



GOBIERNO DE LA CIUDAD DE BUENOS AIRES

Ministerio de Educación

Dirección de Formación Técnico Superior

Instituto de Formación Técnico Superior N° 18

Mansilla 3643, (C1425BBW) Capital Federal

17-11-2025

Práctica Profesionalizante III: Diseño y Arquitectura del Sistema

INTEGRANTES: Alejo Elalle, Carolina Galarza Payaslian,
Alejandro Rodriguez

Profesora: Patricia Litovicius



1. Descripción del Proyecto	2
1.1 Contexto de la situación y problema detectado	2
1.2 Usuarios detectados	2
1.3 Análisis FODA	2
2. Objetivos	3
2.1 Qué necesidad resuelve el sistema	3
2.2 Metas concretas	3
3. Alcance	3
3.1 Qué sí resolverá el sistema	3
3.2 Qué no resolverá el sistema	3
4. Análisis de Usuarios y Experiencia de Usuario (UX)	3
4.1 Perfiles de Usuario	3
4.2 Creación de personas y escenarios de uso	4
4.2.1 User Persona	4
4.2.2 Escenarios de Caso de Uso	5-6
4.3 Wireframes	7
5. Casos de Uso (UML)	8
6. Especificación de Requerimientos	9
6.1 Requerimientos Funcionales	9
6.2 Requerimientos No Funcionales	9
6.3 Requisitos de Arquitectura	10
6.3.1 Modelo Cliente-Servidor	10
6.3.2 Servicios Externos	10
7. Recursos Disponibles	10
7.1 Personal y roles asignados	10
7.2 Materiales	11-12
8. Plan de Trabajo	13
8.1 Diagrama de Gantt	13
9. Gestión de Riesgo	13-14
10. Diseño del Sistema	15
10.1 Arquitectura Cliente-Servidor	15
10.1.1 Justificación	15
10.1.2 Servicios Externos	16-17
10.2 Diagramas UML	17
10.2.1 Diagrama de Despliegue	17
10.2.2 Diagrama de Clases	18
10.2.3 Diagrama de Secuencia	18
10.2.4 Diagrama de Actividad	19
11. Conclusiones Finales	20



1. Descripción del Proyecto

1.1 Contexto de la situación y problema detectado:

Las sucursales de la pizzería en general son pequeñas, pero en cantidad, distribuidas por toda la Ciudad de Buenos Aires.

Sus recursos económicos son limitados y no muy extensos. Los materiales que utilizan para hacer sus productos gastronómicos suelen ser exactamente los mismos para todas sus sucursales, donde se suele encontrar más variabilidad es en la cantidad de empleados y sus cargos/tareas.

Las pizzerías tienen problemas para gestionar su stock de inventario, se han presentado casos donde sus ingredientes pasan a estar en estado de caducidad antes de poder venderlo. Estos alimentos, además de su costo monetario, se desperdician completamente, se vuelve a pedir stock de alimentos que ya hay en cantidad o no se detectan a tiempo que productos llegaron con fechas cortas de vencimiento.

1.2 Usuarios detectados:

Los principales son los encargados de tienda, quienes interactúan con el inventario y/o la carga de productos, mientras que los empleados solo están autorizados a cargar.

1.3 Análisis FODA:

- **Fortalezas:** ¿Qué recursos o ventajas internas tiene su propuesta?
 - Equipo pequeño = Bajo costos operativos.
 - Experiencia en Diseño y Base de Datos.
 - Servicio de Soporte incluido.
 - Personalizable según el cliente.
 - Aplicación simple y ágil.
 - Accesible a cualquier dispositivo móvil.
- **Oportunidades:** ¿Qué ofrece el entorno para el éxito del sistema?
 - Falta de opciones en el mercado.
 - Tendencia a la digitalización en entornos gastronómicos.
 - Aumento de costos de alimentos.
- **Debilidades:** ¿Qué limitaciones o dificultades internas enfrenta su proyecto?
 - Equipo pequeño = tiempos de desarrollo más largos.
 - Falta de experiencia en UI/UX.
- **Amenazas:** ¿Qué riesgos externos podrían afectar al sistema?
 - Recesión económica, dificultad para invertir.
 - Resistencia al cambio.



2. Objetivos

2.1 Qué necesidad resuelve el sistema.

Las pizzerías tienen problemas para gestionar su stock de inventario, se han presentado casos donde sus ingredientes pasan a estar en estado de caducidad antes de poder venderlo. Estos alimentos, además de su costo monetario, se desperdician completamente, se vuelve a pedir stock de alimentos que ya hay en cantidad o no se detectan a tiempo que productos llegaron con fechas cortas de vencimiento.

2.2 Metas concretas:

El objetivo de nuestro proyecto es reducir la pérdida de recursos gastronómicos por caducidades y un mayor control de las existencias de los mismos.

3. Alcance

3.1 Que *sí* resolverá el sistema:

- Gestión de caducidad de productos.
- Gestión de inventario de productos.
- Carga de Productos.
- Descontar stock con cada venta.
- Identificar de forma rápida y precisa qué productos están ingresando con fechas cortas de caducidad, para ser denunciados y reemplazados cuanto antes.

3.2 Que *no* resolverá el sistema:

- Productos perdidos por robo.
- Contabilidad de productos dañados por mala manipulación.
- Al vender una pizza, se estima que se use una cierta cantidad/peso de ingredientes que no siempre va a ser exacto a lo establecido por el sistema.
- Caducidades ocultas por falta de rotación física.

4. Análisis de Usuarios y Experiencia de Usuario (UX)

4.1 Perfiles de Usuario:

Los principales son los encargados de tienda, quienes interactúan con el inventario y/o la carga de productos, mientras que los empleados solo están autorizados a cargar.



4.2 Creación de personas y escenarios de uso:

4.2.1 User Persona:

Nombre	Dardo Fuseneco
Edad	38 años
Ocupación	Encargado de Sucursal
Objetivos	<ul style="list-style-type: none">- Evitar pérdidas por caducidad- Mantener stock siempre actualizado- Mejorar comunicación con el equipo
Necesidades	<ul style="list-style-type: none">- Una visión clara de su stock- Avisos de productos por caducar- Acceso rápido
Frustraciones	<ul style="list-style-type: none">- Lleva todo seguimiento con papel y lápiz- Se entera tarde de productos caducados- Problemas de comunicación con sus empleados sobre el stock
Contexto de uso	<p>El sistema estaría siempre disponible en su celular, ya sea para contar mercadería en el almacén, como en la oficina con el remito de una entrega.</p> <p>Por otra parte siempre estaría recibiendo alertas de productos que están cerca de vencer, pudiendo configurar su frecuencia según el producto.</p> <p>Cada interacción es corta, busca la información que necesita consultar y vuelve a lo que estaba haciendo.</p>
Nombre	Pablo Lamponne
Edad	35
Ocupación	Empleado
Objetivos	<ul style="list-style-type: none">- Cargar los productos recién llegados lo mas rapido posible- Evitar errores al cargar- Evitar pérdidas por caducidad
Necesidades	<ul style="list-style-type: none">- Velocidad y precisión a la hora de escanear productos- Autocompletar el formulario de carga con cada escaneo exitoso- Avisos de productos por caducar
Frustraciones	<ul style="list-style-type: none">- La carga es lenta y tediosa, haciéndose a papel y lápiz primero varias veces para evitar errores y pasando por distintas manos para archivar- Se entera tarde de productos caducados
Contexto de uso	<p>El sistema siempre estaría listo en su celular para cuando llegue nueva mercadería que cargar a la base de datos, de normal ocurriendo en el almacén de la sucursal en la entrada de la misma. Usando el escáner pasa de un producto al otro solo deteniéndose para aclarar la cantidad/peso y la fecha de vencimiento de los mismos.</p> <p>Por otra parte siempre estaría recibiendo alertas de productos que están cerca de vencer.</p> <p>Cada interacción dura dependiendo de la cantidad de la mercadería a cargar pero más allá de recibir alertas de caducidad no debe usarlo para nada más.</p>



4.2.2 Escenarios de Caso de Uso:

Nombre: Carga por Escáner

Actores: Encargado y Empleado

Precondiciones:

- El actor debe estar logueado en el sistema.
- Que la app tenga acceso a la cámara del dispositivo

Flujo principal:

1. El actor ingresa a la app pasando por el proceso de Login.
2. Desde el menú principal ingresa a “Cargar productos”.
3. Toca la función de “Escanear”.
4. Se abre la cámara con una interfaz traslúcida que sirve de guía para apuntar al código correctamente.
5. Detecta el código y revisa que coincida con la base de datos.
6. Una vez encontrado el producto, se sube la información al formulario de carga y se registra.
7. El usuario solo ingresa la cantidad/peso y la fecha de vencimiento.
8. Se termina el procedimiento al darle a “Finalizar”.

Flujo alternativo:

- 4. El usuario niega el permiso de usar la cámara del dispositivo.
 - Se debe buscar en la configuración del dispositivo para habilitar la cámara de forma manual.
 - 4. El dispositivo del usuario no tiene cámara.
 - Se debe utilizar la carga manual.
 - 5. La aplicación no detecta el código de barras en su base de datos.
 - Se procede con carga manual.
 - Se solicita a un administrador cargar la información correspondiente a la base de datos.
-

Nombre: Ver/Interactuar con Reportes

Actores: Encargado y Empleado

Precondiciones:

- El actor debe estar logueado en el sistema.
- Deben existir reportes previamente generados



Flujo principal:

1. El actor ingresa a la app pasando por el proceso de Login.
 2. El actor selecciona la opción “Ver reportes de caducidades” desde el menú principal.
 3. El sistema muestra una lista de reportes con los productos próximos a vencer y su estado.
 4. El actor puede:
 - Filtrar los reportes por fecha y/o tipo de reporte.
 - Marcar un reporte como visto.
 - Repetir las acciones anteriores todas las veces que necesite.
 5. Se termina el procedimiento al salir de la sección de Reportes.
-

Nombre: Configurar Reportes de Caducidades

Actor: Encargado

Precondiciones:

- El actor debe estar logueado en el sistema con su cuenta exclusiva de Encargado.
- Debe existir una configuración de reportes activa (ya sea por defecto o previamente definida).

Flujo principal:

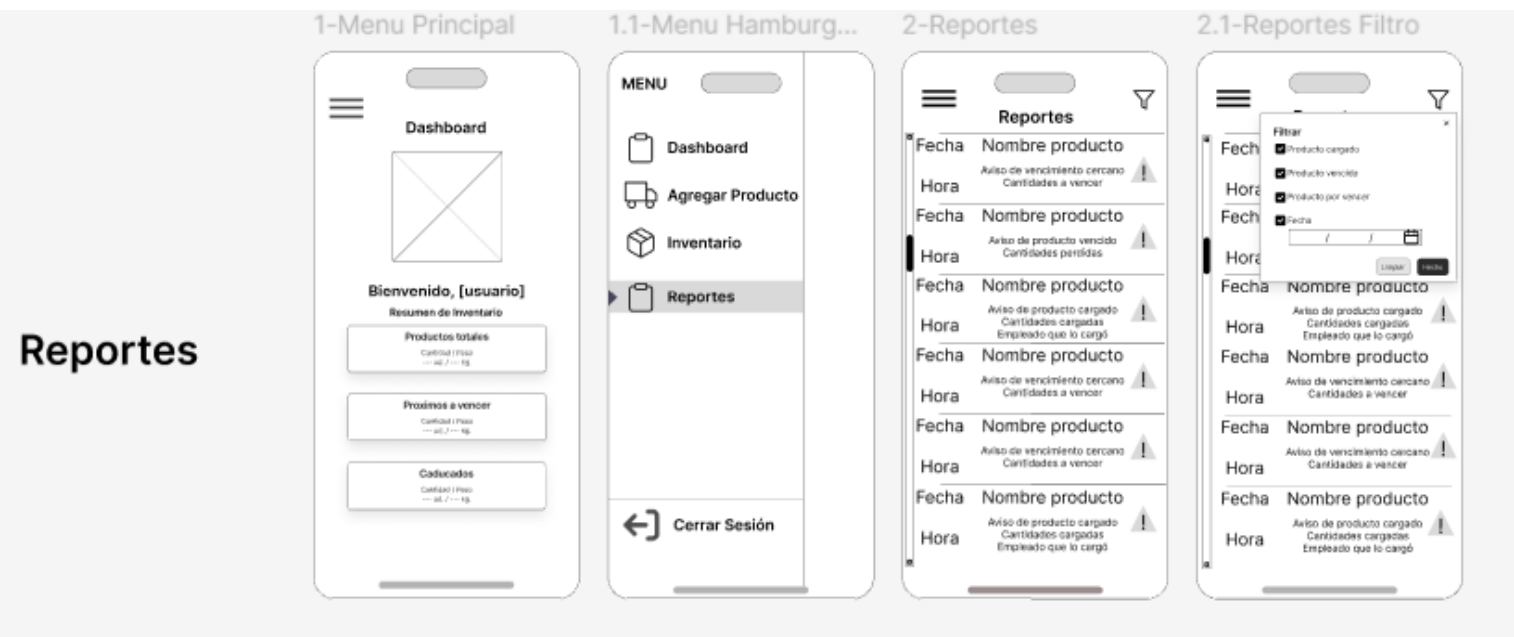
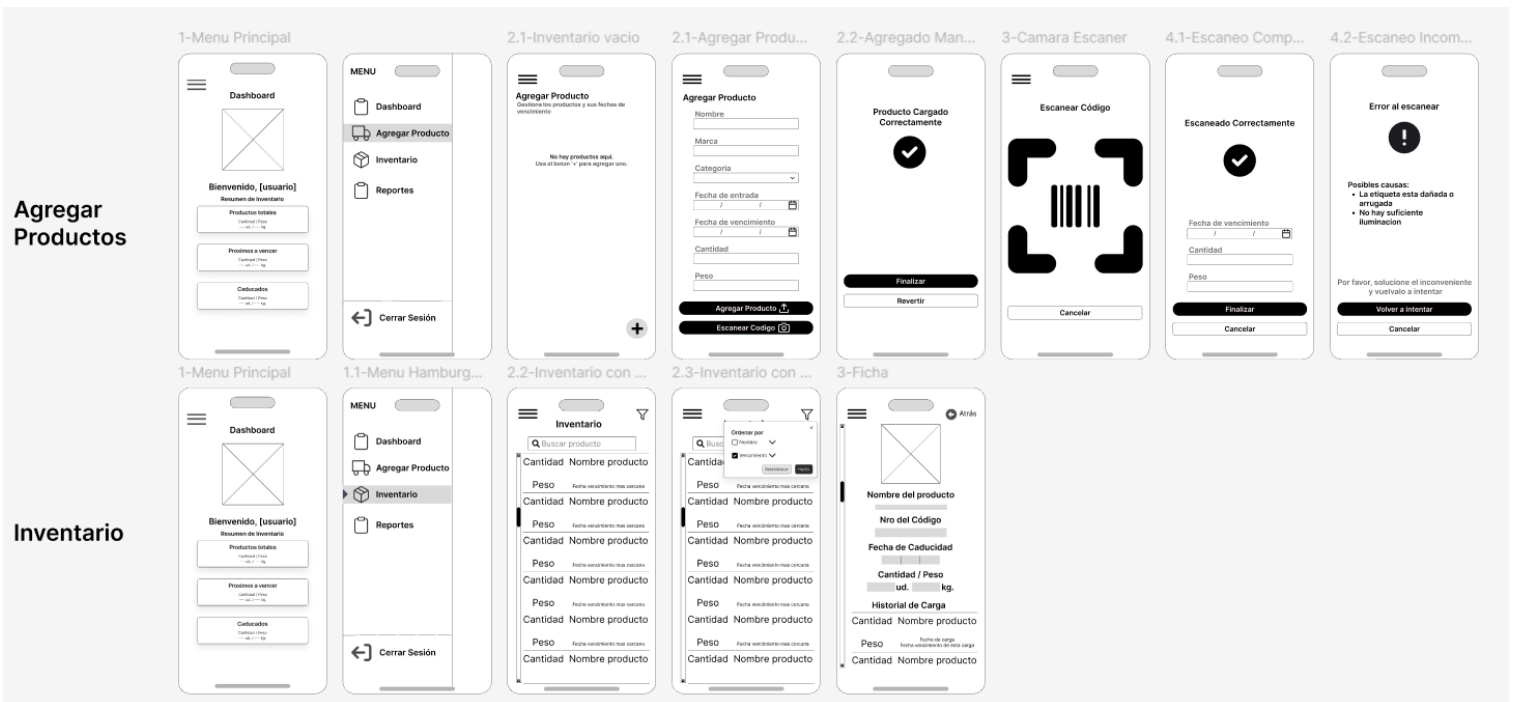
1. El Encargado ingresa a la app pasando por el proceso de Login.
2. Desde el menú principal, selecciona la opción “Configuración del Sistema”.
3. Dentro del menú Configuración del Sistema, seleccionar la opción “Modificar Reportes”
4. El sistema muestra los valores actuales de configuración.
5. El Encargado modifica los parámetros deseados:
 - Días previos a la fecha de caducidad para emitir alerta.
 - Categorías o productos específicos a incluir.
 - Medio de notificación (correo, notificación push-up, etc.).
6. El Encargado confirma los cambios.
7. El sistema guarda la nueva configuración y actualiza los reportes automáticos según los parámetros establecidos.
8. Se termina el procedimiento mostrando un mensaje de confirmación.

Flujos alternativos:

- Si el Encargado selecciona “Cancelar” antes de confirmar, el sistema descarta los cambios.
- Se termina el procedimiento sin modificar la configuración existente.



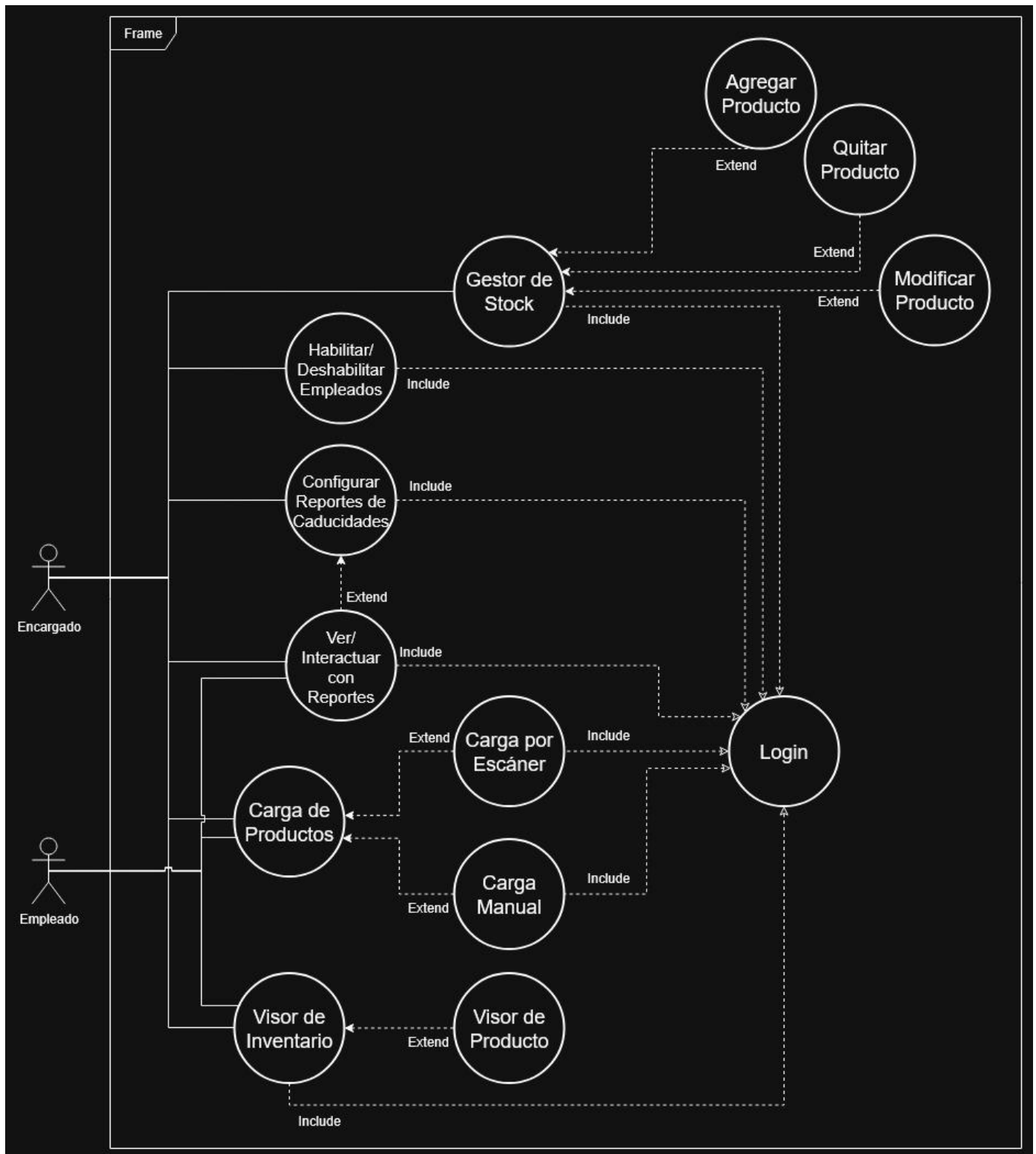
4.3 Wireframes:



Link de Figma [aquí](#).



5.Caso de Uso (UML)



Link de Draw.io [aquí](#).



6. Especificación de Requerimientos

6.1 Requerimientos Funcionales:

RF1: El usuario debe de poder ingresar a la cámara de su dispositivo.

RF2: Al apuntar con la cámara a un código de barras legible debe de identificar el producto.

RF3: Una vez identificado el producto el formulario de carga se debe autocompletar con sus respectivos datos.

RF4: Para identificar dichos productos el sistema debe estar conectado a una base de datos en la nube donde recuperar y comparar información.

RF5: El sistema debe contar con un Login de usuario y contraseña, para identificar administradores de empleados comunes.

RF6: Un administrador puede modificar el inventario en cuanto a cantidades y fechas de vencimiento.

RF7: El administrador puede habilitar y deshabilitar cuentas de empleados.

RF8: Todos los usuarios deben recibir alertas de productos a vencer, vencidos y nuevas cargas al inventario aclarando quien las hizo.

RF9: El sistema debe de poder visualizar qué productos hay cargados, su cantidad en unidades y/o peso y fechas de vencimiento.

RF10: Al visualizar un producto en específico se debe de poder ver, además de su información descriptiva, un historial de las cargas del último mes.

RF11: Por cada venta registrada por el cajero el sistema debe descontar las cantidades de ingredientes del inventario que se utilizaron.

RF12: Si un usuario necesita restablecer su contraseña, va a recibir un código en su email registrado que lo habilita.

6.2 Requerimientos no funcionales:

RNF1: El código de barras en condiciones óptimas de legibilidad e iluminación, la cámara debe escanear su información y transferirla en el formulario en al menos 1 segundo.

RNF2: La aplicación debe de funcionar en cualquier dispositivo móvil ya sea Android o iOS.

RNF3: Un usuario empleado no puede modificar el stock bajo ningún concepto solo el encargado/administrador tendrá ese poder.

RNF4: Se trackea de forma constante toda actividad de todos los usuarios.

RNF5: El email con el que se creó la cuenta es el único habilitado para recibir el código de restablecimiento de contraseña.



6.3 Requisitos de Arquitectura:

6.3.1 Modelo Cliente-Servidor:

Cliente (Aplicación móvil)

- Interfaz principal utilizada por los usuarios.
- Envía solicitudes al backend, recibe respuestas y muestra la información.
- Realiza autenticación mediante Firebase y escaneo de códigos con MLKit.

Servidor (Backend en Node.js)

- Expone la API REST que recibe y procesa las solicitudes del cliente.
- Aplica lógica del negocio: inventario, movimientos, vencimientos, roles, etc.
- Gestiona la comunicación con la base de datos y valida tokens de Firebase.

Base de datos (MySQL)

- Almacena productos, usuarios, movimientos, inventario y registros del sistema.
- Responde a consultas y actualizaciones solicitadas por el servidor.

6.3.2 Posibles servicios externos:

- Firebase Cloud Messaging (FCM) - Envío de notificaciones push sobre vencimientos, nuevos registros y alertas.
- Firebase ML Kit - Procesamiento de cámara y lectura de códigos de barras.
- Firebase Storage - Almacenamiento de imágenes de productos.
- Servicios de nube / hosting - Google Cloud VM para alojar el backend.

7. Recursos disponibles

7.1 Personal y roles asignados:

Alejo Elalle: Analista, Diseñador, Programador, Tester

Carolina Galarza: Analista, Diseñadora, Tester, Coordinadora

Alejandro Rodríguez: Analista, Programador, Tester



7.2 Materiales:

Hardware requerido

Para Desarrolladores

PC Windows

CPU: Intel i5 / Ryzen 5 o superior

RAM: 8–16 GB

Smartphones de prueba (Android obligatorio; iOS opcional)

Emuladores Android (Android Studio)

Para producción

Servidor donde correr el backend.

Servidor o instancia dedicada para la base de datos.

Para usuarios finales

Smartphone Android con:

Cámara funcional

Memoria disponible

Acceso a internet

Software requerido

Frontend

React Native

Bibliotecas claves

@react-native-firebase/auth - autenticación

@react-native-firebase/messaging - notificaciones push

react-native-vision-camera + MLKit - escaneo de códigos

Almacenamiento local

AsyncStorage / SecureStore para guardar UID del usuario o token

Backend

Node.js

Bibliotecas clave

Express

Sequelize o prisma (ORM)

jsonwebtoken para consumir tokens generados por Firebase

bcryptjs para seguridad

cors

dotenv

Servidor dedicado

Google Cloud VM



Base de datos

MySQL

Herramientas para la BD

MySQL Workbench

Servicios Externos

Firebase Authentication

Firebase Cloud Messaging (FCM)

Firebase ML Kit

Firebase Storage

IDEs

VS Code

Android Studio



8. Plan de Trabajo

8.1 Diagrama de Gantt:



Link de Gantt [aquí](#).

9. Gestión de Riesgo

Riesgo	Probabilidad	Impacto	Acción Preventiva/Plan de Mitigación
Fallo en acceso a la cámara del dispositivo	Baja	Media	Probar compatibilidad con cámaras de Android/iOS, pedir permisos explícitos, fallback de reintento.
La cámara no puede leer códigos de barras por mala calidad o iluminación	Media	Media	Guía visual al usuario, algoritmo de lectura robusto, control de enfoque, mensajes de “volver a intentar”.
Identificación incorrecta o fallida del producto	Baja	Media	Validar formato del código, implementar verificación contra múltiples fuentes, logs de fallos.
Autocompletado del formulario falla o trae datos incorrectos	Baja	Baja	Validación cruzada de datos recuperados.
Caída o inaccesibilidad de la base de datos en la nube	Baja	Alta	Redundancia, réplicas en la nube, monitoreo constante, alertas de downtime.

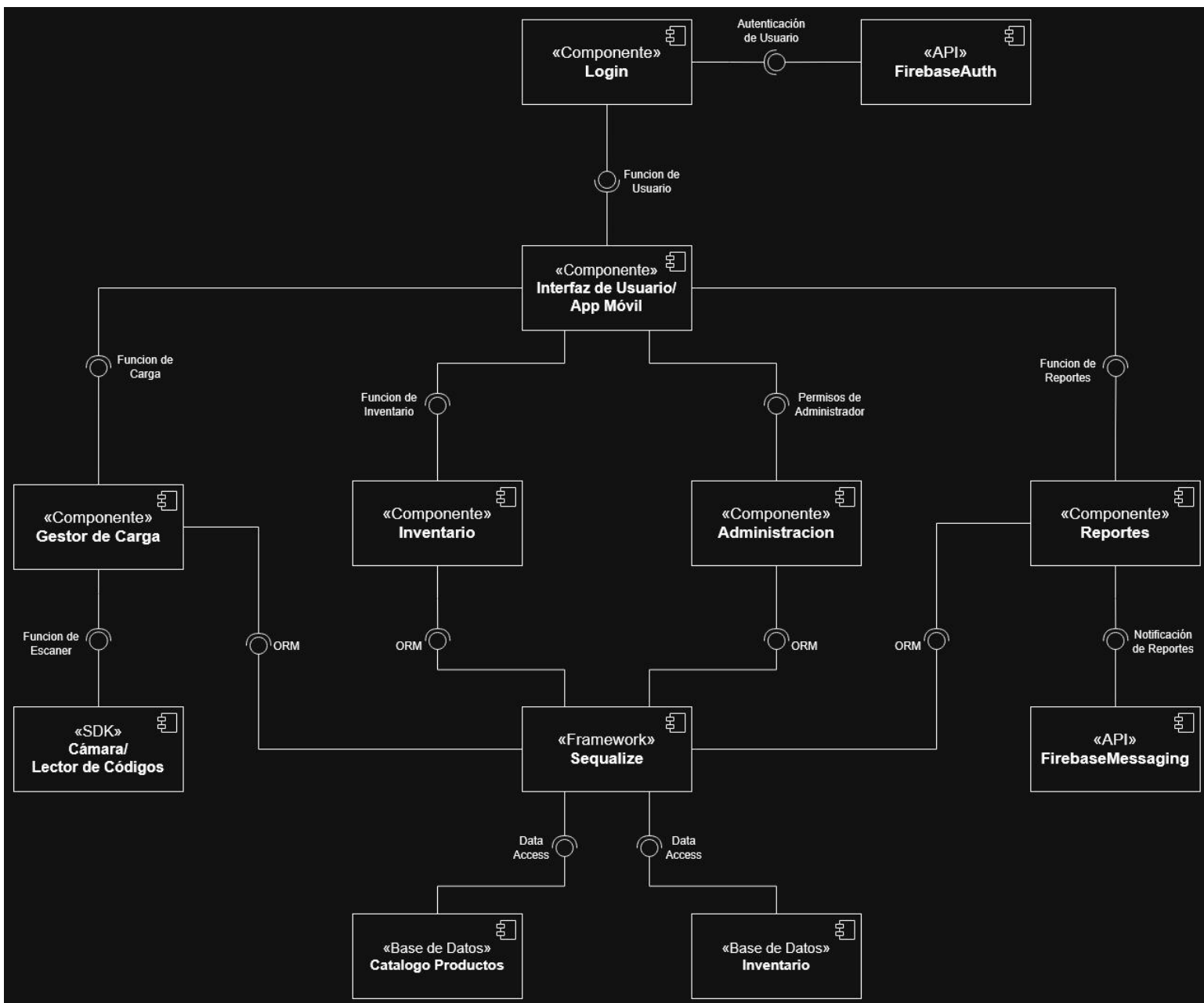


<i>Problemas de autenticación/login</i>	Baja	Alta	Implementar autenticación segura, reintentos limitados, auditoría de accesos.
<i>Empleado sin permisos modifica inventario</i>	Baja	Alta	Control estricto de roles, pruebas de penetración, política RBAC correctamente aplicada.
<i>Fallo al habilitar/deshabilitar cuentas</i>	Media	Media	Validación previa de la acción, historial de cambios, doble confirmación para admins.
<i>No se envían alertas a los usuarios</i>	Media	Baja	Pruebas de envío periódico, sistema redundante de notificaciones.
<i>Visualización incorrecta de inventario</i>	Baja	Alta	Validar datos antes de mostrarlos, pruebas funcionales continuas, caché controlado.
<i>Historial de cargas no registrado o inconsistente</i>	Media	Alta	Auditoría obligatoria, timestamps automáticos y no modificables.
<i>Fallo al descontar ingredientes por ventas</i>	Media	Alta	Integración fuerte con sistema de ventas, transacciones atómicas, pruebas unitarias.
<i>Restablecimiento de contraseña fallido</i>	Baja	Baja	Asegurar servidor de correo, validación del email registrado, monitorear spam/bounces.
<i>Código de restablecimiento enviado al email equivocado</i>	Baja	Alta	Bloqueo absoluto de cambio de email sin verificación, cifrado y logs.
<i>Desempeño lento en dispositivo móvil</i>	Media	Media	Optimización de recursos, pruebas en múltiples dispositivos y versiones de SO.
<i>Actividades de usuarios no registradas correctamente</i>	Baja	Alta	Implementar auditoría continua, backups de logs, política de retención.
<i>Fuga de datos por mala configuración de roles o permisos</i>	Baja	Alta	Revisión periódica de permisos, pruebas de seguridad, cifrado en tránsito y reposo.
<i>Fallas de seguridad al almacenar productos, ventas o usuarios en la nube</i>	Baja	Alta	Hardening del servidor, firewall, monitoreo y cumplimiento de estándares.



10. Diseño del Sistema

10.1.1 Arquitectura Cliente-Servidor:



Link de Draw.io [aquí](#).



10.1.2 Justificación

El diagrama de despliegue representa la infraestructura completa del sistema, mostrando los distintos nodos físicos y lógicos involucrados en la ejecución de la aplicación, así como los artefactos que se ejecutan en cada uno de ellos y las comunicaciones necesarias para que el sistema funcione de forma integrada.

<Estación de Trabajo del Cliente>

Este nodo representa el dispositivo físico del usuario final, desde el cual se accede y ejecuta la aplicación.

<Entorno de Ejecución>

Especifica que el Frontend está desarrollado en React Native, indicando el entorno tecnológico donde se ejecuta la interfaz de usuario.

<Server de la Aplicación>

Este nodo representa el servidor en la nube donde se ejecuta la lógica central del sistema.

<Entorno de Ejecución>

Aloja los diferentes artefactos de tipo REST API, cada uno correspondiente a un módulo funcional:

- **Gestor de Carga:** Administra las operaciones de carga de productos y registro de información.
- **Inventario:** Gestiona el acceso y actualización del inventario en tiempo real.
- **Reportes:** Genera y provee informes del estado del sistema.
- **Administración:** Controla las funciones de gestión de usuarios y permisos.
- **Login:** Autentica a los usuarios mediante credenciales.

<Servicios Externos>

Este nodo agrupa los servicios externos utilizados por el sistema:

- **FirestoreAuth:** Provee las capacidades de autenticación y verificación de identidad.
- **FirestoreMessaging:** Permite el envío de notificaciones push a los usuarios.
- **FirestoreStorage:** Almacena archivos e imágenes que pueden ser utilizados por el sistema.

<Base de Datos de Sistemas>

Este nodo representa el servidor que aloja la base de datos relacional utilizada por la aplicación. Contiene dos artefactos principales:

- **Inventario:** almacena las existencias, cantidades y fechas de vencimiento.
- **Catálogo de Productos:** contiene la información general de los productos, como nombre, marca e imagen descriptiva.



El flujo de interacción entre los nodos sigue una arquitectura cliente-servidor complementada con servicios externos:

El usuario interactúa con la aplicación, que se ejecuta en el nodo Estación de Trabajo del Cliente.

La aplicación envía solicitudes a través de HTTPS hacia el Entorno de Ejecución (Node.js).

El sistema recibe las solicitudes y las procesa mediante los distintos módulos REST API especializados.

Cada API se encarga de validar datos, aplicar reglas de negocio y coordinar la operación solicitada.

Cuando se requiere acceso a datos persistentes, el backend utiliza Sequelize ORM, que traduce las operaciones de alto nivel a consultas SQL y las ejecuta sobre el nodo Base de Datos del Sistema (MySQL).

Para funciones adicionales como autenticación, envío de notificaciones o acceso a archivos, el backend se comunica mediante protocolos HTTP/REST con los Servicios Externos (Firebase), delegando en ellos tareas específicas.

Finalmente, el servidor devuelve los resultados en formato JSON a la aplicación móvil, que actualiza la interfaz y muestra la información al usuario.

10.2 Diagramas UML:

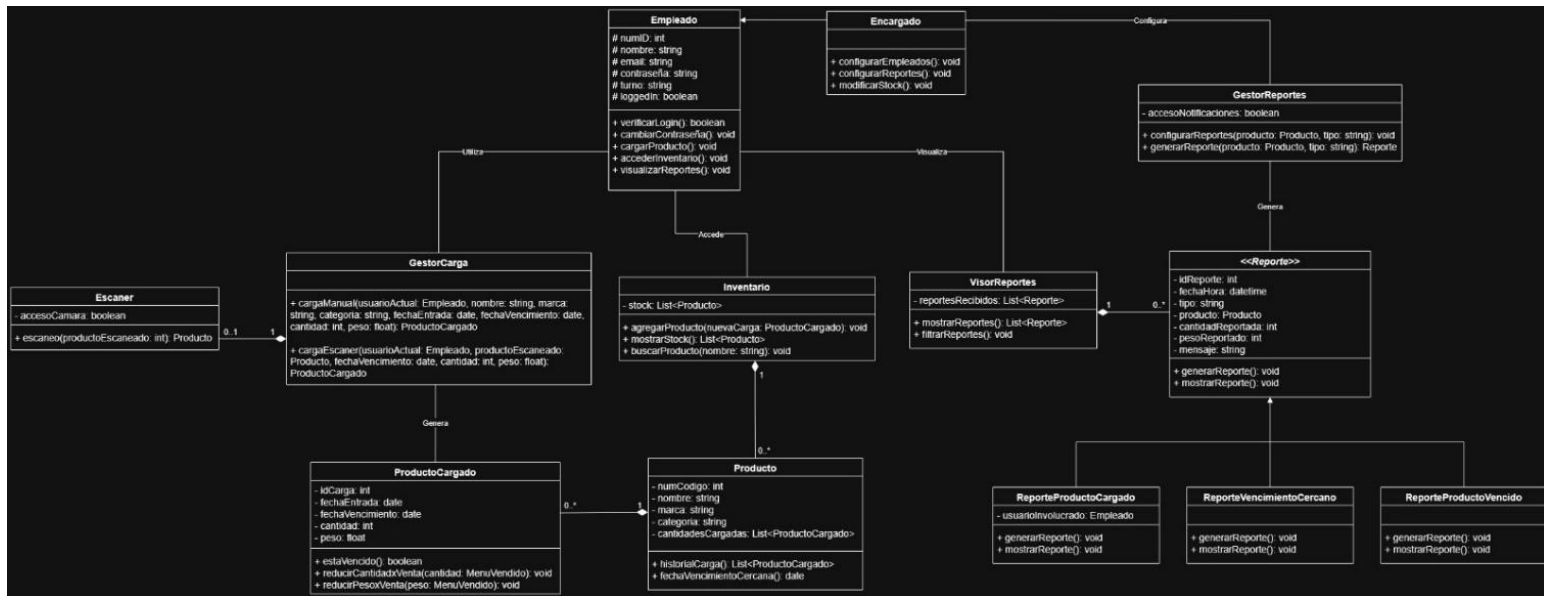
10.2.1 Despliegue:



Link de Draw.io [aquí](#).

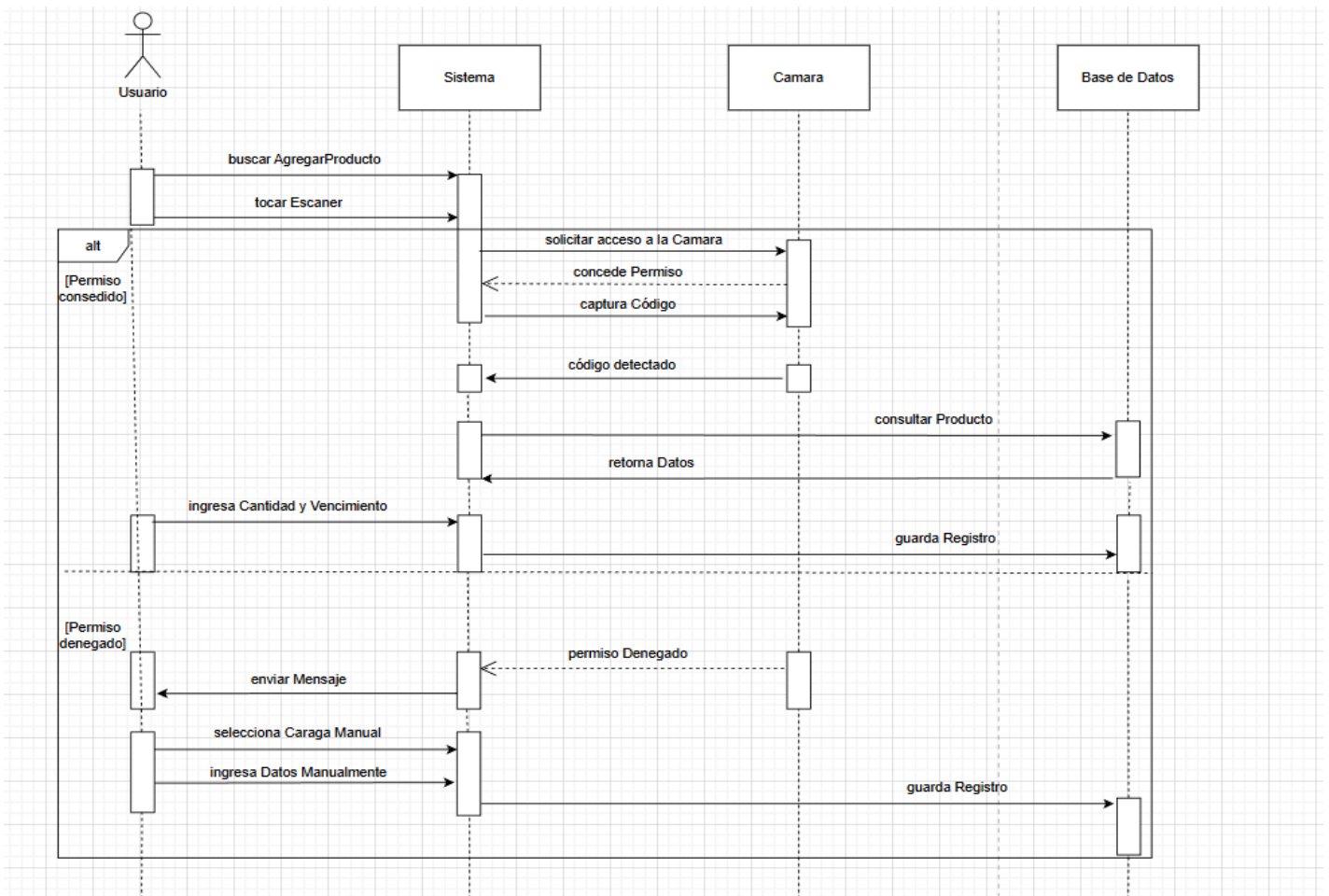


10.2.2 Clases:



Link de Draw.io [aquí](#).

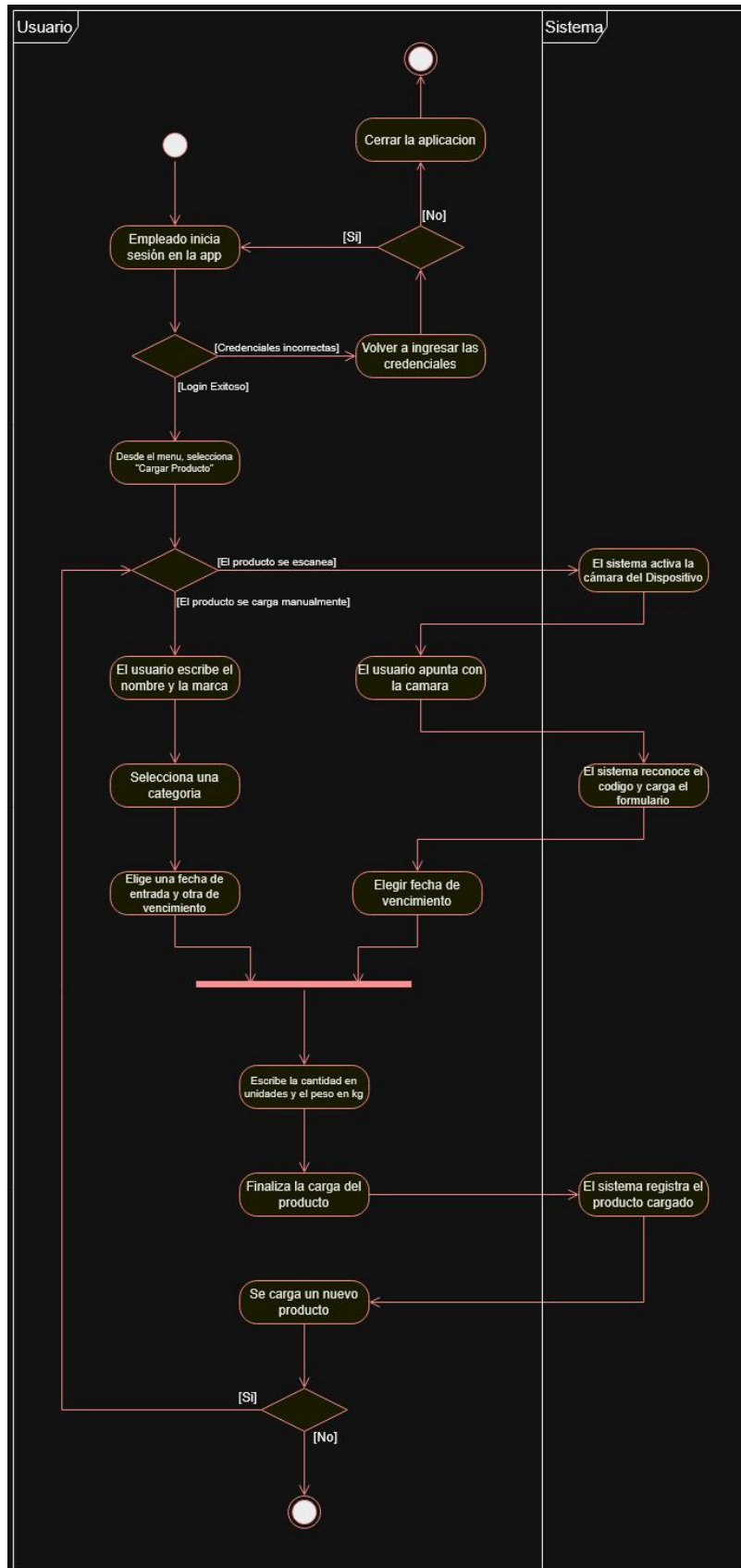
10.2.3 Secuencia:



Link de Draw.io [aquí](#).



10.2.4 Actividad:



Link de Draw.io [aquí](#).



11. Conclusiones Finales

A lo largo del desarrollo del proyecto, el equipo logró incorporar nuevos conocimientos tanto técnicos como metodológicos. El proceso permitió afianzar prácticas de trabajo colaborativo, organización y división de tareas, así como también la aplicación concreta de conceptos vistos durante la cursada, tales como arquitectura y diagramación de sistemas, modelado y gestión de requerimientos.

En cuanto al sistema implementado, se identificaron diversas fortalezas. Entre ellas, la integración de tecnologías modernas como React Native, Firebase y un backend propio en Node.js, lo que otorga una estructura escalable y adaptable a futuras necesidades. Además, el uso de Firebase para autenticación, notificaciones y procesamiento de cámara simplifican tareas complejas y mejora la experiencia del usuario final. Sin embargo, también se reconocieron debilidades y oportunidades de mejora. El sistema puede optimizarse en rendimiento, seguridad y validación de datos. A futuro sería valioso incorporar pruebas automatizadas y mejorar la interfaz de usuario. También podría evaluarse implementar un panel web administrativo para un control más completo del inventario.