

# SISTEMA DE VENTA DE GORRAS

## Grupo N:

- Liang Anderson Ligua Chavarria
- Castillo Merejildo Joshúa Javier
- Tomalá Tomalá Ofelia Jessica
- Alexander Rodolfo Chuquipoma Vallejos
- Marlon Jorge Espinoza Fares

## Módulo IV – DevOps

### Nombre de la tienda: Kawsay Caps S.A.

Esta aplicación es una tienda online de gorras que permite a los usuarios explorar el catálogo, filtrar productos, añadirlos al carrito y completar la compra mediante distintos métodos de pago. Para los administradores y bodegueros, facilita la gestión de productos y clientes de manera eficiente.

### Tecnologías Usadas:

- **Frontend:** React, Bootstrap 5, CSS3, JavaScript, React Router, Axios, Chart.js
- **Backend:** Node.js, Express, Mongoose
- **Base de Datos:** MongoDB Atlas
- **Plataforma de despliegue:** Render
- **Control de versiones:** GitHub

### Instrucciones para ejecutar localmente

#### 1. Clonar el repositorio

En tu símbolo del sistema o terminal de visual studio code usar los siguientes comandos:

- `git clone https://github.com/CastilloJoshuaEE/proyecto-Grupo-N.git`
- `cd proyecto-Grupo-N`

Ya estando en la carpeta “proyecto-Grupo-N” usa los siguientes:

- `git remote remove origin`

#### 2. Configurar el Backend

- Accede a la carpeta “proyecto-Grupo-N” del servidor:  
`cd server`
- Instala las dependencias necesarias:

- npm init

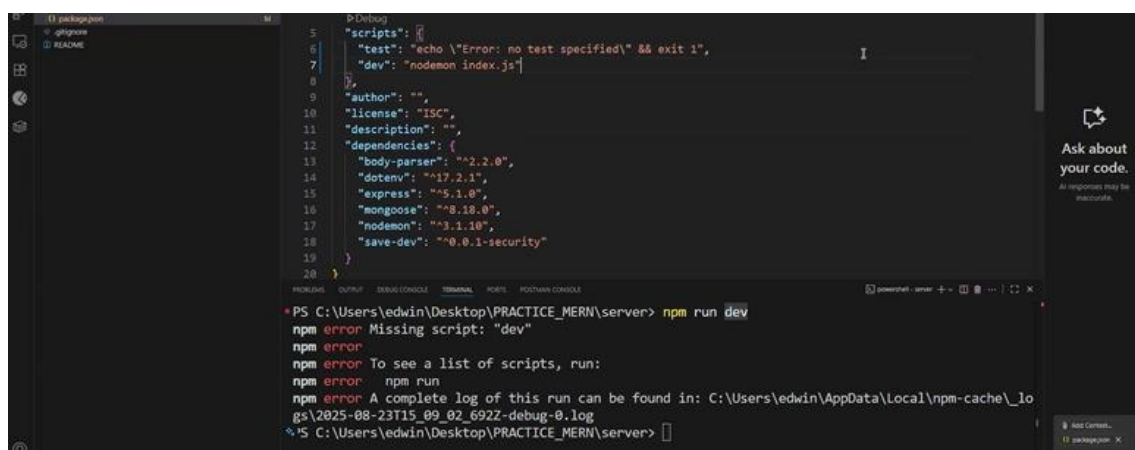
Nota: Test comand, git repository y keywords opcional

- npm i dotenv
- npm i body-parser
- npm i express
- npm i mongoose
- npm install --save-dev nodemon
- npm install dotenv jsonwebtoken bcryptjs
- npm i cors
- npm run dev
- npm install multer

- Inicia el servidor de desarrollo:  
npm run dev

Nota: en el caso de error agregar en el package.json de server agregar “dev”: “nodemon index.js”

Así como se muestra en la imagen:



### 3. Configurar el Frontend

- Usa el siguiente comando en la raíz del “proyecto-Grupo-N” : npx create-react-app client

Nota: En el caso de que se haya creado manualmente la carpeta “client” ser como: npx create-react-app

- Accede a la carpeta “proyecto-Grupo-N” del cliente para instalar las dependencias:  
cd client
- Instala las dependencias necesarias:
  - npm i bootstrap
  - npm i react-router-dom
  - npm i react-hot-toast

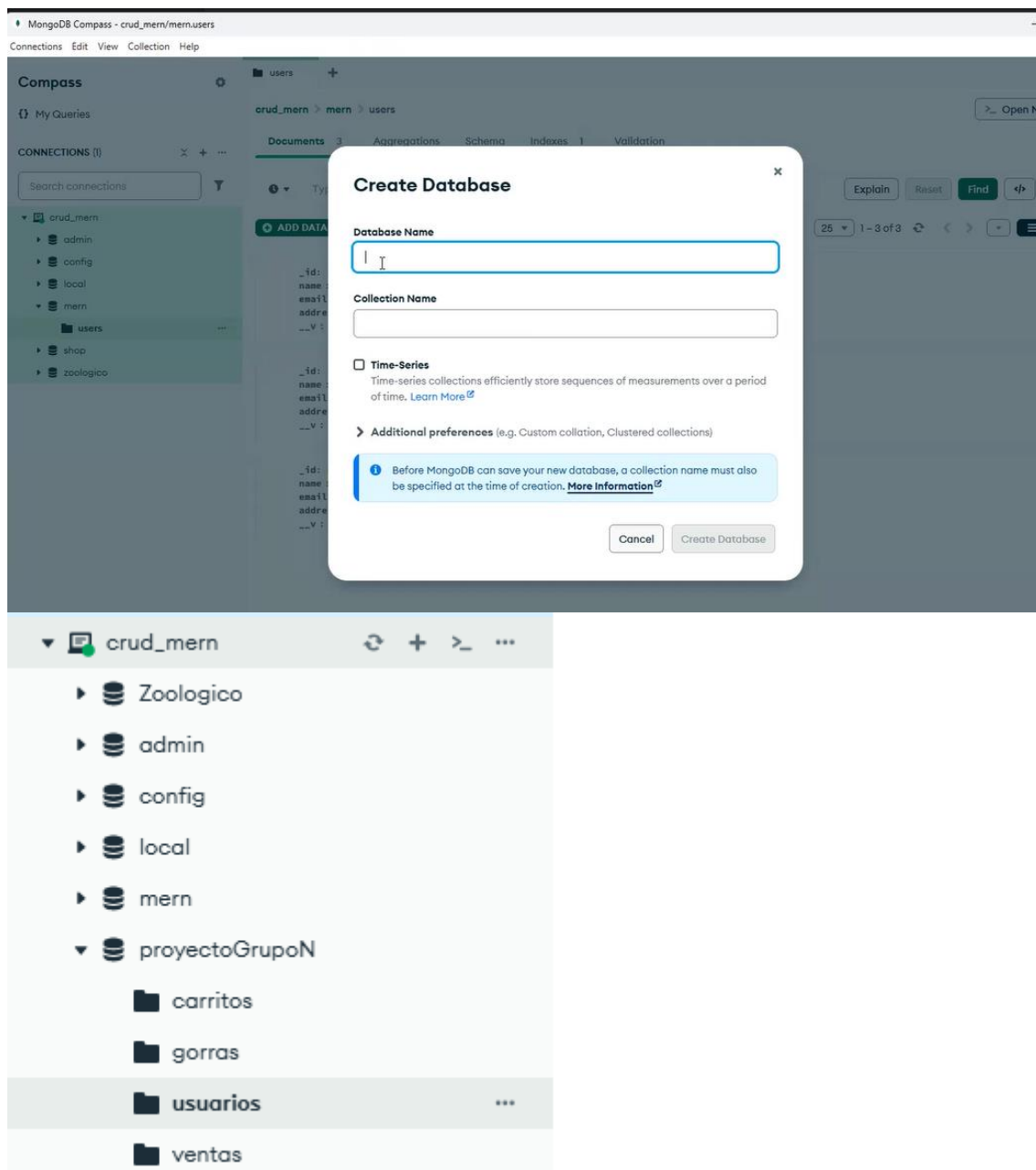
- npm install react-bootstrap bootstrap
- npm i font-awesome
- npm i axios
- npm install chart.js react-chartjs-2
- npm install react-icons
- Generar versión optimizada para producción:  
npm run build
- Inicia el servidor de desarrollo:  
npm run start

Para ejecutar en desarrollo local el proyecto, tener ambas terminales activas:

- Para frontend: D:\Respaldo 2025\Documents\Bootcamp de la fabrica\modulo3\PRACTICE\_MERN\client>npm start
- Para backend: D:\Respaldo 2025\Documents\Bootcamp de la fabrica\modulo3\PRACTICE\_MERN\server> npm run dev

#### **4. Configurar la Base de Datos (MongoDB)**

- Abre MongoDB Compass.
- Crea una nueva conexión con los siguientes datos:
  - URL: mongodb://localhost:27017
  - Nombre de conexión: crud\_mern
  - Nombre de base de datos: proyectoGrupoN
  - Collection Name: usuarios, carritos, gorras, ventas



Si utilizas MongoDB Atlas, reemplaza la MONGO\_URI en el archivo .env con tu cadena de conexión del cluster en la nube.

### Opcional:

#### Generación de JWT Secret

- **Crear Archivo generateSecret.js**

Crea el archivo generateSecret.js en la raíz de la carpeta server:

```
// generateSecret.js
```

```
const crypto = require('crypto');
```

```
const secret = crypto.randomBytes(64).toString('hex');
console.log('JWT_SECRET=', secret);
```

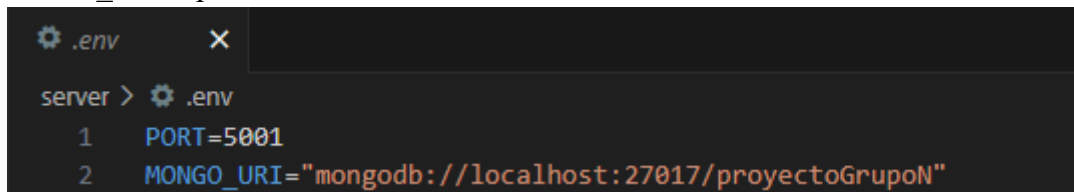
- **Ejecutar el Script**

```
cd server
node generateSecret.js
```

- **Configurar Archivo .env**

Crea/actualiza el archivo .env en la carpeta server:

```
env
PORT=5001
MONGO_URI=mongodb://localhost:27017/proyectoGrupoN
JWT_SECRET=tu_clave_generada_aqui
JWT_EXPIRE=30d
NODE_ENV=production
```



En este caso se creará una nueva conexión en MongoDB Compass en caso de no existir de manera automática al arrancar el programa en local.

- **Eliminar Archivo de Generación (Opcional)**

```
# En Windows
del generateSecret.js
```

```
# En Linux/Mac
rm generateSecret.js
```

## 4. Proceso de Despliegue

La aplicación fue desplegada en la nube utilizando **Render** como plataforma principal. El proceso se realizó de la siguiente manera:

### 1. Preparación del proyecto

- Se validó que tanto **frontend** como **backend** funcionen correctamente en local.
- Se creó un archivo `.env` para cada entorno (desarrollo y producción).
- Se subió el proyecto completo a **GitHub** para poder vincularlo con Render.

Si aún no has creado el repositorio en GitHub, haz esto:

1. Entra a <https://github.com/new>
2. Llena los campos:
  - **Repository name:** proyecto-Grupo-N  
*Público o privado* (como prefieras)
  - (No marques ninguna opción de README o `.gitignore`)
3. Haz clic en **“Create repository”**

Luego, vuelve a tu terminal (Símbolo del sistema o de visual studio code) y ejecuta:

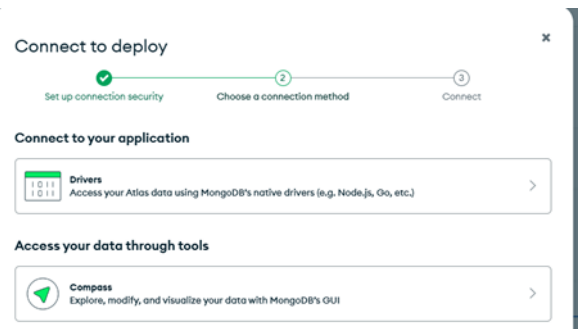
```
D:\Respaldo 2025\Documents\Bootcamp de la fabrica\proyectoGrupoN> git remote set-url origin
```

```
D:\Respaldo 2025\Documents\Bootcamp de la fabrica\proyectoGrupoN>https://github.com/TU_NOMBRE_DE_USUARIO_EN_GIT HUB/proyecto-Grupo-N.git
```

```
D:\Respaldo 2025\Documents\Bootcamp de la fabrica\proyectoGrupoN> git push -u origin main
```

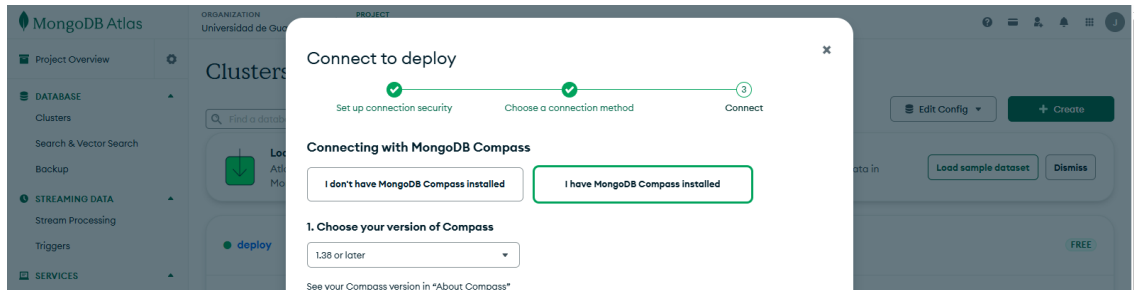
### 2. Preparación del proyecto con MongoDB Atlas

- Crear una cuenta en MongoDB Atlas, de preferencia de email verificación.
- Iniciar sesión en Mongo atlas: <https://account.mongodb.com/account/login>
- Crear un nuevo cluster
  - Escoger un plan: Por ser la primera vez le aparecerá el plan free.
  - Name: deploy
  - Provider: AWS
  - Region: N. Virginia (us-east-1)
  - Tag (optional):
- Crear un usuario de base de datos: usuario y contraseña
- Crear un add current IP ADDRESS: una IP Address y description
- En connect to deploy, nombre escogido previamente en el cluster, hacer el connect to your application con drivers:



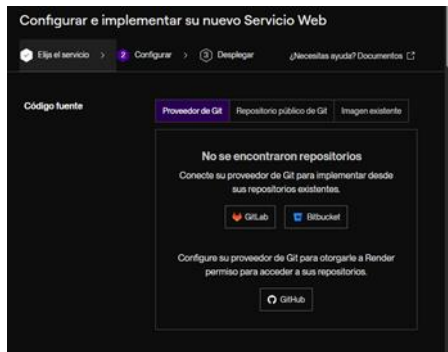
Tener en cuenta el link que se muestra en la sección “3. Add your connection string into your application code”, reemplazar <db\_password> con la contraseña para el usuario de la base de datos.

Si escoge compass le aparecerá esta opción:



### 3. Despliegue en render

1. Ve a <https://render.com>
2. Ingresar a la cuenta de Render de preferencia a usar tu cuenta de GitHub.
3. Haz clic en “create a Project”
4. Haz clic en New → Web services→ New Web Services
5. Vincular el repositorio del proyecto.



- Autorizar a Render, Marca: All repositories
- Debajo verás una lista de permisos como: Lectura y escritura en acciones, commits, despliegues, etc.

## 6. Configurar:

Campo	Valor
Name	proyecto-grupo-n-unificado
Language	Node
Branch	Main
Region	Oregon (US West)
Root Directory	
Git Credentials	Tu correo electrónico
Build Command	cd client && npm install && npm run build && cd ../server && npm install
Pre-Deploy Command	
Start Command	cd server && node index.js
Auto-Deploy	On Commit
Deploy Hook	Dejarla por defecto
Custom Domains	Enabled
PR Previews	Off
Edge caching	Edge caching is only available for paid instances
Service notifications	Use workspace default (Only failure notifications)
Preview environment notifications	Use account default (Disabled)
Health Check path	/healthz
Maintenance Mode	Maintenance mode is only available for paid instances
Instance Type	For hobby projects: Free plan \$0/month 512 (RAM) 0.1 CPU

En “Environment variables” agrega tus variables de entorno:

- PORT: 5001 (Render asigna un puerto dinámico)



- JWT\_SECRET=<clave-secreta>
- JWT\_EXPIRE=30d
- NODE\_ENV=production
- REACT\_APP\_API\_URL=<API que dió el render>
- MONGODB\_URI=<URI-de-MongoDB-Atlas>

No pongas la contraseña con < > en el valor de la variable environment de MONGO\_URI.

## 7. Clic en Create Web Service

Resultado esperado:

```

i ==> Uploaded in 28.4s. Comp
i ==> Build successful 🎉
i ==> Deploying...

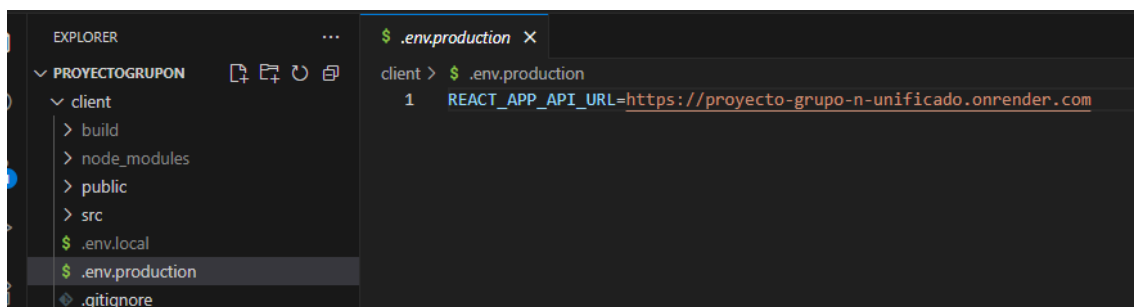
```

```

① Conectado a MongoDB
① Conectado a MongoDB para inserción de datos iniciales
① Insertando datos iniciales...
① Usuarios iniciales insertados correctamente
① Gorras iniciales insertadas correctamente
① Datos iniciales insertados correctamente
① Servidor ejecutándose en puerto 5001
① API disponible en: http://localhost:5001/api
① Endpoints principales:
① - GET http://localhost:5001/api/gorras
① - POST http://localhost:5001/api/usuarios/login
① - POST http://localhost:5001/api/usuarios/registro
① ==> Your service is live 🎉
① ==>

```

Actualizar con el link que lanza de: <https://proyecto-grupo-n-unificado.onrender.com> en .env.production:



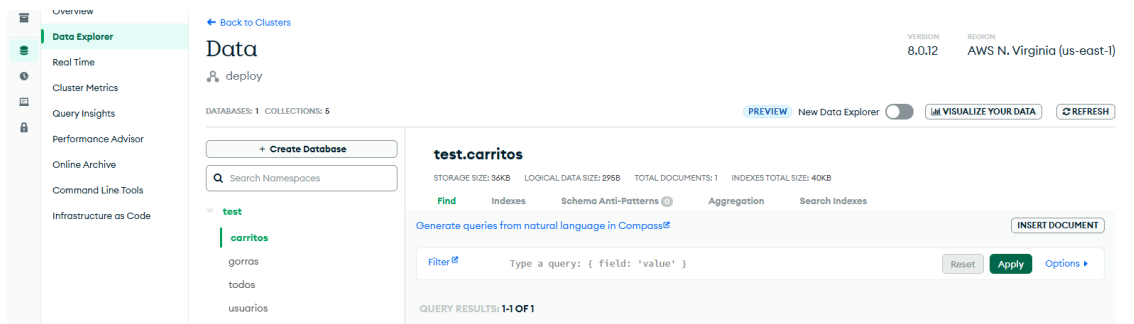
```

EXPLORER
PROYECTOGRUPON
├── client
│   ├── build
│   ├── node_modules
│   ├── public
│   ├── src
│   ├── .env.local
│   ├── .env.production
│   └── .gitignore
└── ...

$.env.production X
client > $.env.production
1 REACT_APP_API_URL=https://proyecto-grupo-n-unificado.onrender.com

```

Se puede revisar los datos que se han ingresado en mongo db Atlas:

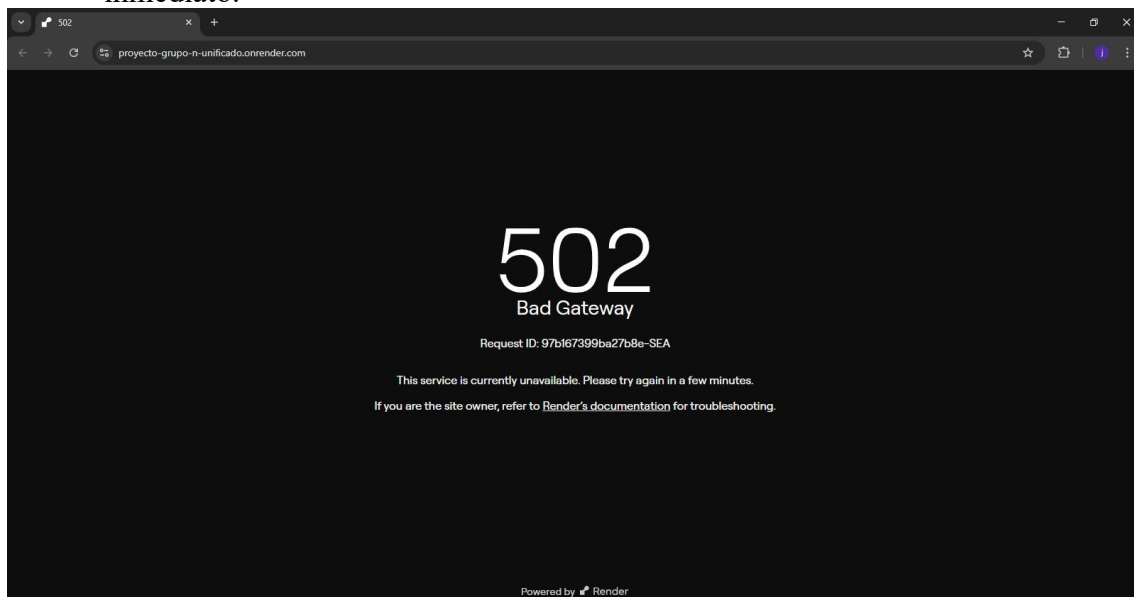


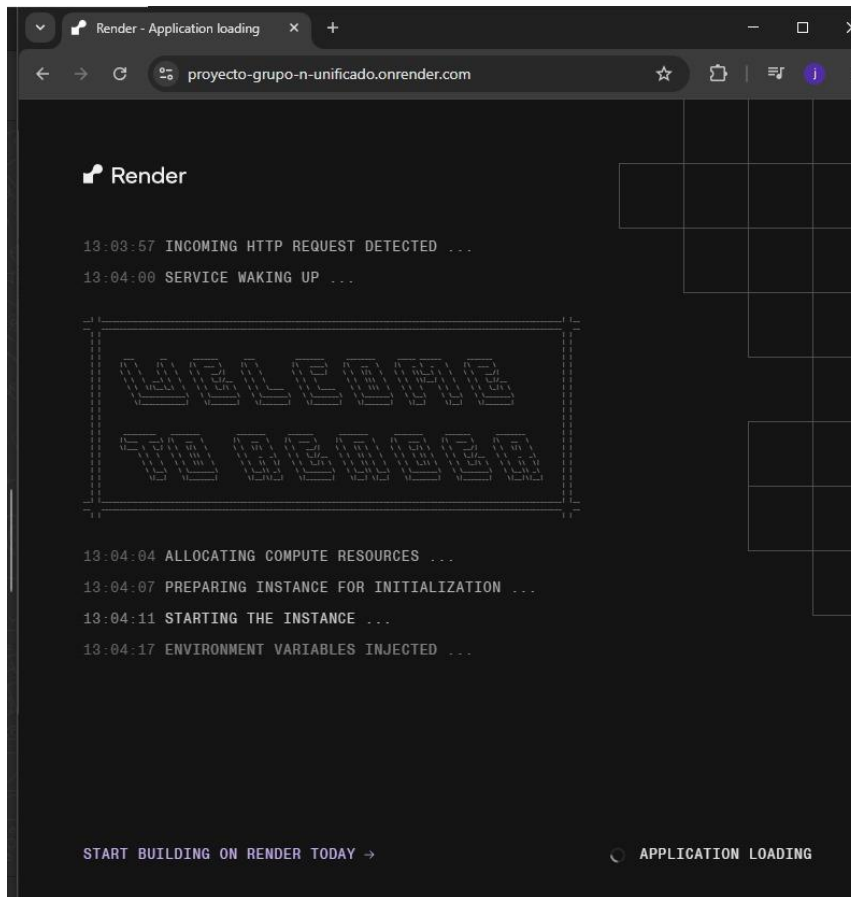
## NOTA IMPORTANTE:

Render, en su plan gratuito, suspende automáticamente las aplicaciones que no reciben tráfico durante cierto tiempo pueden devolver temporalmente un mensaje de 502 Gateway "This service is currently unavailable. Please try again in a few minutes.". Cuando alguien vuelve a acceder a la página, Render reactiva la aplicación, pero eso puede tardar entre 30 segundos y 2 minutos, lo que puede parecer que el sitio está caído cuando no es así.

¿Qué puedes hacer?

- Paciencia al primer acceso: Espera unos segundos si el enlace no responde de inmediato.





- Puedes abrir la app desde el panel de Render para "despertarla" antes de compartir el link.
- Si el presupuesto lo permite, actualiza a un plan de pago, así el servicio permanece activo 24/7 sin suspensión.

## 5. Resultado Final

La aplicación quedó desplegada en la nube:

- **Link del Proyecto desplegado:**

<https://proyecto-grupo-n-unificado.onrender.com>

- **Github Link:** <https://github.com/CastilloJoshuaEE/proyecto-Grupo-N.git>

- Video demostrativo: <https://youtu.be/C7x8YmRnmlY>

- Video demostrativo + Código en zip:

<https://drive.google.com/drive/folders/1vcrxQa8Xm54H3GPOP0h2gQeMu3ZBlbCM?usp=sharing>

## 6. Desafíos y Soluciones

- Uno de los inconvenientes fue que olvidamos configurar correctamente la variable MONGO\_URI en el environment del panel de Render. Inicialmente, pusimos la contraseña dentro de los símbolos < >, lo que generaba error de conexión. La solución

fue sencilla: asegurarnos de escribir el valor completo de la variable sin incluir los caracteres < >.

- Otro problema fue que, en algunos momentos, aparecía el mensaje "**This service is currently unavailable. Please try again in a few minutes.**" junto con el error **502 Gateway**. En un inicio pensamos que el sitio había fallado por alguna mala configuración, pero al revisar todo comprobamos que el error se debía a Render. En su plan gratuito, Render suspende automáticamente las aplicaciones que no reciben tráfico durante un tiempo, y al volver a acceder puede tardar entre 30 segundos y 1 minuto en reactivarse, dando la impresión de que el servicio está caído cuando en realidad no lo está. Frente a esto, discutimos la posibilidad de cambiar a un plan de pago para evitar estas interrupciones, aunque no llegamos a una decisión final.