

Especificación de Requerimientos

Asociación de agricultores

Plaza Campesina

La Plaza

Tabla de Contenido

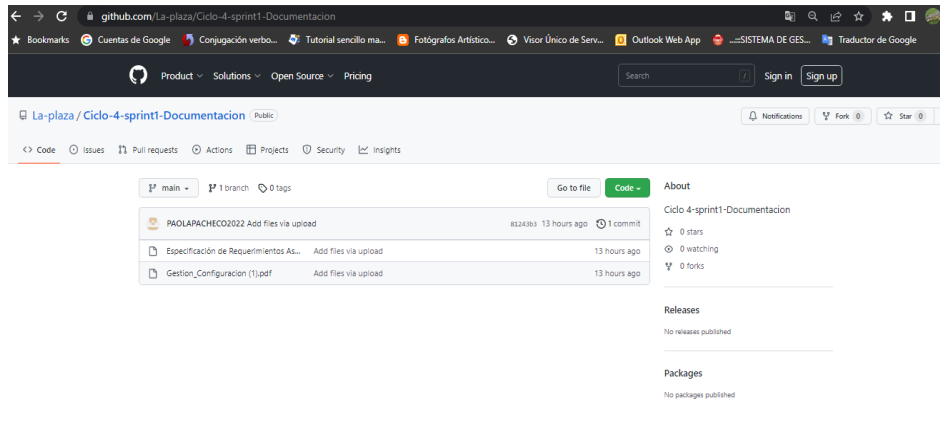
1.	Tabla 1. Nombres, correo electrónico y rol del equipo de trabajo SCRUM.	3
2.	Repositorio GitHub.....	4
3.	Gestión de configuración	4
1.	Product Backlog.	6
2.	Tabla 2. Especificaciones del proyecto SCRUM.	7
3.	Tabla 3. Especificaciones de historias de usuario.	8
4.	Proceso de comercialización:	11
5.	Definición caso de uso:.....	11
6.	Diagrama de secuencia-ejemplo-acceso y petición administrador:	12
7.	Desarrollo frontend.....	1
8.	Desarrollo backend	4
9.	Video: Reunión Sprint review.....	20
10.	Informe de Retrospectiva.	21
11.	Historias a trabajar en el siguiente Sprint.....	21

1. Tabla 1. Nombres, correo electrónico y rol del equipo de trabajo SCRUM.

Nombres y Apellidos	Correo electrónico	Rol
Dora Paola Pacheco Moreno	paolapacheco.moreno@gmail.com	Gestor de Proyecto
Sayda Yamile Cagua Carrillo	Jose206@utp.edu.co	Gestor Base de Datos
Isis Nirvana Segura Valero	isiissegura@gmail.com	Desarrollador Frontend
Julián Andrés SeguraGonzález	julian888s@gmail.com	Desarrollador Backend
José Daniel RamírezSaldaña	Jose206@utp.edu.co	Tester

2. Repositorio GitHub.

<https://github.com/La-plaza/Ciclo-4-sprint1-Documentacion>



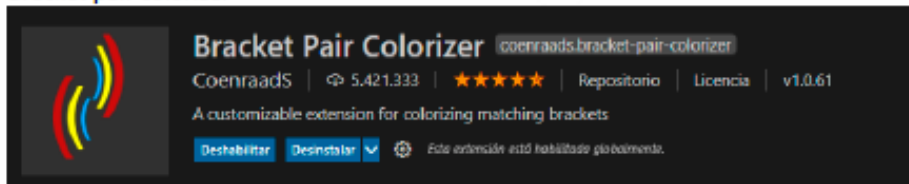
3. Gestión de configuración

Instalación del editor de código

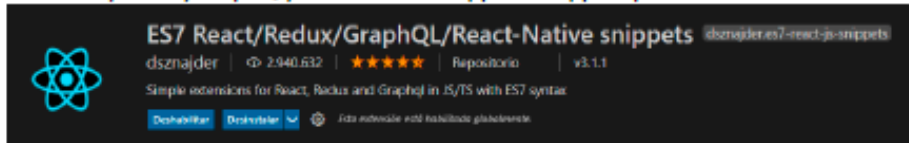
Selección de editor de código: Visual Studio Code

Instalación de extensiones:

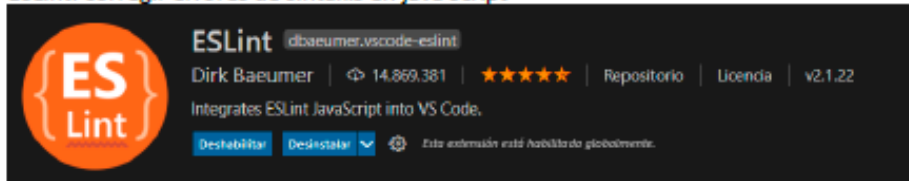
Bracket pair colorizer



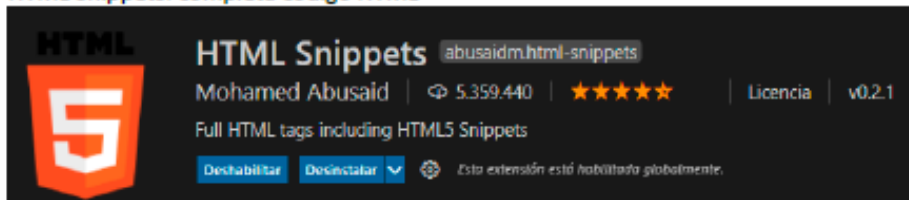
ES7 React/Redux/GraphQL/React-Native snippets: snippets para react



ESLint: corregir errores de sintaxis en java script



HTML Snippets: completa código HTML




Intellisense for CSS class names in HTML: autocompletar o auto rellenar las clases CSS


Intellisense for CSS class names in HTML: autocompletar o auto rellenar las clases CSS

IntelliSense for CSS class names in HTML zignd.html-css-class-completion
 Zignd | 3.459.309 | ★★★★★ | Repositorio | Licencia | v1.20.0
 CSS class name completion for the HTML class attribute based on the definitions found in your workspace.
 Deshabilitar | Desinstalar | Esta extensión está habilitada globalmente.


Reactjs code snippets: Disparadores de código React JS

 **Reactjs code snippets** xabikos.reactsnippets
 charalampos karypidis | 883.330 | ★★★★★ | Repositorio | v2.4.0
 Code snippets for Reactjs development in ES6 syntax
 Instalar


Vscode-pdf: Lector de PDF

 **vscode-pdf** tomoki1207.pdf
 tomoki1207 | 803.861 | ★★★★★ | Repositorio | Licencia | v1.1.0
 Display pdf file in VSCode.
 Instalar

Convertidor de Markdown a PDF

 **Markdown PDF** yzane.markdown-pdf
 yzane | 804.300 | ★★★★★ | Repositorio | Licencia | v1.4.4
 Convert Markdown to PDF
 Instalar

Creador de Markdown

 **Markdown All in One** yzhang.markdown-all-in-one
 Yu Zhang | 2.686.433 | ★★★★★ | Repositorio | Licencia | v3.4.0
 All you need to write Markdown (keyboard shortcuts, table of contents, auto preview and more)
 Instalar

Configuración



Instalación de Postman



POSTMAN



Extensión de Google Chrome: React Developer Tools

chrome web store

Inicio > Extensiones > React Developer Tools



React Developer Tools

Ofrecido por: Facebook

★★★★☆ 1.330

Herramientas para desarrolladores

2.000.000+ usuarios

Desinstalar



React Developer Tools se ha añadido a
Chrome

Haz clic en este icono para utilizar esta extensión.

Para gestionar tus extensiones, haz clic en la opción
Extensiones del menú Herramientas.

1. Product Backlog.

<https://trello.com/b/vZ7CH2SI/sprint-3>

2. Tabla 2. Especificaciones del proyecto SCRUM.

CATEGORÍA	Agro
NOMBRE	Plaza campesina (Asociación de Agricultores)
DESCRIPCIÓN	Las asociaciones agrícolas desempeñan un papel importante para apoyar a los pequeños productores; hombres y mujeres, y grupos marginados. Ofreciendo oportunidades de mercado a servicios como una mejor gestión. La asociatividad es un mecanismo de cooperación entre campesinos que trabajan por un bien común, y permite disminuir costos, acceder a tecnología de punta, acrecentar el poder de negociación y dar estabilidad a los precios de ventas, lo que se ve reflejado en un aumento de la rentabilidad.
OBJETIVO ESTRATÉGICO	Crear un software como apoyo a pequeños productores, con el fin de establecer los precios unitarios de sus productos agrícolas, mejorar su margen de venta y rentabilidad.
PÚBLICO OBJETIVO	Productores y/o campesinos
IMPACTO ESPERADO	Mayores ingresos de los campesinos.

A continuación, se presentan las cartas de información que complementan las historias de usuarios del proyecto: Conexión agro con empresas:

3. Tabla 3. Especificaciones de historias de usuario.

Tabla 3. Especificaciones de historias de usuario.

Historias de Usuario	
Número: 01	Nombre: REGISTRAR USUARIO
Puntos estimados:	
Descripción: Yo como productor líder requiero registrarme en la aplicación, con la finalidad de poder ingresar mis datos al sistema.	
Criterios de Aceptación: Cuando se ingresa un usuario y/o contraseña no válida, el sistema no debe permitir el ingreso y se presentará un error con mensaje: "usuario y/o contraseña incorrecta, por favor ingresar caracteres válidos (Solo se aceptan números y letras)". Los datos para este registro son: Nombres y apellidos y/o razón social, Identificación, Ciudad/Municipio, Dirección, Email.	

Historias de Usuario	
Número: 02	Nombre: INICIAR SESIÓN
Puntos estimados:	
Descripción: Yo como productor líder requiero ingresar al sistema, para poder hacer uso del sistema.	
Criterios de Aceptación: Cuando el ingreso del usuario y contraseña son incorrectos, entonces el sistema NO permitirá el ingreso y el sistema presentará una alerta con el siguiente mensaje: "Usuario y/o contraseña no válida (Sólo se aceptan números y letras)".	

Historias de usuario	
Número: 04	Nombre: CAMBIAR CONTRASEÑA
Puntos estimados:	
Descripción: Yo como productor líder, deseo cambiar la contraseña, para poder ingresar al sistema.	
Criterios de Aceptación: Si el usuario digita un carácter incorrecto, debe aparecer un mensaje de error "carácter no valido, por favor ingrese una contraseña válida" Solo se aceptan números y letras.	

Historias de usuario	
Número: 04	Nombre: CREAR PRODUCTO(Crear)
Puntos Estimados:	
Descripción: Como líder de productores, necesito crear un producto, con la finalidad de estandarizar los valores de producto y los productores que poseen el producto en oferta.	
Criterios de aceptación: El producto debe quedar registrado con un nombre real y único, ya que podrá ser más fácil su ubicación por otros usuarios. Para la creación de un producto se debe tener en cuenta el, Nombre del producto, Descripción del producto (Tipo de producto, Calidad/Estado, Tamaño, Cantidad, Precio) y el Productor que tenga la disposición del producto.	

Historias de usuario	
Número: 05	Nombre: LISTAR PRODUCTOS
Puntos Estimados:	
Descripción: Yo como líder productor necesito ver la lista de productos creados en el sistema.	
Criterios de aceptación: El Líder productor puede validar la lista de productos añadidos, así mismo validar la información agregada, a su vez permite la actualización y eliminación de cada producto.	

Historias de usuario	
Número: 06	Nombre: ACTUALIZAR PRODUCTO (Editar)
Puntos Estimados:	
Descripción: Yo como líder de los productores, necesito modificar el producto creado, con la finalidad de poder realizar algún cambio o actualización en el sistema del mismo.	
Criterios de aceptación:	
El líder de los productores debe actualizar el producto con los requerimientos solicitados por el sistema y de forma clara y concisa, de lo contrario el registro no será válido y/o almacenado en el sistema.	
Para la actualización de un producto se debe tener en cuenta el, Nombre del producto, Descripción del producto (Tipo de producto, Calidad/Estado, Tamaño, Cantidad, Precio) y el Productor que tenga la disposición del producto.	

Historias de usuario	
Número: 07	Nombre: BORRAR PRODUCTO (Eliminar)
Puntos Estimados:	
Descripción: Yo como líder de los productores, necesito eliminar un producto, con la finalidad de poderlo borrar del sistema.	
Criterios de aceptación: El líder de los productores debe tener en claro cuál es el producto que desea eliminar, debe eliminar el producto que no desea seguir ofreciendo y no cuenta con disponibilidad.	

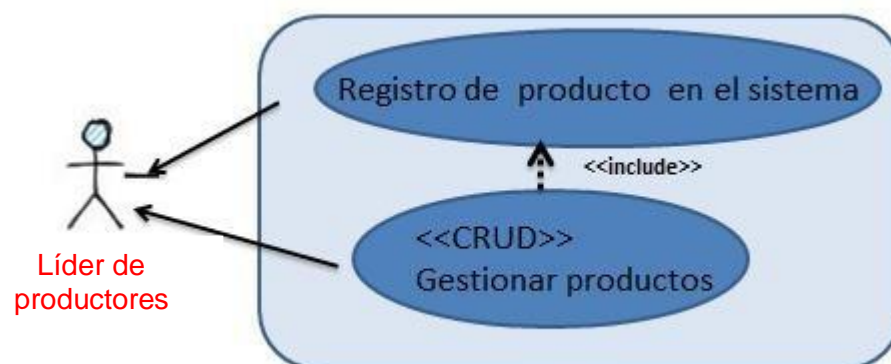
Este software tendrá el siguiente modulo;

- Módulo de productos

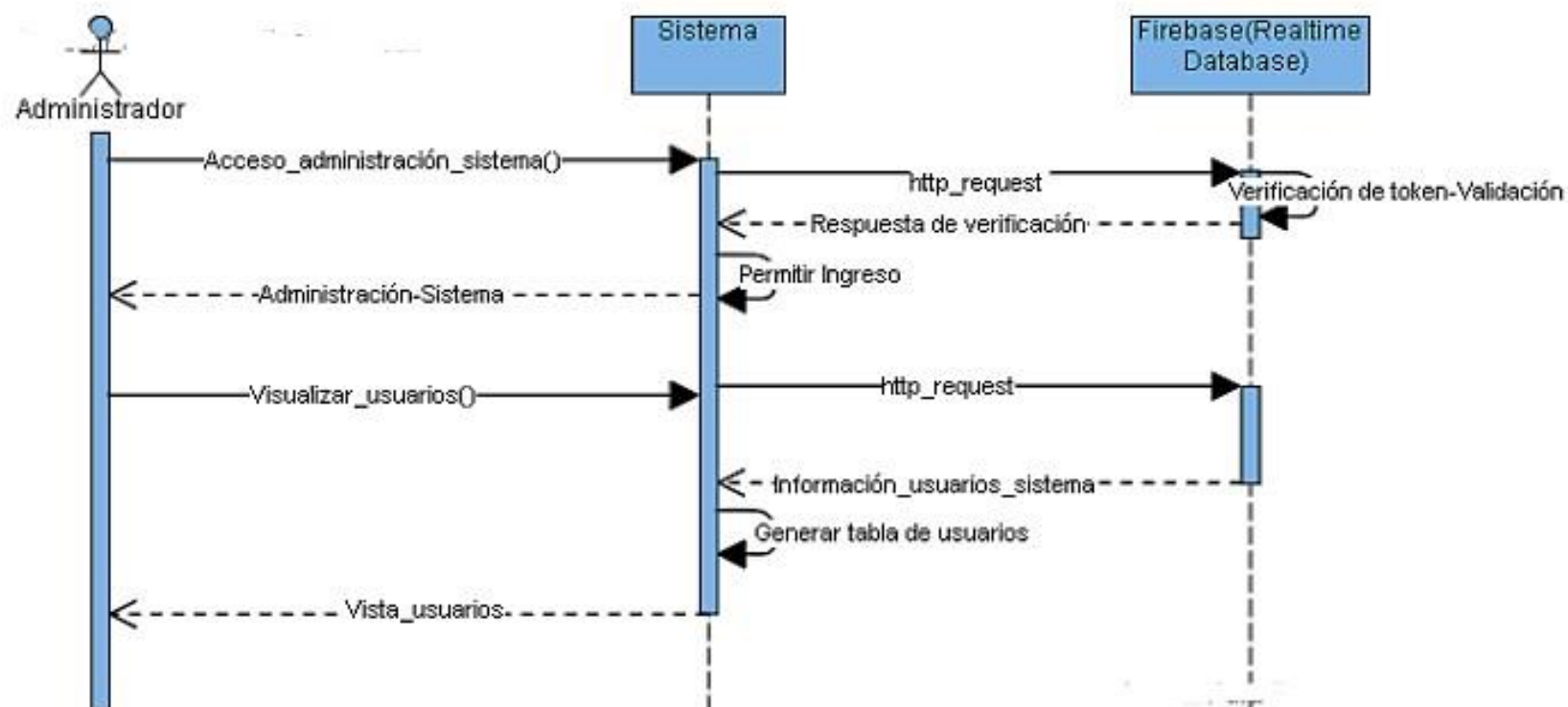
4. Proceso de comercialización:

PRODUTOR —————> PRODUCTO

5. Definición caso de uso:



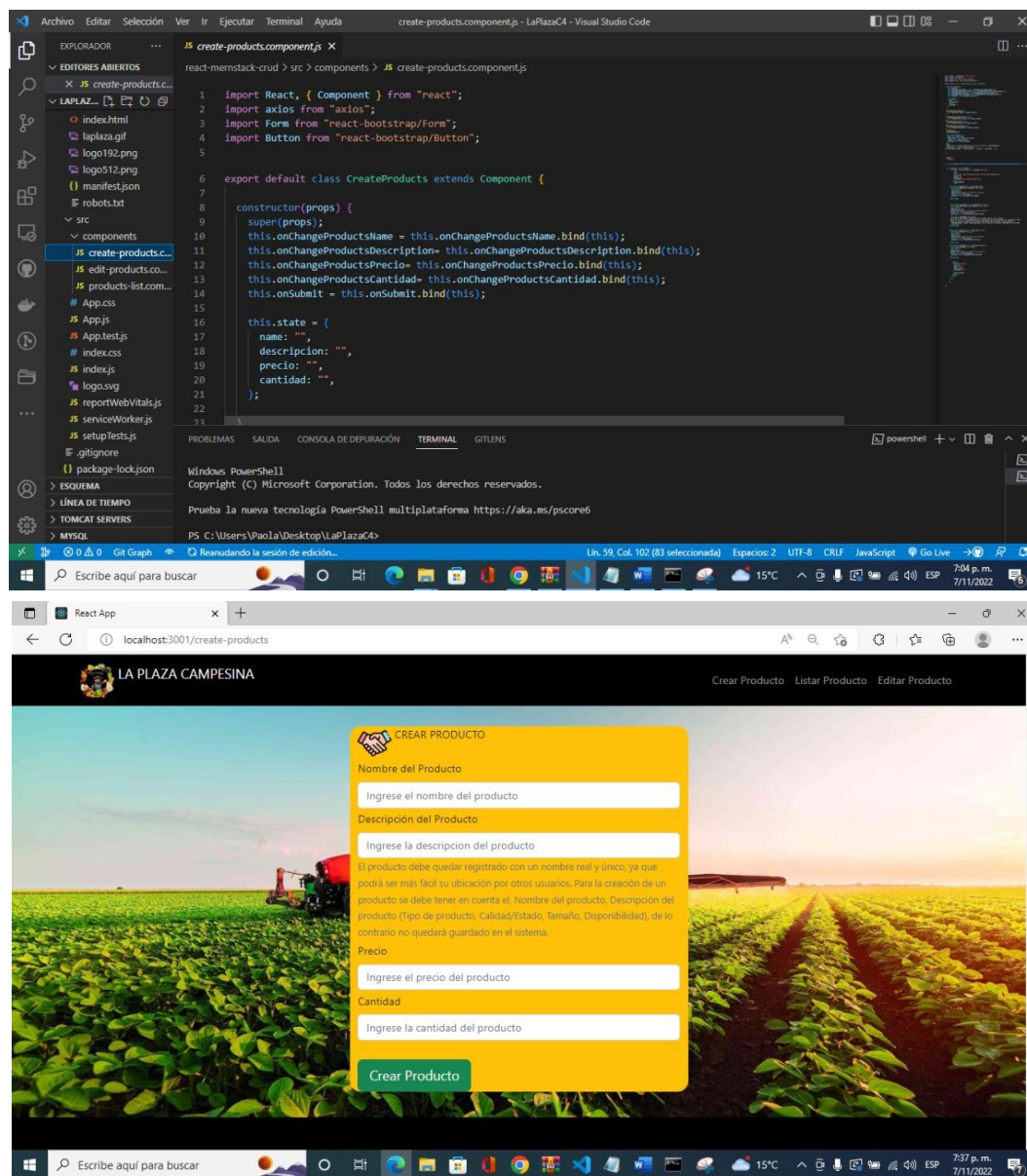
6. Diagrama de secuencia-ejemplo-acceso y petición administrador:

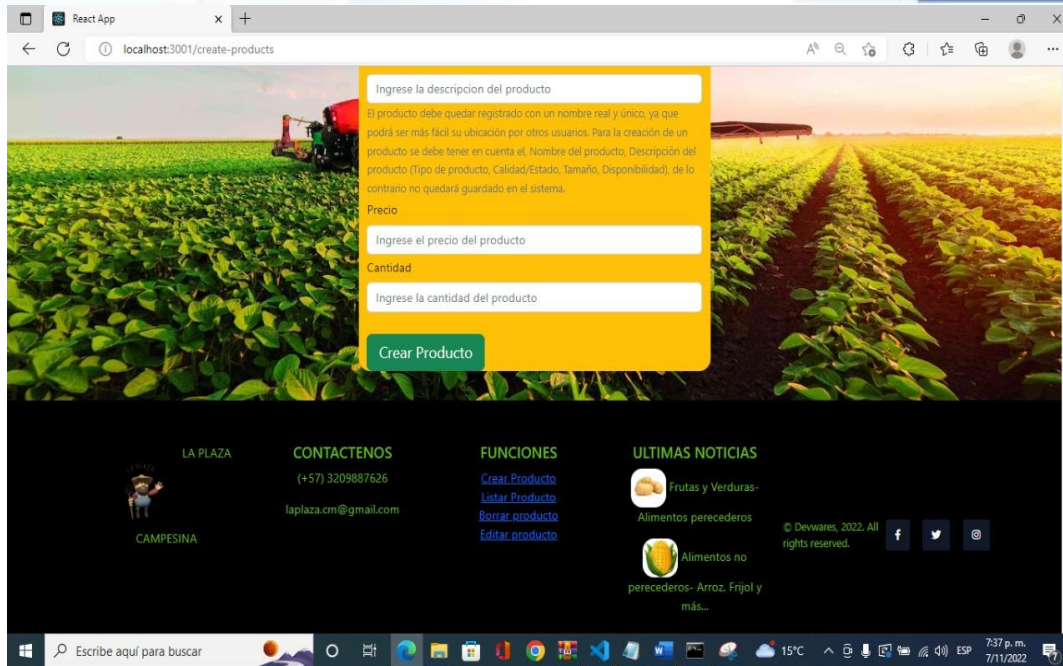


7. Desarrollo frontend

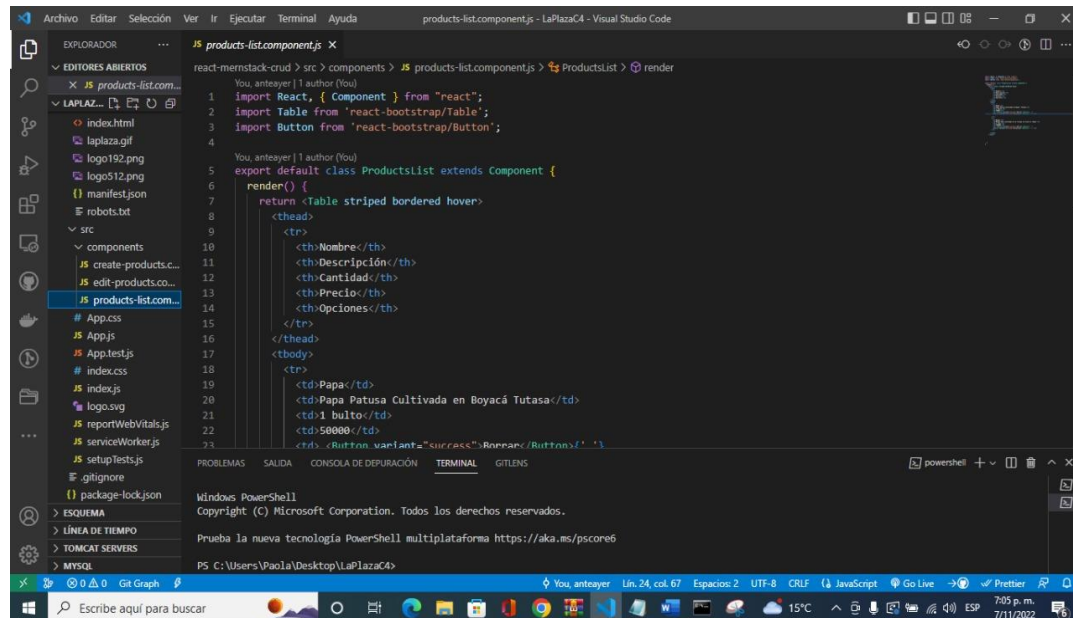
❖ Desarrollo de Componentes

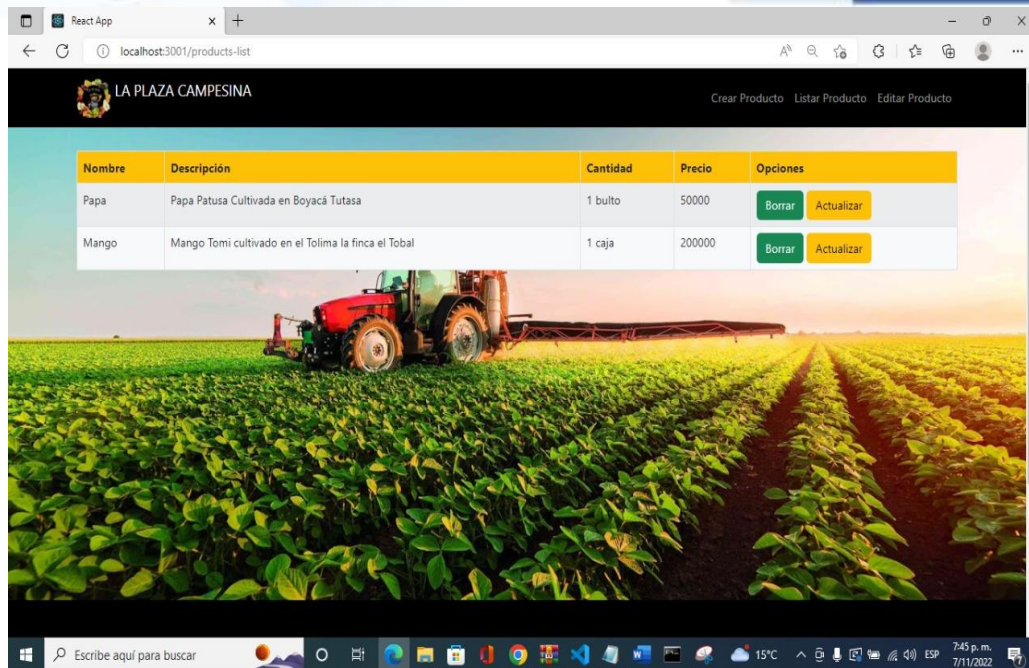
Componente crear producto



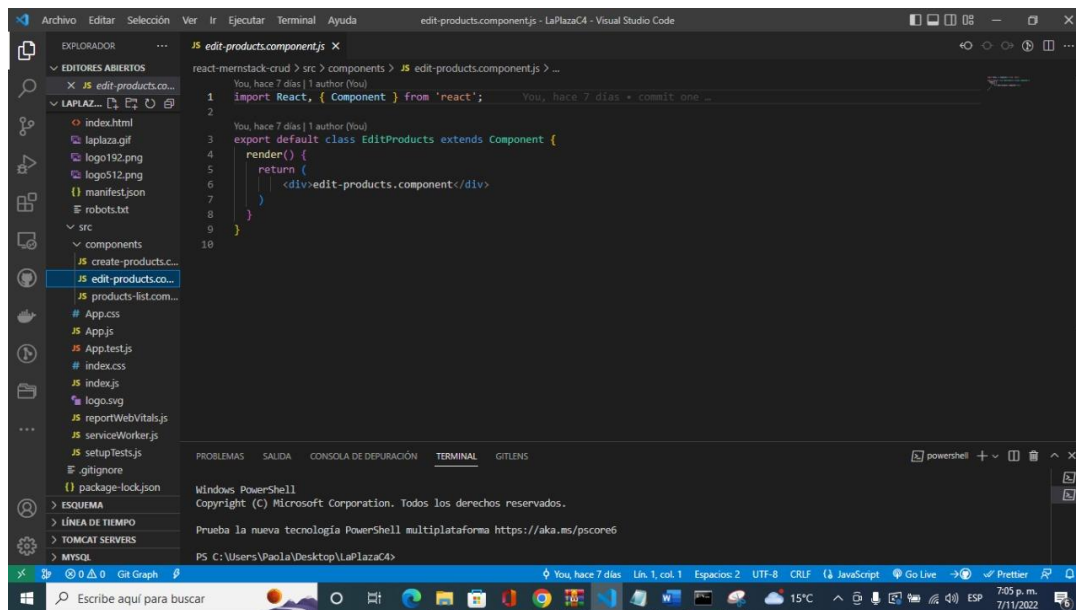


❖ Componente lista de producto





❖ Componente editar productos



❖ Componente app.js

```

1  import React, { useState } from 'react';
2  import ReactDOM from 'react-dom';
3  import { CDBBtn, CDBIcon, CDBBox } from 'cdblreact';
4  import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-dom';
5  import CreateProducts from './components/create-products.component';
6  import EditProducts from './components/edit-products.component';
7  import ProductsList from './components/products-list.component';
8
9  function App() {
10     return (
11       <div className="App">
12         <Router>
13           <header className="App-header">
14             <Navbar bg="black" variant="dark">
15               <Container>
16                 <Navbar.Brand>
17
18                 <Link to="/create-products" className="nav-link">
19                   
26                 </Link>
27               </Navbar.Brand>
28             </header>
29             <Switch>
30               <Route path="/create-products" component={CreateProducts} />
31               <Route path="/edit-products" component={EditProducts} />
32               <Route path="/products-list" component={ProductsList} />
33             </Switch>
34           </Router>
35         </div>
36       </div>
37     );
38   }
39
40   export default App;
  
```

8. Desarrollo backend

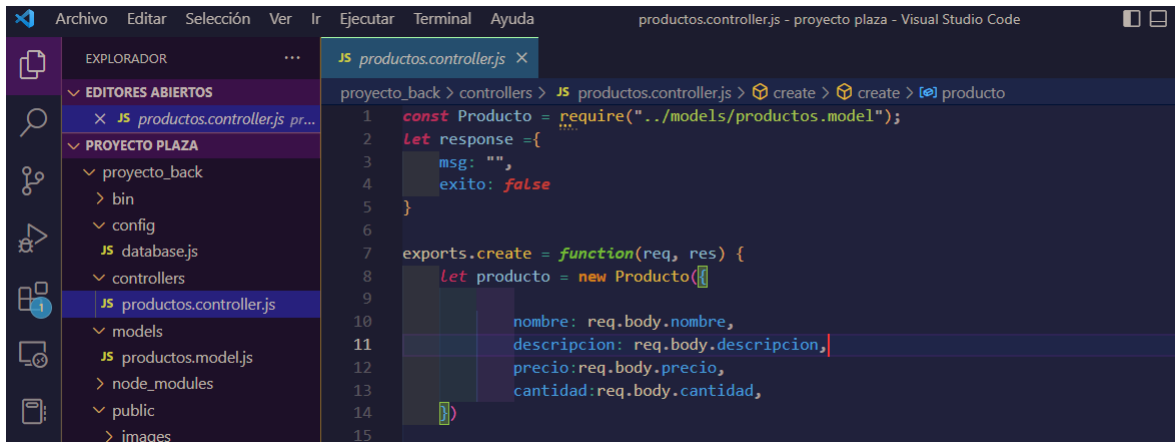
En carpeta config creo el archivo database.js el cual genera la conexión de la base con el host el puerto y nombre de la base de datos.

```

1  const mongoose = require('mongoose');
2
3  const host = "localhost";
4  const port = "27017";
5  const db = "plaza";
6
7  exports.mongooseConnect = () => {
8    const mongoStringConnection = `mongodb://${host}:${port}/${db}`;
9
10    mongoose.connect(mongoStringConnection);
11    mongoose.Promise = global.Promise;
12    const dbConnection = mongoose.connection;
13    dbConnection.on("error", console.error.bind(console, "Mongodb connection error"));
14  }
  
```


En carpeta controllers se genera archivo del crud de productos con las funciones create, save, find, findOne, update y remove.

❖ Crea producto

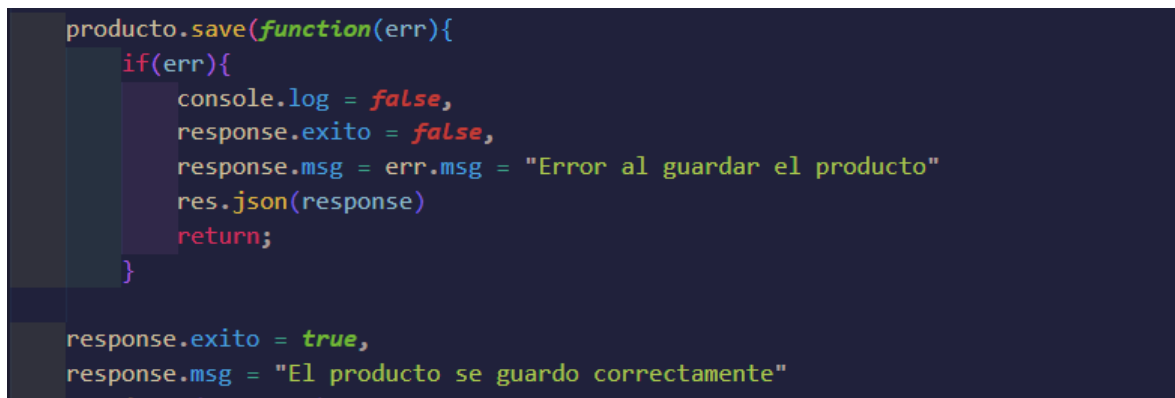


```

1  const Producto = require("../models/productos.model");
2  let response = {
3    msg: "",
4    exito: false
5  }
6
7  exports.create = function(req, res) {
8    let producto = new Producto({
9      nombre: req.body.nombre,
10     descripcion: req.body.descripcion,
11     precio: req.body.precio,
12     cantidad: req.body.cantidad,
13   })
14 }
15

```

❖ Guarda producto



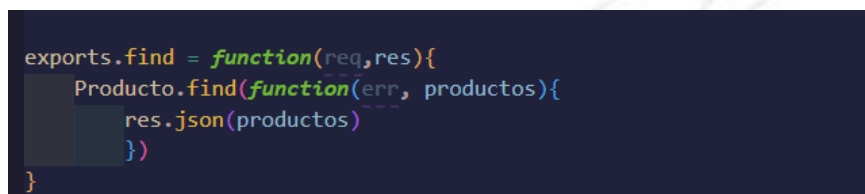
```

producto.save(function(err){
  if(err){
    console.log = false,
    response.exito = false,
    response.msg = err.msg = "Error al guardar el producto"
    res.json(response)
    return;
  }

  response.exito = true,
  response.msg = "El producto se guardo correctamente"
})

```

❖ Buscar productos



```

exports.find = function(req,res){
  Producto.find(function(err, productos){
    res.json(productos)
  })
}

```

❖ Buscar un producto

```

37 exports.findOne = function(req, res){
38   Producto.findOne({_id: req.params.id}, function(err, producto){
39     res.json(producto)
40   })
41 }
42

```

❖ Actualizar producto

```

42 exports.update = function(req,res){
43   let producto = {
44     nombre: req.body.nombre,
45     descripcion: req.body.descripcion,
46     precio: req.body.precio,
47     cantidad: req.body.cantidad,
48   }
49   Producto.findByIdAndUpdate(req.params.id, {$set: producto}, function (err){
50     if(err){
51       console.error(err),
52       response.exito = false,
53       response.msg = "Error al modificar el producto"
54       res.json(response)
55       return;
56     }
57     response.exito = true,
58     response.msg = "El producto se modifico correctamente"
59     res.json(response)
60   })
61 }
62

```

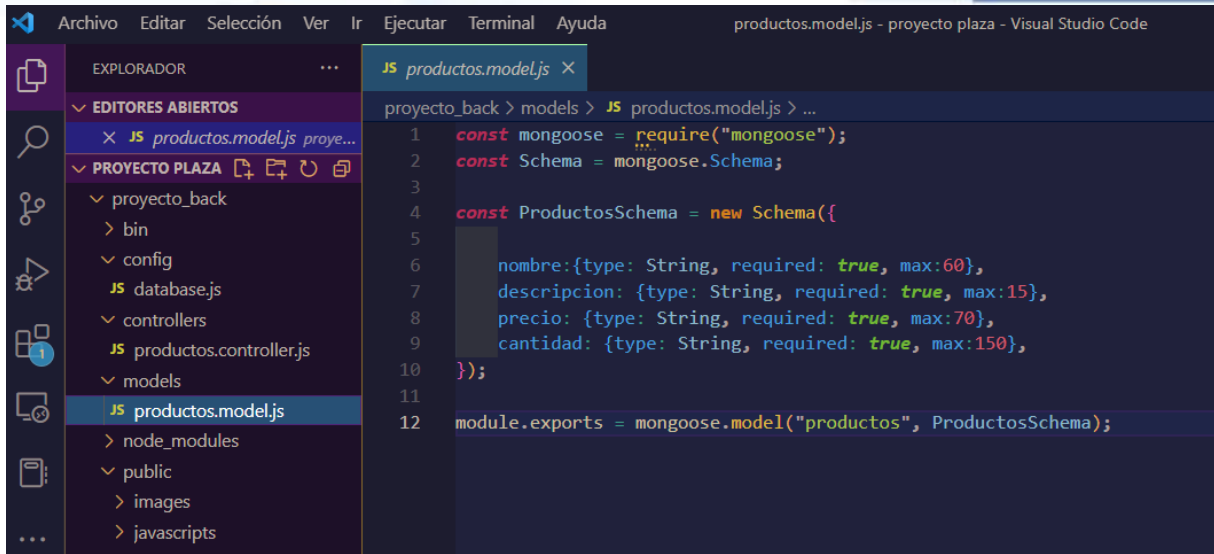
❖ Eliminar producto

```

66 exports.remove = function(req,res){
67   Producto.findByIdAndRemove({_id: req.params.id}, function(err){
68     if(err){
69       console.error(err),
70       response.exito = false,
71       response.msg= "Error al eliminar el producto"
72       return;
73     }
74     response.exito = true,
75     response.msg = "El producto eliminado correctamente"
76     res.json(response)
77   })
78 }
79

```

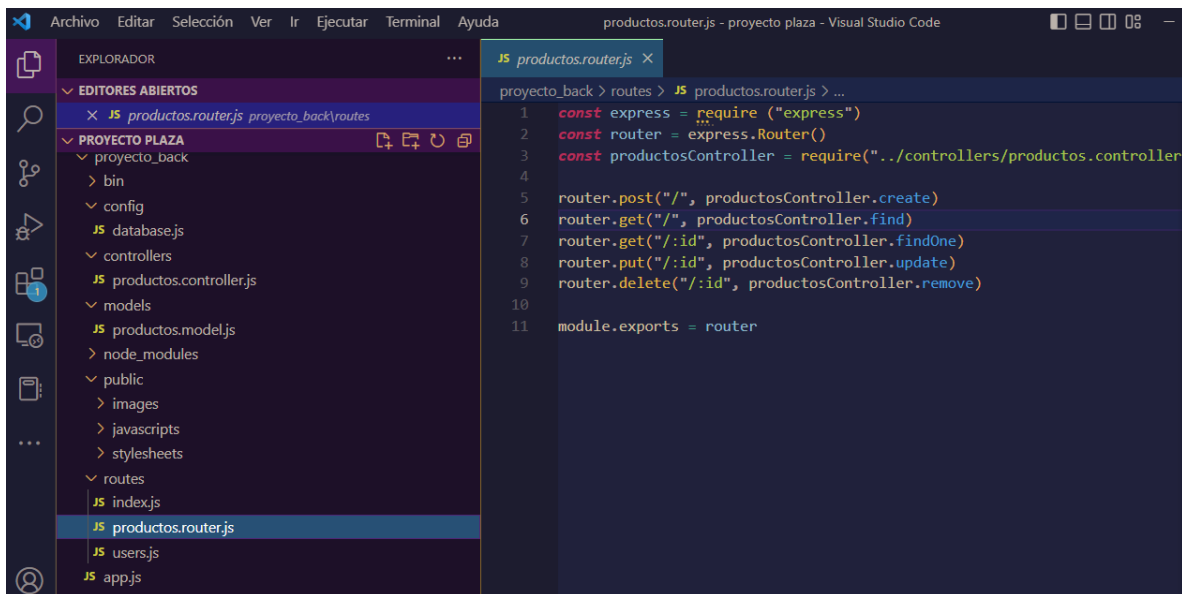
En la carpeta models se crea la cantidad de caracteres por cada atributo que posee la tabla de productos.



```

1  const mongoose = require("mongoose");
2  const Schema = mongoose.Schema;
3
4  const ProductosSchema = new Schema({
5
6    nombre: {type: String, required: true, max:60},
7    descripcion: {type: String, required: true, max:15},
8    precio: {type: String, required: true, max:70},
9    cantidad: {type: String, required: true, max:150},
10  });
11
12  module.exports = mongoose.model("productos", ProductosSchema);
  
```

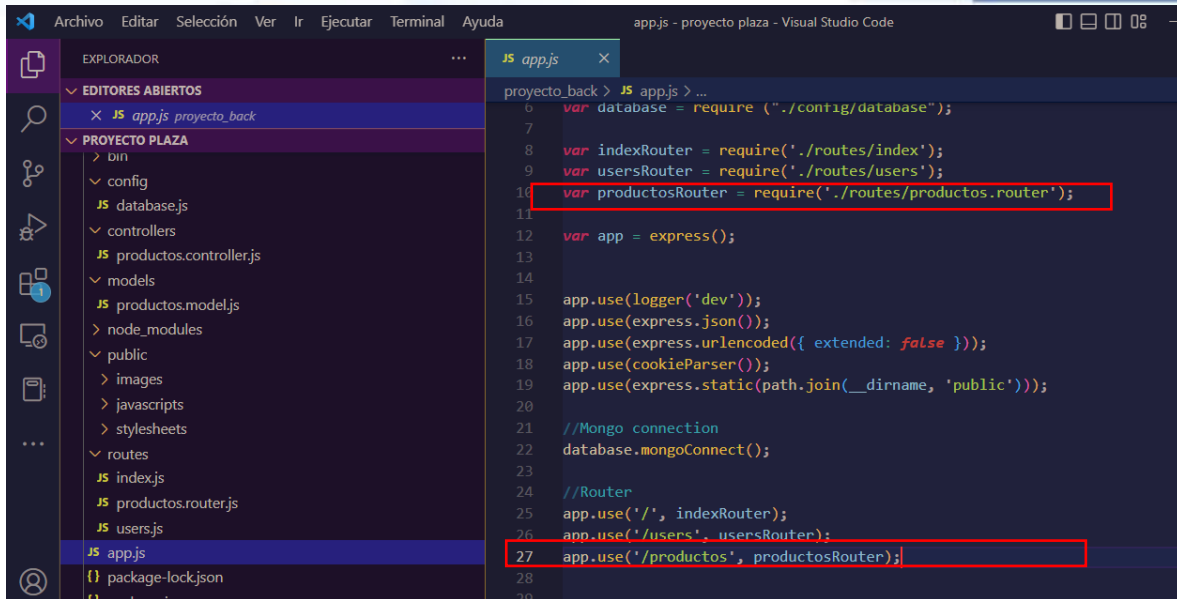
En la carpeta routers se enrutan las órdenes del postman post (crea) get (trae) put(modifica) delete (elimina).



```

1  const express = require ("express")
2  const router = express.Router()
3  const productosController = require("../controllers/productos.controller")
4
5  router.post("/", productosController.create)
6  router.get("/", productosController.find)
7  router.get("/:id", productosController.findOne)
8  router.put("/:id", productosController.update)
9  router.delete("/:id", productosController.remove)
10
11  module.exports = router
  
```

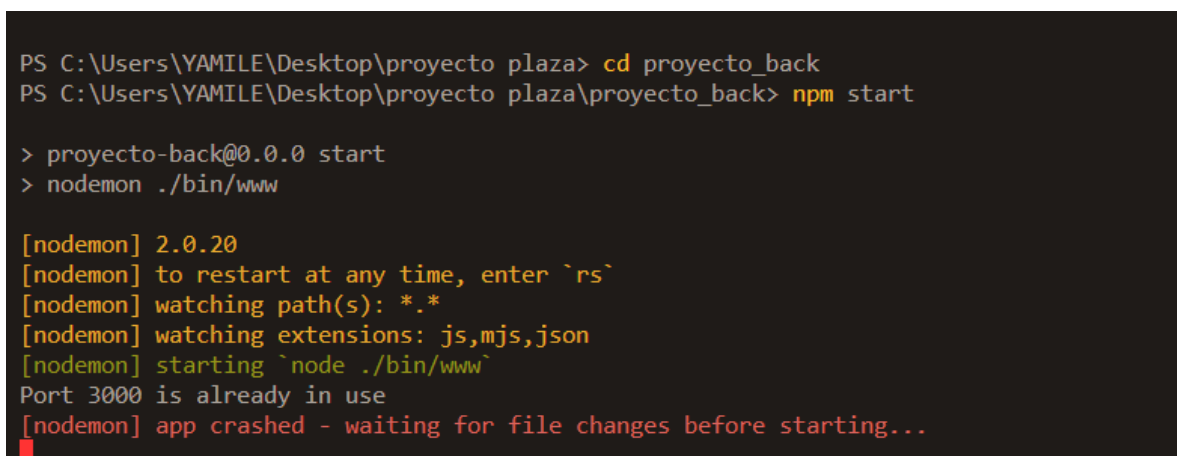
En el archivo principal app.js hacemos el llamado de nuestra variable en este caso línea 10 y 27.



```

1  proyecto_back > JS app.js > ...
2  6  var database = require ('./config/database');
3  7
4  8  var indexRouter = require('./routes/index');
5  9  var usersRouter = require('./routes/users');
6  10 var productosRouter = require('./routes/productos.router');
7  11
8  12 var app = express();
9  13
10 14
11 15 app.use(logger('dev'));
12 16 app.use(express.json());
13 17 app.use(express.urlencoded({ extended: false }));
14 18 app.use(cookieParser());
15 19 app.use(express.static(path.join(__dirname, 'public')));
16 20
17 21 //Mongo connection
18 22 database.mongoConnect();
19 23
20 24 //Router
21 25 app.use('/', indexRouter);
22 26 app.use('/users', usersRouter);
23 27 app.use('/productos', productosRouter);
24 28
25 29
  
```

Después de correr la plataforma.



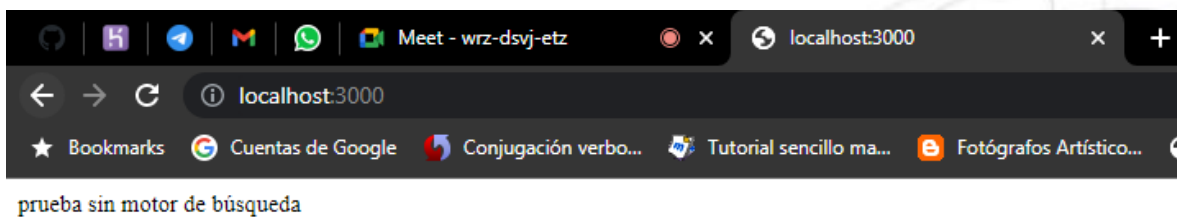
```

PS C:\Users\YAMILE\Desktop\proyecto plaza> cd proyecto_back
PS C:\Users\YAMILE\Desktop\proyecto plaza\proyecto_back> npm start

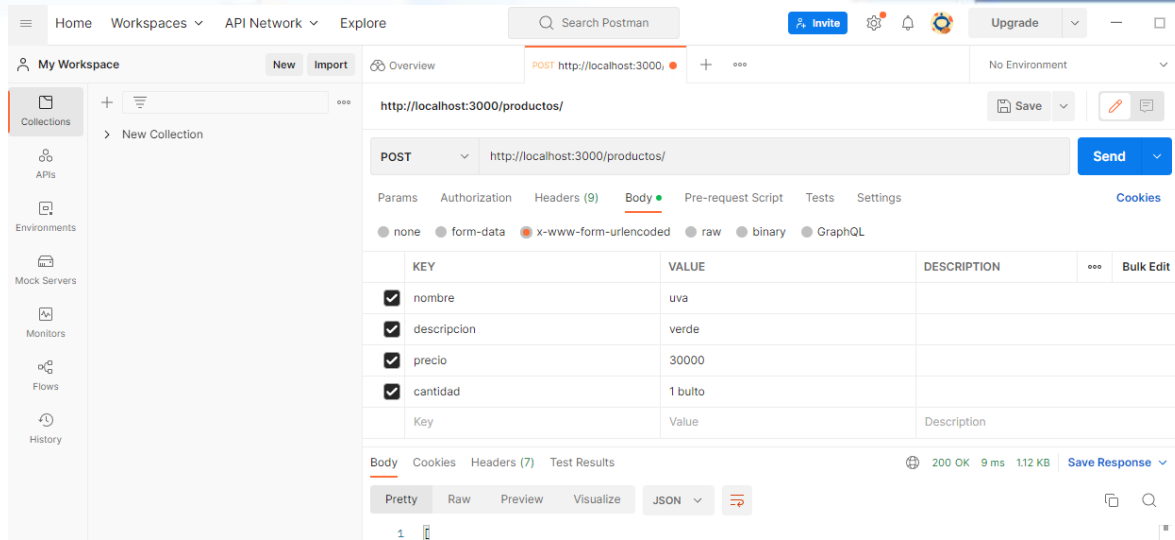
> proyecto-back@0.0.0 start
> nodemon ./bin/www

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/www`
Port 3000 is already in use
[nodemon] app crashed - waiting for file changes before starting...
  
```

Verifico navegador.



Y voy directamente a crear datos en postman con la opción de POST.

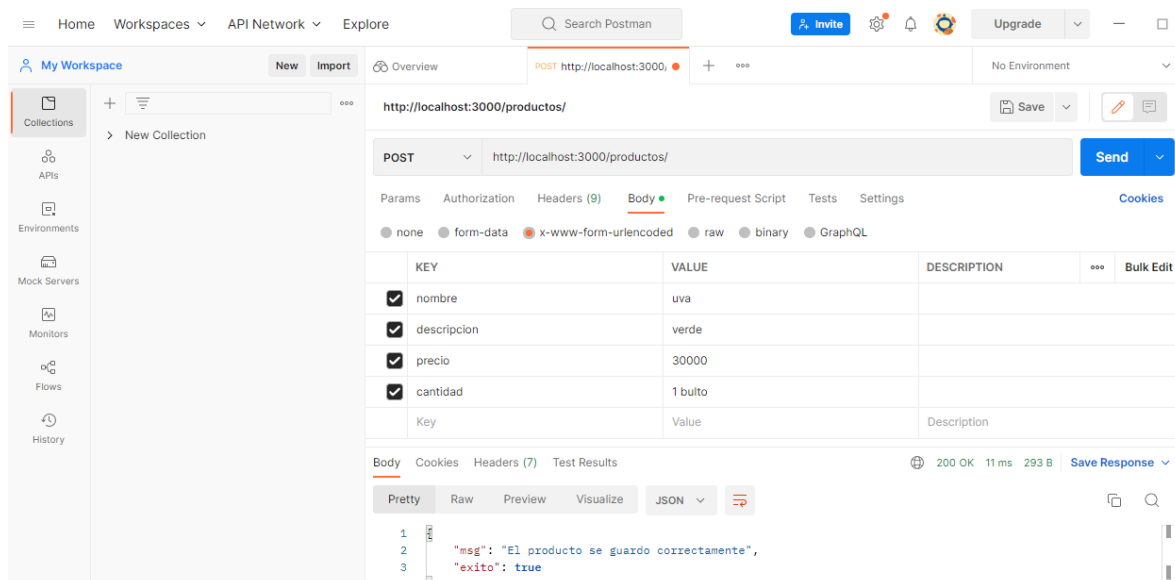


Postman interface showing a POST request to `http://localhost:3000/productos/`. The request body is configured with the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nombre	uva	
<input checked="" type="checkbox"/> descripción	verde	
<input checked="" type="checkbox"/> precio	30000	
<input checked="" type="checkbox"/> cantidad	1 bulto	
Key	Value	Description

The response status is 200 OK, 9 ms, 1.12 KB. The response body is currently empty.

Al dar clic en Send verifico mensaje de confirmación o de error.



Postman interface showing the response to the POST request. The response status is 200 OK, 11 ms, 293 B. The response body is a JSON object:

```

1 {
2   "msg": "El producto se guardo correctamente",
3   "exito": true
}
```

Verifica base en mongo y producto registrado.

Connect View Collection Help

localhost:27017

5 DBS 5 COLLECTIONS

FAVORITE

HOST localhost:27017

CLUSTER Standalone

EDITION MongoDB 6.0.2 Community

My Queries

Databases

Filter your data

- admin
- config
- em
- local
- plaza
- productos

Documents

plaza.productos

7 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' }

ADD DATA VIEW

Displaying documents 1 - 7 of 7

```

description: "lima xxx"
precio: "100000"
cantidad: "1 bulto"
__v: 0

_id: ObjectId('6369a10f5198d8378bd93078')
nombre: "Pera"
descripcion: "ricas en sales minerales y en vitaminas C y A. La vitamina A es neces..."
precio: "80000"
cantidad: "1 bulto"
__v: 0

_id: ObjectId('6369a3dc5198d8378bd93081')
nombre: "uva"
descripcion: "verde"
precio: "30000"
cantidad: "1 bulto"
__v: 0
    
```

MongoDB Compass - localhost:27017/plaza.productos

Connect View Collection Help

localhost:27017

5 DBS 5 COLLECTIONS

FAVORITE

HOST localhost:27017

CLUSTER Standalone

EDITION MongoDB 6.0.2 Community

My Queries

Databases

Filter your data

- admin
- config
- em
- local
- plaza
- productos

Documents

plaza.productos

6 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' }

ADD DATA VIEW

Displaying documents 1 - 6 of 6

```

description: "tangelo"
precio: "100000"
cantidad: "1 bulto"
__v: 0

_id: ObjectId('6366fe23ffd236aba9120a53')
nombre: "lima"
descripcion: "lima xxx"
precio: "100000"
cantidad: "1 bulto"
__v: 0

_id: ObjectId('6369a10f5198d8378bd93078')
nombre: "Pera"
descripcion: "ricas en sales minerales y en vitaminas C y A. La vitamina A es neces..."
precio: "80000"
cantidad: "1 bulto"
__v: 0
    
```

Listo con GET

Postman interface showing a GET request to `http://localhost:3000/productos/`. The response is a JSON array with one object:

```

1 {
2   "_id": "6366e9fde7dfa7e04cfe3fe6",
3   "nombre": "manzana",
4   "descripcion": "verde",
5   "precio": "$50000",
6   "cantidad": "1 bulto",
7   "__v": 0
8 }

```

Para editar copiamos el id de producto a modificar desde la base en mongo

MongoDB Compass interface showing the `plaza.productos` collection. A document is selected for editing:

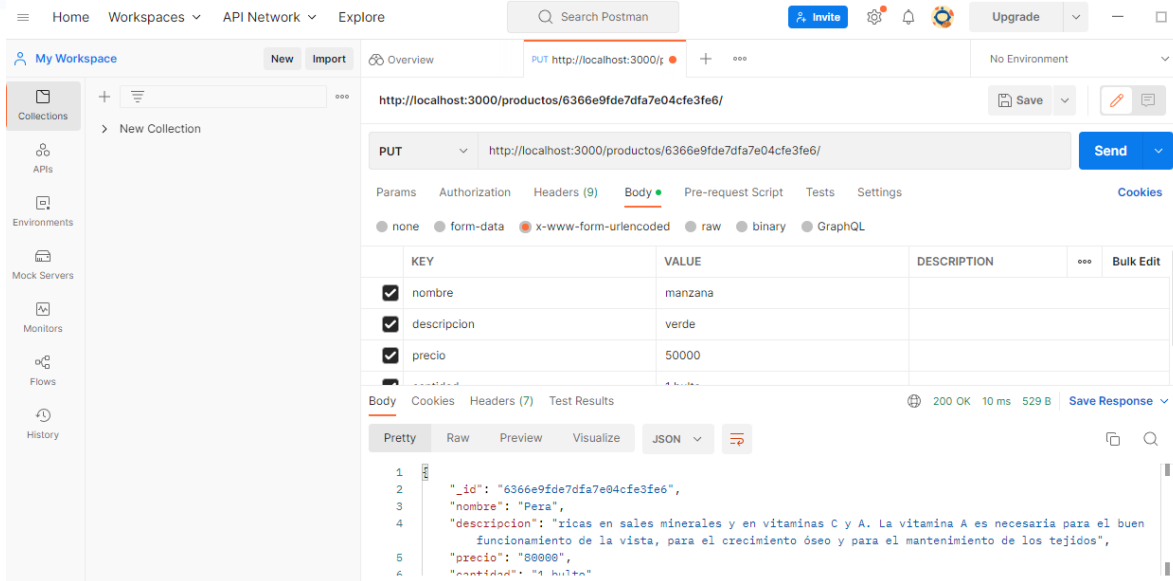
```

1 _id: ObjectId('6366e9fde7dfa7e04cfe3fe6')
2 nombre: "Pera"
3 descripcion: "ricas en sales minerales y en vitaminas C y A. La vitamina A es neces"
4 precio: "80000"
5 cantidad: "1 bulto"
6 __v: 0

```

The `ObjectID` field is highlighted, and the `UPDATE` button is visible.

Posterior en postman con PUT modifiko VALUE de producto



Home Workspaces API Network Explore Search Postman

My Workspace New Import Overview PUT http://localhost:3000/

http://localhost:3000/productos/6366e9fde7dfa7e04cfe3fe6/ Save

PUT http://localhost:3000/productos/6366e9fde7dfa7e04cfe3fe6/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION	Bulk Edit
<input checked="" type="checkbox"/> nombre	manzana		
<input checked="" type="checkbox"/> descripcion	verde		
<input checked="" type="checkbox"/> precio	50000		

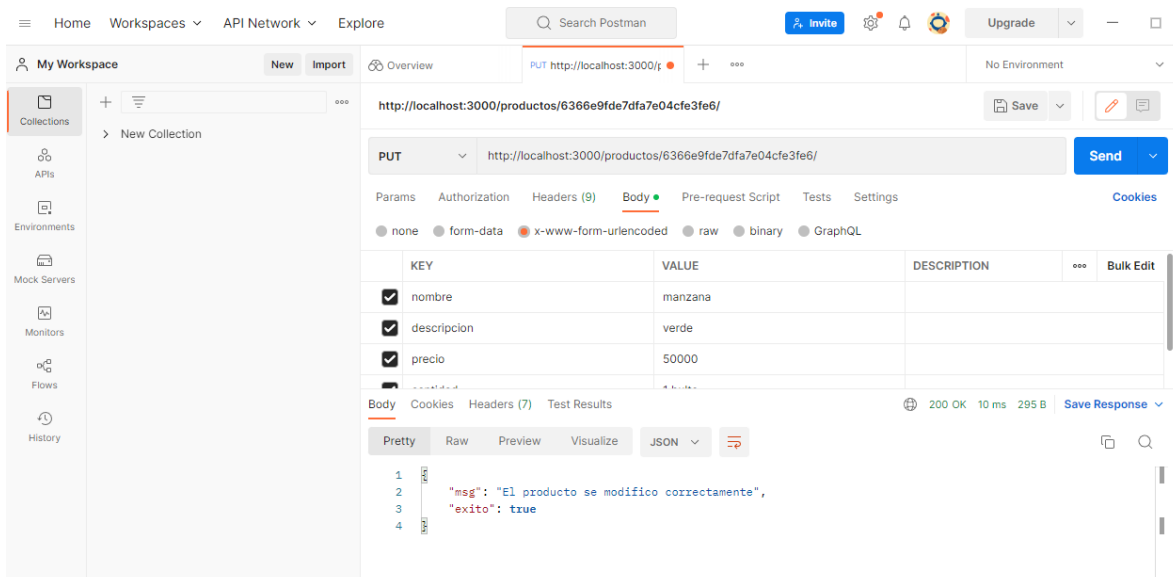
Body Cookies Headers (7) Test Results 200 OK 10 ms 529 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": "6366e9fde7dfa7e04cfe3fe6",
3   "nombre": "Pera",
4   "descripcion": "Ricas en sales minerales y en vitaminas C y A. La vitamina A es necesaria para el buen funcionamiento de la vista, para el crecimiento óseo y para el mantenimiento de los tejidos",
5   "precio": "80000",
6   "cantidad": "1 unit+1"
  }
  
```

Al dar clic en send



Home Workspaces API Network Explore Search Postman

My Workspace New Import Overview PUT http://localhost:3000/

http://localhost:3000/productos/6366e9fde7dfa7e04cfe3fe6/ Save

PUT http://localhost:3000/productos/6366e9fde7dfa7e04cfe3fe6/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION	Bulk Edit
<input checked="" type="checkbox"/> nombre	manzana		
<input checked="" type="checkbox"/> descripcion	verde		
<input checked="" type="checkbox"/> precio	50000		

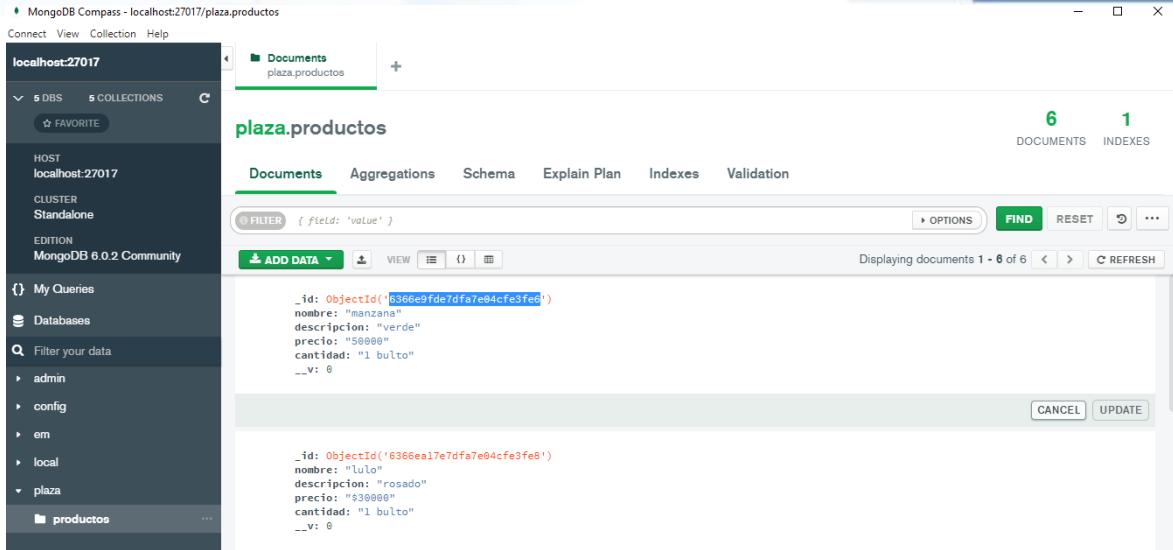
Body Cookies Headers (7) Test Results 200 OK 10 ms 295 B Save Response

Pretty Raw Preview Visualize JSON

```

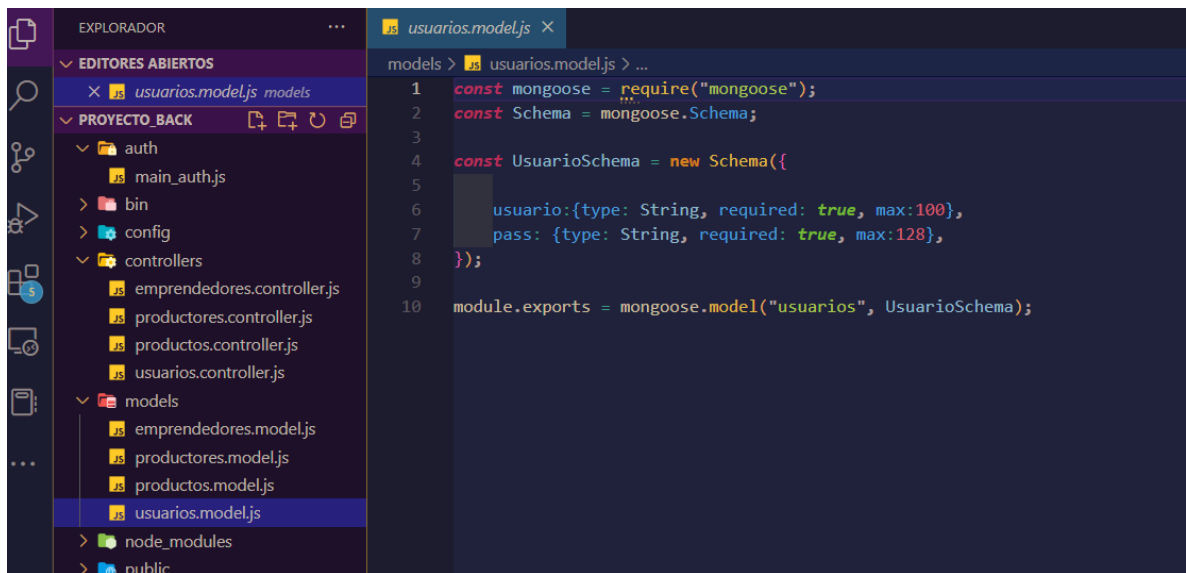
1 {
2   "msg": "El producto se modifico correctamente",
3   "exito": true
4 }
  
```

Emita mensaje el producto se modificó correctamente, verifico en base mongo y producto modificado

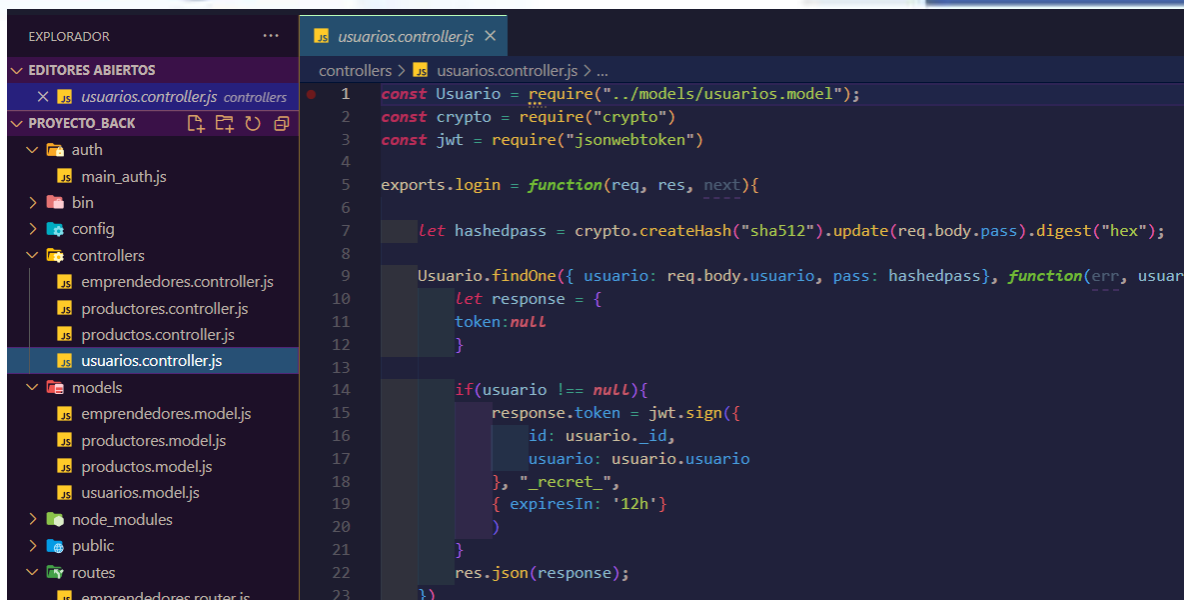


❖ Autenticación y permiso de usuario

Se crea la tabla usuarios en la carpeta models



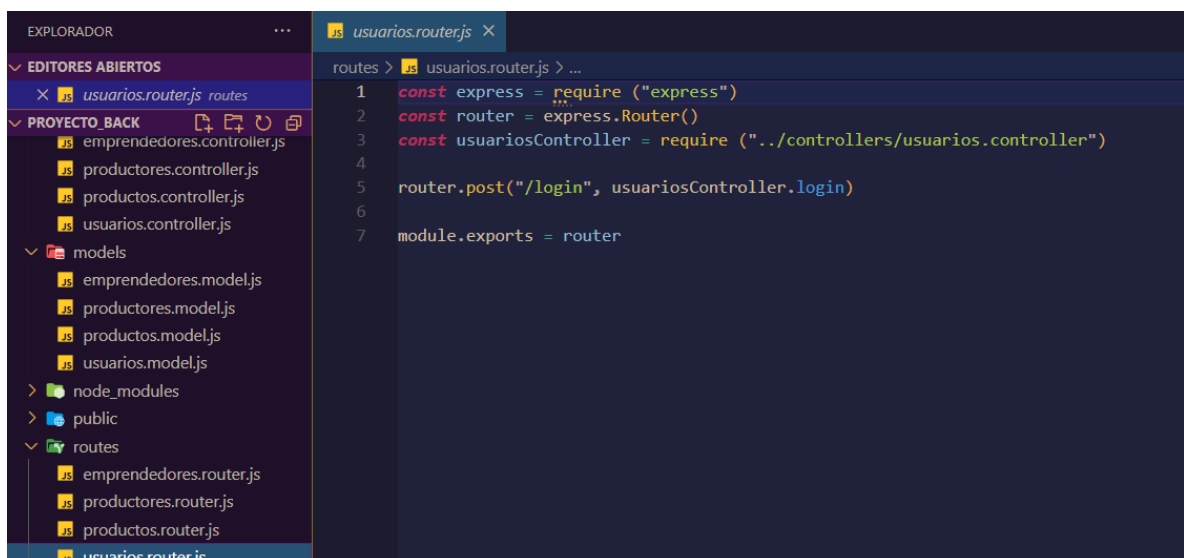
Luego se crea el controlador de usuario



```

1  const Usuario = require("../models/usuarios.model");
2  const crypto = require("crypto")
3  const jwt = require("jsonwebtoken")
4
5  exports.login = function(req, res, next){
6
7      let hashedpass = crypto.createHash("sha512").update(req.body.pass).digest("hex");
8
9      Usuario.findOne({ usuario: req.body.usuario, pass: hashedpass}, function(err, usuario){
10         let response = {
11             token:null
12         }
13
14         if(usuario !== null){
15             response.token = jwt.sign({
16                 id: usuario._id,
17                 usuario: usuario.usuario
18             }, "_secret_",
19             { expiresIn: '12h' }
20         )
21         }
22         res.json(response);
23     })
    
```

Se genera la ruta de usuario



```

1  const express = require ("express")
2  const router = express.Router()
3  const usuariosController = require ("../controllers/usuarios.controller")
4
5  router.post("/login", usuariosController.login)
6
7  module.exports = router
    
```

Se crea el encriptamiento

```
EXPLORADOR
EDITORES ABIERTOS
main_authjs auth
PROYECTO_BACK
auth
main_authjs
bin
config
controllers
emprendedores.controller.js
productores.controller.js
productos.controller.js
usuarios.controller.js
models
emprendedores.model.js
productores.model.js
productos.model.js

main_authjs
auth
1 const jwt = require("jsonwebtoken")
2
3 const auth = (req, res, next) => {
4   try {
5     const token = req.headers.authorization.split(" ")[1]
6     const decoded = jwt.verify(token, "_recret_")
7     req.usuario = decoded
8     next()
9   } catch (error){
10    res.status(401)
11    res.json({code:4, msg:"no tiene autorizacion para ver el contenido"})
12  }
13 }
14
15 module.exports = auth
16
```

Y la app.js se generan los llamados, las variables y los objetos

```
app.js
app.js > ...
1 var createError = require('http-errors');
2 var express = require('express');
3 var path = require('path');
4 var cookieParser = require('cookie-parser');
5 var logger = require('morgan');
6 var database = require ("./config/database");
7 var auth = require ("./auth/main_auth")
8
9 var productosRouter = require('./routes/productos.router');
10 var productoresRouter = require('./routes/productores.router');
11 var emprendedoresRouter = require('./routes/emprendedores.router');
12 var usuariosRouter = require('./routes/usuarios.router');
13 var app = express();
14
```

Se conecta a la base con el encriptamiento

```
//Mongo connection
database.mongoConnect();

app.use('/usuarios',usuariosRouter);
app.use(auth);

//Router
app.use('/productos', productosRouter);
app.use('/productores', productoresRouter);
app.use('/emprendedores', emprendedoresRouter);
```

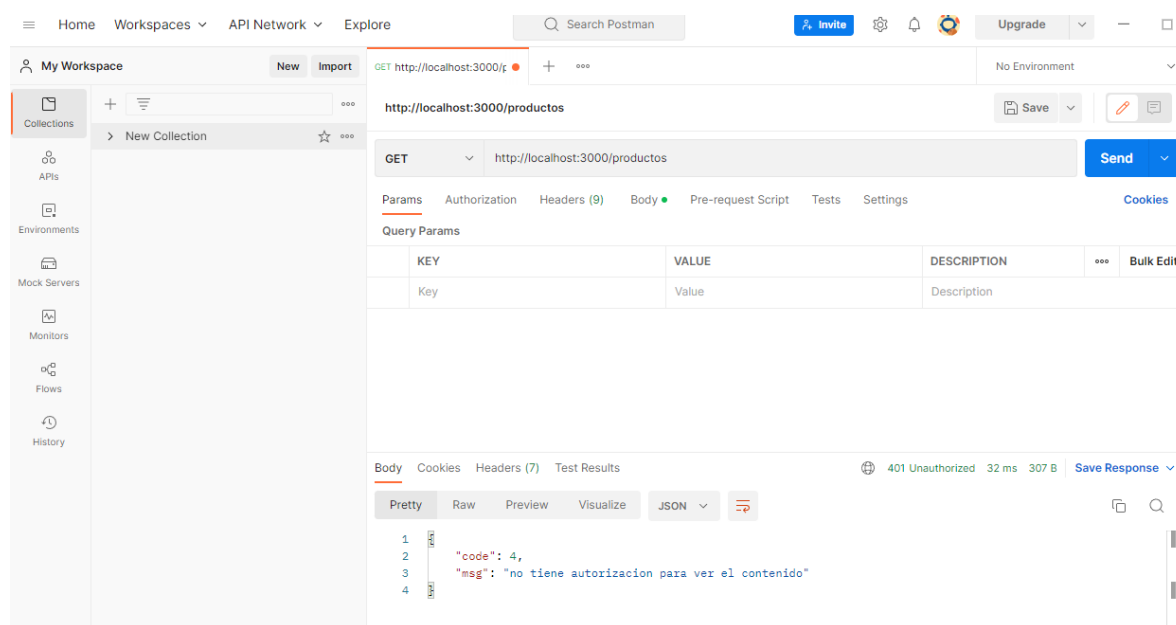
Se ejecuta el programa

```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  JUPYTER  CO
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/www`
GET /productos 401 14.604 ms - 62

```

Verificamos con POSTMAN



Nos arroja notificación “No tiene autorización para ver el contenido”

POST http://localhost:3000/usuarios/login

Body

```
{
  "usuario": "admin",
  "pass": "admin"
}
```

Response (200 OK, 41 ms, 443 B)

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzNzE4NjQ0bWUxYjF1bnJFk0SIsInVzdwFyaW81OjZhZG1pb1IsImh0bCI6MTY2ODM4ODM3NywiZmxwIjoxNjY4NDMxNTc3fQ.za_Wmu5Xq-1j-8HBuh3g3KU5cBpRITutWjpMOBFf1ck"
}
```

Después de verificar que no permite el ingreso verificamos la autenticación en POST y allí descarga automáticamente el token.

POST http://localhost:3000/productos/

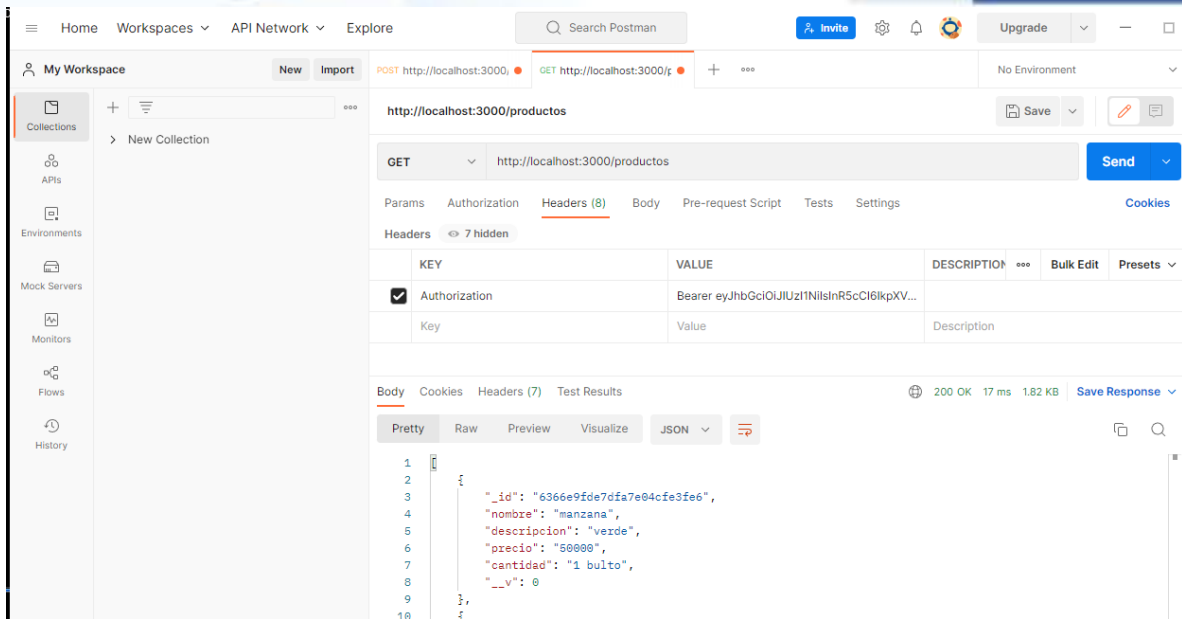
Headers (10)

KEY	VALUE	DESCRIPTION
Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzNzE4NjQ0bWUxYjF1bnJFk0SIsInVzdwFyaW81OjZhZG1pb1IsImh0bCI6MTY2ODM4ODM3NywiZmxwIjoxNjY4NDMxNTc3fQ.za_Wmu5Xq-1j-8HBuh3g3KU5cBpRITutWjpMOBFf1ck	

Response (200 OK, 41 ms, 443 B)

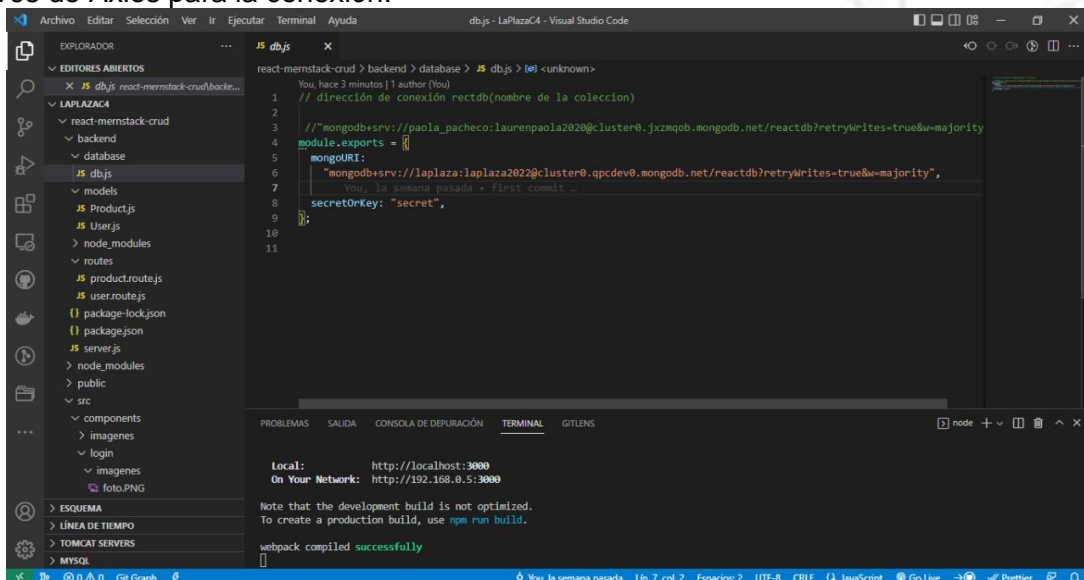
```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzNzE4NjQ0bWUxYjF1bnJFk0SIsInVzdwFyaW81OjZhZG1pb1IsImh0bCI6MTY2ODM4ODM3NywiZmxwIjoxNjY4NDMxNTc3fQ.za_Wmu5Xq-1j-8HBuh3g3KU5cBpRITutWjpMOBFf1ck"
}
```

Este token lo copiamos en el value del GET cual nos permite visualizar el contenido de los productos.



❖ Se hace la conexión Backend y fronted

Para la unión de backend y frontend se crea la carpeta server con su respectivo archivo `server.js` haciendo la unión de las bases de datos en este caso productos y usuarios. Para cada una de las historias de usuario llamadas componentes, se crean una función con los respectivos constructores el cual posteriormente hace el llamado por medio de la `app.js` a través de Axios para la conexión.



En el repositorio mongo atlas se almacenar la base

```

1 let express = require("express");
2 let mongoose = require("mongoose");
3 let cors = require("cors");
4 let bodyParser = require("body-parser");
5 const createError = require("http-errors");
6
7
8 const userRoute = require("../backend/routes/user.route");
9 const productRoute = require("../backend/routes/product.route");
10
11 mongoose.Promise = global.Promise;
12 // eslint-disable-next-line no-undef
13 mongoose.connect('mongodb+srv://laplaza:laplaza2022@cluster0.qpcdev0.mongodb.net/reactdb?retryWrites=true&w=majority');
14 () => {
15   console.log("Database successfully connected!");
16 },
17 (error) => {
18   console.log("Could not connect to database : " + error);
19 }
20 );
21 const app = express();
22 app.use(bodyParser.json());
23 app.use(bodyParser.urlencoded({ extended: true }));
24 app.use(cors());
25 app.use("/users", userRoute);
26 app.use("/products", productRoute);
27
28 PORT
29 const port = process.env.PORT || 4000;
  
```

En server permite la ejecución del backend a través del nodemon.js y además un usuario y producto en un mismo repositorio

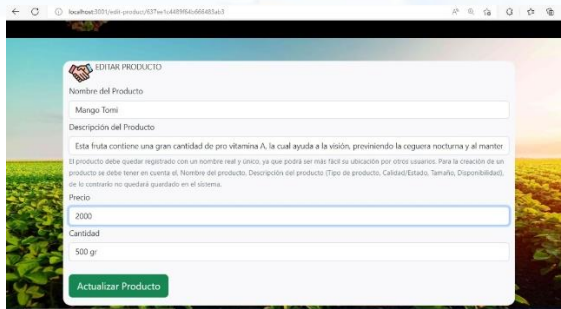
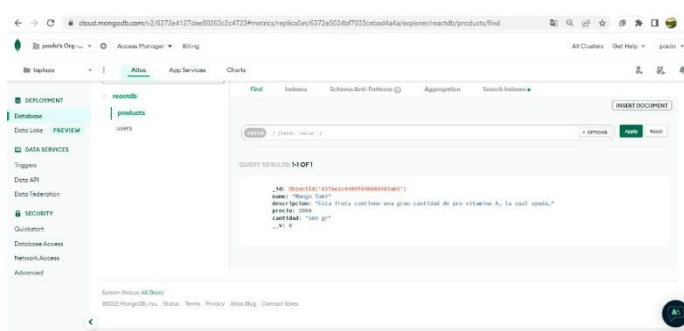
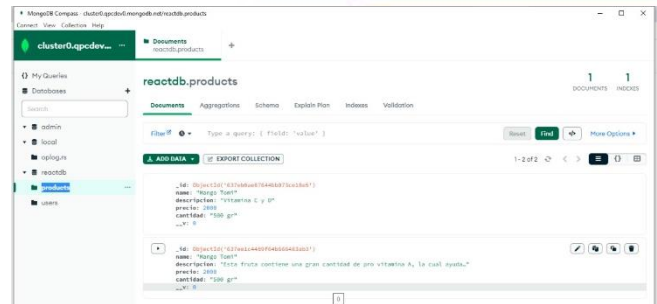
```

C:\Windows\system32\cmd.exe - "node" "C:\Users\Paola\AppData\Roaming\npm\node_modules\nodemon\bin\nodemon.js" server.js
Microsoft Windows [Versión 10.0.19044.2251]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Paola\Desktop\LaPlaza4\react-mernstack-crud\backend>nodemon server.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting 'node server.js'
Connected to port 4000
Database successfully connected!
  
```

De esta forma se conecta con el crud. A continuación, imágenes del fronted de como se modifica producto ya creado “Mango Yulima” y posterior se edita a “Mango Tomi” y finalmente se elimina.



9. Video: Reunión Sprint review.

<https://www.youtube.com/watch?v=-MKRI3y2EAQ>

10. Informe de Retrospectiva.

A lo largo de este ciclo nos permitió dar espacio para la creatividad y recursividad. Aplicar los conocimientos obtenidos en el proyecto y el aporte de cada uno de los integrantes fue clave para poder desarrollar este sprint. Se tuvo éxito en la ejecución de las historias de usuario, las cuales se tenían dispuestas para este ciclo tanto ingreso de sesión, registro de usuario y se conectó lo elaborado del frontend con el backend.

En el desarrollo de este sprint la dificultad que hubo fue en la unión del back con el front pero se logró desarrollar con el apoyo de los tutores designados los cuales aportaron su conocimiento para dar una solución apropiada a este obstáculo. Las mejoras a tener en cuenta en la próxima iteración serían una mejor distribución del trabajo para que sea equitativo, una mejor comunicación para planear y acodar los diferentes trabajos de cada integrante.

En las clases vistas dentro de este ciclo se desarrolló componentes del proyecto (autenticación de los usuarios, generar la conexión de frontend y backend, ingreso de sesión) se trabajó con la herramienta Axios, Mongo Atlas. En el desarrollo de cada sprint incluye: planificación, análisis de requisitos, diseño, codificación, revisión y documentación con el fin de lograr el objetivo de complementar el proyecto de una manera organizada y eficaz.

11. Historias a trabajar en el siguiente Sprint.

Se va a elaborar para el Spring 4 se genera el despliegue en la nube