

PROYECTO INTEGRADO 24/25

FLASH BURGER



Fernando Martínez Ortiz

1. INTRODUCCIÓN

El proyecto consiste en la creación de una plataforma web para una hamburguesería centrada en delivery y take-away.

La plataforma permitirá a los clientes realizar pedidos en línea, personalizar sus hamburguesas, información sobre su pedido, si desea cancelarlo y si desea recogida o envío a domicilio.

Además, se añadirá una aplicación interna para los empleados que permitirá gestionar de manera eficiente los pedidos desde la cocina hasta la entrega.

Mi proyecto innova a la hora de tener una aplicación solo para los empleados es decir mejora en la experiencia y eficacia de la hamburguesería. Esto facilita a los empleados gestionar el proceso de entrega optimizando tiempos y recursos.

Mi aplicación se especializa en el formato delivery y take-away esto me permite desarrollar nuevas características que no están disponibles en hamburgueserías convencionales como la personalización avanzada y las opciones de recogida o entrega.

Por último en el mercado actual los clientes buscan opciones personalizables y la posibilidad de gestionar sus pedidos en cada fase, mi plataforma permite a los usuarios personalizar sus hamburguesas y una experiencia más dinámica

2. OBJETIVOS DEL PROYECTO

1. Página Web Principal de la Hamburguesería:

La página web debe proporcionar información relevante del negocio, incluyendo la ubicación, contacto, promociones y un acceso directo a la plataforma de pedidos, permitiendo que los usuarios conozcan los servicios y ofertas disponibles.

2. Menú de Productos:

Los clientes podrán visualizar todas las hamburguesas y productos disponibles, asegurando una experiencia de compra flexible.

3. Sistema de Pedidos:

Los clientes deberán poder realizar pedidos en tiempo real a través de la plataforma, asegurando una gestión eficiente de las solicitudes de los usuarios, con el objetivo de facilitar el proceso de compra.

4. Información del Pedido:

Los clientes podrán consultar los detalles de su pedido, incluyendo los productos elegidos, cantidades, y el estado del pedido, mejorando la transparencia y la experiencia de usuario.

5. Confirmación del Pedido:

El sistema deberá proporcionar una confirmación inmediata del pedido realizado por parte del cliente, informando al usuario que su solicitud ha sido registrada correctamente.

6. Cancelación del Pedido:

Los clientes tendrán la opción de cancelar su pedido siempre que no haya sido procesado aún, proporcionando flexibilidad y control sobre su compra.

7. Gestión de Pedidos (Aplicación para Empleados):

Los empleados deberán tener acceso a una vista en tiempo real de todos los pedidos entrantes, permitiendo una gestión eficiente de los pedidos en curso y los ya finalizados.

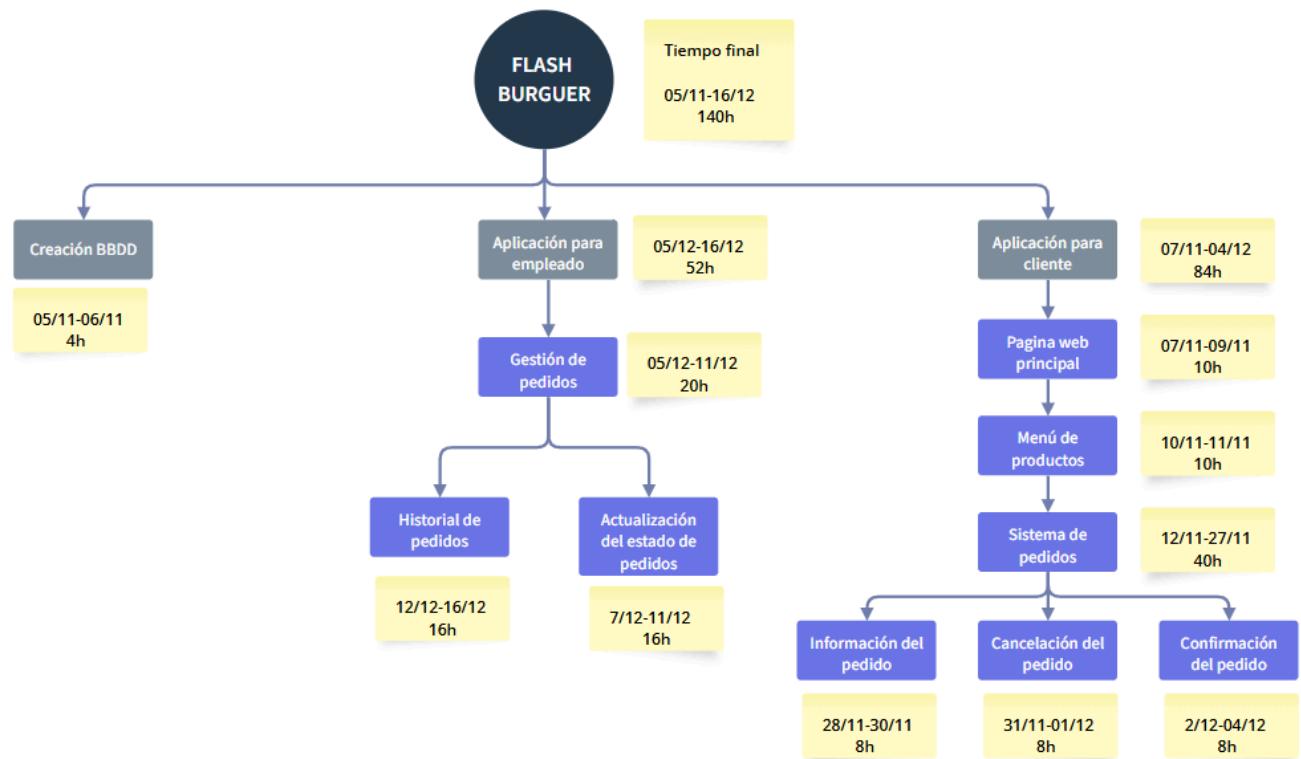
8. Actualización del Estado de los Pedidos:

Los cocineros podrán actualizar el estado de los pedidos marcándolos como "listos", mientras que los repartidores podrán indicar si un pedido ha sido entregado, asegurando que el flujo de trabajo sea visible y actualizado en tiempo real.

9. Historial de Pedidos:

Los empleados deberán poder consultar el historial de pedidos, permitiendo revisar los detalles de entregas anteriores, lo que ayudará a gestionar mejor las operaciones y la atención al cliente.

3. PLANIFICACIÓN DEL PROYECTO



Módulo	Total Horas	Coste (€)
Página web Principal	10	300€
Creación de la base de datos	4	120€
Menú de Productos	10	300€
Sistema de pedidos	40	1200€
Información del pedido	8	240€
Confirmación del pedido	8	240€
Cancelación del pedido	8	240€
Gestión de pedidos	20	600€
Actualización del Estado de pedidos	16	480€
Historial de pedidos	16	480€
Total:	140 Horas	4200€

4. ANÁLISIS Y DISEÑO DEL SISTEMA

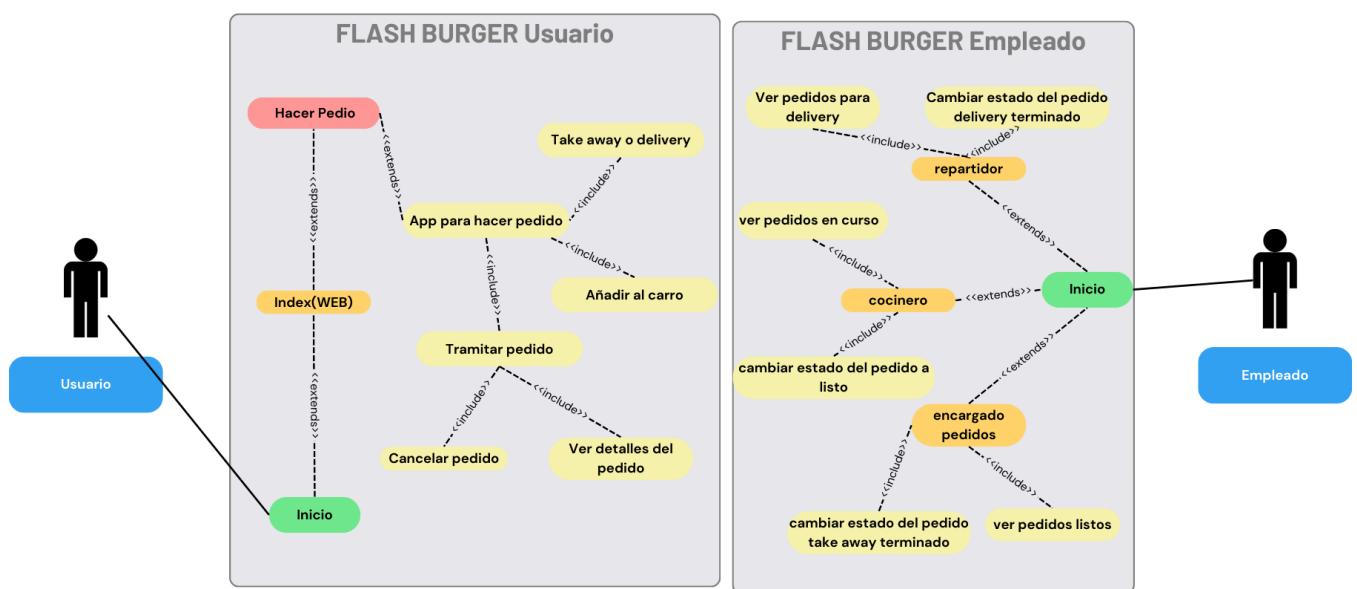


Diagrama entidad relación

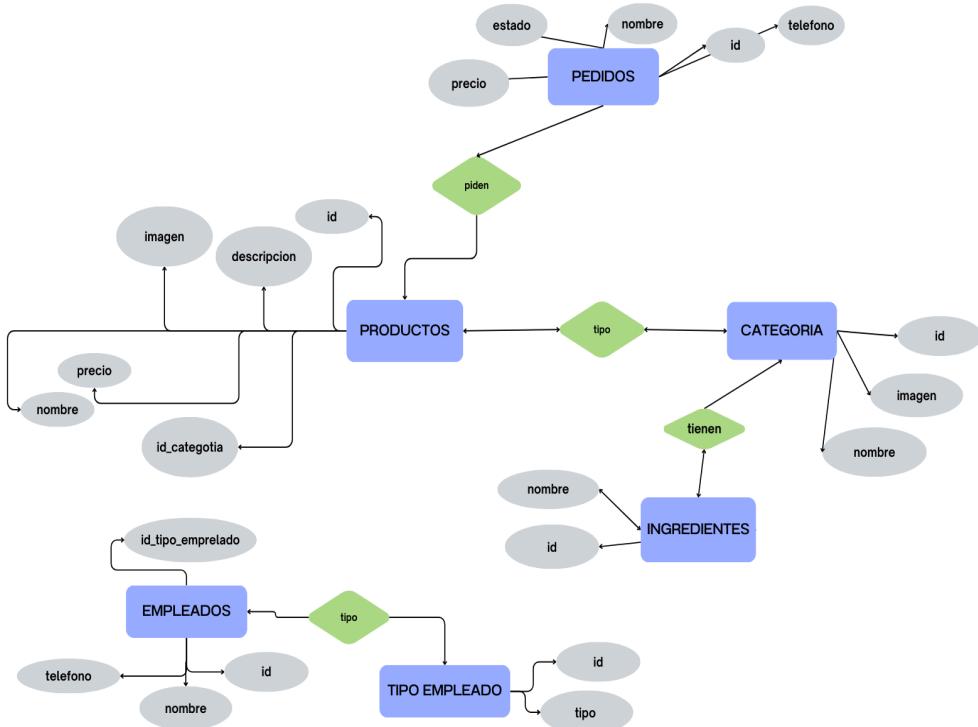
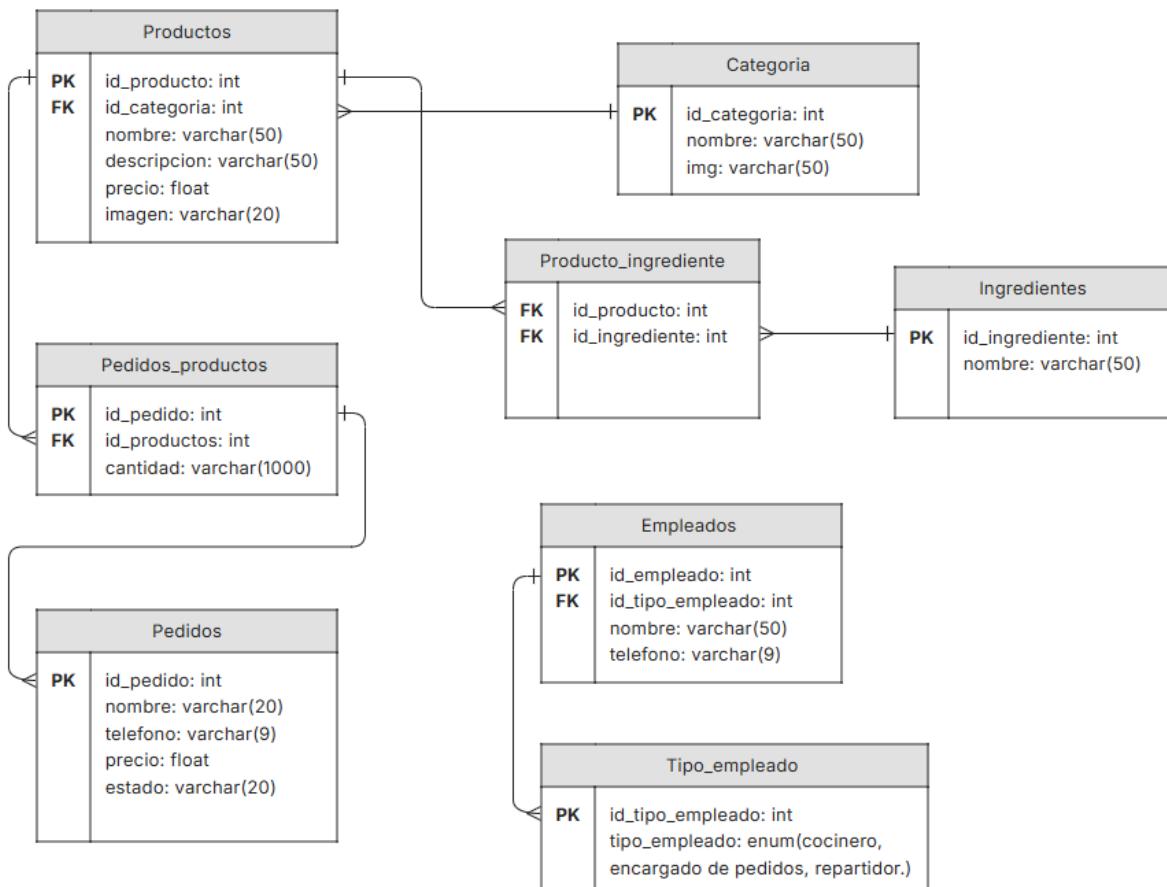


Diagrama de clases:



5. ESPECIFICACIONES DEL SISTEMA

5.1. Justificar el uso de la tecnología y el software empleado Justificar el uso de las tecnologías y el software empleado (C#, Microsoft XNA, Android...)

HTML5 y CSS3

HTML5 permite estructurar contenido web de manera semántica y estandarizada, garantizando compatibilidad con navegadores modernos.

CSS3 proporciona herramientas avanzadas para diseñar interfaces visualmente atractivas, responsivas y accesibles, adecuadas para dispositivos móviles y de escritorio.

Facilita la creación de una interfaz de usuario intuitiva, mejorando la experiencia del cliente al interactuar con la plataforma.

JavaScript

JavaScript es fundamental para la interactividad y dinamismo en aplicaciones web, permitiendo funcionalidades como validaciones de formularios, actualizaciones de contenido sin recargar la página, y respuestas rápidas a las acciones del usuario.

Mejora la fluidez en el uso de la plataforma, permitiendo personalizaciones en tiempo real y actualizaciones instantáneas del estado del pedido.

React.

Su enfoque basado en el DOM virtual mejora significativamente el rendimiento, especialmente en aplicaciones con actualizaciones frecuentes como esta.

Asegura una experiencia de usuario rápida, fluida y modular tanto para clientes como para empleados.

PHP

ideal para manejar solicitudes backend como la gestión de pedidos, el almacenamiento de datos y la lógica de negocio.

Facilita la comunicación segura entre el frontend y la base de datos, garantizando la escalabilidad y funcionalidad de la plataforma.

MySQL

MySQL es un sistema de gestión de bases de datos relacional confiable, que permite organizar y manejar grandes volúmenes de datos con rapidez y eficiencia.

Su estructura relacional es ideal para el manejo de tablas interdependientes como productos, usuarios y pedidos, lo que asegura la consistencia de los datos en tiempo real.

5.2. Instalación y configuración de la aplicación Se deberá documentar la instalación y configuración de la aplicación, así como de las librerías de terceros que se utilice. Si el proyecto tuviera una parte que correspondiera a un Servidor WEB donde albergar la información de la aplicación se debe documentar y justificar la configuración de los diferentes servidores (DNS, FTP, Apache...)

Requisitos previos

Para poder ejecutar esta aplicación, necesitarás tener instalado:

Node.js (incluye npm)

XAMPP

Instalación

Sigue estos pasos para instalar el proyecto:

Instalación de Dependencias de React:

En la carpeta raíz del proyecto, ejecuta el siguiente comando para instalar todas las dependencias necesarias

npm install

Configuración de la Base de Datos:

Inicia XAMPP y asegúrate de que los servicios Apache y MySQL están activados. Accede a phpMyAdmin en <http://localhost/phpmyadmin> para crear la base de datos que necesita el proyecto. Importa el archivo de script SQL (flashburger.sql que está en la carpeta data) en phpMyAdmin para configurar la base de datos con las tablas necesarias.

Configuración de las Funciones PHP

Abre el archivo funcion.php dentro de la carpeta funcionesPHP y verifica que los valores de conexión coincidan con tu configuración local de MySQL. Modifica los valores según sea necesario:

```
define("SERVIDOR_BD", "localhost");
define("USUARIO_BD", "root");
define("CLAVE_BD", "1234");
define("NOMBRE_BD", "flashburger");
```

Mueve la carpeta funcionesPHP dentro de la carpeta htdocs de XAMPP. La ruta final debería ser algo como C:\xampp\htdocs\funcionesPHP. Configura la ruta de acceso a las funciones PHP en el archivo datos.js que encontrarás en la carpeta conf. Asegúrate de que la URL de las funciones PHP apunte al servidor local de XAMPP.

Ejecutar el Proyecto: Una vez que hayas configurado la base de datos, las funciones PHP y las dependencias, vuelve a la raíz del proyecto y ejecuta el siguiente comando para iniciar la aplicación:
npm start

Instalación en servidor

Requisitos previos:

Tener una cuenta en alwaysdata

Tener el código a desplegar en github en un repositorio público

Tener una cuenta en vercel

Instalación de base de datos y llamadas a el php en el alwaysdata

Creamos un site

The screenshot shows the 'Sites' section of the alwaysdata web interface. On the left, there's a sidebar with 'flashburger' selected. In the main area, a table lists a single site: 'flashburger.alwaysdata.net/php' of type 'PHP'. A red box highlights the site name in the table, and a red arrow points to the '+ Add a site' button at the top right.

Configuraremos la ruta donde estará nuestro archivo de llamadas a php:

The screenshot shows the configuration page for the 'flashburger.alwaysdata.net/php' site. The left sidebar shows various service categories like Web, Domains, Databases, and MySQL. The main panel has two tabs: 'Addresses' and 'Configuration'. The 'Configuration' tab is active, showing fields for 'Type' (set to 'PHP'), 'Root directory' (set to '/home/flashburger/www/react/'), and 'PHP version'. A red box highlights the 'Root directory' field, and a red arrow points to it from the left sidebar.

En ese mismo site crearemos un base de datos mysql

The screenshot shows the 'MySQL databases' section of the alwaysdata interface. The left sidebar has 'MySQL' selected under 'Databases'. The main area shows a table with one database entry: 'flashburger_bd'. A red box highlights the database name 'flashburger_bd'. Above the table, a message box displays connection details: 'MySQL host: mysql-flashburger.alwaysdata.net' and 'Version: 10.11 (mariadb)'. A red box highlights this message box.

Importaremos nuestra base de datos con el script de sql y definiremos un usuario

The screenshot shows the phpMyAdmin interface for importing SQL files. The left sidebar lists databases, and the main area is titled 'Importing into the current server'. A red box highlights the 'Import' button in the top navigation bar. Another red box highlights the 'flashburger.sql' file selected in the 'File to import' dropdown.

Instalacion de aplicación web en vercel

Añadiremos un nuevo proyecto:

The screenshot shows the Vercel dashboard under the 'Overview' tab. A red box highlights the 'Add New...' button in the top right corner of the search and filter bar.

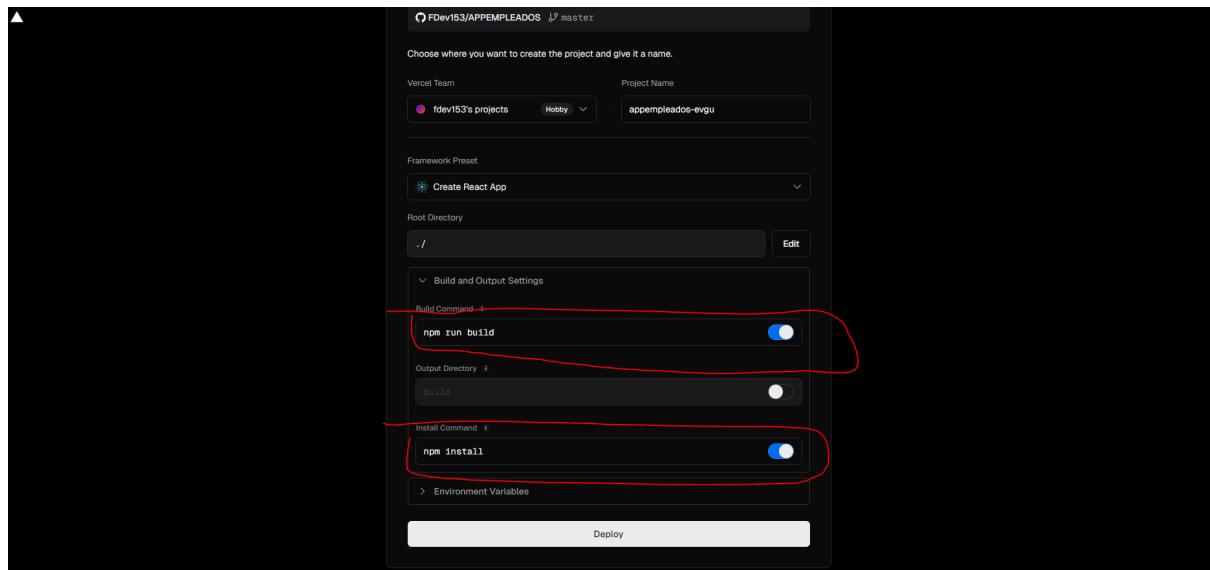
Elegiremos el que queramos desplegar:

The screenshot shows two side-by-side sections of the Vercel interface. On the left, the 'Import Git Repository' section shows two recent repositories: 'APPEMPLEADOS' (48m ago) and 'FLASHBURGERS' (2h ago), each with an 'Import' button. A red arrow points to the 'Import' button for 'APPEMPLEADOS'. On the right, the 'Clone Template' section displays four template cards: 'Next.js Boilerplate' (Next.js logo), 'AI Chatbot' (AI Chatbot logo), 'Commerce' (Commerce logo), and 'Vite + React Starter' (Vite + React logo). A red arrow points to the 'Framework' dropdown menu at the top right of the template section.

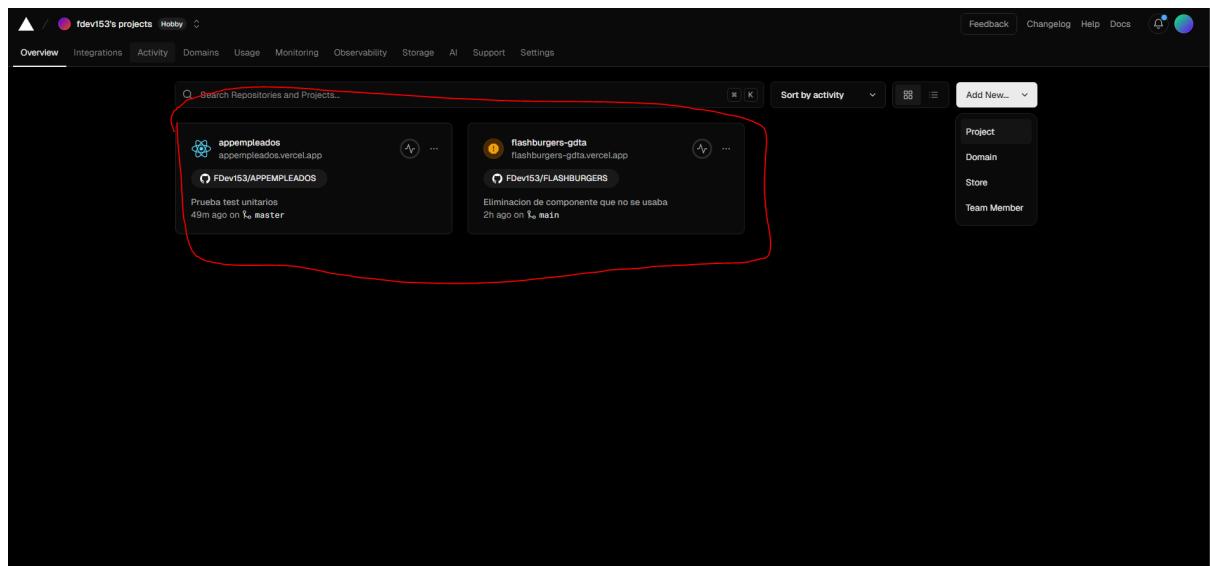
Configuraremos el despliegue

Comando de instalacion: npm install

Comando para iniciar la construccion: npm run build



Finalmente tendremos nuestro proyecto desplegado



Librerías utilizadas

React:

react-dom: Proporciona métodos específicos para interactuar con el DOM en el navegador.

react-router-dom: Permite el enrutamiento en aplicaciones de una sola página con React.

reactstrap: Componentes de UI basados en Bootstrap para React.

react-scripts: Proporciona scripts y configuraciones predeterminadas para aplicaciones React creadas con Create React App.

react-hot-toast: Utilizada para mostrar notificaciones tipo "toast" (pequeñas alertas que aparecen temporalmente en la pantalla).

Test unitarios:

@testing-library/react: Proporciona utilidades para realizar pruebas de componentes React, permitiéndote renderizarlos, interactuar con ellos y hacer aserciones sobre su comportamiento y su estructura.

@testing-library/jest-dom: Extiende Jest con aserciones adicionales para facilitar las comprobaciones sobre el DOM, como si un elemento está presente, tiene una clase específica, contiene cierto texto, entre otras.

jest: Es el framework de pruebas que se utiliza para ejecutar y organizar las pruebas en JavaScript, proporcionando un entorno para ejecutar las pruebas unitarias, mocks y aserciones.

Conexion a bbdd:

axios: Para realizar solicitudes HTTP, utilizada en el código para interactuar con APIs o servicios externos

Estilos

sweetalert2: Una librería para mostrar alertas y notificaciones personalizadas.

5.3. Elaboración de las especificaciones hardware del sistema Indicar las especificaciones hardware del sistema. Indicar procesador (tipo/velocidad), memoria (tipo/velocidad/cantidad), periféricos (cámara Web, impresora...). En el caso de necesitar un sistema servidor/cliente, se debe indicar las necesidades tanto del servidor como de las estaciones cliente, y las características de la red.

Procesador:

Tipo: Intel Core i5 o AMD Ryzen 5.

Velocidad: 3.0 GHz o superior.

Núcleos: 6 núcleos.

Memoria RAM:

Tipo: DDR4.

Velocidad: 2666 MHz o superior.

Cantidad: 16 GB o superior (se recomienda 32 GB para mayor fluidez en tareas concurrentes).

Almacenamiento:

Tipo: SSD (para mejorar tiempos de acceso a datos).

Capacidad: 500 GB o superior (dependiendo de la cantidad de datos que se manejen).

Conexión de Red:

Tipo: Ethernet Gigabit (para un acceso rápido y estable a la red, especialmente en servidores dedicados).

Recomendación de ancho de banda: 10 Mbps o superior (dependiendo de la cantidad de tráfico esperado).

Sistema Operativo:

Windows (dependiendo de las preferencias y compatibilidad del cliente).

Periféricos:

UPS (Fuente de Alimentación Ininterrumpida): Para proteger el servidor en caso de cortes de energía.

6. ESPECIFICACIONES DEL SOFTWARE

6.1. Descripción de las operaciones

Se describirán las distintas operaciones que se pueden llevar a cabo en la aplicación.

Operación 1: Realizar Pedido

Nombre: Realizar Pedido

Actor: Usuario

Resultado: Un pedido se registra en la base de datos y se muestra la confirmación.

Descripción:

El usuario selecciona productos del catálogo y los agrega a la cesta de compras. Al confirmar el pedido, la información del cliente y los productos seleccionados se envía al servidor para su registro.

Parámetros:

nombreCliente (String): Nombre del cliente

telefonoCliente (String): Teléfono del cliente

direccionCliente (String, opcional): Dirección de entrega

pedidoCliente (Array): Lista de productos seleccionados

cantidadTotal (Number): Cantidad total de productos

carrito (Array): Lista detallada con ID, nombre, cantidad y precio de productos

Precondición:

El carrito debe contener al menos un producto.

Los datos del cliente (nombre y teléfono) deben estar completos.

Postcondición:

El pedido queda registrado en la base de datos con un identificador único.

El carrito se vacía y el estado se reinicia.

Excepciones:

Error en la conexión con la base de datos.

Parámetros faltantes o incorrectos.

FLASH BURGER

Hamburguesas Clásica
Jugosa hamburguesa de carne de res angus, con queso cheddar derretido, lechuga fresca, tomate, cebolla y salsa de la casa, todo en un pan brioche.

Hamburguesa Clásica
12€

Hamburguesa BBQ
Hamburguesa vegetariana a base de garbanzos y espinacas, acompañada de aguacate, rúcula, tomate y mayonesa vegana en un pan integral.

Hamburguesa BBQ
11.5€

Hamburguesa BBQ
Hamburguesa de carne de cerdo ahumada, bañada en salsa BBQ casera, con aros de cebolla crujientes y queso cheddar, servida en un pan de sésamo.

Hamburguesa BBQ
13.5€

Ver ingredientes

Ver ingredientes

Ver ingredientes

Finalizar pedido Cerrar

FLASH BURGER

Hamburguesa Clásica
Jugosa hamburguesa de carne de res angus, con queso cheddar derretido, lechuga fresca, tomate, cebolla y salsa de la casa, todo en un pan brioche.

Hamburguesa Clásica
12€

Hamburguesa BBQ
Hamburguesa vegetariana a base de garbanzos y espinacas, acompañada de aguacate, rúcula, tomate y mayonesa vegana en un pan integral.

Hamburguesa BBQ
11.5€

Hamburguesa BBQ
Hamburguesa de carne de cerdo ahumada, bañada en salsa BBQ casera, con aros de cebolla crujientes y queso cheddar, servida en un pan de sésamo.

Hamburguesa BBQ
13.5€

Ver ingredientes

Ver ingredientes

Ver ingredientes

Pedido realizado con éxito!
Confirme el pedido para continuar.

Cancelar pedido Confirmar pedido

Operación 2: Añadir Producto al Carrito

Nombre: Añadir Producto al Carrito

Actor: Usuario

Resultado: El producto se agrega o actualiza en la lista del carrito.

Descripción:

El usuario selecciona un producto del catálogo para agregarlo al carrito. Si el producto ya está en el carrito, se actualiza la cantidad y el precio total.

Parámetros:

nombre (String): Nombre del producto

precio (Number): Precio unitario del producto

cantidad (Number): Cantidad seleccionada

id_pro (Number): Identificador del producto

Precondición:

El producto debe existir en el catálogo.

Postcondición:

El carrito contiene el producto con la cantidad actualizada.

El precio total del carrito se actualiza.

Excepciones: Error en la actualización del estado del carrito.

FLASH BURGER

The screenshot shows a user interface for an online food ordering platform. On the left, there's a sidebar with categories: Entrantes (with an icon of a sandwich), Hamburguesas (with an icon of a burger), Bebidas (with an icon of a drink), Postres (with an icon of a cake), and Salsas (with an icon of condiments). The main area displays a grid of six food items with their details and a 'Ver ingredientes' (View ingredients) button. A success message at the top right says 'Producto añadido con éxito' (Product added successfully). A shopping cart icon in the top right corner shows a red notification.

Imagen	Nombre	Descripción	Precio	Opciones
	Croquetas de Jamón	Deliciosas croquetas caseras rellenas de jamón ibérico, acompañadas de una salsa de alioli casero.	8.5€	Añadir
	Ensalada César	Mezcla de lechugas frescas, crujientes trozos de pan tostado, pollo a la parrilla, queso parmesano rallado y aderezo César.	9€	Añadir
	Calamares a la Romana	Anillos de calamar tiernos y crujientes, rebozados y fritos a la perfección, servidos con limón y mayonesa.	10.5€	Añadir

Operación 3: Cancelar Pedido

Nombre: Cancelar Pedido

Actor: Usuario

Resultado: El pedido es eliminado de la base de datos.

Descripción:

Si el usuario decide cancelar un pedido ya creado, se realiza un DELETE en las tablas correspondientes de la base de datos.

Parámetros:

id_pedido (Number): Identificador único del pedido a cancelar

Precondición:

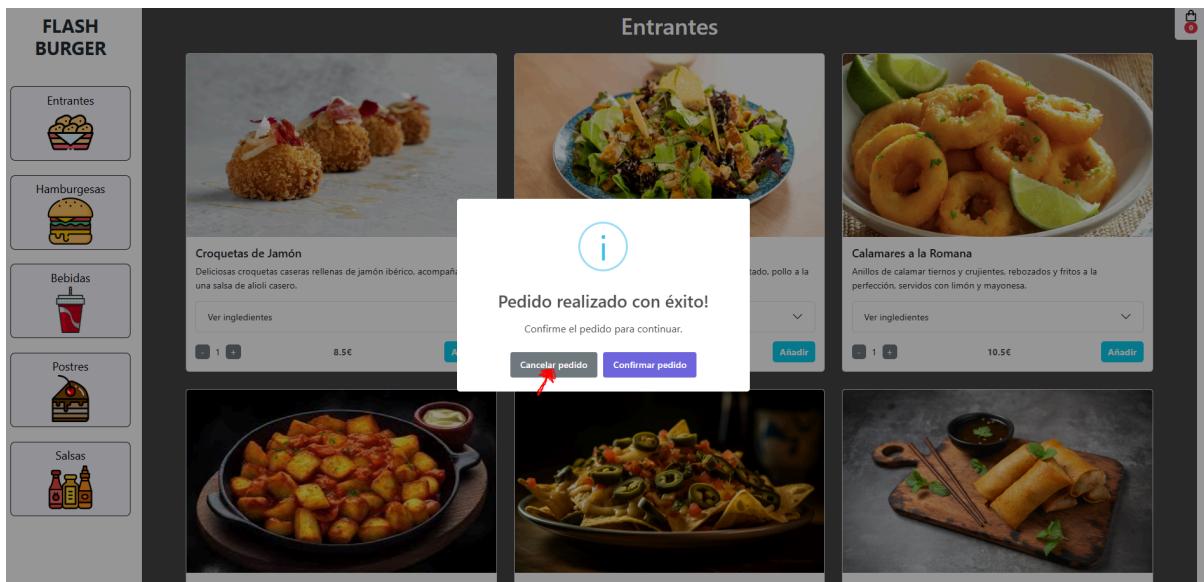
El pedido debe existir en la base de datos.

Postcondición:

El pedido y sus productos relacionados se eliminan de las tablas correspondientes.

Excepciones:

Error en la conexión con la base de datos.



Operación 4: Ver Detalles del Pedido

Nombre: Ver Detalles del Pedido

Actor: Usuario

Resultado: Se muestran los detalles de un pedido específico.

Descripción:

Al realizar un pedido, se recuperan sus datos de la base de datos, mostrando información como productos, cantidades, precio total y estado del pedido.

Parámetros:

id_pedido (Number): Identificador único del pedido

Precondición:

El pedido debe existir en la base de datos.

Postcondición:

Se muestran los detalles del pedido en una ventana de confirmación.

Excepciones:

Error en la recuperación de los datos del pedido.

FLASH BURGER

Entrantes

Hamburguesas Clásica

Jugosa hamburguesa de carne de res angus, con queso cheddar derretido, lechuga fresca, tomate, cebolla y salsa de la casa, todo en un pan brioché.

Ver ingredientes

1 12€

Hamburguesa Clásica Vegana

Hamburguesa vegana con queso cheddar derretido, lechuga fresca, tomate, cebolla y salsa de la casa, todo en un pan brioché.

Ver ingredientes

1 13.5€

i

Pedido realizado con éxito!

Confirme el pedido para continuar.

Hamburguesa de Pollo

Pechuga de pollo a la parrilla con salsa BBQ, queso cheddar, cebolla caramelizada y tocino crujiente.

Ver ingredientes

1 12€

Hamburguesa de Salmón

Pechuga de salmón ahumado, queso cheddar derretido, lechuga fresca, tomate, cebolla y salsa de la casa.

Ver ingredientes

1 13.5€

Hamburguesa BBQ

Hamburguesa de carne de cerdo ahumada, bañada en salsa BBQ casera, con aros de cebolla crujientes y queso cheddar, servida en un pan de sésamo.

Ver ingredientes

1 13.5€

FLASH BURGER

Entrantes

Hamburguesa Clásica

Jugosa hamburguesa de carne de res angus, con queso cheddar derretido, lechuga fresca, tomate, cebolla y salsa de la casa, todo en un pan brioché.

Ver ingredientes

1 12€

Hamburguesa BBQ

Hamburguesa de carne de cerdo ahumada, bañada en salsa BBQ casera, con aros de cebolla crujientes y queso cheddar, servida en un pan de sésamo.

Ver ingredientes

1 13.5€

Número de pedido: 153

La cocina empezará a preparar su pedido

Estado del pedido: **En curso**

Nombre: Fer
Teléfono: 633221231

Productos

- Croquetas de Jamón : 1
- Hamburguesa Clásica : 1

Precio: 20.5 €

Pedido: Recogida en local

Hamburguesa de Pollo BBQ

Pechuga de pollo a la parrilla con salsa BBQ, queso cheddar, cebolla caramelizada y tocino crujiente.

Ver ingredientes

1 12€

Hamburguesa Clásica Vegana

Hamburguesa vegana con queso cheddar derretido, lechuga fresca, tomate, cebolla y salsa de la casa, todo en un pan brioché.

Ver ingredientes

1 13.5€

Hamburguesa Vegana

Hamburguesa vegana a base de garbanzos, lentejas y verduras, servida con aguacate, tomate, lechuga y mayonesa vegana.

Ver ingredientes

1 13.5€

Operación 5: Actualizar Estado del Pedido

Nombre: Actualizar Estado del Pedido

Actor: Administrador (o sistema)

Resultado: El estado del pedido se actualiza en la base de datos.

Descripción:

El administrador o sistema actualiza el estado de un pedido, por ejemplo, marcándolo como "en preparación" o "finalizado".

Parámetros:

id_pedido (Number): Identificador único del pedido

Precondición:

El pedido debe existir en la base de datos.

Postcondición:

El estado del pedido se actualiza correctamente en el servidor.

Excepciones:

Error en la actualización del estado en la base de datos.

The screenshot shows the 'Empleados' section of the Flash Burgers administration interface. At the top, there are three tabs: 'Pedidos en Curso' (Orders in Progress), 'Pedidos Listos' (Ready Orders), and 'Pedidos Terminados' (Completed Orders). The 'Pedidos Listos' tab is currently selected. Below the tabs, there are four order cards, each representing a different order number (132, 133, 134, 141). Each card lists the items in the order and includes a 'A Recoger' (Pickup) button and a 'Listo' (Ready) button. A red arrow points to the 'Listo' button for Pedido N° 141.

Operación 6: Obtener Todos los Pedidos

Nombre: Obtener Todos los Pedidos

Actor: Administrador

Resultado: Se muestra una lista con todos los pedidos realizados.

Descripción:

El sistema recupera todos los pedidos existentes en la base de datos y los almacena en el estado todospedidos.

Parámetros:

Ninguno

Precondición:

La base de datos debe contener al menos un pedido.

Postcondición:

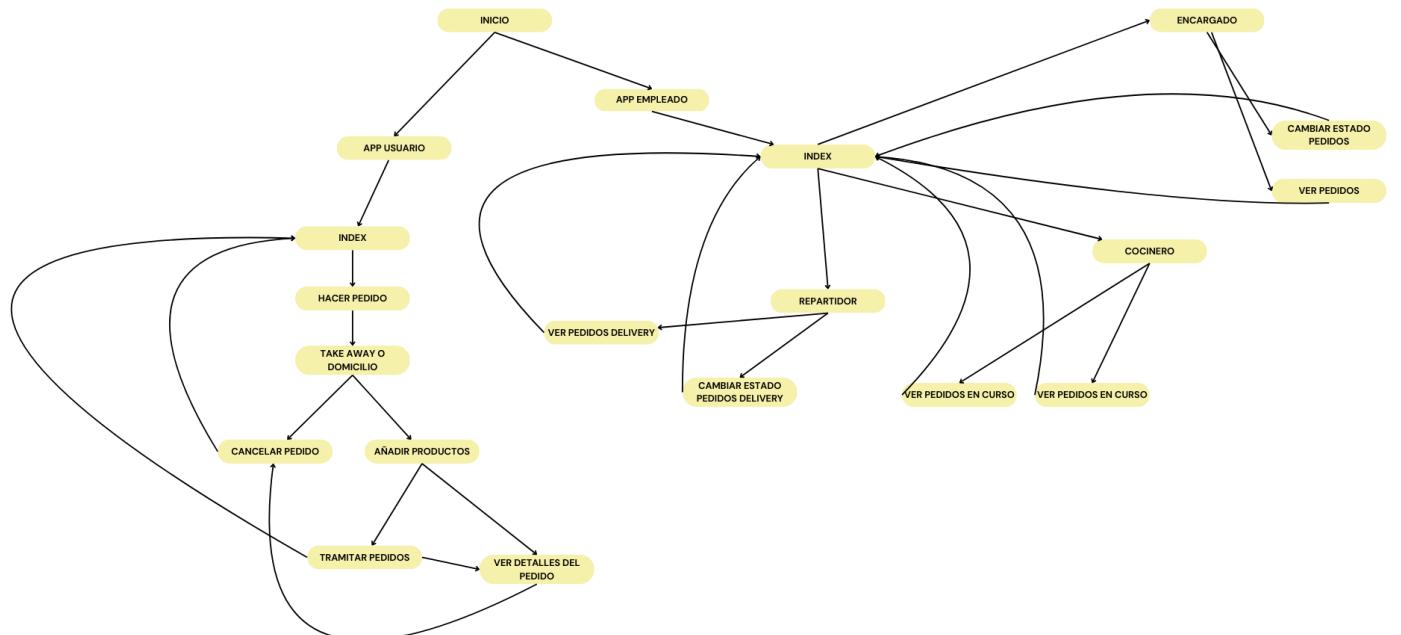
Los pedidos se muestran correctamente en la interfaz de administrador.

Excepciones:

Error en la consulta a la base de datos

Pedidos en Curso	Pedidos Listos	Pedidos Terminados
Pedido N° 135 • Ensalada César 1 A Recoger En Curs	Pedido N° 136 • Ensalada César 1 A Recoger En Curs	Pedido N° 139 • Salsa de Queso Azul 1 • Salsa de Chipotle 1 • Salsa de Yogur 1 • Salsa de Mostaza y Miel 1 • Salsa cheddar 1 A Recoger En Curs
Pedido N° 140 • Cerveza Estrella Galicia 3 A Recoger En Curs	Pedido N° 142 • Croquetas de Jamón 3 A Recoger En Curs	Pedido N° 143 • Croquetas de Jamón 1 Domicilio En Curs
Pedido N° 144 • Ensalada César 1 A Recoger En Curs	Pedido N° 145 • Croquetas de Jamón 1 A Recoger En Curs	Pedido N° 146 • Croquetas de Jamón 1 A Recoger En Curs

6.2. Interfaz y Navegación de la aplicación



7. CÓDIGO FUENTE RELEVANTE

```
Swal.fire({
  title: "Pedido realizado con éxito!",
  text: "Confirme el pedido para continuar.",
  icon: "info",
  showCancelButton: true,
  confirmButtonText: "Confirmar pedido",
  cancelButtonText: "Cancelar pedido",
}).then((result) => {
  if (result.isConfirmed) {
    this.SweetAlert2(id_pedido, datos);
  } else if (result.dismiss === Swal.DismissReason.cancel) {
    this.cancelarpedido(id_pedido);
  }
});
```

Complejidad:

- Ventana de confirmación y cancelación de pedido
- Maneja interacciones en tiempo real.
- Permite confirmar pedidos y cancelarlos, actualizando el estado en consecuencia.

```
● ● ●

// Función para manejar el clic en los enlaces del menú
const handleClick = (id) => {
  setNumero(id);
  props.menuelegir(id);
  setMenuVisible(false);
  document.body.classList.remove('menu-aberto');
  document.getElementById('hamburger-icon').classList.remove('open');
};

// Función para manejar el clic en el ícono del menú hamburguesa
const toggleMenu = () => {
  setMenuVisible(!menuVisible);
  if (!menuVisible) {
    document.body.classList.add('menu-aberto');
    document.getElementById('hamburger-icon').classList.add('open');
  } else {
    document.body.classList.remove('menu-aberto');
    document.getElementById('hamburger-icon').classList.remove('open');
  }
};

// UseEffect para detectar el redimensionamiento y cerrar el menú en cualquier cambio de tamaño
useEffect(() => {
  const handleResize = () => {
    setMenuVisible(false);
    document.body.classList.remove('menu-aberto');
    document.getElementById('hamburger-icon').classList.remove('open');
  };
  // Agregar el event listener para el resize
  window.addEventListener('resize', handleResize);

  // Limpiar el event listener cuando el componente se desmonte
  return () => {
    window.removeEventListener('resize', handleResize);
  };
}, []);


```

Complejidad:

- Gestiona manualmente la visibilidad del menú hamburguesa mediante clases CSS, lo que requiere sincronización constante entre el estado React y el DOM.
- Controla la apertura y cierre del menú con un estado local (menuVisible), lo que implica lógica condicional para alternar y aplicar estilos dinámicos.
- Escucha cambios en el tamaño de la ventana para cerrar el menú automáticamente, asegurando una experiencia responsiva y limpiando el evento al desmontar el componente.

```
● ● ●

añadirLista = (nombre, precio, cantidad, id_pro) => {
  const produIndex = this.state.carrito.findIndex(item => item.nombre === nombre);
  toast.success('Producto añadido con éxito');

  if (produIndex !== -1) {
    // Producto ya existe en el carrito
    this.setState(prevState => {
      const nuevaCantidad = prevState.carrito[produIndex].cantidad + cantidad;
      const nuevoPrecio = precio * nuevaCantidad;

      return {
        carrito: prevState.carrito.map((item, index) => {
          if (index === produIndex) {
            return { ...item, precio: nuevoPrecio, cantidad: nuevaCantidad };
          }
          return item;
        }),
        precioTotal: prevState.precioTotal + (precio * cantidad),
        cantidadTotal: prevState.cantidadTotal + cantidad
      };
    }, () => {
      console.log('Carrito actualizado:', this.state.carrito); // Asegurando que se imprima el estado
      actualizado
    });
  } else {
    // Producto no existe en el carrito, añadir nuevo
    const precioProducto = precio * cantidad;
    this.setState(prevState => ({
      carrito: [...prevState.carrito, { id_producto: id_pro, nombre: nombre, precio: precioProducto,
      cantidad: cantidad }],
      precioTotal: prevState.precioTotal + precioProducto,
      cantidadTotal: prevState.cantidadTotal + cantidad
    }), () => {
      console.log('Carrito actualizado:', this.state.carrito); // Asegurando que se imprima el estado
      actualizado
    });
  }
}
```

Complejidad:

- Comprueba si un producto ya existe en el carrito y, si es así, actualiza cantidad y precio.
- Si no existe, lo añade como un nuevo elemento.
- Gestiona simultáneamente el estado global del carrito y realiza cálculos dinámicos del precio y cantidad

8. CONCLUSIONES DEL PROYECTO

Temporalización final	
Módulo	Total Horas
Página web Principal	10
Creación de la base de datos	4
Menú de Productos	10 -> 11
Sistema de pedidos	40 -> 42
Información del pedido	8
Confirmación del pedido	8
Cancelación del pedido	8
Gestión de pedidos	20
Actualización del Estado de pedidos	16 -> 14
Historial de pedidos	16 -> 15
Implementación de github actions	5h
Implementación de test unitarios	3h
Total:	140 -> 148 Horas

En el proyecto se añado la implementación de github actions y la implementación de test unitarios los cuales finalmente fueron añadidas a la temporalización final ya que ocupan bastante tiempo

En futuras versiones y modificaciones se mejorará:

- La interfaz visual para que sea más clara y atractiva para el usuario.
- El funcionamiento del servidor.
- Velocidad de carga de la aplicación web.

9. APÉNDICES

Pruebas realizadas:

App principal:

- Comprobación de si se ha renderizado el contenido del componente principal “Apppedidos” para que no este vacío
- Verificar que el carrito muestra correctamente un precio inicial de 0 cuando no tiene ningún producto
- Verificar si en el catálogo se muestra tanto el nombre de la categoría como los productos correspondientes
- Verificar en el menú que se encuentran las categorías correspondientes con el título de la empresa
- Verificar que los pedidos se muestren correctamente
- Comprobar si el producto se renderiza correctamente y el listado de ingredientes está cerrado inicialmente

App empleados:

- Comprobación de si se ha renderizado el contenido del componente principal “Login” para que no este vacío
- Verificar en “vistacocinero” si se muestran los pedidos y se renderiza el título principal
- Verificar en “vistarepartidor” si se muestran los pedidos , se muestran los productos y se renderiza el título principal
- Verificar en “vistaencargado” si se muestran los pedidos , se muestran los productos y se renderiza el título principal

10. BIBLIOGRAFÍA

Iconos:

<https://www.flaticon.es/>

<https://boxicons.com/>

Imágenes:

<https://www.freepik.com>

Imagenes de código:

<https://carbon.now.sh>

Consulta de problemas:

<https://stackoverflow.com/>

Despliegue:

<https://vercel.com/>

<https://www.alwaysdata.com/en/>