

Resumen de Tecnologías

Backend

- **Node.js**: Entorno de ejecución para JavaScript en el servidor.
- **Express**: Framework web para Node.js.
- **TypeScript**: Superconjunto de JavaScript con tipado estático.
- **PostgreSQL**: Base de datos relacional.
- **Jest**: Framework de pruebas para JavaScript.
- **ts-jest**: Preprocesador para Jest con soporte para TypeScript.
- **dotenv**: Carga variables de entorno desde un archivo .env

Frontend

- **React**: Biblioteca de JavaScript para construir interfaces de usuario.
- **TypeScript**: Superconjunto de JavaScript con tipado estático.
- **Jest**: Framework de pruebas para JavaScript.
- **React Testing Library**: Biblioteca para probar componentes de React.


Base de Datos

- **PostgreSQL**: Base de datos relacional de código abierto.


Qué partes han dado más problemas a la hora de implementar

1. Tuve que enfatizarle mucho que me diera los comandos para una consola en Windows, le itere mucho sobre este tema ya que la mayoría errores que tuve fue pq me daba comandos para ejecutar en Linux.

En algunos casos no entendía el error entonces copiaba y pegaba el error en el chat.

 **caguanet** ✕

PS D:\Proyectos\IA4Devs\LTI\backend> mkdir -p src/api/controllers src/api/routes src/domain src/infrastructure src/application
mkdir: A positional parameter cannot be found that accepts argument 'src/api/routes'. PS
D:\Proyectos\IA4Devs\LTI\backend> touch src/api/controllers/UserController.ts src/api/routes/UserRoutes.ts src/domain/User.ts src/infrastructure/Database.ts src/application/UserService.ts src/index.ts touch: The term 'touch' is not recognized as a name of a cmdlet, function, script file, or executable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.

 **GitHub Copilot**

En Windows, los comandos `mkdir -p` y `touch` no funcionan de la misma manera que en Unix. Aquí tienes los comandos equivalentes para PowerShell:

Crear la estructura de carpetas

```
New-Item -ItemType Directory -Path "src/api/controllers"  
New-Item -ItemType Directory -Path "src/api/routes"  
New-Item -ItemType Directory -Path "src/domain"  
New-Item -ItemType Directory -Path "src/infrastructure"  
New-Item -ItemType Directory -Path "src/application"
```

2. Enfatizarle sobre que carpeta ejecutar el comando que me proporcionaba
3. Me solicito ejecutar comandos sobre archivos que no tenia, para solventarlo le solicite que me diera los comandos para crear los archivos que faltaban en el proyecto como fueron los de test.
4. Al compilar la aplicacion genero un error, me encontré que se requería los test dentro de los archivos



caguanet

@workspace como puedo implementar una solución al error que está presentando el archivo

UserService.test.ts:1-7

Workspace

> Referencias de 27 usadas

Para ayudarte a resolver el error en el archivo `UserService.test.ts`, primero asegúrate de que la clase `UserService` y su método `createUser` estén correctamente implementados y exportados.

Aquí tienes un ejemplo de cómo podría verse la implementación de `UserService`:

```
// backend/src/application/UserService.ts
export class UserService {
  createUser(name: string) {
    return { name };
  }
}
```

Aplicar en el editor



caguanet

Este proyecto contiene un BackEnd usando NodeJS, Express y Typescript y un FrontEnd usando ReactJS y Typescript.



GitHub Copilot

Entendido. Vamos a asegurarnos de que tanto el backend como el frontend estén correctamente configurados y que las pruebas unitarias funcionen adecuadamente.

Backend

1. Asegúrate de que `UserService` esté correctamente implementado:

```
// backend/src/application/UserService.ts
export class UserService {
  createUser(name: string) {
    return { name };
  }
}
```

5. Quise arriesgarme ejecutando los test y no me funcionaron, este lo ejecuto sin problemas en el docker



caguanet

PS D:\Proyectos\IA4Devs\LTI\backend> npm test

```
backend@1.0.0 test echo "Error: no test specified" && exit 1
```

"Error: no test specified"



caguanet

no veo este archivo // frontend/jest.config.js /** @type {import('ts-jest').JestConfigWithTsJest} */ module.exports = {
testEnvironment: "jsdom", transform: { "^.+\\.tsx?\$": ["ts-jest", {}], }, };



GitHub Copilot

Vamos a crear el archivo `jest.config.js` tanto para el backend como para el frontend. Aquí tienes los pasos para hacerlo:

Backend

1. Crear el archivo `jest.config.js` en el directorio `backend` :

```
cd backend  
New-Item -ItemType File -Path "jest.config.js"
```

2. Agregar la configuración de Jest al archivo `jest.config.js` :

```
// backend/jest.config.js  
/** @type {import('ts-jest').JestConfigWithTsJest} */  
module.exports = {  
  testEnvironment: "node",  
  transform: {
```

6. Cuando le solicite el docker-compose este ya tenía el contexto sobre el sistema operativo sobre el que estaba trabajando

caguanet

Crea el Docker Compose donde estará un PostgreSQL usa archivo .env para mantener información de usuario y password. Genera un password seguro.



GitHub Copilot

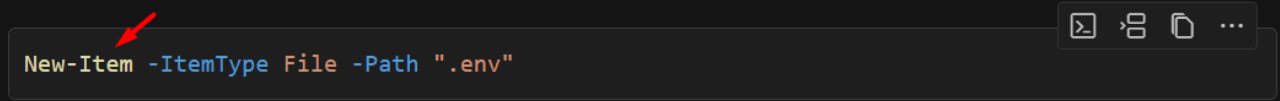
Vamos a crear un archivo `docker-compose.yml` para configurar un contenedor de PostgreSQL y un archivo `.env` para almacenar la información sensible como el usuario y la contraseña.

Paso 1: Crear el archivo `.env`

1. Generar un password seguro:

Puedes usar una herramienta en línea o un generador de contraseñas para crear una contraseña segura. Aquí tienes un ejemplo de contraseña segura: `S3cur3P@ssw0rd!`.

2. Crear el archivo `.env` en el directorio raíz del proyecto:



```
New-Item -ItemType File -Path ".env"
```

3. Agregar la información de usuario y contraseña al archivo `.env`:

```
POSTGRES_USER=yourusername  
POSTGRES_PASSWORD=S3cur3P@ssw0rd!  
POSTGRES_DB=yourdatabase
```

7. La conexión a la base de datos para la creación de la tabla se me dificultó mucho a través de comandos, tuve que iterar mucho los errores que me salían pero al final logré crear la tabla.
8. 🚀 Fue interesante cuando quise dockerizar toda la solución, porque me levantó el back y la base de datos pero el front no debido a que el puerto estaba ocupado, sin embargo detecté que no tenía el pool de conexión a la base de datos y lo agregué en paso a paso de instrucciones.



caguanet

@workspace ahora bien necesito dockerizar el backend como el frontend utilizando el docker compose pero manteniendo la base de datos postgres en este estable la conexiones en el docker que se requieran

Workspace

> Referencias de 33 usadas

Para dockerizar tanto el backend como el frontend y mantener la base de datos PostgreSQL, puedes actualizar tu archivo `docker-compose.yml` y crear los archivos `Dockerfile` para el backend y el frontend. Aquí tienes los pasos detallados:

Paso 1: Crear `Dockerfile` para el Backend

Crea un archivo `Dockerfile` en el directorio `backend` con el siguiente contenido:

```
# backend/Dockerfile
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./
RUN npm install
```

Paso 4: Actualizar el Backend para Conectar con PostgreSQL

Asegúrate de que el backend esté configurado para conectarse a PostgreSQL utilizando las variables de entorno. Ya tienes esto configurado en `Database.ts`:

```
import { Pool } from 'pg';
import dotenv from 'dotenv';

dotenv.config();

const pool = new Pool({
  user: process.env.POSTGRES_USER,
  host: process.env.POSTGRES_HOST,
  database: process.env.POSTGRES_DB,
  password: process.env.POSTGRES_PASSWORD,
  port: Number(process.env.POSTGRES_PORT),
});

export default pool;
```

Evidencia


Contenedores corriendo:













Containers [Give feedback](#)

Container CPU usage ⓘ **0.02% / 200%** (2 CPUs available)

Container memory usage ⓘ **235.48MB / 1.88GB**

Show charts

Search  ☐ Only show running containers

<input type="checkbox"/>	Name ↑	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	Iti	-	-	-	0.02%	52 seconds ago	  
<input type="checkbox"/>	backend	7b06bd2dd2b2	Iti-backend	3000:3000 ↗	0%	53 seconds ago	  
<input type="checkbox"/>	frontend	e06414ae94e7	Iti-frontend	8080:80 ↗	0%	52 seconds ago	  
<input type="checkbox"/>	postgres	de27250fb961	postgres:latest	5432:5432 ↗	0.02%	54 seconds ago	  


Backend:

localhost:3000/api/users

Obtener todos los usuarios

Front:

localhost:8080



Edit src/App.tsx and save to reload.

[Learn React](#)

Conclusiones

- Logre levantar el proyecto con muy pocas intervenciones a nivel de código

- Se requiere algo de conocimiento para poder implementar otras estrategias mas especificas de lo que necesitamos que ejecute.
- A la primera no sacas un resultado satisfactorio requieres muchas iteraciones, paciencia y habilidades técnicas.
- 🙌 Me siento satisfecho y contento con el resultado.

Conclusiones

[caguanet/AI4Devs-LTI: Proyecto creado con instrucciones proporcionadas por IA](#)