# UNIVERSIDAD DON BOSCO
# FACULTAD DE INGENIERÍA
# ESCUELA DE COMPUTACIÓN
# DSP441

## Manual Técnico


**Desarrolladores :**

Heysel Guadalupe Argueta Hernández AH230907

Andrés de Jesus Montano Gonzalez MG230907

Jorge alexander Martinez Gonzalez MG232198

Erick Francisco Magaña Ramos MR230130

Emerson Josue Gabriel Rosales GR232330

Alisson Yasmin Vivas CastroVC230868

## Contenido

# INTRODUCCION

Bienvenido al Manual Técnico de StoreSv. Este documento proporciona una guía detallada sobre la arquitectura, tecnologías utilizadas, instalación y configuración, estructura de la base de datos, y contactos del equipo de desarrollo de la aplicación.

A lo largo de este manual, encontrarás información esencial para comprender y desarrollar StoreSv, desde su diseño arquitectónico hasta su implementación práctica. Cada sección está diseñada para brindar una comprensión completa de cómo funciona el sistema y cómo puede ser configurado y mantenido.

# Arquitectura Del Proyecto

El sistema "StoreSv" se basa en una arquitectura cliente-servidor, donde el cliente es una aplicación móvil desarrollada en React Native y el servidor es una API desarrollada y creada en Node.js.  La base de datos es la que almacenara la información del usuario tanto de los nuevos como de los que ya poseen una cuenta.

## Componentes Principales:

Cliente (Frontend):

La aplicación móvil proporciona una interfaz intuitiva para que los usuarios busquen productos, realicen compras y administren sus pedidos de una manera muy efectica.

Servidor (Backend):

La API desarrollada gestiona la lógica del negocio, procesa las solicitudes del cliente y se comunica con la base de datos.

Base de Datos:

se utilizó como base de datos relacional para almacenar cada dato de información que pueda llegarse a ingresar .

Patrones de Diseño:

Se implementa una arquitectura MVC (Modelo-Vista-Controlador) en el backend para separar la lógica de la aplicación en diferentes capas. Esto permite un mayor modularidad y facilita el mantenimiento del código .

## Escalabilidad y Mantenimiento:

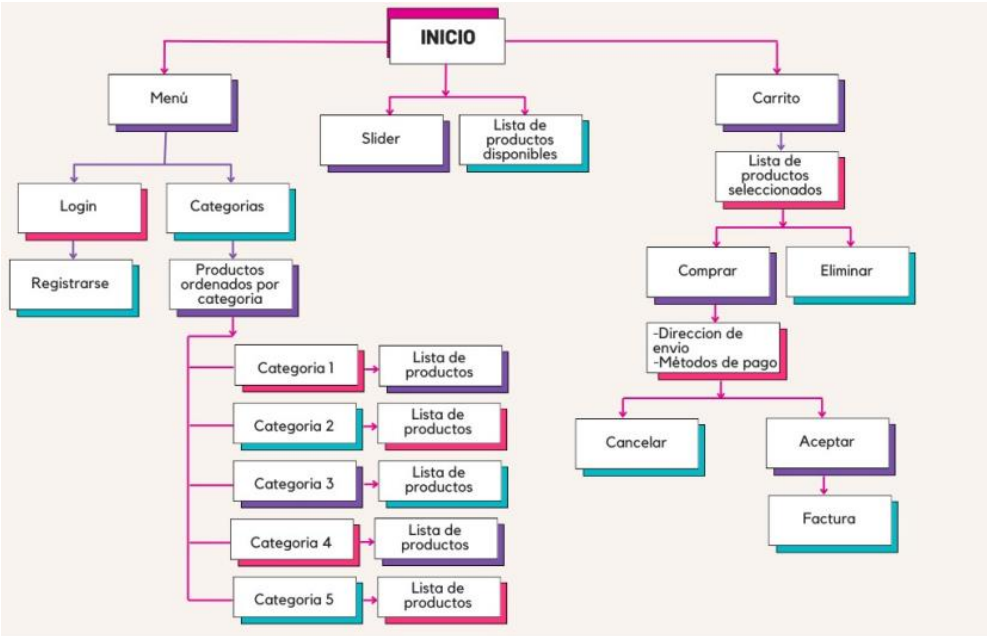
La arquitectura está diseñada para ser escalable, lo que significa que puede manejar un crecimiento en el número de usuarios y transacciones sin sacrificar el rendimiento. El modularidad del código y la separación de preocupaciones facilitan el mantenimiento a largo plazo del sistema.

Seguridad:

Se implementan medidas de seguridad como autenticación de usuarios mediante tokens JWT para proteger las solicitudes al servidor, encriptación de contraseñas en la base de datos y comunicación segura a través de HTTPS para proteger los datos del usuario durante la transmisión.

Esta arquitectura garantiza una experiencia de compra fluida y segura para los usuarios, al tiempo que proporciona una base sólida para futuras mejoras y escalabilidad del sistema.

# Diagrama de la Aplicación



# Tecnologías Utilizadas

En este caso las tecnologías que se optaron por utilizar para una mejor agilidad o desarrollo son :

## GitHub

GitHub es una herramienta esencial para los desarrolladores de software. Es utilizada por millones de personas en todo el mundo para trabajar en una amplia gama de proyectos, desde pequeños scripts hasta grandes aplicaciones empresariales. Donde ofrece una mayor administración de cada proyecto alojado en la plataforma

### ¿Por qué se escogió?

En parte se escogió por lo antes mencionado de que ofrece una mayor administración y orden a la hora de estar desarrollándolo y puedes tener el control de cada desarrollador que esta participando en el proyecto y también por que ofrece una estabilidad de alojamiento grande y de una mayor capacidad y ayuda a gestionar problemas que pueden llegar a presentarse a la hora del desarrollo.

## GitLab

GitLab es una opción popular para empresas y organizaciones que buscan una plataforma integral para el desarrollo de software

### ¿Por qué se escogió?

Fue utilizado GitLab para la gestión de configuración, lo que incluirá la configuración del entorno mediante variables de entorno y archivos de configuración.

## Notion

Notion es una herramienta poderosa que se puede utilizar para una amplia gama de propósitos, desde organización personal hasta colaboración en equipo.

### ¿Por qué se escogió?

Notion nos permitió hacer seguimiento de tareas, documentar el proyecto y colaborar de forma efectiva con el equipo de desarrollo y tener un seguimiento de cada paso desarrollado del proyecto.

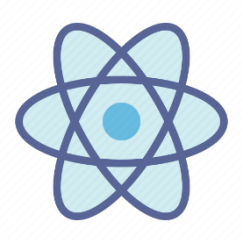
## Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para la plataforma Android, creado por Google y basado en IntelliJ IDEA de JetBrains. Está diseñado específicamente para el desarrollo de aplicaciones Android y ofrece una amplia gama de herramientas y funcionalidades para crear aplicaciones de alta calidad y funcionalidad.

### ¿Por qué se escogió?

Escogimos Android Studio para poder desarrollar y hacer pruebas en la aplicación simulando un dispositivo Android, lo que nos permitió asegurarnos de que la aplicación funcione correctamente en la plataforma Android

## React Native



React Native es un marco popular de código abierto que se utiliza para crear aplicaciones móviles nativas para iOS y Android utilizando JavaScript. Permite a los desarrolladores crear aplicaciones que se ven y se sienten como aplicaciones nativas, sin dejar de aprovechar los beneficios del desarrollo de JavaScript.

### Framework

Fue usado como framework principal para el desarrollo de la aplicación móvil, permitiéndonos crear una aplicación para Android con un código base. Siendo utilizado en el apartado del Frontend

## Node.js

Node.js es una herramienta poderosa para desarrolladores que desean aprovechar JavaScript para crear aplicaciones web y más. Su naturaleza basada en eventos, su vasto ecosistema y su compatibilidad multiplataforma lo convierten en una opción popular para el desarrollo web moderno

### ¿Por qué se escogió?

Para el desarrollo del apartado backend, fue utilizado Node.js, que nos permitió desarrollar el backend utilizando JavaScript, lo que facilito la sincronización con el Frontend y la implementación de la lógica del servidor.

## Visual Studio Code

es un editor de código fuente potente y versátil que satisface las necesidades de los desarrolladores modernos. Su naturaleza gratuita y de código abierto, combinada con sus amplias funciones y personalización, lo convierten en una opción popular para programadores de todos los niveles.

### ¿Por qué se escogió?

Como IDE (Integrated Development Environment), Visual Studio Code es altamente funcional y que ofrece numerosas características y extensiones útiles para el desarrollo de aplicaciones React Native y Node.j y tiene una gran potencia a la hora de estar desarrollando cada parte de la aplicación.

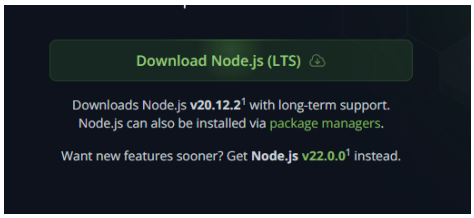
## HTTP/HTTPS.


HTTP/HTTPS se escogió como parte de nuestra API (Interfaz de Programación de Aplicaciones) para definir cómo los componentes del sistema que se comunicarán entre sí. Esto nos permitirá establecer una comunicación segura y estructurada entre los diferentes componentes de la aplicación, como el Frontend (desarrollado en React Native) y el backend (desarrollado en Node.js).

# Instalación y configuración Básica

1. **Primero paso**

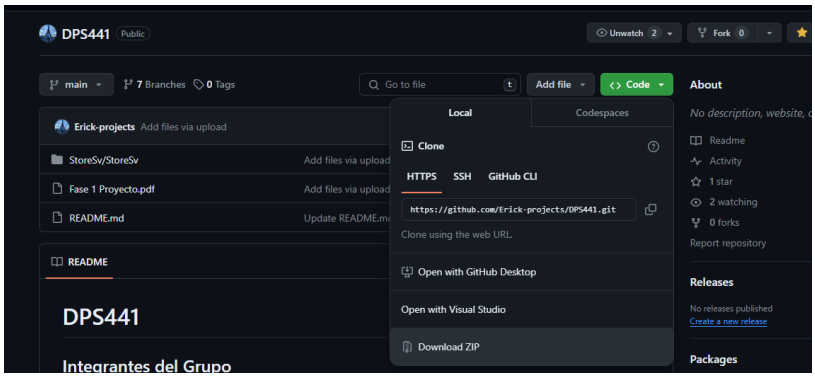
   Debes instalar node js (link directo de descarga https://nodejs.org/en)




   Tienes que darle clic al botón descargar y al terminar tienes que ejecutarlo ya al ejecutarlo tendrás el entorno de node para todo tu equipo y lo podrás utilizar en cualquier momento

2. **Segundo Paso**

   Luego tienes que irte a nuestro repositorio https://github.com/Erick-projects/DPS441 tienes que descargar carpeta que esta alojada ahí




   Si deseas clonar el repositorio directamente en tu pc solamente tienes que ejecutar en la consola de Git (git clone “**URL de nuestro repositorio**”)

3. **Tercer Paso**

   Luego podrás instalar cada dependencia utilizada con este comando este es un código básico pero principal para cada instalación

```
npm install
```

4. **Cuarto Paso**

   Para la conexión a la base de datos y que todo funcione correctamente debes de ejecutar estas dependencias.

   npm install firebase @react-native-firebase/app @react-native-firebase/auth

   npm install @react-native-firebase/app @react-native-firebase/auth

   con estas dependencias tendrás respuesta a la base de datos y su funcionalidad correcta.

5. **Quinto Paso**

   Si aún no lo has hecho, instalación de las dependencias necesarias para el desarrollo de aplicaciones móviles con React Native debes de ejecutar esta línea siempre en el CMD de tu ordenador

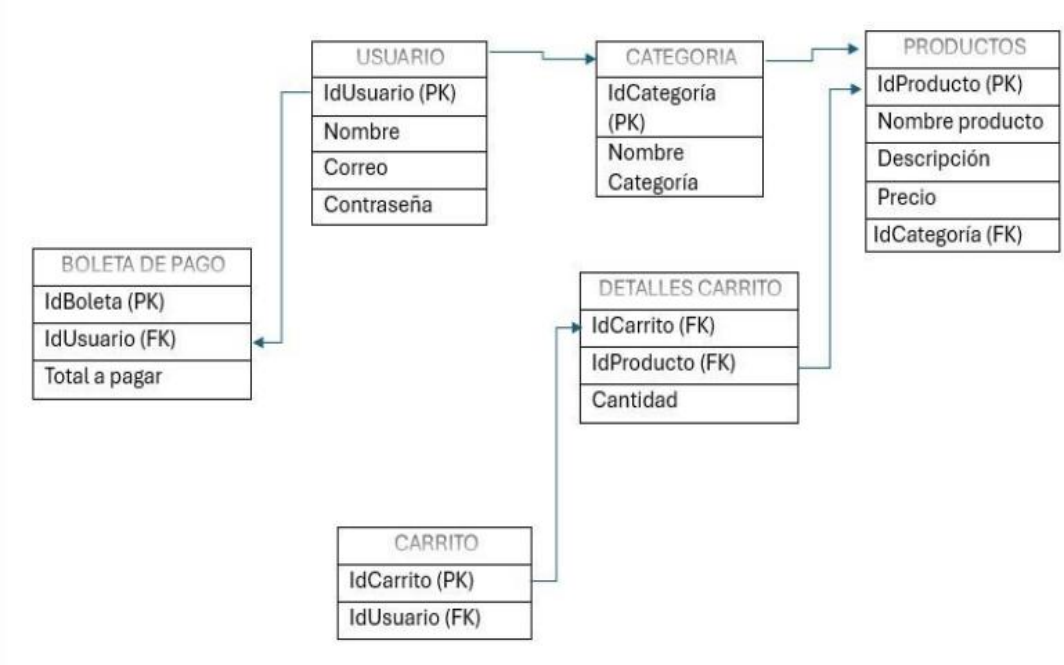
   npm install -g react-native-cli.

6. **Sexto Paso**

   Para poder ejeuctar y visualizar la app en entorno Android
   npx React-native run-Android

# Estructura de la Base de Datos

**Diagrama UML.**




El diagrama que se muestra es la estructura de la base de datos para nuestra app y está compuesta por las siguientes tablas:

**Usuario:**

Esta tabla almacena información sobre los usuarios de la tienda. Los campos de esta tabla incluyen:

Id Usuario (PK): Un identificador único para cada usuario.

Nombre: El nombre del usuario.

Correo: La dirección de correo electrónico del usuario.

Contraseña: La contraseña del usuario.

**Categoría:**

Esta tabla almacena información sobre las categorías de productos de la tienda. Los campos de esta tabla incluyen:

IdCategoria (PK): Un identificador único para cada categoría.

Nombre: El nombre de la categoría.

**Producto:**

Esta tabla almacena información sobre los productos de la tienda. Los campos de esta tabla incluyen:

IdProducto (PK): Un identificador único para cada producto.

Nombre_producto: El nombre del producto.

Descripción: Una descripción del producto.

Precio: El precio del producto.

IdCategoria (FK): Un identificador de la categoría a la que pertenece el producto.

**Boleta_de_pago:**

Esta tabla almacena información sobre las boletas de pago de los pedidos realizados en la tienda. Los campos de esta tabla incluyen:

IdBoleta (PK): Un identificador único para cada boleta de pago.

IdUsuario (FK): Un identificador del usuario que realizó el pedido.

Total_a_pagar: El total a pagar por el pedido.

**Detalles_carrito:**

Esta tabla almacena información sobre los detalles de los carritos de compra de los usuarios. Los campos de esta tabla incluyen

IdCarrito (FK): Un identificador del carrito de compra al que pertenece el detalle.

IdProducto (FK): Un identificador del producto que se encuentra en el carrito.

Cantidad: La cantidad del producto que se encuentra en el carrito

.

**Carrito:**

Esta tabla almacena información sobre los carritos de compra de los usuarios. Los campos de esta tabla incluyen:

IdCarrito (PK): Un identificador único para cada carrito de compra.

IdUsuario (FK): Un identificador del usuario al que pertenece el carrito.

Relaciones entre tablas

Las tablas de la base de datos están relacionadas entre sí de la siguiente manera:

**Usuario:** La tabla Usuario está relacionada con la tabla Boleta_de_pago mediante una relación de uno a muchos. Esto significa que un usuario puede tener muchas boletas de pago, pero una boleta de pago solo puede pertenecer a un usuario.

**Usuario:** La tabla Usuario también está relacionada con la tabla Carrito mediante una relación de uno a muchos. Esto significa que un usuario puede tener muchos carritos de compra, pero un carrito de compra solo puede pertenecer a un usuario.

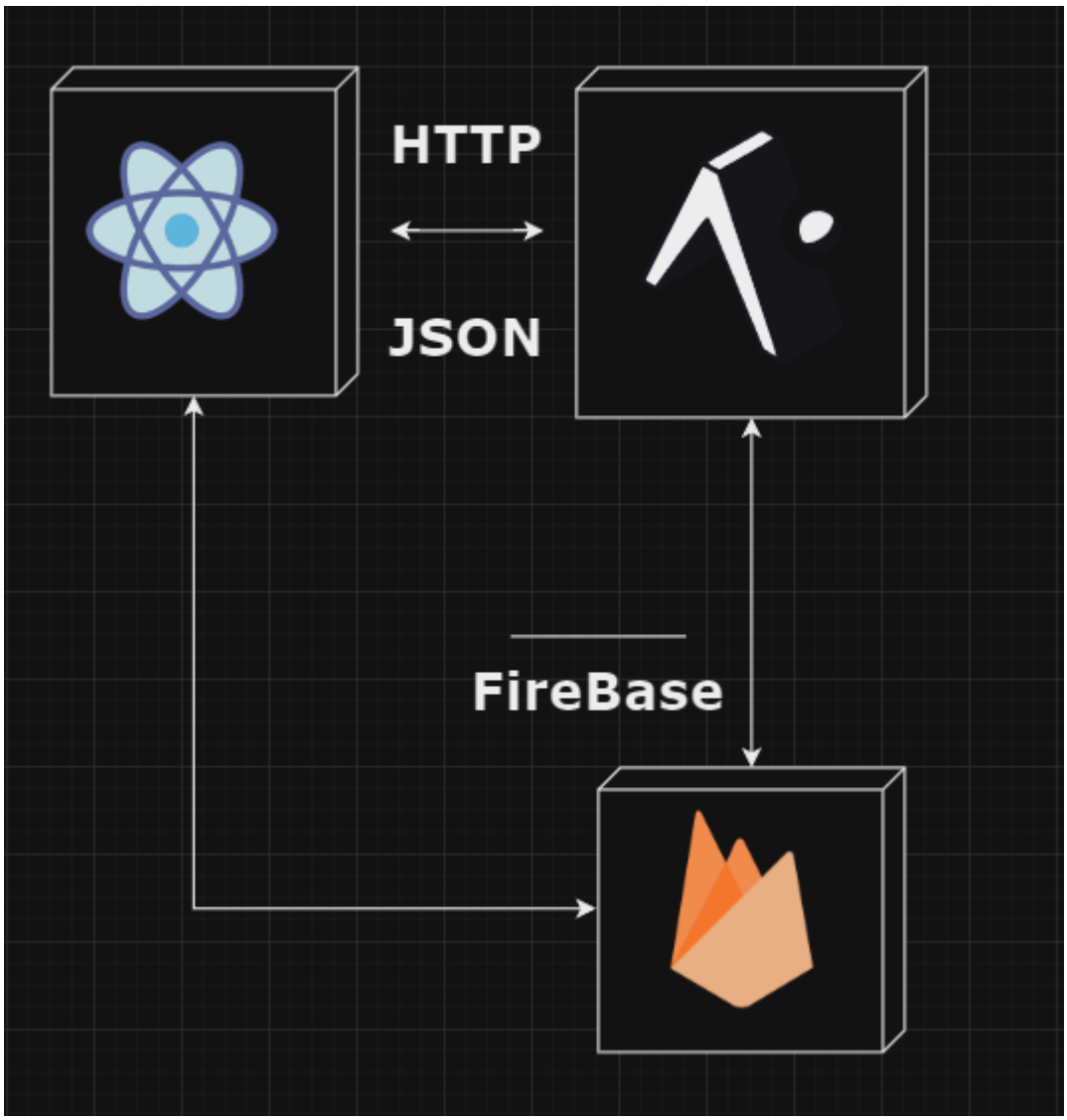
**Categoría:** La tabla Categoría está relacionada con la tabla Producto mediante una relación de uno a muchos. Esto significa que una categoría puede tener muchos productos, pero un producto solo puede pertenecer a una categoría

.

**Producto:** La tabla Producto está relacionada con la tabla Detalles_carrito mediante una relación de muchos a muchos. Esto significa que un producto puede estar en muchos carritos de compra, y un carrito de compra puede contener muchos productos.

**Carrito:** La tabla Carrito está relacionada con la tabla Detalles_carrito mediante una relación de uno a muchos. Esto significa que un carrito de compra puede tener muchos detalles de productos, pero un detalle de producto solo puede pertenecer a un carrito de compra.

# Diagrama de Arquitectura



# Contáctenos

**Heysel Guadalupe Argueta Hernández**
(heyselhrnndz@gmail.com)

**Andrés de Jesus Montano Gonzalez**
(andresmontano801@gmail.com)

**Jorge alexander Martinez Gonzalez**
(soyalexgg2@gmail.com)

**Erick Francisco Magaña Ramos**
(Erick.ramos.23308@gmail.com)

**Emerson Josue Gabriel Rosales**
(egabriel2251763@gmail.com)

**Allison Yasmin Vivas Castro**
(alissoncastrov93@**gmail.com**)

# Conclusión

El Manual Técnico de StoreSv abarca aspectos críticos para comprender, implementar y mantener el sistema. Ofrece una visión detallada de su arquitectura, tecnologías utilizadas, instalación y configuración, así como la estructura de la base de datos. Además, proporciona un recurso valioso para el equipo de desarrollo, asegurando una implementación exitosa y una gestión eficiente del sistema StoreSv. Este documento es una referencia esencial que cubre desde los aspectos más técnicos hasta la operativa diaria del sistema