SISTEMAS DE LA INFORMACIÓN
3º INGENIERÍA INFORMÁTICA
PRÁCTICA 5

846088 ABEL ROMEO LANCINA
841723 DIEGO ROLDÁN URUEÑA
841972 PABLO MORENO MUÑOZ

ÍNDICE

Introducción

Esta memoria forma parte de la práctica 5 de Sistemas de la Información de 3º de Ingeniería Informática.

El presente documento describe el diseño y desarrollo de un sistema de información orientado a satisfacer las demandas de suministro confiable de componentes informáticos para una amplia gama de clientes. A través de este sistema, se pretende abordar las necesidades de suministro confiable, variedad de productos y eficiencia de costes, para atender tanto a particulares como a empresas de todos los tamaños, incluyendo fabricantes y ensambladores de computadoras.

La aplicación está disponible en http://pcarp.store.

Objetivos y Alcance funcional de la aplicación

Necesidad a Resolver

Suministro Confiable

En un mundo cada vez más digital, las empresas dependen en gran medida de la tecnología para operar de manera eficiente. Nuestra empresa se compromete a abordar la necesidad de suministros confiables de componentes de ordenadores que cumplan con los más altos estándares de calidad. Esto permitirá a las organizaciones mantener un rendimiento óptimo de sus sistemas informáticos sin interrupciones costosas.

Variedad de Productos

Reconocemos que las empresas tienen necesidades tecnológicas diversas. Por lo tanto, ofreceremos una amplia gama de productos, desde procesadores de última generación hasta unidades de almacenamiento de alto rendimiento y tarjetas gráficas avanzadas. Nuestra variedad de productos garantizará que nuestros clientes encuentren soluciones específicas para sus requisitos únicos.

Nuestros suministros tienen un amplio rango de público objetivo ya que podemos ayudar a suministrar a empresas de cualquier categoría ya sean grandes empresas con las últimas tecnologías, como también suministrar a PYMES con elementos de gama media, también suministrar a particulares con los requisitos individuales.

Eficiencia de Costes

La gestión eficiente de los costes es esencial para el éxito empresarial. Nuestra empresa se compromete a ayudar a las organizaciones a optimizar sus presupuestos de tecnología al ofrecer precios competitivos y opciones personalizadas que se adapten a sus necesidades y metas financieras.

Público Destinatario

Particulares

Nuestro público objetivo principal incluye a profesionales de tecnología de la información y administradores de sistemas que buscan fuentes confiables de componentes de ordenadores.

Empresas de Todos los Tamaños

Esta oferta está diseñada para satisfacer las necesidades de empresas de cualquier tamaño, desde startups y pymes hasta grandes corporaciones. Se adaptarán las soluciones en función de la dimensión de las empresas que contacten. Las más pequeñas se gestionarán como un cliente final habitual, por otro lado se tramitarán de forma privada los pedidos de las empresas más grandes, pues se ajustará un presupuesto para cada cliente.

Fabricantes y Ensambladores de Computadoras

Se dispone de otra sección en la empresa donde ensamblan sus propias computadoras, brindando componentes de alta calidad para sus operaciones de ensamblaje y producción.

Origen de la Información

Proveedores Confiables

Se establecerán relaciones estratégicas con fabricantes líderes de componentes de ordenadores en todo el mundo. Esto permitirá acceder a productos de primera calidad y mantener un flujo constante de suministros.

Investigación de Mercado

Se utilizarán datos de investigación de mercado actualizados para comprender las tendencias emergentes y las necesidades cambiantes de los clientes. Esta información ayudará a adaptar y expandir el catálogo de productos de manera eficiente.

Feedback de Clientes

Se valorará el feedback y las opiniones de los clientes para mejorar constantemente nuestros productos y servicios, de tal forma que se establecerán canales de comunicación abiertos para recopilar comentarios y responder de manera proactiva a las necesidades de nuestros clientes.

Información a Ofrecer

Catálogo de Productos Completo

Nuestro catálogo de productos será exhaustivo e incluirá procesadores, tarjetas gráficas, unidades de almacenamiento, placas base, memorias RAM y mucho más. Cada producto estará acompañado de descripciones detalladas y especificaciones técnicas, así como el precio de cada uno.

Precios Competitivos y Ofertas Especiales

Ofreceremos precios altamente competitivos y promociones especiales para garantizar que nuestros clientes obtengan un producto de alta calidad y sea una gran inversión a futuro.

Objetivos

Satisfacción del Cliente

El principal objetivo será asegurar la satisfacción del cliente en cada etapa de su experiencia con nosotros. Implementaremos encuestas de satisfacción

y mecanismos de retroalimentación para medir y mejorar continuamente la calidad de nuestros servicios.

Crecimiento Sostenible

Buscamos un crecimiento constante y sostenible en el mercado. Esto incluirá la expansión de nuestra cartera de productos.

Además gracias a la colaboración de nuestra otra empresa ensambladora de ordenadores, obtendremos un público más amplio aún, acercándonos a un público más casual interesado en un compra más cómoda del producto final ensamblado

Rentabilidad

La rentabilidad es esencial para garantizar nuestra capacidad de inversión en investigación y desarrollo, así como en la mejora de nuestros servicios. Nos esforzamos por mantener una operación rentable a largo plazo.

Perfiles de usuarios que interactúan con el sistema

El Administrador, como su nombre lo indica, es el encargado de gestionar y supervisar el sistema. Para acceder a su panel de control, debe iniciar sesión como administrador. Una vez autenticado, el Administrador tiene una amplia gama de acciones a su disposición. En primer lugar, puede administrar el catálogo de productos disponibles en la plataforma de venta. Esto implica la capacidad de agregar nuevos productos al inventario, eliminar productos que ya no estén disponibles o actualizar la información de productos existentes. Esta información puede incluir el nombre del producto, su precio, una descripción detallada, la lista de productos compatibles y la estética del producto. Esta flexibilidad permite mantener actualizado y relevante el catálogo de productos para satisfacer las necesidades cambiantes del mercado y de los clientes.

Por otro lado, el administrador puede gestionar los pedidos existentes en la base de datos. De esta forma, es capaz de modificar el estado de los mismos, el cual varía entre “solicitado”, “enviado” y “entregado”, y puede eliminar pedidos existentes. Una vez modificado esto, los cambios se verán presentes en los datos de la cuenta de cada usuario registrado.

Por otro lado, el Cliente, que puede ser tanto un particular como una empresa (sin diferencia entre ambos), tiene acceso a la plataforma en una capacidad diferente que el Administrador. Algunos clientes pueden optar por registrarse en el sistema, lo que les proporciona ciertas ventajas, como la capacidad de realizar compras y recibir un seguimiento más personalizado de sus actividades en la plataforma. Los clientes registrados pueden explorar el catálogo de productos, comparar opciones y, lo más importante, realizar compras de productos que les interesen. La plataforma garantiza que solo los clientes registrados tengan la capacidad de completar transacciones, lo que mejora la seguridad y la trazabilidad de las operaciones.

En resumen, el sistema de usuario de la plataforma reconoce dos roles fundamentales: el Administrador, encargado de la gestión de productos y la administración del sistema, y el Cliente, que puede ser tanto un particular como una empresa, con la capacidad de explorar y comprar productos. A pesar de estas diferencias, el objetivo principal es proporcionar una experiencia excepcional a todos los usuarios y garantizar que cada uno encuentre lo que busca en la plataforma.

Funcionalidades del sistema

El objetivo principal de la aplicación es vender componentes de ordenadores a los clientes finales, por lo tanto ha de ser una aplicación sencilla para que todos los clientes puedan entenderla y hacer compras.

Para ello, el sistema debe permitir ordenar y filtrar los distintos productos del sistema por marca, modelo, precio, estética, prestaciones, … También debe mostrar una lista detallada de los componentes en la cesta de compra. Se deberá mostrar también esta cesta de la compras, con los productos seleccionados para comprar. El sistema debe ser capaz de gestionar el pago

de una manera intuitiva como ya hacen la mayoría de servicios de compras online.

Por otro lado, el administrador tiene una interfaz completamente diferente a la del usuario convencional, pues consta de dos zonas; la primera permite ver todos los productos, y modificarlos como desee, así como añadir nuevos productos o eliminar los obsoletos. La segunda zona es la de los pedidos, en la que se muestran todos los pedidos de los usuarios y este puede modificar su estado entre solicitado, enviado o entregado, según el pedido.

Entidades de información

En una empresa de componentes de ordenadores, existen diversas entidades de información que se necesitan gestionar de forma persistente. Estas entidades incluirían, entre otras:

- Productos y componentes: Información detallada sobre cada producto que ofrecemos, incluyendo especificaciones técnicas, precios, disponibilidad, marcas y modelos.

- Inventario: Datos sobre el stock disponible de cada componente, incluyendo la cantidad en existencia y la ubicación en almacén.

- Pedidos: Registros de todas las transacciones de compra realizadas por los clientes, incluyendo productos adquiridos, fechas de compra, precios y métodos de pago.

- Usuarios registrados: Información de los usuarios registrados en el sitio web, incluyendo nombres de usuario, contraseñas (encriptadas), preferencias de configuración y detalles de la cuenta.

- Información financiera: Datos relacionados con transacciones financieras, como ingresos, gastos, cuentas por cobrar y cuentas por pagar.

La captura, almacenamiento y devolución de esta información serían esenciales para gestionar eficazmente la empresa, optimizar las operaciones, brindar un servicio de calidad a los clientes y tomar decisiones informadas sobre estrategias comerciales y de marketing.

Aspectos de diseño

En el diseño del sistema, es crucial considerar varios aspectos adicionales para garantizar su eficiencia, seguridad y facilidad de uso. Algunos de estos aspectos clave incluyen:

1. Seguridad:

- Autenticación y Autorización: Implementar un sólido sistema de autenticación de usuarios para garantizar que solo los usuarios autorizados puedan acceder a las funciones de administración y realizar compras. Esto podría incluir la verificación de dos factores para cuentas de alto valor.
- Protección de Datos: Asegurarse de cumplir con regulaciones de protección de datos.
- Seguridad de Pagos: Garantizar que las transacciones de pago sean seguras

2. Accesibilidad:

- Asegurarse de que el sistema sea usable por personas con discapacidades, cumpliendo con estándares de accesibilidad web.

3. Usabilidad:

- Diseñar una interfaz intuitiva para una fácil navegación y búsqueda de productos.
- Carrito de Compras: Facilitar la gestión del carrito de compras, permitiendo a los usuarios agregar, eliminar y modificar productos fácilmente.

4. Diseño Estético:

- Imágenes de Productos: Utilizar imágenes de alta calidad de los productos para que los usuarios puedan ver claramente los detalles de los componentes.

5. Mantenimiento y Escalabilidad:

- Diseñar la arquitectura del sistema de manera que sea escalable para manejar un posible crecimiento de la plataforma y facilite el mantenimiento.

6. Compatibilidad con Navegadores:

- Garantizar que el sistema funcione de manera consistente en diferentes navegadores web populares, como Chrome, Firefox, Safari e Internet Explorer.

7. Soporte al Cliente:

- Proporcionar información de contacto y opciones de soporte para que los usuarios puedan obtener asistencia en caso de problemas o preguntas.

8. Política de Devoluciones y Garantías:

- Proporcionar información clara sobre la política de devoluciones y las garantías para generar confianza en los compradores.

Planteamiento del sistema y storyboard de la aplicación

Funcionalidades del sistema de información web

- Gestión de productos: Permite a los administradores agregar, editar y eliminar productos y componentes de ordenadores. Los usuarios pueden ver detalles, precios y disponibilidad de productos.

- Gestión de inventario: Rastrea el stock disponible de cada componente.

- Gestión de clientes: Permite a los usuarios registrarse, iniciar sesión y gestionar sus perfiles.

- Carrito de compras: Los usuarios pueden agregar productos al carrito, ver y editar su contenido, y proceder al proceso de compra.

- Proceso de compra: Los usuarios pueden ingresar la información de envío y pago para completar la compra de productos.

- Gestión de pedidos: Los administradores pueden ver y gestionar los pedidos, cambiar su estado y generar facturas.

- Gestión de precios: Permite a los administradores gestionar los precios de compra.

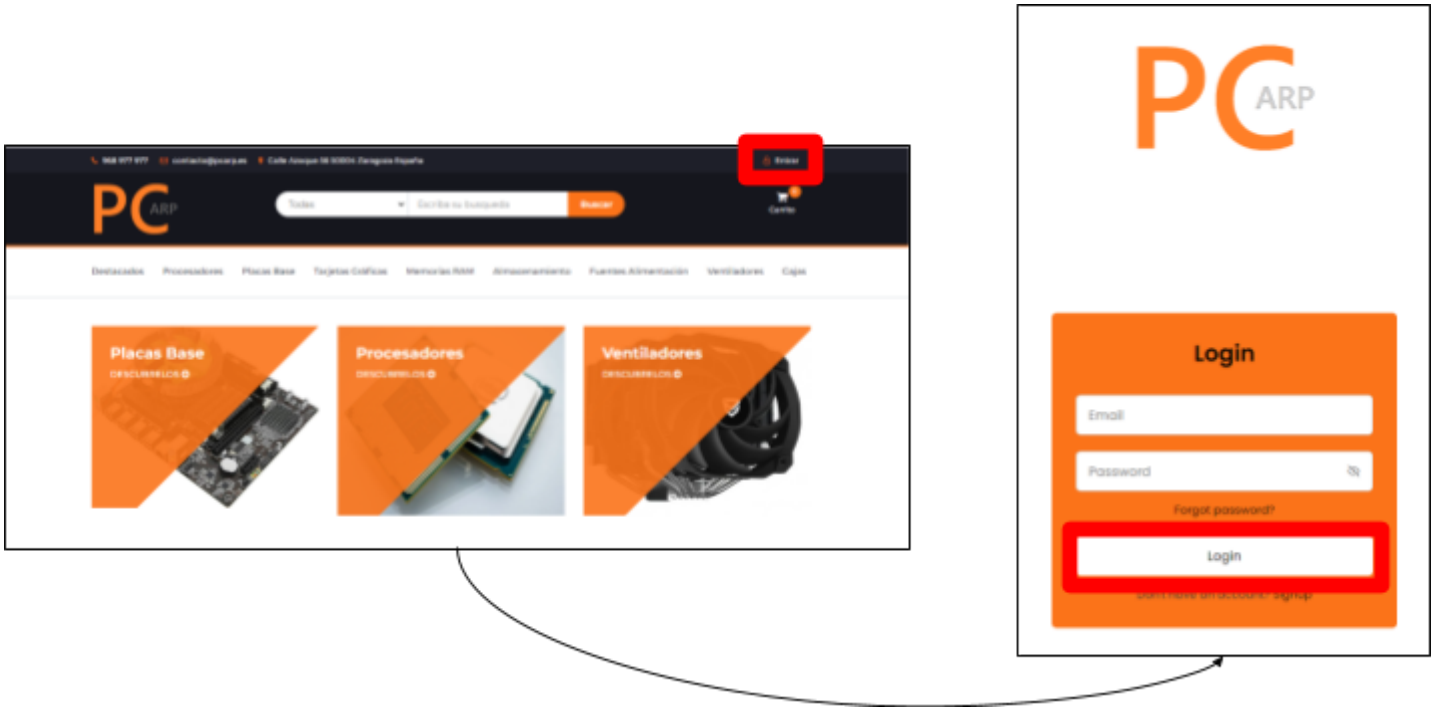
Modelo de negocio / Financiación

- Modelo de negocio: se ha optado por un modelo de negocio de venta directa de componentes de ordenadores a través del sitio web, generando ingresos por la venta de productos.

- Estrategias de ingresos: Además de las ventas directas, podrías considerar ingresos adicionales a través de publicidad en el sitio web, y a su vez se trabajará conjuntamente con una empresa gestora de ordenadores ya ensamblados.

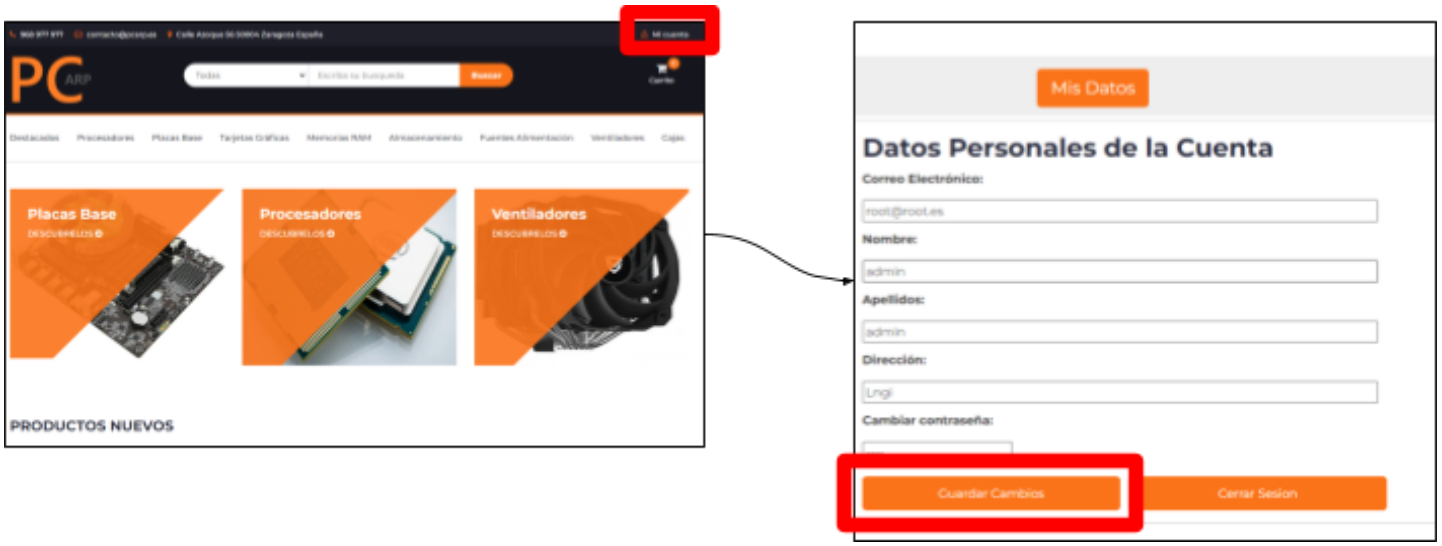
Storyboard

Crear un storyboard es un proceso creativo que implica diseñar visualmente la interfaz de usuario del sistema de información web. Como la web ya está finalizada, se ha realizado mediante capturas de pantalla del sitio, indicando en las pertinentes,qué condiciones deben darse para ello.

Historia 1: Iniciar sesión

Cuando un usuario no está registrado, aparecerá el icono “Entrar”, el cual lo redirige a la pantalla de login, desde la cual puede iniciar sesión en una existente o recuperar la contraseña de su cuenta si es que la ha olvidado.
Historia 2: Crear cuenta

Esta situación ocurre en el mismo escenario que la anterior, salvo que en este caso, el usuario no tiene una cuenta creada, por lo que debe crear una en la zona “Signup”.

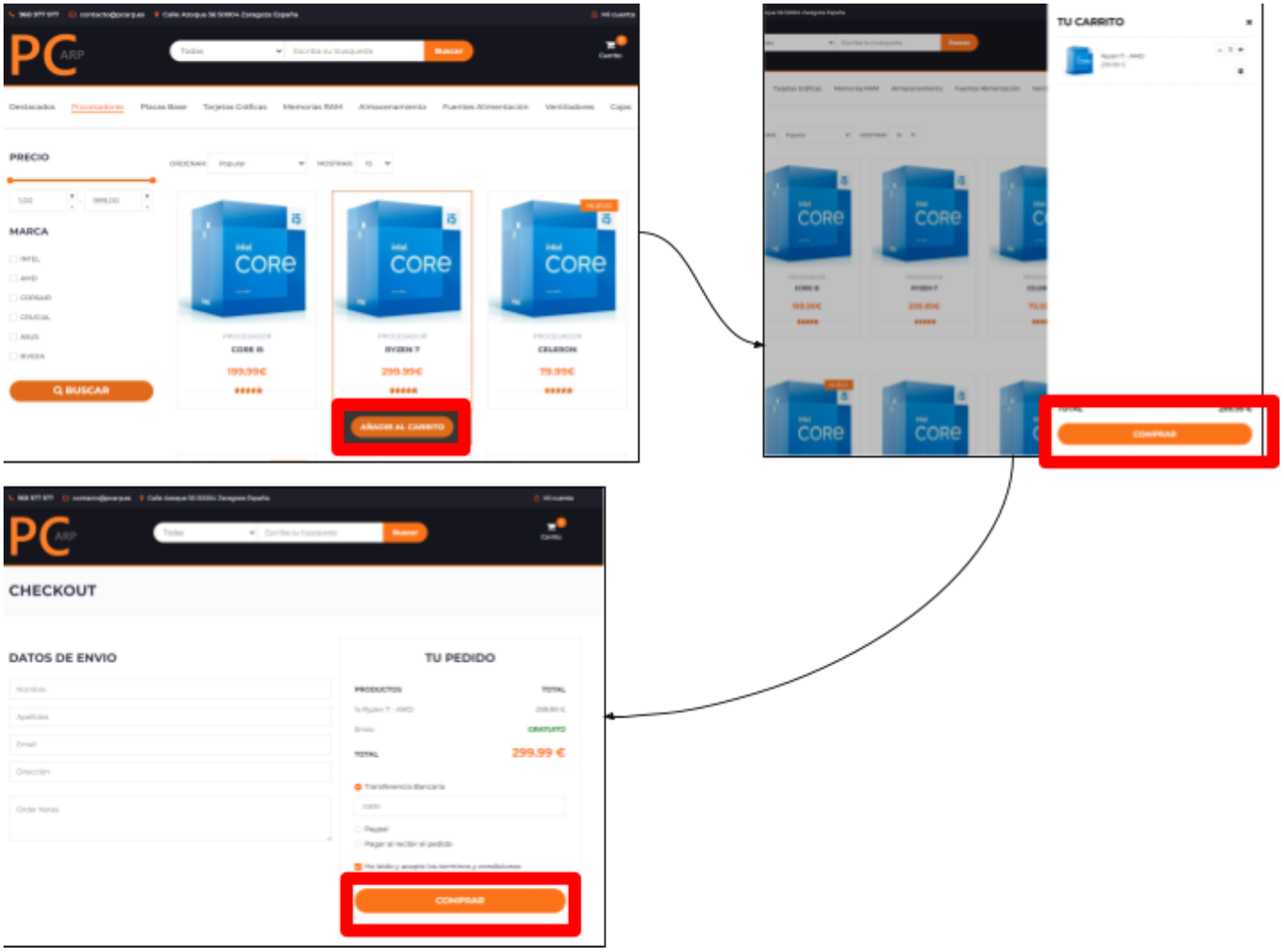
Historia 3: Cambiar atributos cuenta



Cuando un usuario está registrado en la web, en lugar de aparecer el botón “Entrar”, se mostrará el botón “Mis datos”, cuya finalidad es redirigir al usuario a una pantalla donde puede visualizar sus datos y modificarlos si desea.

Historia 4: Ver producto

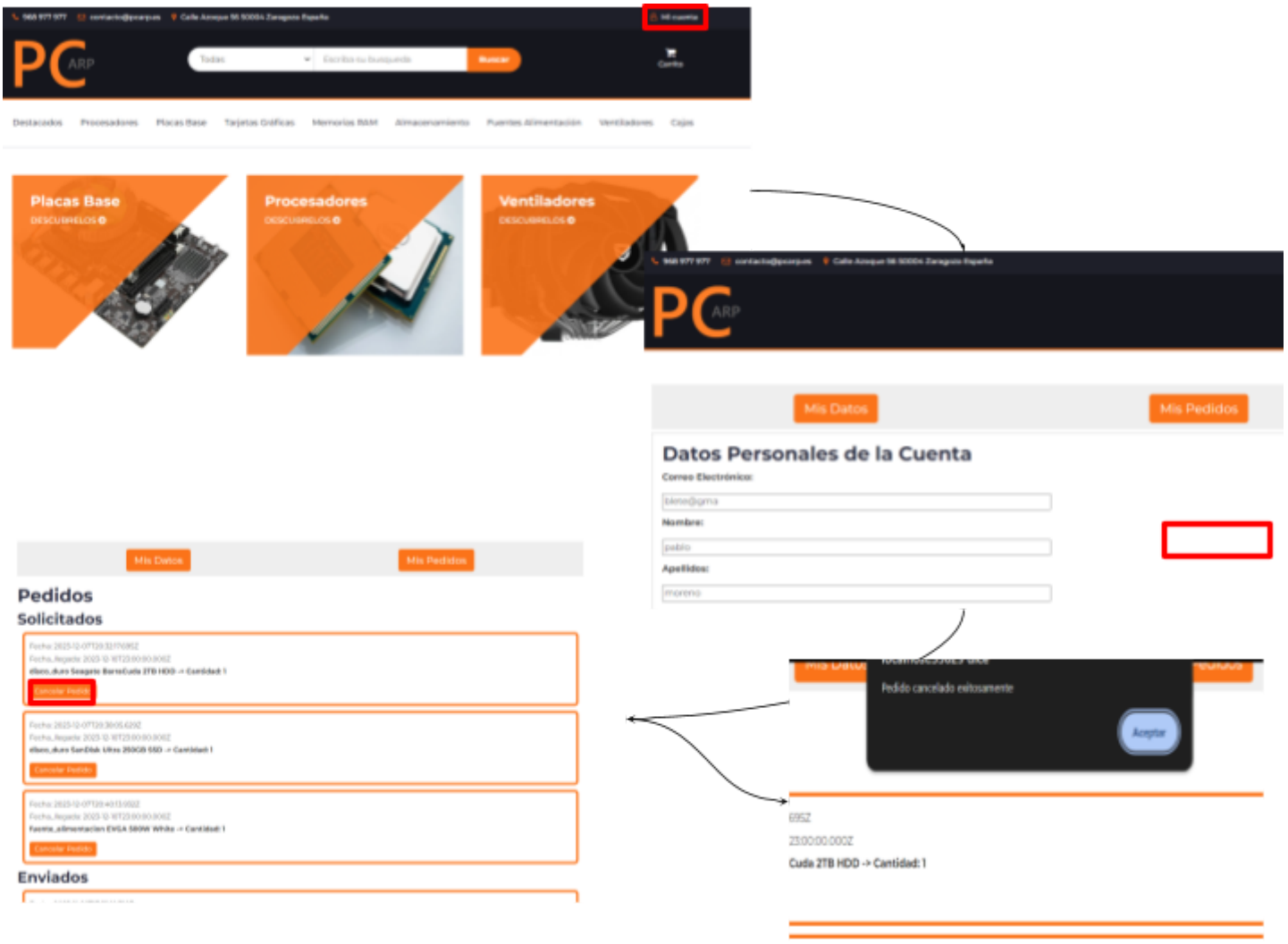
Historia 5: Comprar productos



Un usuario que ha iniciado sesión puede acceder a la interfaz de compra, desde la cual puede comprar todo aquello que haya añadido al “carrito”.

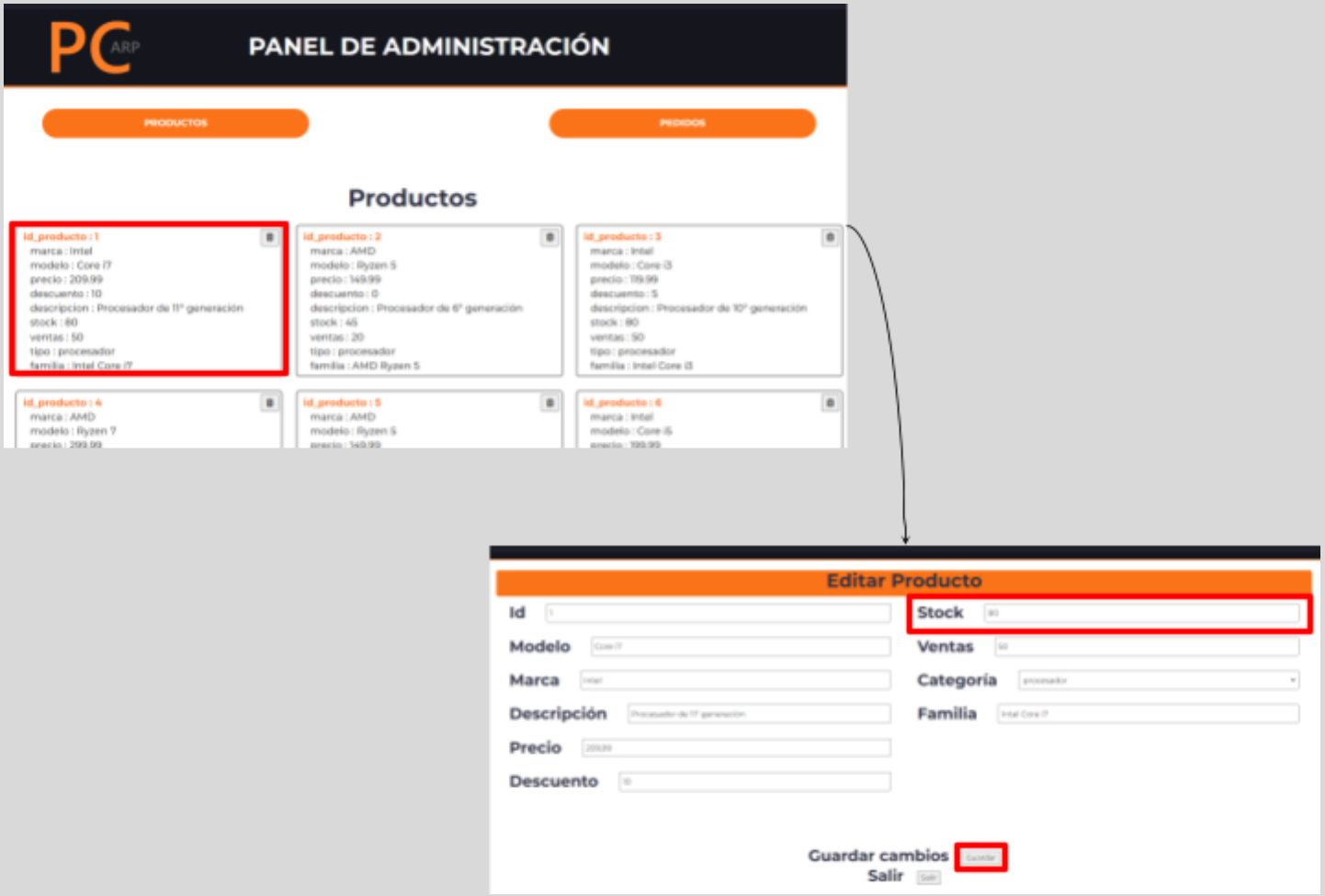
Historia 6: Ver estado pedido

Historia 7: Cancelar pedido



Cuando un pedido se encuentra en estado = “solicitado”, es posible cancelar, en cualquier otro caso no se podría.

Historia 8: Modificar stock



Historia 9: Modificar atributos de un producto

Esta historia es equivalente a la anterior, con la diferencia de que en lugar de modificar el campo stock, se modificaría cualquier otro campo del producto. El stock es un atributo de un producto

Historia 10: Añadir producto nuevo

Entorno de Desarrollo

- IDE: La decisión ha sido el usar un IDE como Visual Studio Code por su versatilidad y utilidad para el desarrollo web.

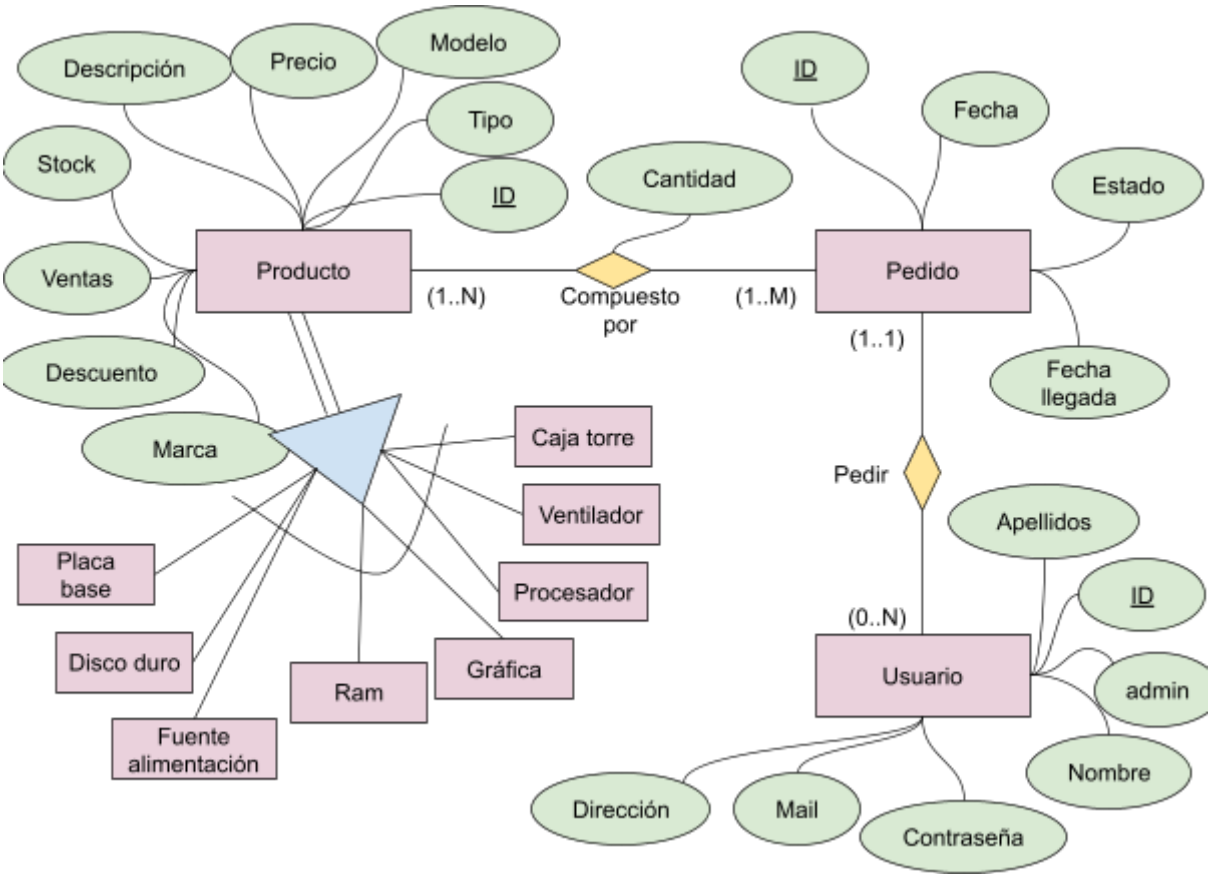
- Base de Datos: PostgreSQL ha sido nuestra elección, debido a ser un sistema de gestión de bases de datos relacional de código abierto y uno de los sistemas de gestión de bases de datos más populares en el mundo del desarrollo de aplicaciones.

- Servidor Web: Node.js es una herramienta versátil que es utilizada en una amplia variedad de aplicaciones, desde aplicaciones web en tiempo real hasta servidores de API REST y microservicios. Es un entorno de tiempo de ejecución de JavaScript de código abierto que permite ejecutar código JavaScript en el servidor.

- Herramientas de Utilidad: Se han considerado herramientas como Git para el control de versiones entre los distintos componentes del grupo.

Modelo de datos del sistema

Modelo Entidad-Relación

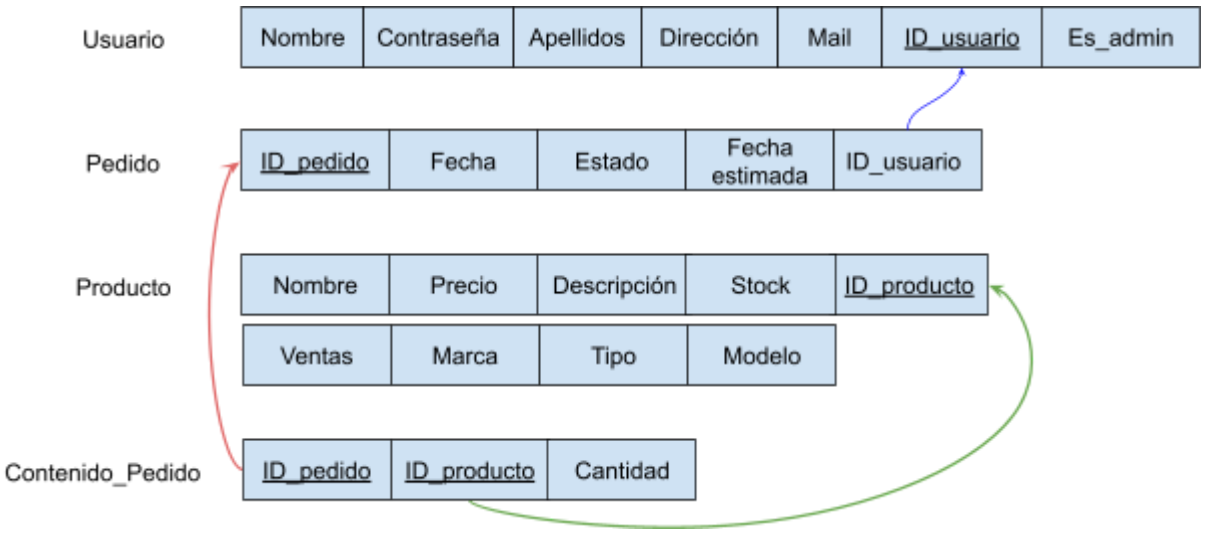


El modelo entidad-relación consta de 3 entidades principales: Producto, Pedido y Usuario, y otras 8 que derivan de producto: Placa base, Ventilador, Ram, Disco duro, Fuente alimentación, Gráfica, Caja torre y Procesador.

Todos los atributos que cuelgan de las entidades son “not null”, pues deben tener siempre un valor, y no pueden quedarse vacíos ni tener varios valores.

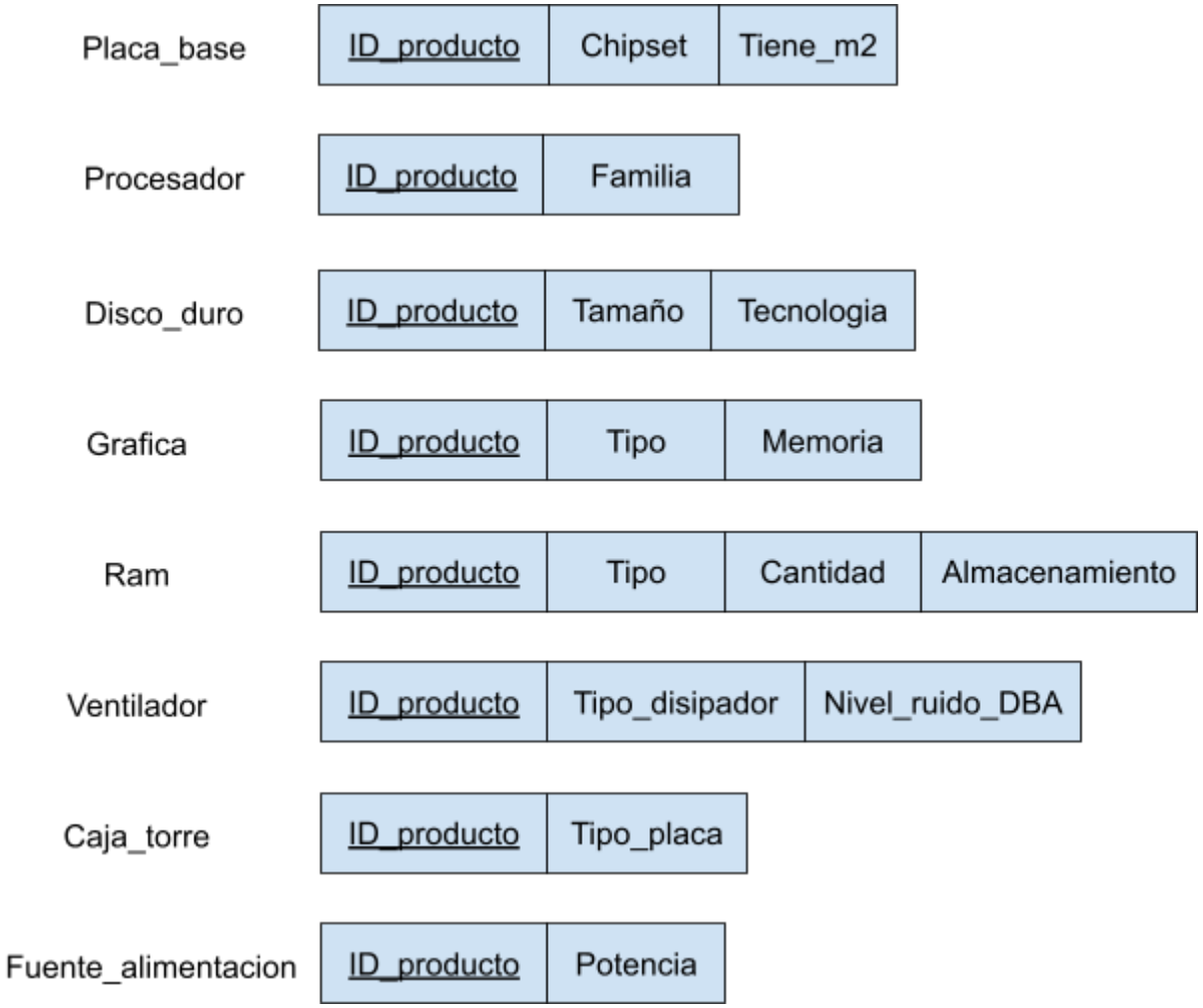
Cada especificación de la entidad Producto (Placa base, Disco duro, Procesador, …) tiene atributos específicos que no se han mostrado en el modelo entidad-relación por falta de espacio. Los atributos aparecen en la página 25 en el modelo relacional.

Modelo Relacional



En este esquema relacional traducido del modelo entidad-relación anterior se pueden observar todas las entidades que lo conforman, junto con las relaciones que tengan una relación N:M. Esto es debido a que aquellas que son del tipo 1:N, pueden establecerse en la propia tabla de la entidad cuya participación es 1.

Al contener cada tipo de producto unos atributos específicos, se ha optado por crear tablas aparte para cada tipo de producto existente, de tal forma que a las tablas anteriores se van a unir las siguientes entidades.


Placa_base
ID_producto
Chipset
Tiene_m2
Procesador
ID_producto
Familia
Disco_duro
ID_producto
Tamaño
Tecnologia
Grafica
ID_producto
Tipo
Memoria
Ram
ID_producto
Tipo
Cantidad
Almacenamiento
Ventilador
ID_producto
Tipo_disipador
Nivel_ruido_DBA
Caja_torre
ID_producto
Tipo_placa
Fuente_alimentacion
ID_producto
Potencia

Sentencias de creación creación de tablas SQL

Debido a las dependencias de atributos entre tablas, hay que seguir determinado orden en la creación de las tablas. De esta forma se ha definido el código necesario para ello:

```
Unset
CREATE TABLE usuario (
	id_usuario		SERIAL PRIMARY KEY,
	nombre			VARCHAR (20) NOT NULL,
	apellidos		VARCHAR (30) NOT NULL,
	contrasena		VARCHAR (20) NOT NULL,
	direccion		VARCHAR (30),
    mail		VARCHAR (50) NOT NULL UNIQUE,
	es_admin		BOOLEAN NOT NULL
);

CREATE TABLE pedido (
	id_pedido		SERIAL PRIMARY KEY,
	id_usuario		INT REFERENCES Usuario (id_usuario),
	fecha			TIMESTAMP NOT NULL DEFAULT current_timestamp,
	fecha_llegada	DATE CHECK (fecha_llegada>DATE(fecha)),
	estado		VARCHAR (10) NOT NULL DEFAULT 'pendiente' CHECK (estado
= 'procesando' OR estado = 'enviado' OR estado = 'entregado')
);

CREATE TABLE producto (
	id_producto	SERIAL PRIMARY KEY,
	marca			VARCHAR(25) NOT NULL,
	modelo			VARCHAR(50) UNIQUE NOT NULL,
	precio		REAL NOT NULL CHECK (precio > 0),
	descuento	REAL CHECK ((descuento >= 0 ) AND (descuento < 100)), --
en porcentaje
	descripcion	VARCHAR (70),
	stock			INT NOT NULL,
	ventas			INT NOT NULL DEFAULT 0 CHECK (ventas >= 0),
	tipo			VARCHAR CHECK (tipo='placa_base' OR
                                                tipo='procesador' OR
                                                tipo='disco_duro' OR
                                                tipo='grafica' OR
                                                tipo='ram' OR
                                                tipo='ventilador' OR
                                                tipo='caja_torre' OR
```

```
tipo='fuente_alimentacion')
);

CREATE TABLE placa_base (
	id_producto	INT REFERENCES Producto (id_producto),

	chipset			VARCHAR(15) NOT NULL,
	tiene_m2		INT NOT NULL
);

CREATE TABLE procesador (
	id_producto	INT REFERENCES Producto (id_producto),

	familia			VARCHAR(15) NOT NULL
);

CREATE TABLE disco_duro (
	id_producto	INT REFERENCES Producto (id_producto),

	tamano		INT NOT NULL,-- en GB
	tecnologia		VARCHAR(3) NOT NULL CHECK ((tecnologia='HDD') OR
(tecnologia='SSD') OR (tecnologia='M2'))
);

CREATE TABLE grafica (
	id_producto	INT REFERENCES Producto (id_producto),

	tipo		VARCHAR(6)  NULL CHECK ((tipo='NVIDIA') OR
(tipo='AMD') OR (tipo='intel')),
	memoria			INT NOT NULL -- en GB
);

CREATE TABLE ram (
	id_producto	INT REFERENCES Producto (id_producto),

	tipo		VARCHAR(4)  NOT NULL CHECK ((tipo='DDR5') OR
(tipo='DDR4') OR (tipo='DDR3')),
	cantidad		INT NOT NULL, -- cantidad de ranuras que vienen
	almacenamiento	INT NOT NULL -- GB en cada ranura
);

CREATE TABLE ventilador (
    ID INT REFERENCES Producto (id_producto),
    tipoDisipador VARCHAR(35),
    nivelRuidoDBA DECIMAL(4, 2)
);
```

```
CREATE TABLE caja_torre (
       id_producto     INT REFERENCES Producto (id_producto),
       tipo_placa VARCHAR(20)
);

CREATE TABLE fuente_alimentacion (
       id_producto     INT REFERENCES Producto (id_producto),

       potencia            INT NOT NULL -- en W
);

CREATE TABLE contenido_pedido (
       id_pedido           INT REFERENCES Pedido (id_pedido),
       id_producto         INT REFERENCES Producto (id_producto),
    cantidad        INT NOT NULL,
       PRIMARY KEY (id_pedido, id_producto)
);
```

Diseño de Value Objects (VO)

Las clases VO (Value Objects) son objetos que encapsulan un conjunto de propiedades o atributos que interactúan con una base de datos, los VO se utilizan para mapear los resultados de las consultas.

En el constructor de cada clase VO, se definen las propiedades que corresponden a los campos de la base de datos. Estas propiedades se inicializan con los valores proporcionados al crear una instancia del VO.

Cuando se recupera información de la base de datos, los resultados se mapean a objetos VO en las funciones del DAO. Los objetos VO se utilizan para estructurar los datos y facilitar su manejo en la aplicación.

```javascript
class VOUsuario {
```

```
constructor(id_usuario,nombre,apellidos,contrasena,mail,direccion,es_admin)
{
        this.id_usuario = id_usuario;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.contrasena = contrasena;
        this.direccion = direccion;
        this.mail = mail;
        this.es_admin = es_admin;
    }
}

module.exports = VOUsuario
```

Diseño de Data Access Object (DAO)

Las clases DAO (Data Access Object) proporcionan una interfaz entre la base de datos y la lógica de la aplicación. En el siguiente ejemplo, se muestra la implementación de la clase DAOusuario, que se encarga de interactuar con la tabla de usuarios en la base de datos.

JavaScript

```
const VOusuario=require ('../VO/VOusuario')

class DAOusuario {

  constructor(database) {
    this.database = database;
  }

  async obtenerTodos() {
    try {
      // Realiza la consulta a la base de datos para obtener el usuario con
el ID proporcionado
      const result = await this.database.query('SELECT * FROM usuario');
```

```
      if (result.rows.length === 0) {
        return null;
      }

      const VOusuarios = result.rows.map((usuario) => new
VOusuario(usuario.id_usuario,
usuario.nombre,usuario.apellidos,usuario.contrasena,usuario.mail,usuario.dir
eccion));
      return VOusuarios;

    } catch (error) {
      throw error;
    }
    // Implementación para obtener todos los usuarios
  }

  async obtenerPorId(id) {
    // Implementación para obtener un usuario por su ID desde la base de
datos
    try {
      // Realiza la consulta a la base de datos para obtener el usuario con
el ID proporcionado
      const result = await this.database.query('SELECT * FROM usuario WHERE
id_usuario = $1', [id]);

      if (result.rows.length === 0) {
        return null;
      }
      const usuario = result.rows[0];
      const res = new VOusuario(usuario.id_usuario,
usuario.nombre,usuario.apellidos,usuario.contrasena,usuario.mail,usuario.dir
eccion);
      return [res];

    } catch (error) {
      throw error;
    }
  }
  async obtenerPorNombre(nombre) {
    // Implementación para obtener un usuario por su ID desde la base de
datos
    try {
      // Realiza la consulta a la base de datos para obtener el usuario con
el ID proporcionado
      const result = await this.database.query('SELECT * FROM usuario WHERE
nombre = $1', [nombre]);

      if (result.rows.length === 0) {
```

```
        return null;
      }
      const usuario = result.rows[0];
      const res = new VOusuario(usuario.id_usuario,
usuario.nombre,usuario.apellidos,usuario.contrasena,usuario.mail,usuario.dir
eccion);
      return [res];

    } catch (error) {
      throw error;
    }
  }

  async insertar(usuario) {
    try {
      const query = 'INSERT INTO usuario (nombre, apellidos, contraseña,
direccion, mail VALUES ($1, $2, $3, $4, $5)';
      const values = [usuario.nombre, usuario.apellidos, usuario.contrasena,
usuario.direccion, usuario.mail];
      await this.database.query(query, values);

    } catch (error) {
      throw error;
    }
  }

  async actualizar(usuario) {
    try {
      const query = 'UPDATE usuarios SET nombre = $1, apellidos = $2,
contrasena = $3, direccion = $4, email = $5 WHERE id_usuario = $6';
      const values = [usuario.nombre, usuario.apellidos, usuario.contrasena,
usuario.direccion, usuario.mail, usuario.id_usuario];
      await this.database.query(query, values);
    } catch (error) {
      throw error;
    }
  }

  async eliminar(id) {
    try {
      const query = 'DELETE FROM usuario WHERE id_usuario = $1';
      const value= [i];
      await this.database.query(query, value);
    } catch (error) {
      throw error;
    }
  }

}
```

```
module.exports = DAOusuario
```

En la clases DAO se encapsulan las operaciones relacionadas con las tablas, como recuperar todos los atributos, buscar un usuario por su ID, insertar, actualizar y eliminar datos. Cada una de estas operaciones se implementa como un método en la clase.

Por ejemplo en la función obtenerTodos(), se realiza una consulta a la base de datos para obtener todos los usuarios. Luego, los resultados se mapean a objetos Value Object (VO) correspondientes a través del constructor de VOUsuario.

JavaScript

```
async obtenerTodos() {
    try {
      // Realiza la consulta a la base de datos para obtener el usuario con
el ID proporcionado
      const result = await this.database.query('SELECT * FROM usuario');

      if (result.rows.length === 0) {
        return null;
      }

      const VOusuarios = result.rows.map((usuario) => new
VOusuario(usuario.id_usuario,
usuario.nombre,usuario.apellidos,usuario.contrasena,usuario.mail,usuario.dir
eccion));
      return VOusuarios;

    } catch (error) {
      throw error;
    }
    // Implementación para obtener todos los usuarios
  }
```

La función obtenerPorId(id) permite buscar un pedido por su ID. Similar a obtenerTodos(), mapea el resultado a un VO de usuario y lo devuelve.

Diferencias con respecto al prototipo

Tras intentar realizar las distintas características de lo que en un principio iba a suponer la página web, se han realizado varias modificaciones en relación al diseño e implementación de la misma.

Entidades

Conforme se avanzaba en la implementación, se veía inviable el representar todas las entidades mencionadas en los anteriores diseños, por ende se han mantenido las más esenciales y desechado las más imprescindibles.

Aquellas que no forman parte de la implementación final son la de las reseñas, puesto que no se dispuso del tiempo necesario para su realización; los proveedores tampoco han sido representados debido a que se observó que era información relevante; y por último, las incompatibilidades entre productos, que al igual que las reseñas, iban a suponer un sobrecoste en el tiempo de creación y era bastante difícil mantener la escalabilidad a medida que aumentaban los productos.

Storyboard

En este apartado, los cambios son en su mayoría estéticos, y mantienen las funcionalidades prometidas en un principio, como por ejemplo, los pasos para realizar una compra, que han sido fusionados para amenizar el trabajo, pudiendo realizarla desde la zona de checkout directamente en lugar de pasar por varias pestañas.

La única modificación significativa ha sido para los usuarios que son administradores, puesto que se ha suprimido su rol como comprador, obligándolos a crearse otra cuenta si es que quieres realizar compras. Por lo demás, el administrador puede realizar todo aquello que se estableció en un primer momento.

Despliegue

Se ha decidido hacer el despliegue de la aplicación de forma que sea accesible desde Internet, para que sea accesible desde cualquier dispositivo sin necesidad de conocimientos extra para ejecutar docker o alguna otra herramienta.

Creación de Máquina Virtual Linode:

Iniciamos el despliegue configurando una máquina virtual en Linode, un proveedor de servicios en la nube. Seleccionamos las especificaciones necesarias para soportar nuestra aplicación, considerando aspectos como CPU, RAM y almacenamiento. De esta forma tenemos una máquina con una dirección IP pública accesible. Se puede acceder a la máquina virtual

Descarga e Instalación de Nginx:

Descargamos e instalamos Nginx en la máquina virtual recién creada. Nginx actuará como nuestro servidor web, gestionando las solicitudes HTTP y sirviendo los archivos de la aplicación.

Configuración de Nginx:

Configuramos Nginx para que sirva los archivos estáticos de la aplicación y redirija las solicitudes al servidor de aplicaciones. Establecemos los archivos de configuración, definiendo rutas y parámetros esenciales.

Descarga de PM2 y Inicio del Daemon:

Utilizamos PM2, un administrador de procesos para Node.js, para garantizar la ejecución continua de nuestra aplicación. Iniciamos PM2 como un daemon para que supervise y reinicie la aplicación en caso de fallos.

Configuración del Reverse Proxy:

Configuramos un reverse proxy en Nginx para dirigir las solicitudes http entrantes al servidor Node.js manejado por PM2. Esto establece una capa intermedia que mejora la seguridad y la eficiencia de la comunicación entre el cliente y el servidor.

Conexión con el Dominio:

Asociamos nuestro dominio con la dirección IP de la máquina virtual Linode. Configuramos un registro DNS de tipo A, apuntando el dominio a la dirección IP correspondiente, asegurándonos de que las solicitudes al dominio se dirijan correctamente a nuestra máquina virtual.

Pruebas Finales:

Realizamos pruebas exhaustivas para asegurarnos de que la aplicación funciona correctamente en el entorno de producción. Verificamos la conectividad del dominio y la capacidad de respuesta de la aplicación.

Nuestra aplicación está disponible en http://pcarp.store, accesible desde cualquier dispositivo con un navegador instalado.

Se puede acceder a la máquina virtual que hostea la aplicación por ssh al usuario root a pcarp.store utilizando la contraseña sisinf.

En caso de no funcionar correctamente el acceso a la aplicación utilizando el dominio, probar directamente con la dirección IP http://172.104.252.156.

Cuestiones necesarias para el uso de la aplicación

La web está disponible para cualquier usuario, esté registrado o no, pero no podrá utilizar todas las características de la misma. Para ello debe estar logueado con un cuenta creada con anterioridad.

Es posible crear una cuenta en cualquier momento. Primero se debe seleccionar la opción “Entrar”, arriba a la derecha de la web. Una vez allí, se pueden introducir las siguientes credenciales si es que no se desea crear un nueva cuenta:

Usuario no administrador

usuario: “luis@gmail.com”.

contraseña: “secure123”

Usuario administrador

usuario: “root@root.es”

contraseña: “toor”

Desde estos dos usuarios se es capaz de realizar todas las historias descritas en el storyboard.

Cronograma

El cronograma de desarrollo se dividió en varias fases, desde la planificación hasta la implementación y pruebas. Cada fase tuvo asignados plazos específicos para garantizar un progreso constante y evitar retrasos significativos.

Planificación y Diseño: Se dedicó tiempo a definir los objetivos, requisitos y diseño del sistema. Esto incluyó la creación de modelos de datos, interfaces de usuario y la estructura general de la aplicación.

Desarrollo: La implementación del sistema se dividió en módulos y se asignaron plazos para cada uno. Se utilizaron metodologías ágiles para adaptarse a posibles cambios en los requisitos.

Pruebas: Se reservó tiempo para pruebas exhaustivas, incluyendo pruebas de unidad, integración y pruebas de aceptación.

Despliegue: Se planificó la transición del sistema del entorno de desarrollo al entorno de producción, incluyendo la configuración del servidor y la resolución de posibles problemas de implementación.

Qué ha costado más

El desarrollo de ciertas funcionalidades específicas dentro de los jss, principalmente alguna función para mostrar las tarjetas de los productos, pedidos, etc.
También destacar la capa visual implementada en html y css, ya que ninguno dominaba este lenguaje y se ha invertido mucho tiempo buscando documentación y ejemplos.

Qué se ha aprendido

El grupo adquirió conocimientos significativos en áreas como el diseño de bases de datos, desarrollo web, implementación de servidores, configuración de entornos de producción web incluyendo nginx o reverse-proxys. También se mejoraron las habilidades de trabajo en equipo y gestión de proyectos.

Qué no se ha podido implementar

Por falta de tiempo hay funcionalidades que no se han terminado de implementar y que finalmente han sido descartadas.
En un principio fue difícil plantear correctamente las funcionalidades que dispondría la web ya que no se sabía lo que se podría llegar a lograr con las herramientas usadas para la creación de la página web.

Un apartado que se hubiese implementado de ser posible son las cookies, en las que tras un determinado tiempo de sesión sin que el navegador accediera al sitio web se destruyeran, esto se ha implementado con sessionStorage() que permite guardar variables globales entre las páginas del sitio web, pero al cerrar el navegador, se eliminan automáticamente.

Valoración grupal

La valoración general del grupo sobre el trabajo realizado fue positiva. Se lograron hitos importantes según el cronograma, y el equipo pudo superar desafíos técnicos y colaborar de manera efectiva.
Además destacar la gran cantidad de nuevas tecnologías para nosotros que hemos entendido y dominado para esta práctica y finalmente lograr una aplicación funcional y visualmente más que agradable.

Gestión de horas

Actividad	Roldán Urueña, Diego	Romeo Lancina, Abel	Moreno Muñoz, Pablo	Total
Creación de VO y DAO	4 horas	11 horas	3 horas	**18 horas**
Gestión de la base de datos	3 horas	3 horas	9 horas	**15 horas**
Creación de HTML y CSS	17 horas	15 horas	18 horas	**50 horas**
Códigos JavaScript	15 horas	10 horas	11 horas	**36 horas**
Realización de pruebas de usuario	8 horas	8 horas	8 horas	**24 horas**
Realización de la memoria	1 hora	6 horas	4 horas	**11 horas**
Despliegue	6 horas	1 hora	1 hora	**8 horas**
Total	**54 horas**	**54 horas**	**54 horas**	**162 horas**

Conclusión

En conclusión, desarrollar esta aplicación ha sido un proceso desafiante pero gratificante para el equipo. Se proporcionó una plataforma eficiente y accesible. Al diseñar el sistema se consideraron aspectos importantes como la accesibilidad, la facilidad de uso y el mantenimiento.

La implementación del despliegue en la nube ha permitido que la aplicación sea accesible desde cualquier lugar, proporcionando una experiencia de usuario fluida.

Aunque algunas funciones ideales no se pudieron implementar debido a limitaciones de tiempo, el equipo obtuvo conocimientos valiosos sobre desarrollo web, diseño de bases de datos y gestión de proyectos. La

evaluación global del grupo es positiva, reconociendo los logros alcanzados y la adquisición de nuevas tecnologías.

En conclusión, esta aplicación no sólo satisface las necesidades del mercado de suministro de componentes informáticos, sino que también refleja los esfuerzos y el compromiso del equipo para superar los desafíos y ofrecer productos funcionales y satisfactorios.

Bibliografía

Documentación oficial de Node.js:

Documentation | Node.js

Documentación oficial de Express.js:

Express.js

Documentación oficial de HTML y CSS

CSS: Cascading Style Sheets | MDN

HTML: HyperText Markup Language | MDN

Documentación oficial de Visual Studio Code:

Documentation for Visual Studio Code

Documentación del curso Sistemas de la Información 3º Ing. informática:

Moodle Unizar

Documentación sobre el uso de nginx

Nginx-Docs