



Trabajo de Fin de Ciclo

Administración de Sistemas Informáticos en Red

Ciclo Formativo de Grado Superior

Salesianos Atocha



API_INFINITY 1.0

Fang En Zhang

Javier Salazar Díaz

Tutor: Jose Manuel

Contenido

1. Introducción	3
1.1. Objetivo	3
1.1.1. Objetivo principal	3
1.1.2. Objetivo secundario	3
1.2. Razones	3
1.3. Justificación	4
1.4. Análisis de lo existente	5
1.5. Propuesta detallada	6
2. Evaluación de costes	6
3. Planificación inicial y real	7
4. Fundamentos teóricos	7
5. análisis de requisitos	8
6. Diseño y estructurado	8
6.1. Estructura	9
6.2. Programación estructurada.	10
6.3. Diagrama de Base de datos	11
7. Tecnología empleada y Entorno de programación	13
7.1. Entorno de programación	13
7.2. Visual Studio Code	13
7.3. Web APIs	14
7.3.1. REST API	14
7.4. JavaScript	15
7.4.1. ES/ECMAScript 6.00	15
7.5. NodeJS	16
7.5.1. Framework	16
7.6. NPMJS	22
7.7. Bootstrap	23
7.8. MySQL	24
8. Implementación	25
8.1. Uso del Instalador de paquetes NPM	25
8.2. Servidor web con Node.JS Express	26
8.3. Configuración del Handlebars	26
9. Pruebas y resultados	27
9.1. Comprobar el buscador.	27
9.2. Comprobar la barra de navegador de categorías.	28

9.3.	Comprobar la galería de fotos.....	29
9.4.	Comprobar el mostrador de los productos con más vistos.	30
9.5.	Comprobar el mostrador de productos aleatorio.	31
9.6.	Comprobar la función registrar la cuenta.....	32
9.7.	Comprobar la función de loguear.	33
9.8.	Comprobar las funciones del usuario.	34
9.9.	Carrito de compra.....	35
9.10.	Carrito de compra.....	36
10.	Conclusión.....	37
11.	Bibliografía.....	38

1. Introducción

Este apartado hará un breve resumen de que es "Infinity 1.0", comentará cuales han sido las motivaciones que nos han llevado a desarrollarla, explicará que se pretende realizando este trabajo y que tiempos se han sido necesarios para su realización, así como enumerará las razones de dichos objetivos.

Con él se pretende poner en contexto al lector y situarlo dentro del tema que será tratado en esta breve memoria.

1.1. Objetivo

1.1.1. Objetivo principal

A través de estos dos semestres de estudios avanzados, hemos entendido la programación y el desarrollo de la página web y los contenidos relacionados. Debido al gusto especial por la programación, decidimos crear y desarrollar un sitio web de ventas en línea similar a Amazon y Aliexpress. Esperamos poner en práctica lo que hemos aprendido a través de este proyecto, obtener experiencia práctica y tener un profundo conocimiento de Conocimientos relacionados con Front- End y Back-End, por ejemplo el uso del "Framework" de JavaScript las plantillas de HTML y CSS.

1.1.2. Objetivo secundario

Aprender cómo es una página web que nos podamos encontrar en cualquier lado, como crear una página web funcional, teniendo en cuenta si alguna vez en el trabajo nos surge hacer algo así, partir de una base y no empezarla desde cero, sabiendo como funciona, como poder utilizar las herramientas que nos da cada una de las opciones que podamos elegir.

1.2. Razones

Nuestro proyecto enfoca nuestras mayores virtudes en este sector, queremos enfocarnos en cosas muy similares a la hora de seguir estudiando o adquiriendo experiencia en este mundo, entonces, hemos creído en esta idea acorde con esto ya que implementar nuestros puntos más fuertes ha conseguido que podamos seguir adelante con ambición, con ganas, solucionando errores y trabajando duro para llegar hasta el final, así creando una página web al gusto de nosotros dos.

1.3. Justificación

FangEn Zhang.

En primer lugar, quiero mencionar que este Trabajo de Fin de Ciclo me ha permitido juntar dos temas que me han hecho ser lo que soy.

Para comprender esto mejor, creo es necesario retroceder un poco en el tiempo y explicar que desde siempre me ha gustado la informática y la programación. Debido por la profesión de mi padre, hace que siempre tengo curiosidad a cualquier dispositivo o instrumentos automatizados y descubrir cómo es su funcionamiento sobre todo de los dispositivos electrónicos como el automóvil teledirigido, ventilador, etc. Cuando mi padre me regaló primer "Family Computer", es la consola de videos juegos para la televisión más conocido y vendido en los años 90 y principio de los 2000 y desde entonces me llamó la atención el mundo de la informática. En aquel momento mi conocimiento de la informática no es nada más que los videos juegos. Más tarde, después de haber cursado dos ciclos formativos de FP, allí es cuando descubrí lo que es amplia el mundo de la informática y entre todos aquellos lo que más me atrae es la programación, porque con él puedo convertir mi sueño en algo real, y mi sueño siempre ha sido crear algo que puede interactuar con las personas, y tener Auto-juicio en ciertos puntos.

El poder hacer este proyecto me permite acercar un paso más a lo que quiero hacer, y al mismo tiempo adquiriendo la experiencia de practico, además de esto también me permite auto-evaluar a los conocimientos que adquirió durante estos dos años.

Javier Salazar.

Para mi este proyecto ha enfocado lo que más me gusta hacer en este ámbito, que es la programación, el manejo de información mediante bases de datos... El hecho de aprender, buscar cosas nuevas y ponerlas en práctica, llegando a hacer aplicaciones, no muy diferentes a lo aprendido en el curso, pero lo suficiente para tener algo más de conocimiento y de "experiencia" en este sector con la dificultad que supone aprender algo de forma autónoma y desde cero. Al igual que alimentar mi imaginación para posibles aplicaciones en el futuro.

Yo empecé hace 6 años en una Formación Profesional Básica de Informática debido a mi afán por los videojuegos y el mundo de internet, cada vez fui profundizando más tanto en clase como yo mismo, buscando información por mi cuenta, viendo videos, noticias... Cada vez interesándome más por este mundo hasta donde estoy ahora.

1.4. Análisis de lo existente

“Infinity 1.0” no es más que otra página web API, desarrollado completamente con el Framework de JavaScript Node.JS, Bootstrap, Handlebars, CSS y MySQL. la ventaja de desarrollar una página web con JavaScript Node.JS comparando con otras tecnologías como PHP:

- Tiene soporte de servidor incorporado.
- Una sola sintaxis para el lado del cliente y del servidor del sitio web. Esto mejora la reutilización del código y facilita el trabajo de un desarrollador fullstack.
- Módulo de almacenamiento en caché: los módulos se descargan y se inicializan cuando se les llama por primera vez, luego están constantemente disponibles.
- El módulo Stream facilita el trabajo con archivos grandes.
- Node tiene una sintaxis casi idéntica a JavaScript, por lo que es fácil de aprender y aprender para los desarrolladores de JS.

La ventaja del “Infinity 1.0” comparando con otros desarrollado por JavaScript Node.JS:

- Una comunidad para la venta de cualquier tipo de objetos
- Fácil manejo de la página web
- Chat a tiempo real con cualquiera de nuestros usuarios conectados
- Carrito de la compra para revisar/aplazar sus pedidos
- Alta seguridad mediante contraseñas encriptadas con “*bcriptjs*”
- Bajo coste en nuestros productos

1.5. Propuesta detallada

El proyecto "Infinity 1.0" es una página de compraventa de REST API. todo el proyecto comparte con el mismo diseño que está dividido en tres partes Encabezado, Cuerpo o contenido y piel de página:

- El encabezado contiene una barra de navegación, buscador, entrada al sistema de registrar e iniciar sesión.
- en el cuerpo contiene todos los contenidos que quieren mostrar para el usuario.
- en la piel de la página contiene todas las informaciones de la página como proveedor, contacto, etc.

Las páginas del proyecto pueden clasificar en tres tipos Home, Usuarios y Productos

MySQL, utilizado para registrar y almacenar los datos de cada usuario, como fecha de creación, tipo de usuario, nombre, contraseña, productos, etc.

JavaScript y su Framework, la herramienta fundamental para el desarrollo el proyecto "infinity 1.0", con ellos logramos crear y configurar propio servidor web y Aplicaciones en web. Con el módulo Express

2. Evaluación de costes

"Infinity 1.0" es como resto servidor Web, puede ejecutar en cualquier PC cuyo requisito superior al requisito mínimo del Linux Ubuntu. A demás de eso, para accederla por cualquier usuario que navega por internet también requiere una línea de internet y un servidor de Dominio o alquiler o comprar un nombre de dominio:

- Alquiler una línea de internet $25\text{€} * 12\text{Meses} = 300\text{€}$ Anual
- Cualquier PC que cumpla el requisito mínimo o superior al que pide Linux Ubuntu Desktop. con un precio media de 450€.

3. Planificación inicial y real

Comentar dos aspectos fundamentales:

- Tiempo y plazo de entrega
- Como organizar el desarrollo del proyecto

Primero, el plazo de entrega tenía que ser antes de empezar las FCT, ya que si no el trabajo se complica bastante.

- 4 semanas para buscar información y aprender a aplicarlo
- 4 semanas para desarrollar el proyecto
- 2 semanas para la escritura de la memoria

Todo ha ido muy bien, respetando los tiempos y sin muchos problemas, hemos mejorado y no hemos necesitado tanto tiempo como teníamos pensado al principio, por lo que la planificación real ha sido más o menos así:

- 2 semanas para buscar información y aprender a aplicarlo
- 3 semanas para desarrollar el proyecto
- 2 semanas para la escritura de la memoria

Hemos conseguido lo que nos propusimos, una página web totalmente funcional con aplicaciones nuevas, poco habituales y que no habíamos visto a lo largo del curso, como el Chat a tiempo real, cifrado unidireccional (las funciones HASH se basan en realizar un cálculo que devuelve un valor cada vez distinto, de longitud fija sobre el mismo texto).

4. Fundamentos teóricos

“Infinity 1.0” es un proyecto enfocado a desarrollar una página web de compraventas con múltiples aplicaciones de API. este es un Proyecto interdisciplinario que incluye:

- Administración de sistemas operativos:
 - Gestión de permisos del entorno ejecución Linux Ubuntu.
- Servicios de Red e Internet:
 - Crear un entorno de ejecución de para alojamiento del servicio Web y DNS.
 - Configuración del servidor Web y DNS
- Implantación de Aplicaciones Web:
 - HTML
 - CSS
 - JavaScript
- Administración de Base de datos:
 - Diseño de base de datos
 - Diagrama de Entidad y Relación
 - Triggers
 - Procedimientos y Funciones
 - Usuarios y Permisos

5. análisis de requisitos

Debido a la excelente compatibilidad de JavaScript, para su ejecución tan sólo requiere el mismo requisito para la ejecución de un sistema Linux Ubuntu:

- 20 GB de espacio libre en disco.
- 1 GB de memoria RAM.
- Procesador a 1 GHz o superior.
- Un dispositivo DVD o un puerto USB para el soporte de instalación.

6. Diseño y estructurado

una vez que se han establecido el entorno para la ejecución del Node.JS, el diseño es la primera de tres actividades técnicas: diseño, implementación y pruebas. cada actividad transforma la información de forma que al final se obtiene una página web API validado.

Sin diseño del software nos arriesgamos a construir un sistema inestable, un sistema que falle cuando se realicen pequeños cambios y que sea difícil de probar, cuya la calidad no pueda ser evaluada hasta más adelante.

En esta fase se alcanza con mayor precisión una solución para el diseño y el estructurado del nuestro web API.

En este apartado se comentará:

- En primer lugar, explica la estructura del proyecto.
- la programación estructurada.
- La función y aplicación de las tecnologías que comento en el apartado anterior.
- Diseño de la base de datos

6.1. Estructura

El proyecto está guardado en una carpeta principal llamada **PROJECT**, que a su vez está dividida en dos carpetas: **Node-Modules**, **SRC**. Y dos archivos: **Package-lock.json**, **Package.json**. Archivos que contienen un listado de los módulos instalados, la versión, nombre del módulo, script...

Node-Modules es donde guardamos todos los módulos que hemos utilizado para nuestro proyecto.

SRC es donde trabajamos en el proyecto, es decir, es la carpeta de archivos propios funcionales. Está dividida en 6 carpetas: **Config**, **Database**, **Helpers**, **Public**, **Routes**, **Views**.

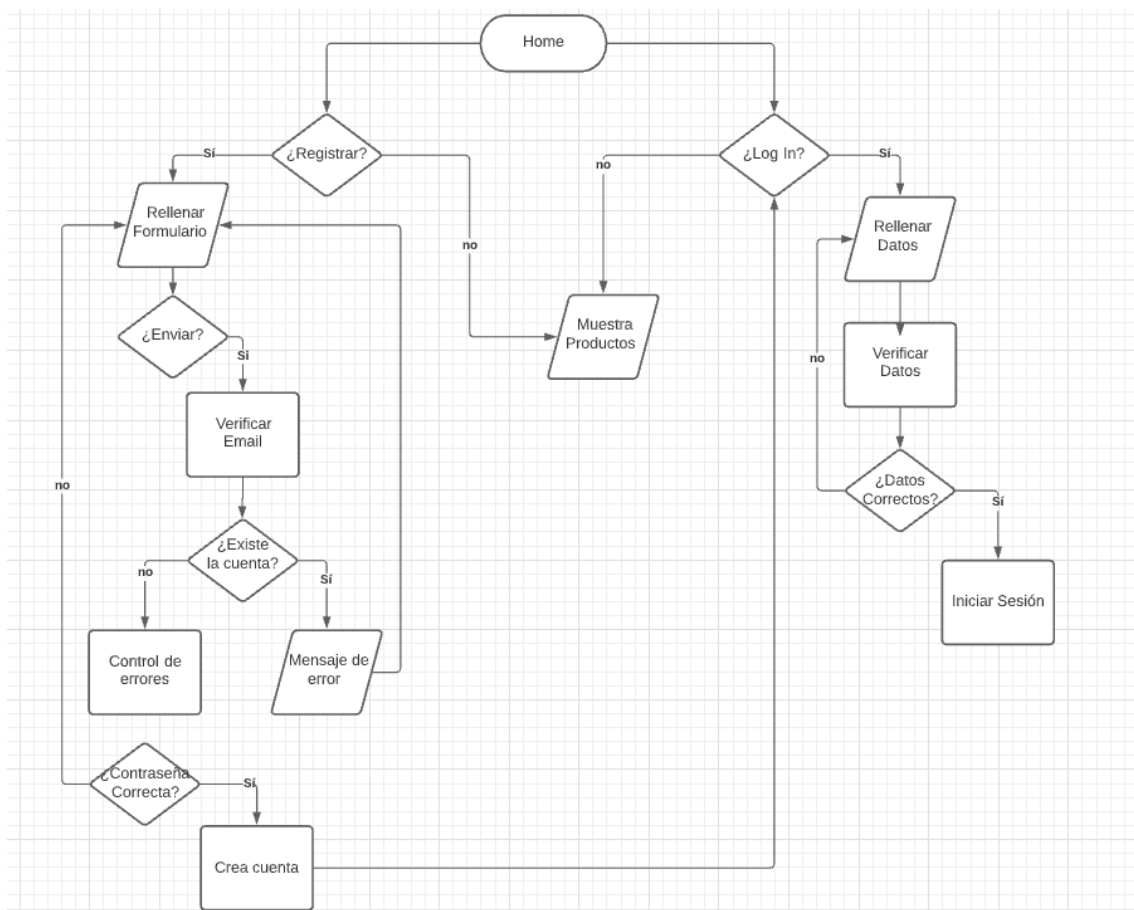
- **Config:** Donde guardamos los ficheros configuración de las aplicaciones web y del entorno.
- **Database:** Guardamos el fichero con el código de nuestra base de datos.
- **Helpers:** Guardamos los ficheros auxiliares del proyecto.
- **Public:** Guardamos todos los archivos estáticos y cualquier archivo que queremos enviar navegador y que pueda pintar en la pantalla.
 - o **CSS:** contiene el archivo CSS general.
 - o **IMG:** contiene las imágenes que ha subido cada usuario
 - o **JavaScript:** contiene aplicaciones como; control de formularios, chat, contador de visitas, slider.
- **Routes:** Donde guardamos todas las rutas para nuestro servidor. Tenemos un fichero principal **Index.js** y tres secundarios, **Products.js**, **Profile.js**, **User.js**.
- **Views:** Almacena todos los archivos que vamos a enviar al navegador.
 - o **Layouts:** Guardamos el diseño principal de nuestra página web.
 - o **Partials:** Pequeñas partes de HTML que podemos reutilizar en cualquier vista.
 - o **Product:** Guardamos todas las vistas relacionadas con el producto.
 - o **Users:** Guardamos todas las vistas relacionadas con el usuario

6.2. Programación estructurada.

La programación estructurada es una forma de escribir programas de computadora de forma clara, utilizando únicamente tres estructuras: secuencia, selección e iteración.

Ventajas:

- Los programas son más fáciles de entender, pueden ser leídos de forma secuencial y no hay necesidad de tener que rastrear saltos de líneas (GOTO) dentro de los bloques de código para intentar entender la lógica interna.
- La estructura de los programas es clara, puesto que las sentencias están más ligadas o relacionadas entre sí.
- Se optimiza el esfuerzo en las fases de pruebas y depuración.
- Se reducen los costos de mantenimiento. Análogamente a la depuración, durante la fase de mantenimiento, modificar o extender los programas resulta más fácil.
- Los programas son más sencillos y rápidos de confeccionar.
- Aumenta el rendimiento de los programadores.



6.3. Diagrama de Base de datos

Diagrama de Entidad-Relación:

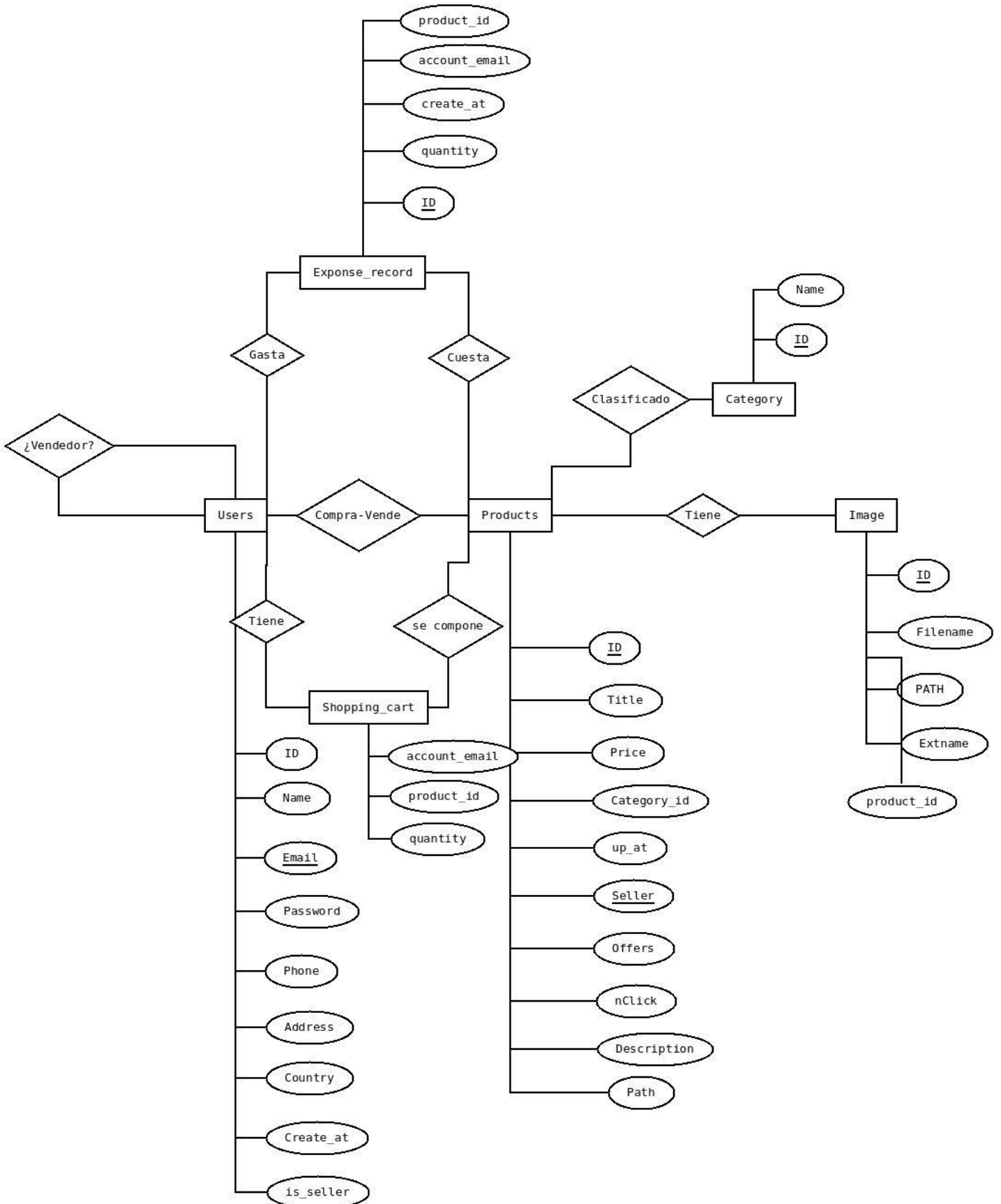
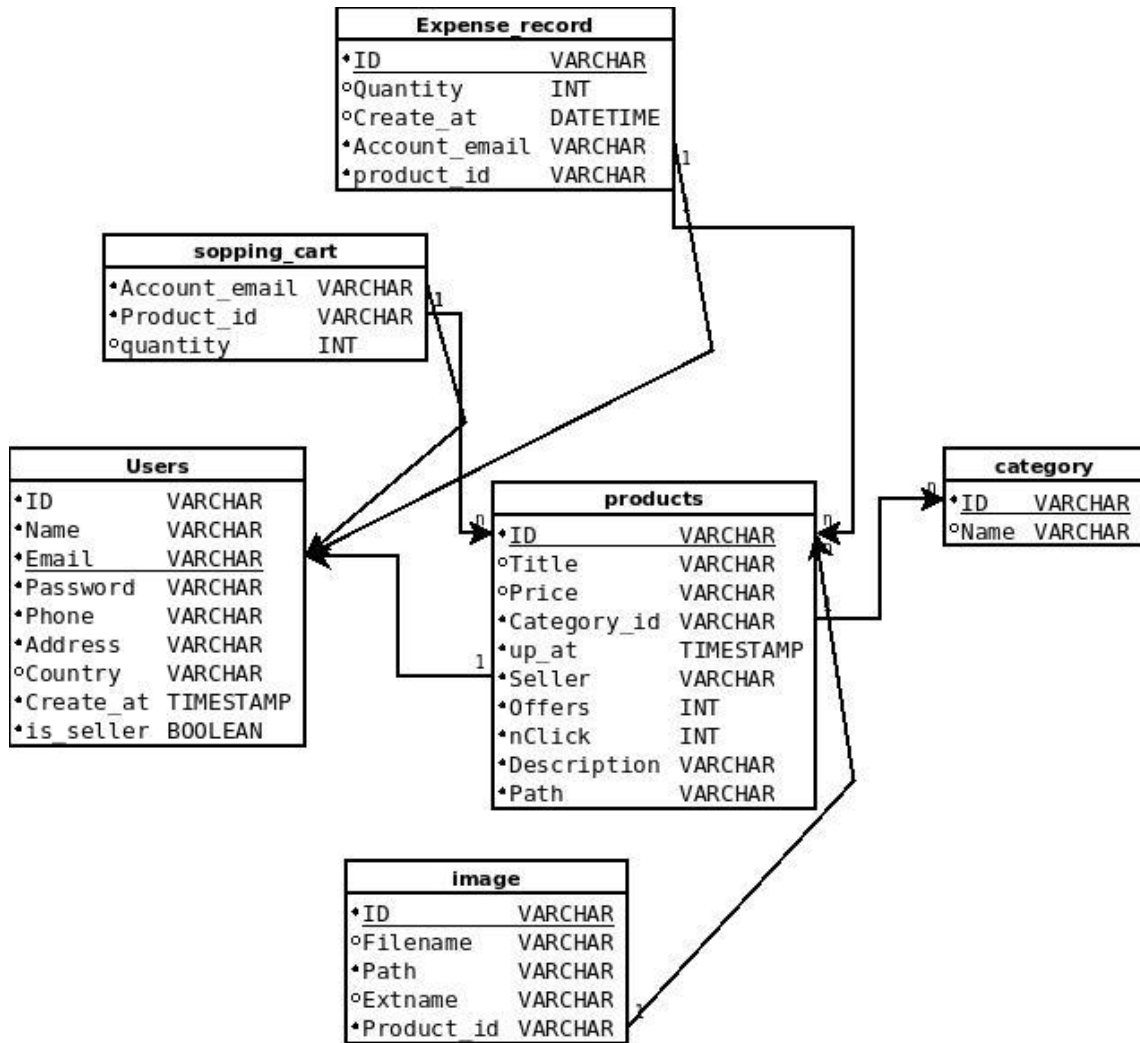


Diagrama de Clases:



7. Tecnología empleada y Entorno de programación

7.1. Entorno de programación

"Infinity 1.0", un proyecto de multi entornos. gracias a las dependencias de Express que convierte "Infinity 1.0" a un proyecto de multi entornos, es decir, se puede ejecutar perfectamente tanto en Linux como Windows y MAC, sin necesidad hacer ningún cambio, tan sólo tener instalada NPM, Node.JS y el base de datos MySQL.

7.2. Visual Studio Code

Es un editor de código fuente que permite trabajar con diversos lenguajes de programación, admite gestionar tus propios atajos de teclado y refactorizar el código, desarrollado por Microsoft para Windows, Linux y macOS.

Visual Studio Code es un editor con código abierto y nos proporciona una utilidad para descargar y gestionar extensiones con las que podemos personalizar y potenciar esta herramienta.

Visual Studio Code se basa en **Electron**, un Framework que se utiliza para implementar **Chromium** y Node.js como aplicaciones para escritorio, que se ejecuta en el motor de diseño **Blink**.



7.3. Web APIs

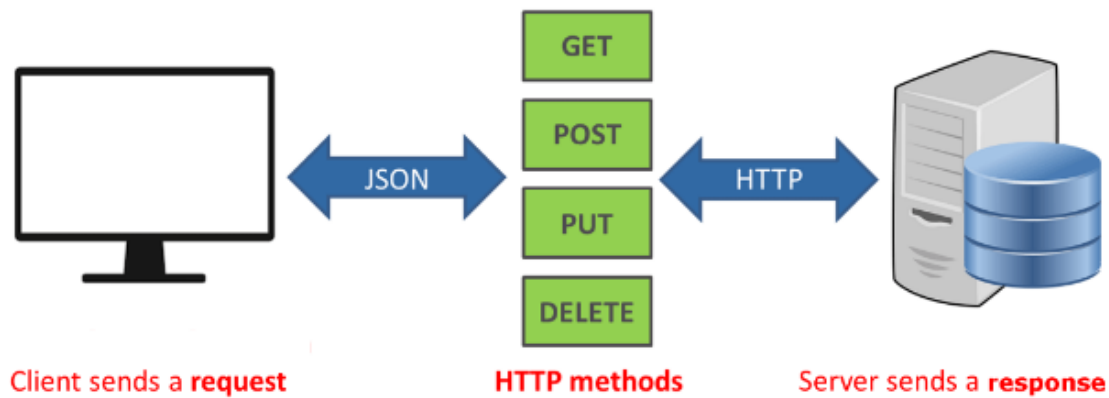
Las **Interfaces de Programación de Aplicaciones** son construcciones disponibles en los lenguajes de programación que permiten a los desarrolladores crear funcionalidades complejas de una manera **simple**, sirve para:

- **Enviar datos**
- **Manejar los datos flexibles**

7.3.1. REST API

Interfaz de aplicaciones para transferir datos es un tipo de arquitectura de desarrollo web que se apoya totalmente en el **estándar HTTP**. REST nos permite crear servicios y aplicaciones que pueden ser usadas por **cualquier dispositivos o cliente que entienda HTTP**, por lo que es increíblemente más simple y convencional que otras alternativas como SOAP y XML-RPC.

REST se definió **en el 2000** por Roy Fielding, como el autor principal también de la especificación **HTTP**. Podríamos considerar REST como un **FRAMEWORK** para construir aplicaciones web respetando **HTTP**.



7.4. JavaScript

Un **lenguaje interpretado** que se embebe en una página web HTML, es decir, se las analiza y procesa las instrucciones del navegador en el momento que deben ser ejecutadas. También se define como un lenguaje de programación orientado a objetos, basado en **prototipos**, **imperativo**, débilmente **tipado** y **dinámico**. Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web **dinámicas** y JavaScript del lado del servidor. Se diseñó con una sintaxis similar a **C**, aunque adopta nombre y convenciones del **JAVA**.

Todos los navegadores de modernos interpretan el código de **JavaScript** integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del **Documento Object Model (DOM)**.



7.4.1. ES/ECMAScript 6.00

Es un estándar de JavaScript, la primera versión publicó en el año 1997 llamada ECMA-262. ECMAScript define un lenguaje de tipos dinámicos ligeramente inspirado en Java y otros lenguajes del estilo de C. Soporta algunas características de la programación orientada a objetos mediante objetos basados en prototipos y pseudoclases.



7.5. NodeJS

Un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación **JavaScript**, **asíncrono**, con E/S de datos en una arquitectura **orientada a eventos** y basado en el **motor V8** de Google.

Node.JS utiliza un modelo de entrada y salida sin bloqueo controlado por eventos que lo hace y eficiente (con entrada nos referimos a solicitudes y con salida a respuestas). Puede referirse a cualquier operación, desde leer o escribir archivos de cualquier tipo hasta hacer una solicitud HTTP.

La idea principal de Node.JS es usar el modelo de entrada y salida sin bloqueo y controlado por eventos para seguir siendo liviano y eficiente frente a las aplicaciones en tiempo real de uso de datos que se ejecutan en los dispositivos.



7.5.1. Framework

Un Framework es un marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software. Con él podemos agilizar los procesos de desarrollo ya que evita tener que escribir código de forma repetitiva, asegura unas buenas prácticas y la consistencia del código, por lo tanto, podemos decir que un Framework es un conjunto de herramientas y módulos que pueden ser reutilizados para varios proyectos

Ventajas:

- Ahorra tiempo
- Facilita los desarrollos colaborativos, al dejar definidos unos estándares de programación
- Al estar ampliamente extendido, es más fácil encontrar herramientas, módulos e información para utilizarlo.
- Proporciona mayor seguridad, al tener gran parte de las potenciales vulnerabilidades resueltas.
- Normalmente existe una comunidad detrás, un conjunto de desarrolladores que pueden ayudar a responder consultas.
- Alta rendimiento y la escalabilidad en aplicaciones web.
- Portabilidad con Microsoft Windows, OS X, Linux, etc.

Node.JS Framework:

- *Express*:



Es el Framework web más popular de Node.JS, y es la librería subyacente para un gran número de otros Frameworks web de Node populares. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL.
- Integración con motores de renderización de “vistas” para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones “middleware” adicional en cualquier punto dentro de la tubería de manejo de la petición.

Ejemplo práctico para crear un servidor web básico con Node.JS Express:

```
// Se carga el módulo de HTTP
var http = require("http");

// Creación del servidor HTTP, y se define la escucha
// de peticiones en el puerto 8000
http.createServer(function(request, response) {

    // Se define la cabecera HTTP, con el estado HTTP (OK: 200) y el tipo de contenido
    response.writeHead(200, {'Content-Type': 'text/plain'});

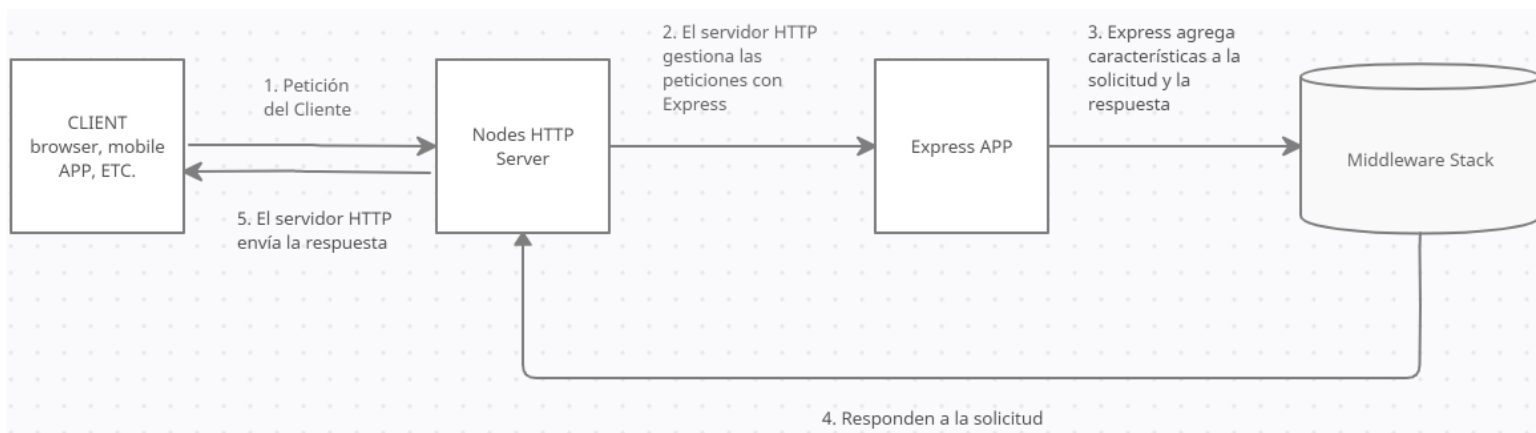
    // Se responde, en el cuerpo de la respuesta con el mensaje "Hello World"
    response.end('Hola Mundo!\n');
}).listen(8000);

// Se escribe la URL para el acceso al servidor
console.log('Servidor en la url http://127.0.0.1:8000/');
```

¿Qué agrega el Express.js a Node.js?

- abstrayendo mucha complejidad:
 - Simple y eficiente (un ejemplo, para subir una imagen con node.js puede ocupar más de 3 líneas, pero utilizando Express bastará con una sola línea)
 - refactorizar en el controlador de solicitudes monolíticas en muchos controladores de solicitudes más pequeños (todas las peticiones que llegan del navegador al servidor son manejada en una sola función “APP.get('/',(req,res)=>{})”).
 - Mantenable
 - Modular

Diagrama de flujo de Node.JS Express:



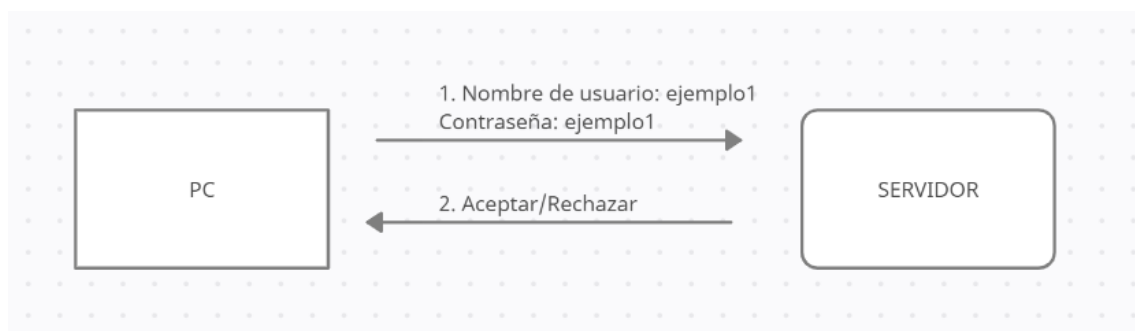
- Routing
 - rompe el manejador de solicitudes monolítico en trozos más pequeños.
 - La ejecución depende del método de la petición que solicita el cliente (URL y HTTP).
 - La ruta definida con el método HTTP, como: GET, POST, PUT, DELETE.

GET sirve para hacer peticiones.

POST sirve para actualizar o añadir algo.

PUT sirve para actualizar

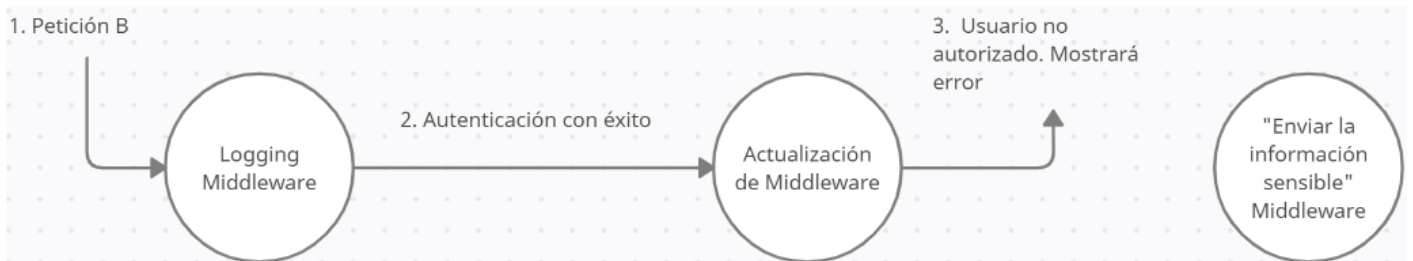
DELETE sirve para eliminar.



- Middleware (función intermediaria de Express)
 - Función que procesa antes de llegar a las rutas
 - Gestión de las funciones (son funciones que permiten manejar las peticiones)
 - Función que se activan por la ruta del navegador que recibe Express
 - Ejecutada por el orden en el que son añadidas
 - Funcionalidad:
 - Autorización
 - Log In
 - Gestión de la configuración de http



Estos son tres funciones independientes y cada una lleva su propia función



EJEMPLO:

Registro de la petición

```
const express = require ('express');
const app = express();

//middleware manual
app.use (function(req,res,next){ //'next' sign/sirve para continuar
  console.log('request url: '+req.url);      /**Registra la información de la petición de ruta*
  next();      /**Recibe la información del navegador y del servidor para continuar*
});

app.use ((req, res, next)=>{
  console.log('Ha pasado esta funcion');
  next();
});
//rutas

app.get('/', (req,res)=>{
  res.end('Hello word!');
});

app.get ('/login',(req,res)=>{
  res.end('Logging');
});

app.get('*', (req,res)=>{
  res.end('Archivo no encontrado');
});

app.listen(3000,()=>{
  console.log('servidor iniciado');      /**habilita el puerto 3000 e imprime
  /**por pantalla 'Servidor iniciado' al iniciar el servidor*
});
```

- *Templates/Plantillas o modelo:*

handlebars



Express

Son HTML que pueden reutilizar, agregar funcionalidad de otra programación a nuestro HTML, para reducir la complejidad de estar escribiendo todo a mano.

- Vista de HTML dinámico.
- Trabaja con JSON
- Ejemplos de motores de plantillas: EJS, ERB, Jinja2, Razor, Handlebars, Jade, Pug, etc. En este caso hemos utilizado Handlebars porque es uno de los más utilizados y “fácil” de manejar.
 - Los motores de plantillas lo que hacen es potenciar el HTML, pudiendo utilizar bucles, condicionales...

```

{{#if success_msg }}
<div class="alert alert-success alert-dismissible fade show" role="alert">
  {{success_msg}}
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
</div>
{{/if}}

{{#if error_msg }}

<div class="toast align-items-center text-white bg-primary border-0" role="alert" aria-live="assertive" aria-atomic="true">
  <div class="d-flex">
    <div class="toast-body">
      Hello, world! This is a toast message.
    </div>
    <button type="button" class="btn-close btn-close-white me-2 m-auto" data-bs-dismiss="toast" aria-label="Close"></button>
  </div>
</div>
{{/if}}
```

- **Express-Session:** almacena la sesión en el servidor, solo guarda el ID de sesión en la propia cookie, no los datos de sesión.
- **Express-MySQL-session:** Es un módulo de Express que permite mantener y guardar la conexión entre JavaScript y MySQL.
- **Express MyConnection:** Es el middleware Connect / Express proporciona una API coherente para las conexiones MySQL durante el ciclo de vida de solicitud / respuesta. Admite tres estrategias diferentes para administrar conexiones de base de datos
- **Morgan:** Es un middleware para la captura de solicitudes HTTP para Node. js para su posterior registro y seguimiento
- **Bcryptjs:** es una función de hashing de passwords diseñado por Niels Provos y David Maxieres, basado en el cifrado de Blowfish. Se usa por defecto en sistemas OpenBSD y algunas distribuciones Linux y SUSE.
- **Passport:** es un middleware de autenticación para Node. js que vamos a utilizar para la administración de sesiones.
- **Passport-local:** es un módulo del middleware de Passport que permite la integración fácil de una estrategia de autenticación local simple utilizando nombres de usuario y contraseñas.
- **Connect-flash:** Es un paquete para Express bastante util, nos permite mostrar mensajes en la pantalla bajo ciertas condiciones. Por ejemplo, para avisar cuando un usuario no tiene suficientes permisos como para realizar una acción, o al realizar cualquier otra acción como entrar o salir de su cuenta.
- **Multer:** es un middleware para Express y Node. js que hace que sea fácil manipular este multipart/form-data cuando tus usuarios suben archivos. En este tutorial, te mostraré cómo usar la biblioteca Multer para manejar diferentes situaciones de carga de archivos en Node.
- **Fs-Extra:** Es una mejora del middleware FS que permite implementa la programación asíncronica para procesar su creación, lectura, modificación, borrado etc.
- **UUID:** es un identificador único; personalmente lo uso para generar cadenas aleatorias y criptográficamente seguras. Hoy veremos cómo generar un ID único o identificador único con Node.JS y un paquete llamado UUID
- **Socket.io:** una biblioteca de JavaScript para aplicaciones web en tiempo real. Permite la comunicación bidireccional en tiempo real entre clientes y servidores web. Tiene dos partes: una biblioteca del lado del cliente que se ejecuta en el navegador y una biblioteca del lado del servidor para Node.js.

7.6. NPMJS



Es un gestor de paquetes desarrollado en lenguaje Javascript a través del cual Podemos obtener cualquier librería con tan solo una línea de instrucción, nos permite agregar dependencias de forma simple, distribuir paquetes y administrar cualquier módulo relacionado con NodeJS.

Ha sido utilizado durante años por los desarrolladores de JavaScript para compartir herramientas, instalar varios módulos y administrar sus dependencias. Funciona de dos maneras:

- Es un repositorio para la publicación de proyectos Node.js de código abierto, es decir, cualquiera puede publicar y compartir herramientas escritas en JavaScript.
- Herramienta en línea de comandos que ayuda a interactuar con plataformas en línea, como navegadores y servidores. Instalando o desinstalando paquetes, gestionar versiones y dependencias necesarias para ejecutar en un Proyecto.

NPM se trabaja con un archivo de extensión. `.json` para guardar y registrar los módulos que hemos utilizado para el proyecto.

Los metadatos muestran algunos aspectos del proyecto en el siguiente orden:

- El nombre del proyecto
- La versión inicial
- Descripción
- El punto de entrada
- Comandos de prueba
- El repositorio git
- Palabras clave
- Licencia
- Dependencias
- Dependencias de desarrollo

7.7. Bootstrap



Es un kit de herramientas de Código abierto para desarrollos web con HTML, CSS y JavaScript. Sirve para dar forma(estilo) a tu sitio web a través del uso de librerías CSS y JavaScript.

Te permite crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas.

Compatible con:

- Google Chrome
- Mozilla Firefox
- Internet Explorer (Windows)
- Microsoft Edge (Windows, Android, iOS, Windows 10 Mobile)
- Safari (MAC, iOS)
- Opera (MAC, Windows)

7.8. MySQL



Es un Sistema de gestión de bases de datos relacionales de código abierto (RDBMS) con un modelo cliente-servidor. **RDBMS** es un software o servicio utilizado para crear y administrar bases de datos basadas en un modelo relacional.

A continuación, se hará un breve repaso de la historia de MySQL destacando sus momentos más significativos:

- Desarrollado por una compañía sueca llamada MySQL AB en el año 1994
- En el año 2008, la compañía **Sun Microsystems** tomó el control por completo cuando compró MySQL AB.
- El gigante de la tecnología estadounidense Oracle adquirió **Sun Microsystems** en el año 2010.

8. Implementación

8.1. Uso del Instalador de paquetes NPM

NPM es un instalador de paquetes, a cada vez iniciemos e instalemos un módulo o una dependencia se auto genera un archivos llamado "package.json", este archivos guarda las informaciones como el nombre del paquete o dependencia, numero versión, Script y descripción.

Comando para iniciar npm y crear el archivo package.json:

- "npm init --yes"

Comando para instalar y desinstalar un módulo o paquetes:

- "npm install xxxx --save"
- ""npm remove xxxx"

Información del proyecto como nombre, versión del NPM, Archivo JS principal y el Script

```
{
  "name": "final",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "dev": "nodemon src/index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.0.0",
    "bcryptjs": "^2.4.3",
    "connect-flash": "^0.1.1",
    "express": "^4.17.1",
    "express-handlebars": "^5.1.0",
    "express-myconnection": "^1.0.4",
    "express-session": "^1.17.1",
    "fs-extra": "^9.0.1",
    "method-override": "^3.0.0",
    "morgan": "^1.10.0",
    "multer": "^1.4.2",
    "mysql": "^2.18.1",
    "passport": "^0.4.1",
    "passport-local": "^1.0.0",
    "socket.io": "^2.3.0",
    "timeago.js": "^4.0.2",
    "uuid": "^8.3.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.4"
  }
}
```

Una lista del registro con el nombre y la versión de las dependencias instalado.

Una lista con el nombre y la versión de las dev-dependencias, son utilizado exclusivamente para el entorno de desarrollo.

8.2. Servidor web con Node.JS Express

Sentencias para llamar al módulo Express y asignar a la variable "app"

Establecer una ruta inicial, donde vamos a poner nuestro formulario (HTML) y parámetro.

```
1 const express = require('express');
2 // Un simple servidor con Express
3 const app = express();
4 app.set('port', process.env.PORT || 3000);
5
6 app.get('/', (req, res) => {
7     res.end("Worked!");
8 });
9
10 const server = app.listen(app.get('port'), () => {
11     console.log("Server on Port", app.get('port'));
12 });
```

Declarar variable de puerto para el entorno. (por defecto se utilizará el que nos dar el servidor, si no existe utilizará el puerto 3000).

Sentencias para la declaración del puerto de nuestro servidor

8.3. Configuración del Handlebars

Asignar la ruta donde guarda los archivos estáticos a la variable **views**

Nombre de la plantilla del diseño que va a utilizar.

Declaración de las carpetas que va a guardar .hbs

```
1 const express = require('express');
2 const exphbs = require('express-handlebars');
3 // Un simple servidor con Express
4 const app = express();
5
6 //Setting
7 app.set('port', process.env.PORT || 3000);
8 app.set('views', path.join(__dirname, 'views'));
9 app.engine('.hbs', exphbs({
10     defaultLayout: 'main',
11     layoutsDir: path.join(app.get('views'), 'layouts'),
12     partialsDir: path.join(app.get('views'), 'partials'),
13     extname: '.hbs'
14 }));
15 app.set('view engine', '.hbs');
16
17 app.get('/', (req, res) => {
18     res.end("Worked!");
19 });
20
21 const server = app.listen(app.get('port'), () => {
22     console.log("Server on Port", app.get('port'));
23 });
```

Llamar al mdulo express-handlebars

Declarar el uso de Handlebars y configuración

Declarar la extensión de los archivos

9. Pruebas y resultados

9.1. Comprobar el buscador.

Para esta prueba simplemente se ha introducido la palabra clave al área de buscador y cliquesa al botón o presionar al teclado “**Enter**” de búsqueda para iniciar la búsqueda.

Tras realizar dicha prueba, se obtiene el resultado que se puede ver en la figura xx y que se muestra a continuación.

The image shows two screenshots of the INFINITY website's search interface. The top screenshot shows the search bar with the text 'i have' and a magnifying glass icon. Below the search bar, there are category links: Arts & Crafts, Books & Cinema, Computers, Electronics, Garden & pets, Smart Home, Sports & Outdoors, Toys & Baby, and Video Games. The bottom screenshot shows the search results for 'i have'. On the left, there is a 'Filter' sidebar with a 'Select' dropdown, a 'Price' field, an 'Order by' dropdown, and a green 'Filter' button. The main content area displays a book cover for 'El inversor inteligente' by Benjamin Graham. The book cover features a pair of glasses and the text: 'Prólogo y apéndice de Warren E. Buffett', 'Ventura capitalista y el fundador por Jason Zenger', 'Benjamin Graham', 'El inversor inteligente', and '“Sin lugar a dudas, el mejor libro sobre inversión jamás escrito” Warren E. Buffett'. Below the book cover, the text 'i have to creat...' is partially visible, followed by 'sdadas' and the price '13.3€'.

9.2. Comprobar la barra de navegador de categorías.

En esta prueba se intenta mostrar los productos según su categoría.

Así tras hacer un clic a nombre de categoría nos mostrará los productos sólo de esa categoría.

The screenshot shows the INFINITY website interface. At the top, there is a search bar with the placeholder text "Search for anything". Below the search bar, a horizontal menu lists categories: Arts & Crafts, Books & Cinema, Computers, Electronics, Garden & pets, and Smart Home. The "Books & Cinema" category is highlighted. On the left, a "Filter" sidebar is visible, containing a "Select" dropdown menu, a "Price:" field with two input boxes, an "Order by" dropdown menu, and a green "Filter" button. The main content area displays two book products. The first product is "El inversor inteligente" by Benjamin Graham, with a price of 13.3€. The second product is "CAMINO DE SERVIDUMBRE" by Primitivo HATEK, with a price of 3€.

The screenshot shows the INFINITY website interface. At the top, there is a search bar with the placeholder text "Search for anything". Below the search bar, a horizontal menu lists categories: Arts & Crafts, Books & Cinema, Computers, Electronics, and Garden & pets. The "Computers" category is highlighted. On the left, a "Filter" sidebar is visible, containing a "Select" dropdown menu, a "Price:" field with two input boxes, an "Order by" dropdown menu, and a green "Filter" button. The main content area displays two laptop products. The first product is "computer", with a price of 279€. The second product is "computer2", with a price of 279€.

9.3. Comprobar la galería de fotos.



Search for anything



Sign In/
Join



- Arts & Crafts
- Books & Cinema
- Computers
- Electronics
- Garden & pets
- Smart Home
- Sports & Outdoors
- Toys & Baby
- Video Games



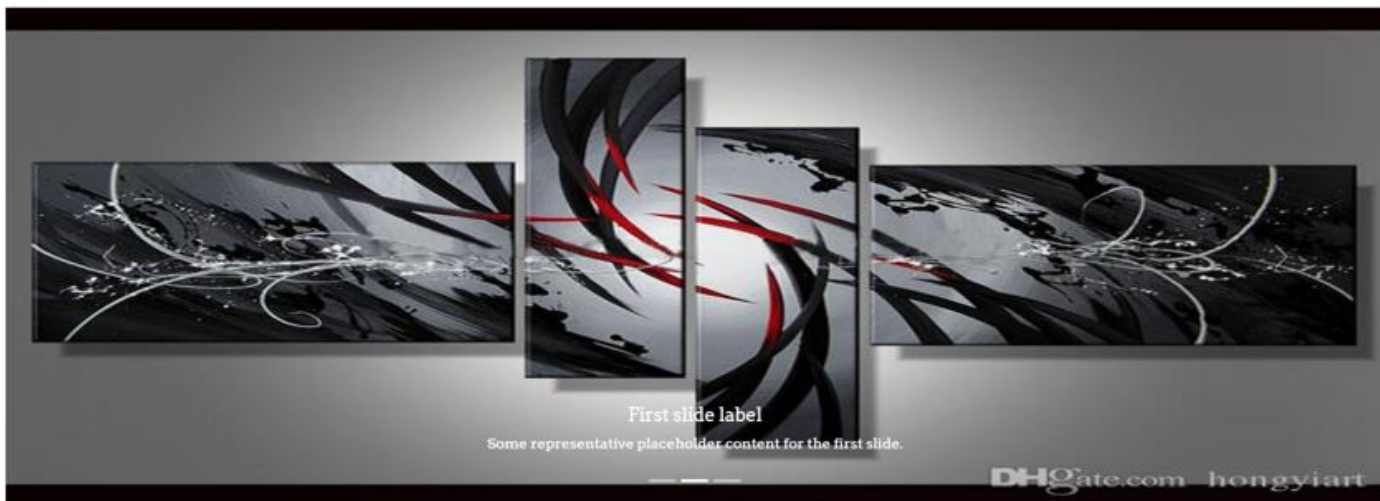
Search for anything



Sign In/
Join

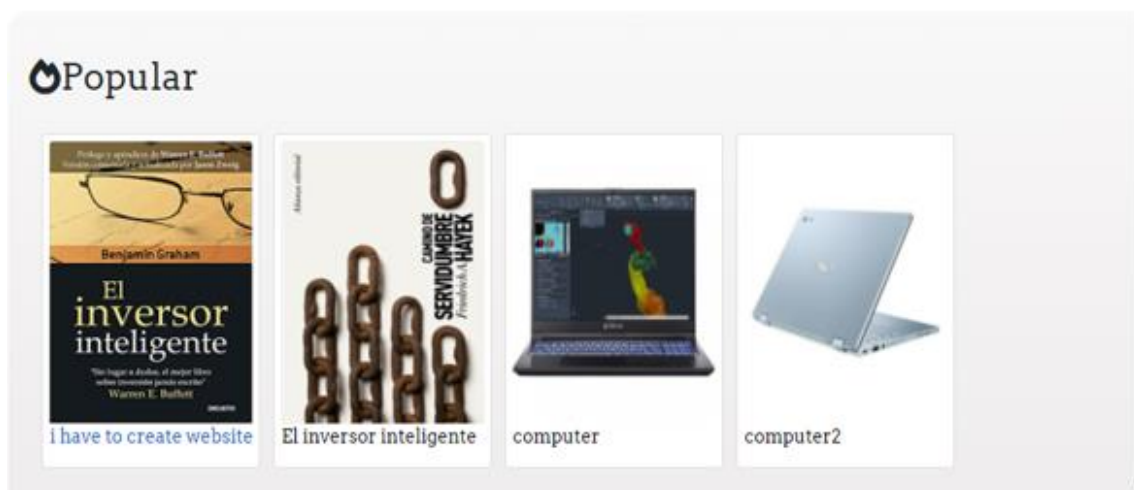
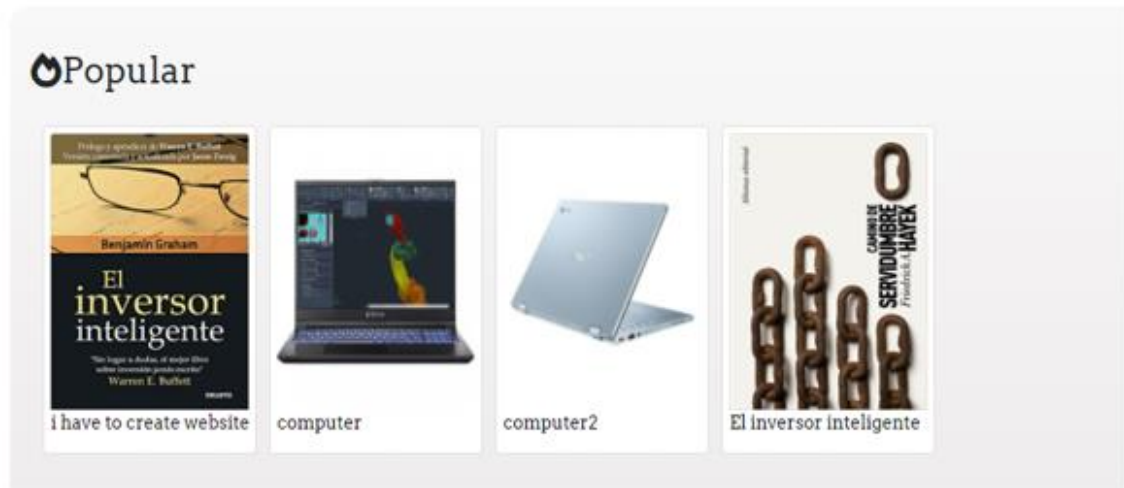


- Arts & Crafts
- Books & Cinema
- Computers
- Electronics
- Garden & pets
- Smart Home
- Sports & Outdoors
- Toys & Baby
- Video Games



9.4. Comprobar el mostrador de los productos con más vistos.

Esta prueba intenta hacer clic en el producto para aumentar su número de clics. Podemos juzgar si está funcionando normalmente clasificando los productos en el cuadro de popularidad.



9.5. Comprobar el mostrador de productos aleatorio.

Cada vez que actualizamos la página sale aleatoriamente los productos.

Recommend for you



Recommend for you




9.6. Comprobar la función registrar la cuenta.

Al enviar los datos quedan registrados en la base de datos y se comprueba que todo se haya introducido correctamente, si ha sido correcto nos saldrá un mensaje en verde, mientras que si algo ha salido mal, nos saldrá un mensaje rojo explicándonos ese fallo

User Name test2	Email address test2@test.com	
Password	Confirm_password	
Phone 123123123	Address qweqwewqe	Country España
<input checked="" type="checkbox"/> Agree to terms and conditions		
Sign UP		

You are registered

Account Login




Email address
Password
Sign In

New customer? [Start here.](#)

9.7. Comprobar la función de loguear.

Consultará en la base de datos los parámetros introducidos y comprobará que todo concuerde con lo que tenemos registrado. En el caso de que esto falle, nos enseñará un mensaje de error. En caso contrario, entrará en la cuenta y nos enseñará todos los datos guardados respecto a esta.

Account Login



Email address
test2@test.com

Password
.....|

[Sign In](#)

New customer? [Start here.](#)

profile

WellCome **test2**

Profile	User Datas
Messages	Email : test2@test.com
Settings	Phone: 123123123
Cart	Address: qweqwewqe
	Country: España
	Create At: Sun Mar 21 2021 16:25:52 GMT+0100 (Central European Standard Time)

9.8. Comprobar las funciones del usuario.

El chat implementado mediante un módulo de NodeJS que lo que hace es establecer una conexión en vivo con toda la comunidad conectada a nuestro servidor.

profile

WellCome **test2**

Profile
Messages
Settings
Cart

lets Chat

Send

profile

WellCome **test2**

Profile
Messages
Settings
Cart

lets Chat

fangen zhang : hola test2

Send

9.9. Carrito de compra

Al añadir un artículo al carrito quedará registrado en la base de datos y al entrar en él, hará una consulta que nos enseñe que tiene cada persona en el carrito.



Search for anything

test2@test.com

El inversor inteligente

3€

Quantity

Previous **1** 2 3 Next



Search for anything

test2@test.com

El inversor inteligente

3€

Quantity

computer2

279€

Quantity

Previous **1** 2 3 Next

9.10. Carrito de compra

Al entrar en este vínculo pedirá las credenciales de administrador, en la segunda foto enseña cómo puede administrar, (borrar, editar, etc.) usuarios.

192.168.1.175:3000/users/manager

INFINITY
... SLOGON HERE ...

profile

WellCome fangen zhang

- Profile
- My product
- Up product
- Messages**
- Settings
- Cart

lets Chat

fangen zhang : hola test2





▲ No seguro | 192.168.1.175:3000/users/manager

INFINITY
... SLOGON HERE ...

Search for anything

User Email

Search

id	Email	Password	Phone Number	Created At	Admin?	Seller?	Action
8810913f-3349-43a1-a6f6-a4762867ef90	test	test@test.com	qwewqewqe	Sat Mar 20 2021 08:38:28 GMT+0100 (Central European Standard Time)	0	1	 
0aa529b5-fb57-4f3c-bc1f-be95b1428517	test2	test2@test.com	España	Sun Mar 21 2021 16:25:52 GMT+0100 (Central European Standard Time)		0	 

10. Conclusión

Hemos juntado nuestros puntos más fuertes dentro de todas las posibilidades que teníamos, añadiendo conocimientos a parte que hemos adquirido mediante tutoriales, videos, información de internet.

También valernos por nosotros mismos para aprender algo nuevo, diferente y conseguir implementarlo correctamente para crear algo totalmente funcional.

Los dos empezamos hace 6 años en esta rama, nos conocemos perfectamente y sabemos hasta donde podemos llegar cada uno, sabemos trabajar juntos y eso ha hecho que podamos sacar este proyecto adelante y adquiriendo los conocimientos los dos juntos.

11. Bibliografía

Video de tutorial para Node.JS y sus módulos:

<https://fazitweb.com/aqwsz>

JavaScript:

<https://es.wikipedia.org/wiki/JavaScript>

ECMAScript:

<https://es.wikipedia.org/wiki/ECMAScript>

<https://openwebinars.net/blog/que-es-ecmascript/>

Rest Api:

<https://aprendiendoarduino.wordpress.com/2019/10/27/api-rest/>

Visual Studio Code:

<https://blog.aitana.es/2018/10/16/visual-studio-code/>

NodeJS

<https://openwebinars.net/blog/que-es-nodejs/>

https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction/