

Proyecto Lenguajes de Programación

Título del Proyecto:

Aplicación de compra/venta de bienes o servicios - TiendaSegura

Integrantes del Grupo: G10

- Sebastian Holguin Vargas
- Derian Baque Choez
- Carlos Ronquillo Brush

1. Problemática a resolver

La mayoría de los locales comerciales / tiendas están siendo forzadas a cerrar debido a la ola de inseguridad y extorsiones (vacunas) que se viven día a día en el país. Los negocios más afectados son las tiendas de barrio o locales para las comunidades por su fácil exposición a la inseguridad ya que no cuentan con seguridad privada y el modo de atención al cliente es el tradicional abierto al público, esto expone al tendero a ser identificado, amenazado y por último una víctima más de inseguridad y extorsión. Actualmente estas tiendas no pueden operar bajo pedido o modalidad de “tienda oculta a puerta cerrada”.

Este proyecto busca preservar la economía barrial y proteger la vida de los comerciantes y clientes. Al trasladar toda la interacción de compra / venta a un entorno digital y seguro, se elimina la necesidad de tener la reja abierta y reduce el riesgo de extorsión.

2. Objetivos Propuestos

Objetivo General:

- Ofrecer una solución digital segura y adaptable que permita a los comercios de barrio operar bajo la modalidad de “tienda oculta a puerta cerrada”, minimizando la exposición a la inseguridad y extorsión.

Objetivos Específicos:

- Elaborar un sistema de gestión de catálogo que permita al tendero cargar sus productos y mantener la oferta de su negocio actualizada, este proceso no deberá tomar más de 1 minuto por producto.
- Desarrollar funcionalidades de búsqueda, secciones de detalle y compra de los productos en la aplicación del cliente.
- Implementar un sistema de autenticación para tendero, vecinos y/o clientes, garantizando el acceso a la plataforma de manera segura.

- Simular un botón de emergencia en caso de que ocurra alguna amenaza para el tendero.

3. Alcance del Proyecto

Incluye:

- App para el Tendero: Gestiona pedidos, catálogo, botón de pánico al 911.
- App para el Vecino / Cliente: Visualiza catálogo y pedidos, incluye carrito de compras y simulación de pagos.

No incluye:

- Integración de sistemas de cobros bancarios.
- Lógica de mapa para delivery.

4. Características de la Solución

1. Gestor de productos e inventario para tiendas pequeñas.
2. Búsqueda automática de imágenes de productos con base al nombre y marca del mismo.
3. Botón de pánico al 911.

5. Requerimientos Funcionales Asignados

Requerimiento	Descripción	Responsable
Crear un producto (escritura)	Esta funcionalidad permite al tendero crear productos y cargarlos en el catálogo disponible de la tienda con la siguiente información x producto: stock, nombre, categoría, precio, imagen del producto.	Sebastian Holguin
Ver pedidos de los clientes (lectura y escritura)	Esta función permite al tendero ver los pedidos de los clientes por orden de llegada (fecha) y aceptar o rechazar el pedido si así el tendero lo considera.	Sebastian Holguin
Buscar imagenes de productos automáticamente (scrapping)	Esta opción permite al tendero cargar una imagen del producto automáticamente con base al nombre y marca del mismo. Además de tener la opción de aceptar la sugerencia de la	Sebastian Holguin

	imagen o tomar por sí mismo una fotografía del producto.	
Usar carrito de compra (escritura)	Esta función permite al cliente agregar productos al carrito de compras y almacenar dicha información en la base de datos MongoDB, validando un límite máximo de 25 ítems por carrito.	Derian Baque
Visualizar y actualizar el total de la lista de productos carrito de compra (lectura y escritura)	Esta función permite consultar el carrito almacenado en la base de datos, calcular dinámicamente el total de los productos y actualizar el valor total del carrito en la base de datos, mostrando al cliente el precio final.	Derian Baque
Consultar productos y tiendas cercanas. (lectura)	Esta función permite buscar productos y tiendas disponibles. El cliente podrá buscar los productos a través de una barra de búsqueda ingresando el nombre del producto, marca o categoría.	Carlos Ronquillo
Guardar historial de búsqueda y compras (lectura y escritura)	El usuario puede consultar su historial de compras clasificando las compras por fecha y tienda.	Carlos Ronquillo

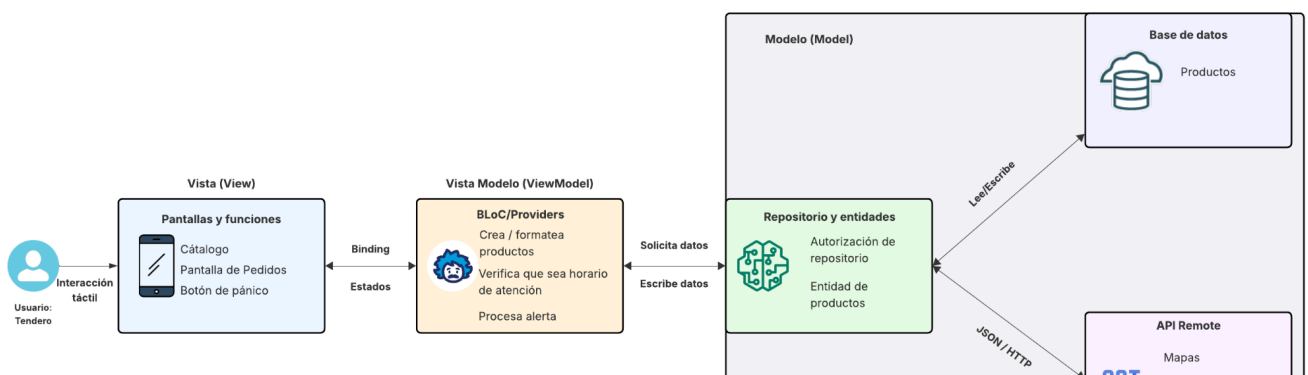
7. Arquitectura de la Aplicación

Tipo de Aplicación: Móvil

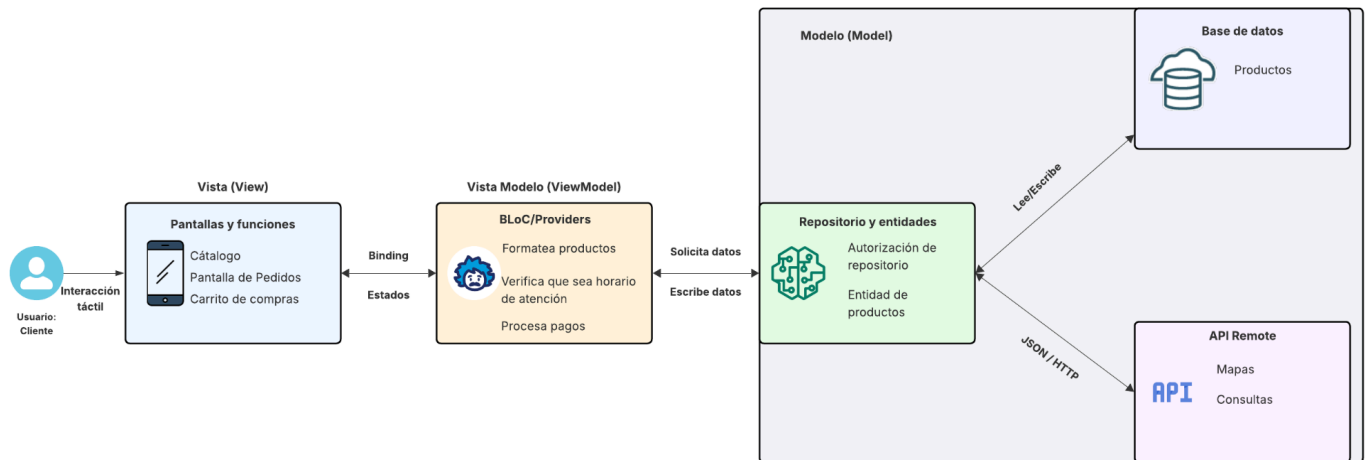
Las tiendas de barrios, especialmente de comunidades marginadas, los tenderos suelen tener celulares de gama media-baja y requieren de movilidad constante dentro del local además de que incluye la facilidad de tomar fotos. Una aplicación móvil permite el uso de notificaciones a los usuarios por lo que el tipo de aplicación móvil es la ideal.

Patrón de Arquitectura: MVVM

Tendero



Cliente/Vecino



Herramientas y Frameworks:

- Frameworks:
 - Flutter: Se debe desplegar la aplicación en Android que es la marca de celulares dominante en las comunidades y también para desplegar la app en simultáneo en sistemas iOS.
 - NodeJs: Indispensable para el backend con base en JavaScript/TypeScript.
- API:
 - Google, para integrar consultas de Google Maps e incluso el uso de Gemini y búsqueda de imágenes integradas.
- Servicios:
 - Autenticación: Firebase para autenticación de credenciales.
 - Base de datos: MongoDB (NoSQL) para la flexibilidad en la subida y cambios de productos en la tienda.

8. Lenguajes de Programación

Back-end:

- **TypeScript - JavaScript (Conocido)**
 - **Pros:**
 - Fácil de leer
 - **Tipo de datos:** TypeScript usa tipado fuerte sobre JavaScript. Esto facilita definir interfaces estrictas.

Confiable: Como es de tipado fuerte detecta los errores de datos antes de compilar.

- Fácil de escribir

Abstracción: El lenguaje posee un nivel alto de abstracción mediante clases y módulos, además que cuenta con el ecosistema NPM.

- o **Contras:**

- Dificultad para leer

Al ser un superconjunto de JavaScript, a veces permite referencias mutables que pueden llevar a confusiones si no se configuran las reglas en TSConfig.

Simplicidad: El manejo de promesas asincrónicas

Front-end:

- **Dart (no conocido)**

- o **Pros:**

- Fácil de escribir

Expresividad: Dart es muy expresivo al momento de escribir interfaces. Al declarar los componentes esto permite describir visualmente la jerarquía de la pantalla en el propio código.

Ortogonalidad: Dart es muy consistente, la declaración de componentes/widgets es reutilizable durante todo el código.

- **Fácil de leer:**

Tipo de datos: Dart utiliza seguridad nula sólida, esto garantiza que las variables no puedan ser nulas, eliminando errores comunes.

- o **Contras:**

- Dificultad para leer:

Diseño de sintaxis: Debido a su estructura de árbol de widgets, el código tiene indentación profunda y muchos paréntesis de cierre, lo cual afecta un poco en la legibilidad.

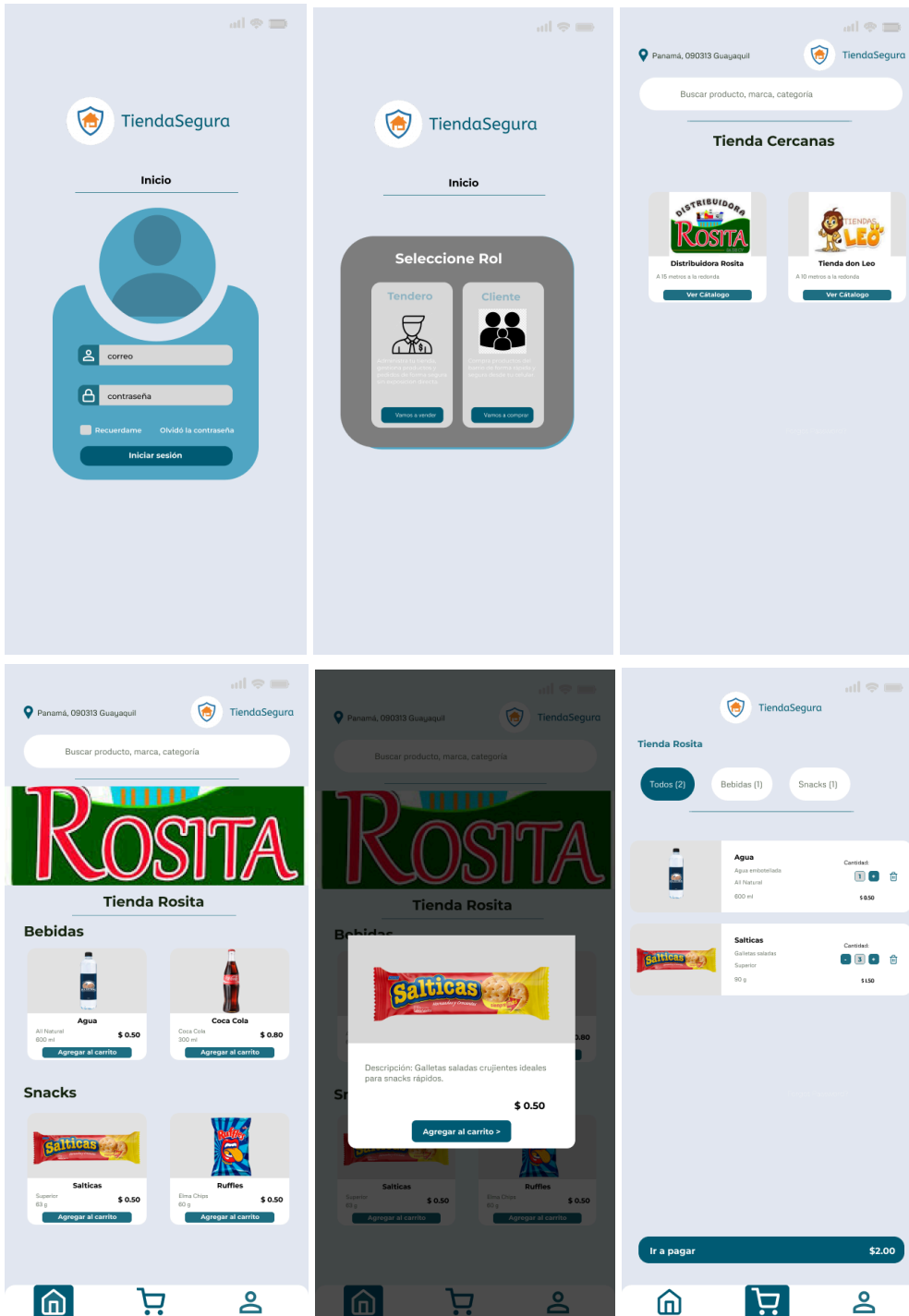
Simplicidad: Para nuestro grupo que viene de ver la materia de Desarrollo Web Y Aplicaciones Móviles puede afectar un poco en la curva de aprendizaje ya que Dart mezcla lógica y diseño en el mismo archivo.

9. Prototipo de Baja Fidelidad

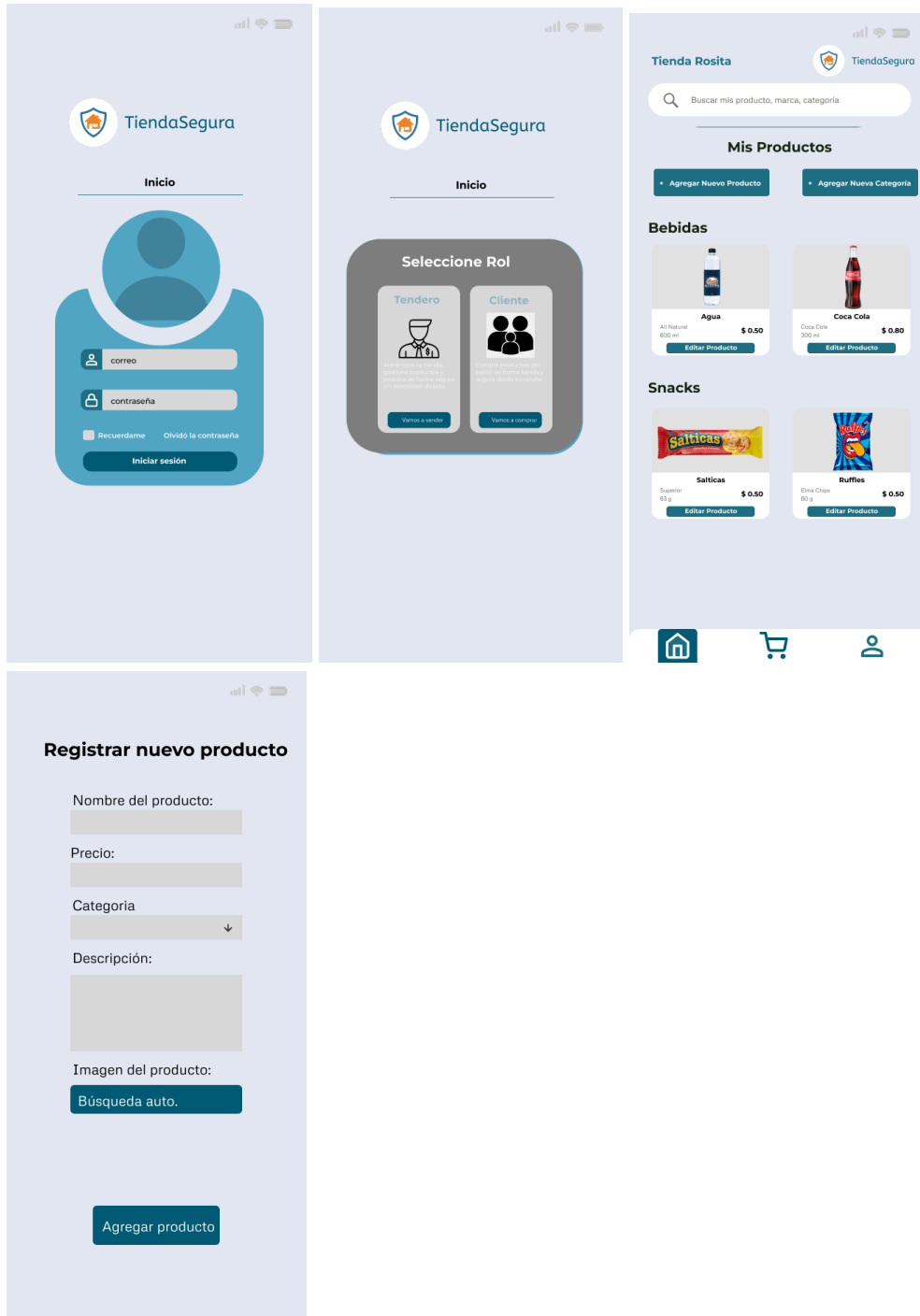
Herramienta de Diseño Utilizada: Figma

Capturas:

Cliente



Tendero



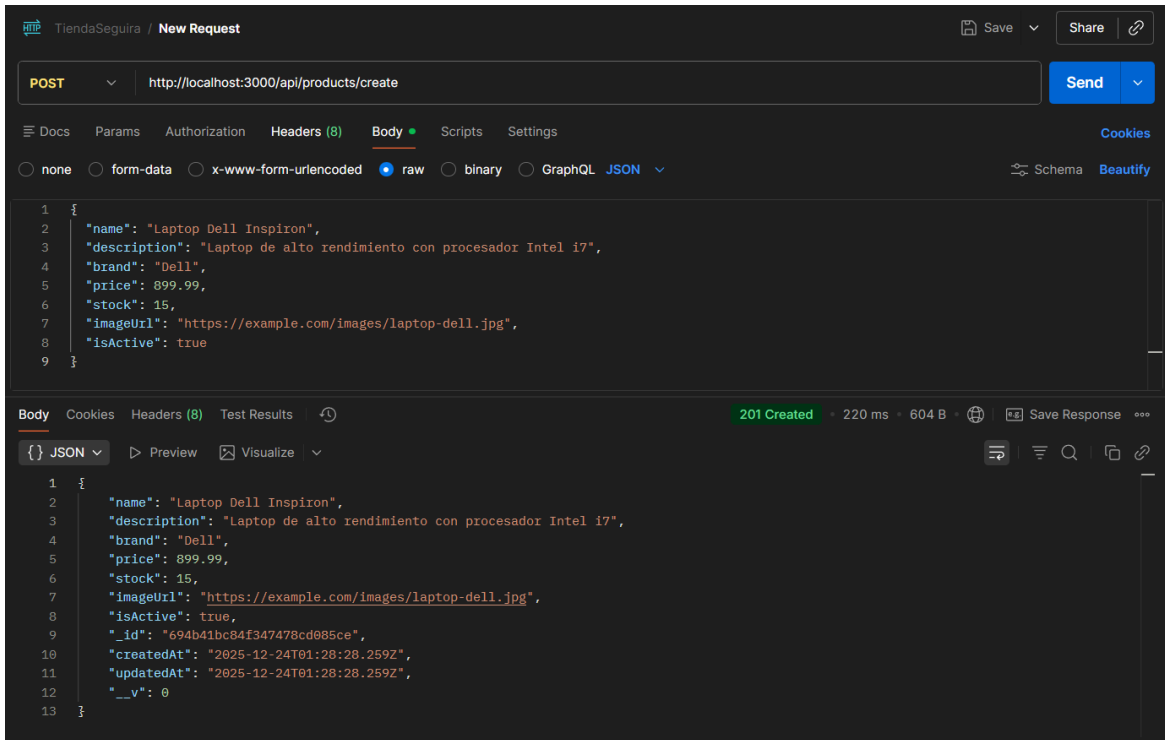
10. Implementación

10.1 Backend

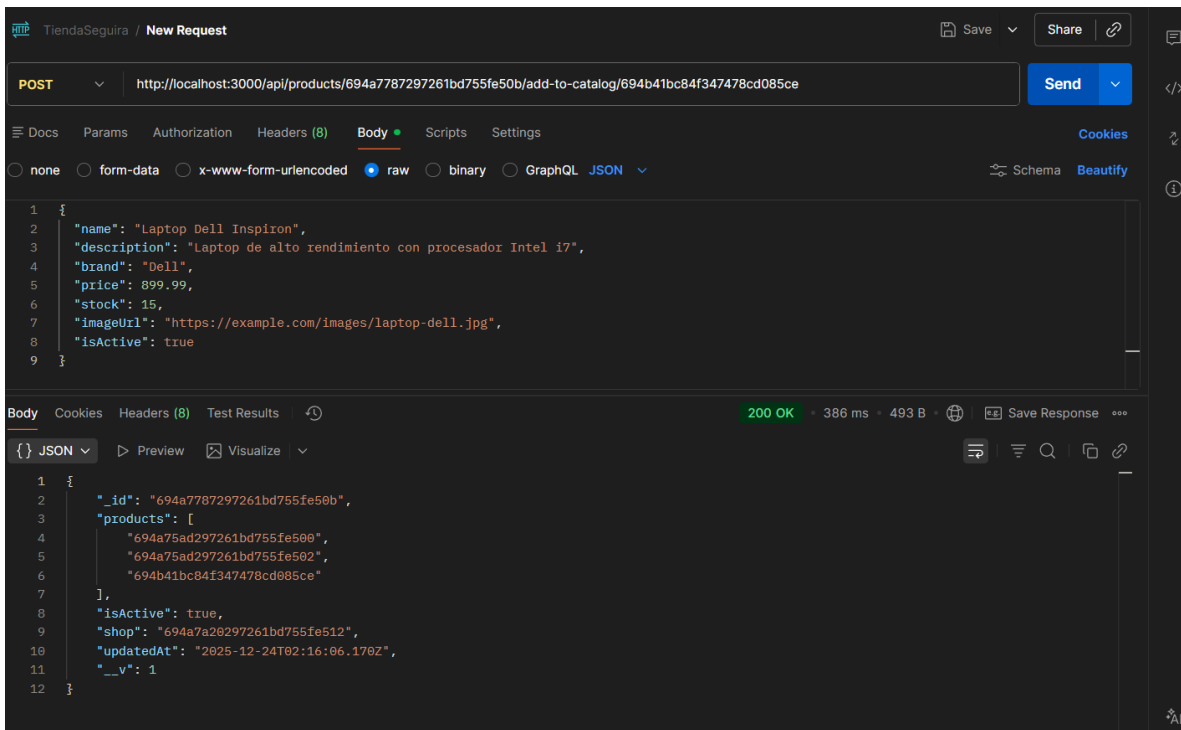
Implementación	Responsable	Estado
Crear un producto	Sebastian Holguin	✓ Hecho
Ver pedidos de los clientes	Sebastian Holguin	✓ Hecho
Buscar imagenes (scraping)	Sebastian Holguin	✓ Hecho
Agregar productos al carrito de compra	Derian Baque	✓ Hecho
Visualizar y actualizar el total del carrito de compra	Derian Baque	✓ Hecho
Consultar productos y tiendas cercanas. (lectura)	Carlos Ronquillo	✓ Hecho
Guardar historial de búsqueda y compras (lectura y escritura)	Carlos Ronquillo	✓ Hecho

10.1.1 Evidencia

Crear un Producto y agregarlos al catálogo

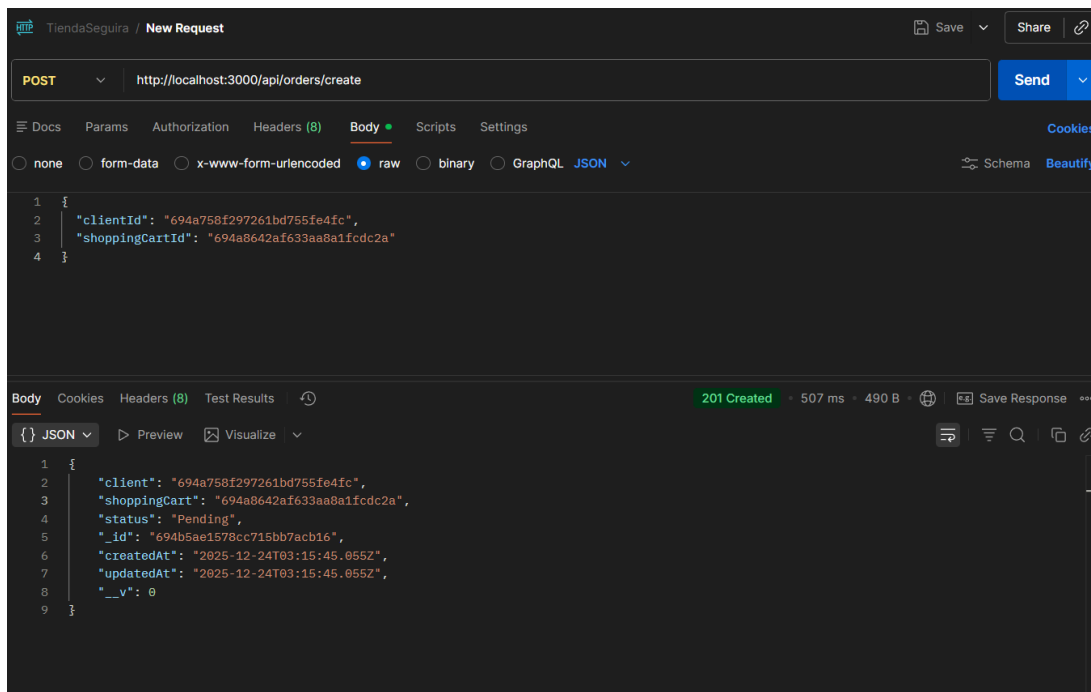


POST para crear un producto



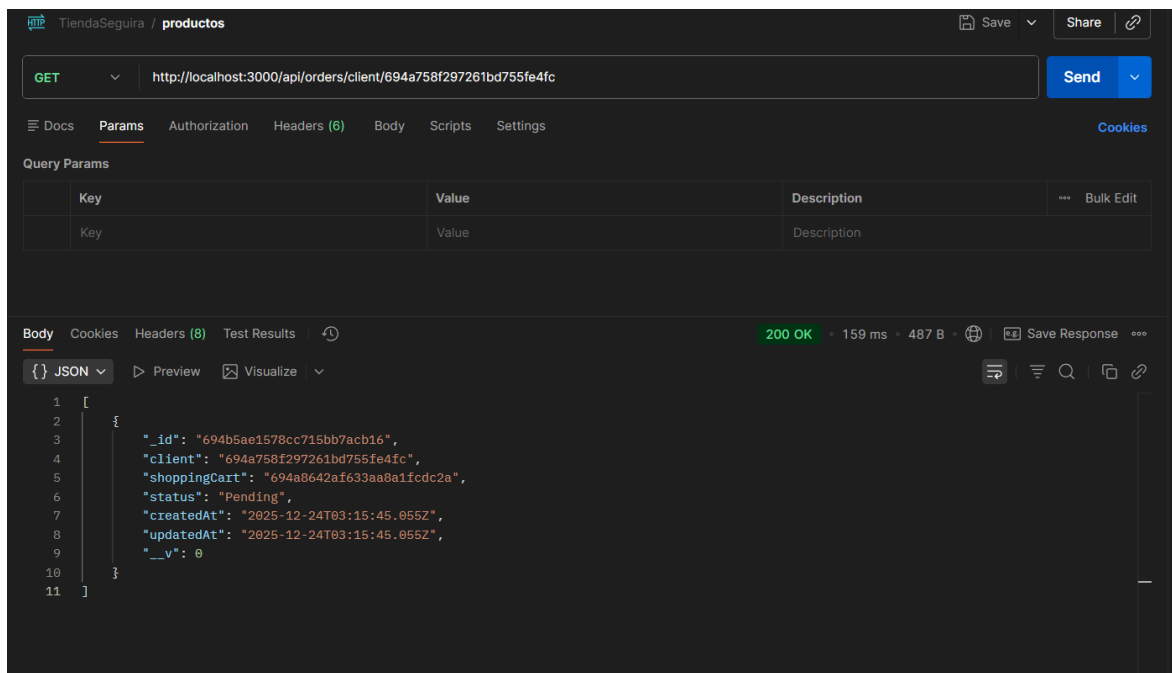
POST para agregar un producto al catálogo

Crear una orden de compra



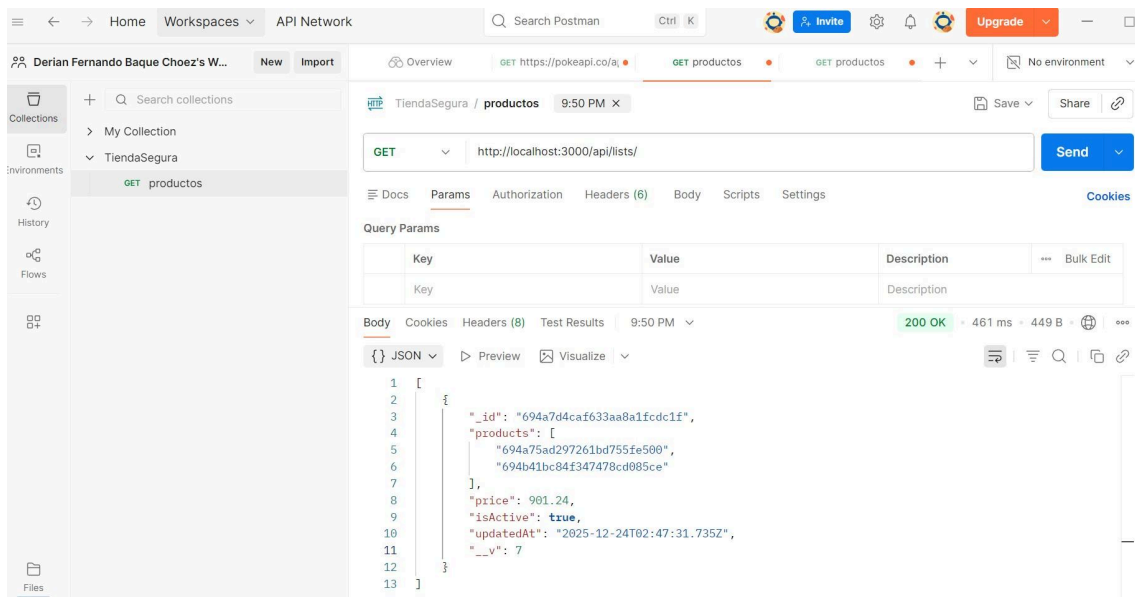
POST para agregar un orden de compra

Ver una orden de compra de un cliente

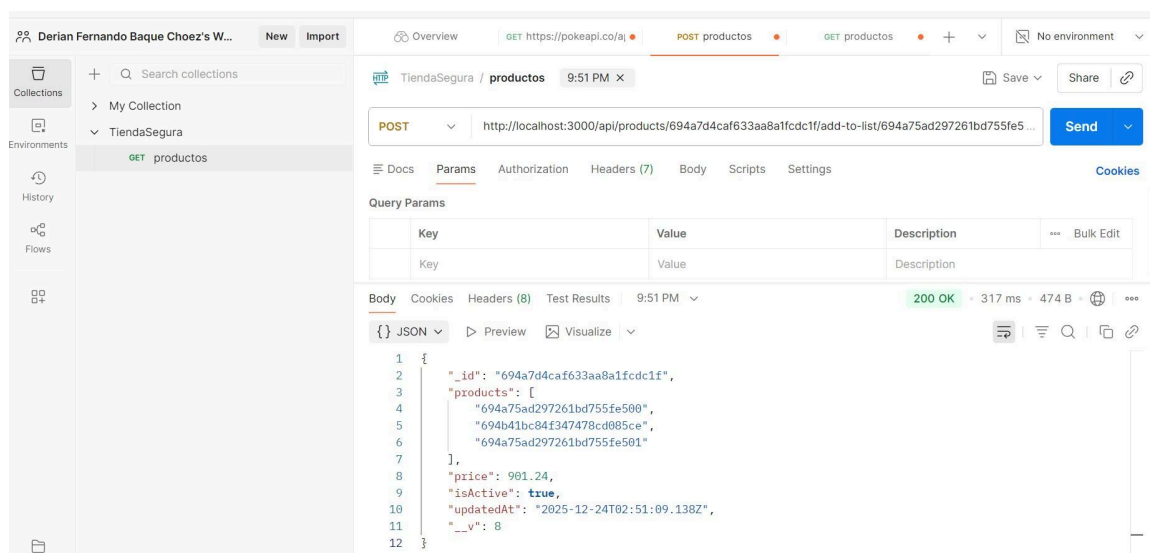


Get para consultar la orden de los clientes

Visualizar productos, agregar productos al carrito de compra y actualizar el total del carrito de compra



Get para obtener información del carrito de compras



Post para agregar un producto mediante id al carrito de compras

Postman interface showing a GET request to `https://pokeapi.co/api/lists/` (URL corrected from `api` to `lists`) returning a 200 OK status. The response body is JSON, displaying a list of products with details like `_id`, `products` array, `price`, `isActive`, `updatedAt`, and `__v`.

Query Params table:

Key	Value	Description	Bulk Edit
Key	Value	Description	

Response Body (JSON):

```
1 [
2   {
3     "_id": "694a7d4caf633aa8a1fcdc1f",
4     "products": [
5       "694a75ad297261bd755fe500",
6       "694b41bc84f347478cd085ce",
7       "694a75ad297261bd755fe581"
8     ],
9     "price": 903.34,
10    "isActive": true,
11    "updatedAt": "2025-12-24T02:53:57.125Z",
12    "__v": 8
13  }
14 ]
```

Get para obtener información actualizada del carrito de compras

```
[Vista previa] README.md JS catalogController.js X
backend > controllers > JS catalogController.js > ...
21 * buscar productos y tiendas
22 * GET /api/catalog/search?q=texto
23 */
24 async function searchCatalog(req, res, next) {
25   try {
26     const { q, clientId } = req.query;
27
28     if (!q) {
29       return res.status(400).json({
30         message: 'Debe ingresar un texto para buscar'
31       });
32     }
33
34     // GUARDAR HISTORIAL DE BÚSQUEDA
35     if (clientId) {
36       await SearchHistory.create({
37         client: clientId,
38         query: q
39       });
40     }
41
42     const regex = new RegExp(q, 'i');
43
44     const products = await Product.find({
45       $or: [
46         { name: regex },
47         { brand: regex },
48         { description: regex }
49       ]
50     });
51
52     const shops = await Shop.find({
53       name: regex
54     });
55
56     res.json({ products, shops });
57   } catch (err) {
58     next(err);
59   }
60 }
61 }
62
```

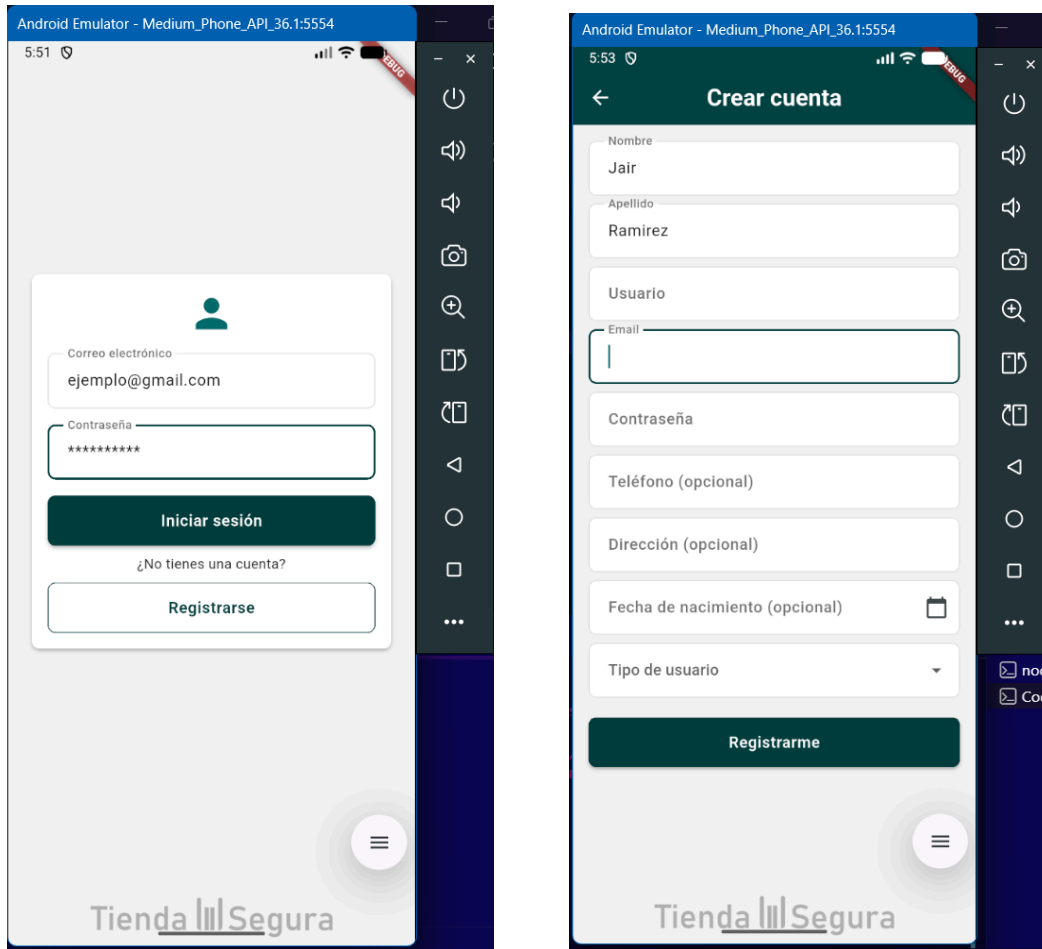
Implementacion metodo searchCatalog

10.2 FrontEnd - Visualización

Implementación	Responsable	Estado
Login	Sebastian Holguin	✓ Hecho
Registro de usuario	Sebastian Holguin	✓ Hecho
Catálogo de productos	Derian Baque	✓ Hecho
Carrito de compras	Derian Baque	✓ Hecho

Registro de productos	Carlos Ronquillo	✓ Hecho
-----------------------	------------------	---------

10.2.1 Evidencia




Login y Registro de usuario


Tienda Segura

Tiendasegura

Bebidas




Coca Cola
\$1.25
Agregar




Agua
\$0.60
Agregar

Snacks




Ruffles
\$0.75
Agregar




Salticas
\$0.50
Agregar

Tienda Segura

Tiendasegura




Coca Cola
\$1.25
Agregar




Agua
\$0.60
Agregar

Snacks







Ruffles
\$0.75
Agregar



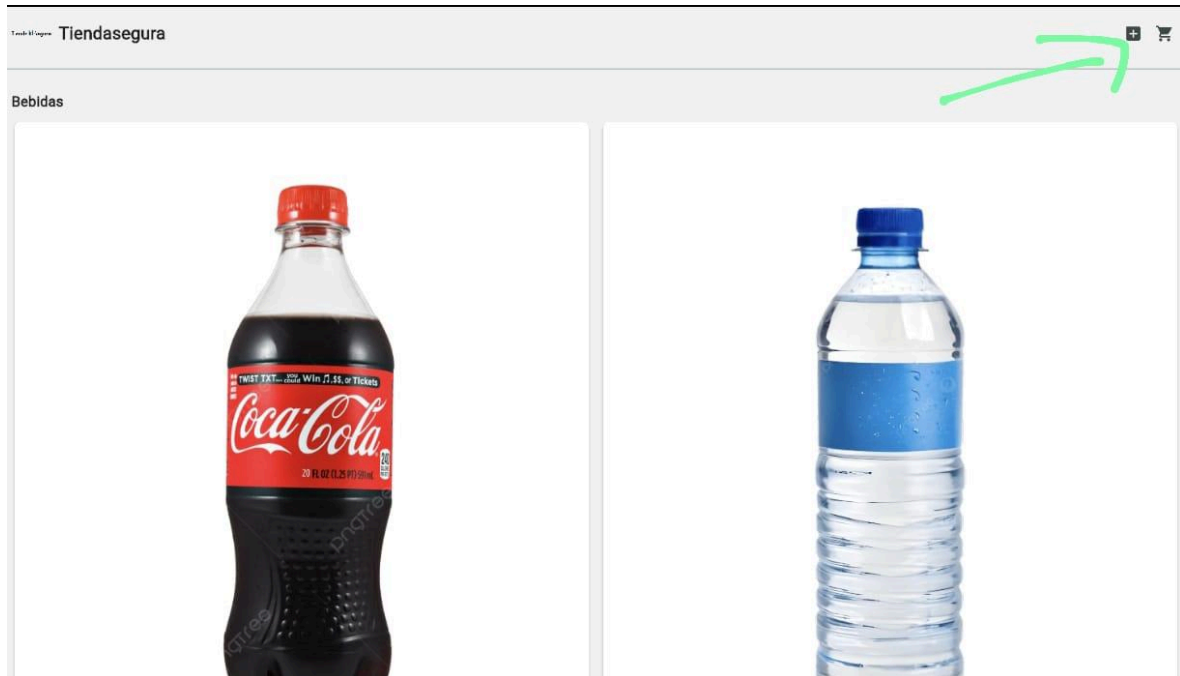
Salticas
\$0.50
Agregar

Categoría de productos

Tiendasegura			
	Agua	- 1 +	
	Cantidad: 1		
	\$0.60		
	Coca Cola	- 1 +	
	Cantidad: 1		
	\$1.25		
Total a pagar:		\$1.85	

Tiendasegura			
	Agua	- 4 +	
	Cantidad: 4		
	\$2.40		
	Coca Cola	- 2 +	
	Cantidad: 2		
	\$2.50		
Total a pagar:		\$4.90	

Carrito de Compras



← Registrar Producto

Nombre

Marca (opcional)

Precio (opcional)

Cantidad neta (opcional) Unidad

URL imagen (opcional)

Crear producto

11 Link del repositorio

<https://github.com/Sebhvarg/tiendasegura.git>

12. Referencias

Flutter Team. (2024). *Flutter documentation*. Google.
<https://docs.flutter.dev>

Dart Team. (2024). *Dart language tour*. Google.
<https://dart.dev/guides>

Node.js Foundation. (2024). *Node.js documentation*. OpenJS Foundation.
<https://nodejs.org/en/docs>

Microsoft. (2024). *TypeScript handbook*.
<https://www.typescriptlang.org/docs>

MongoDB Inc. (2024). *MongoDB documentation: NoSQL database*.
<https://www.mongodb.com/docs>

Firebase Team. (2024). *Firebase authentication documentation*. Google.
<https://firebase.google.com/docs/auth>

Google Developers. (2024). *Google Maps Platform documentation*.
<https://developers.google.com/maps>

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.

Fowler, M. (2003). *Patterns of enterprise application architecture*. Addison-Wesley.

Richards, M., & Ford, N. (2020). *Fundamentals of software architecture*. O'Reilly Media.

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill.

Statista Research Department. (2023). *Mobile operating system market share worldwide*.
<https://www.statista.com>

OWASP Foundation. (2023). *OWASP mobile top 10: Mobile security risks*.
<https://owasp.org/www-project-mobile-top-10/>