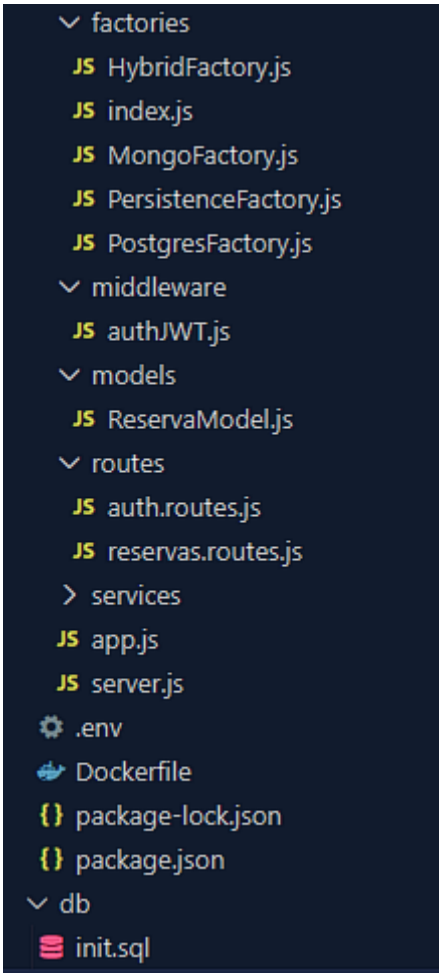
• **DAO: UserPostgresDAO y ReservaMongoDAO** (implementan interfaces).

• **Abstract Factory: HybridFactory** Crea instancias de **DAOs** según la familia (modo híbrido).

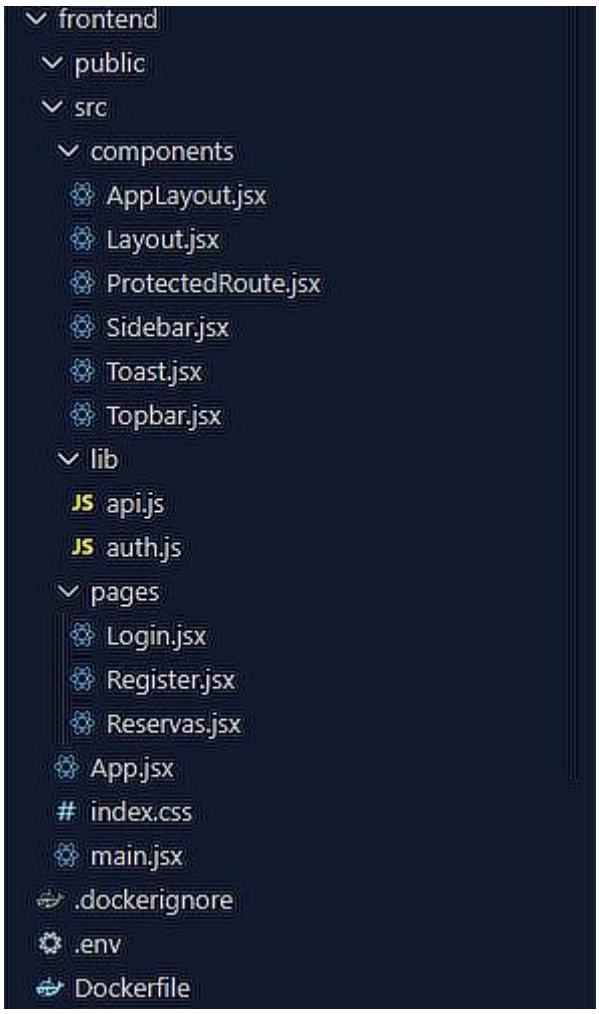
• **Capas: routes → controllers → services → daos → bases de datos.**

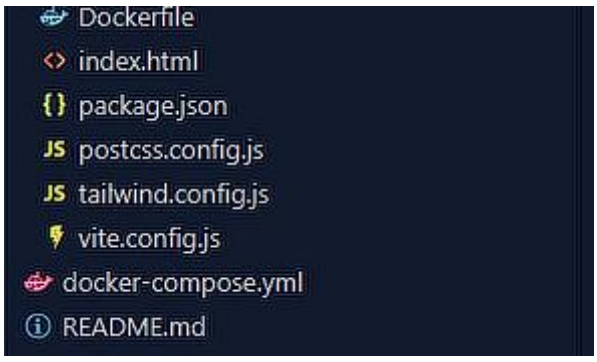
BACKEND


GESTOR_LABORATORIO
backend
sql
init.sql
src
config
env.js
mongo.js
postgres.js
controllers
auth.controller.js
reservas.controller.js
daos
interfaces
ReservaDAO.js
UserDAO.js
mongo
ReservaMongoDAO.js
postgres
UserPostgresDAO.js
dtos
ReservaDTO.js
UserDTO.js
factories


factories
HybridFactory.js
index.js
MongoFactory.js
PersistenceFactory.js
PostgresFactory.js
middleware
authJWT.js
models
ReservaModel.js
routes
auth.routes.js
reservas.routes.js
services
app.js
server.js
.env
Dockerfile
package-lock.json
package.json
db
init.sql

FRONTEND:





3. Endpoints principales (CRUD)

Autenticación (PostgreSQL):

• **POST /auth/register —** Crea usuario

• **POST /auth/login —** Valida credenciales y entrega JWT

Reservas (MongoDB, protegidas por JWT):

• **POST /reservas —** Crea reserva

• **GET /reservas/mine —** Lista reservas del usuario

• **DELETE /reservas/:id —** Elimina una reserva del usuario

4. Reporte combinado de bases de datos

Endpoint protegido por JWT:

• **GET /reportes/usuario-reservas**

Combina: Datos del usuario (PostgreSQL) + listado y resumen de reservas (MongoDB).

5. Pasos para levantar en otro computador

Requisitos: Docker y Docker Compose instalados.

1) Descomprimir el proyecto.

2) Abrir terminal en la carpeta raíz del proyecto.

3) Ejecutar: docker compose up --build

4) Abrir: http://localhost:5173

5) Registrar usuario, iniciar sesión y crear reservas. En la pestaña 'Reporte' se visualiza el reporte combinado.

Crear cuenta

Regístrate para empezar.

Nombre

bryan

Email

bryanfabricio2013@gmail.com

Contraseña

••••••

Crear cuenta

¿Ya tienes cuenta? Inicia sesión

Iniciar sesión

Accede para gestionar tus reservas.

Email

bryanfabricio2013@gmail.com

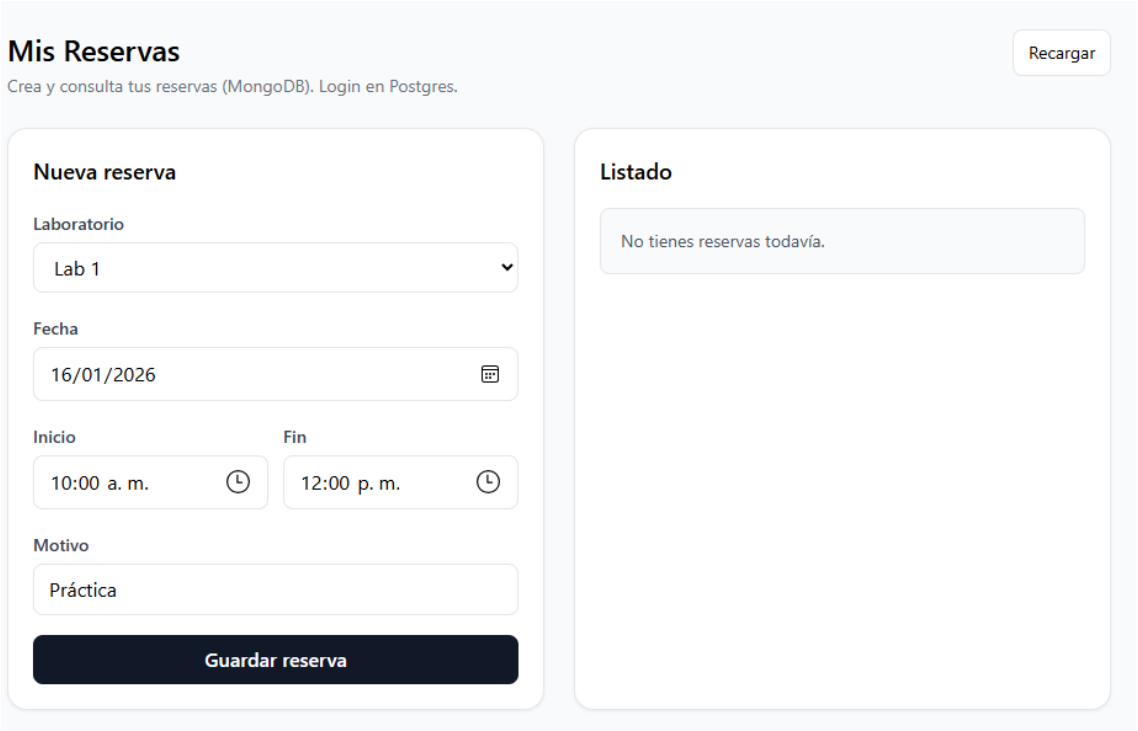
Contraseña

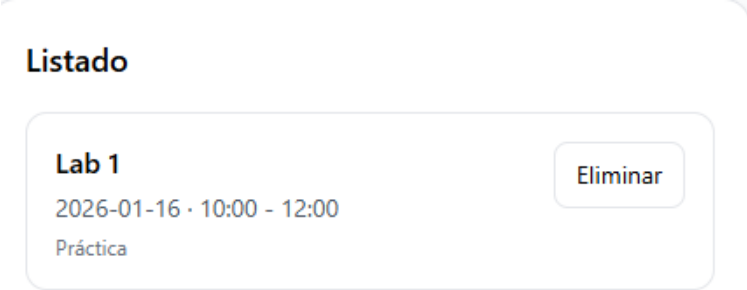
••••••

Entrar

¿No tienes cuenta? Regístrate

Tip
Configura `VITE_API_URL` en `frontend/.env` si tu backend no está en localhost:3000.





6. Archivos de configuración incluidos

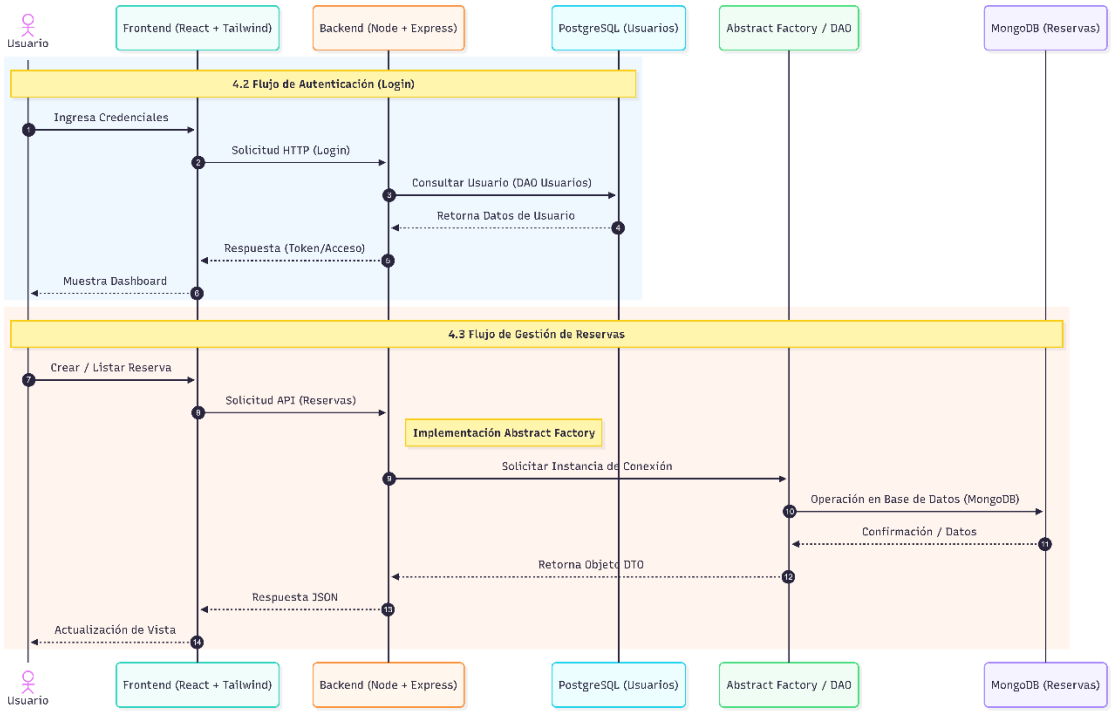
• **docker-compose.yml —** Orquesta contenedores.

• **db/init.sql —** Inicializa tabla users en PostgreSQL.

• **backend/Dockerfile, frontend/Dockerfile —** Builds reproducibles.

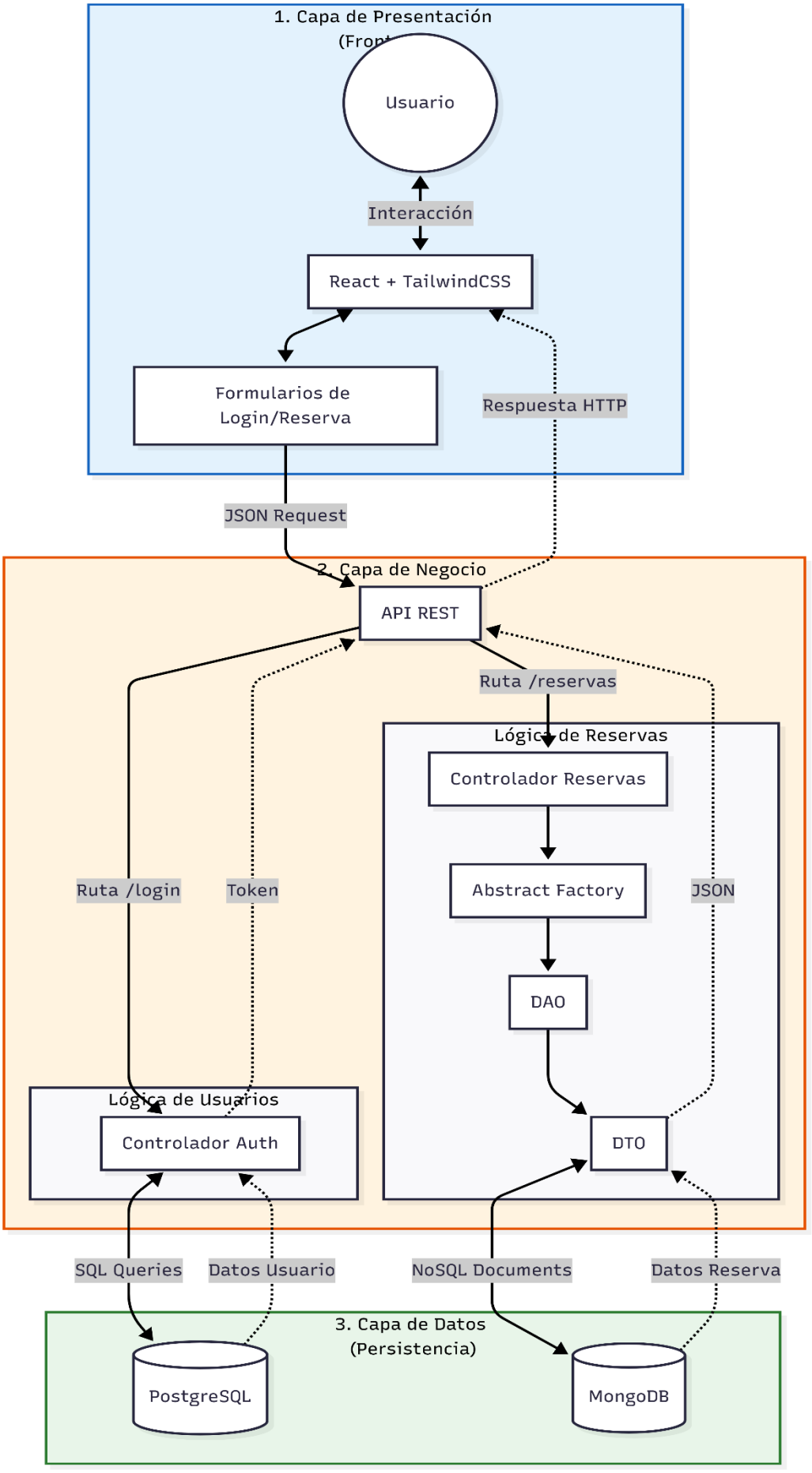
7. Diagrama de Secuencia (Login y Reservas)

Este diagrama combina los flujos descritos en los puntos **4.2** y **4.3** del manual. Muestra cómo el usuario interactúa con el **Frontend**, y cómo el **Backend** separa la lógica de usuarios (PostgreSQL) de la de reservas (MongoDB) usando el patrón **Abstract Factory**



8. Diagrama de Flujo (Casos de Uso y Arquitectura)

Este diagrama representa el flujo lógico general del sistema basándose en los **Casos de Uso** (Registrarse, Iniciar sesión, Crear, Listar, Eliminar) y la **Arquitectura del Sistema**.


1. Capa de Presentación
Usuario
Interacción
React + TailwindCSS
Formularios de Login/Reserva
Respuesta HTTP
JSON Request
2. Capa de Negocio
API REST
Ruta /reservas
Lógica de Reservas
Controlador Reservas
Abstract Factory
DAO
DTO
JSON
Ruta /login
Token
Lógica de Usuarios
Controlador Auth
SQL Queries
Datos Usuario
NoSQL Documents
Datos Reserva
3. Capa de Datos
(Persistencia)
PostgreSQL
MongoDB