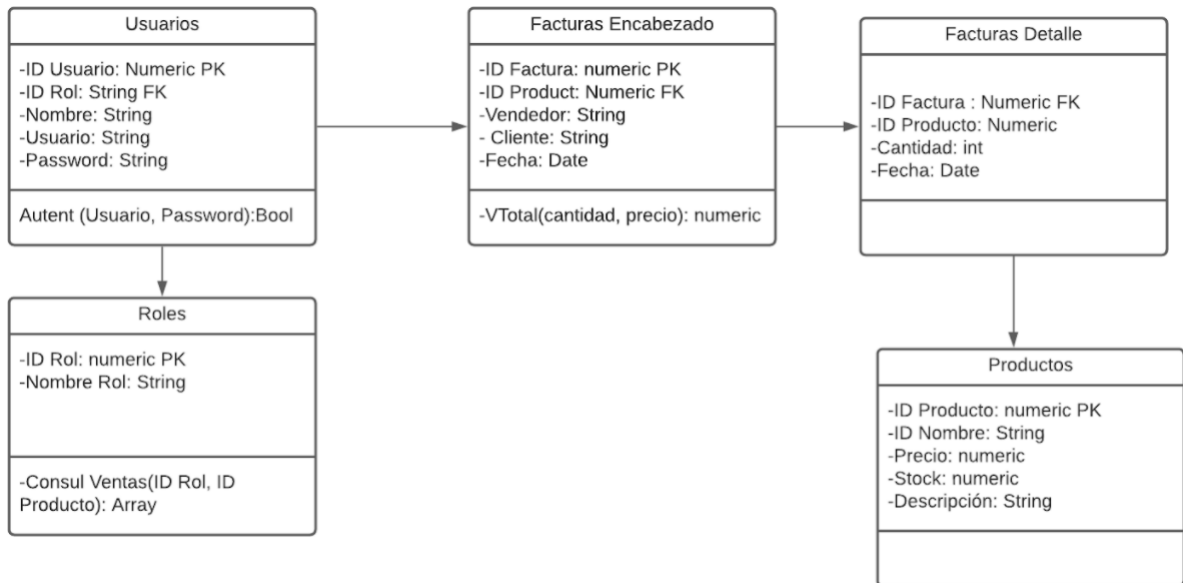


## SPRINT 3

1. Refina el diagrama de clases planteado en el sprint 2.



2. Configura la estructura básica del Backend del proyecto, usando los comandos adecuados para cada proceso

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the following structure:

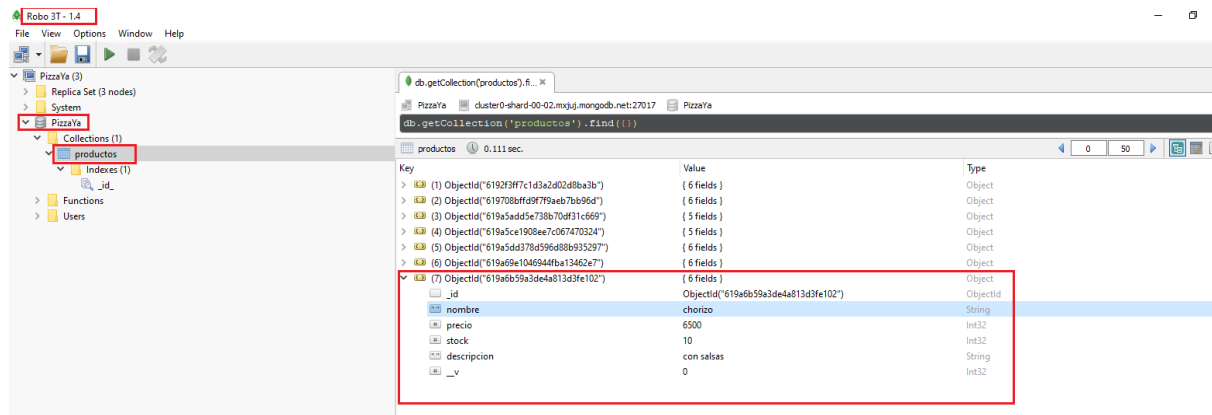
- blogify
- PizzaYa
- backend
- controllers
- posts.js
- models
- product.js
- node\_modules
- routes
- posts.js
- .gitignore
- app.js
- index.js
- package-lock.json
- package.json
- server.js
- frontend

The code editor shows the content of **product.js**:

```
1 const mongoose = require("mongoose");
2 //modelo de Productos
3 const productosSchema = mongoose.Schema({
4   nombre: { type: String, required: true },
5   precio: { type: Number, required: true },
6   stock: { type: Number, required: true },
7   descripcion: { type: String },
8 });
9
10 //exporte lo que tenemos de estructura de DB
11 module.exports = mongoose.model("Producto", productosSchema);
```

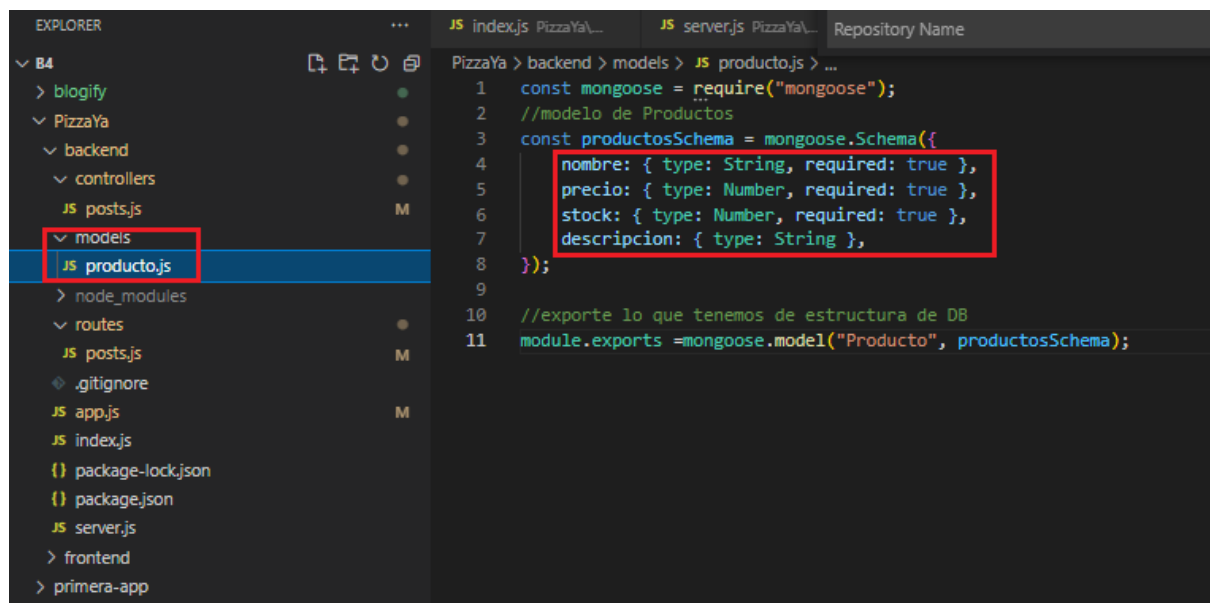
3. Instala y configura las dependencias necesarias para trabajar con MongoDB, Express, JSON, etc.

Para trabajar con MongoDB, se está utilizando Robo 3t, nuestro proyecto PizzaYa tiene una colección llamada productos:

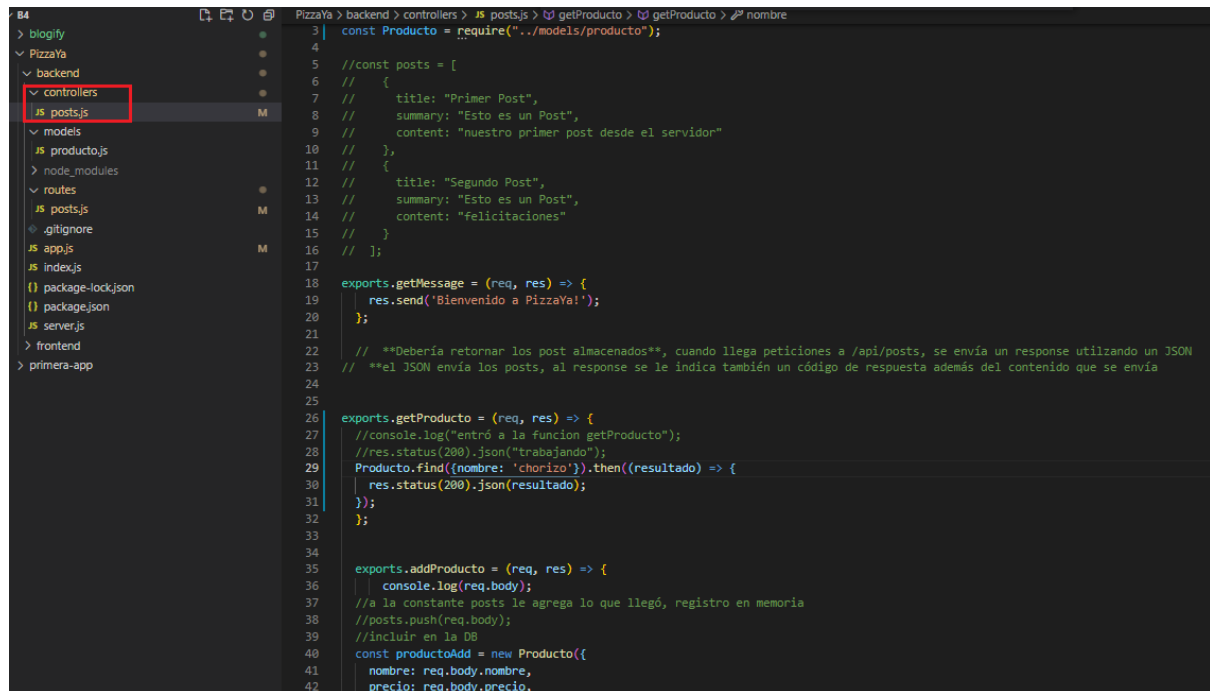


4. Implementa los Modelos y Controladores en Backend del proyecto.

Modelo:



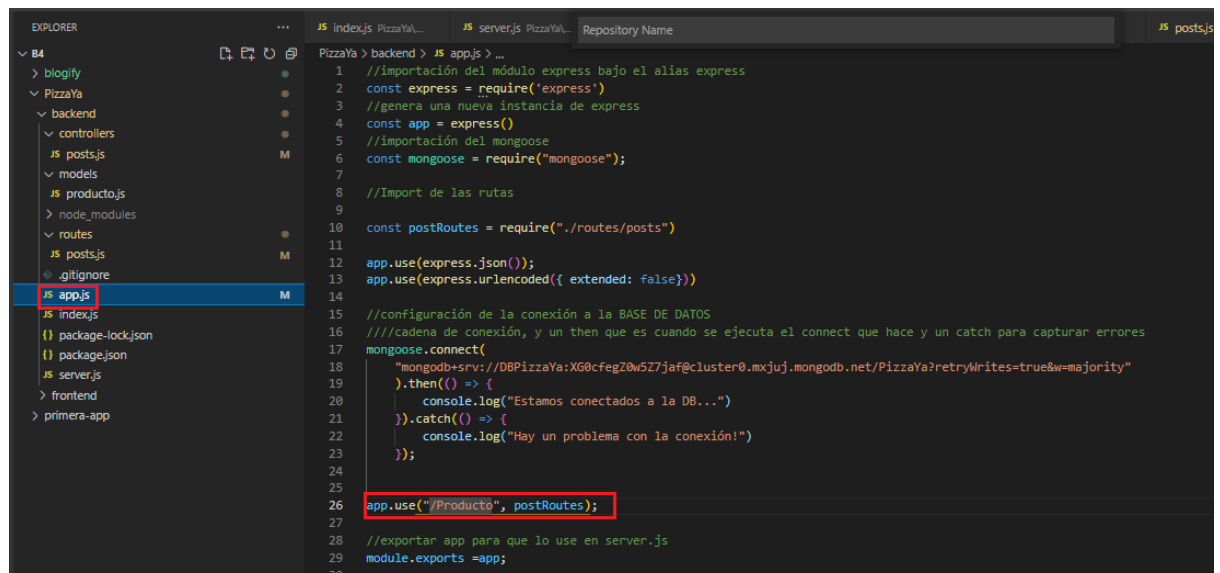
Controlador:



```
3 | const Producto = require("../models/producto");
4
5 //const posts = [
6 // {
7 //   title: "Primer Post",
8 //   summary: "Esto es un Post",
9 //   content: "nuestro primer post desde el servidor"
10 // },
11 // {
12 //   title: "Segundo Post",
13 //   summary: "Esto es un Post",
14 //   content: "felicitaciones"
15 // }
16 // ];
17
18 exports.getMessage = (req, res) => {
19   res.send("Bienvenido a PizzaYa!");
20 };
21
22 // **Debería retornar los post almacenados**, cuando llega peticiones a /api/posts, se envía un response utilizando un JSON
23 // **el JSON envía los posts, al response se le indica también un código de respuesta además del contenido que se envía
24
25
26 exports.getProductos = (req, res) => {
27   //console.log("entró a la función getProductos");
28   //res.status(200).json("trabajando");
29   Producto.find({nombre: 'chorizo'}).then((resultado) => {
30     res.status(200).json(resultado);
31   });
32 };
33
34
35 exports.addProducto = (req, res) => {
36   console.log(req.body);
37   //a la constante posts le agrega lo que llegó, registro en memoria
38   //posts.push(req.body);
39   //incluir en la DB
40   const productoAdd = new Producto({
41     nombre: req.body.nombre,
42     precio: req.body.precio,
```

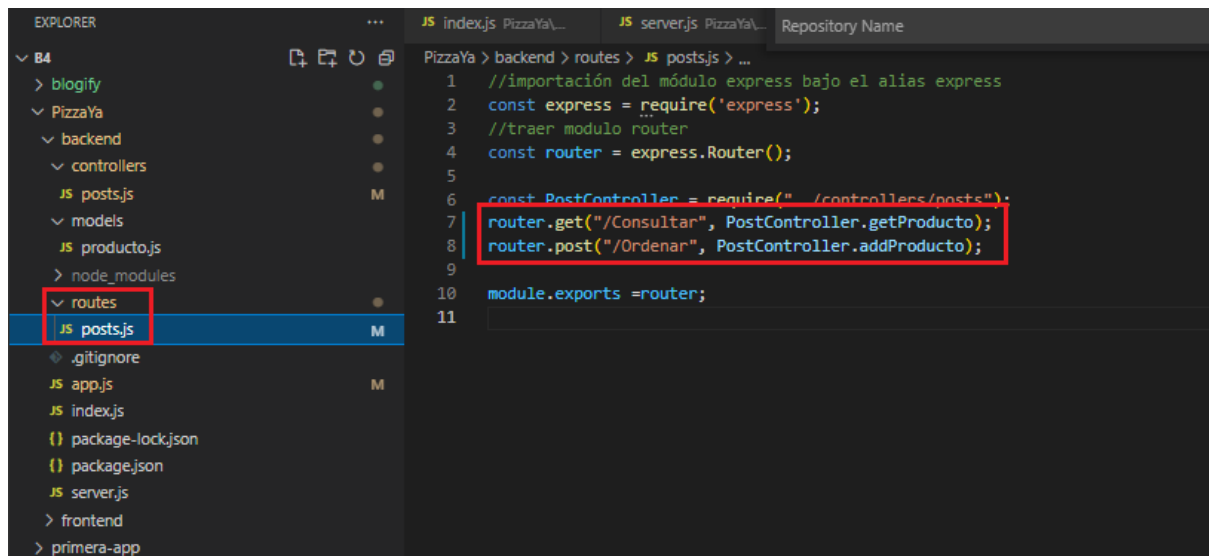
5. Crea las rutas que permitan realizar las principales tareas con la base de datos y se han identificado previamente en los wireframes como Crear, Actualizar, Eliminar y Buscar.

Ruta Raíz: /Producto



```
1 //Importación del módulo express bajo el alias express
2 const express = require('express')
3 //genera una nueva instancia de express
4 const app = express()
5 //Importación del mongoose
6 const mongoose = require("mongoose");
7
8 //Import de las rutas
9
10 const postRoutes = require("./routes/posts")
11
12 app.use(express.json());
13 app.use(express.urlencoded({ extended: false}))
14
15 //configuración de la conexión a la BASE DE DATOS
16 ///cadena de conexión, y un then que es cuando se ejecuta el connect que hace y un catch para capturar errores
17 mongoose.connect(
18   "mongodb+srv://DBPizzaYa:YG0cfeg20w5Z7jaf@cluster0.mongoddb.net/PizzaYa?retryWrites=true&w=majority"
19 ).then(() => {
20   console.log("Estamos conectados a la DB...")
21 }).catch(() => {
22   console.log("Hay un problema con la conexión!")
23 });
24
25
26 app.use("/Producto", postRoutes);
27
28 //exportar app para que lo use en server.js
29 module.exports = app;
30
```

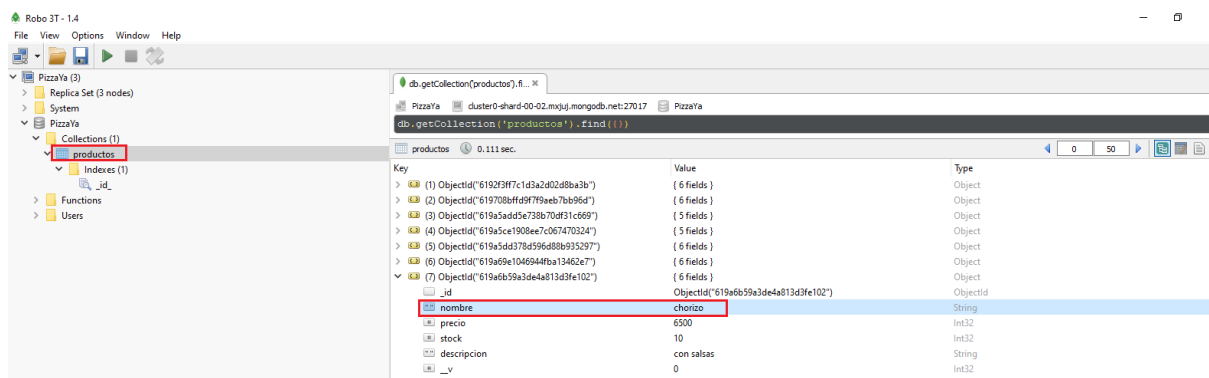
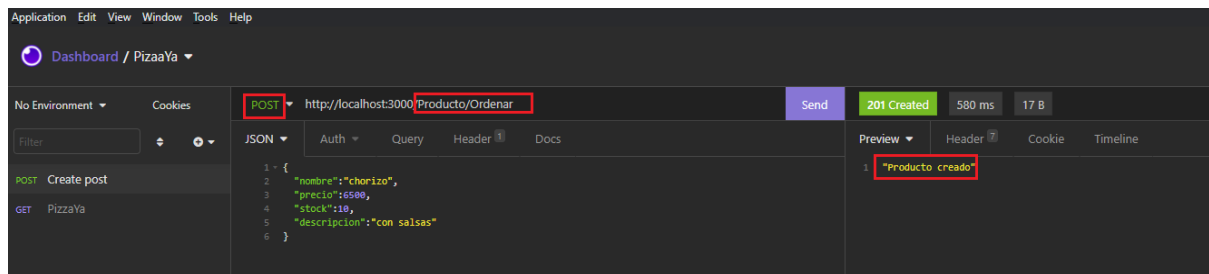
Para crear un producto la ruta es: /Producto/Ordenar y para generar una consulta para facturas la ruta es: /Producto/Consultar:



```
1 //importación del módulo express bajo el alias express
2 const express = require('express');
3 //traer modulo router
4 const router = express.Router();
5
6 const PostController = require("../controllers/posts");
7 router.get("/Consultar", PostController.getProducto);
8 router.post("/Ordenar", PostController.addProducto);
9
10 module.exports = router;
11
```

6. Realiza pruebas de las rutas usando Insomnia o Postman.

Se crea producto en la base de datos, llamado chorizo, en la siguiente imagen se observa el post enviado con la estructura de json y con Robo 3t se puede visualizar que el producto se creó:



Para consultar, se envía un get vía Insomnia donde se trae el resultado de una consulta a la base de datos para el campo nombre, específicamente “chorizo”:

