



Nuevo proyecto

Manual de instrucciones
2022

Tabla de contenidos

1. Aplicación BSALE TEST	3
1.1. Inicio del Proyecto	5
2. configuración de la conexión	6
Configuración de app	8
3. Middlewares.....	11

1. Aplicación BSALE TEST

Se creará una página de una tienda que utilice la base de datos alojada en heroku que se proporciona.

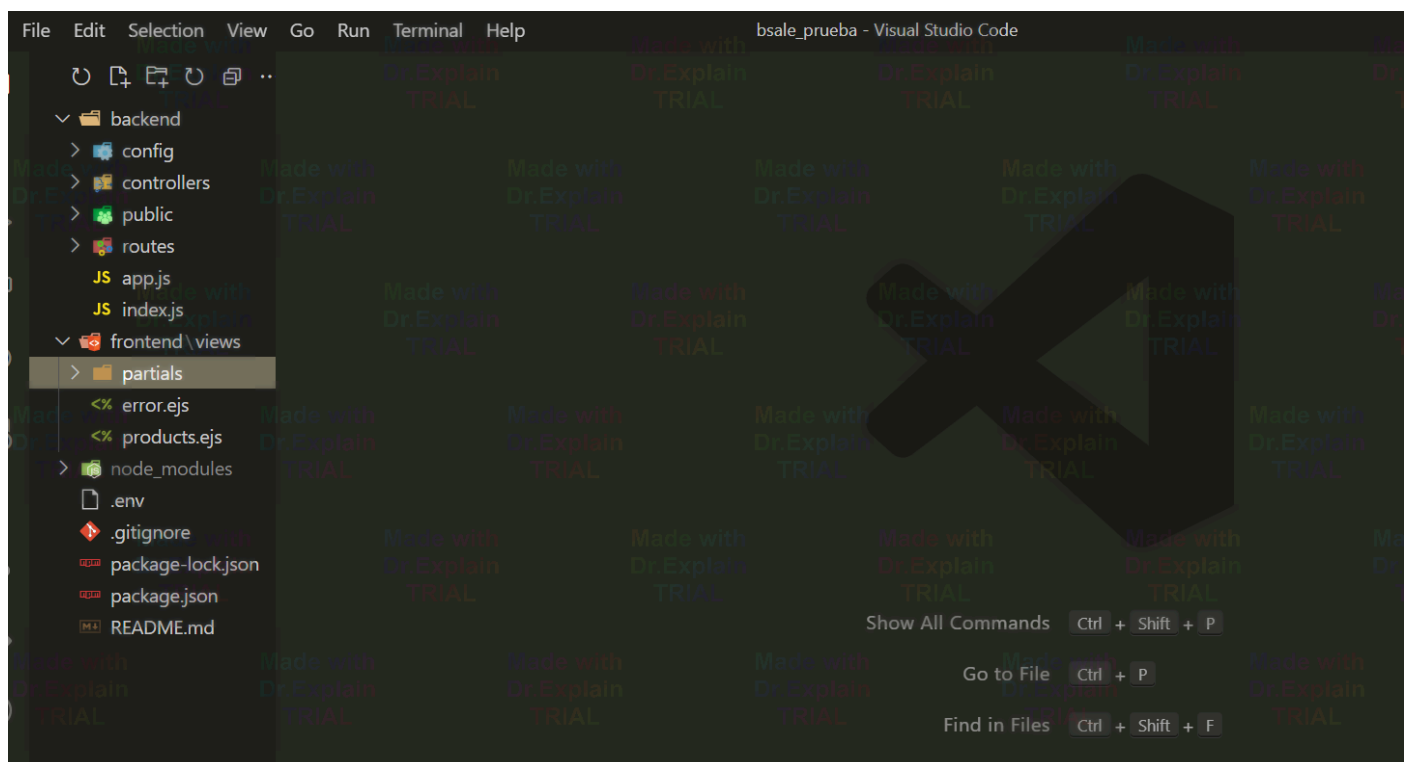
Se procede a crear las carpetas en del BACKEND y FRONTEND para separar cada etapa.

Se trabajará con la metodología MVC (Model View Controller) por lo que dentro de la carpeta BACKEND se crearán las carpetas:

- controllers: donde se manejará la conexión con la base de datos y las rutas (ENDPOINT)
- routes: las rutas o endpoints
- config: la configuración de la base de Datos
- public: donde se guardaran los archivos de imagenes, estilos o JS adicionales

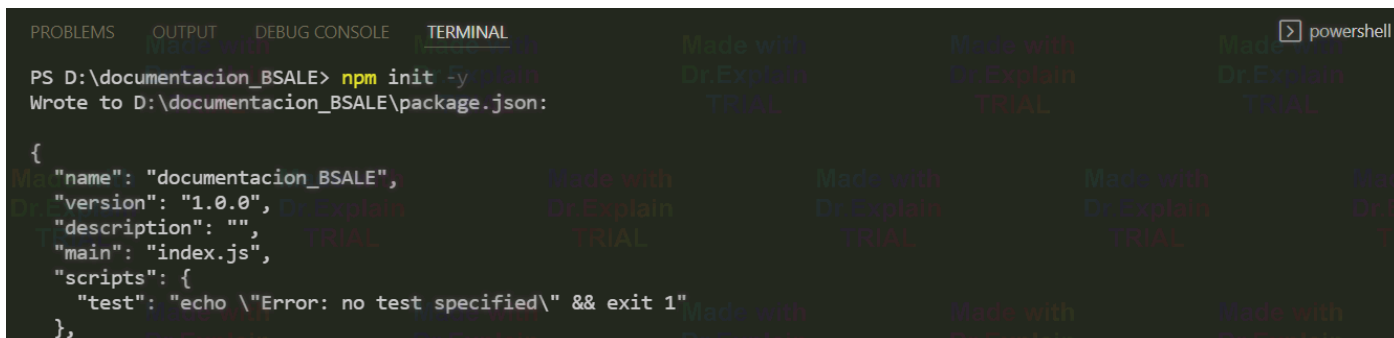
y dentro de FRONTEND se creará la carpeta VIEWS. Dentro de VIEW, por lo general se guarda una carpeta PARTIALS.

Es un proyecto pequeño y bastaría con una sola página para mostrar toda la tienda. Sin embargo por buenas practicas se recomienda colocar una pagina de header y otra de footer que se utilizará como plantillas de todas las páginas.



1.1. Inicio del Proyecto

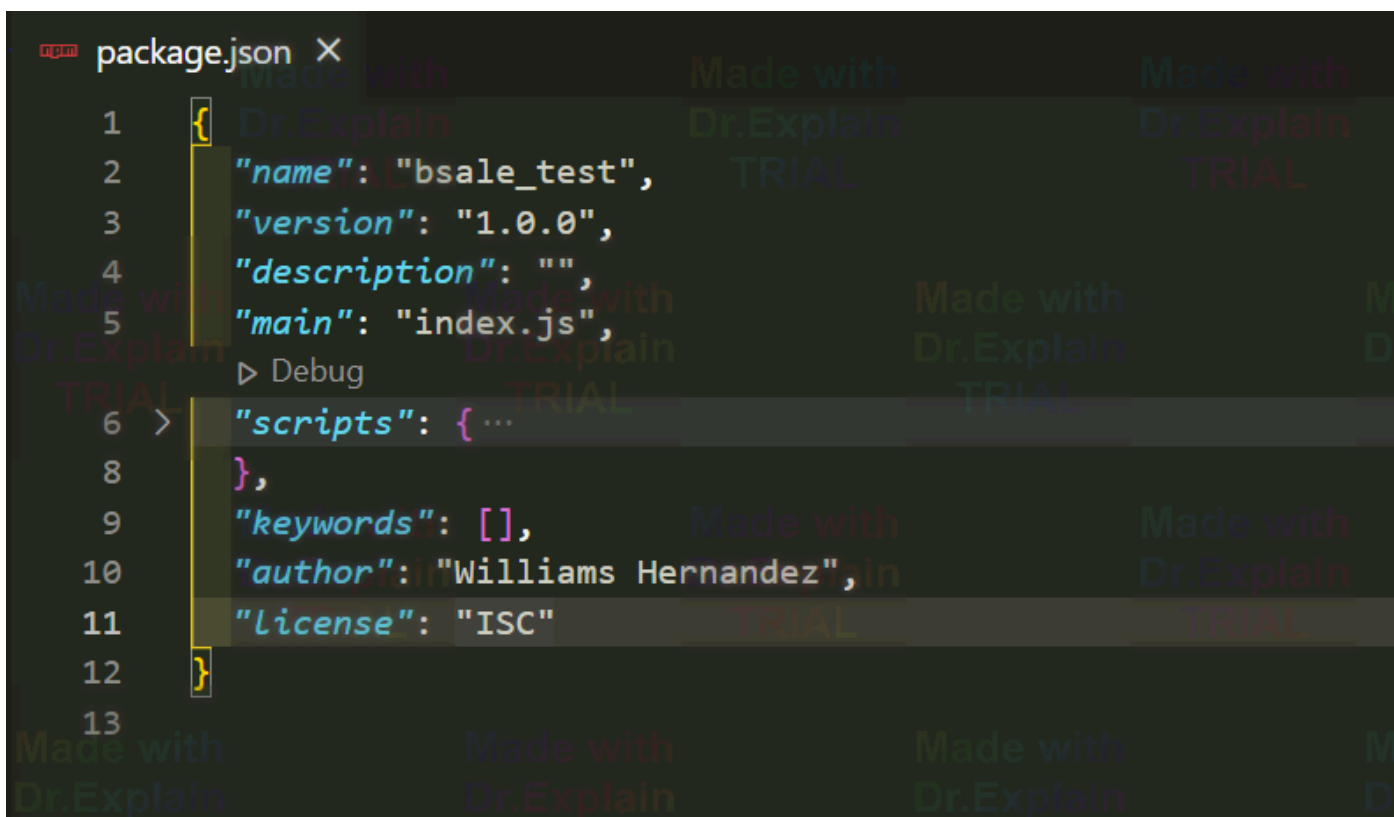
Se comienza con el comando:



```
PS D:\documentacion_BSALE> npm init -y
Wrote to D:\documentacion_BSALE\package.json:

{
  "name": "documentacion_BSALE",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "Williams Hernandez",
  "license": "ISC"
}
```

Con ello se creará el archivo package.json donde se guardarán todas las dependencias utilizadas en el proyecto.

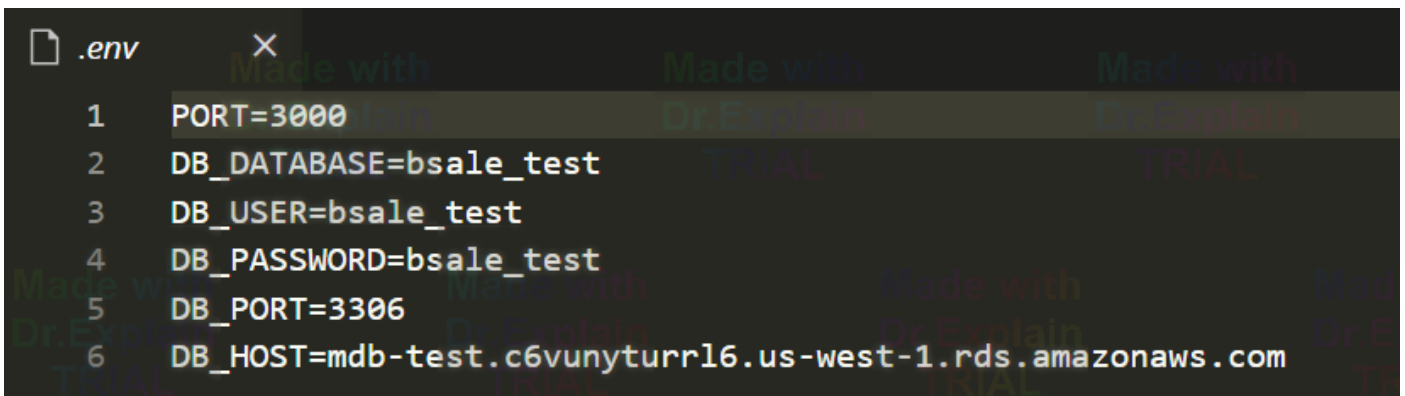


```
package.json
1  {
2    "name": "bsale_test",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"
8    },
9    "keywords": [],
10   "author": "Williams Hernandez",
11   "license": "ISC"
12 }
```

2. configuración de la conexión

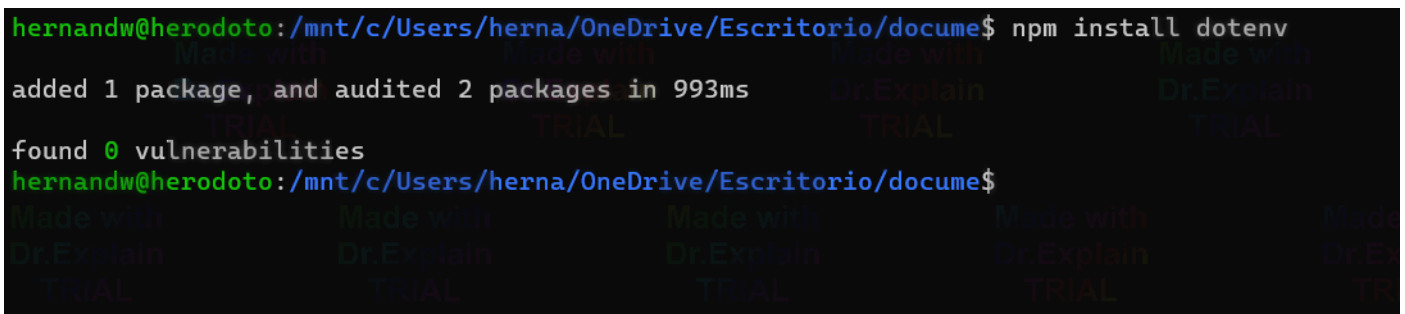
Dentro de la carpeta CONFIG, se guardará la conexión a la Base de Datos en un archivo con igual nombre. Como se utilizará luego HEROKU para el deploy (Despliegue en Producción). Esta configuración será con el motor de base de datos MySQL. para efectos de este proyecto la Base de datos y las tablas utilizadas son proporcionadas. Por lo que no será necesario crearlas.

Para mayor seguridad de igual forma se guardarán los datos en variables de entorno, porque así no serán expuestas los datos en el repositorio de GIT. Para manejar las variables de entorno se instala la dependencia DOTENV y se guardan en un archivo .env en la raíz del proyecto.



```
.env
1 PORT=3000
2 DB_DATABASE=bsale_test
3 DB_USER=bsale_test
4 DB_PASSWORD=bsale_test
5 DB_PORT=3306
6 DB_HOST=mdb-test.c6vunyturrl6.us-west-1.rds.amazonaws.com
```

Procedemos a instalar dotenv:



```
hernandw@herodoto:/mnt/c/Users/herna/OneDrive/Escritorio/docume$ npm install dotenv
added 1 package, and audited 2 packages in 993ms
found 0 vulnerabilities
hernandw@herodoto:/mnt/c/Users/herna/OneDrive/Escritorio/docume$
```

y ahora guardamos la configuración en el archivo config.js:

```
JS config.js X
1 require("dotenv").config({ path: ".env" });
2 const mysql = require("mysql2");
3
4 //Datos de Conexión
5 const connection = mysql.createConnection({
6   host: process.env.DB_HOST,
7   user: process.env.DB_USER,
8   password: process.env.DB_PASSWORD,
9   database: process.env.DB_DATABASE,
10 });
11
12 module.exports = connection;
13
```

En este caso utilizamos variables de entorno y ya no es necesario colocar los datos visibles. Hacemos una conexión al archivo .env para conectarnos con los datos, contraseñas de la base de Datos.

Para evitar que estos archivos se suban al guardar el proyecto crearemos un archivo .gitignore donde se colocan de los archivos y carpetas que no deben subirse a github.

```
.gitignore X
1 node_modules
2 .env
3 package-lock.json
```

En nuestro caso la carpeta node_modules

Configuración de app

dentro de la carpeta BACKEND se creará un archivo app.js e index.js. En el primero se guardará todas las conexiones a la base de datos, se creará el servidor. Conexión a los controladores y rutas. Además de la configuración del motor de plantilla, carpeta publica y middlewares.

Procedemos primero a instalar las dependencias que utilizaremos:

- npm install mysql
 - npm install morgan
 - npm express
- o también se pueden instalar los 3 en un mismo comando

```
hermandw@herodoto:/mnt/c/Users/herna/OneDrive/Escritorio/docume$ npm i express morgan mysql2
added 69 packages, and audited 71 packages in 8s
2 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
hermandw@herodoto:/mnt/c/Users/herna/OneDrive/Escritorio/docume$
```

Procedemos a llamar al servidor express y configurar el puerto

```
require("dotenv").config({ path: ".env" });
const express = require("express");
const app = express();
const path = require("path");
const morgan = require("morgan");
// Configuraciones
const PORT = process.env.PORT || 3000;

// Levantamiento del servidor
app.listen(PORT, async (req, res) => {
  console.log(`Servidor activo on port: ${PORT}`);
});
```

1. La primera línea: El archivo que vincula a las variables de entorno
2. 2da y 3ra línea llamamos al servidor express y configuramos una constante llamada app que nos permitirá conectarnos al servidor.

3. la consta PATCH es propia de node, por lo que procedemos solo a llamarla. Ella nos permitirá al configurar los middlewares, los views, los archivos static y la carpeta publica poder tener acceso sin necesidad de escribir toda la ruta.

La dependencia morgan, nos permitirá ver las llamadas a los middleware desde la consola.

y finalmente levantamos el servidor de Express a través del PORT. Este constante lo definimos como variable de entorno, y en caso de que no se encuentre que conectará al puerto 3000. Una vez hagamos el deploy, heroku asignará este puerto de manera aleatoria.

Establecemos la conexión a la Base de Datos con llamando al archivo config y verificamos que la conexión sea correcta. En caso de que sea correcta nos mostrará por consola el mensaje "Database server running!" y en caso que no logré conectarse nos mostrará un error:

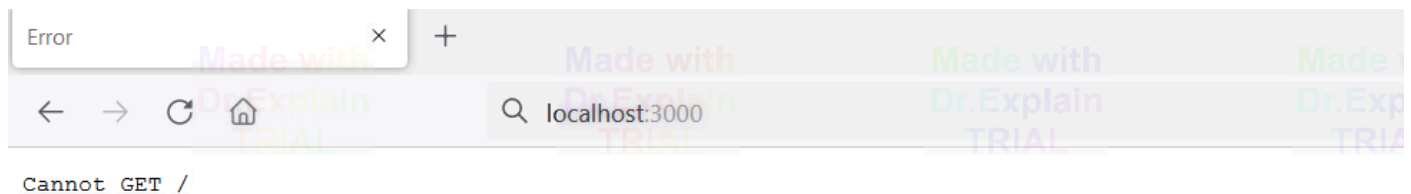
```
5  const morgan = require("morgan");
6  const connection = require('./config/config')
7
8  // Configuraciones
9  const PORT = process.env.PORT || 3000;
10
11
12  // Check connect
13  connection.connect(error => {
14    if (error) throw error;
15    console.log('Database server running!');
16  });
17
18  //Levantamiento del Servidor
19  app.listen(PORT, async(req, res) => {
20    console.log(`Servidor activo on port: ${PORT}`);
21  });
```

Si la configuración del servidor es correcta, nos aparecerá por consola:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\documentacion_BSALE> node backend/app
Servidor activo on port: 3000
Database server running!
```

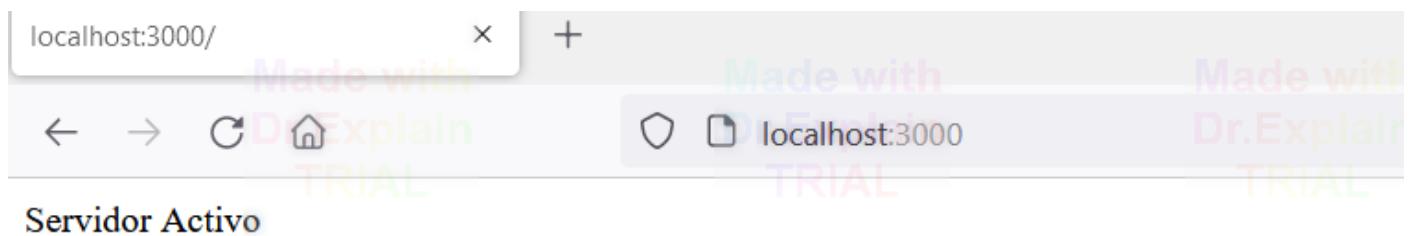
en el navegador podemos ver ya el servidor aunque todavia no hemos configurado una ruta que nos permita navegar.



para que estemos seguros de que está cargando correctamente crearemos un endpoint o ruta que nos permitirá generar un mensaje en el navegador si la conexión es correcta:

```
17
18
19 app.get('/', (req, res) => {
20   res.send('Servidor Activo');
21 });
22
```

y ahora se nos visualiza en el navegador lo siguiente:



3. Middlewares

Cuando instalamos la dependencias dijimos que MORGAN nos permitirá ver mensajes por consola una vez que accedamos a una ruta del navegador, con un mensaje con código 200 si la conexión es correcta o con un error. Para ello primero debemos llamar a la const morgan y luego configurar su llamado.

```
5  const morgan = require("morgan");
6  const connection = require('./config/config');
7  const morgan = require('morgan'); plain
8  TRIAL TRIAL
```

y luego la configuración en el archivo app:

```
//Middlewares
app.use(morgan("dev"));
```

eso nos permitirá ver por consola la conexión. Si volvemos a ejecutar el servidor veremos lo siguiente por consola:

```
PS D:\documentacion_BSALE> node backend/app
Servidor activo on port: 3000
Database server running!
GET / 304 2.368 ms - -
```

El código 304, no se considera error, lo que nos indica que no hubo cambios en la ventana, por ende el navegador puede usar la misma versión almacenada en su caché.

cada vez que hacemos un cambio en el archivo app debemos ejecutar el comando `node backend/app` para ver los resultados en el navegador. Lo que hace que el trabajo desarrollado sea muy tedioso. para ello instalaremos una dependencia que nos permitirá ver los cambios automáticamente, sin necesidad de estar levantando el servidor cada vez que realizamos un cambio. NODEMON levantará el servidor automáticamente, como es una dependencia de desarrollo y no la subiremos a producción las instalamos con el siguiente comando

```
> npm install --save-dev nodemon
```

veremos que en el archivo `package.json` se verá en una sección aparte indicándonos que es una dependencia pero de desarrollo:

```

2  "name": "bsale_test",
3  "version": "1.0.0",
4  "description": "",
5  "main": "index.js",
  > Debug
6  "scripts": {
7    "test": "echo \\\"Error: no test specified\\\" && exit 1"
8  },
9  "keywords": [],
10 "author": "Williams Hernandez",
11 "license": "ISC",
12 "dependencies": {
13   "dotenv": "^16.0.0",
14   "express": "^4.17.3",
15   "morgan": "^1.10.0",
16   "mysql2": "^2.3.3"
17 },
18 "devDependencies": {
19   "nodemon": "^2.0.15"
20 }

```

aprovechamos de configurar los middlewares que nos permitan poder leer archivos JSON en el navegador

```

//middleware
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(morgan("dev")); // ver las consultas en la consola

```

Express.json() y express.urlencoded nos permiten leer archivos de json en el navegador. Son dependencias de express y que no son necesarias de instalar. Antiguamente debia instalarse body-parser para utilizarlas.

Created with Dr.Explain
Unregistered version