UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA DE COMPUTACIÓN

SISTEMA DE GESTIÓN DE BODEGACIÓN MULTIFUNCIONAL

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Ciencias de la Computación

AUTOR: Ericsson Alejandro Aguirre Gómez

TUTOR: Julio Ricardo Proaño Orellana

Quito - Ecuador

2025

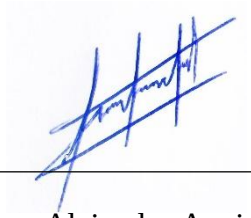
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Ericsson Alejandro Aguirre Gómez con documento de identificación N°1727029454 manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 25 de febrero del 2025

Atentamente,



Ericsson Alejandro Aguirre Gómez
1727029454

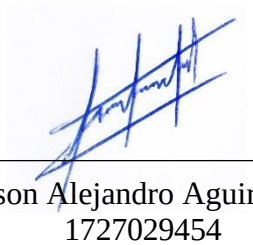
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Ericsson Alejandro Aguirre Gómez con documento de identificación N° 1727029454; expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy el autor del Proyecto Técnico: "Sistema de Gestión de Bodegación Multifuncional", el cual ha sido desarrollado para optar por el título de: Ingeniero en Ciencias de la Computación, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 25 de febrero del 2025

Atentamente,



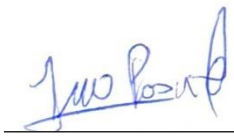
Ericsson Alejandro Aguirre Gómez
1727029454

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Julio Ricardo Proaño Orellana, con documento de identificación N° 0103909412, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: SISTEMA DE GESTIÓN DE BODEGACIÓN MULTIFUNCIONAL, realizado por Ericsson Alejandro Aguirre Gómez con documento de identificación N° 1727029454, obteniendo como resultado final el trabajo de titulación, bajo la opción de Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 25 de febrero del 2025

Atentamente,



Ing. Julio Ricardo Proaño Orellana, PhD.

0103909412

DEDICATORIA

Con todo mi corazón, dedico este proyecto a mis padres, quienes han sido mi mayor fuente de inspiración, apoyo y fortaleza. Su tiempo, esfuerzo y energía han sido fundamentales para que pueda alcanzar este gran logro. Gracias por estar siempre a mi lado, brindándome su amor, sabiduría y motivación en cada paso de este camino universitario. Su ejemplo de dedicación y perseverancia ha sido mi mayor enseñanza, y este triunfo también les pertenece a ustedes Gracias por no dejarme solo cuando yo más los necesite.

A mis profesores, les expreso mi más sincero agradecimiento por compartir su conocimiento y por el valioso tiempo que dedicaron a nuestra formación a lo largo de estos años en la universidad. Sus enseñanzas no solo marcaron mi vida académica, sino que también dejaron una huella imborrable en mi desarrollo profesional y personal. Gracias por su paciencia, compromiso y por impulsarnos a dar siempre lo mejor de nosotros mismos.

Este gran logro no habría sido posible sin ustedes, porque cada uno de ustedes ha contribuido significativamente a la realización de este sueño. Por eso, les dedico este gran proyecto con todo mi amor, gratitud y respeto.

Hoy, celebro junto a ustedes la culminación de este gran proyecto, que espero sea una contribución valiosa para futuras empresas y negocios. Más allá de ser el cierre de una etapa, este trabajo representa el inicio de un camino lleno de nuevas oportunidades y desafíos, inspirado en la innovación tecnológica y respaldado por el apoyo y las enseñanzas de quienes me han acompañado en este camino Universitario.

Ericsson Alejandro Aguirre Gómez

AGRADECIMENTO

Con mucha alegría y gratitud, quiero reconocer a mi padre por su amor infinito, su apoyo incondicional y la paciencia que siempre me ha brindado. A mi madre, gracias por tus sabios consejos y por los sacrificios que has hecho por mí y nuestra familia; cada acto que han realizado me ha inspirado a no rendirme ni poder detener cada uno de mis sueños por algo que si se puede lograr. Este logro es tan mío como de ustedes.

Agradezco profundamente a Dios por darme la fortaleza y la salud necesarias para alcanzar esta gran meta. Sin su guía, este camino habría sido mucho más difícil.

A mi familia cercana, gracias por acompañarme en esta etapa académica y por contribuir a mi crecimiento personal y profesional. Su compañía ha sido clave para llegar hasta aquí.

Finalmente, a todos aquellos que, con una palabra, un gesto o un momento compartido me ayudaron a seguir adelante, les agradezco de corazón. Cada granito de arena hizo la diferencia y me permitió completar este sueño tan anhelado.

Ericsson Alejandro Aguirre Gómez

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Este proyecto es patrocinado por la empresa Diekos SA, misma que se especializa en el desarrollo de aplicaciones de software, la empresa presenta actualmente un reto importante con el avance de la tecnología y al ofrecer soluciones de software que se adapten a las necesidades específicas de los clientes. Se ha visto en la necesidad de crear un sistema que facilite la gestión y administración de bodegas a lo largo de la cadena de suministros.

Hasta ahora, la falta de una herramienta de este tipo ha limitado la capacidad de satisfacer las expectativas de los clientes, afectando sus negocios. Con esta solución de software, buscamos simplificar el trabajo tanto para clientes, administradores y operadores, ofreciendo un sistema que permite un control completo a través de una aplicación web y una aplicación móvil. Nuestro principal objetivo es reemplazar los procesos manuales por digitales, mejorando la seguridad de la información y agilizando tiempos y procesos relacionados con pedidos. Esto ayudará a reducir las tareas repetitivas y a mejorar la experiencia de los usuarios.

Para llevar a cabo este proyecto, se realizó pruebas exhaustivas para garantizar la funcionalidad del software; como los accesos para usuarios, administradores y clientes, Además, se utilizó nuevas tecnologías para desarrollar este sistema:

- **SQL Server:** Utilizado como motor de base de datos para almacenar la información de manera segura.
- **NET Minimal API:** Entorno de desarrollo en Microsoft Visual Studio, que permite desarrollar de forma más simplificada endpoints para ser integrada en la aplicación web, no hace uso de controladores habituales.
- **Visual Studio Code:** Herramienta esencial para la edición del código de la aplicación web y móvil.
- **Node.js:** Entorno multiplataforma de código abierto utilizada para ejecutar JavaScript

en el desarrollo de las funcionalidades junto con React y React Native.

- **Tailwind CSS:** Framework que nos ayudó a diseñar una interfaz web moderna y funcional.
- **React Native:** Framework que nos permitió desarrollar aplicaciones multiplaforma móviles tanto para iOS como para Android y web.

Palabras clave: Transformación digital y Optimización de Procesos, control en tiempo real, swagger, react native, tailwind css.

ABSTRACT

This project is sponsored by Diekos SA, a company that specializes in the development of software applications; the company currently presents a major challenge with the advancement of technology and to offer software solutions that are tailored to the specific needs of customers. It has found the need to create a system that facilitates the management and administration of warehouses throughout the supply chain.

Until now, the lack of such a tool has limited the ability to meet customer expectations, affecting their business. With this software solution, we seek to simplify the work for customers, managers and operators, offering a system that allows complete control through a web application and a mobile application. Our main objective is to replace manual processes with digital ones, improving information security and streamlining order-related processes and times. This will help reduce repetitive tasks and improve the user experience.

To carry out this project, we have conducted extensive tests to ensure the functionality of the software, such as access for users, administrators and customers, in addition, we use new technologies to develop this system:

- **SQL Server:** Used as a database engine to store information securely.
- **NET Minimal API:** Development environment in Microsoft Visual Studio, which allows to develop in a more simplified way endpoints to be integrated into the web application, it does not make use of usual drivers.
- **Visual Studio Code:** Essential tool for editing the code of the web and mobile application.
- **Node.js:** Open source cross-platform environment used to run JavaScript in the development of functionalities along with React and React Native.
- **Tailwind CSS:** Framework that helped us design a modern and functional web

interface.

- **Ract Native:** Framework that allowed us to develop cross-platform mobile applications for iOS, Android and web.

Keywords: Digital Transformation and Process Optimization, real-time control, swagger, react native, tailwind css.

CAPÍTULO I

ANTECEDENTES Y GENERALIDADES

1.1. Introducción

En la actualidad, gestionar bien los inventarios es clave para el éxito de muchas empresas. La creciente demanda de productos y la necesidad de optimizar los procesos logísticos han llevado a las organizaciones a buscar soluciones creativas que les ayuden a ser más eficientes en la administración de sus existencias. En este contexto, un sistema informático de almacén se convierte en una herramienta invaluable. No solo automatiza tareas y reduce errores, sino que también maximiza la efectividad y mejora la situación económica de la empresa. Además, permite a las empresas adaptarse rápidamente a los cambios en la demanda, lo que se traduce en una reducción de costos y tiempos de espera. Sin embargo, es fundamental que los líderes comprendan bien tanto las capacidades como las limitaciones de esta tecnología para aprovecharla al máximo.

Este trabajo de tesis se enfoca en desarrollar un sistema informático de gestión de bodegas que responda a las necesidades específicas de las empresas distribuidoras. Su principal objetivo es facilitar el control y la gestión de inventarios, optimizando los procesos de recepción, almacenamiento y despacho de productos. A través de un enfoque metodológico que combina investigación y desarrollo ágil, se busca implementar una solución que no solo mejore la eficiencia operativa, sino que también ofrezca una interfaz amigable para los usuarios.

La relevancia de este sistema radica en su capacidad para brindar información en tiempo real, lo que permite a los administradores tomar decisiones informadas que impacten positivamente en la rentabilidad y en la calidad del servicio al cliente. Por lo tanto, esta tesis no solo enriquecerá el conocimiento académico sobre la gestión de bodegas, sino que también se convertirá en un recurso práctico para mejorar los procesos logísticos en el entorno empresarial.

1.2. Problema de estudio

Actualmente, en Diekos S.A., es proveedor de desarrollo de aplicaciones y enfrenta un desafío importante; dado que sus clientes solicita un software que permita la gestión y administración de bodegas a lo largo de todo el proceso operativo. El no disponer de un aplicativo ha afectado la capacidad de cubrir las necesidades requeridas de los clientes y sus negocios. Es por esta razón que se necesita crear un software que sea amigable, de fácil uso, destinado específicamente a la gestión y administración de bodegas, este aplicativo será ofertado a los emprendedores que no disponen de una herramienta para controlar tanto a sus clientes como a sus productos, dando a la empresa Diekos S.A. un realce importante que le permita ser más competitivo en el mercado. El objetivo de esta aplicación es pasar de un proceso manual a un proceso digital mejorando la seguridad de la información, aportando a los clientes a mejorar en tiempo y procesamiento de los pedidos con el fin de tener una reducción significativa de procesos manuales

1.2.1. *Antecedentes*

En las empresas, sobre todo en las que manejan grandes cantidades de productos, el no tener un buen sistema para organizar las bodegas se convierte en un verdadero problema. Sin un software especializado, todo se hace a mano o con herramientas que no son tan eficaces, como las hojas de cálculo. Esto puede traer varios inconvenientes errores con el inventario, demoras en la entrega y dificultades para encontrar lo que se necesita cuando es urgente.

Cuando las empresas no cuentan con un buen sistema de bodegación, surgen varios problemas que afectan su capacidad para lograr lo más importante: entregar los productos a tiempo y satisfacer a los clientes. El no tener la visibilidad en tiempo real del inventario, se pueden prometer productos que no están disponibles, los productos pueden dañarse o perderse por no ser manejados correctamente. Además, si la bodega no está bien organizada y los productos no están en su puesto, el proceso de seleccionar los productos para enviar puede

tomar más tiempo de lo que afecta los tiempos de entrega.

Al no contar con una herramienta adecuada para gestionar todo esto, la empresa corre el riesgo de perder clientes para las entregas incompletas, equivocadas o demoradas, lo que termina afectando su reputación y la capacidad de mantenerse competitiva en el mercado.

La empresa Diekos asume esta gran responsabilidad solventando la necesidad de diversos modelos de empresa existente en el mercado proporcionando un software de alta calidad seguro confíale y de fácil uso por el usuario.

1.2.2. *Importancia*

El no tener un buen software para administrar la bodega puede traer consecuencias serias. La falta de control en el inventario puede hacer que se pierdan productos, lo que no solo significa perder productos existentes, sino también más gastos para la empresa, porque hay que reponer lo que falta. Además, si no se cumplen los tiempos de entrega, los clientes no quedarán contentos, y eso afecta directamente la reputación del negocio. Los procesos manuales son más lentos y propensos a errores, lo que retrasa aún más todo. Un software adecuado transforma esos procesos, haciendo todo más rápido y preciso, lo que mejora la eficiencia, reduce los costos y asegura que el producto llegue a tiempo, todo de manera automatizada, sin tanta intervención humana. Esto no solo ahorra dinero, sino que asegura que el negocio siga funcionando sin contratiempos aumentado la economía y ganancia permitiendo expandir a la empresa que con lleva la generación de más plazas de trabajo.

1.2.3. *Delimitación*

Este software no está limitado a un solo tipo de empresa si no que también está diseñado para ser utilizado de manera global, beneficiando a cualquier negocio que necesite optimizar su negocio dentro del sistema de bodegación. En el ámbito de las empresas las compañías de cualquier tamaño y industria, tanto a escala local como internacional, necesitan herramientas que les faciliten gestionar su inventario de manera eficaz y automatizada. Con el software

desarrollado por parte de Diekos S.A, no importa cuán grande o pequeña sea la operación, dado se ajusta a las demandas particulares de cada compañía, contribuyendo a disminuir gastos, prevenir pérdidas de productos, acelerar los plazos de entrega y, finalmente, potenciar la competitividad en el mercado tecnológico.

1.3. Justificación

Diekos S.A. no cuenta con una aplicación para la administración y gestión de bodegas, lo que impide solucionar y resolver el problema fundamental, como lo menciono el técnico de la empresa Diekos S.A lo cual afecta directamente la competitividad de sus clientes. Al administrar sus inventarios de forma manual, muchas compañías y emprendedores experimentan faltantes en la gestión de sus inventarios, lo que puede resultar en errores, pérdidas de productos. La capacidad de expandirse en los negocios se ve limitada por la falta de un software de bodega. Disponer de herramientas tecnológicas sofisticadas es una necesidad en un mercado cada vez más competitivo. Los emprendedores podrán tener procesos automáticos reemplazando los procesos manuales, disminuir los errores y aumentar la gestión con el software. Esto les brindará una ventaja competitiva significativa, además de aumentar su productividad y ganancias. Para evitar las pérdidas de la información, producto y económicas, se ha considerado la elaboración de un software que permita a los clientes llevar su información de manera segura y organizada en el manejo del inventario dentro de la bodega.

1.4. Objetivos

1.4.1. *Objetivo General*

El desarrollo del Sistema de Gestión de Bodegación Multifuncional tiene como propósito optimizar la administración de inventarios y los procesos logísticos. Este sistema está diseñado para automatizar tareas, realizar el seguimiento en tiempo real de los productos e integrarse con otros sistemas empresariales. Con ello, se busca mejorar la eficiencia operativa, reducir costos y asegurar entregas puntuales y precisas, ofreciendo una solución integral a las

necesidades logísticas de la organización.

1.4.2. *Objetivos Específicos*

Entender las necesidades de cada usuario: Precisar qué funciones específicas deberá contener el software para que estos sean capaces de cumplir con las exigencias de los diferentes roles de usuario; el sistema resultante se adapta a cualquier tipo de negocio y a las particularidades que lo acompañen.

Definir un sistema escalable y adaptable: Diseñar una alternativa que consienta en expandir todas sus funciones con el tiempo; es la única forma de poder llevar un control estricto y exhaustivo acerca de los inventarios, sin importar la complejidad o el tamaño.

Crear un modelo funcional del sistema: Desarrollar un prototipo del software, el cual representa una primera herramienta para visualizar cómo se llevará a cabo la administración de bodegas.

Garantizar el correcto funcionamiento: Llevar a cabo una serie de pruebas para verificar que las funcionalidades del sistema trabajen como se espera, corrigiendo cualquier aspecto que deba ser solventado previo a la implementación.

1.5. Alcance

El alcance del proyecto se centra en el desarrollo de un software que centralice y proteja los datos en el manejo de bodegas. El software incluirá módulos para la gestión de usuarios, productos y pedidos, utilizando tecnologías avanzadas para optimizar los procesos operativos y mejorar la experiencia del usuario. El sistema que será web y móvil Este software solo contemplará

1.5.1. *Módulo de Usuario*

El módulo de Usuario permitirá la gestión de diferentes roles, incluyendo Administrador, Operador y Cliente. Este módulo proporcionará funcionalidades para agregar, editar y eliminar usuarios, así como asignar permisos específicos según el rol de cada usuario. Las funcionalidades específicas incluirán:

- Incluir tres roles: Administrador, Operador y Cliente
- Permitir al Administrador gestionar los demás módulos (Usuario, Producto y Pedido).
- Permitir al Cliente auto gestionar sus pedidos mediante la página web, posterior al registro en el sistema.

1.5.2. *Módulo de Producto*

El módulo de Producto permitirá la gestión e información sobre los productos de forma diaria. Incluirá funcionalidades para agregar, actualizar y eliminar productos, gestionar el inventario y realizar un seguimiento del estado y ubicación de los productos dentro de la bodega. Las funcionalidades específicas incluirán:

Almacenamiento de información en catálogos, colocando los atributos y localización para el despacho que será gestionado en el módulo de Pedido.

1.5.3. *Módulo de Pedido*

El módulo de Pedido permitirá la gestión de pedidos y el control de stock dentro del sistema. Este módulo facilitará la creación, actualización y seguimiento de pedidos, asegurando

que los niveles de inventario se mantengan actualizados y que se pueda realizar una gestión eficiente de las existencias. Las funcionalidades específicas incluirán:

- Gestión de pedidos y control de stock.
- Armar pedidos mediante el uso del aplicativo móvil.

1.5.4. *Seguridad y Protección de Datos*

El software garantizará la seguridad y protección de los datos mediante la implementación de protocolos avanzados de seguridad. Esto incluirá la encriptación de datos, autenticación de usuarios y auditorías de seguridad regulares para prevenir accesos no autorizados y asegurar la integridad de los datos

Mejoras en la Experiencia del Usuario

Se pondrá un énfasis especial en mejorar la experiencia del usuario mediante una interfaz intuitiva y fácil de usar, así como mediante la implementación de funcionalidades que faciliten la gestión de bodegas.

Para el usuario:

- Incluir tres roles: Administrador, Operador y Cliente.
- Permitir al Administrador gestionar los demás módulos (Usuario, Producto y Pedido).
- Permitir al Cliente gestionar pedidos desde el aplicativo móvil.
- Permitir al Cliente auto gestionar sus pedidos mediante la página web, posterior al registro en el sistema.

<u>En el sistema no se realizará:</u>

- El sistema web no será implementado en el ambiente de producción, solo en ambiente de test.
- Solo para administración y gestión de bodegas
- El sistema móvil solo será para dispositivos Android en este proyecto de titulación.

CAPÍTULO II

MARCO TEÓRICO

En la actualidad, Diekos S.A., un proveedor de desarrollo de aplicaciones está enfrentando un gran desafío: la falta de un software adecuado para controlar y administrar bodegas de manera efectiva. Esta carencia no solo limita su capacidad para ofrecer soluciones tecnológicas de alta calidad. Este marco teórico explora los conceptos y teorías clave detrás de la gestión de bodegas, la importancia de automatizar estos procesos, y cómo la falta de un software adecuado está afectando a las empresas de los clientes.

2.1. Gestión de Bodegas

La gestión de las bodegas es primordial para que las empresas funcionen adecuadamente. En lo relacionado con el almacenamiento y el movimiento de productos, desde la planificación del sitio físico categorizado en el almacén hasta el seguimiento de los inventarios. Un manejo eficaz garantiza que los productos estén disponibles cuando se requieran, que los costos se reduzcan y que la operativa en general se mejore (Bowersox, Closs, & Cooper, 2013).

Qué incluye: Controlar cómo ingresan y salen los productos es fundamental para administrar una bodega. El recibir la mercadería, almacenarlas adecuadamente y prepararlas para su envío (Lambert y Stock, 2013).

Por qué es importante: El manejar la administración de bodegas garantiza que los productos siempre estén disponibles para satisfacer la demanda de los consumidores, disminuye los costos de bodegaje y contribuye a que todo funcione de manera más fluida (Chopra & Meindl, 2016).

La automatización es necesaria para la administración de bodegas. Usar tecnología para hacer el trabajo más sencillo y efectivo implica automatizar la gestión de bodegas. La automatización puede acelerar los procesos, aumentar la precisión de los inventarios y

disminuir los errores (Kumar & Saini, 2014).

Beneficios de la Automatización: Al construir tareas automáticas se tiene la seguridad y garantía de hacer más preciso, seguro los diferentes pasos que están involucrados en la administración de bodega y menos propenso a cometer errores. La conexión con otros sistemas que posibilitan el seguimiento del inventario es parte de esto. (Heizer & Render, 2017)

Software para la administración de bodegas. Para hacer frente a las limitaciones y necesidades actuales, es necesario desarrollar un software enfocado en la administración de bodegas. Este software debería tener funciones como la administración de órdenes, la creación de informes detallados y el control de inventarios en tiempo real (Bowersox, Closs y Cooper, 2013).

Un software para las bodegas: debe proporcionar informes útiles sobre el estado de los productos, administrar pedidos y ver el inventario en tiempo real (Bowersox, Closs y Cooper, 2013).

Beneficios: La respuesta más efectiva a las necesidades del mercado, la precisión de los inventarios y la reducción de costos son posibles con un software adecuado (Mentzer, Min y Zacharia, 2001).

2.2. Relevancia para Emprendedores

Para los emprendedores, que a menudo no cuentan con un sistema de gestión adecuado, un software especializado puede ser un cambio radical. Les permitirá manejar mejor sus bodegas y mejorar su eficiencia operativa sin tener que gastar una fortuna en personalizaciones (Laudon & Laudon, 2014).

Accesibilidad y Flexibilidad: Un software adaptable a diferentes necesidades es ideal para emprendedores, ya que les permite ajustar el sistema a sus requerimientos específicos sin costos adicionales (Laudon & Laudon, 2014).

Mejora en la Gestión: Implementar un buen sistema de gestión puede hacer que los

procesos sean más eficientes, reducir errores y mejorar la manera en que los emprendedores manejan sus productos (Turban, Pollard, & Wood, 2018).

CAPÍTULO III

METODOLOGÍA

3.1. Enfoque de investigación

Este proyecto aplica un enfoque de investigación aplicada para desarrollar una solución tecnológica eficiente para la gestión de inventarios en un depósito. El uso de una metodología ágil, como Scrum permitirá la iteración y mejora continua del producto. Se incorporarán tecnologías como React, Minimal API, Node.js y Tailwind CSS.

3.2. Fases del desarrollo de software

3.2.1. *Análisis de requisit*

Entrevistas y observaciones: Se recopilan los requisitos del emisor del depósito para identificar áreas clave de la gestión de inventario, incluido el seguimiento de productos, la gestión de entradas y salidas y la generación automática de informes.

Un documento de requisitos preparado: Las historias de usuario y los casos de uso incluyen gestión de inventario en tiempo real, entradas y salidas de productos y alertas sobre los niveles de existencias que bajan la línea.

3.2.2. *Diseño del sistema*

Diagramas UML: Los casos de uso, las clases y los diagramas de secuencia se han diseñado sin necesidad de utilizar Microsoft Visual Studio.

Prototipos: Se han realizado prototipos de la interfaz de usuario a través de React y Tailwind CSS para obtener comentarios y experiencias de los usuarios antes de la utilización del módulo.

3.2.3. *Desarrollo e Implementación*

Lenguaje utilizado y marco: Los sistemas se desarrollaron con React para el frontend y Minimal API en Microsoft Visual Studio para el backend, lo que garantiza un código fácil y

limpio.

Metodología de Desarrollo: Se han utilizado sprints cortos durante un periodo escaso para conseguir sistemas de trabajo mínimamente viables y feedback en todo momento.

Base de datos utilizada: Un servidor SQL con claves primarias para productos, movimientos de inventario y tablas de usuarios.

3.2.4. *Pruebas*

Pruebas Unitarias: Se ha comprobado la prueba de los módulos con Reactores y Minimal API.

- Pruebas Integradas: Se ve como interactúa el módulo con este.
- Pruebas de Usuario: Se ha definido el éxito por módulos que han sido normalizados.

3.3. Implementación y capacitación

Se ha instalado una patente en Mediocre mediante software y se ha preparado una guía de actividades para el usuario con la instalación del software. Se han sintetizado y entregado a los usuarios actas orgánicas sobre la influencia de los sistemas.

3.4. Herramientas y tecnologías

Backend: API mínima en Microsoft Visual Studio

Frontend: React.js en Vite para desarrollos rápidos

Base de datos: SQL Server en la recopilación de datos

Estilo y diseño: Tailwind CSS para un diseño eficiente y moderno

CAPÍTULO IV

DESARROLLO SISTEMAS DE BODEGACIÓN MULTIFUNCIONAL

En este capítulo se describen todos los pasos ejecutados para la configuración del ambiente.

4.1. Diseño de la base de Datos

La base de datos que se utilizó es SQL Server, la que nos permitió almacenar un diseño de las tablas, para luego por medio de .Net Core realizar la codificación de las pantallas y las funcionalidades del programa. Para esto se realizó el análisis de los prerrequisitos de los ambientes y determinar las aplicaciones a instalar. Por medio del diseño realizado un diagrama que nos ayuda a entender de mejor manera como está estructurada la base de datos en nuestra aplicación web

Tablas principales. *A continuación, el detalle de algunas de las tablas de la base de datos:*

Tabla 1

Descripción de las tablas SQL principales

Tabla	*Descripción*
AspNetUsers	Contiene los campos de registro de los usuarios que interactúan en el sistema.
Pedidos	Permite crear el identificador único para cada pedido
Productos	Contiene el registro del producto
Categorías	Contiene las categorías del producto
Producto Pedidos	Tabla intermedia para relacionar pedidos y productos
Ubicaciones	Contiene la localización del producto dentro de la bodega

AspNetRoles	Contiene el registro de roles del sistema
AspNetUserRoles	Tabla intermedia que permite relacionar la tabla de usuarios con la de roles
AspNetRoleClaims	Define permisos o "claims" asociados a roles
AspNetUserClaims	Define permisos o "claims" específicos para usuarios.
AspNetUserLogins	Registra el inicio de sesión de los usuarios
AspNetUserTokens	Registra tokens de autenticación (para recordar sesiones o validar accesos)
Rutas	Registra las rutas o URLs disponibles en el sistema
RoleRutas	Relaciona la tabla de roles con rutas
EFMigrationsHistory	Tabla del sistema utilizada por Entity Framework para registrar los cambios o migraciones realizadas en la base de datos.

Nota. Se presenta las tablas utilizadas en la aplicación del Sistema de Bodegación Elaborado por: El autor

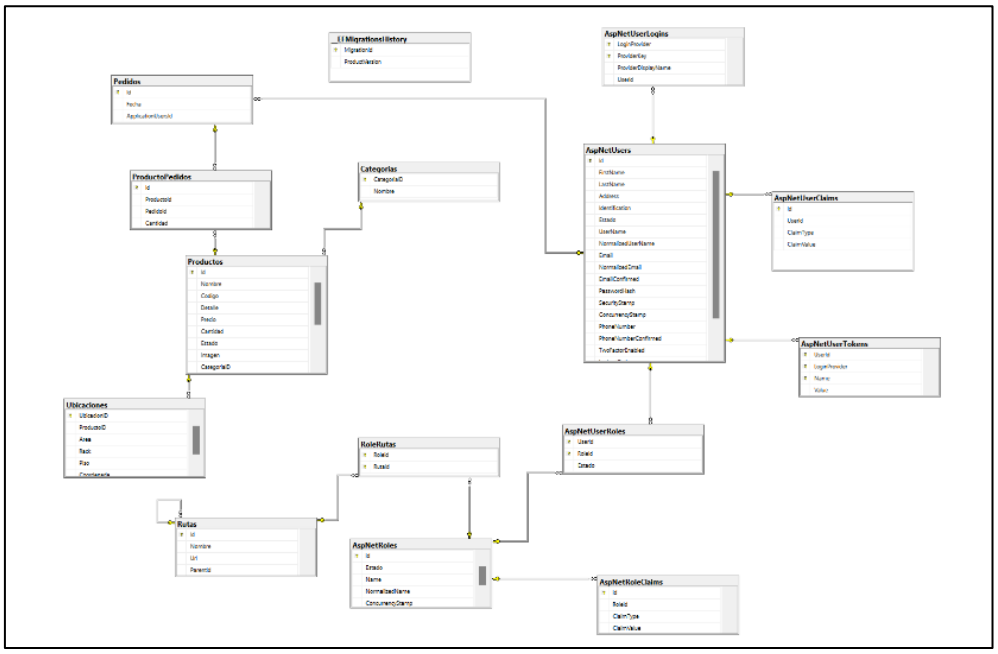
Diagrama de la Base de Datos

La base de datos es creada mediante la clase de .net Core realizada mediante la migración para ello debemos crear en el código las diferentes tablas y estructuras necesarias en el código luego de eso utilizamos comandos como **add migration** acompañado del nombre que queremos asignar en nuestra migración para aplicar los cambios y generar la base de datos ya debemos haber creado nuestra base de datos para mi caso es llamada **SA_Diekos** que será la

que utilizaren todo este proceso de migración así podemos llenar con los datos necesarios que necesitemos.

Figura 1

Diagrama de la Base de Datos



Nota. Se presenta Diagrama entidad-relación. Elaborado por: El autor

4.2. Servicio Login de Usuario (Back)

Para realizar el servicio login he ocupado el framework llamado Identity, el cual es utilizado en la configuración inicial, aquí podemos ver una línea de código **app.MapGroup("/identity").MapIdentityApi<ApplicationUser>()**; que automáticamente asigna las funcionalidades de identity API, así gestiona lo que es autorización y autenticación de los usuarios.

La **MapIdentityApi<ApplicationUser>(),** herada de identity para añadir campos necesarios como nombre, apellido etc.

Obteniendo como resultado una respuesta endpoints estandarizada para operaciones de usuario como:

- Registro
- Inicio de Sección (login)
- Confirmación de correo electrónico
- Restablecimiento de contraseña

Figura 2

Código de las Librerías Utilizadas en nuestro Login

A continuación, en la **Figura 2**, se muestra el código del servicio login de usuario.

```
app.MapGet("/sa_diekos/listuser", async (ApplicationDbContext context) =>
{
    try
    {
        var user = await (from u in context.Users
                          where u.EmailConfirmed==true
                          select new
                          {
                              u.Id,
                              u.UserName,
                              u.Email,
                              u.FirstName,
                              u.LastName,
                              u.Address,
                              u.Identification,
                              u.PhoneNumber,
                              u.Estado
                          }).ToListAsync();

        return Results.Ok(user);
    }
```

Nota. Se presenta las herramientas utilizadas para el login. Elaborado por: El autor

4.2.1. *Login de Usuario (Front)*

Para realizar nuestro login utilizamos la librería react.js y herramientas como YUP para validación de los campos del formulario, Tailwind es para generar estilos en nuestra aplicación. El consumo el servicio de Minimal API para la integración de la información en el servidor, obtenemos un formulario que permite a los usuarios ingresar su correo electrónico y contraseña para iniciar sesión

Los estados de este componente son 3 principales:

- formData: Se almacena los datos ingresados por el usuario sea correo o contraseña
- errors: Se muestra mensajes de error relacionados con la validación del formulario
- errorMessage: Nos muestra mensajes de error generales solo cuando ocurren problemas como credenciales incorrectas o errores del servidor.

Validación con Yup

Aquí definimos las reglas de validación que se mostraran cuando el usuario ingrese información errónea o incorrecta.

Gestión de Errores Temporales

Si hay un error de credenciales incorrectas nos va a mostrar un mensaje en errorMessage durante 5 segundos posterior este tiempo desaparecerá automáticamente, esto nos ayudar a tener la seguridad del sistema evitando que se pegue texto en el campo de contraseña y almacena tokens en sessionStore para disminuir riesgos de accesos que no están autorizados.

4.2.2. *Servicio Reseteo Contraseña (back)*

Para este servicio se define un endpoint de tipo post en la API así nos permite manejar el restablecimiento de contraseña de un usuario es decir que el cliente envía una solicitud post a la ruta /api/identity/resetpassword como un modelo de resetPasswordModel que contiene lo que es email,newpassword y confirmación de una nueva contraseña además de eso nos permite validar la salida mediante el servidor lo que es contraseña ingresada que coincidan que el usuario exista en la base de datos el token de restablecimiento de contraseña sea valido

Figura 3

Modelo de Datos (ResetPasswordModel)

```
app.MapPost("/api/identity/resetpassword", async (ResetPasswordModel model, UserManager<ApplicationUser> userManager)
    =>
{
    if (model.newPassword != model.confirmNewPassword)
    {
        return Results.BadRequest("Passwords do not match.");
    }

    // Buscar el usuario por su correo electrónico
    var user = await userManager.FindByEmailAsync(model.email);
    if (user == null)
    {
        return Results.BadRequest("User not found.");
    }
    var resetCode = await userManager.GeneratePasswordResetTokenAsync(user);
    // Decodificar el token antes de usarlo, si es necesario
    var decodedToken = Uri.UnescapeDataString(resetCode);
    // Restablecer la contraseña
    var result = await userManager.ResetPasswordAsync(user, decodedToken, model.newPassword);
    if (result.Succeeded)
```

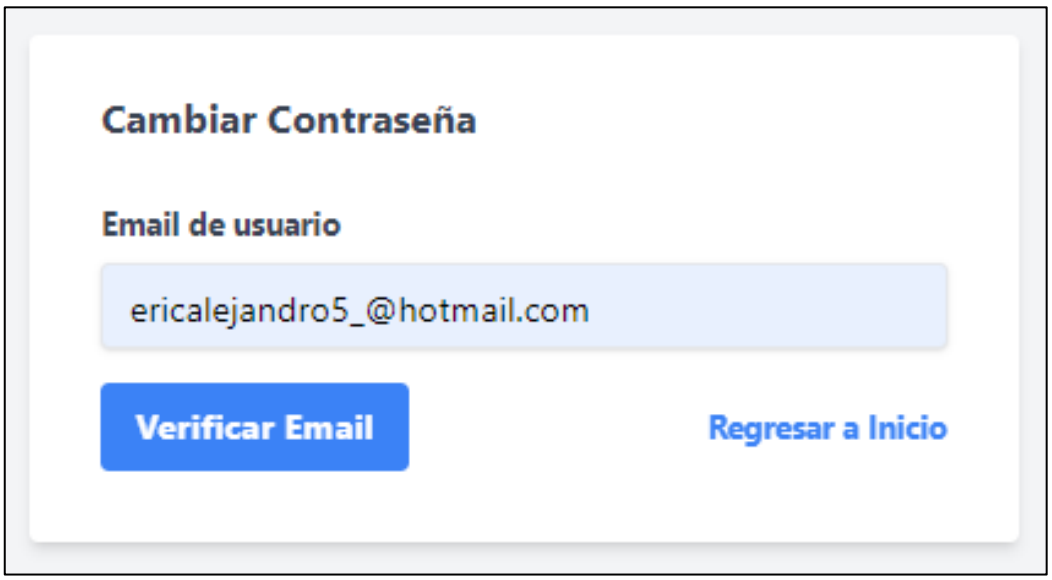
Nota. Se presenta el Código para restablecer password. Elaborado por: El autor

4.2.3. *Reseteo contraseña (From)*

Para este componente de react llamando reseteo de contraseña realizamos un consumo de los servicios es decir que los estados y validaciones están configuradas en la programación con YUP ya que valida que los campos sean correctos antes de enviarlos al servidor.

Figura 4

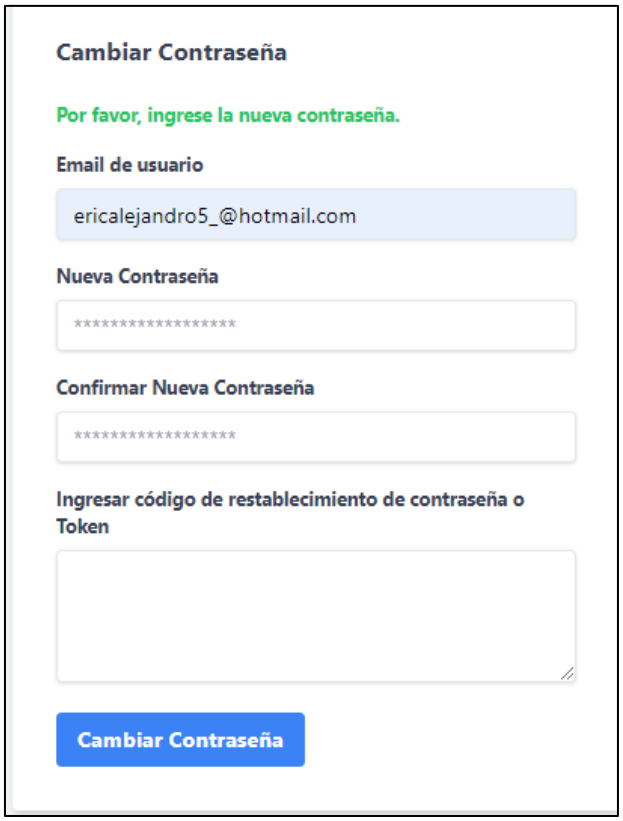
Pantalla de Reseteo Contraseña



Nota. Imagen de verificación de correo para restablecer la contraseña. Elaborado por: El autor

Figura 5

Procedimiento del Cambio de Contraseña



Nota. Formulario sobre la recuperación de contraseña. Elaborado por: El autor

4.2.4. *Servicio creación de Usuario (Back)*

En el servicio /sa_diekos/newuserclient se encuentra la creación o actualizaciones de usuario que se asigna roles específicos en el sistema que se utiliza .net core

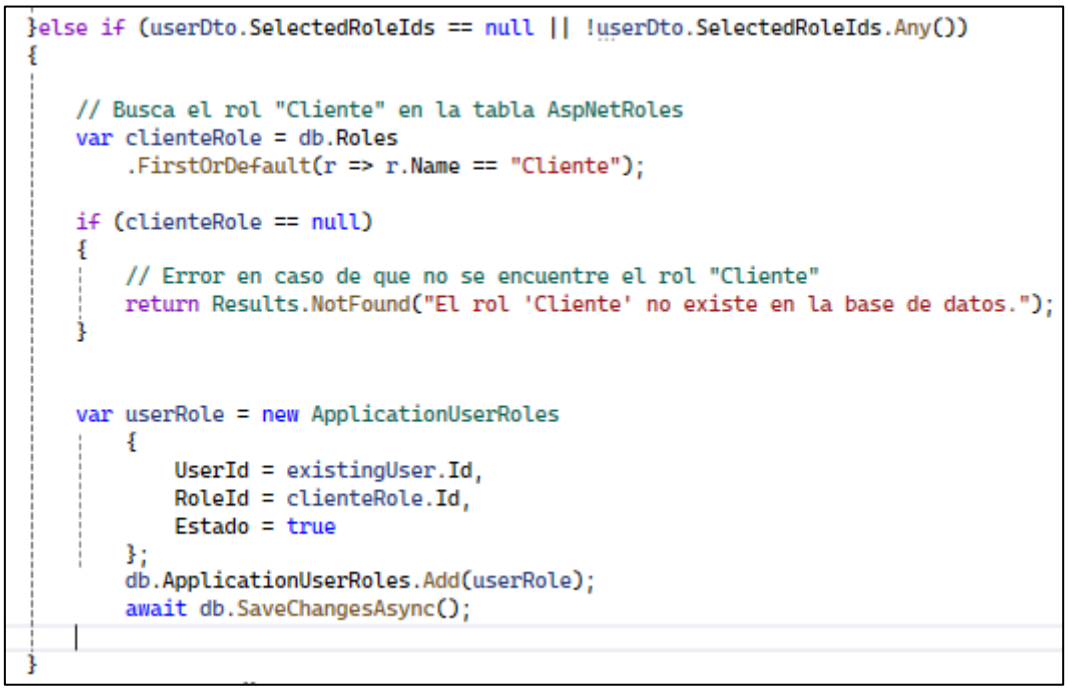
A continuación, en la **Figura 6** vamos a ver un poco del código que existe el rol de Administrador el cual verifica la lista SelectRoleIds es vacía o nula si no hay roles seleccionados busca el rol Cliente en la base de datos.

Si el rol Cliente no existe devuelve un error diciendo el rol no existe en la base de datos y si el rol Cliente existe lo asigna al usuario.

Figura 6

Validaciones de Código

Gestión de Usuarios y Asignación Automática de roles



```
}else if (userDto.SelectedRoleIds == null || !userDto.SelectedRoleIds.Any())
{

    // Busca el rol "Cliente" en la tabla AspNetRoles
    var clienteRole = db.Roles
        .FirstOrDefault(r => r.Name == "Cliente");

    if (clienteRole == null)
    {
        // Error en caso de que no se encuentre el rol "Cliente"
        return Results.NotFound("El rol 'Cliente' no existe en la base de datos.");
    }


    var userRole = new ApplicationUserRoles
        {
            UserId = existingUser.Id,
            RoleId = clienteRole.Id,
            Estado = true
        };
        db.ApplicationUserRoles.Add(userRole);
        await db.SaveChangesAsync();

}
```

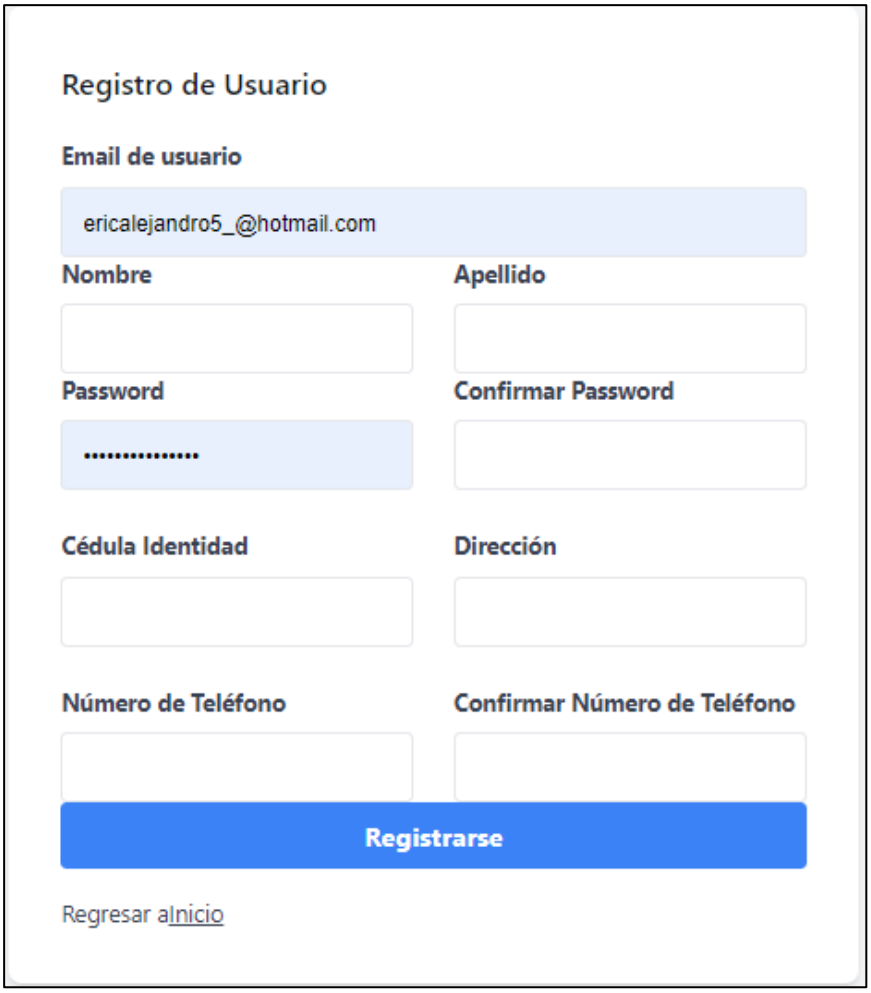
Nota. Verificación si el usuario es rol o Cliente. Elaborado por: El autor

4.2.5. *Creación de Usuario (Front)*

Para la creación del usuario es básicamente lo mismo ya que aquí lo más importante de React es gestionar el formulario de registro de usuario recalcando lo más importante que es la validación robusta del software para garantizar que los datos sean correctos un manejo de flujo completo registrando el usuario el asignar roles predeterminados automáticamente y que la interfaz sea amigable para que el usuario le permita saber que falta de corregir utilizando mensajes claros.

Figura 7

Gestión de Usuarios y Asignación Automática de roles



Nota. Verificación si el usuario es rol o Cliente. Elaborado por: El autor

4.2.6. *Servicio eliminación de Usuario (Back)*

Este servicio permite eliminar de manera visual a los usuarios, sin embargo, dentro de la programación solo corresponde a un cambio de estado de Activo a Inactivo.

Figura 8

Código Eliminar Usuarios

```
app.MapDelete("/sa_diekos/deleteuser", async ([FromBody] DeleteUserDTO deleteUserDto, ApplicationDbContext db) =>
{
    try
    {
        var idToDelete = deleteUserDto.Id;

        var userToDelete = await db.Users.FindAsync(idToDelete);
        if (userToDelete == null)
        {
            return Results.NotFound("Usuario no encontrado");
        }
        userToDelete.Estado = false;
        await db.SaveChangesAsync();

        return Results.Ok("Usuario eliminado correctamente");
    }
    catch (Exception ex)
    {
        return Results.Problem("Error al eliminar el usuario. " + ex.Message);
    }
});
```

Nota. Imagen sobre deshabilitar el usuario en la API. Elaborado por: El autor

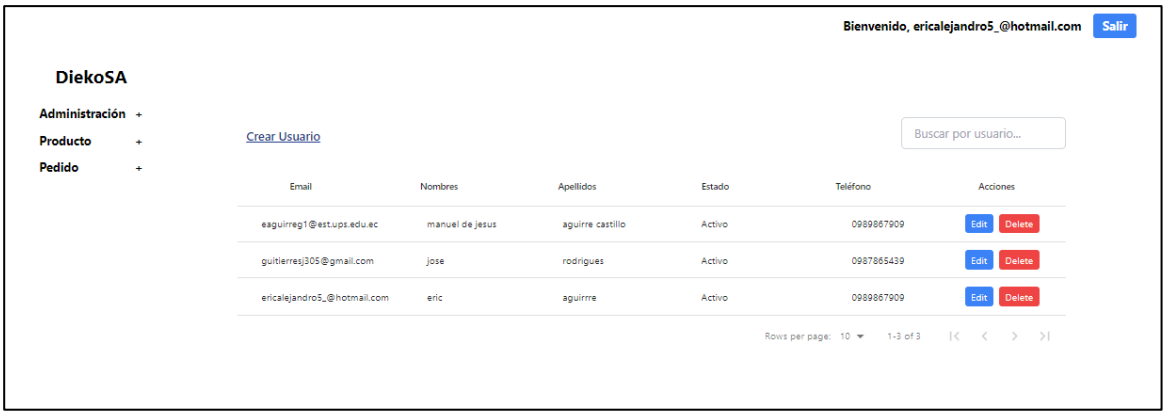
4.2.7. *Deshabilitar Usuario (Front)*

Este proceso debe realizarlo el Administrador del sistema, dado que será el único que podrá volverlo habilitar posterior se requiera una nueva activación de usuario.

Para realizar el procedimiento de eliminar (delete) el usuario primeramente la persona que debe realizar la des habilitación, la eliminación solo corresponde a un cambio de estado debido a que el sistema guarda todas las acciones realizadas.

Figura 9

Pantalla de Usuarios

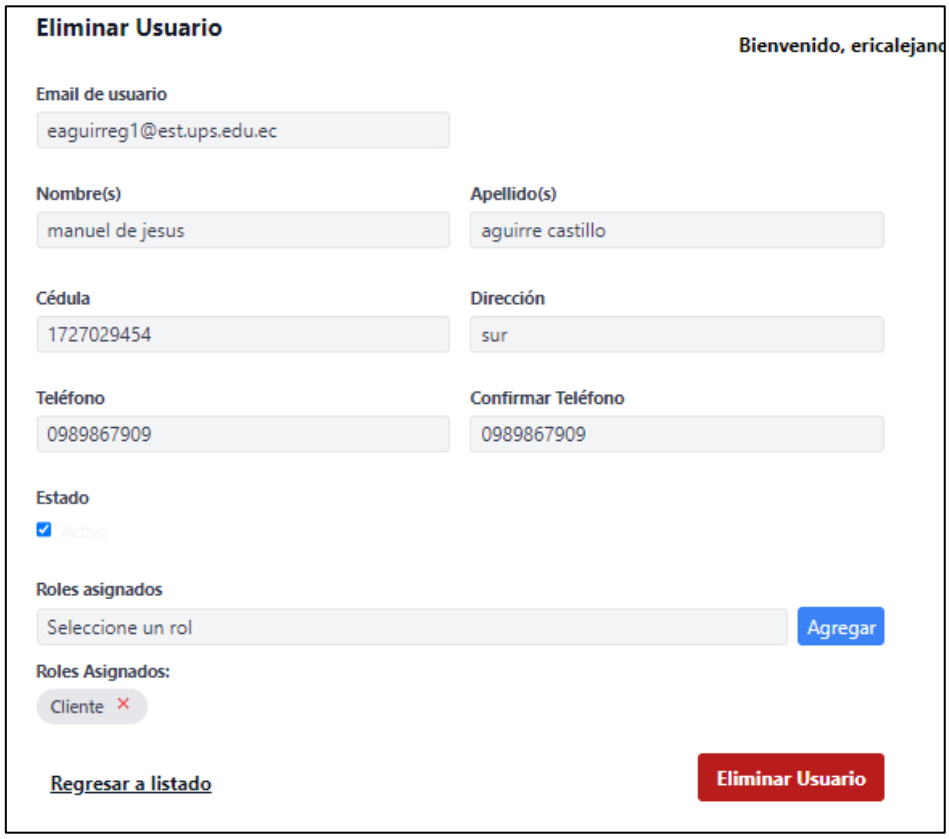


Nota. Nos permitirá desactivar el usuario en la API. Elaborado por: El autor

Figura 10

Deshabilitar el Usuario que Tiene como Rol Cliente

Al eliminar el usuario cliente del API no tendra acceso ni uso de la API.

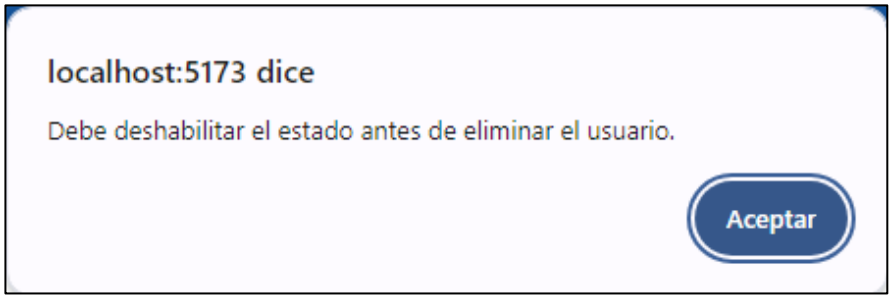


Nota: Procedemos a eliminar al usuario cliente del API para revocar su acceso y ingreso de la misma. Elaborado por: El autor

Figura 11

Mensaje de Advertencia

Para eliminar el usuario primero se debe deshabilitar al usuario, caso contario aparecerá un mensaje indicando de advertencia que la acción que se debe realizar.

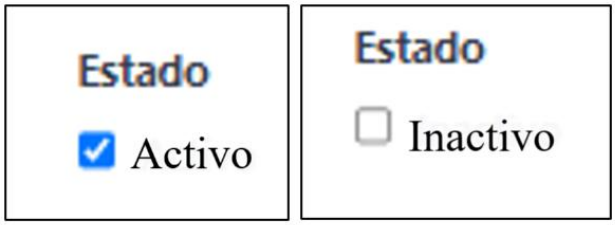


Nota. Como vemos el mensaje en pantalla sin deshabilitar el usuario no podemos eliminar.

Elaborado por: El autor

Figura 12

Gráfico del estado a del usuario (Activo/ Inactivo).

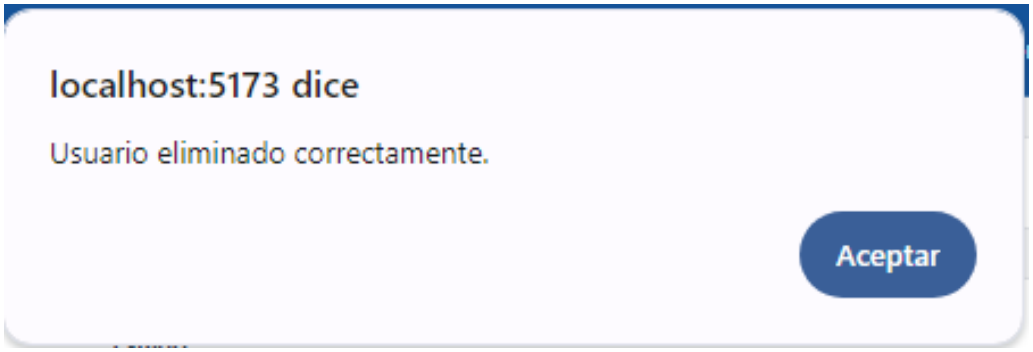


Nota. Deshabilitamos el usuario y procedemos a eliminar el usuario. Elaborado por: El autor

Figura 13

Notificación de usuario eliminado

Deshabilitamos el usuario y cómo podemos ver el usuario ha sido eliminado.



Nota. Imagen nos indica mensaje de usuario eliminado. Elaborado por: El autor

Figura 14

Visualización de que el usuario esta deshabilitado de la API

Resultado de eliminación del usuario

Nota. Usuario deshabilitado de la Api. Elaborado por: El autor

4.2.8. *Servicio Listado de Usuario (Back)*

Para este servicio se ocupa el componente que permite realizar el paginado el cual muestra de 10 en 10 los registros, en nuestra base de datos se busca todos los usuarios que se encuentren registrados y se muestra información de cada uno.

Figura 15

Código que permite enlistar Usuarios

.

```
app.MapGet("/sa_diekos/listuser", async (ApplicationDbContext context) =>
{
    try
    {
        var user = await (from u in context.Users
                          where u.EmailConfirmed==true
                          select new
                          {
                              u.Id,
                              u.UserName,
                              u.Email,
                              u.FirstName,
                              u.LastName,
                              u.Address,
                              u.Identification,
                              u.PhoneNumber,
                              u.Estado
                          }).ToListAsync();

        return Results.Ok(user);
    }
    catch (Exception ex)
    {
        return Results.Problem("An error occurred while retrieving the users. " + ex.Message);
    }
});
```

Nota. Código nos permitirá visualizar en pantalla la lista de usuarios. Elaborado por: El autor

4.2.9. *Listado de Usuario (Front)*

En nuestra API podemos observar los usuarios registrados y podemos editar, eliminar usuarios cuantas veces sea necesario realizar alguna actualización.

Figura 16

Imagen del listado Usuario

Nos permitirá visualizar en pantalla la lista de usuarios

Nota. Pantalla de usuarios usuarios utilizados y registrados a utilizar. Elaborado por: El autor

4.2.10. *Servicio creación de Roles (Back)*

Para este servicio se mapea en la ruta /sa_diekos/rol permitiendo obtener una lista de todos los roles almacenados en la base de datos, se usa TolistAsync() esto es para devolver todos los registros de la tabla roles en formato JSON, mediante este código se va a crear el rol y se va a poder observar que se verifica si el rol existe antes de crearlo para evitar duplicados.

Figura 17

Código del servicio de Roles

El código administra lo roles de manera eficiente y robusta.

```
//Crea rol
app.MapPost("/sa_diekos/crearrol", async (CrearRolDTO crearol, ApplicationDbContext db) =>
{
    // Normaliza el nombre del rol para comparación
    var normalizedName = crearol.rol?.ToUpper();

    // Verifica si el rol ya existe
    var existingRole = await db.Roles
        .FirstOrDefaultAsync(r => r.NormalizedName == normalizedName);

    if (existingRole != null)
    {
        // Si el rol ya existe, devuelve un mensaje de error
        return Results.BadRequest(new { Message = "El rol ya existe" });
    }

    // Crea un nuevo rol
    var newRole = new ApplicationRole
    {
        Id = Guid.NewGuid().ToString(),
        Name = crearol.rol,
        NormalizedName = normalizedName,
        ConcurrencyStamp = Guid.NewGuid().ToString(),
        Estado=crearol.estado,
        // Asigna otros campos necesarios
    };

    db.ApplicationRole.Add(newRole);
    await db.SaveChangesAsync();

    // Devuelve un mensaje de éxito
    return Results.Ok(new { Message = "Rol creado exitosamente" });
});
```

Nota. Código de administración de roles. Elaborado por: El autor

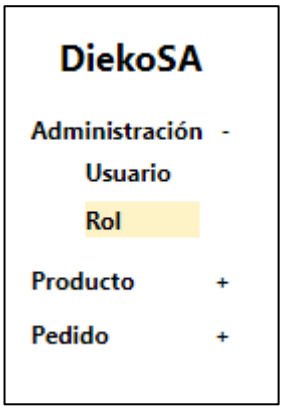
4.2.11. *Creación de Roles (Front)*

Menú de Rol como podemos ver nuestro menú rol se encuentra creado 3 roles que es Administrador, Cliente, Operador para lo cual la empresa Diekos S.A ocupara de manera correcta.

Figura 18

Ingreso al menú de Administración – Rol

El proceso para ingresa al rol es mediante el administrador.



Nota. Visualización de menú de administración. Elaborado por: El autor

Figura 19

Pantalla de roles de la aplicación.

Esta pantalla solo es visualizada por el administrador

Nota. Pantalla de Roles Utilizados. Elaborado por: El autor

Figura 20

Crear Rol

Para la creación de un rol en la aplicación se realiza de la siguiente manera:

Dar clic en link de crear usuario.

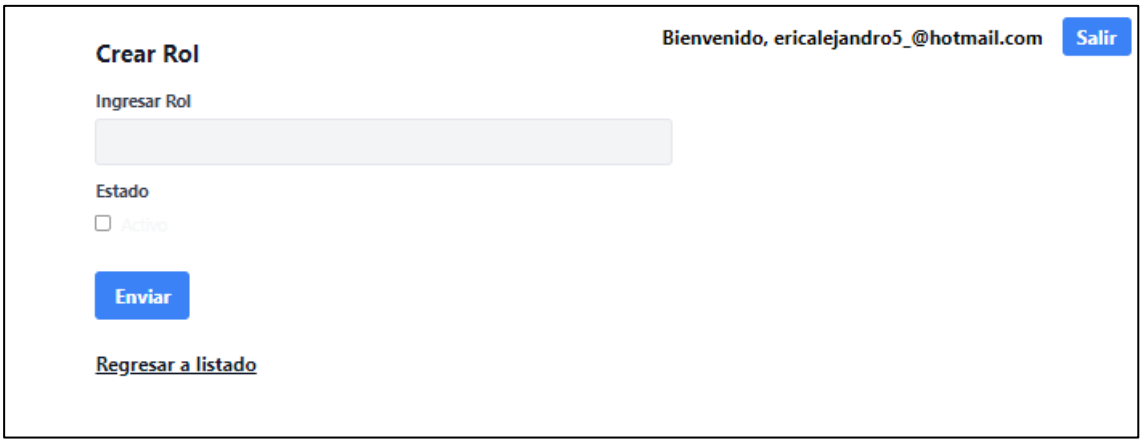
Nota. Link de creación de rol. Elaborado por: El autor

Imagen del proceso de crear un nuevo rol si fuera necesario.

Figura 21

Pantalla de Crear Rol

En esta pantalla se debe colocar el nombre del rol, activar el rol por medio del check y enviar para crear el registro en la base de datos.



Nota. Link de creación de rol para registrar en el listado. Elaborado por: El autor

4.2.12. *Servicio Crear Producto (Back)*

Para este servicio está diseñado para recibir datos de un formulario que incluye información detallada del producto su imagen y ubicación en la API del método de extraer datos del formulario. Este código nos permite crear producto con la información necesaria que desea ver el usuario recibiendo datos, proceso de imagen, validación de datos y guardando el producto

Figura 22

Imagen Extraer Datos del Formulario.

```
// Leer datos del formulario
var categoriaId = int.Parse(form["categoriaID"]);
var nombre = form["nombre"];
var codigo = form["codigo"];
var detalle = form["detalle"];
var precio = decimal.Parse(form["precio"]);
var cantidad = int.Parse(form["cantidad"]);
var estado = bool.Parse(form["estado"]);
var area = form["area"];
var rack = form["rack"];
var piso = form["piso"];
var coordenada = form["coordenada"];
```

Nota. Extrae los campos como nombre, código, detalle, precio, cantidad, categoría, y estado del formulario. Elaborado por: El autor

Figura 23

Código de conversión de Imagen.

```
byte[]? imagenBytes = null;
var imagenFile = form.Files["imagen"];

if (imagenFile != null)
{
    using (var memoryStream = new MemoryStream())
    {
        await imagenFile.CopyToAsync(memoryStream);
        imagenBytes = memoryStream.ToArray();
    }
}
```

Nota. Convierte la imagen en un arreglo de bytes (byte[]) para almacenarla en la base de datos.

Elaborado por: El autor

Figura 24

Código almacenamiento de Imagen.

Código de almacenamiento de la imagen en memoria string en memora stram

```
byte[]? imagenBytes = null;
var imagenFile = form.Files["imagen"];

if (imagenFile != null)
{
    using (var memoryStream = new MemoryStream())
    {
        await imagenFile.CopyToAsync(memoryStream);
        imagenBytes = memoryStream.ToArray();
    }
}
```

Nota. Almacenamiento de imágenes, Elaborado por: El autor

Figura 25

Código Crear el Modelo de Producto.

Usa los datos del DTO para crear un objeto del modelo Producto.

```
// Crear el DTO
var productoDto = new CrearProductoDTO
{
    Nombre = nombre,
    Codigo = codigo,
    Detalle = detalle,
    Precio = precio,
    Cantidad = cantidad,
    Estado = estado,
    Imagen = imagenBytes,
    CategoriaID = categoriaId,
    Area = area,
    Rack = rack,
    Piso = piso,
    Coordenada = coordenada,
    FechaCreacion = DateTime.Now
};
```

Nota. DTO formulario de almacenamiento de productos, Elaborado por: El autor

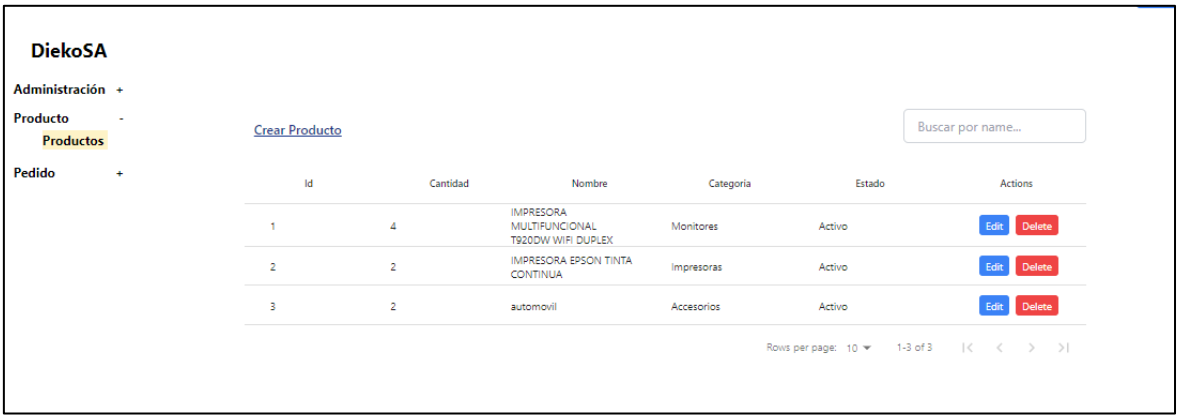
4.2.13. *Crear Producto (Front)*

La API para realizar la creación del pedido primero ingresamos a nuestro submenú y vemos los productos creados como en la imagen muestra.

Figura 26

Imagen del panel de menú de la API.

El administrador es el responsable de crear un producto, con los datos que solicitan en el formulario.

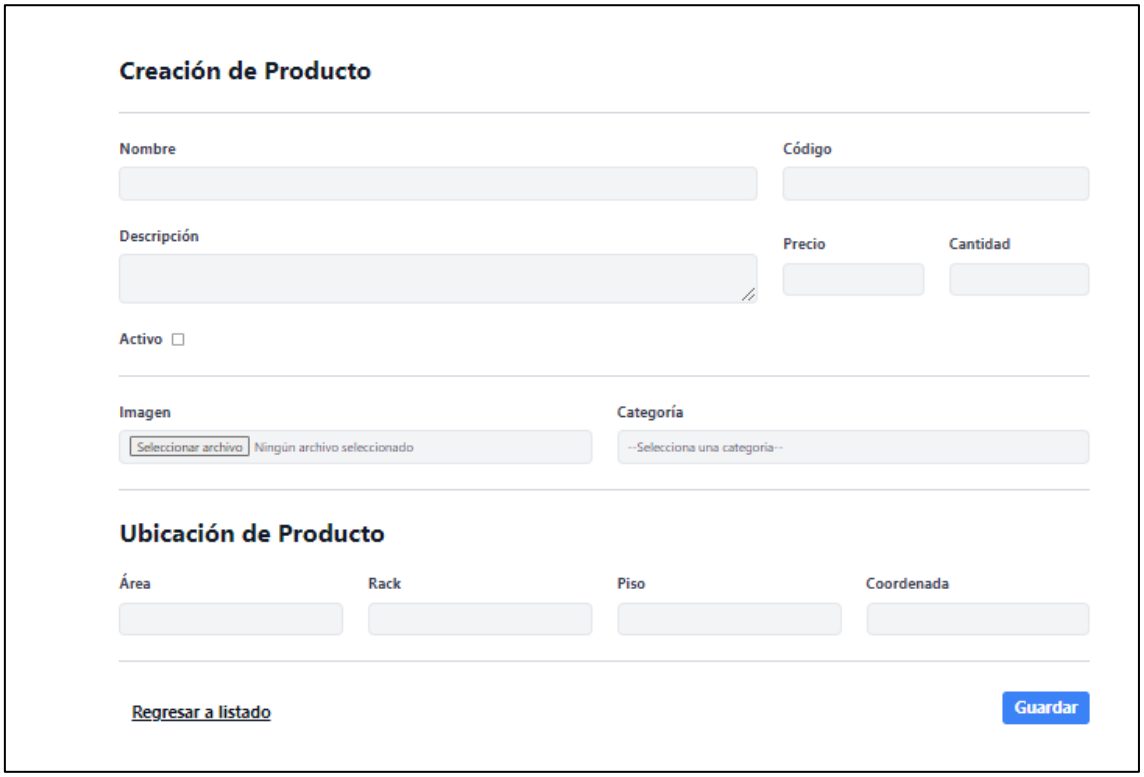


Nota. DTO formulario de almacenamiento de productos, Elaborado por: El autor

Figura 27

Crear un nuevo producto.

Entorno de crear nuevo producto Administrador.

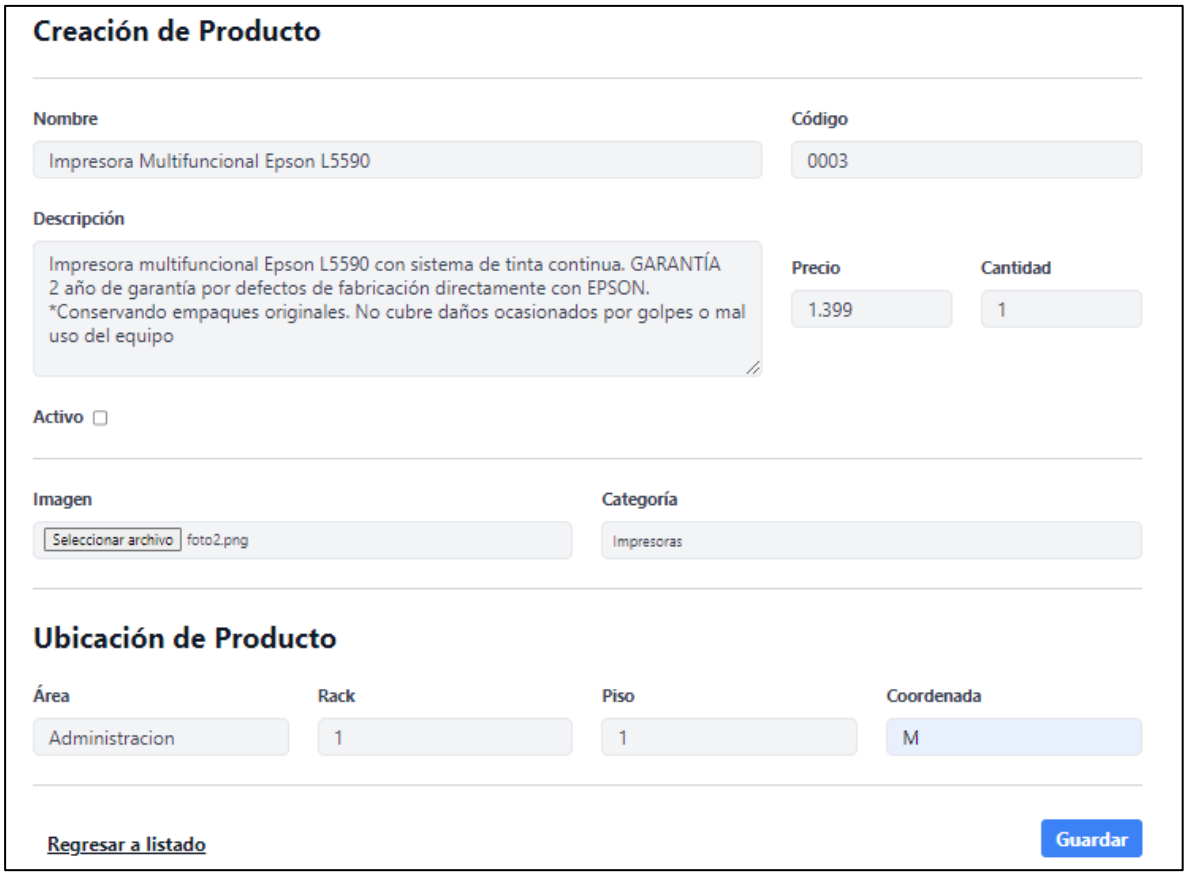


Nota. Formulario de creación de producto, Elaborado por: El autor

Para crear un producto damos clic en crear uno nuevo y solo el administrador tiene los accesos para poder crear un nuevo producto.

Figura 28

Creación de un nuevo producto.

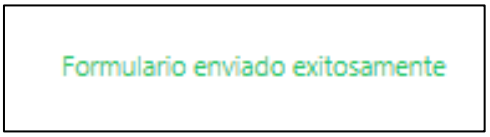


Nota. Detalles de los campos de creación de producto, Elaborado por: El autor

Figura 29

Mensaje de formulario completado

Posterior la creación del producto, aparecerá el un mensaje de formulario envido exitosamente.



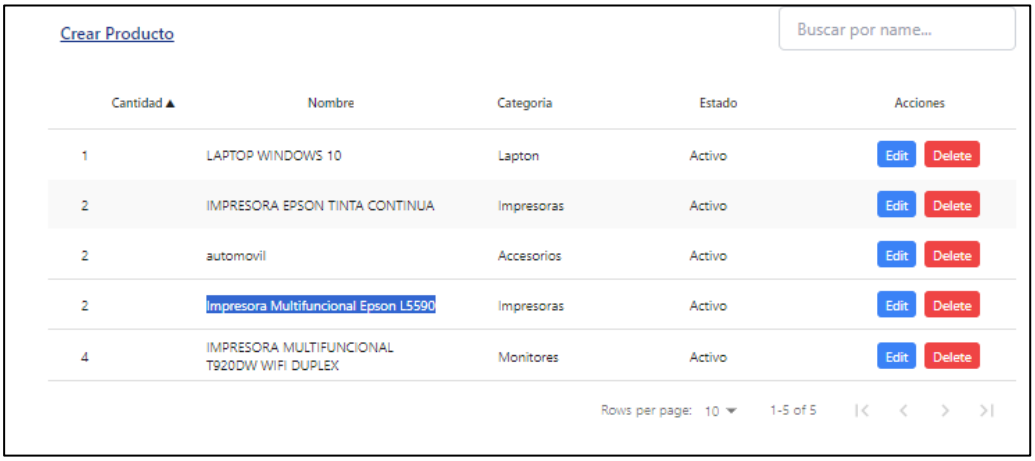
Nota. Notificación de creación del producto, Elaborado por: El autor

En nuestra pestaña de productos podemos ver que el producto ya está creado y nos muestra de manera correcta tendiendo también las opciones de editar y eliminar

si fuera necesario.

Figura 30

Producto Creado.



Nota. Listado de productos creados, Elaborado por: El autor

4.2.14. *Servicio Eliminar Producto (Back)*

En la siguiente figura vamos a poder ver un poco del código que nos permite eliminar un producto o cambiar su estado a inactivo, en la base de datos utilizamos nuestro id como identificador así podemos buscar un producto si el producto no fuera encontrado nos devuelve como mensaje Producto no encontrado.

Figura 31

Código Eliminar Producto.

Eliminación lógica de productos creados por el administrador

```
app.MapDelete("/sa_diekos/eliminarproducto/{id}", async (int id, ApplicationDbContext db) =>
{
    var producto = await db.Productos.FindAsync(id);
    if (producto == null)
    {
        return Results.NotFound(new { message = "Producto no encontrado" });
    }

   // db.Productos.Remove(producto);
   producto.Estado=false;
    await db.SaveChangesAsync();

    return Results.Ok(new { message = "Producto eliminado exitosamente" });
});
```

Nota. Código de eliminación de productos, Elaborado por: El autor

4.2.15. *Eliminar Producto (Front)*

En esta pantalla vamos a visualizar que el producto creado se deshabilitará de nuestra base de datos y de nuestro API, en esta parte la extracción de información muestra la disponibilidad de productos y la disponibilidad.

Figura 32

Eliminar producto

El producto fue eliminado como vemos en la imagen ya no será visible para el usuario

Nota. Pantalla de eliminación de productos, no se visualiza botón delete, Elaborado por: El autor

4.2.16. *Servicio enlistar Producto (Back)*

En nuestro endopint GET nos lista los productos activos con su respectiva información y los detalles de cada producto con su ubicación permitiéndonos estructurar correcta que se envía al cliente, mostrando los campos necesarios.

Figura 33

Código de productos.

Destaca que el código forma parte de nuestra API que fue creada para gestionar y consultar los productos activos

```
app.MapGet("/sa_diekos/prodcutoactivos", async (ApplicationDbContext db) =>
{
    var productos = await db.Productos
        .Include(p => p.Categoria) // Asegúrate de que la relación esté configurada
        .Where(p => p.Estado == true)
        .Select(p => new ProductoDTO
        {
            Id = p.Id,
            Nombre = p.Nombre,
            Codigo = p.Codigo,
            Detalle = p.Detalle,
            Precio = p.Precio,
            Cantidad = p.Cantidad,
            Estado = p.Estado,
            Imagen = p.Imagen,
            CategoriaID = p.CategoriaID,
            NombreCategoria = p.Categoria.Nombre, // Aquí accedes directamente al nombre de la categoría
            Area = p.Area,
            Rack = p.Rack,
            Piso = p.Piso,
            Coordenada = p.Coordenada
        })
        .ToListAsync();

    return Results.Ok(productos);
});
```

Nota. Código de validación de productos activos, Elaborado por: El autor

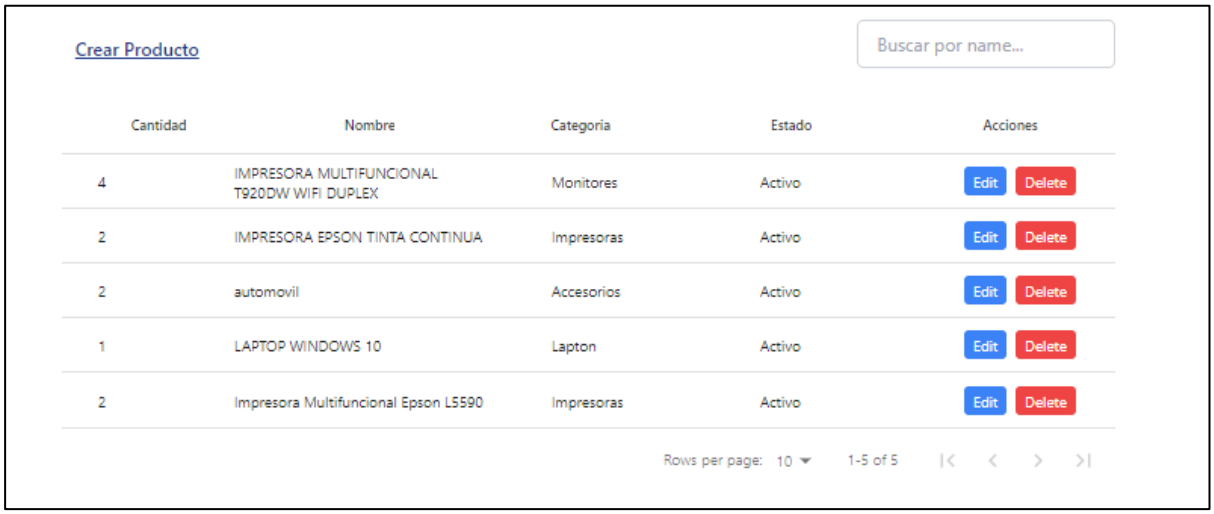
4.2.17. *Enlistar Producto (Front)*

Observamos todos los productos creados.

Figura 34

Listado de productos.

Productos activos que se encuentran dentro de nuestra API



Nota. Productos registrados en la Api, Elaborado por: El autor

4.2.18. *Servicio de Crear el Pedido (Back)*

En esta sección se recibe los datos del usuario creado y genera un pedido con la fecha actual también reconoce la lista de productos enviados en el pedido por cada uno del producto creando una relación con el pedido indicando la cantidad guarda el pedido y luego los productos asociados.

Figura 35

Código de creación de Pedido.

```
// realizar pedido
app.MapPost("/api/pedidosusuario", async (PedidoDTO pedidoDto, ApplicationDbContext db) =>
{
    // Crear nuevo pedido
    var nuevoPedido = new Pedidos
    {
     ApplicationUsersId = pedidoDto.UsuarioId,
       Fecha = DateTime.Now
    };

    db.Pedidos.Add(nuevoPedido);
    await db.SaveChangesAsync();

    // Agregar productos al pedido
    foreach (var productoPedido in pedidoDto.Productos)
    {
        var nuevoProductoPedido = new ProductoPedido
        {
            PedidoId = nuevoPedido.Id,
            ProductoId = productoPedido.ProductoId,
            Cantidad = productoPedido.Cantidad
        };
        db.ProductoPedidos.Add(nuevoProductoPedido);
    }

    await db.SaveChangesAsync();

    return Results.Ok(new { message = "Pedido guardado exitosamente" });
});
```

Nota. Productos activos que se encuentran dentro de nuestra API, Elaborado por: El autor

4.2.19. *Crear pedido (Front)*

El cliente puede ingresar al menú para poder ingresar en nuestro submenú y ingresar a realizar pedidos.

Figura 36

Panel del menú de la API.

Ingreso al menú de realizar pedidos en modo Usuario

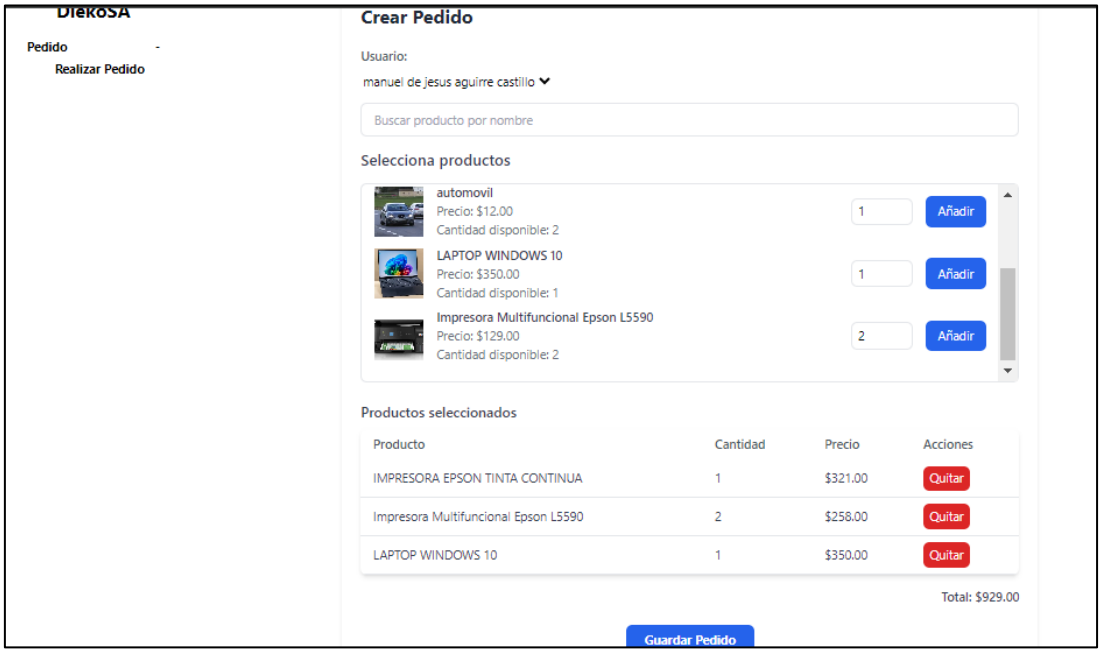
Nota. Pantalla con menú de pedidos perfil usuario, Elaborado por: El autor

En la siguiente imagen escogemos 3 productos para realizar la creación del pedido como podemos observar en cada producto nos indica el precio y la cantidad disponible de los productos en stock, una vez que seleccionemos los productos tendremos la suma total de cada producto adicional a esto también tendremos que seleccionar el nombre del usuario para poder completar todos los pasos.

Figura 37

Creación de un Pedido.

Ingreso al menú de realizar pedidos en modo Usuario



Nota. Pantalla con menú de pedidos perfil usuario, Elaborado por: El autor

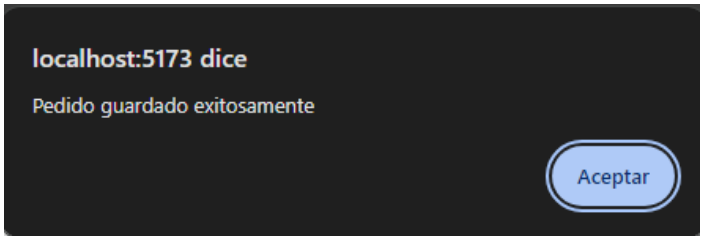
Ingresamos a nuestro menú como usuario cliente para poder ingresar en nuestro submenú y ingresar a realizar pedidos.

Al darle clic Guardamos el pedido nos indica la siguiente imagen.

Figura 38

Guardar el pedido.

El pedido está guardado y vuelve a recargarse la página para poder pedir el siguiente producto, verificación de Pedido guardado en nuestra base de datos sql

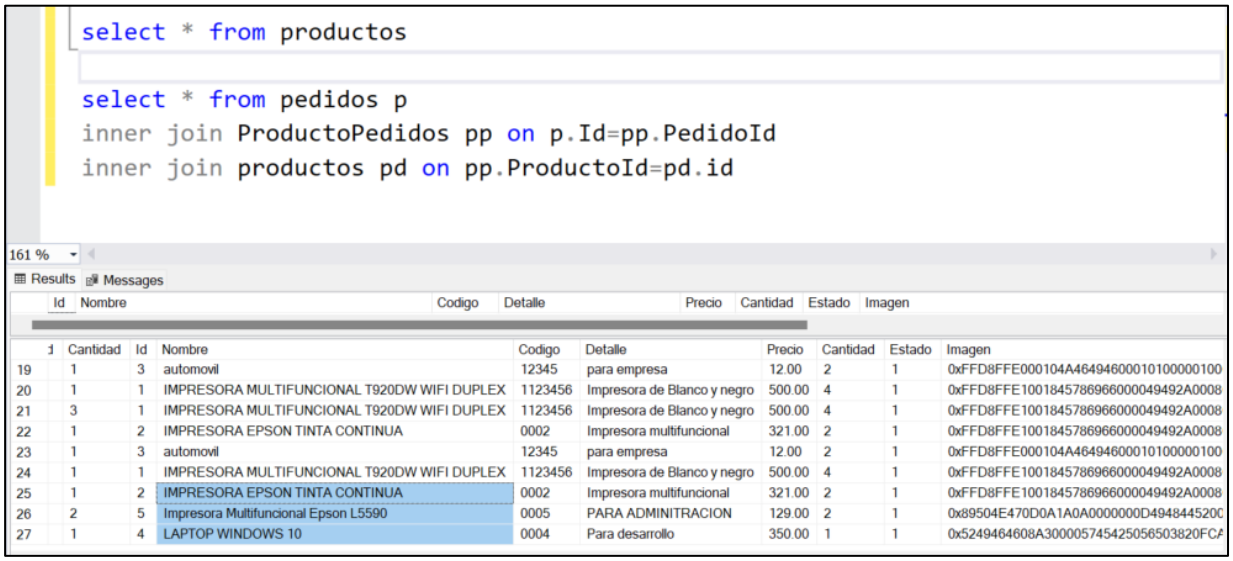


Nota. Notificación mediante mensaje de acción, Elaborado por: El autor

Figura 39

Consulta de Base de Datos.

Realizando las consultas en nuestra base de datos verificamos si esta correctamente registrado el pedido Usuario



```
select * from productos

select * from pedidos p
inner join ProductoPedidos pp on p.Id=pp.PedidoId
inner join productos pd on pp.ProductoId=pd.id
```

161 %

Results | Messages

Id	Nombre	Codigo	Detalle	Precio	Cantidad	Estado	Imagen

	Cantidad	Id	Nombre	Codigo	Detalle	Precio	Cantidad	Estado	Imagen
19	1	3	automovil	12345	para empresa	12.00	2	1	0xFFD8FFE000104A4649460001010000010
20	1	1	IMPRESORA MULTIFUNCIONAL T920DW WIFI DUPLEX	1123456	Impresora de Blanco y negro	500.00	4	1	0xFFD8FFE1001845786966000049492A0008
21	3	1	IMPRESORA MULTIFUNCIONAL T920DW WIFI DUPLEX	1123456	Impresora de Blanco y negro	500.00	4	1	0xFFD8FFE1001845786966000049492A0008
22	1	2	IMPRESORA EPSON TINTA CONTINUA	0002	Impresora multifuncional	321.00	2	1	0xFFD8FFE1001845786966000049492A0008
23	1	3	automovil	12345	para empresa	12.00	2	1	0xFFD8FFE000104A4649460001010000010
24	1	1	IMPRESORA MULTIFUNCIONAL T920DW WIFI DUPLEX	1123456	Impresora de Blanco y negro	500.00	4	1	0xFFD8FFE1001845786966000049492A0008
25	1	2	IMPRESORA EPSON TINTA CONTINUA	0002	Impresora multifuncional	321.00	2	1	0xFFD8FFE1001845786966000049492A0008
26	2	5	Impresora Multifuncional Epson L5590	0005	PARA ADMINITRACION	129.00	2	1	0x89504E470D0A1A0A0000000D494844520
27	1	4	LAPTOP WINDOWS 10	0004	Para desarrollo	350.00	1	1	0x5249464608A3000057454250565038 20FC

Nota. Notificación de acciones en la Base de datos, Elaborado por: El autor

4.2.20. *Editar el pedido (Front)*

Observamos que permite al usuario cambiar la cantidad de un producto que este seleccionado, validando que la cantidad no supere el máximo que se encuentra disponible dentro de la API.

Figura 40

Código de Editar Pedido.

Se realiza el cambio de la cantidad de un producto en el pedido utilizando handleCantidadChange.

```
const handleCantidadChange = (productoId, nuevaCantidad, cantidadDisponible) => {
    if (parseInt(nuevaCantidad) > cantidadDisponible) {
        alert(`La cantidad máxima disponible para este producto es ${cantidadDisponible}`);
        nuevaCantidad = cantidadDisponible; // Limitar la cantidad a la disponible
    }

    setCantidadProducto({
        ...cantidadProducto,
        [productoId]: parseInt(nuevaCantidad)
    });
};
```

Nota. Código de edición de pedido, Elaborado por: El autor

4.2.21. *Eliminar el Pedido (Front)*

Se encuentra la eliminación de productos seleccionados utilizando una función que nos permita un mejor manejo en el código utilizando handleRemoveProducto que permite al usuario quitar un producto que no este acorde a su compra también elimina la imagen del producto si esta seleccionada

Figura 41

Código Eliminar Productos.

El código puede ajustar la cantidad de productos seleccionados o eliminar

```
94      const handleRemoveProducto = (productoId) => {
95          setSelectedProductos(selectedProductos.filter(prod => prod.productoId !== productoId));
96          if (selectedProductos.find(prod => prod.productoId === productoId)?.imagenUrl === imageUrl) {
97              setImageUrl('');
98          }
99      };
```

Nota. Código de eliminar productos, Elaborado por: El autor

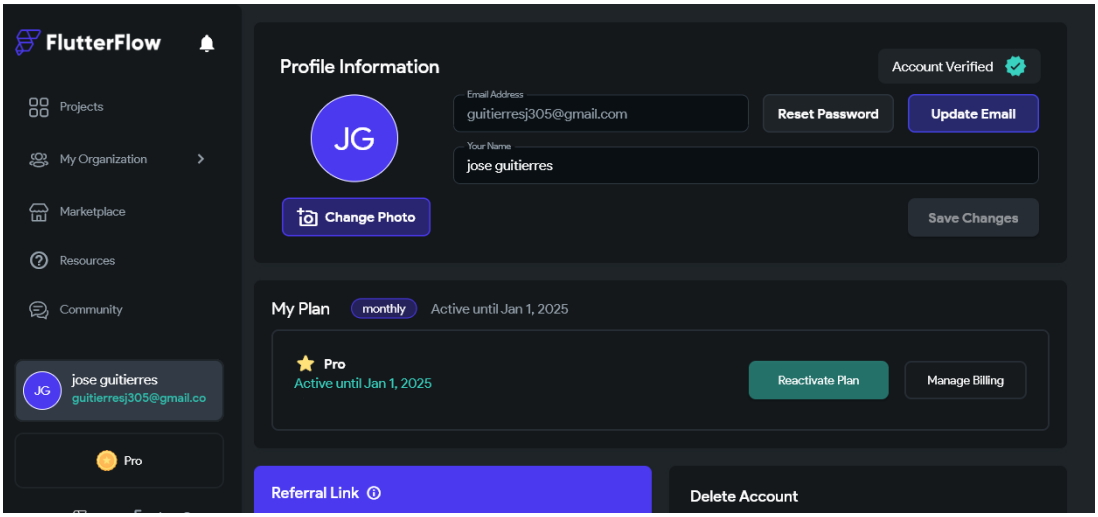
4.2.22. *Servicio login Aplicativo móvil (Back)*

Para empezar a realizar el Servicio móvil hemos podido obtener buenos resultados de nuestro proceso ocupando actualizaciones, instalaciones de paquetes que son necesarios para poder presentar nuestra aplicación móvil uno de ellos es poder descargar FlutterFlow iniciamos sección para ingresar al portal web después de eso realizamos la compra para poder utilizar todos los paquetes del programa Flutter Flow. Cómo debería estar configurado para poder utilizar localmente la aplicación de Flutterflow.

Figura 42

Pantalla suscripción en Flutterflow.

Es necesario suscribirse para poder utilizar las herramientas.



Nota. Validación de la cuenta Flutterflow, Elaborado por: El autor

En la conexión de nuestra API nos muestra el error 503 que nos indica que nuestra API local no está accesible desde FlutterFlow esto se debe a varios motivos relacionados con la configuración de la API. Es necesario generar una ip publica que apunte a nuestra API local para eso he investigado varias herramientas como ngrok localtunnel conversando con los técnicos y gerente de la empresa auspician.

Que es ngrok es una herramienta que nos permite a crear un túnel seguro entre nuestra API y una URL publica para descargar la aplicación primero tenemos que descomprimir el zip para poderlo instalar como una aplicación normal luego de eso procederemos a instalar en el celular una aplicación que se llama Autenticator abrimos la aplicación después de ser descargada y nos va a llegar un código para poder verificar nuestras credencias e identidad.

A continuación explicare que es reactive y en que ocupamos para realizar nuestro procedimiento de desarrollo móvil, como sabemos este desarrollo móvil es un framework, que nos permite desarrollar en JavaScript y React para construir aplicaciones en Android y IOS a diferencias de varios lenguajes investigados reac native no es necesario de utilizar visitas web

en lugar de eso se puede compilar en base a componentes nativos lo que proporciona un rendimiento superior y una experiencia única de conocer y aprender cómo se pudo desarrollar la aplicación.

Figura 43

Código de verificación Aplicación ngrok.

Aplicación completa con el código de verificación actualizado

Nota. Registro de numero para acceder a nuestra descarga de Ngrok, Elaborado por: El autor

*C*onfiguramos el código de verificación con jwt que significa que nos ayuda aceptar un token emitido por nuestro servicio especificado para nuestra prueba la URL que nos emitió el servicio es la siguiente que verán en nuestro BACK

Figura 44

Código de Verificación.

. Ubicación del token sea válido y emitido por esa URL requerida

```
// Configuración de autenticación y autorización
builder.Services.AddAuthentication("Bearer")
    .AddJwtBearer("Bearer", options =>
    {
        options.Authority = "https://43e8-157-100-143-177.ngrok-free.app"; // URL pública de ngrok
        options.TokenValidationParameters = new Microsoft.IdentityModel.Tokens.TokenValidationParameters
        {
            ValidateAudience = false,
            ValidateIssuer = true,
            ValidIssuer = "https://43e8-157-100-143-177.ngrok-free.app" // URL pública de ngrok
        };

        options.Events = new JwtBearerEvents
        {
            OnAuthenticationFailed = context =>
            {
                Console.WriteLine("OnAuthenticationFailed: " + context.Exception.Message);
                return Task.CompletedTask;
            },
            OnTokenValidated = context =>
            {
                Console.WriteLine("OnTokenValidated: " + context.SecurityToken);
                return Task.CompletedTask;
            }
        };
    });

builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("MyPolicy", policy =>
        policy.RequireClaim("scope", "myapi.read"));
});

// Configuración de base de datos y dependencias
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DeafaultConnection")));

builder.Services.AddIdentityApiEndpoints<ApplicationUser>()
    .AddEntityFrameworkStores<ApplicationDbContext>();

builder.Services.Configure<MailKitOptions>(builder.Configuration.GetSection("MailKitOptions"));
builder.Services.AddTransient<IEmailSender<ApplicationUser>, EmailSender>();
```

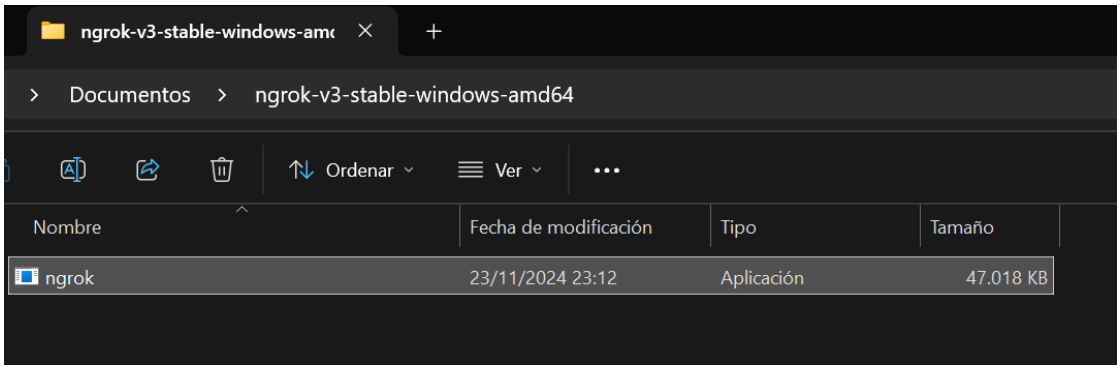
Nota. Validación de token de seguridad, Elaborado por: El autor

*E*jecutamos nuestra aplicación ngrok para que nos genere una ip publica clic derecho en ejecutar como administrador

Figura 45

Imagen para Ejecutar Ngrok

Una vez descargado y descomprimido nos saldrá de esta manera nuestra aplicación que realiza un puente en nuestra API para genera una IP pública



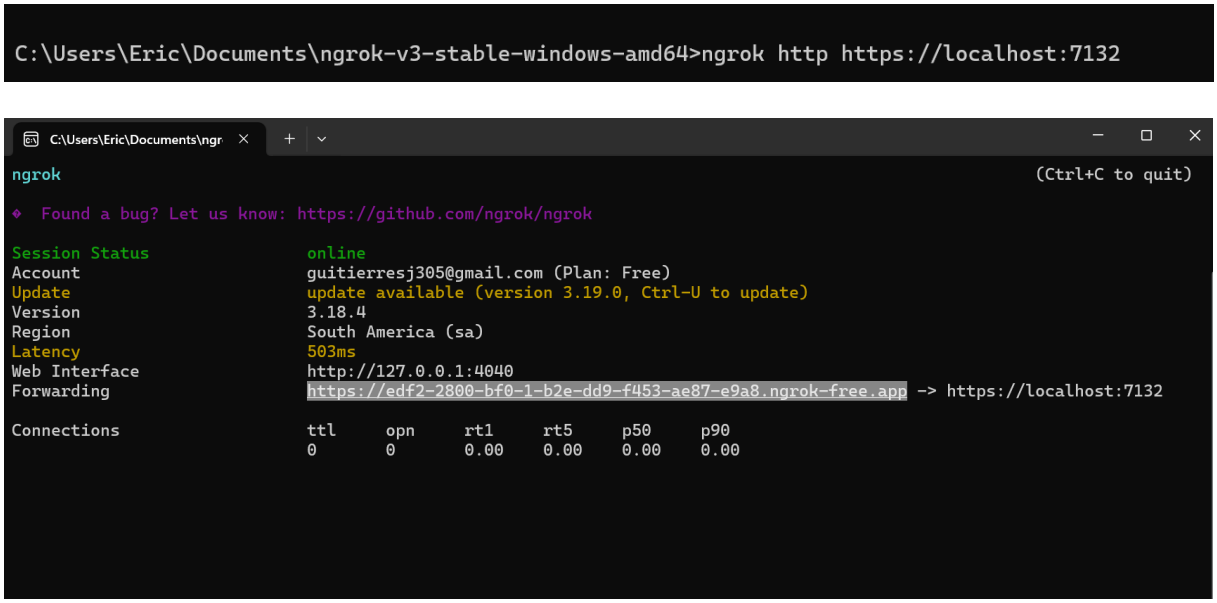
Nota. Ejecución de Ngrok, Elaborado por: El autor

Observamos que con la línea de comando ngrok http https://localhost:7132, nos va a entregar nuestra IP publica nueva para colocar en nuestro entorno de desarrollo front y back para que se pueda ejecutar los servicios de nuestro aplicativo móvil.

Figura 46

Línea de código para Ejecutar ngrok

La aplicación ngrok nos da acceso una dirección IP publica generado ubicado en USA



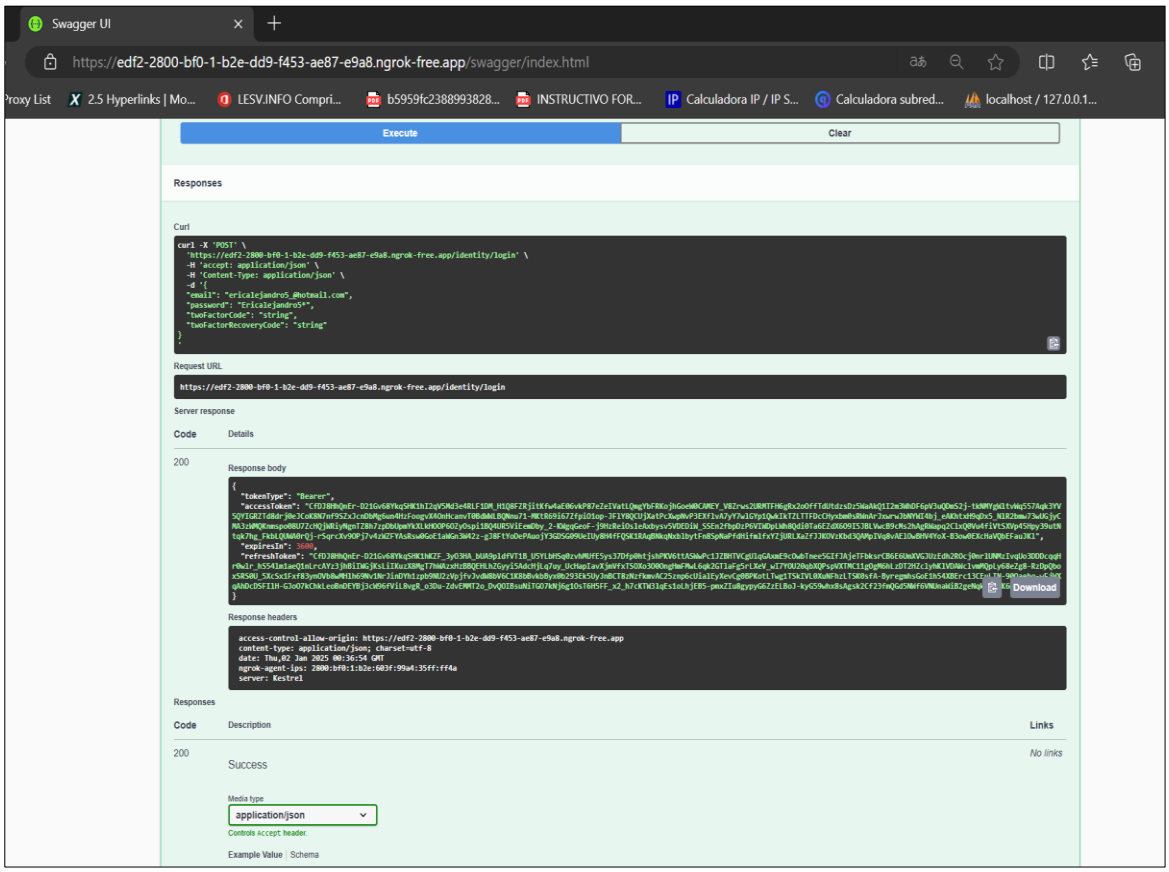
Nota. Código de ejecución del programa, Elaborado por: El autor

Podemos observar que nuestra dirección creada por ngrok de forma dinámica https://edf2-2800-bf0-1-b2e-dd9-f453-ae87-e9a8.ngrok-free.app ya está creada como una IP publica validamos funcione realizando una prueba en nuestro login y vemos que recibimos como mensaje 200 Success que significa que está funcionando correctamente

Figura 47

Imagen de la IP pública.

Realizando este procedimiento podemos obtener nuestra ip publica la API



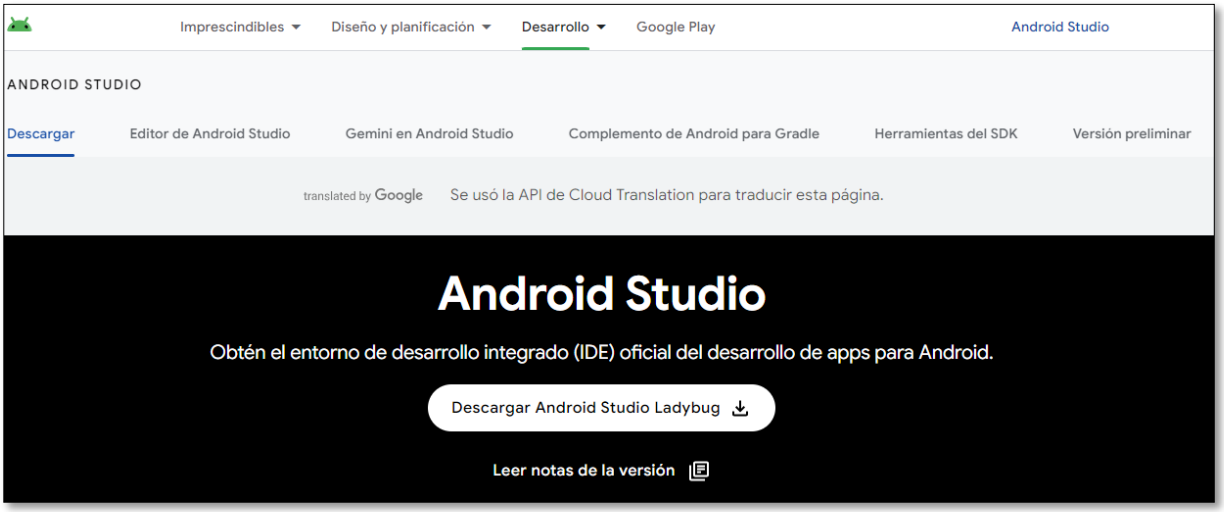
Nota. Pantalla de comprobación de servicios, Elaborado por: El autor

4.2.23. *Instalación de Android Studio*

Se accede al link de descargar de Android Studio para poder visualizar nuestro emulador en el computador.

Figura 48

Instalación Android Studio

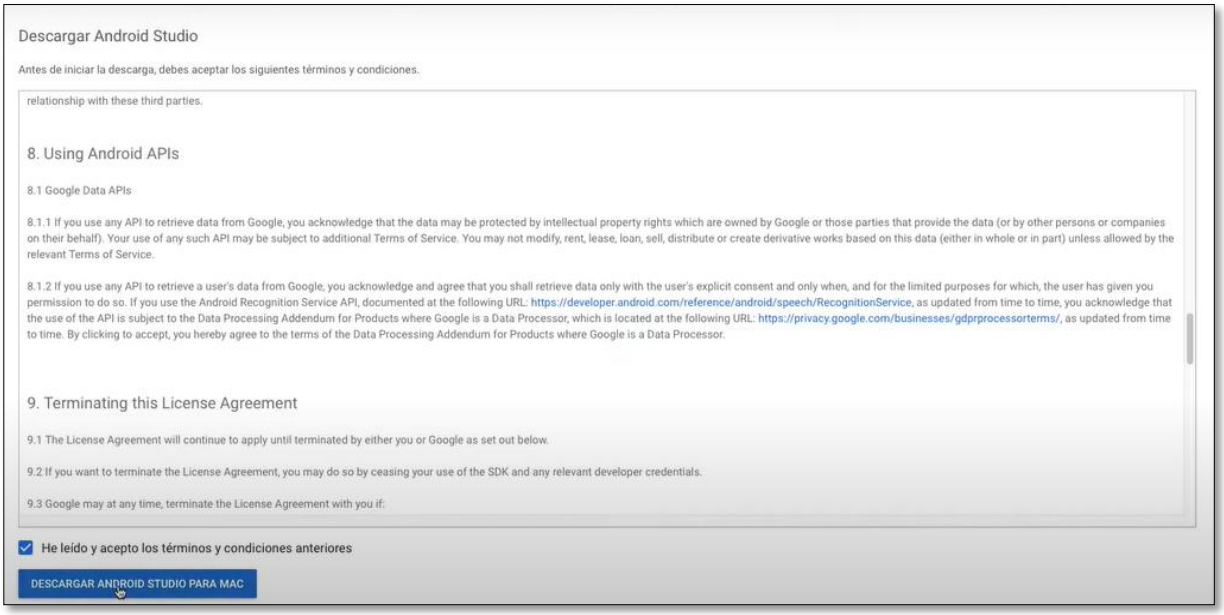


Nota. Pantalla principal de Android studio, Elaborado por: El autor

El siguiente paso es poder aceptar los términos y detenidamente revisar en que sistema operativo vamos a trabara si Mac o Windows y descargamos la aplicación

Figura 49

Aceptamos Términos y Condiciones



Nota. Pantalla de aceptación de términos para descargar ANDROID STUDIO, Elaborado por: El autor

Figura 50

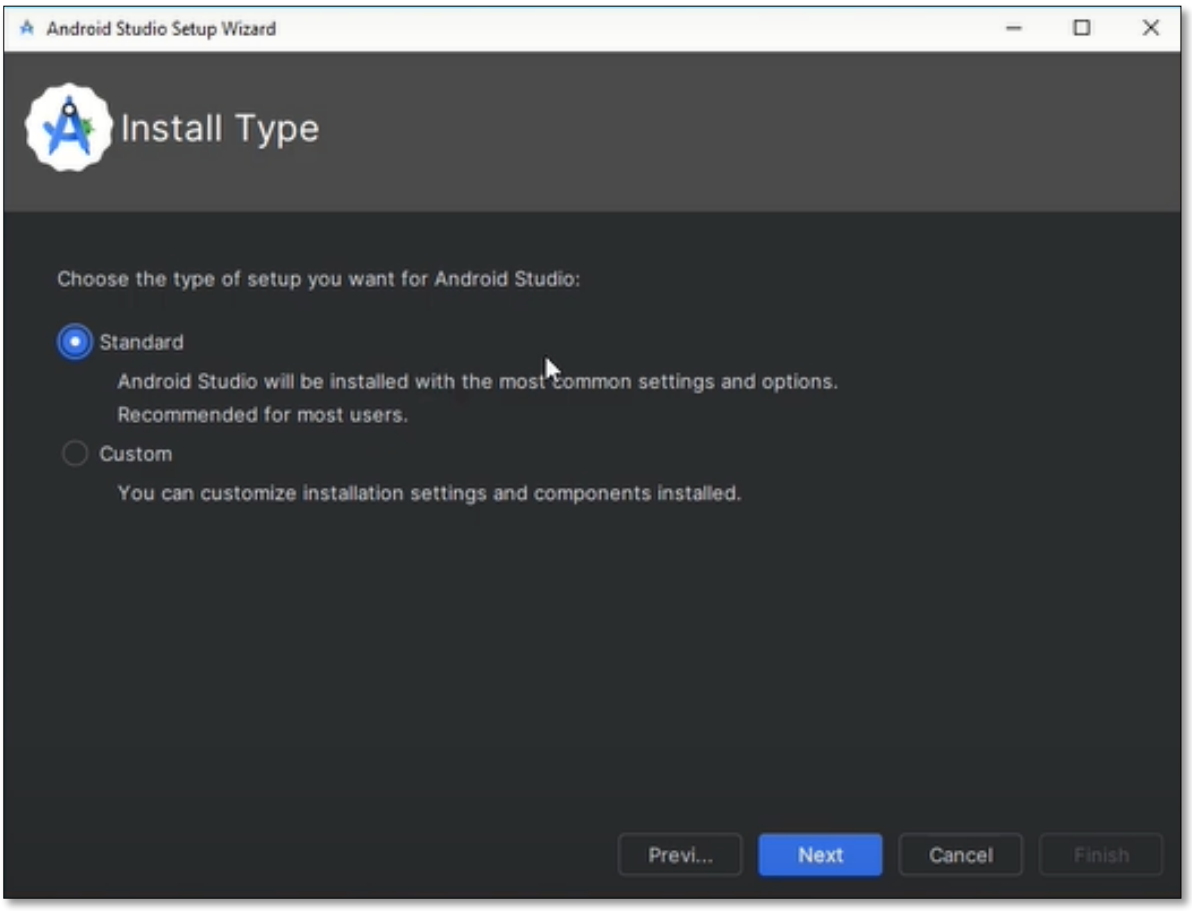
Descarga Finalizada

En esta pantalla nos da la bienvenida como vemos esta activo el check para que podamos acceder al panel de Android Studio

Nota. Pantalla de inicio de instalación, Elaborado por: El autor

Figura 51

Estándar Actualizado



Nota. Pantalla de tipo de instalación, Elaborado por: El autor

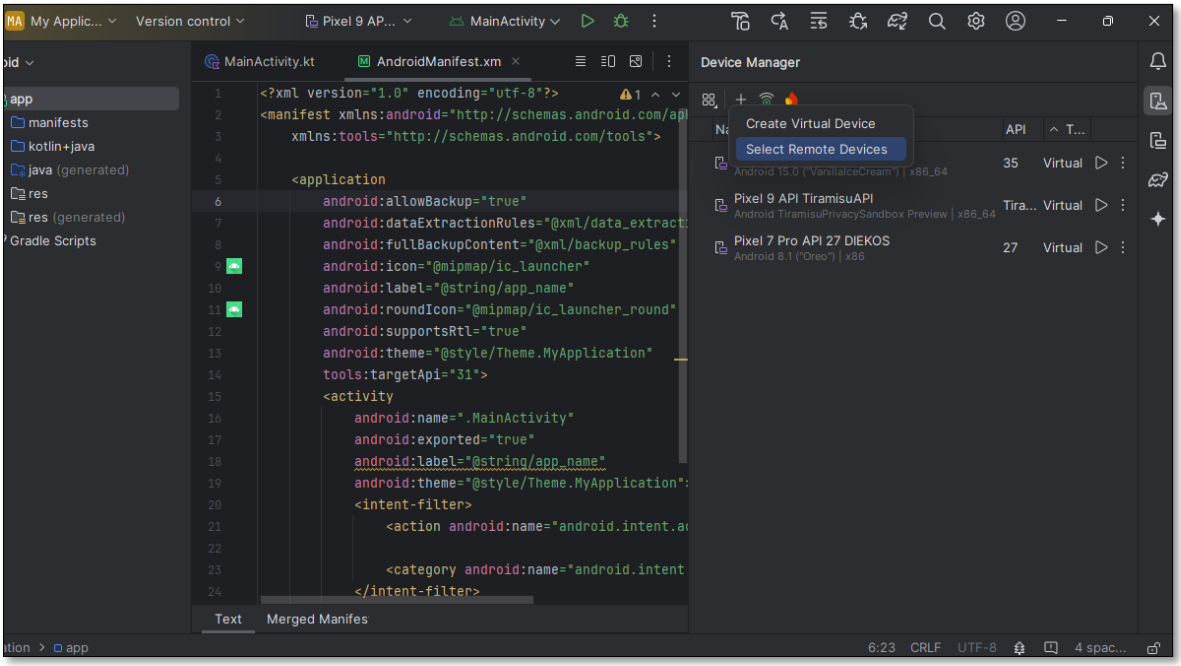
Es importante instalar en estándar ya que las configuraciones nos recomiendan porque nos permite actualizar en casos de error o mensajes de Google.

4.2.24. *Creación de emulador Android*

Terminado el proceso de descarga podemos ingresar a nuestra aplicación a poder crear el emulador en Android estudio seleccionando Create Virtual Devices

Figura 52

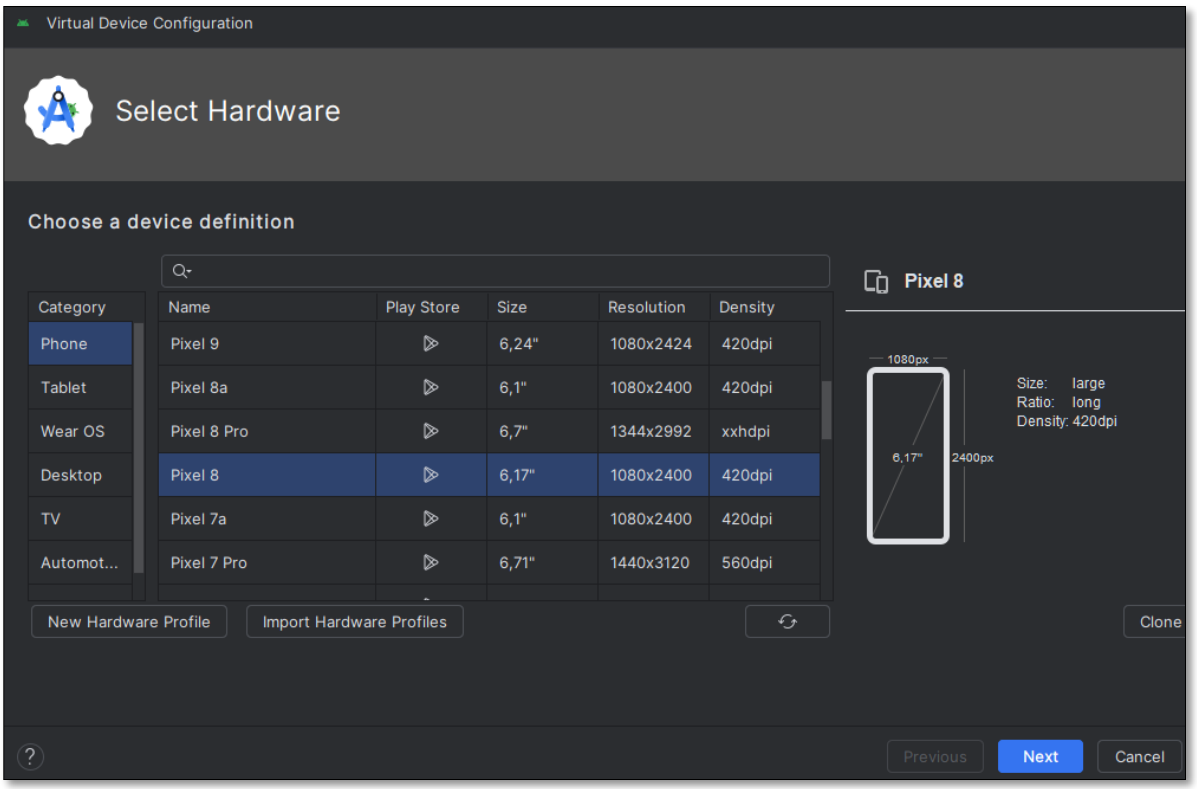
Aplicación Android Studio



Nota. Pantalla Inicio de aplicación Android Studio cuando esta instalado, Elaborado por: El autor

Figura 53

Instalación del Hardware del Móvil



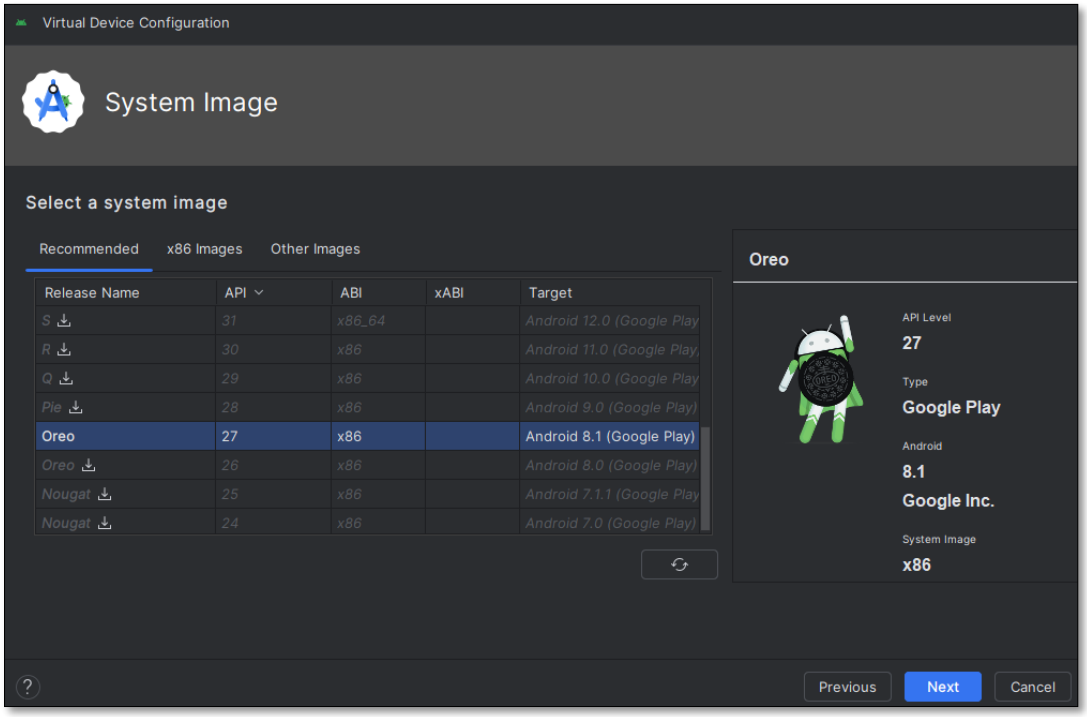
Nota. Pantalla Inicio donde se selecciona el Software de uso IOs/Android, Elaborado por: El autor

Seleccionamos el Hardware pixel 7 Pro que para nuestro computador pueda trabajar sin interrupciones como sabemos la aplicación de Android Studio requiere de más memoria en el pc

Figura 54

Instalación de Imagen

Seleccionamos la imagen para la instalación de nuestro emulador Android existen varias imágenes que es necesario descargar lo que usaremos si disponemos de un almacenamiento o memoria del pc adecuado.

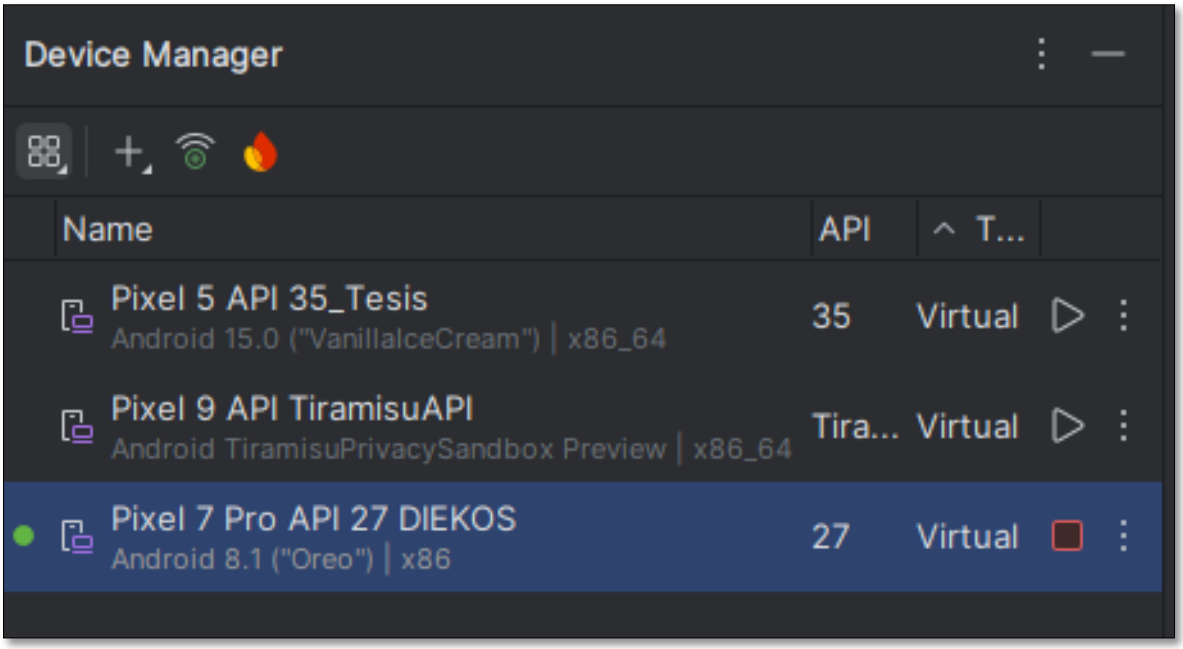


Nota. Elección de Imagen de acuerdo a las necesitadas Elaborado por: El autor

Figura 55

Emulador Android Descargado

Yo he realizado la instalación de 3 emuladores para poder realizar nuestras pruebas necesarias de la API así poder escoger cual puedo utilizar sin ningún inconveniente.

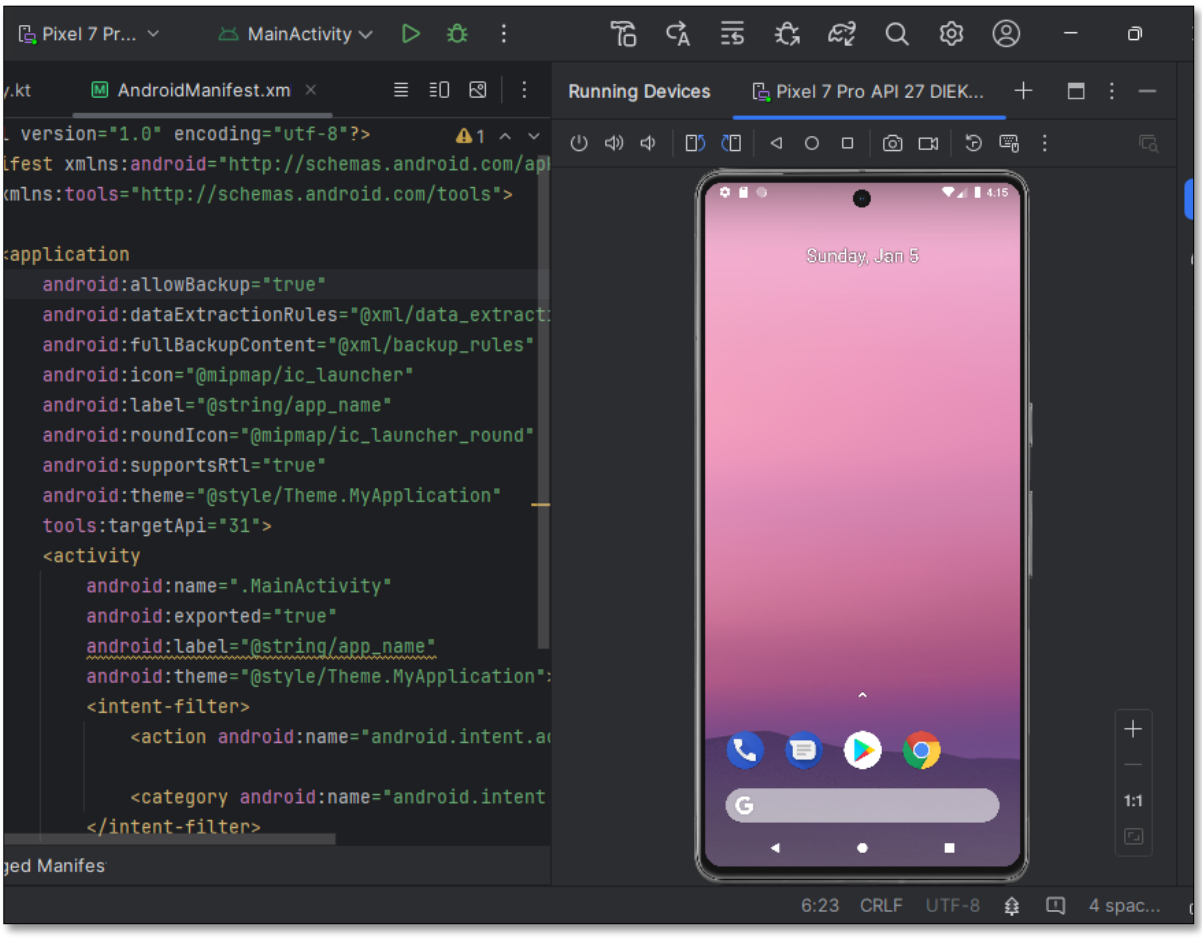


Nota. El emulador que se está usándose encuentra activo cuando tiene color rojo Elaborado por: El autor

Figura 56

Dispositivo Virtual

Nos muestra la ejecución del emulador sin ningún inconveniente

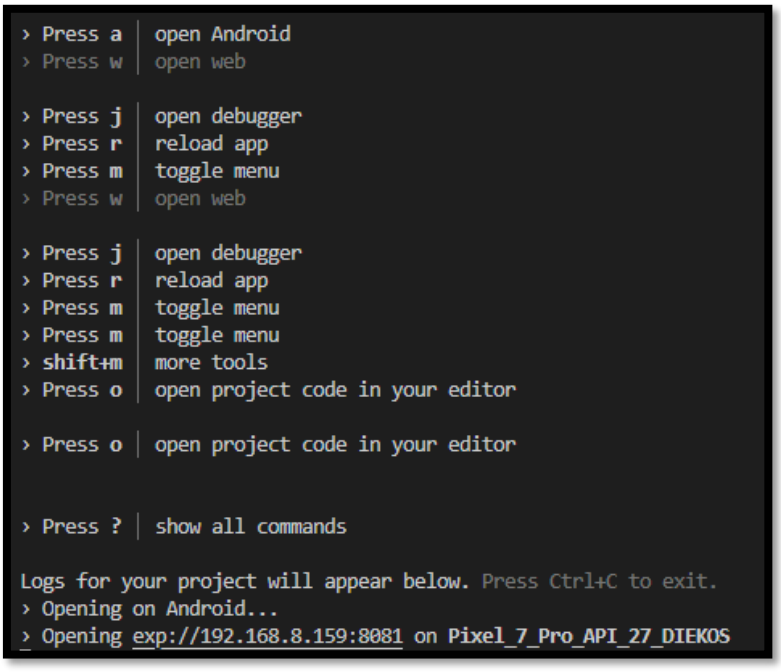


Nota. Pantalla del quipo emulador seleccionado en procesos anteriores. Elaborado por: El autor

Para poder presentar nuestra API en el emulador ejecutamos en nuestro entorno desarrollo que es visual code para poder encontrar nuestro móvil utilizando expo

Figura 57

Ejecución de Código expo

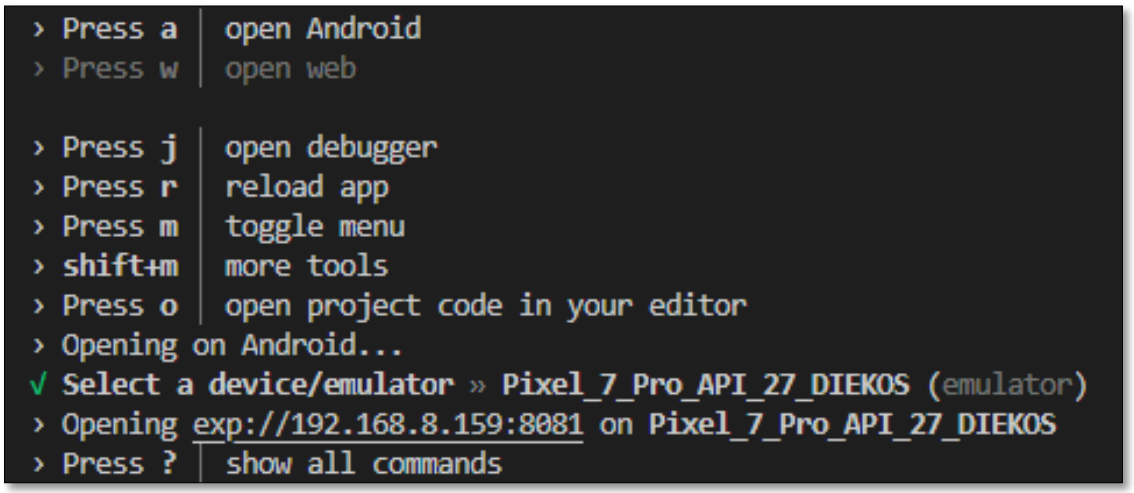


Nota. Ejecución de los procesos. Elaborado por: El autor

Figura 58

Selección de Dispositivo Android

Escogemos nuestro emulador con shift + a para seleccionar los emuladores disponibles como vemos en la imagen se muestra un check de color verde.



Nota. Selección de dispositivo Android. Elaborado por: El autor

4.2.25. *Descargar App móvil en Play Store EXPO*

Expo es una herramienta de desarrollo que centra en escribir codigo en React Native sin necesidad de configuraciones complejas basado en proyectos nativos para IOS y Android

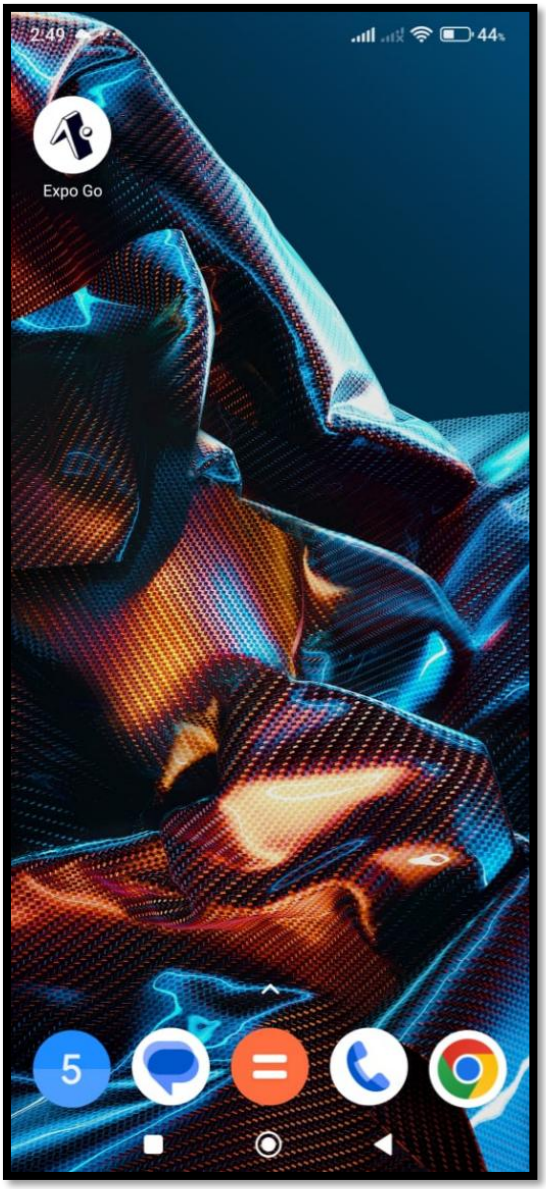
Figura 59

Descarga de Aplicación Expo por la tienda Play Store

Nota. Aplicación instalada en el celular de Expo Geo Elaborado por: El autor.

Figura 60

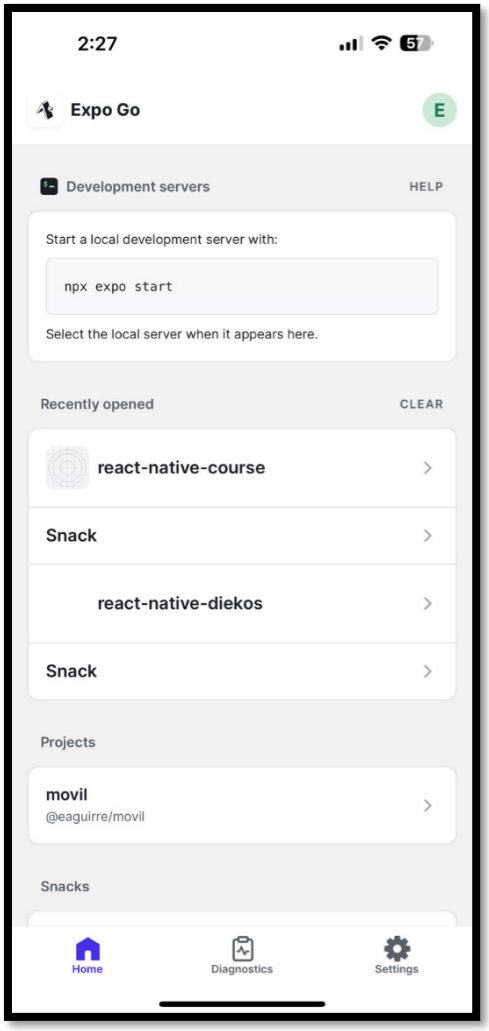
Aplicación Expo en Móvil



Nota. La pantalla muestra la aplicación Expo GO en su celular Elaborado por: El autor.

Figura 61

API Diekos Reconocido



Nota. Se presenta la pantalla de la aplicación Elaborado por: El autor.

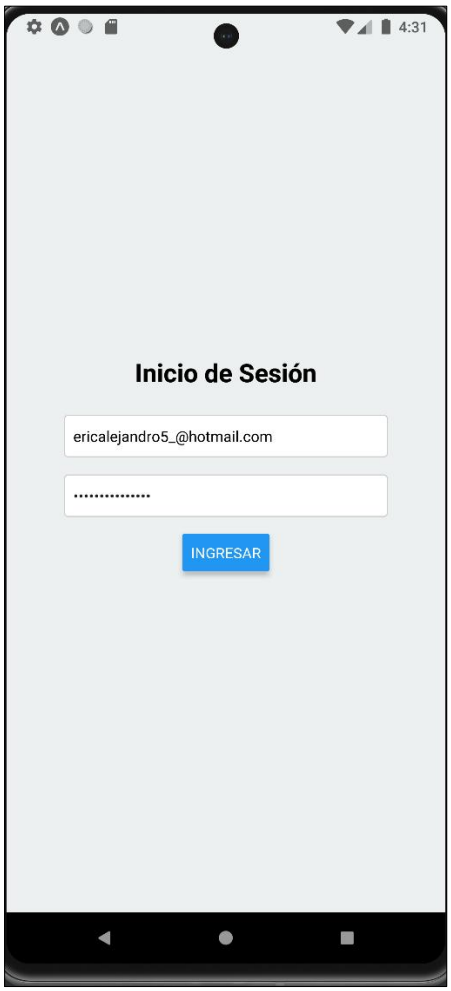
4.2.26. *Login Aplicativo Movil (Front)*

Ingresamos con nuestras credencias al panel de pedidos

Figura 62

Login de Usuario

Desde nuestro emulador ingresamos al panel de pedidos



Nota. Pantalla de inicio de sesión móvil. Elaborado por: El autor.

4.2.27. *Servicio Modulo para hacer el pedido del producto (Back)*

Para este desarrollo del formulario voy a explicar el código de componente React Native que es creado con el nombre de PedidoUsuario al iniciar el componente obtiene los datos del cliente que está registrado en la base de datos, usuarios activos y productos disponibles en la API adicional a esto también podemos ver que hay un botón para cerra sección

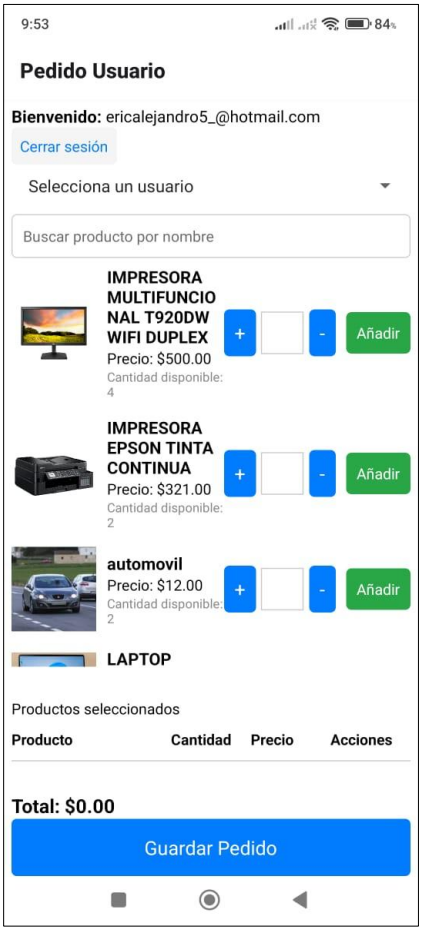
4.2.28. *Modulo para hacer el pedido del producto (Front)*

Ingresamos con nuestras credencias al panel de pedidos

Figura 63

Formulario de Pedido

Imagen donde se encuentras todos los pedidos activos.



Nota. Pantalla de pedidos. Elaborado por: El autor.

4.2.29. *Servicio crear pedido móvil (Back)*

Se encargar de guardar el pedido dentro de nuestra función llamada hadleGuardarPedido lo que realiza es una solicitud HTTP al servidor verificando que se haya seleccionado un usuario o varios productos que sean seleccionados igual crea un objeto con nuestro ID de usuario y productos seleccionados

Figura 64

Imagen Guardar Pedido.

Desde nuestro emulador ingresamos al panel de pedidos

```
const handleGuardarPedido = async () => {
    if (!selectedUsuario) {
        Alert.alert("Error", "Por favor, selecciona un usuario.");
        return;
    }
    if (selectedProductos.length === 0) {
        Alert.alert("Error", "Por favor, selecciona al menos un producto.");
        return;
    }

    const pedido = {
        usuarioId: selectedUsuario,
        productos: selectedProductos
    };

    try {
        const response = await fetch('https://edf2-2800-bf0-1-b2e-dd9-f453-ae87-e9a8.ngrok-free.app/api/pedidosusuario', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(pedido),
        });

        if (response.ok) {
            Alert.alert("Éxito", "Pedido guardado exitosamente.");
            setSelectedUsuario('');
            setSelectedProductos([]);
            setCantidadProducto({});
        } else {
            Alert.alert("Error", "Ocurrió un error al guardar el pedido.");
        }
    } catch (error) {
        console.error("Error al guardar el pedido:", error);
        Alert.alert("Error", "Ocurrió un error al guardar el pedido.");
    }
};
```

Nota. Pantalla de pedido. Elaborado por: El autor.

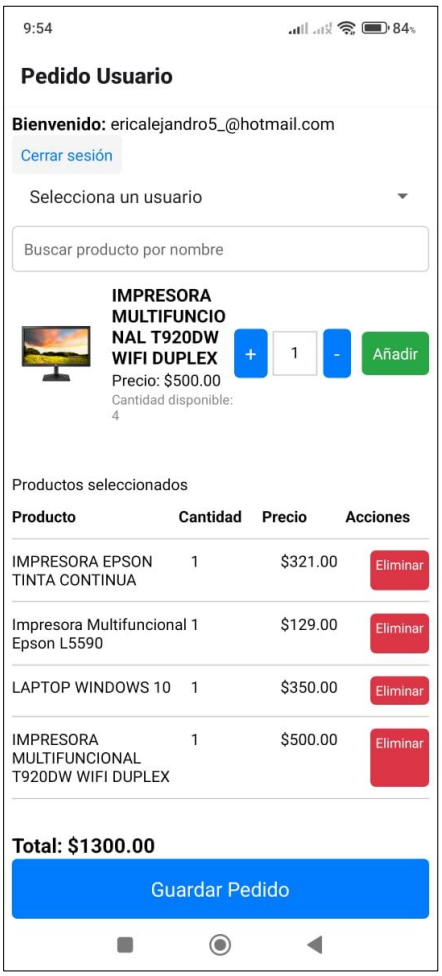
4.2.30. *Crear pedido móvil (Front)*

El formulario está diseñado para crear un nuevo pedido como lo vemos en la imagen se seleccionó 4 productos para guardar el pedido.

Figura 65

Imagen Crear Pedido

Todo el formulario que ves en la aplicación móvil puede seleccionar producto buscar y eliminar



Nota. Pantalla de crear pedido. Elaborado por: El autor.

4.2.31. *Servicio Editar Pedido Móvil (Back)*

Nos permite cambiar la cantidad de un producto seleccionado por el usuario

Figura 66

Código Editar Pedido

La función handleCantidadChange realiza el cambio de la cantidad del producto

```
90    const handleCantidadChange = (id, cantidad, maxCantidad) => {
91        if (cantidad > 0 && cantidad <= maxCantidad) {
92            setCantidadProducto(prevState => ({
93                ...prevState,
94                [id]: cantidad,
95            }));
96        }
97    };
```

Nota. Código de editar pedido. Elaborado por: El autor.

4.2.32. *Servicio eliminar pedido móvil (Back)*

Eliminar producto de la lista de seleccionados

Figura 67

Código Eliminar Pedido.

La función realiza una eliminación ene l móvil de su producto escogido

```
112   const handleRemoveProducto = (productoId) => {
113       setSelectedProductos(selectedProductos.filter(prod => prod.productoId !== productoId));
114   };
```

Nota. Código de eliminar pedido. Elaborado por: El autor.

4.2.33. *Editar Pedido Móvil (Front)*

Si el Usuario desea escoger más de un producto en la parte de productos seleccionados podemos ver la actuación de su nuevo pedido hasta que el cliente este satisfecho de los productos a comprar.

Figura 68

Imagen Editar Pedido

La función handleCantidadChange realiza el cambio de la cantidad del producto

Nota. Código de editar pedido. Elaborado por: El autor.

4.2.34. *Eliminar Pedido Móvil (Front)*

En la imagen podemos observar que teníamos 4 productos lo cual eliminado 2 y vamos a guardar el pedido con 2 productos que nosotros como usuarios escogimos.

Figura 69

Código Eliminar Pedido.

La función handleCantidadChange realiza el cambio de la cantidad del producto

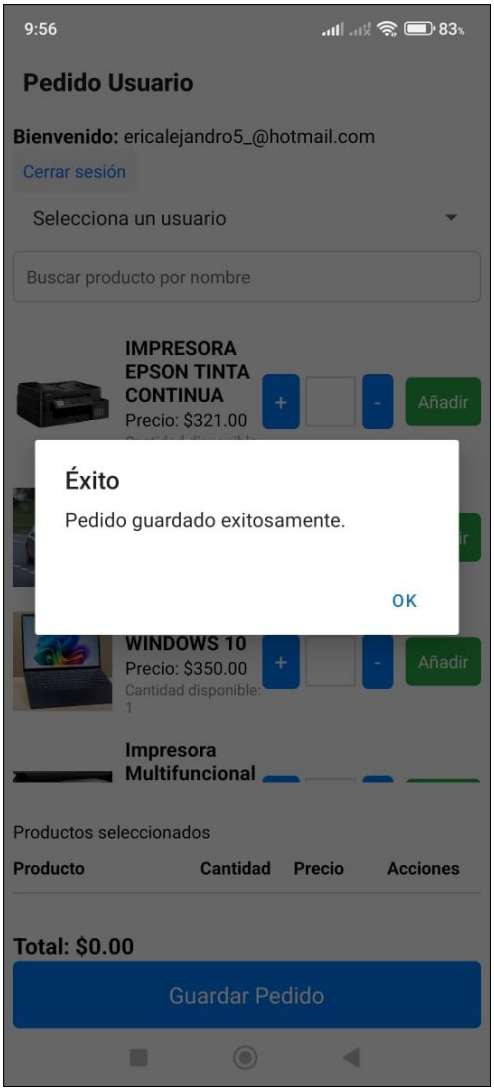
Nota. Código de eliminar pedido. Elaborado por: El autor.

El pedido fue completado con éxito ahora si desean seguir comprando pueden hacerlo y si no tienen algún pedido a realizar pueden cerrar la sección.

Figura 70

Imagen Pedido Guardado

Para comprobar que nuestro pedido fue completado con éxito también podríamos verlo en nuestra base sql que todo el pedido esta guardado.



Nota. Notificación de pedido guardado exitoso. Elaborado por: El autor.

CONCLUSIONES

El desarrollo del sistema de gestión de bodegación multifuncional administrar la bodega permitiendo a las empresas automatizar y optimizar la gestión de inventarios, lo que llevará a una mayor eficiencia en las operaciones diarias. Al disminuir la intervención manual, se obtendrá la reducción de e errores y se mejorarán los tiempos de respuesta en la gestión de productos y pedidos dentro de la bodega.

La implementación de este software nos ayudara a reducir los costos operativos relacionados con el almacenamiento y manejo de los productos. El control en tiempo real del inventario y la optimización de las áreas y espacios de la bodega ayudarán a evitar múltiples pérdidas de productos y reducir la realización de inventarios innecesarios.

El software de gestión de bodega realizado por Diekos S.A. para las empresas obtenido entregas garantizará más ágil para los pedidos, lo que resultará más entregas puntuales y precisas. Esto fortalecerá la relación con los clientes, quienes valoran la confiabilidad y la rapidez en la entrega de productos, mejorando la reputación de la empresa.

Al automatizar los procesos y mejorar la eficiencia operativa, permitirán que las empresas estén mejor posicionadas para presentar ofertas y obtener demandas de sus productos de igual manera cubriendo las necesidades del cliente y manejo eficiente de recursos en una ventaja competitiva en el mercado.

RECOMENDACIONES

Es crucial que las empresas capaciten a su personal en el uso del nuevo sistema de bodegación multifuncional propuesto por Diekos. S.A. La automatización puede generar resistencia al cambio si no se les proporciona el entrenamiento adecuado. Invertirnos en tiempo y recursos de formar que los empleados permitirán un mejor uso más eficiente y aprovechamiento de las ventajas software.

El software debe ser flexible y adaptable a las necesidades específicas y requeridas por la empresa. Recomiendo que se disponga la funcionalidad de realizar actualizaciones periódicas y ajustando los cambios de las operaciones del mercado tecnológico para mantener la competitividad.

Una vez implementado el sistema, se debe monitorear su desempeño regularmente y evaluar si realmente está logrando los objetivos de optimización, reducción de costos y mejora en la satisfacción del cliente. Este análisis obtenido indicara y detectara las áreas qué requieren mejora y ajuste del proceso según sea necesario.

Es aconsejable que el sistema de bodegas multifuncional se fusiona con otros sistemas empresariales, tales como los de ventas, facturación y distribución de productos. Esto facilitará el intercambio de información, disminuyendo la posibilidad de equivocaciones y optimizando la toma de decisiones.

REFERENCIAS

Bowersox, D. J., Closs, D. J., & Cooper, M. B. (2013). *Supply chain logistics management* (5th ed.). McGraw-Hill.

Chopra, S., & Meindl, P. (2016). *Supply chain management: Strategy, planning, and operation* (6th ed.). Pearson.

Lambert, D. M., Stock, J. R., & Ellram, L. M. (2013). *Fundamentals of logistics management*. McGraw-Hill.

Kumar, S., & Saini, R. (2014). Warehouse management: *Automation and optimization. Springer*

Heizer, J., & Render, B. (2017). Operations management*: Sustainability and supply chain management* (12th ed.). Pearson.

Laudon, K. C., & Laudon, J. P. (2014). Management information systems: *Managing the digital firm* (13th ed.). Pearson.

Mentzer, J. T., Min, S., & Zacharia, Z. G. (2001). Defining supply chain management. *Journal of Business Logistics*, 22(2), 1-25. https://doi.org/10.1002/j.2158-1592.2001.tb00001.x

Turban, E., Pollard, C., & Wood, G. (2018). Information technology for management: *Digital strategies for insight, action, and sustainable performance* (11th ed.). Wiley

React. (12 de mayo de 2023). Creación y anidamiento de componentes https://react.dev/learn

DevDocs. (21 de junio de 2023). Creación de interfaces de usuario a partir de componentes https://devdocs.io/react/

React. (23 de marzo de 2023). Components and Props. https://legacy.reactjs.org/docs/components-and-props.html

FlutterFlow. (2024). FlutterFlow. Desarrollo visual que permite crear aplicaciones móviles https://flutterflow.io/

Tailwind CSS. (2024). Tailwind CSS. Framework de CSS para diseño rápido de interfaces. https://tailwindcss.com

Node.js. (s.f.). Download | Package Manager. https://nodejs.org/en/download/package-manager

TechRiders. (s.f.). Login y registro con Auth0 en React. https://techriders.tajamar.es/login-y-registro-con-auth0-en-react/

Microsoft. (2024). Visual Studio. Entorno integrado de desarrollo (IDE) para múltiples plataformas. https://visualstudio.microsoft.com/es/

Vite. (2024). Vite. Herramienta de compilación rápida para aplicaciones web modernas. https://vitejs.dev/

Google. (2024). Android Studio. Entorno de desarrollo integrado oficial para Android. https://developer.android.com/studio

Expo. Expo: Herramienta para el desarrollo rápido de aplicaciones con React Native. https://expo.dev/

Meta Platforms. (2024). React Native. Framework para crear aplicaciones móviles multiplataforma con JavaScript. https://reactnative.dev/

FlutterFlow, Inc. (2024). FlutterFlow. Plataforma de desarrollo visual para aplicaciones móviles y web. https://www.flutterflow.io/

Lawani, E. J. (2024). Form validation with YUP. DEV Community. https://dev.to/lawaniej/form-validation-with-yup-2kmg