**Hito 1 - Diseño y prototipo**

1. **Diseño de la interfaz gráfica Definición de la navegación entre las vistas**

**Página Principal**

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

**Registro de Usuarios**

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

**Inicio de sesión**

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

**Mi perfil publicaciones**Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

**Mi perfil carrito**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Mi perfil Listado de pedidos** Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Formulario de publicaciones** Interfaz de usuario gráfica, Sitio web

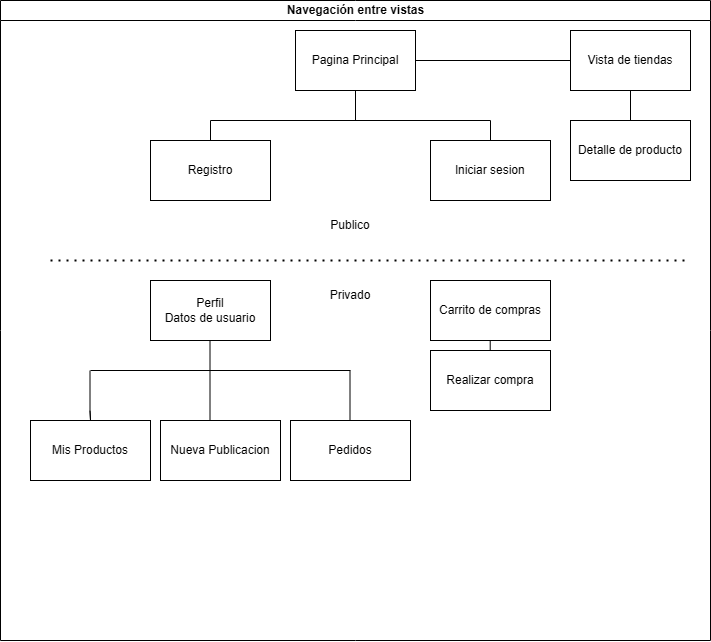
Descripción generada automáticamente **Galería de Publicaciones** Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

**Vista de detalle de una publicacion**Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

1. **Definición de la navegación entre las vistas**



1. **Listado de dependencias a utilizar en el proyecto**

**Frontend**

El frontend de este proyecto está construido con React. A continuación, se listan las dependencias principales que se necesitan instalar:

**React y Herramientas Relacionadas**

* react: Biblioteca de JavaScript para construir interfaces de usuario.
* react-dom: Paquete que permite a React interactuar con el DOM.
* react-scripts: Conjunto de scripts y configuraciones utilizadas por Create React App.
* react-bootstrap: Librería de componentes adaptadas para uso en React.
* axios: Librería para hacer solicitudes HTTP en JavaScript.
* react-bootstrap-icons: Paquete que permite usar los íconos de Bootstrap Icons como componentes de React.

**Navegación**

* react-router-dom: Biblioteca para manejar rutas en aplicaciones web construidas con React.

**Estilos**

* styled-components: Permite escribir código CSS en JavaScript para estilizar componentes de React.

**Herramientas de Desarrollo**

* prettier: Formateador de código para mantener un estilo consistente.

Para instalar todas estas dependencias, puedes ejecutar el siguiente comando en tu terminal:

npm install react react-dom react-scripts react-router-dom styled-components prettier

**Backend**

El backend de este proyecto está construido con Express.js. A continuación, se listan las dependencias principales que se necesitan instalar:

**Express y Middleware**

* express: Framework de Node.js para aplicaciones web y API.
* cors: Middleware para habilitar CORS (Compartir Recursos de Origen Cruzado).
* body-parser: Middleware para analizar el cuerpo de las solicitudes entrantes en un middleware antes de tus manejadores.
* morgan: Middleware para Express para registrar solicitudes HTTP.

**Autenticación y Seguridad**

* jsonwebtoken: Implementación de JSON Web Tokens para Node.js.
* bcryptjs: Biblioteca para hash de contraseñas.

**Base de Datos**

* pg: Cliente de PostgreSQL para Node.js.
* pg-format
* sequelize: ORM de Node.js basado en promesas para Postgres, MySQL, MariaDB, SQLite y Microsoft SQL Server.

**Herramientas de Desarrollo**

* Node: Node.js es un entorno de ejecución de JavaScript en el lado del servidor.
* nodemon: Herramienta que ayuda a desarrollar aplicaciones basadas en Node.js al reiniciar automáticamente la aplicación cuando se detectan cambios en los archivos.
* dotenv: Módulo que carga variables de entorno desde un archivo .env.

Para instalar todas estas dependencias, puedes ejecutar el siguiente comando en tu terminal:

npm install express cors body-parser jsonwebtoken bcryptjs pg sequelize nodemon dotenv

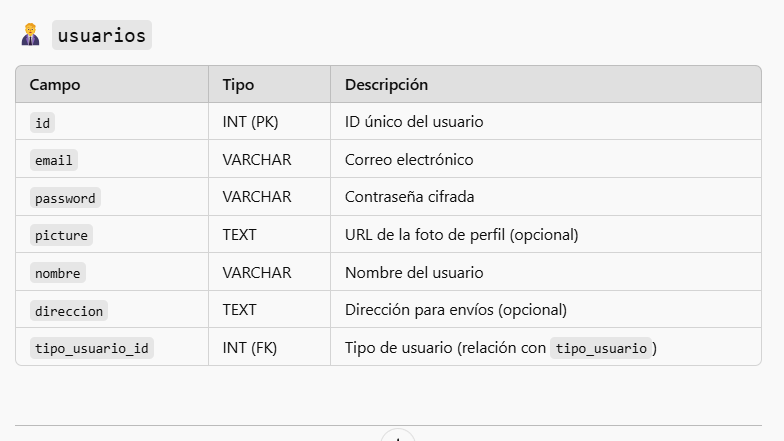
1. **Diseño las tablas de la base de datos y sus relaciones**Diagrama

   Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

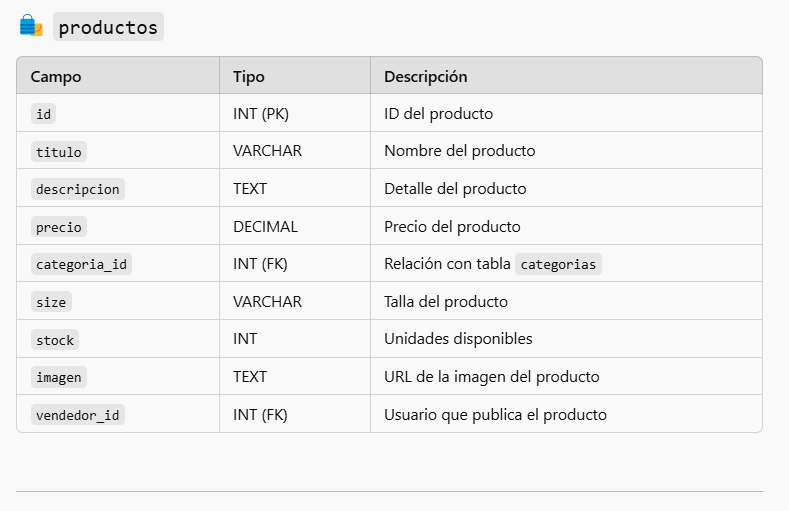
Tabla

Descripción generada automáticamente

Tabla

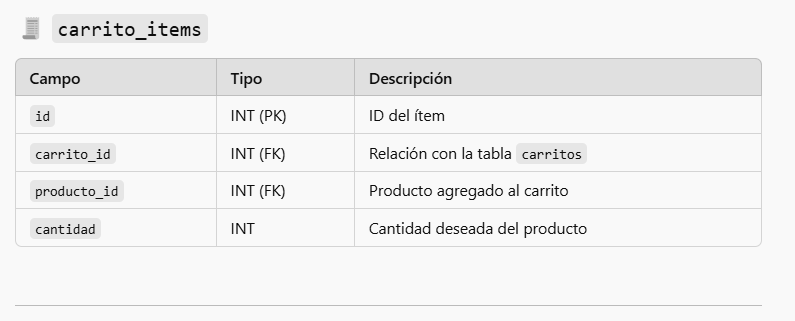
Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente





1. **Diseñar el contrato de datos de la API REST**

**📄 Contrato de Datos - API REST (con respuestas de error)**

**1. Registrar nuevo usuario**

**POST** /registro

**Request Body:**  
{  
"email": "nuevo\_usuario@ejemplo.com",  
"password": "contraseñaSegura123",  
"nombre": "María González",  
"direccion": "Avenida Principal 456, Ciudad, País"  
}

**Response (201 Created):**  
{  
"id": 1,  
"email": "nuevo\_usuario@ejemplo.com",  
"nombre": "María González",  
"direccion": "Avenida Principal 456, Ciudad, País",  
"created\_at": "2025-03-21T10:00:00Z"  
}

**Errores comunes:**

* 400 Bad Request: Datos faltantes o formato inválido  
  { "error": "El campo 'email' es obligatorio" }
* 409 Conflict: El email ya está registrado  
  { "error": "Ya existe un usuario con ese email" }

**2. Crear nuevo producto**

**POST** /productos

**Request Body:**  
{  
"titulo": "Camiseta Deportiva",  
"descripcion": "Camiseta de algodón 100%, ideal para actividades deportivas",  
"precio": 29.99,  
"categoria\_id": 2,  
"size": "M",  
"stock": 50,  
"imagen": "<https://ejemplo.com/camiseta-deportiva.jpg>",  
"vendedor\_id": 1  
}

**Response (201 Created):**  
{  
"id": 3,  
"titulo": "Camiseta Deportiva",  
"precio": 29.99,  
"stock": 50,  
"created\_at": "2025-03-21T10:05:00Z"  
}

**Errores comunes:**

* 400 Bad Request: Campos inválidos o incompletos  
  { "error": "El campo 'precio' debe ser numérico" }
* 404 Not Found: categoria\_id o vendedor\_id no existe  
  { "error": "La categoría especificada no existe" }

**3. Agregar producto al carrito**

**POST** /carrito

**Request Body:**  
{  
"usuario\_id": 1,  
"producto\_id": 3,  
"cantidad": 2  
}

**Response (201 Created):**  
{  
"id": 15,  
"usuario\_id": 1,  
"producto\_id": 3,  
"cantidad": 2,  
"subtotal": 59.98  
}

**Errores comunes:**

* 400 Bad Request: Datos inválidos o cantidad no válida  
  { "error": "La cantidad debe ser mayor a 0" }
* 404 Not Found: Usuario o producto no encontrado  
  { "error": "Producto no encontrado" }
* 409 Conflict: Stock insuficiente  
  { "error": "No hay suficiente stock disponible" }

**4. Crear nueva categoría**

**POST** /categorias

**Request Body:**  
{  
"nombre": "Electrónica"  
}

**Response (201 Created):**  
{  
"id": 2,  
"nombre": "Electrónica"  
}

**Errores comunes:**

* 400 Bad Request: Campo faltante o vacío  
  { "error": "El nombre de la categoría es obligatorio" }
* 409 Conflict: Categoría ya existente  
  { "error": "La categoría ya existe" }

**5. Crear nuevo tipo de usuario**

**POST** /tipo-usuario

**Request Body:**  
{  
"nombre": "Administrador"  
}

**Response (201 Created):**  
{  
"id": 1,  
"nombre": "Administrador"  
}

**Errores comunes:**

* 400 Bad Request: Campo vacío  
  { "error": "El nombre del tipo de usuario es obligatorio" }
* 409 Conflict: Ya existe ese tipo de usuario  
  { "error": "Tipo de usuario ya registrado" }

**6. Autenticar un usuario (Login)**

**POST** /login

**Request Body:**  
{  
"email": "usuario@ejemplo.com",  
"password": "contraseñaSegura123"  
}

**Response (200 OK):**  
{  
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
"usuario": {  
"id": 1,  
"email": "usuario@ejemplo.com",  
"nombre": "María González"  
}  
}

**Errores comunes:**

* 400 Bad Request: Falta email o password  
  { "error": "Email y contraseña son requeridos" }
* 401 Unauthorized: Credenciales incorrectas  
  { "error": "Email o contraseña incorrectos" }

**🔧 Errores generales**

**500 Internal Server Error**  
Puede aparecer en cualquier endpoint si ocurre un error inesperado en el servidor.  
{ "error": "Error interno del servidor. Intenta nuevamente más tarde." }