

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Análisis y Diseño 1

Sección N

Ing. Jose Manuel Ruiz Juarez

Aux. César Dionicio Sazo Mayen

**Proyecto 2**  
**Manual Tecnico**

All for one - Grupo No. 7	
Nombre	Carnet
Erick Fernando Sánchez Mejía	201503878
Melyza Alejandra Rodriguez Contreras	201314821
Helmut Efraín Najarro Álvarez	201712350
Christopher Alexander Acajabon Gudiel	201404278

# Objetivo general

Desarrollar una aplicación utilizando una arquitectura de N capas, que cumpla con las expectativas de una tienda en línea. Todo esto aplicando los principios de integración y entrega continua con la ayuda de herramientas que brindan la automatización de estos procesos, como lo es Jenkins.

Al mismo tiempo trabajar bajo un marco de trabajo ágil para una gestión flexible y que soporte cambios.

## Objetivos específicos

Utilizar docker como herramienta de contenerización para el desarrollo y ejecución en diferentes entornos.

Utilizar Git como controlador de versiones para el desarrollo en conjunto, guiándose por el flujo de trabajo que representa Gitflow.

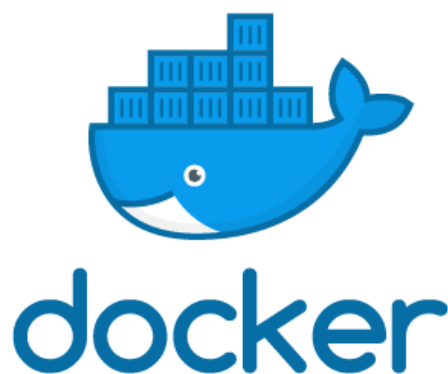
Utilizar Jenkins como herramienta de automatización para la implementación de CI/CD.

## Tecnologías utilizadas

### Docker

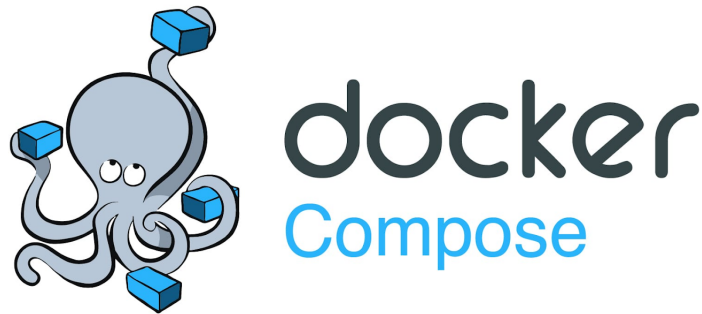
Es una plataforma abierta de desarrollo, envío, y ejecución de aplicaciones. Docker te permite separar tus aplicaciones de tu infraestructura de trabajo, para así entregar el software de forma rápida. Con Docker se puede manejar la infraestructura de la misma forma que se manejan las aplicaciones. Tomando ventaja de la metodología de envío, prueba, y despliegue de código de forma rápida de Docker, se puede significativamente reducir el retraso entre escritura de código y ejecución en producción.

Docker provee la capacidad de empaquetar y ejecutar aplicaciones en un entorno vagamente aislado llamado contenedor. La aislación y seguridad permite ejecutar muchos contenedores simultáneamente en un host dado. Los contenedores son ligeros y contienen lo necesario para ejecutar la aplicación.



# Docker compose

Compose es una herramienta para definir y ejecutar un multi contenedor de aplicaciones de Docker. Con compose se utiliza un archivo YAML para configurar los servicios de las aplicaciones. Luego con un comando, se crea e inician todos los servicios de la configuración.



# Node.js

Diseñado para crear aplicaciones de red escalables, utilizando un entorno de ejecución de JavaScript orientado a eventos asincrónicos.

Node.js lleva el modelo de eventos un poco más allá. Incluye un bucle de eventos como runtime de ejecución en lugar de una biblioteca. En otros sistemas siempre existe una llamada de bloqueo para iniciar el bucle de eventos. Por lo general, el comportamiento se define mediante devoluciones callbacks de llamada al iniciarse un script y al final se inicia un servidor a través de una llamada de bloqueo. Node.js simplemente entra en el bucle de eventos después de ejecutar el script de entrada y sale cuando no hay más devoluciones callbacks de llamada para realizar.



# TypeScript

TypeScript es un lenguaje de código abierto que se basa en JavaScript, una de las herramientas más utilizadas del mundo, al agregar definiciones de tipos estáticos.

TypeScript extiende la sintaxis de JavaScript, por tanto cualquier código JavaScript existente debería funcionar sin problemas. Está pensado para grandes proyectos, los cuales a través de un compilador de TypeScript se traducen a código JavaScript original.

TypeScript soporta ficheros de definición que contengan información sobre los tipos de librerías JavaScript existentes, similares a los ficheros de cabeceras de C/C++ que describen la estructura de ficheros de objetos existentes.



# React JS

React es una librería de JavaScript declarativa, eficiente y flexible para construir interfaces de usuario. Permite componer IUs complejas de pequeñas y aisladas piezas de código llamadas “componentes”.



# Máquinas virtuales, Google Cloud

Google Cloud Platform se trata de la suite de infraestructuras y servicios que Google utiliza a nivel interno y, ahora, disponible para cualquier empresa, de tal forma que sea aplicable a multitud de procesos empresariales.

Cuando hablamos de Google Cloud Platform (GCP), estamos ante todas las herramientas de Google disponibles en la nube que hasta ahora se ofrecían por separado. Este conjunto de servicios ofrecen prestaciones muy dispares; desde machine learning hasta Inteligencia artificial pasando por el big data, todo englobado bajo el paraguas del cloud computing.

Una máquina virtual es la creación a través de software de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento etc.



## MySQL

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo y una de las más populares en general junto a Oracle y Microsoft SQL Server, todo para entornos de desarrollo web.



# MySQL Workbench

MySQL Workbench es el entorno integrado oficial de MySQL, fue desarrollado por MySQL AB, y permite a los usuarios administrar gráficamente las bases de datos MySQL y diseñar visualmente las estructuras de las bases de datos, MySQL Workbench reemplaza el anterior paquete de software, MySQL GUI Tools.



## Git

Git es un controlador de versiones es un sistema que graba cambios a un archivo o conjunto de archivos a través del tiempo, permitiendo hacer llamadas a versiones específicas. Esto funciona gracias a que el sistema de control de versiones tiene una simple base de datos que mantiene todos los cambios en archivos bajo una versión de control.



## Git flow

Es un flujo de trabajo de Git. El flujo de GitFlow define un modelo estricto de ramificación diseñado alrededor de la publicación del proyecto. Proporciona un marco sólido para gestionar proyectos más grandes.

En realidad, es una especie de idea abstracta de un flujo de trabajo de Git. Quiere decir que ordena qué tipo de ramas se deben configurar y cómo fusionarlas.

# Justificación de tecnologías utilizadas

## Docker

- Se puede programar localmente y compartir el trabajo con los compañeros, usando contenedores.
- Impulsar aplicaciones a un entorno de prueba y ejecutarlas de forma manual y automáticas
- Al encontrar errores, pueden ser corregidos en el entorno de desarrollo para volver a implementarlos.

## Docker compose

- Múltiples entornos aislados en un solo host.
- Mantiene el volumen de datos cuando los contenedores son creados, al momento de ejecutar docker-compose up, si encuentra algún contenedor de alguna ejecución previa, copia el volumen del contenedor antiguo al nuevo contenedor. Este proceso asegura que cualquier dato que se haya creado en volumen no esté perdido.
- Soporta variables en el archivo compose. Se puede utilizar estas variables para personalizar la composición para diferentes entornos, o diferentes usuarios.
- Documentar y configurar todos los servicios de dependencias de la aplicación (bases de datos, colas, caches, APIs, etc).

## Node.js

- Utiliza HTTP, diseñado teniendo en cuenta la transmisión de operaciones con streaming y baja latencia. Lo cual hace que Node.js sea muy adecuado para la base de una librería o un framework web.
- Orientado a eventos asíncronos.
- Incluye un bucle de eventos como runtime de ejecución en lugar de una biblioteca.

## TypeScript

- Es fácil de aprender, y todo el equipo puedo ponerse al día con él en poco tiempo.

- Mejora la ayuda contextual mientras escribes código, ya que es más rico en información que JavaScript, en editores que lo soportan, notarás una gran diferencia y disminuirá los errores que cometes tú y tu equipo.
- Permite crear código estandarizado en todo el equipo de trabajo, dejando mucho menos margen para los problemas y para hacerlo crecer en el futuro a medida que haya mejor soporte nativo en los navegadores para ciertas características.
- Funciona bien con las librerías y frameworks de Front-End que estés utilizando. Algunas, como Angular, Ember o Aurelia le sacan especial partido, e incluso el uso combinado de TypeScript + React se está convirtiendo en norma.

## React JS

- Flexible.
- Dinámico.
- Ligero.
- Reutilización de código.

## Google cloud, Máquinas virtuales

- Precios accesibles: Una de las grandes ventajas de GCP, es que se cobra solamente el tiempo que se utiliza, y además es posible conseguir descuentos.
- Rendimiento mejorado: Tener una gran capacidad es primordial al momento de tener una página web, y las máquinas de Google nos pueden proveer esta capacidad demandada.
- Excelente seguridad: El modelo de seguridad de Google tuvo 15 años de desarrollo, lo que, aunado a los 500 expertos que se encargan de supervisar esta área, nos proporciona una protección de primera. La seguridad de Google funciona de extremo a extremo, con una codificación bastante compleja que mantendrá nuestros datos a salvo.
- Rápida restauración de datos: En ocasiones es necesario restaurar algún archivo o dato, ya sea porque se eliminó por equivocación o porque no se consideraba necesario. Google permite la restauración de datos en un tiempo extremadamente reducido: menos de un segundo. Esto es una gran ventaja comparado con otros proveedores, los cuales pueden tardar hasta horas en completar estas tareas y por una tarifa más alta que la que Google ofrece.



# MySQL

- Es una base de datos gratuita, al ser de código abierto, no tiene coste, con el ahorro que eso conlleva.
- Es muy fácil de usar, podemos empezar a usar la base de datos MySQL sabiendo unos pocos comandos.
- Es una base de datos muy rápida, su rendimiento es estupendo sin añadirle ninguna funcionalidad avanzada.
- Utiliza varias capas de seguridad, contraseñas encriptadas, derechos de acceso y privilegios para los usuarios.
- Pocos requerimientos y eficiencia de memoria, tiene una baja fuga de memoria y necesita pocos recursos de CPU o RAM.

# MySQL Workbench

- Multiplataforma: Windows.
- Permite el manejo de archivos.sql
- Desarrollar diagramas ER
- Software libre, distribuido bajo licencia GPL
- Permite crear script a partir del modelo creado y viceversa.

# Git

- Permite tener múltiples ramas locales que pueden ser completamente independientes entre sí. La creación, fusión y eliminación de esas líneas de desarrollo toma segundos.
- Las operaciones se realizan localmente, lo que le da una gran ventaja de velocidad en sistemas centralizados que constantemente tienen que comunicarse con un servidor en algún lugar.
- Garantiza la integridad criptográfica de cada parte de su proyecto. Cada archivo y confirmación se suma de verificación y se recupera mediante su suma de verificación cuando se vuelve a verificar.

## Git flow

- Este flujo de trabajo es ideal para los flujos de trabajo de software basados en publicaciones
- Ofrece un canal específico para las correcciones de producción.
- Es ideal para los proyectos que tienen un ciclo de publicación programado.
- Asigna funciones muy específicas a las diferentes ramas y define cómo y cuándo deben interactuar.

# Requerimientos

## Requerimientos funcionales

1. Login para ingresar al sistema.
2. Login para poder agregar productos al carrito.
3. El sistema debe de contar con un apartado de cerrar sesión.
4. Registro de usuarios al sistema.
5. Listar todos los productos.
6. Listar productos por categoría.
7. Registrar venta.
8. El sistema enviará correo electrónico con la factura cuando se haga una compra.
9. Agregar productos al carrito de compras.
10. Ver carrito de compras.
11. Eliminar un producto del carrito.
12. Eliminar todos los productos del carrito.
13. Actualizar la cantidad de productos del carrito.

## Requerimientos no funcionales

1. Aplicación web responsive.
2. El correo electrónico del usuario no debe repetirse entre usuarios.
3. Se guardará la URL de las imágenes en la base de datos.
4. El sistema contará con un paginado para la parte de los productos.
5. El sistema actualizará el stock de los productos automáticamente.
6. Las fechas son generadas automáticamente basándose en la zona horaria de la red.
7. En la pantalla login, notificaciones al intentar loguearse con campos vacíos.
8. En la pantalla login, notificaciones al fallar en el logueo.
9. Validación de campos vacíos en módulo de registro.

# Requisitos

## Hardware

### Docker:

### Backend de WSL 2:

- Windows 10 de 64 bits: Home, Pro, Enterprise o Education, versión 1903 (compilación 18362 o superior).
- Habilite la función WSL 2 en Windows, para obtener instrucciones detalladas consulte <https://docs.microsoft.com/en-us/windows/wsl/install-win10>
- Se requieren los siguientes requisitos previos de hardware para ejecutar correctamente WSL 2 en Windows 10:

- Procesador de 64 bits.
- RAM del sistema de 4GB
- El soporte de virtualización de hardware a nivel de BIOS debe estar habilitado en la configuración del BIOS.

#### Backend de Hyper-V y contenedores de Windows:

- Windows 10 de 64 bits: Pro, Enterprise o Education (compilación 17134 o superior).
- Las características de Hyper-V y contenedores de Windows deben estar habilitadas.
- Se requieren los siguientes requisitos previos de hardware para ejecutar correctamente Client Hyper-V en Windows 10:
  - Procesador de 64 bits.
  - RAM del sistema de 4GB
  - El soporte de virtualización de hardware a nivel de BIOS debe estar habilitado en la configuración del BIOS

#### **Node.js:**

Node.js puede funcionar en la mayoría de las computadoras de la actualidad, ya que se ha visto funcionando en dispositivos con 256 o 128 MB de RAM y con tan solo un núcleo en el procesador.

#### **TypeScript:**

Al igual que Node.js, TypeScript puede funcionar en la mayoría de las computadoras de la actualidad.

#### **MySQL:**

- 512 Mb de memoria Ram
- 1 GB de espacio de disco duro
- Arquitectura del sistema 32/64 bit

#### **MySQL Workbench:**

- Procesador doble núcleo de 2 Ghz (4 núcleos recomendado)
- 4 gigabytes de RAM (6 gigabytes recomendado).
- Pantalla con una resolución mínima de 1024×768 píxeles (1280×1024 recomendado).

#### **Visual studio code:**

Es una pequeña descarga (<100 MB) y ocupa un espacio en disco de 200 MB, VS Code es liviano y debería ejecutarse fácilmente en el hardware actual.

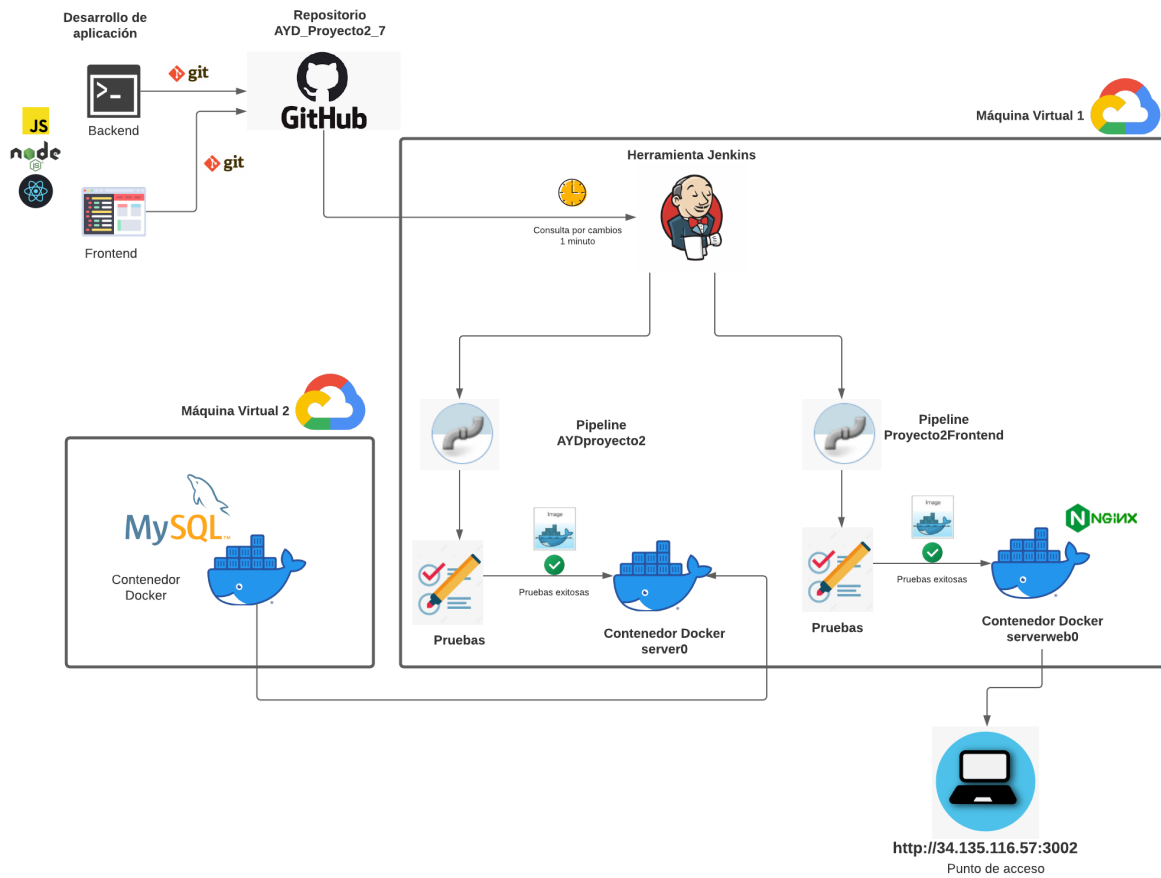
Se recomienda:

- Procesador de 1,6 GHz o más rápido
- 1 GB de RAM

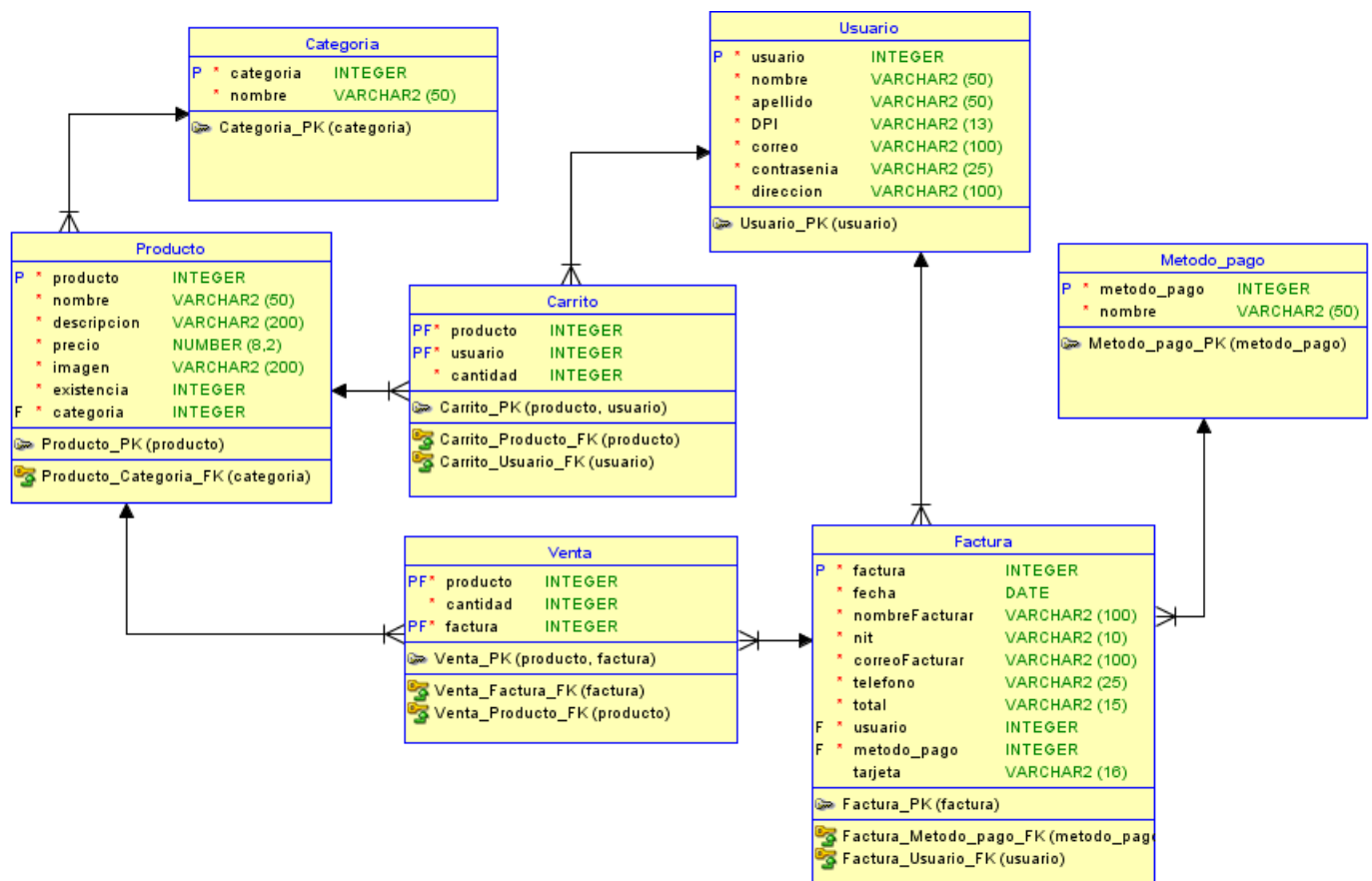
## Software

Sistema operativo	Windows 10
Docker	versión 20.10.6
Docker-compose	versión 1.29.1
Node.js	versión 14.16.1
TypeScript	version 4.2.4
MySQL	version 8.0.23 for Win64 on x86_64 (MySQL Community Server - GPL)
MySQL Workbench	versión 8.0
Visual Studio Code	versión 1.56.2
Git	versión 2.31.1
Git flow	versión 1.12.3
Kernel de Linux para Windows (necesario para Docker desktop)	Puede descargar en <a href="https://docs.microsoft.com/en-us/windows/wsl/install-win10">https://docs.microsoft.com/en-us/windows/wsl/install-win10</a> (paso 4)
Google Chrome	Versión 72 o una más reciente (puede utilizar su navegador de preferencia)

# Arquitectura del sistema



## Diagrama Entidad-Relación



# Modelo Físico

```
CREATE DATABASE marketplace;
```

```
USE marketplace;
```

```
CREATE TABLE categoria (
    categoria    INTEGER          NOT NULL    AUTO_INCREMENT ,
    nombre       VARCHAR(50)      NOT NULL
    PRIMARY KEY (categoria)
);
```

```
CREATE TABLE producto (
    producto     INTEGER          NOT NULL    AUTO_INCREMENT ,
    nombre       VARCHAR(50)      NOT NULL
    descripcion  VARCHAR(200)     NOT NULL
    precio       DECIMAL(8, 2)    NOT NULL
    imagen       VARCHAR(600)     NOT NULL
    existencia   INTEGER          NOT NULL
    categoria    INTEGER          NOT NULL
    PRIMARY KEY (producto)
```

```

        FOREIGN KEY (categoria)      REFERENCES categoria (categoria)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    );

CREATE TABLE metodo_pago (
    metodo_pago INTEGER      AUTO_INCREMENT ,
    nombre       VARCHAR(50)  NOT NULL      ,
    PRIMARY KEY (metodo_pago)
);

CREATE TABLE usuario (
    usuario      INTEGER      NOT NULL      AUTO_INCREMENT ,
    nombre       VARCHAR(50)  NOT NULL      ,
    apellido     VARCHAR(50)  NOT NULL      ,
    dpi          VARCHAR(13)  NOT NULL      ,
    correo       VARCHAR(100) NOT NULL      UNIQUE          ,
    contrasenia  VARCHAR(25)  NOT NULL      ,
    direccion    VARCHAR(100) NOT NULL      ,
    PRIMARY KEY (usuario)
);

CREATE TABLE carrito (
    producto     INTEGER NOT NULL      ,
    usuario      INTEGER NOT NULL      ,
    cantidad     INTEGER NOT NULL      ,
    PRIMARY KEY (producto, usuario) ,
    FOREIGN KEY (producto) REFERENCES producto (producto) ON
DELETE CASCADE ON UPDATE CASCADE ,
    FOREIGN KEY (usuario)  REFERENCES usuario  (usuario)  ON
DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE factura (
    factura      INTEGER      NOT NULL      AUTO_INCREMENT ,
    fecha        DATE         NOT NULL      ,
    nombrefacturar VARCHAR(100) NOT NULL      ,
    ,
    nit          VARCHAR(10)   NOT NULL      ,
    ,
    correofacturar VARCHAR(100) NOT NULL      ,
    telefono     VARCHAR(25)   NOT NULL      ,
    ,
    total        VARCHAR(15)   NOT NULL      ,
    ,
    usuario      INTEGER      NOT NULL      ,
    metodo_pago  INTEGER      NOT NULL      ,
    tarjeta     VARCHAR(16)   ,

```

```

        PRIMARY KEY (factura)
FOREIGN KEY (usuario) REFERENCES usuario
(usuario) ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (metodo_pago) REFERENCES metodo_pago
(metodo_pago) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE venta (
    producto    INTEGER NOT NULL
    ,
    factura     INTEGER NOT NULL
    ,
    cantidad    INTEGER NOT NULL
    ,
    PRIMARY KEY (producto, factura)
    ,
    FOREIGN KEY (factura) REFERENCES factura (factura) ON
DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (producto) REFERENCES producto (producto) ON
DELETE CASCADE ON UPDATE CASCADE
);

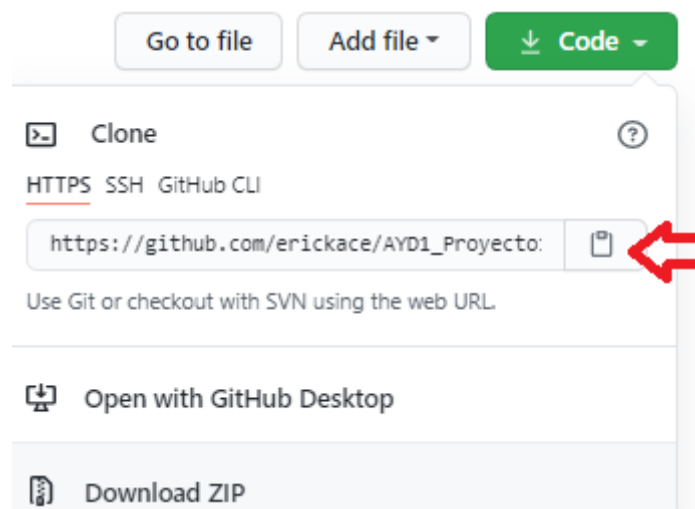
```

# Herramientas de desarrollo (instalación de tecnologías)

## Descargar proyecto

### Forma 1: Clonar repositorio

1. Ve a Github y obtén la ruta del repositorio:

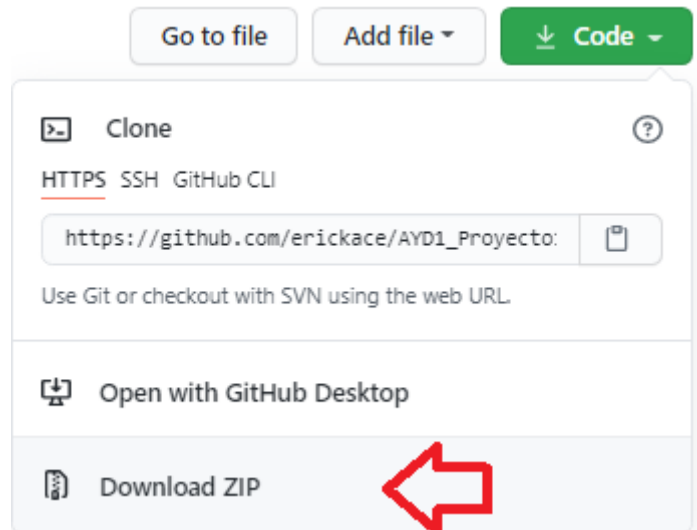




2. Crea el directorio donde quieras meter el proyecto.
3. Acceder al directorio desde tu consola y escribir:  
**git clone https://github.com/erickace/AYD1\_Proyecto1\_7.git**
4. Listo, ya tienes el proyecto clonado en tu ordenador.

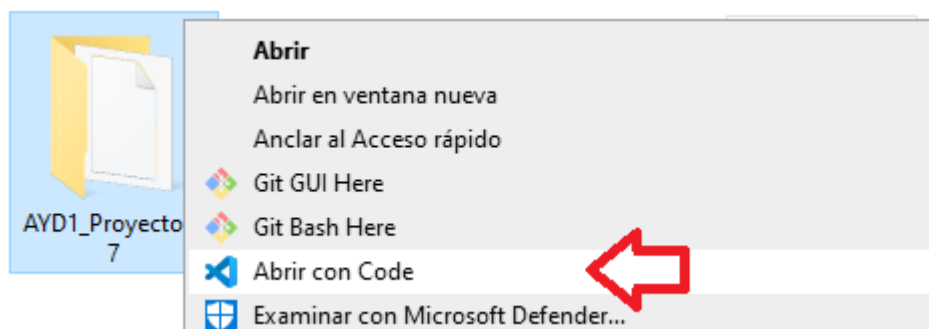
## Forma 2: Descargar ZIP

1. Descargar el fichero.



2. Mover el zip al directorio que desees.
3. Descomprimir el fichero.
4. Listo, ya tienes el proyecto clonado en tu ordenador.

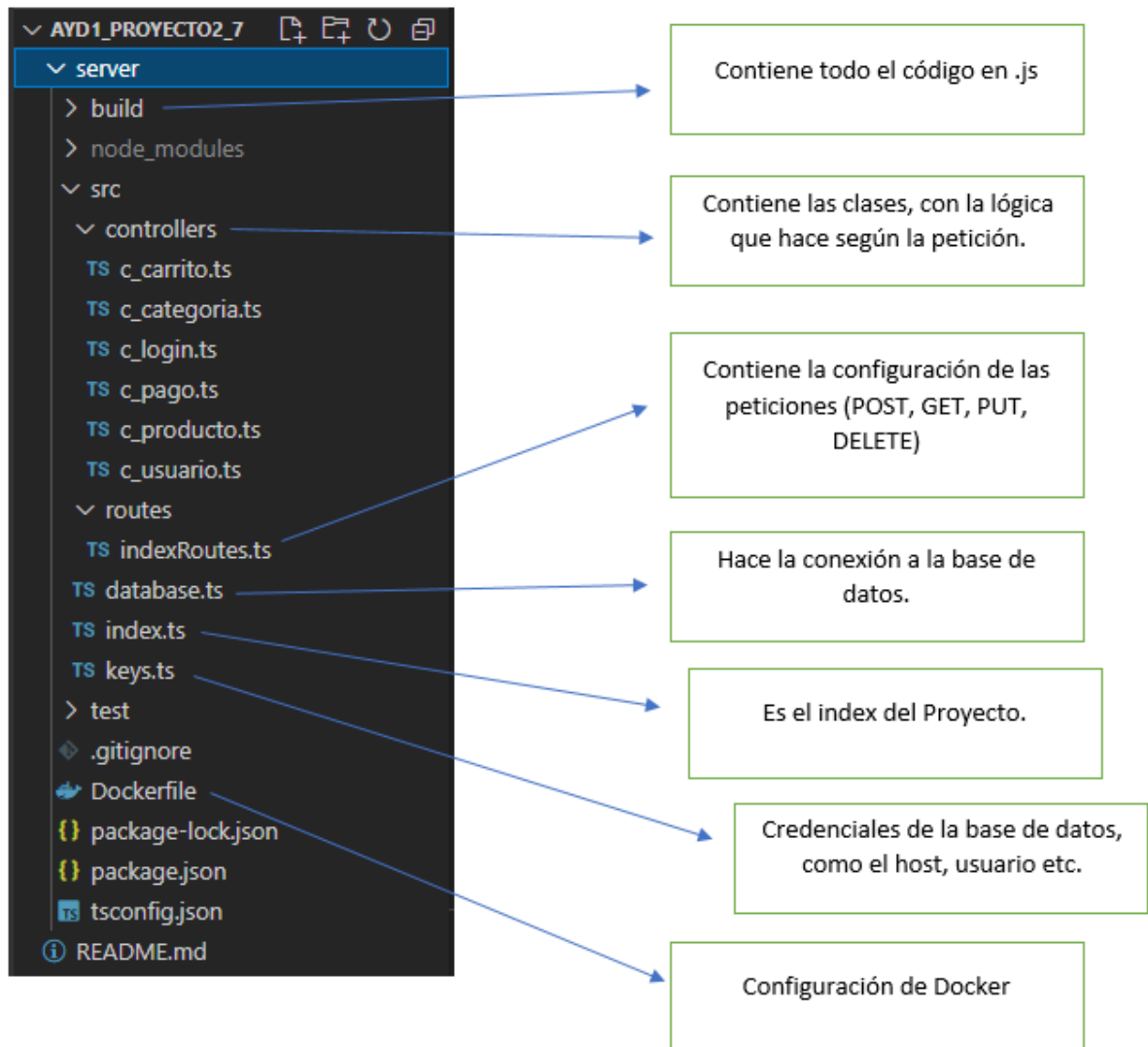
## Abrir proyecto con Visual Studio Code



# Explicación código

## Backend

### ESTRUCTURA DE CÓDIGO:



Index.ts:

- Importaciones:

```
import express, {Application} from 'express';

import morgan from 'morgan';

import cors from 'cors';

import indexRoutes from './routes/indexRoutes';
```

- Configuración:

Contiene la configuración como la de express, cors, morgan y el puerto.

```
8  class Server{
9
10     public app: Application;
11
12     //se ejecuta al instanciar la clase, y devolvera objeto tipo express
13     constructor(){
14         this.app = express();
15         this.config();
16         this.routes();
17     }
18
19     //encargado de configurar la variable app
20     config(): void {
21         this.app.set('port',process.env.PORT || 3000);
22         this.app.use(morgan('dev'));
23         this.app.use(cors());
24         this.app.use(express.json());
25         this.app.use(express.urlencoded({extended:false})); // para enviar desde un formulario html
26     }
27
28     //para definir de app las rutas de nuestro servidor
29     routes(): void {
30         this.app.use('/',indexRoutes);
31     }
32
33     //para poder inicializar el servidor
34     start(): void {
35         const serverWeb=this.app.listen(this.app.get('port'), "0.0.0.0", () => {
36             console.log("Ejecutando Server en port",this.app.get('port'));
37         });
38     }
39
40 }
41
42 const server = new Server();
43 server.start();
```

## keys.ts

- Contiene las credenciales de la base de datos, la cual se usa en el archivo database.ts

```
TS keys.ts X
server > src > TS keys.ts > [⌘] default
1  export default{
2      database:{
3          host: '34.132.46.67',
4          port: 23306,
5          user: 'root',
6          password: 'grupo7',
7          database: 'marketplace'
8      }
9  }
```

## database.ts

- Hace la conexión con la base de datos.

```
TS database.ts X
server > src > TS database.ts > [⌘] default
1  import mysql from 'promise-mysql';
2
3  import keys from './keys';
4
5  const pool = mysql.createPool(keys.database);
6
7  pool.getConnection().then(connection => {
8      pool.releaseConnection(connection);
9      console.log('DB MySQL is connect');
10 });
11
12 export default pool;
```

## indexRoutes.ts

Contiene toda la configuración de las peticiones tipo POST, GET, PUT, DELETE.

- Se hace un import de la clase el cual contiene la función a utilizar:

```
import {c_login} from '../controllers/c_login';
```

- Se le asigna un tipo de petición, la ruta y se llama al método:

```
this.router.post('/login', c_login.loguear);
```

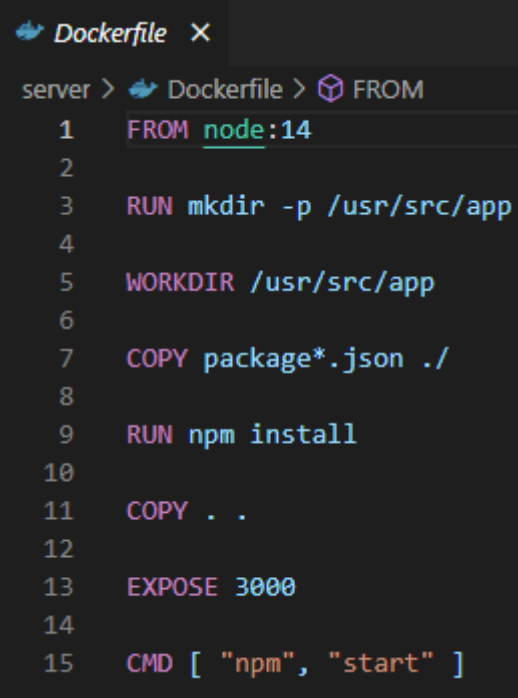
## Carpeta controllers

Contiene todas las clases, en la cual están las funciones a utilizar, por ejemplo la de Login:

- Tiene el import con la conexión a la base de datos.
- Contiene la función de loguear, la cual se encarga de verificar si un usuario existe en la base de datos.
- Y lo mismo pasa con las demás clases, contiene las funciones según lo que se quiera hacer, como obtener datos, guardar, actualizar y eliminar.

```
TS c_login.ts X
server > src > controllers > TS c_login.ts > ...
1  import { Request, Response } from 'express';
2
3  import pool from '../database';
4
5  class C_login{
6      public async loguear(req:Request,res:Response){
7          await pool.query(`SELECT
8                          *
9                          FROM
10                         usuario
11                         WHERE
12                             correo = '${req.body.correo}'
13                             AND contrasenia = '${req.body.contrasenia}'`, (err: any, rows: any, fields: any) => {
14
15              if (!err) {
16                  if(rows.length > 0){
17                      res.status(200).send(rows[0]);
18                  }else{
19                      res.status(500).send('false');
20                  }
21              } else {
22                  res.status(500).send('error');
23              }
24          });
25      }
26  }
27  export const c_login = new C_login();
```

## Dockerfile

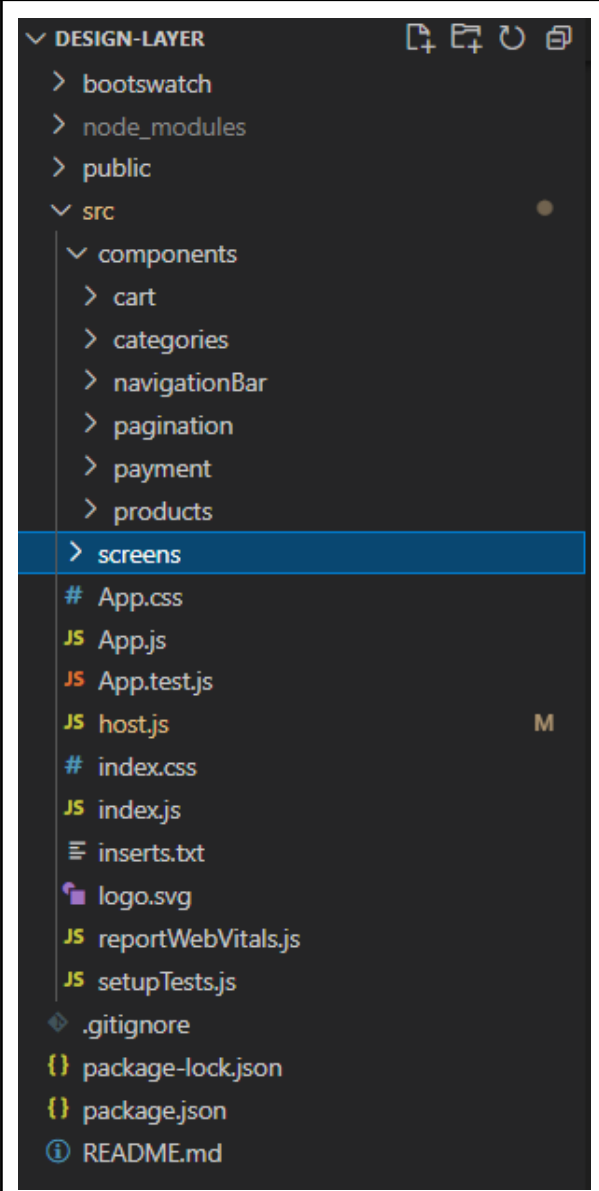


```
server > Dockerfile > FROM
1 FROM node:14
2
3 RUN mkdir -p /usr/src/app
4
5 WORKDIR /usr/src/app
6
7 COPY package*.json ./
8
9 RUN npm install
10
11 COPY . .
12
13 EXPOSE 3000
14
15 CMD [ "npm", "start" ]
```

1. **FROM:** obtengo la imagen de node
2. **RUN:** creo una carpeta
3. **WORKDIR:** accedo a esa carpeta
4. **COPY:** Se copian todos los archivos .json
5. **RUN:** Se instalan todas las dependencias
6. **COPY:** Se copia todo el código en la carpeta creada
7. **EXPOSE:** Se expone el puerto 3000
8. **CMD:** Se ejecuta comando para levantar el servidor.

# Frontend

Estructura del proyecto:

	<ol style="list-style-type: none"><li>1. <b>bootstrap:</b> esta carpeta contiene archivos de css que cambian el aspecto del frontend.</li><li>2. <b>node_modules:</b> esta carpeta contiene las dependencias necesarias para que el proyecto se ejecute correctamente.</li><li>3. <b>public:</b> esta carpeta contiene imágenes, iconos además del index.html</li><li>4. <b>src</b><ol style="list-style-type: none"><li>a. <b>componentes:</b> contiene todos los componentes creados para una mejor reutilización de código.</li><li>b. <b>screens:</b> contiene componentes que son usados por react-router-dom para la renderización de páginas.</li></ol></li><li>5. <b>host.js:</b> contiene una constante que es la dirección en donde está alojado el backend.</li><li>6. <b>App.js:</b> contiene el componente que es exportado para mostrar las páginas.</li></ol>
--	--

src/components

cart/cardItemCard.js

Nombre	tipo	params	Propósito
deleteItem	void	-	Eliminar un artículo dentro del carrito de compras

```
deleteItem() {  
  fetch(HOST+'eliminarProductoCarrito', {  
    method: 'DELETE', // or 'PUT'  
    body: JSON.stringify({  
      producto:parseInt(this.props.id),  
      usuario:parseInt(this.props.user)}), // data can be `string` or {object}!  
    headers:{  
      'Content-Type': 'application/json'  
    }  
  }).then(res => res.json())  
  .catch(error => alert('Articulo no eliminado.'))  
  .then(response => {  
    if(response){  
      window.location.replace('/cart')  
    }  
  });  
}
```

Nombre	tipo	params	Propósito
editQuantity	async	e: Evento	Editar la cantidad de algun producto dentro del carrito



```

async editQuantity(e){
  if(e.target.value>=0){
    this.setState({changing:true, quantity:parseInt(e.target.value)})
    await axios.post(HOST+'actualizarCarrito',{
      producto:parseInt(this.props.id),
      usuario: parseInt(this.props.user),
      cantidad:parseInt(e.target.value)
    }).then((res)=>{
      this.setState({changing:false})
    }).catch((err)=>{
      console.log(err)
    })
  }
}
}

```

payment/creditCard.js

Nombre	tipo	params	Propósito
confirmPurchase	async	-	Registrar una compra. Comprar todos los productos dentro del carrito, y enviar un correo de confirmacion.

```

async confirmPurchase() {

  this.setState({sending:true})

  await axios.post(HOST + 'pagarAhora', {
    usuario: this.props.user,
    nombre: this.state.name,
    nit: this.state.nit,
    telefono: this.state.phone,
    metodo_pago: this.state.payment,
    tarjeta: this.state.card,
    total: Math.round(this.props.total),
    correo: this.state.email
  })

  .then(async (res) => {
    // handle success
    alert('Compra exitosa');
    //console.log(res.data)
    await this.sendEmail(res.data.no_factura)
    window.location.replace("/cart/" + this.props.user);
  })
  .catch(err => {
    // handle error
    alert('Hubo un problema. Intenta luego.')
    //console.log(err);
  })

  //      this.exportPDFWithMethod();
}

```

Nombre	tipo	params	Propósito
addToCart	async		Agrega una cantidad específica de un producto dentro del carrito

```
async addToCart() {  
      
    //(this.props)  
    if (parseInt(this.state.quantity) > 0) {  
        if (parseInt(this.state.quantity) > this.props.stock) {  
            alert('No hay tantas unidad disponibles. Consulte el stock')  
        } else {  
            await axios.post(HOST + "agregarProductoCarrito/", {  
                producto: parseInt(this.props.id),  
                usuario: parseInt(this.props.user),  
                cantidad: parseInt(this.state.quantity)  
            }).then(res => {  
                if (res.data === "agregado") {  
                    alert('Producto agregado')  
                    window.location.reload();  
                }  
            }).catch(err => {  
                console.log(err)  
            })  
        }  
    }  
}
```

src/screens

cart.js

Nombre	tipo	params	Propósito
getTotal	integer		Retorna el valor total de compra del carrito.
<pre>getTotal(){   var total = 0;   this.state.cartItems.forEach(item =&gt; {     total += item.precio*item.cantidad   });   return total; }</pre>			

login.js

Nombre	tipo	params	Propósito
login	async		Verificar si el usuario está registrado dentro de la aplicación.

```

async login() {
    //if (typeof (Storage) !== "undefined") {}

    await axios.post(HOST + 'login', {
        correo: this.state.email,
        contrasenia: this.state.pass
    })
        .then(async (res) => {
            // handle success
            //console.log(res.data)
            if (typeof (Storage) !== "undefined") {
                localStorage.setItem('user', res.data.usuario);
                window.location.replace('/');
            }
            else{
                alert('El navegador no soporta ciertas funciones para el despliegue de esta aplicación. Intento con otro.')
            }
        })
        .catch(err => {
            // handle error
            alert('Credenciales incorrectos')
        })
    }
}

```

## productsList.js

Nombre	tipo	params	Propósito
fillPages	async		Se encarga de hacer el paginado de productos, mostrando 10 productos por página.

```

async fillPages() {

    var page = []
    let pages = [];

    ...
    await this.state.items.forEach((item, i) => {
        ...
        if (!Number.isInteger((i + 1) / 10)) {
            page.push(item)
        } else {
            page.push(item)
            pages.push(page);
            page = [];
        }
    })
    ...
    await pages.push(page);
    ...
    await this.setState({ pages: pages });
}

```

register.js

Nombre	tipo	params	Propósito
register	async		Se encarga de registrar a un usuario dentro de la aplicación.

```

async register(e) {
  e.preventDefault();
  if(this.validate()){
    await axios.post(HOST + 'registrarUsuario', {
      nombre:this.state.name,
      apellido:this.state.lastName,
      dpi:this.state.dpi+ "",
      correo:this.state.email,
      contrasenia:this.state.pass,
      direccion:this.state.address
    })
    .then(async (res) => {
      // handle success
      if(res.data){
        alert('Registro exitoso. Sera redirigido al Login.')
        window.location.replace('/login')
      }
      console.log(res.data)
    })
    .catch(err => {
      // handle error
      alert('Error en el registro.')
    })

    await e.preventDefault();
  }
}

```

searchResults.js

Nombre	tipo	params	Propósito
searchForProduct	async		Se encarga de buscar un producto que cumpla con la palabra a buscar por medio de una Regex.

```

async searchForProduct() {

  var prodsFound = [];
  var product = this.props.match.params.product.slice(1);
  await axios.get(HOST + 'obtenerProductos')
    .then((res) => {
      // handle success
      var temp = "";
      res.data.productos.forEach(p => {
        temp = p.nombre_producto + ""
        if (temp.search(product) !== -1) {
          prodsFound.push(p)
        }
      });
      this.setState({ items: prodsFound, found:true }, () => this.fillPages())
    })
    .catch(function (error) {
      console.log(error)
    })
}

```

ticket.js

Nombre	tipo	params	Propósito
exportDPFWithMethod	async		Se encarga de descargar el PDF de una factura en específico.

```

async exportPDFWithMethod() {

  let element = this.pdfExportComponent.current
  const doc = await savePDF(element, {
    paperSize: "auto",
    margin: 40,
    fileName: `Factura electrónica`,
  });
}

```