

## **Laboratorio N.3.**

**Curso:** ST0255 Telemática.

**Título:** Desplegando una Web App considerando Front y Back end.

**Objetivo:** Desarrollar habilidades que permita el desplegar una aplicación web teniendo en cuenta una arquitectura cliente/servidor así como de “N” capas (layer) (presentación/lógica de negocio/persistencia de datos) distribuida en diferentes máquinas.

**Duración:** 45 mins.

### **1. Introducción**

En la actualidad el despliegue de las aplicaciones es una parte fundamental para el éxito de las organizaciones. Igualmente, no se puede negar que, muchas de las aplicaciones se desarrollan considerando una vista web. Es por esta razón, que entender los elementos que componen una aplicación web (el cliente así como servidor), el protocolo (http) que soporta la comunicación entre los elementos así como la arquitectura, tanto a nivel de software como de despliegue, es un aspecto fundamental para el proceso de formación en el área de la computación.

En este laboratorio, se desarrollaran las habilidades necesarias para el despliegue de una aplicación web que considera una arquitectura de 3 capas (presentación/lógica de negocio/persistencia de datos).

### **2. Recursos**

Para el desarrollo de este laboratorio se requiere una cuenta en AWS que le permita desplegar diferentes instancias EC2 para instalar sistema operativo Amazon Machine Image (AMI).

### **3. Descripción de la Aplicación Web**

El proyecto a desplegar en este laboratorio es una aplicación web. La aplicación permite visualizar una colección de recursos, para efectos de este caso, libros. Igualmente, cuando el usuario selecciona alguno de los recursos, se ofrece una vista con información detallada sobre el recurso seleccionado. La información de los recursos (libros) se encuentra almacenada en base de datos. La aplicación tiene tres (vistas): raíz (“/”, home), descripción detallada de los recursos libros y acerca de.

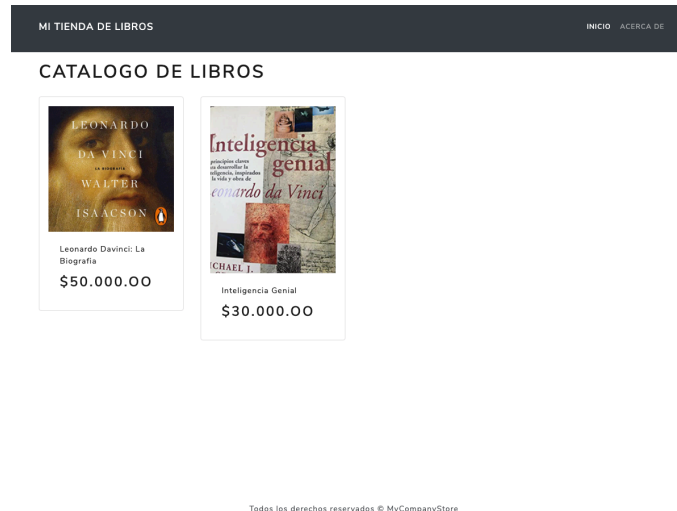


Figura 1. Vista del home de la aplicación.

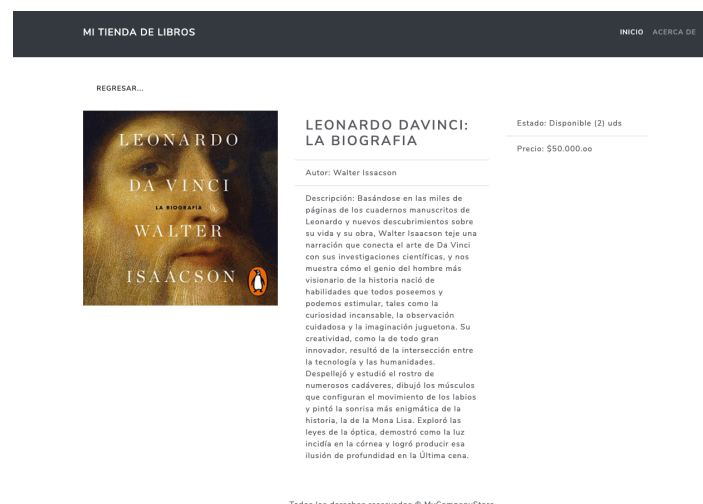


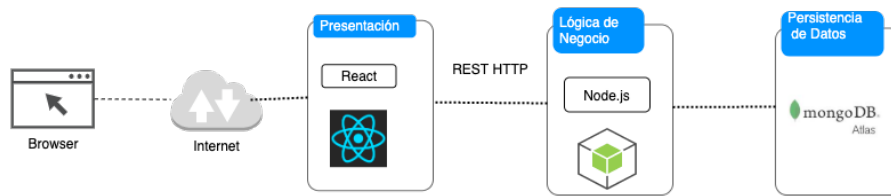
Figura 2. Vista detallada para un objeto libro.



Figura 3. Vista detallada Acerca de.

**3.1. Arquitectura de la Aplicación:** Es una aplicación web la cual emplea un estilo arquitectónico de división en capas. Para efectos de este proyecto se definió una arquitectura de tres (3) capas. Capa de presentación, lógica de negocio y persistencia de datos. Este es un estilo arquitectónico que permite que cada capa se ejecute sobre su propia

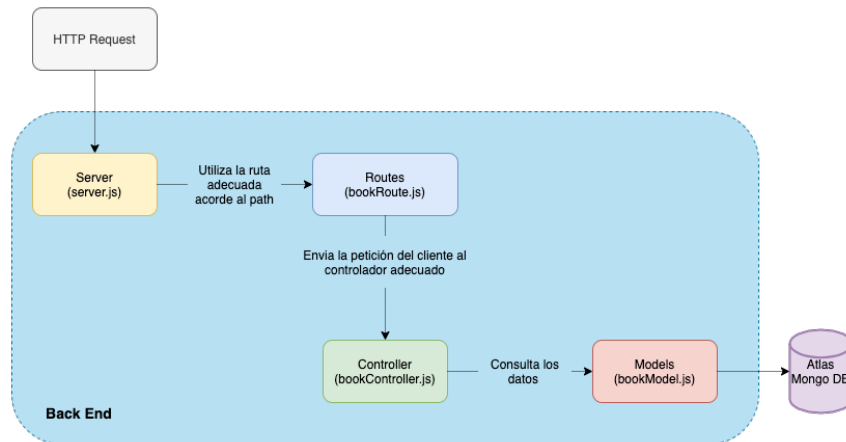
infraestructura. De igual forma, es posible que un equipo de desarrollo se concentre en cada capa.



A continuación se ofrece una descripción de la aplicación para cada una de las capas

- **Descripción de capa de presentación (Front End):** Esta capa se encarga de todos los aspectos relacionados con la interfaz de usuario. A esta capa se accede a través del browser. Igualmente, se encarga de la comunicación con el back end, en este caso, utilizando una API REST. Particularmente en este laboratorio, la capa de presentación o front end se encuentra desarrollado utilizando React. Ésta se define como una librería de java script para desarrollar la capa de presentación. En este sentido, la principal función de react es renderizar código html en el browser. Para efectos del manejo de estilos (CSS), se empleo react-bootstrap
- **Descripción del Back End:** En este estilo arquitectónico, esta es la capa de la mitad en la cual se ejecuta y lleva a cabo la lógica de negocio. El back end se encuentra desarrollado empleando tecnología de scripting del lado del servidor, para este caso node.js. Igualmente utilizamos un framework de desarrollo de aplicaciones web denominado express así como mongoose para la conexión con la bases de datos.
- **Descripción de la persistencia de datos:** En esta capa es donde la información es gestionada y almacenada. En este laboratorio, los datos se almacenan en un motor de bases de datos orientado a documentos, para efectos de esta aplicación Mongo DB. Particularmente, estaremos utilizando Mongo DB Atlas, el cual es un servicio de bases de datos tipo NO SQL como servicio (DBaaS) en la nube.

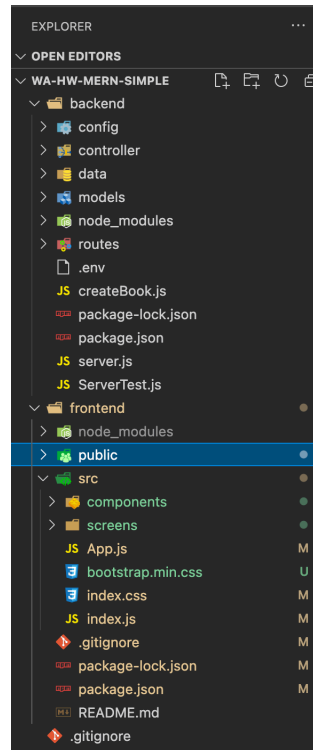
En el siguiente diagrama se puede observar el flujo principal que se requieren implementar para el manejo de las peticiones http.



El módulo “server.js” inicializa el proceso servidor en el puerto 5000, establece la conexión con la base de datos y, una vez recibe las peticiones http, selecciona y usa el módulo de ruta (bookRoute.js) adecuado para el procesamiento de la petición http entrante. Aquí, se envía la información al controlador apropiado (“bookControler”) para que este obtenga los datos que se solicitan desde el front end a través del model (“bookModels.js”). En este caso lo que se envía de respuesta es un Java Script Object Notation (JSON) hacia el front end. El modelo se encarga de obtener los datos solicitados de la base de datos.

#### 4. Estructura del Proyecto.

El proyecto se encuentra estructurado en dos grandes carpetas: frontend y backend. En la imagen se puede encontrar la estructura así como los diferentes folders que componen el proyecto.



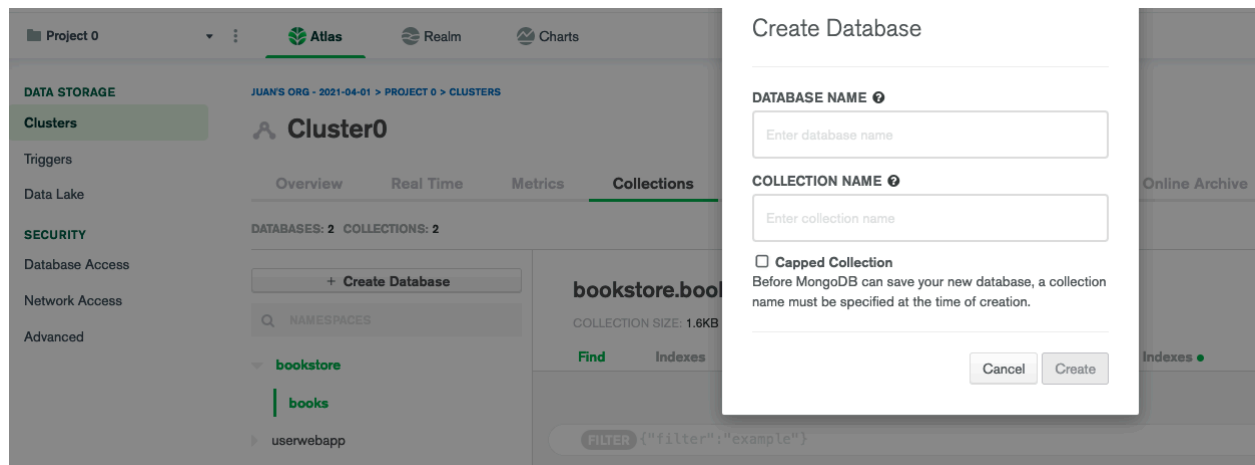
## 5. Configuración de la Base de Datos en ATLAS: Mongo DB.

Para el desarrollo del proyecto, vamos a requerir una bases de datos NO-SQL por documentos como Mongo DB. Para esto en la siguiente URL <https://www.mongodb.com/cloud/atlas> regístrese y cree una cuenta que le permita de manera gratuita desplegar una base de datos.

Una vez registrado en el proyecto y cluster que usted definio, por favor de click en collections para crear una nueva colección:

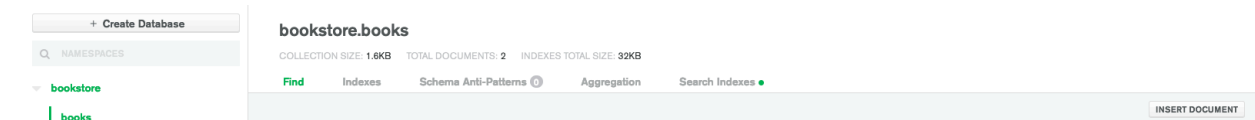


**Ahora de click en Create Database:**

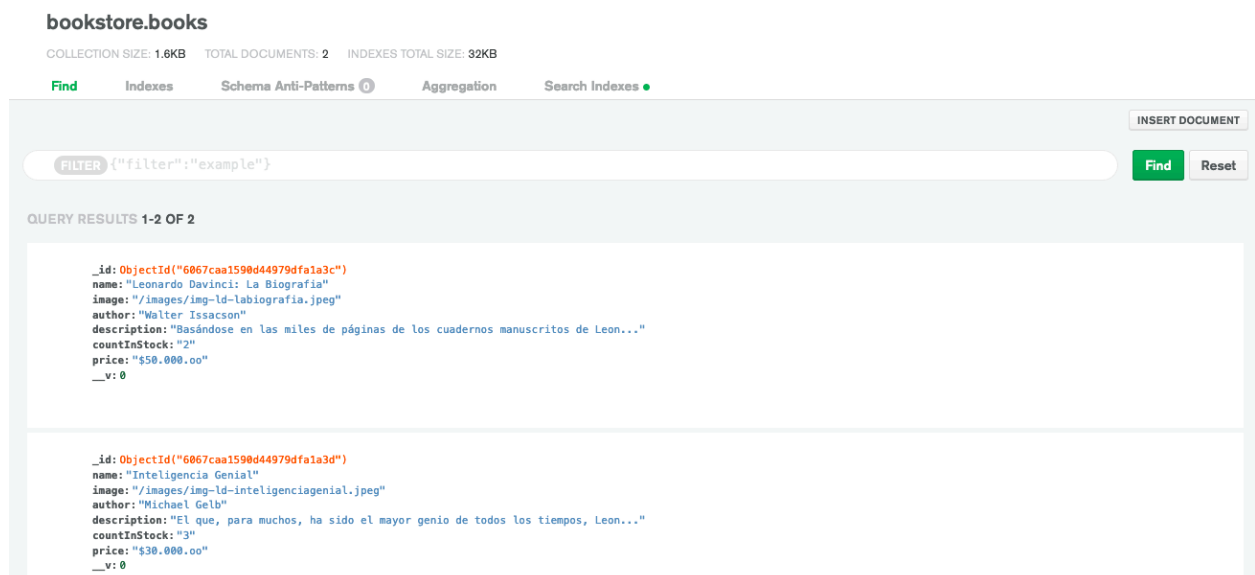


Digite el nombre de la base de datos, para este caso DATABASE NAME = bookstore y COLLECTION NAME = books.

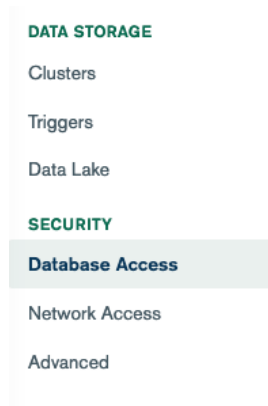
Ahora inserte los documentos, para esto por favor de click en INSERT DOCUMENT, y digite la información.



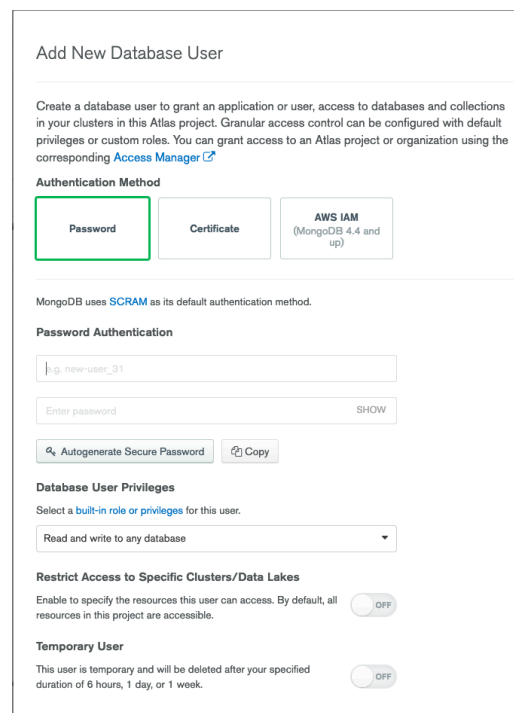
**Nota:** La información a digitar esta en la carpeta backend del proyecto en un directorio que se denomina data, en un archivo books.js. Al final debe obtener la siguiente vista de su base de datos:



Ahora vamos a crear el usuario y password para conectarnos, para esto en la sección de Database Access:



de click en Add New Database User y agregue la información que le solicitan:



Finalmente, vamos a obtener el string de conexión a la base de datos. Para esto vamos y de click en la opción de **clusters**, luego en la opción de **connect**, ahora escoja **connect your application** y copie el string que aparece (deber ser parecido a este)

```
mongodb+srv://<user>:<password>@cluster0.bban6.mongodb.net/myFirstDatabase?retryWrites=true&w=majority
```

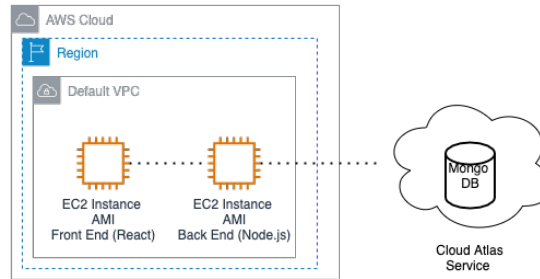
**Nota:** En el archivo .ENV ubicado en la carpeta del back end, agregue su string de conexión a la base de datos.

## 6. Despliegue de la Aplicación Web.

A continuación se presenta el conjunto de pasos para poder desplegar la aplicación web.

### 6.1. Arquitectura de Despliegue.

A continuación en la figura, se puede observar la vista de despliegue para la aplicación en AWS.



Con un poco mas de detalle, en la instancia EC2 seleccionada para desplegar el front end, vamos a utilizar un servidor web nginx para alojar el front end. En este caso vamos a desplegar todo lo relacionado con el front end el directorio raíz que nos permita alojar el contenido html, css e imágenes. Para esto, en el directorio de la código fuente desarrollado, debemos seleccionar la carpeta build que se genero anteriormente. En este máquina instalaremos el servidor web nginx para soportar el front end y utilizarlo como reverse proxy con el fin de podernos comunicar con el back end.

### 6.2. Instancie dos (2) máquinas EC2 en la consola de AWS.

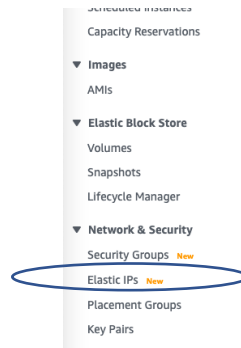
Por favor siga los pasos e instrucciones de la guía de laboratorio número dos para realizar este paso.

### 6.3. Asociando una dirección IP pública elástica:

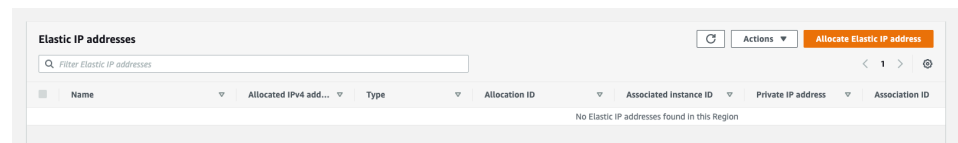
Para efectos del despliegue de la aplicación, para cada una de las instancias desplegadas vamos a requerir direcciones IP públicas. Recuerde que una dirección IP pública es aquella que es valida en el contexto de Internet. Para efectos de este laboratorio vamos a solicitar un tipo de dirección IP pública en AWS que se denominan direcciones IP elásticas. Una característica de estas direcciones IP es que son estáticas (por esto se quiere decir que no cambian en el tiempo). Esto es importante para el despliegue de una aplicación, dado que si la máquina se cae y se reinicia, se mantiene asociada la misma dirección IP pública, lo cual no afectará la prestación del servicio. Para asociar una dirección IP elástica, siga los siguientes pasos que se describen a continuación:

En el menú de “Red y Seguridad (Network and Security)”, seleccione la opción de IP elásticas ”

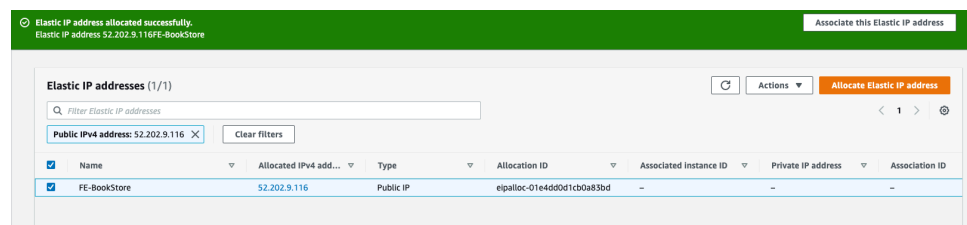




Ahora, escoja la opción de “Allocate Elastic IP Address”.



Una vez se haya asignado la dirección la escogemos y la asociamos a la instancia que designemos para el front end.



Ahora asocie la IP elástica a la máquina que desee (bien sea front end y back end).

### Associate Elastic IP address

Choose the instance or network interface to associate to this Elastic IP address (52.202.9.116)

**Elastic IP address: 52.202.9.116**

**Resource type**  
Choose the type of resource with which to associate the Elastic IP address.

☒ Instance  
☐ Network interface

**Instance**

I-0732fdc40d63b0cf0 (Backend) - running

I-0d12c08db1b1b1d67 (Front end) - running

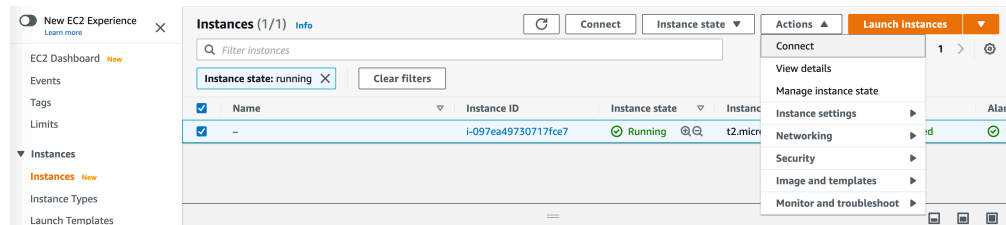
**Reassociation**  
Specify whether the Elastic IP address can be reassociated with a different resource if it already associated with a resource.

☐ Allow this Elastic IP address to be reassociated

Cancel Associate

## 6.4. Sesión con la EC2 instanciada via SSH.

- En el menú del servicio EC2, seleccione la instancia con la cual desea establecer la conexión.



- Cuando seleccione la opción de conectarse, le aparecerá un cuadro parecido. Tenga presente que debe cambiar los permisos de el archivo key (.pem).

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is ST0263-TET.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
❏ chmod 400 ST0263-TET.pem
```

4. Connect to your instance using its Public DNS:

```
❏ ec2-3-88-84-127.compute-1.amazonaws.com
```

Example:

```
❏ ssh -i "ST0263-TET.pem" ubuntu@ec2-3-88-84-127.compute-1.amazonaws.com
```

- En una terminal de consola desde su máquina, inicie una sesión ssh contra el servidor que configuro tal como lo indica la sesión de “example” en la sesión anterior.
- Como resultado debe haber podido establecer la sesión con la instancia EC2.

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-94-206:~$
```

## 6.5. Desplegando el Front End.

Para cada una de las instancias que se desplegaron, actualicemos las aplicaciones que tenemos instaladas en la máquina:

```
$ sudo yum update
```

## 6.6. Instalar el servidor web Nginx en el Front End.

Para efectos de este laboratorio, vamos a utilizar un servidor web como Nginx, con el fin de soportar el despliegue de nuestro front end. Por favor, digite el siguiente comando para proceder a instalar el servidor Nginx.

```
$ sudo amazon-linux-extras install nginx1.12
```

Una vez está instalado vamos a proceder a configurar el servidor. Para esto debemos modificar el archivo de configuración de nginx ubicado en /etc/nginx

```
$ sudo nano /etc/nginx/nginx.conf
```

```
server {
    listen      80 default_server;
    listen      [::]:80 default_server;
    server_name _;
    root        /usr/share/nginx/html/bookstore;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
        location = /404.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
```

### 6.6.1. Instalando la aplicación.

Ahora se va a crear el directorio bookstore. Tenga presente que para poder realizar la copia por favor cambie los permisos en el directorio destino:

```
$ chmod 777 /usr/share/nginx/html/bookstore
```

Descargue y descomprima el código de la aplicación en su estación de trabajo (igual lo puede hacer en la máquina server) que se encuentra en EAFIT Interactiva. Va a observar que se crea la siguiente estructura de directorios:

```
README.md  backend  frontend
```

Proceda a instalar node.js en la máquina de la siguiente forma:

```
$ sudo yum install -y gcc-c++ make
$ curl -sL https://rpm.nodesource.com/setup_14.x | sudo -E bash -
```

Ahora si instalemos la versión de node.js:

```
sudo yum install -y nodejs
```

Verifiquemos la versión:

```
$ node -v  
$ npm -v
```

Ahora se requiere que instale las dependencias, para esto ubicado en el directorio frontend ejecute el comando:

```
$ npm install
```

Va a observar que se crea la carpeta node\_modules y queda la estructura de la siguiente forma:

```
README.md  build  node_modules  package-lock.json  package.json  public  src
```

Para efectos de este laboratorio, nos interesa generar el código compilado de producción para la aplicación desarrollada. Para esto, realizaremos lo siguiente:

```
$cd frontend
```

Dado que la aplicación esta construida en React, se requiere crear los diferentes archivos para el entorno de producción. Para esto se requiere que ejecute el comando:

```
$ npm run build
```

Posterior a la ejecución de este comando se creará la carpeta build, la cual contiene el código html, css, imágenes que se deben publicar en nuestro servidor web.

```
README.md  build  node_modules  package-lock.json  package.json  public  src
```

Copie la carpeta build desde su máquina (o desde el lugar donde la tiene) hasta el directorio bookstore que se creo en la instancia de EC2. En caso de que este en una máquina remota, procedamos a realizar una copia segura a través del siguiente comando:

```
$ scp -i xxx.pem -r /path/build/* ec2-user@ec2-X-X-X-X.compute-  
1.amazonaws.com:/usr/share/nginx/html/bookstore
```

**Nota:**

- XXX.pem es el archivo que usted genero para conectarse a la máquina.
- /path equivale a la ruta donde usted tiene ubicado la carpeta backend del proyecto.
- [ec2-user@ec2-X-X-X-X.compute-1.amazonaws.com](https://ec2-user@ec2-X-X-X-X.compute-1.amazonaws.com)
- :/home/ec2-user : la ruta donde va a copiar los archivos en el servidor destino.

Cambie los permisos al directorio

```
$ chmod 777 /usr/share/nginx/html/bookstore/
```

Ahora, se procede a verificar el estado del servicio web:

```
$ sudo systemctl status nginx.service
```

Si el servicio esta caído, por favor suba el servicio.

```
$ sudo systemctl start nginx.service
```

```
[ec2-user@ip-172-31-42-150 bookstore]$ sudo systemctl status nginx.service
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2021-04-08 11:14:10 UTC; 16s ago
     Process: 4156 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
     Process: 4153 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 4152 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Main PID: 4159 (nginx)
      CGroup: /system.slice/nginx.service
              └─4159 nginx: master process /usr/sbin/nginx
                  └─4160 nginx: worker process

Apr 08 11:14:10 ip-172-31-42-150.ec2.internal systemd[1]: Starting The nginx HTTP and reverse proxy server...
Apr 08 11:14:10 ip-172-31-42-150.ec2.internal nginx[4153]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Apr 08 11:14:10 ip-172-31-42-150.ec2.internal nginx[4153]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Apr 08 11:14:10 ip-172-31-42-150.ec2.internal systemd[1]: Failed to read PID from file /run/nginx.pid: Invalid argument
Apr 08 11:14:10 ip-172-31-42-150.ec2.internal systemd[1]: Started The nginx HTTP and reverse proxy server.
```

```
sudo chkconfig nginx on
```

## 6.7. Desplegando el back end.

Antes que todo debemos instalar la versión de node. Para esto configuremos el repositorio.

```
$ sudo yum install -y gcc-c++ make
$ curl -sL https://rpm.nodesource.com/setup_14.x | sudo -E bash -
```

Ahora si instalemos la versión de node.js:

```
sudo yum install -y nodejs
```

Verifiquemos la versión:

```
$ node -v
$ npm -v
```

Creemos un pequeño servidor web para verificar funcionalmente que todo este correcto, para esto cree un archivo en el directorio raíz de la aplicación

```
$ sudo nano TestServer.js
```

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
```

```
res.end('Welcome Node.js');
}).listen(3000, "127.0.0.1");
console.log('Server running at http://127.0.0.1:3000/');
```

Para ejecutar este código digite el comando:

```
$ node TestServer.js
```

```
[ec2-user@ip-172-31-62-153 BookStore]$ node TestServer.js
Server running at http://127.0.0.1:3000/
```

Se procede a realizar la copia de los archivos:

```
scp -i XXXX.pem -r /path/backend/ ec2-user@ec2X-X-X-X.compute-
1.amazonaws.com:/home/ec2-user
```

**Nota:**

- XXXX.pem es el archivo que usted genero para conectarse a la máquina.
- /path equivale a la ruta donde usted tiene ubicado la carpeta backend del proyecto.
- [ec2-user@ec2X-X-X-X.compute-1.amazonaws.com](https://ec2-user@ec2X-X-X-X.compute-1.amazonaws.com)
- :/home/ec2-user : la ruta donde va a copiar los archivos en el servidor destino.

Ahora entramos al directorio backend y ejecutamos el comando;

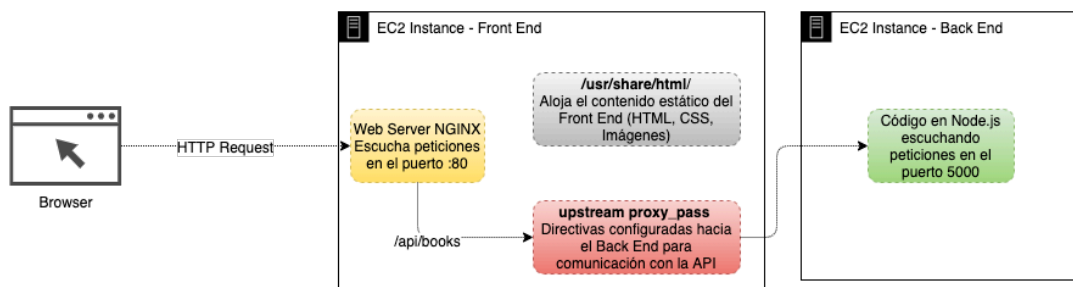
```
$ npm install
```

Esto permitirá la instalación de los módulos

En este punto debemos permitir el acceso a la bases de datos a nuestro servidor de bases de datos en la nube

```
$ node server.js
```

Ahora configuramos el web server (nginx) para que reciba las peticiones dirigidas a la api: /api/books y las redirecciones al backend.



Para esto agregamos la directiva upstream antes de la de server en el archivo nginx.conf

```
$ sudo nano /etc/nginx/nginx.conf
```

```
upstream backend{
    server 172.31.50.214:5000;
}
```

Luego modificamos la directiva server y, dentro de esta, agregamos lo siguiente:

```
location /api/books {
    proxy_pass http://backend;
}
```

Al final, el archivo queda con el siguiente aspecto:

```
include /etc/nginx/conf.d/*.conf;

upstream backend{
    server 172.31.84.76:5000;
}

server {
    listen      80 default_server;
    listen      [::]:80 default_server;
    server_name _;
    root        /usr/share/nginx/html/bookstore;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }

    location /api/books{
        proxy_pass http://backend;
    }
}
```

## 6.8. Verificación de la aplicación:

En un browser seleccione coloque la dirección IP elástica del front end y debe visualizar la página!

