

## 1. Estudio del problema y análisis del sistema

El proyecto que tengo que realizar es el desarrollo de un backend en Node.js para mi aplicación angular llamada Pokedock, esta deberá permitir almacenar usuarios, noticias y enviar Noticias a todos los usuarios del sistema registrados previamente.

Lo desarrollo utilizando Node.js con Express en el backend, además de implementar una base de datos con PostgreSQL, JWT lo voy a usar para el tema de autenticación de los usuarios, y por último Nodemailer para enviar correos electrónicos de Ethereum como servicio de prueba.

### 1.2. Funciones y resultados esperados

El sistema necesita llevar a cabo las siguientes funciones:

- Autenticación y manejo de usuarios: Podemos guardar en la base de datos los usuarios registrados y los usuarios pueden iniciar sesión.
- Manejo de noticias: Podemos añadir noticias en nuestra base de datos y en nuestro fronted hace una petición a la base de datos de manera que va cargando automáticamente las noticias cuando añadimos noticias ya que recoge las noticias de nuestra base de datos desplegada.
- Notificación a través de correo electrónico: Enviar correos a los usuarios registrados al publicar una noticia nueva.

### 1.3. Metas

- Crear una API REST segura y escalable que gestione usuarios y noticias.
- Establecer un sistema de seguridad con la validación de Token para el inicio de sesión de nuestros usuarios y así no tener problemas a la hora de hacer peticiones desde ese usuario logado.
- Enviar a los usuarios el correo electrónico cada vez que haya nuevas noticias

### 1.4. Modelado de la solución

#### 1.4.1. Recursos humanos

- Desarrollador Backend: Responsable del diseño, implementación y mantenimiento del sistema.
- Administrador de Base de Datos: Para la gestión y optimización de PostgreSQL.
- DevOps: Para la configuración y despliegue en entornos de producción (Render, el cual es mi caso).

#### 1.4.2. Recursos hardware

- Servidor en la nube (Render): Para el despliegue del backend y la base de datos PostgreSQL.
- Base de datos PostgreSQL en la nube: Para el almacenamiento de usuarios y noticias.

#### 1.4.3. Recursos software

- Node.js y Express: Framework para construir la API REST.
- PostgreSQL: Base de datos para gestionar usuarios y noticias.
- JWT (jsonwebtoken): Autenticación y manejo de sesiones.
- Nodemailer: Para el envío de correos electrónicos.
- Ethereum: Servicio de prueba para correos.
- dotenv: Para la gestión de variables de entorno.
- Render: Para el despliegue del backend en producción.

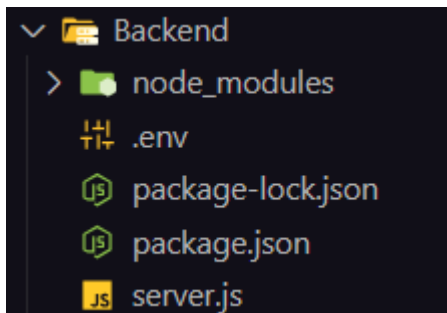
## 2. Ejecución de la práctica

### 2.1. Documentación técnica e implementación de la aplicación

- El backend está desarrollado con Node.js y Express.
- La base de datos es PostgreSQL a la que nos conectamos a través del paquete pg.

```
"pg": "^8.13.3"
```

### Estructura del proyecto



## Dependencias principales

```
"dependencies": {  
  "bcrypt": "^5.1.1",  
  "bcryptjs": "^2.4.3",  
  "body-parser": "^1.20.3",  
  "cors": "^2.8.5",  
  "dotenv": "^10.0.0",  
  "express": "^4.21.2",  
  "jsonwebtoken": "^9.0.2",  
  "nodemailer": "^6.10.0",  
  "pg": "^8.13.3"
```

- Registro de usuario: Para poder realizar un registro en nuestra aplicación pues tenemos que rellenar los datos del formulario y una vez se envíen los datos, llega a nuestro endpoint donde si son correctas las validaciones pues se añadirán a nuestra base de datos.
- Inicio de sesión: Cuando un usuario inicie sesión en la aplicación, si está todo correcto pues genera un token de validación y este hará como con el registro nos llegará la petición post realizada desde el front hacia nuestro endpoint para comprobar que los datos son correctos y así generar correctamente el token de validación.
- Publicación de noticias: Podremos publicar noticias a través de nuestro endpoint para obtener las noticias que podremos agregar enviando peticiones a nuestro servidor desde postman.
- Notificación por correo: Una vez se introduzca una noticia automáticamente se añadirá en la parte front de nuestra aplicación de modo que cuando se añada esta noticia, automáticamente se enviará un correo a todos los usuarios registrados de nuestra aplicación, enviará el correo gracias a Ethereal, que sirve como correo externo para así poder enviar correos.

## Seguridad

- JWT: Se utiliza como sistema de validación cuando un usuario quiera iniciar sesión en la aplicación, se utiliza por seguridad.
- CORS: Configurado para permitir solicitudes desde el frontend.
- dotenv: Para gestionar información sensible como contraseñas y tokens.

### 3. Documentación del sistema

#### 3.1. Manual de instalación y configuración de la aplicación

##### Requisitos

- Render
- Node
- Postgre

##### Instalación

1. Clonar de mi repositorio el proyecto

```
git clone https://github.com/alejandro-source/Backend/tree/email
```

2. Instalar dependencias

- express → Para crear el servidor backend
- bodyParser → Para poder analizar las solicitudes JSON
- cors → Permite realizar solicitudes desde otros dominios
- dotenv → Manejar variables de entorno, en este caso Render usa don'tenv
- jsonwebtoken → Token de validación para cuando incie sesión el usuario
- nodemailer → Para poder enviar la notificación a todos los usuarios registrados
- pg → Para poder crear nuestra base de datos en Render

3. Configurar variables de entornos

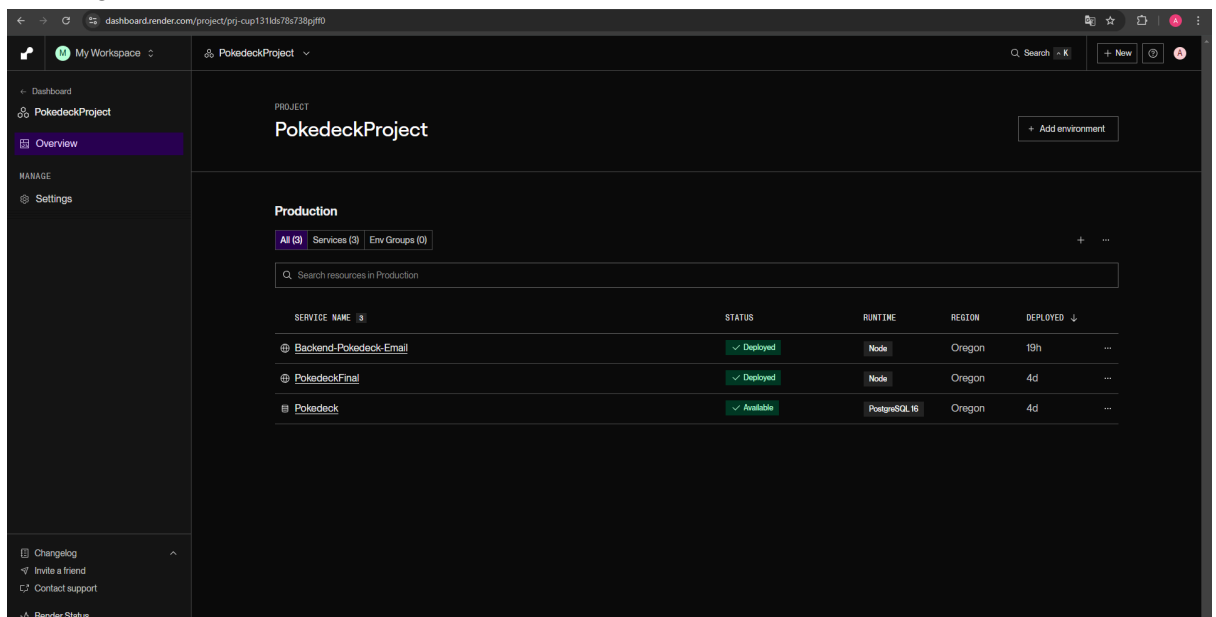
```
.env x
Backend > .env
1 # Variables de configuración para la base de datos PostgreSQL
2 DB_HOST=dpg-cup13khopnds73909190-a
3 DB_USER=pokedeck_user
4 DB_PASSWORD=wNnjExI0ddC4VJBk1HqxJzTrrrpc2Dbb
5 DB_NAME=pokedeck_e7zz
6 DB_PORT=5432 # Este es el puerto predeterminado para PostgreSQL
7
8 # Puerto del servidor de Node.js (se usará en local)
9 PORT=3000
10
11
```

#### 4. Crear las tablas de la base de datos

```
CREATE TABLE usuarios (  
  id SERIAL PRIMARY KEY,  
  nombre VARCHAR(100) NOT NULL,  
  email VARCHAR(100) UNIQUE NOT NULL,  
  password VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE noticias (  
  id SERIAL PRIMARY KEY,  
  titulo VARCHAR(200) NOT NULL,  
  contenido TEXT NOT NULL,  
  fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  imagen TEXT  
);
```

#### 5. Desplegar el servidor

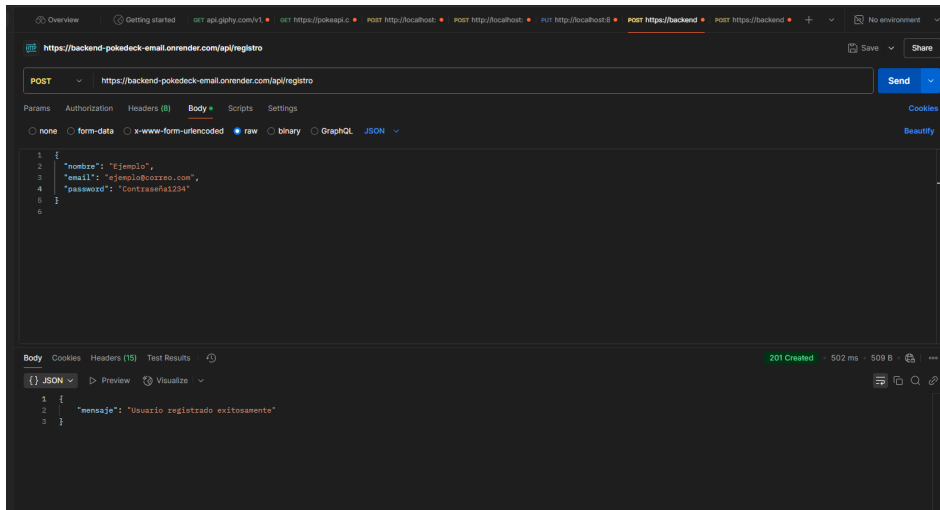


The screenshot shows the Render dashboard for a project named 'PokedeckProject'. The left sidebar contains navigation links: 'Dashboard', 'PokedeckProject', 'Overview' (highlighted), 'MANAGE', 'Settings', 'Changelog', 'Invite a friend', 'Contact support', and 'Render Status'. The main content area displays the 'Production' environment. At the top, there are tabs for 'All (3)', 'Services (3)', and 'Env Groups (0)'. Below these is a search bar labeled 'Search resources in Production'. A table lists the deployed services:

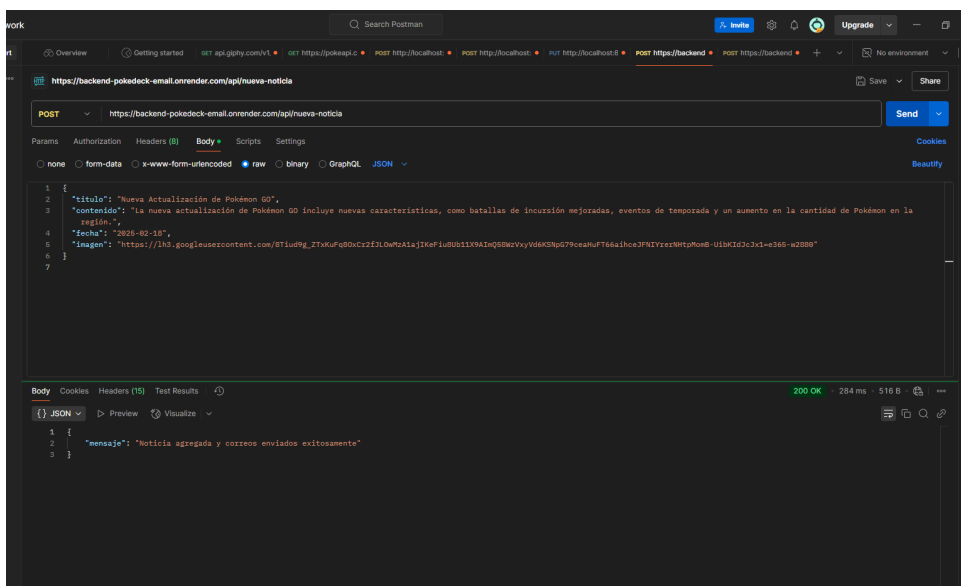
SERVICE NAME	STATUS	RUNTIME	REGION	DEPLOYED
Backend-Pokedeck-Email	✓ Deployed	Node	Oregon	19h
Pokedeck-Front	✓ Deployed	Node	Oregon	4d
Pokedeck	✓ Available	PostgreSQL 15	Oregon	4d

## 3.2. Manual de usuario

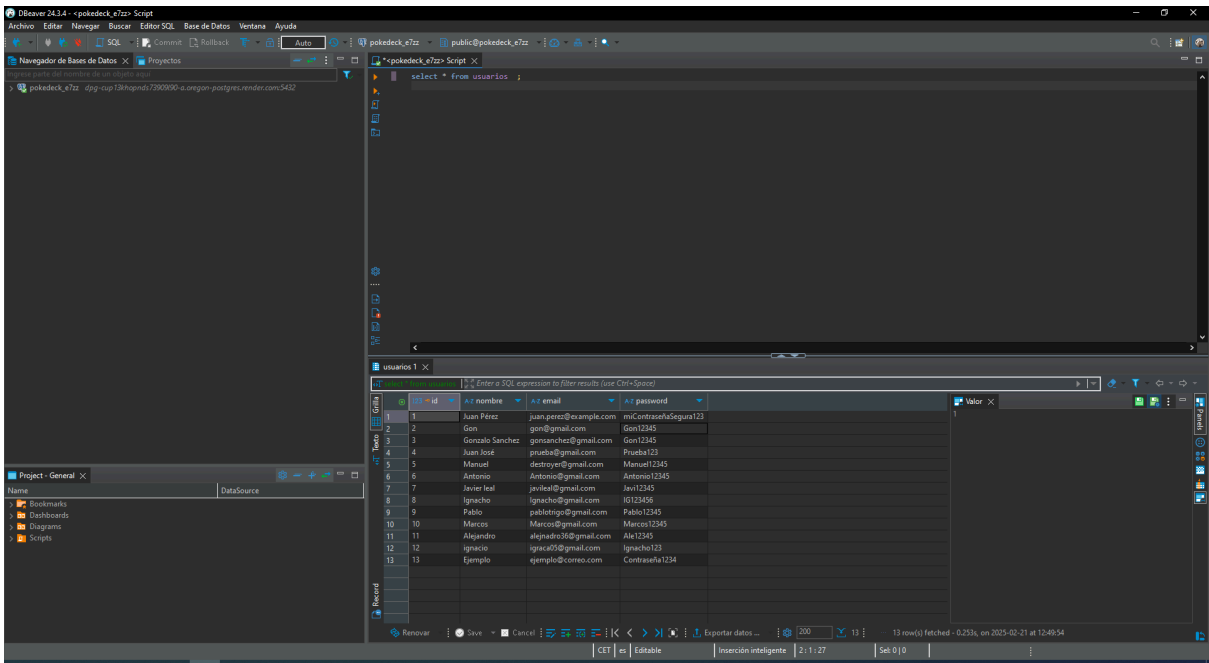
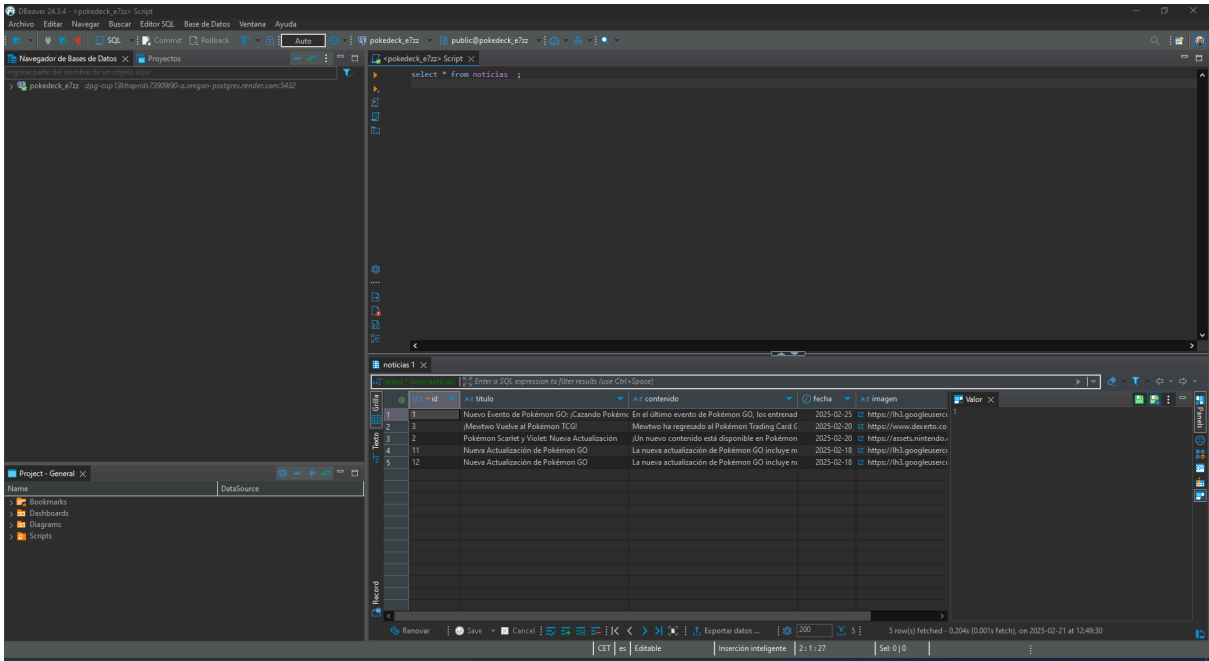
### Creación de un usuario desde Postman



### Creación de una noticia desde Postman



Con la aplicación dbeaver podemos manejar nuestra base de datos desplegada en render



## 4. Conclusiones finales

### 4.1. Grado de cumplimiento de los objetivos fijados

El proyecto cumple satisfactoriamente con los objetivos propuestos, proporcionando una seguridad a la página web cuando un usuario se registre, inicie sesión y también a la hora de enviar correos cuando haya una noticia nueva

### 4.2. Propuesta de modificaciones o ampliaciones futuras del sistema implementado

- Encriptar las contraseñas con bcrypt para así tener más seguridad
- Añadir nuevas funcionalidades CRUD

## 5. Bibliografía

- <https://stackoverflow.com/questions/60248452/is-there-a-compatibility-list-for-angular-angular-cli-and-node-js>
- [https://www.reddit.com/r/node/comments/13uoxxr/is\\_nodejs\\_even\\_considered\\_for\\_serious\\_backend/?rdt=41540](https://www.reddit.com/r/node/comments/13uoxxr/is_nodejs_even_considered_for_serious_backend/?rdt=41540)
- <https://www.youtube.com/watch?v=p3Eq84HmBPg&t=17927s>
- <https://faztweb.com/contenido/nodejs-http-frameworks>