# **ARQUITECTURA DEL SISTEMA**

|  |  |
| --- | --- |
| **NOMBRE DEL PROYECTO** | **SIGLAS DEL PROYECTO** |
| Sistema de Gestión de Pedidos en Línea enfocado en una Cevichería | SGC |

1. **Introducción**

El Sistema de Gestión de Pedidos en Línea para una Cevichería (SGC) adopta una arquitectura Full Stack basada en el patrón de diseño MVC (Model-View-Controller). Esta arquitectura permite una clara separación de responsabilidades, mejora la mantenibilidad, escalabilidad y facilita el desarrollo modular.

1. **Componentes Principales y Tecnología**

|  |  |  |
| --- | --- | --- |
| **Componente** | **Descripción** | **Tecnología** |
| **Frontend (Vista)** | Interfaz de usuario que gestiona la experiencia e interacción. | React.js |
| **Backend (Controlador y Modelo)** | Lógica del negocio, control de rutas, acceso a datos y procesamiento de pagos. | Node.js, Express, Mongoose |
| **Base de Datos (Modelo)** | Almacenamiento persistente de datos de usuarios, productos y pedidos. | MongoDB |
| **Procesamiento de Pagos** | Gestión segura de transacciones financieras a través de un servicio externo. | Stripe API |

1. **Descripción de la Arquitectura MVC**

|  |  |  |
| --- | --- | --- |
| **Capa** | **Función** | **Componentes específicos** |
| **View** | Presenta la interfaz gráfica, recibe interacción del usuario y envía peticiones al backend. | React: Componentes UI, rutas (React Router), gestión estado (Context API) |
| **Controller** | Recibe peticiones HTTP, valida y procesa la lógica del negocio, y responde al frontend. | Express: Rutas REST, controladores, middlewares (JWT, validaciones) |
| **Model** | Representa los datos y las reglas del negocio, maneja el acceso a la base de datos. | Mongoose: Modelos de datos para Usuarios, Productos, Pedidos |

1. **Detalles Técnicos y Roles**

|  |  |
| --- | --- |
| **Componente** | **Detalle** |
| **Frontend** | * Construido con React.js * Componentes reutilizables * Gestión global del estado con Context API * Rutas para navegación con React Router * Diseño responsive para móviles y escritorio |
| **Backend** | * API RESTful construida con Node.js y Express * Manejo de rutas y controladores separados * Autenticación con JSON Web Tokens (JWT) * Validación y sanitización de datos * Integración con Stripe para pagos seguros |
| **Base de Datos** | * MongoDB con esquemas definidos en Mongoose * Colecciones: Usuarios, Productos, Pedidos * Índices para optimización de consultas frecuentes |
| **Stripe** | * Gestión de pagos desde el backend usando la API oficial * Manejo seguro de tokens y transacciones * Confirmación de pago antes de registrar pedido |

* 1. **Descripción de componentes:**

**4.1.1 Frontend – React.js**

**Estructura:**

**/components:** Botones, formularios, ítems del menú, carrito.

**/pages:** Inicio, menú, pedido, checkout, perfil.

**/assets:** Imágenes y estilos.

**/context:** Gestión de estado global con Context API.

**/hooks:** Lógica reutilizable (hooks personalizados).

**4.1.2 Backend – Node.js + Express**

**Estructura:**

**/routes:** Rutas REST para usuarios, pedidos, productos, pagos.

**/controllers:** Lógica de negocio (CRUD, validaciones).

**/models:** Modelos Mongoose (producto, pedido, usuario).

**/middlewares**: Validación, autenticación JWT, manejo de errores.

**4.3 Base de Datos – MongoDB**

**Colecciones principales:**

**Productos**: id, nombre, precio, categoría, imagen.

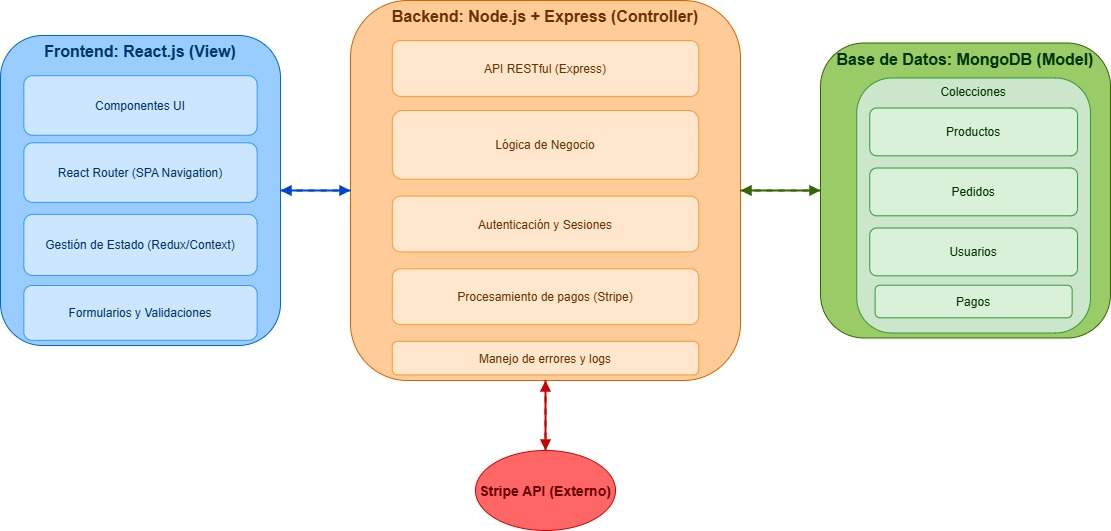
**Pedidos:** id, usuario, productos, totales, estado.

**Usuarios:** id, nombre, email, contraseña, dirección.

1. **Consideraciones No funcionales**

|  |  |
| --- | --- |
| **Requisito** | **Descripción** |
| Seguridad | Autenticación con tokens JWT, validaciones en backend, uso seguro de Stripe |
| Rendimiento | Respuestas rápidas (< 1s en peticiones comunes) |
| Disponibilidad | 24/7 mediante despliegue en servicios en la nube |
| Escalabilidad | Independencia en despliegue de capas para crecimiento futuro |
| Usabilidad | UI responsiva, intuitiva y accesible |

1. **Diagrama de la arquitectura del sistema**

****