

Especificación de Requerimientos

Asociación de agricultores

Plaza Campesina

La Plaza

Tabla de Contenido

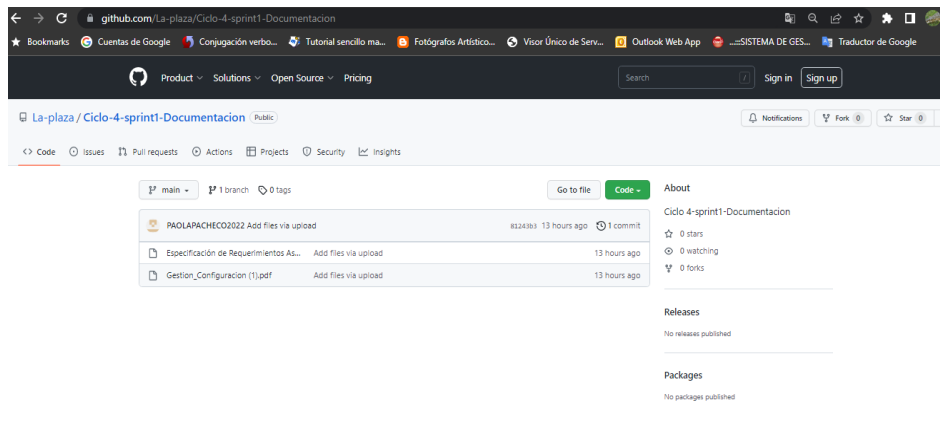
1.	Tabla 1. Nombres, correo electrónico y rol del equipo de trabajo SCRUM.	3
2.	Repositorio GitHub.....	4
3.	Gestión de configuración	4
1.	Product Backlog.	6
2.	Tabla 2. Especificaciones del proyecto SCRUM.	7
3.	Tabla 3. Especificaciones de historias de usuario.	8
4.	Proceso de comercialización:	11
5.	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.	13
	https://www.youtube.com/watch?v=9-d20gius8A	13
10.	Informe de Retrospectiva	13
11.	Historias a trabajar en el siguiente Sprint.	14

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>



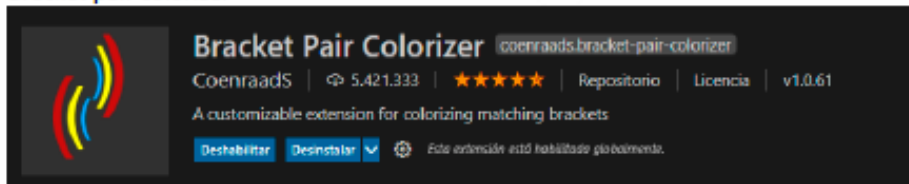
3. Gestión de configuración

Instalación del editor de código

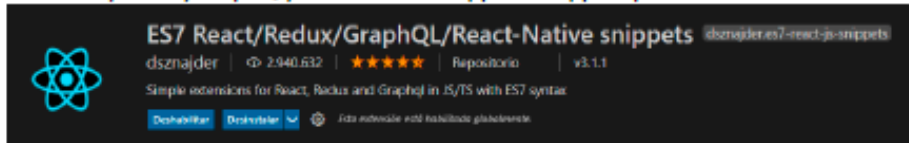
Selección de editor de código: Visual Studio Code

Instalación de extensiones:

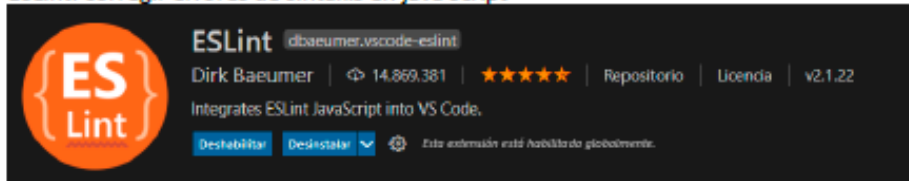
Bracket pair colorizer



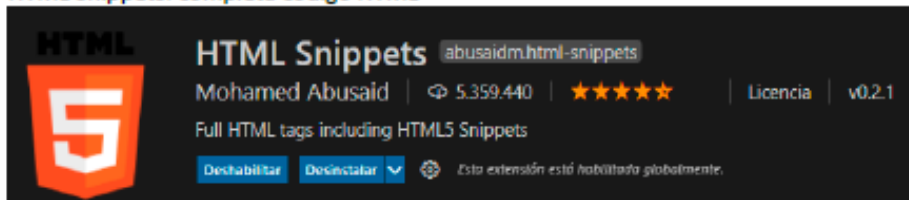
ES7 React/Redux/GraphQL/React-Native snippets: snippets para react



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



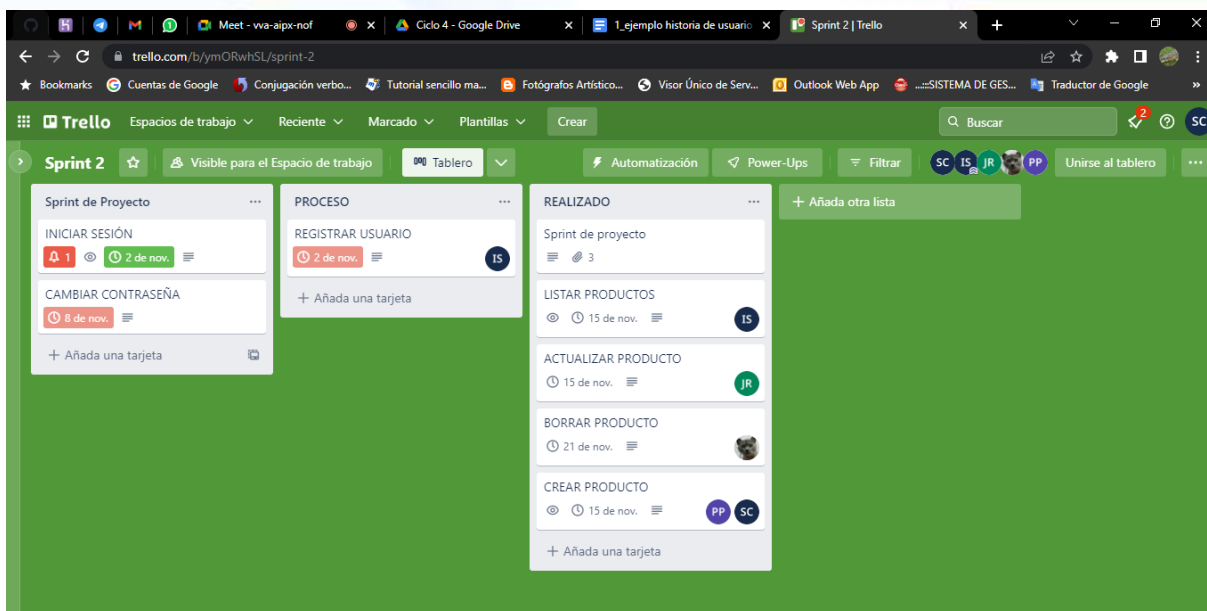
React Developer Tools se ha añadido a
Chrome

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/ymORwhSL/sprint-2>



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.	
<p>Criterios de Aceptación:</p> <p>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)”.</p> <p>Los datos para este registro son:</p> <p>Nombres y apellidos y/o razón social, Identificación, Ciudad/Municipio, Dirección, Email.</p>	

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.	
<p>Criterios de Aceptación:</p> <p>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)”.</p>	

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.	

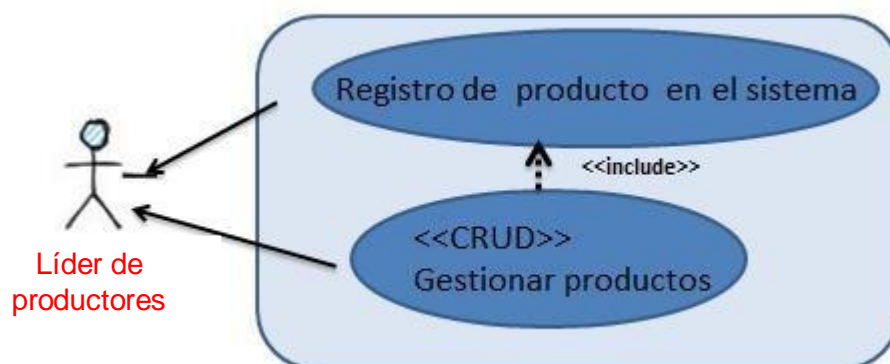
Este software tendrá el siguiente modulo;

- Módulo de productos

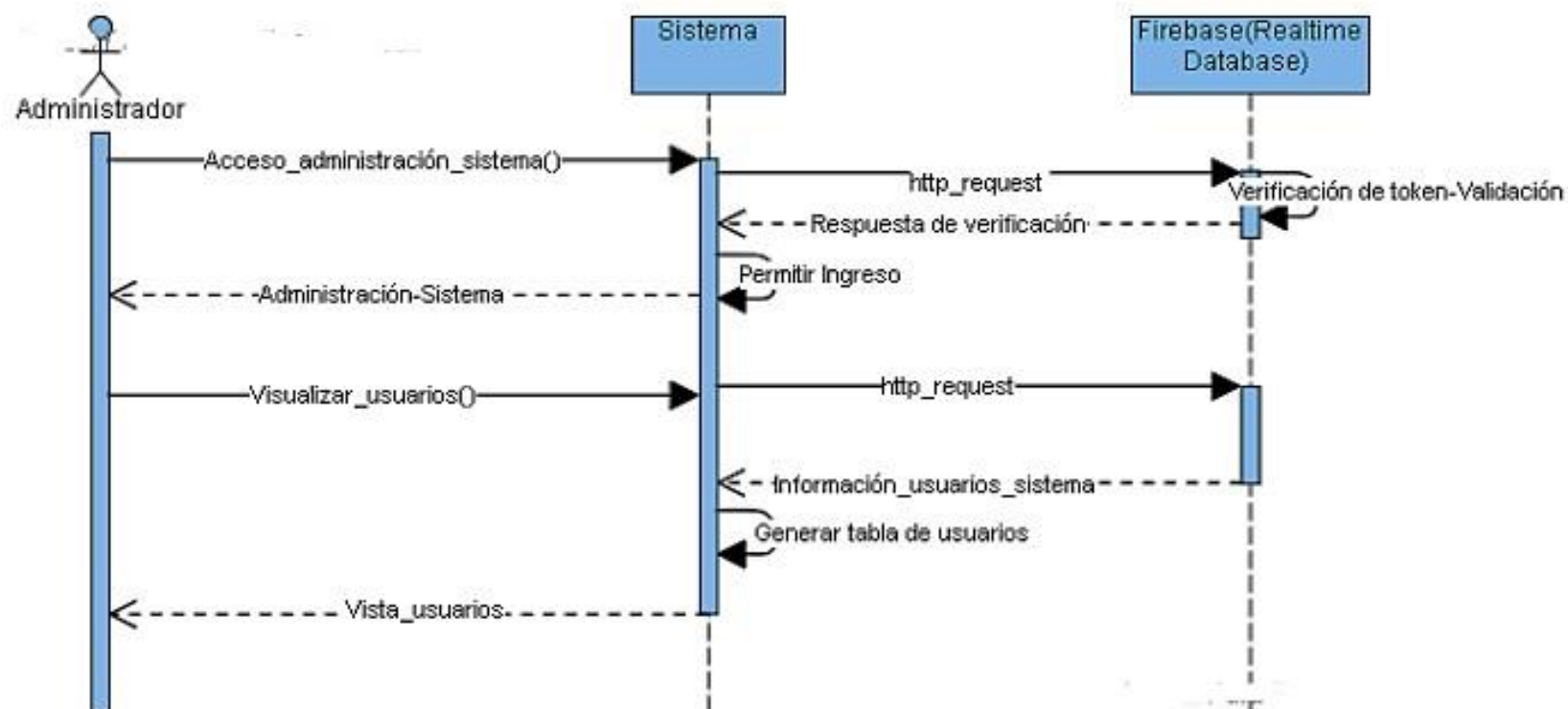
4. Proceso de comercialización:

PRODUTOR → PRODUCTO

5. Definición caso de uso:



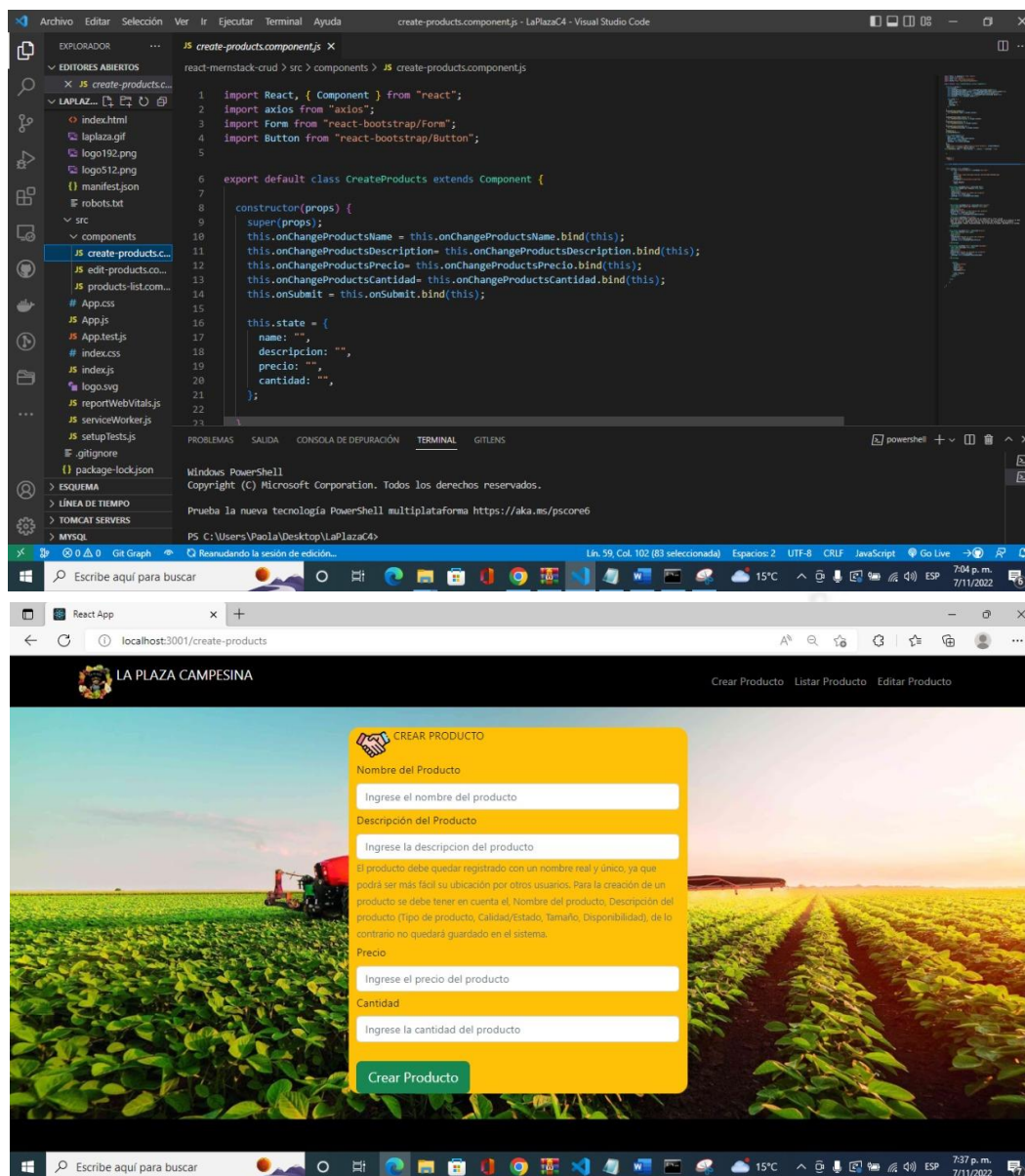
6. Diagrama de secuencia-ejemplo-acceso y petición administrador:

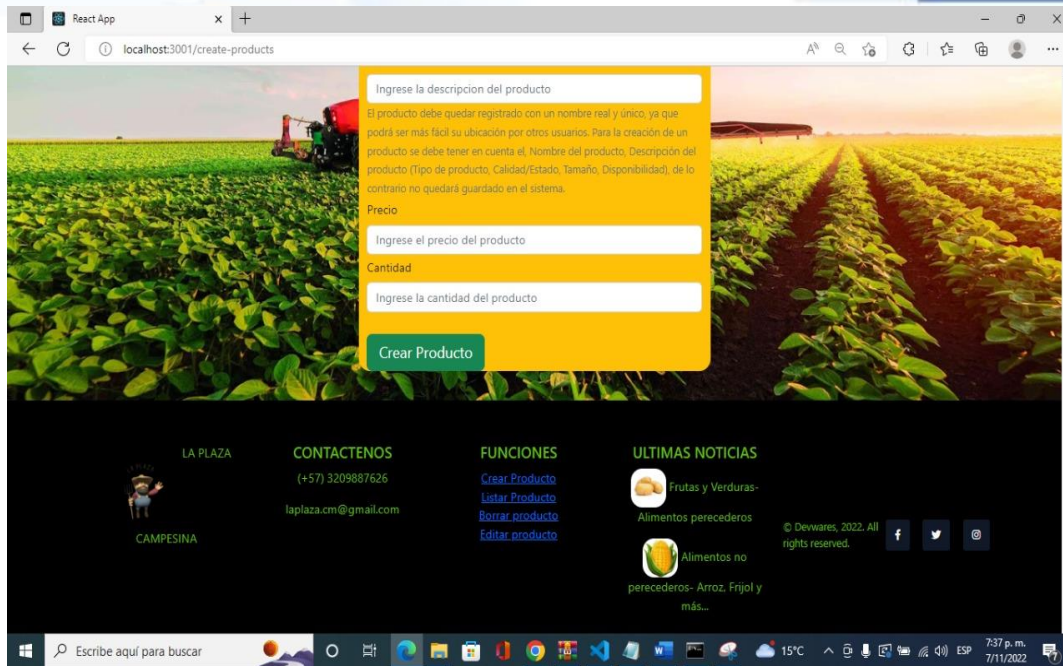


7. Desarrollo frontend

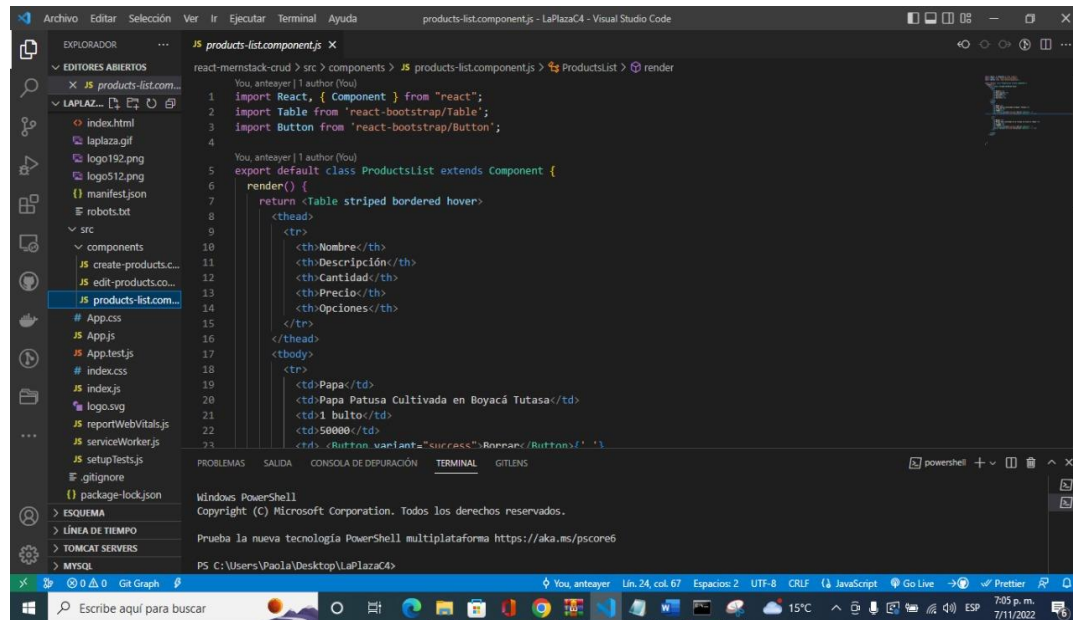
❖ Desarrollo de Componentes

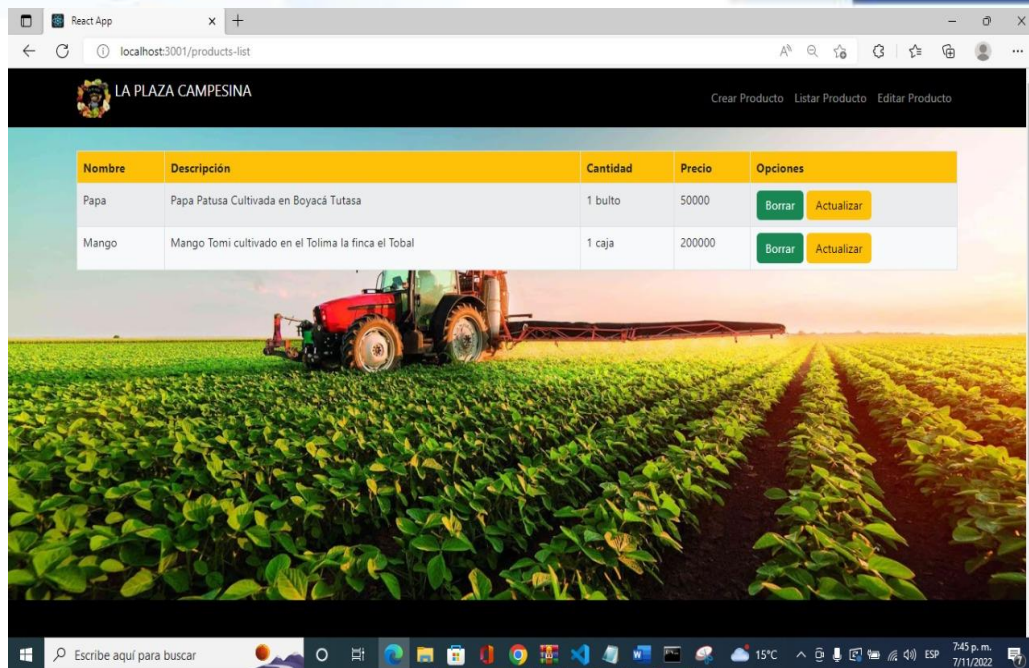
❖ Componente crear producto



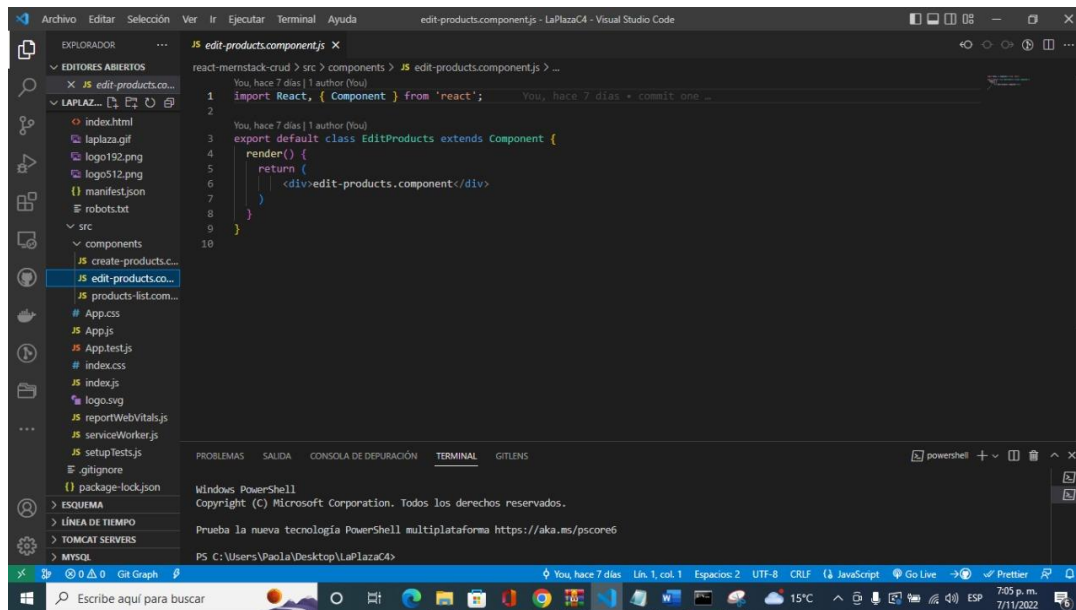


❖ Componente lista de producto





❖ Componente editar productos



❖ Componente app.js

```

1  react-mernstack-crud > src > JS App.js > ...
2  import './App.css';
3  import { CDBBtn, CDBIcon, CDBBox } from './cdblreact';
4
5  import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-dom';
6
7  import CreateProducts from './components/create-products.component';
8  import EditProducts from './components/edit-products.component';
9  import ProductsList from './components/products-list.component';
10
11 function App() {
12   return (
13     <div className="App">
14       <Router>
15         <header className="App-header">
16           <Navbar bg="black" variant="dark">
17             <Container>
18               <Navbar.Brand>
19
20               <Link to="/create-products" className="nav-link">
21                 
28               </Link>
29             </Navbar.Brand>
30           </Navbar>
31         </header>
32       </Router>
33     </div>
34   );
35 }
  
```

8. Desarrollo backend

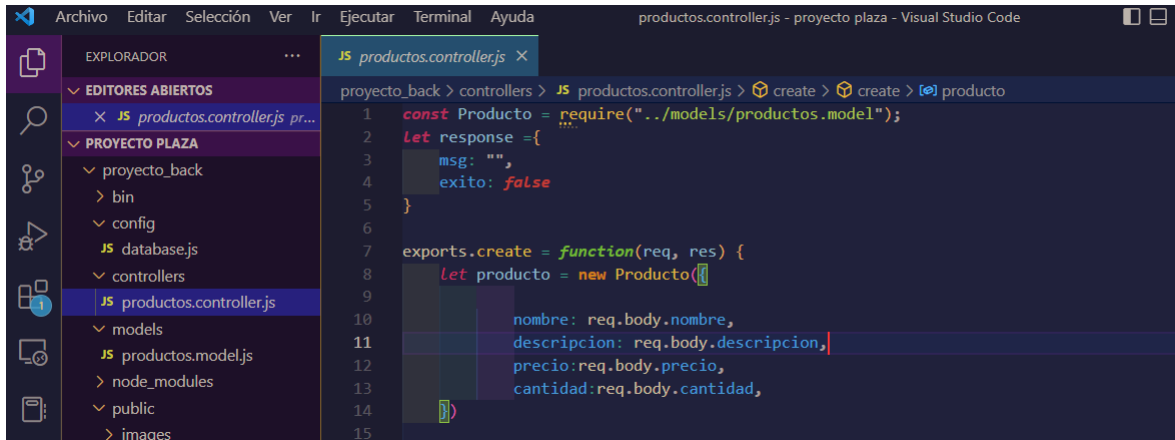
En carpeta config creo el archivo database.js el cual genera la conexión de la base con el host el puerto y nombre de la base de datos.

```

1  proyecto_back > config > JS database.js > mongoConnect > mongoConnect
2
3  const mongoose = require('mongoose');
4
5  const host = "localhost";
6  const port = "27017";
7  const db = "plaza";
8
9  exports.mongoConnect = () => {
10   const mongoStringConnection = `mongodb://${host}:${port}/${db}`;
11
12   mongoose.connect(mongoStringConnection);
13   mongoose.Promise = global.Promise;
14   const dbConnection = mongoose.connection;
15   dbConnection.on("error", console.error.bind(console, "Mongodb connection error"));
16 }
  
```


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

❖ Crea producto

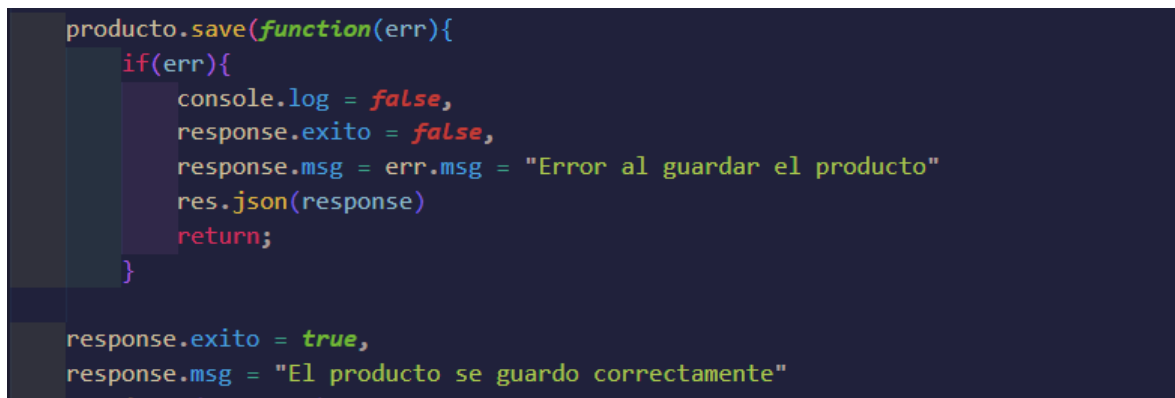


```

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



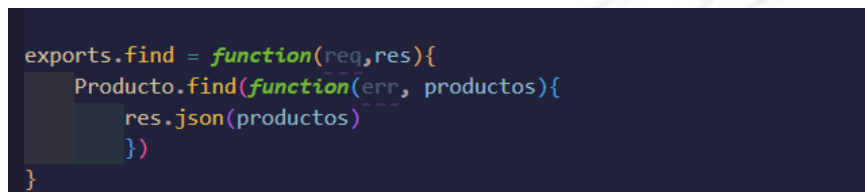
```

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

```

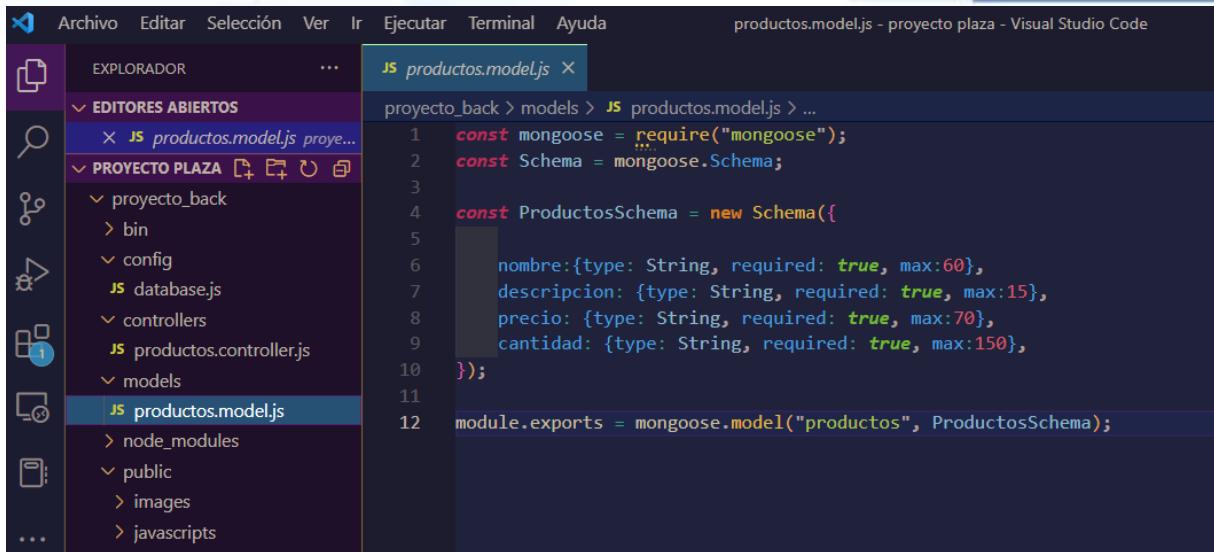
❖ Eliminar producto

```

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

```

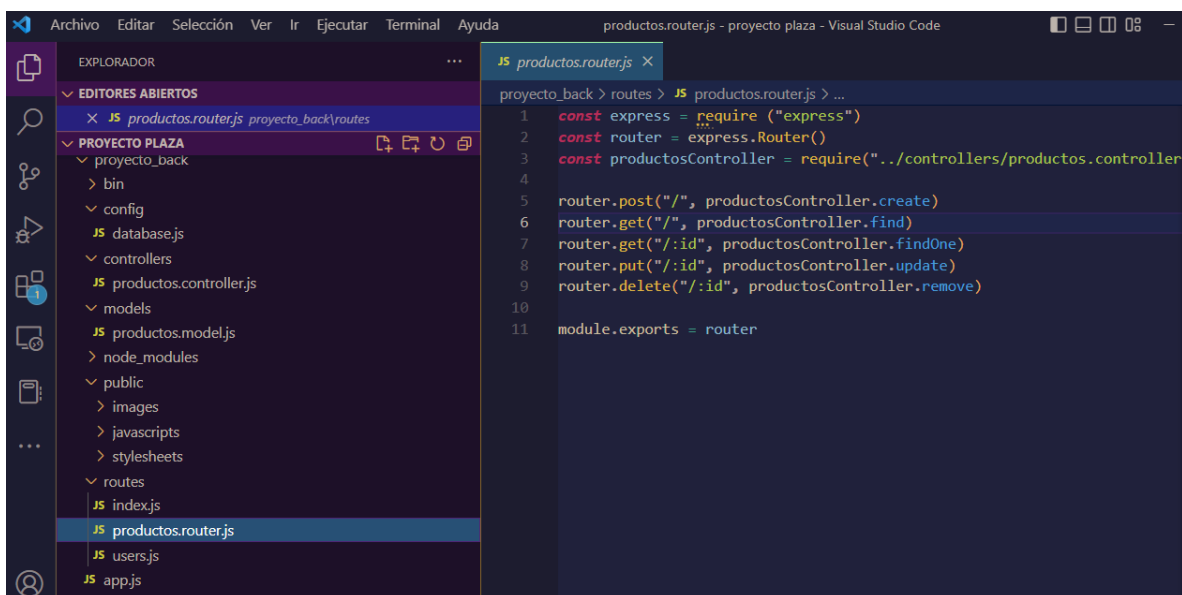
En la carpeta models se crea la cantidad de caracteres por cada atributo que posee la tabla de productos.



```

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);
  
```

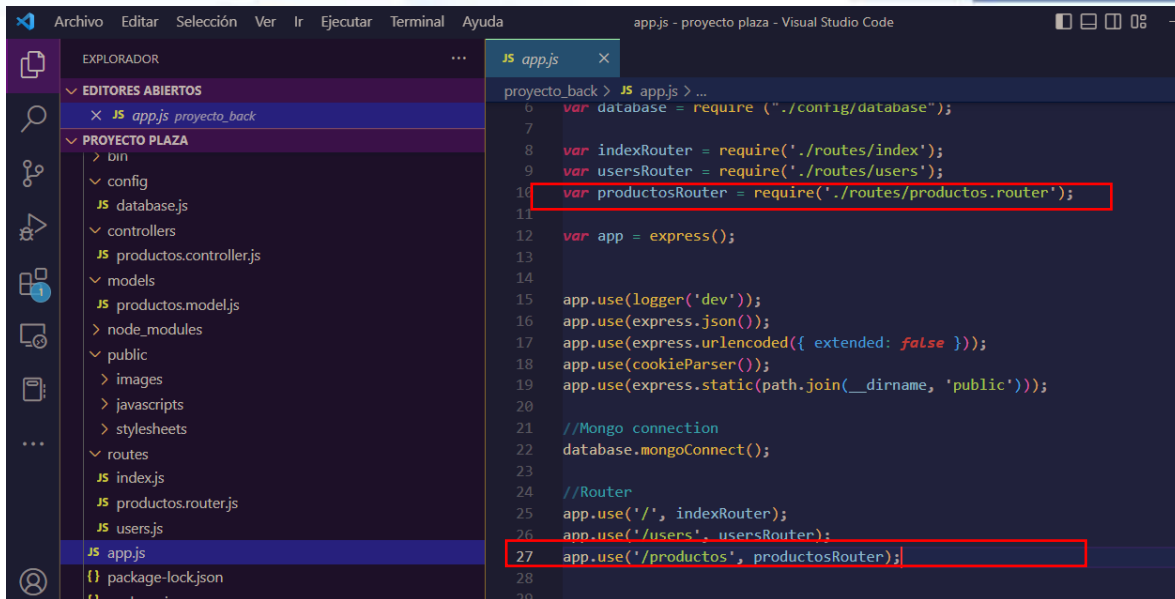
En la carpeta routers se enrutan las órdenes del postman post (crea) get (trae) put(modifica) delete (elimina).



```

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
  
```

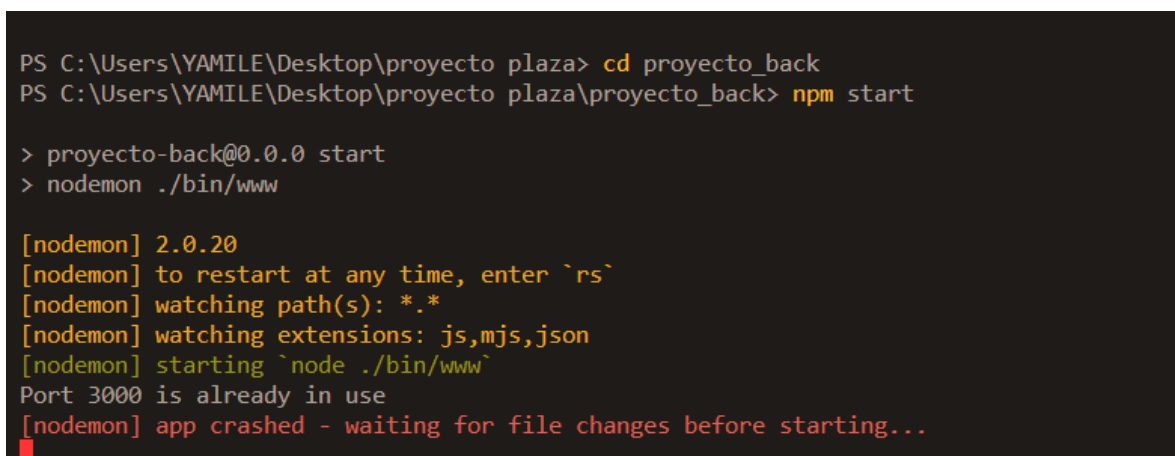
En el archivo principal app.js hacemos el llamado de nuestra variable en este caso línea 10 y 27.



```

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.



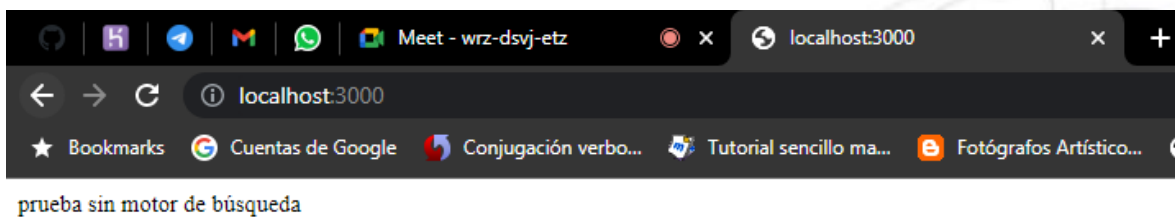
```

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

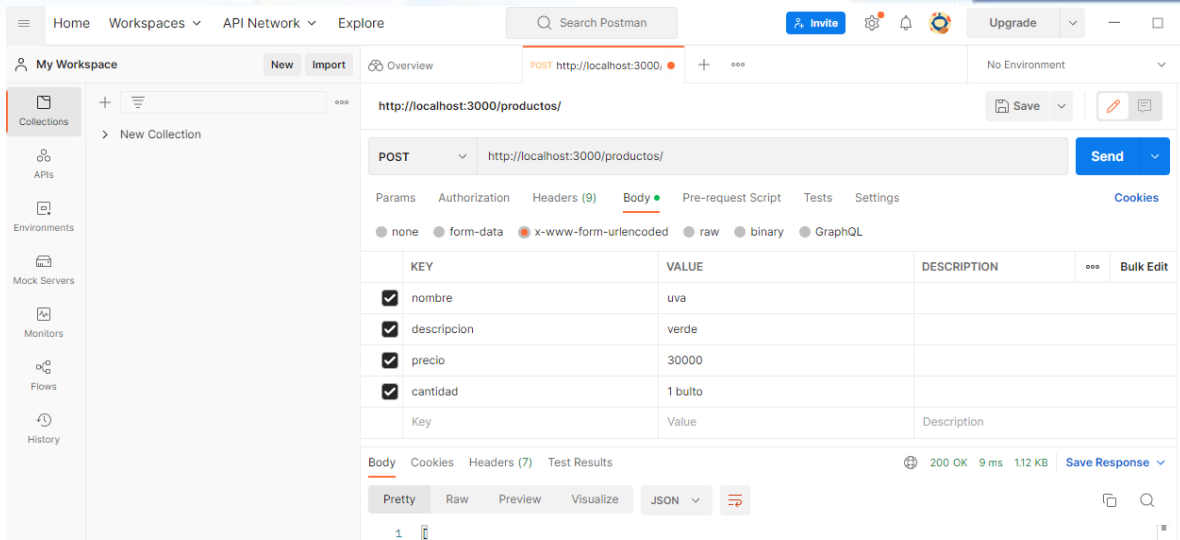
> proyecto-back@0.0.0 start
> nodemon ./bin/www

[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.

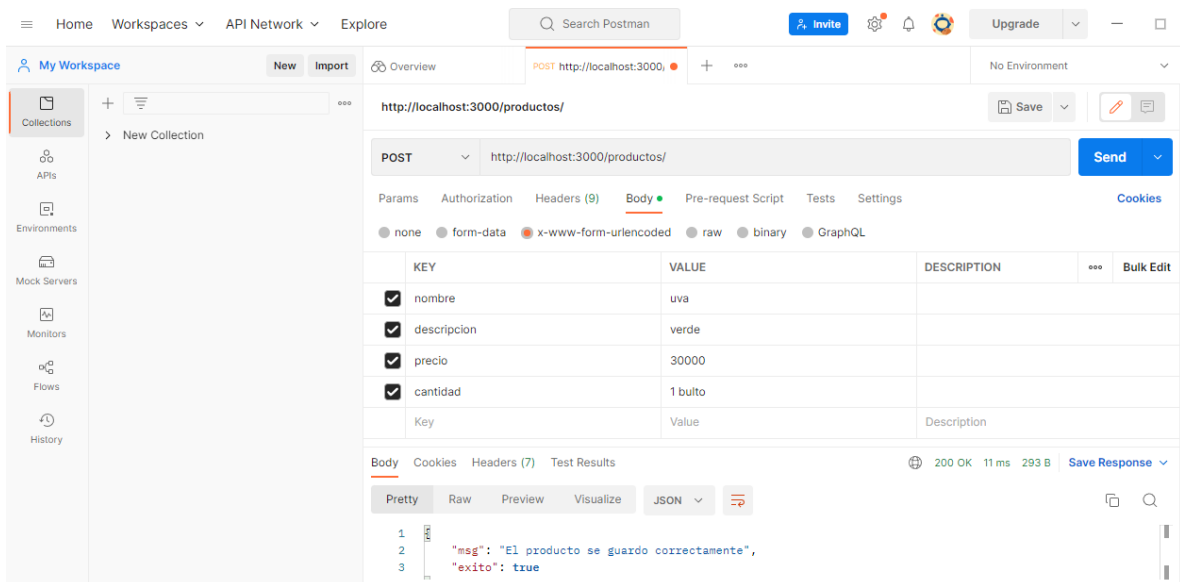


Postman interface showing a POST request to `http://localhost:3000/productos/`. The request body is configured with the following data:

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 interface shows the 'Body' tab selected, with a 'Send' button available.

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



Postman interface showing the response received after clicking 'Send'. The status is `200 OK` with a response time of `11 ms` and a body size of `293 B`. The response body is displayed in the 'Body' tab:

```

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

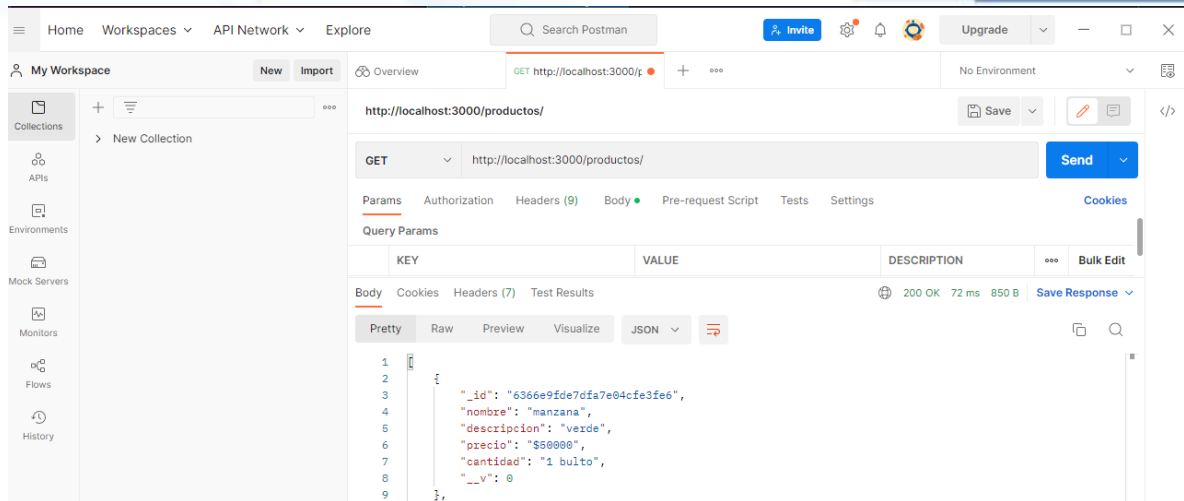
```

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



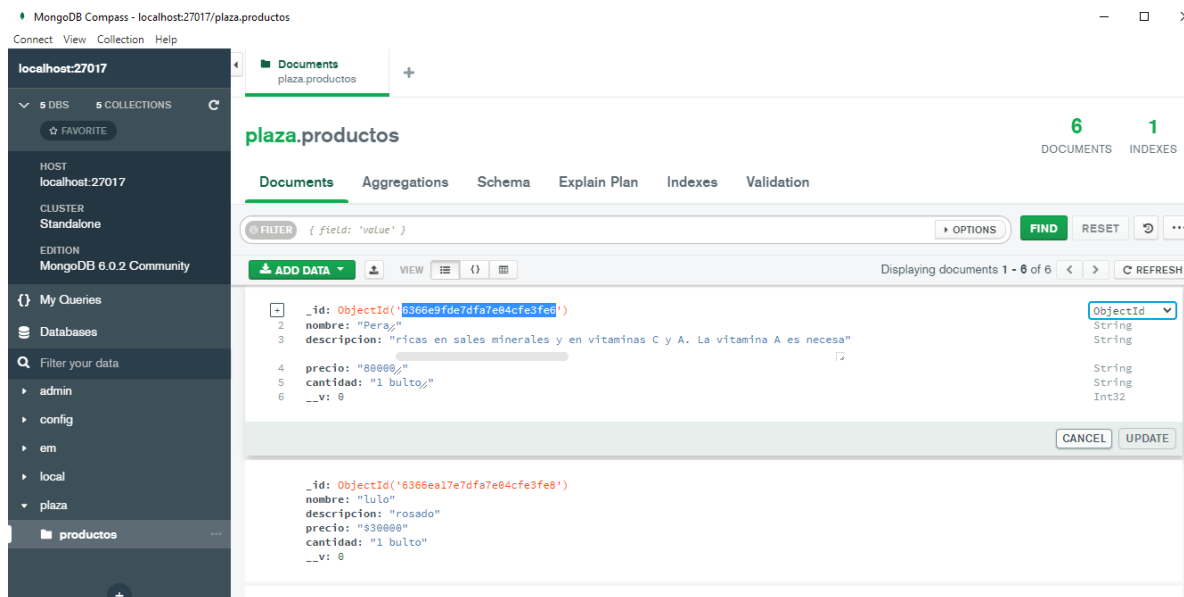
Postman interface showing a GET request to `http://localhost:3000/productos/`. The response is a JSON array with one object:

```

1 {
2   "_id": "6366e9fde7dfa7e04cfe3fe6",
3   "nombre": "manzana",
4   "descripcion": "verde",
5   "precio": "$50000",
6   "cantidad": "1 bulto",
7   "__v": 0
8 }
9 ]

```

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 update:

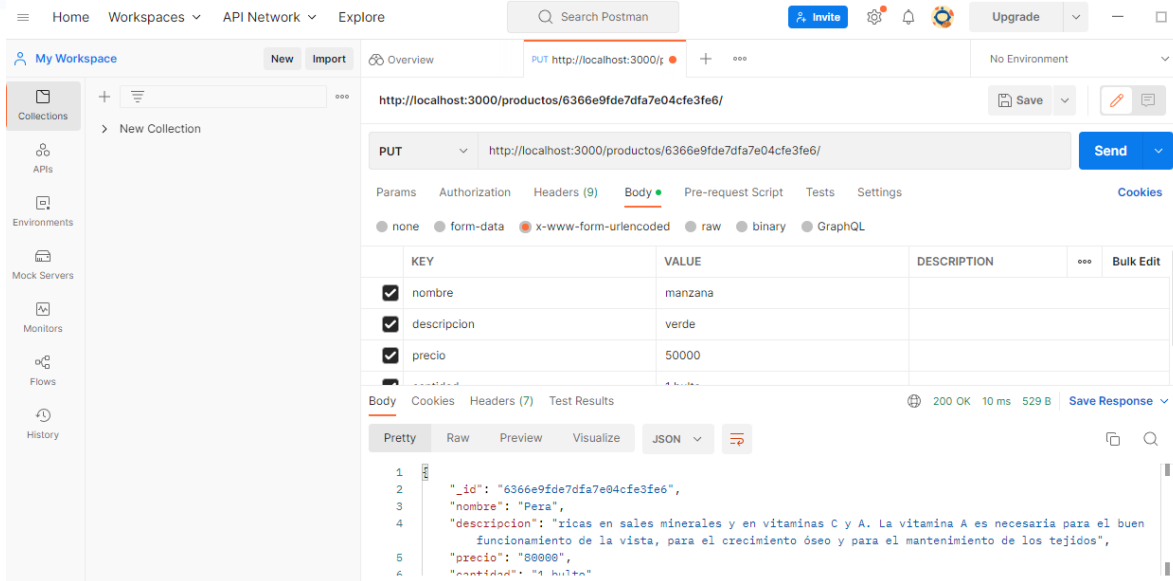
```

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 update operation is configured to update the `value` field.

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		

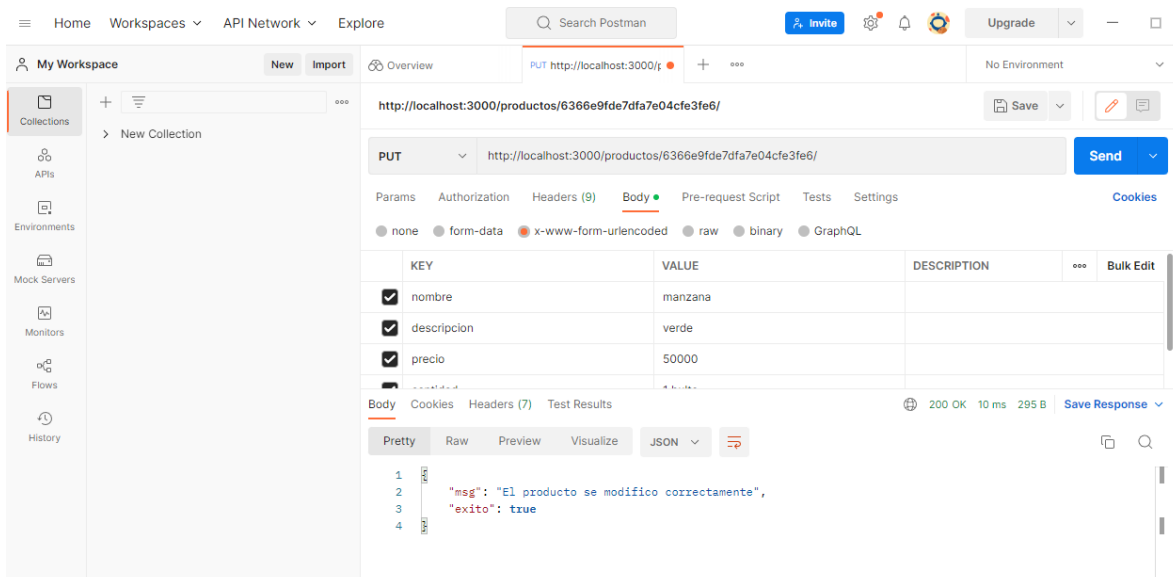
Body Cookies Headers (7) Test Results 200 OK 10 ms 529 B Save Response

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		

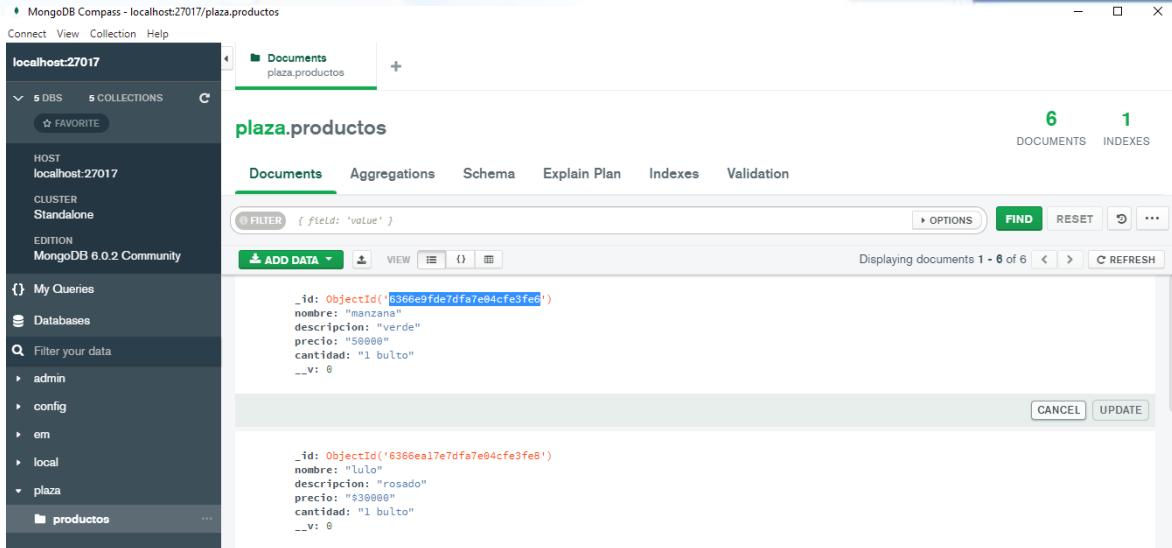
Body Cookies Headers (7) Test Results 200 OK 10 ms 295 B Save Response

Pretty Raw Preview Visualize JSON

```

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

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



9. Video: Reunión Sprint review.

<https://www.youtube.com/watch?v=9-d20gius8A>

10. Informe de Retrospectiva.

Desde la experiencia acumulada alrededor de estos meses, la cual en este ciclo nos ha permitido innovar y desarrollar nuestra creatividad como grupo. Así también de manera individual estamos aprendiendo. ha sido un constante aprendizaje junto con la práctica nos ha permitido tener una perspectiva más amplia del trabajo, que se desarrolla dentro de la plataforma.

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, En las clases vistas dentro de este ciclo se desarrolló componentes del proyecto (crear producto, editar producto, lista de producto) con su respectiva visual en frontend y en el backend.

Esta experiencia ha sido enriquecedora para todos nosotros, al involucrarnos como equipo. Hemos participado de manera diligente y constante para aportar a cada una de las etapas solicitadas, Lo aprendido a lo largo de este sprint ha sido a detalle implementando MERN la cual es un conjunto de servicios tecnológicos utilizados para construir y ejecutar una única aplicación, incluye las tecnologías Mongo DB, Express.js, React.js y Node.js la cual permite a los desarrolladores crear sitios web (y aplicaciones) completos; usando React (con JavaScript o TypeScript) del lado del cliente (front-end) y Node JS del lado del servidor (back-end). Así podremos dominar tanto la parte visual (la experiencia del usuario) como la parte algorítmica y lógica del servidor, dando como resultado un proyecto más

elaborado y detallado.

11. Historias a trabajar en el siguiente Sprint.

Se va a elaborar para el Spring 3 las historias de usuario de ingreso de sesión, registro de usuario, y se va a conectar lo elaborado con frontend y el backend.