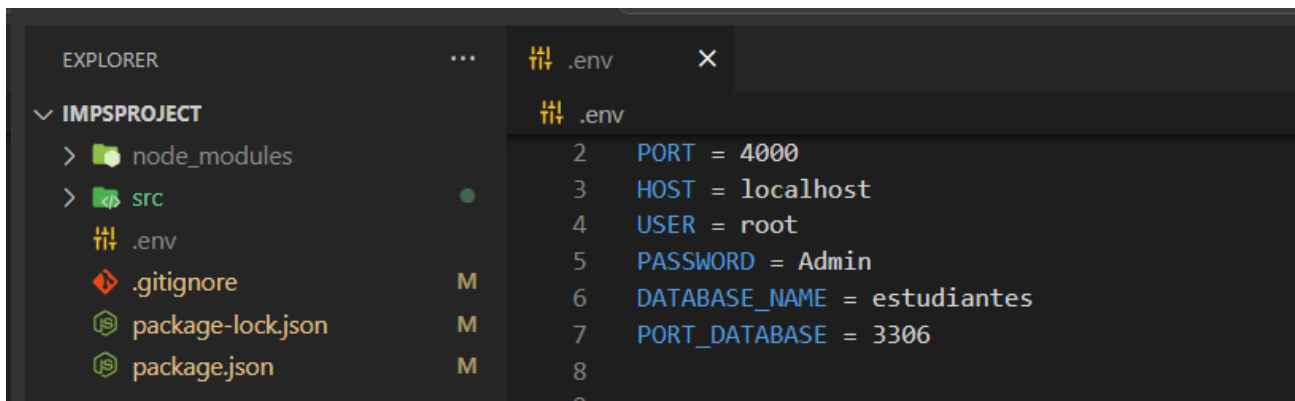


## Conexión a base de datos y configuración de variables de entorno

Para la parte de base de datos utilizaremos MySQL 5.7, anexo enlace de descargue:

<https://dev.mysql.com/downloads/windows/installer/5.7.html>

Una vez tenga completa la instalación de MySQL, corra el script que se encuentra en el aula virtual (Sinapsis). Luego ubíquese en la raíz del proyecto y cree un nuevo archivo con el extensión **.env** el cual contendrá lo siguiente



```
EXPLORER
  IMPSPROJECT
    > node_modules
    > src
    .env
    .gitignore
    package-lock.json
    package.json

.env
2 PORT = 4000
3 HOST = localhost
4 USER = root
5 PASSWORD = Admin
6 DATABASE_NAME = estudiantes
7 PORT_DATABASE = 3306
8
9
```

Donde:

**PORT** = es el número de puerto donde se iniciará el servidor

**HOST** = es la dirección donde se alojará la aplicación

**USER** = es el nombre usuario configurado al momento de instalar MySQL

**PASSWORD** = es la contraseña elegida al momento de instalar MySQL

**DATABASE\_NAME** = es el nombre de la base de datos

**PORT\_DATABASE** = es el puerto de la base de datos por defecto es 3306  
caso contrario debe indicar el número que usted configuró

## Luego instale el paquete siguiente

**npm install dotenv**

Posteriormente diríjase al archivo **index.js** que se encuentra en la raíz de la carpeta **src** y agregue la siguiente línea que permite configurar el paquete instalado anteriormente, para manejar las variables de entorno

```
require('dotenv').config()
```

El archivo debe quedarle así

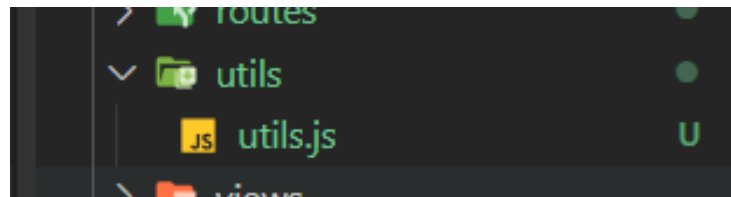


```
src > JS index.js M X
1  const express = require('express');
2  // Inicializaciones
3  const app = express();
4
5  require('dotenv').config()
6
7  // Ajustes del servidor
8  app.set('port', process.env.PORT || 4500);
9
10 // Iniciar el servidor
11 app.listen(app.get('port'), () => {
12   console.log('Servidor iniciado en el puerto: ', app.get('port'));
13 });
```

Como puede notar la configuración del puerto ahora se toma del archivo **.env** creado anteriormente, de tal forma que, si dicho número de puerto es nulo. Se tomará el número de puerto **4500** tal como se indica.

## Creación de carpeta para utilidades

En la raíz de la carpeta **src** cree una nueva carpeta con el nombre **utils**, dentro de ella agregue un archivo con el nombre **utils.js** debe quedarle así



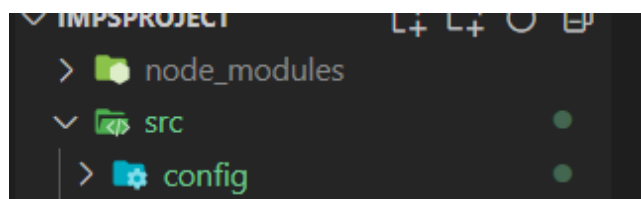
En el archivo que acaba de crear agregue las siguientes líneas de código, que son algunas constantes de errores al momento de conectarnos a la base de datos

```
src > utils > JS utils.js > <unknown> > CONSTANTS
1  module.exports = {
2  ...
3    CONSTANTS: {
4      // *** Algunos codigos de error con la base de datos ***
5      PROTOCOL_CONNECTION_LOST: 'PROTOCOL_CONNECTION_LOST',
6      ER_CON_COUNT_ERROR: 'ER_CON_COUNT_ERROR',
7      ECONNREFUSED: 'ECONNREFUSED',
8      ER_ACCESS_DENIED_ERROR: 'ER_ACCESS_DENIED_ERROR',
9      // *** Fin bloque codigos base de datos ***
10     }
11   }
12
```

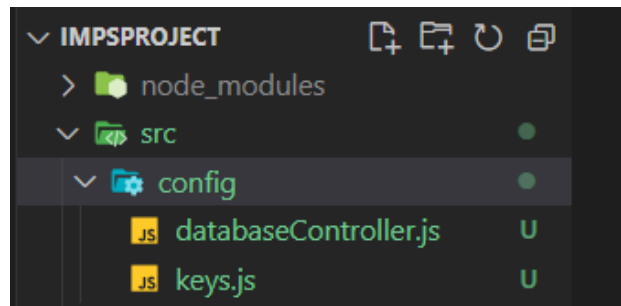
Cabe resaltar que esos solo son algunos errores comunes.

## Creación de carpeta para configuraciones

Siempre en la raíz de la carpeta **src** agregue una nueva carpeta con el nombre **config** debe quedar así



Dentro de la carpeta que acaba de crear añada un archivo con el nombre **databaseController.js** que tal como su nombre lo indica servirá para configurar la conexión a la base de datos, siempre dentro de la carpeta **config** agregue otro archivo llamado **keys.js** el cual nos permitirá armar un objeto con las configuraciones para establecer conexión con la base de datos, la estructura debe verse así



Abra el archivo **keys.js** y agregue las líneas siguientes

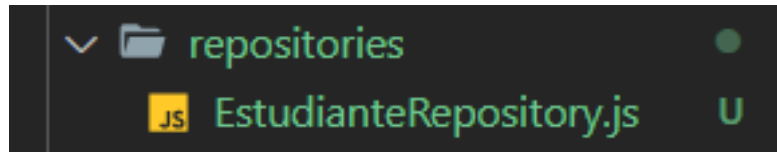
```
js keys.js U x
src > config > js keys.js > ...
1  require('dotenv').config()
2
3  module.exports = {
4      // Configurando objeto para inyectarlo en el pool de conexiones
5      database: {
6          host: process.env.HOST,
7          user: process.env.USER,
8          port: process.env.PORT_DATABASE,
9          password: process.env.PASSWORD,
10         database: process.env.DATABASE_NAME
11     }
12 }
```

En archivo **databaseController.js** agregue lo siguiente

```
src > config > .js databaseController.js > pool.getConnection() callback
1  const mysql = require('mysql2');
2  const { promisify } = require('util');
3  const { database } = require('./keys');
4  const { CONSTANTS } = require('../utils/utils');
5
6  const pool = mysql.createPool(database); // Se crea el pool de conexiones
7
8  // Iniciando conexion con la base de datos
9  pool.getConnection((error, conexion) => {
10     // Validar si la conexion tiene algun tipo de error
11     if (error) {
12         // Validando codigos de error mas comunes
13         switch (error.code) {
14             case CONSTANTS.PROTOCOL_CONNECTION_LOST:
15                 // Indica que la conexion con la base de datos está perdida
16                 console.error('DATABASE CONNECTION WAS CLOSED');
17                 break;
18                 // Indica que existen demasiadas conexiones
19             case CONSTANTS.ER_CON_COUNT_ERROR:
20                 console.error('DATABASE HAS TO MANY CONNECTIONS');
21                 break;
22                 // Indica que la conexion fue rechazada
23             case CONSTANTS.ECONNREFUSED:
24                 console.error('DATABASE CONNECTION WAS REFUSED');
25                 break;
26                 // Indica que el acceso esta denegado
27             case CONSTANTS.ER_ACCESS_DENIED_ERROR:
28                 console.error('ACCESS DENIED FOR USER');
29                 break;
30             default:
31                 console.error('Error de base de datos no encontrado');
32                 break;
33         }
34     }
35
36     // Si la conexion es exitosa, imprimir un mensaje indicandolo
37     if(conexion) {
38         console.log('Conexion establecida con la base de datos');
39         conexion.release();
40     }
41
42     return;
43 });
44
45 // Configurando PROMISIFY para permitir en cada consulta un async/await (promesas)
46 pool.query = promisify(pool.query);
47
48 module.exports = pool;
```

## Creación de carpeta repositorio

Siempre en la raíz de la carpeta **src** agregue una nueva carpeta con nombre **repositories** dentro de esta, agregue un nuevo archivo llamado **EstudianteRepository.js** la estructura debe lucir así

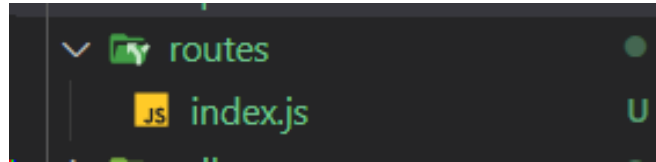


Ahora dentro del archivo **EstudianteRepository.js** agregue las siguientes líneas de código

```
JS EstudianteRepository.js U X
src > repositories > JS EstudianteRepository.js > ...
1  const pool = require('../config/databaseController');
2
3  module.exports = {
4
5      // Consulta para obtener todos los estudiantes
6      obtenerTodosLosEstudiantes: async() => {
7          try {
8              const result = await pool.query('SELECT * FROM estudiantes');
9              return result;
10         } catch (error) {
11             console.error('Ocurrió un problema al consultar la lista de estudiantes: ', error);
12         }
13     }
14 }
```

Estas líneas permiten consultar la información de la tabla **estudiantes** en la base de datos previamente creada.

Diríjase a la carpeta **src/routes** y agregue un nuevo archivo llamado **index.js** la estructura debe lucir así



Dentro de ese archivo coloque las líneas de código siguientes

```
JS index.js U X
src > routes > JS index.js > ...
1 // Este archivo sera utilizado para configurar todas las rutas principales del sistema
2 const express = require('express');
3 const router = express.Router();
4 const estudianteRepository = require('../repositories/EstudianteRepository');
5
6
7 // Configuración de ruta inicial de la aplicación
8 router.get('/', async (request, response) => {
9   // Probando conexión con la base de datos.
10   const lstEstudiantes = await estudianteRepository.obtenerTodosLosEstudiantes();
11   console.log('Listado: ', lstEstudiantes);
12
13   response.send('Bienvenido al laboratorio de IMPS');
14 });
15
16 module.exports = router;
17
```

Lo que hacen esas líneas de código es, configurar una ruta inicial para la aplicación, es decir, se configura un endpoint de entrada.

Ahora diríjase al archivo **index.js** que se encuentra en la raíz de la carpeta **src** y agregue la siguiente línea que permite configurar una ruta, básicamente es un middleware de configuración de ruta

```
// Configuración de rutas  
app.use(require('./routes')); // Node automáticamente busca el index.js del módulo
```

El archivo debe lucir así

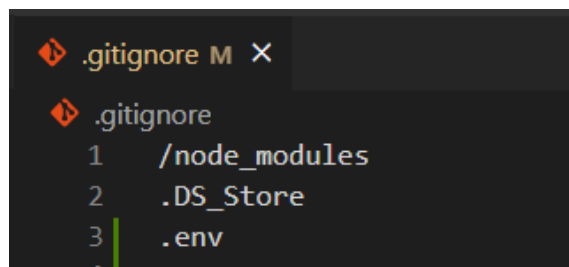


```
JS index.js M X  
src > JS index.js > ...  
1  const express = require('express');  
2  // Inicializaciones  
3  const app = express();  
4  
5  require('dotenv').config()  
6  
7  // Ajustes del servidor  
8  app.set('port', process.env.PORT || 4500);  
9  
10 // Configuración de rutas  
11 app.use(require('./routes')); // Node automáticamente busca el index.js del módulo  
12  
13 // Iniciar el servidor  
14 app.listen(app.get('port'), () => {  
15   console.log('Servidor iniciado en el puerto: ', app.get('port'));  
16 });  
17  
18
```



## Excluyendo el archivo .env al momento de subir cambios al repositorio en GitHub

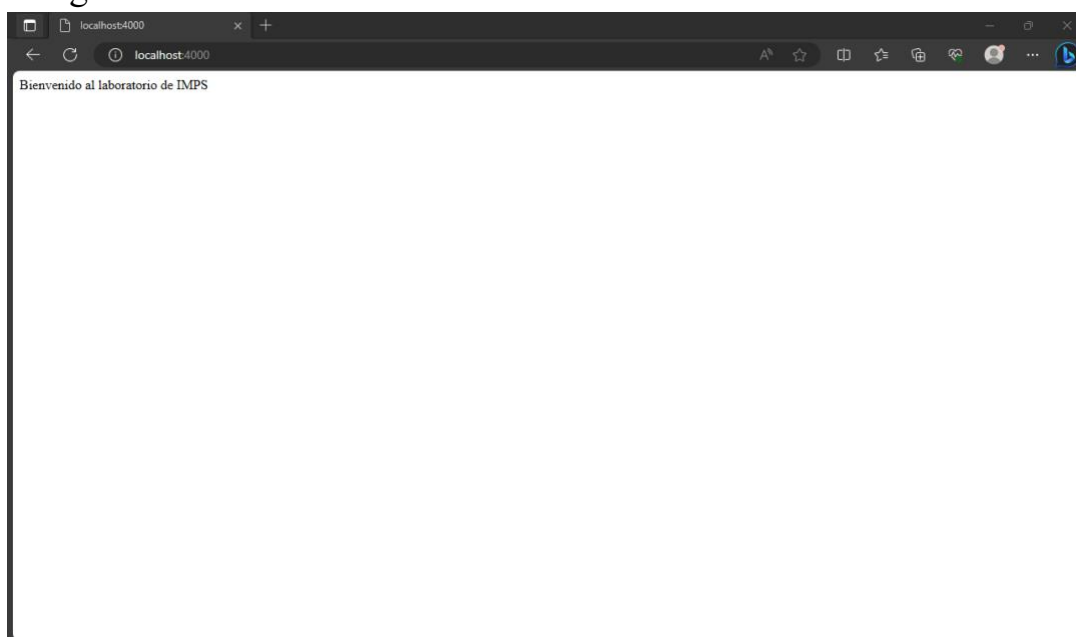
Por tratarse de datos sensibles sobre nuestra conexión a la base de datos, se recomienda no subirlos al repositorio alojado en la nube. De tal forma que, abra el archivo **.gitignore** y agregue el archivo **.env** así



```
.gitignore M X
.gitignore
1 /node_modules
2 .DS_Store
3 .env
```

## Corriendo la aplicación y probando ruta inicial

Abra la terminal en Visual Studio Code y ejecute el comando **npm run dev** para iniciar el servidor, al momento que cargue la aplicación deberá visualizar lo siguiente en su navegador



En la terminal deberá visualizar los siguientes datos

```
PS C:\Users\Dimas\Desktop\Proyectos\IMPSPProject> npm run dev

> impsproject@1.0.0 dev
> nodemon src/

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/`
Servidor iniciado en el puerto: 4000
Conexion establecida con la base de datos
Listado: [
  {
    idestudiante: 'ud87-a98-b40',
    nombre: 'Sarita',
    apellido: 'Brummell',
    email: 'sbrummell6@google.es',
    idcarrera: 'I04',
    usuario: 'OctavioConn'
  },
  {
    idestudiante: 'wo31-d04-a04',
    nombre: 'Peter',
    apellido: 'Maddern',
    email: 'pmaddern7@elpais.com',
    idcarrera: 'I04',
    usuario: 'TijuanaJaskolski'
  },
]
```

## Tarea

- Crear commit con el nombre “Conexión a base de datos”
- Subir los cambios a GitHub
- En el espacio habilitado en sinapsis, suba un PDF con una captura de pantalla del commit, esto lo puede hacer utilizando el comando **git log --oneline**

Anexo captura de como deberá verse la imagen que usted subirá

```
● PS C:\Users\Dimas\Desktop\Proyectos\IMPSPProject> git log --oneline
f816989 (HEAD -> main, origin/main) Conexion a base de datos
45a5a76 Inicio del proyecto
○ PS C:\Users\Dimas\Desktop\Proyectos\IMPSPProject> █
```