Instituto Politécnico Nacional (IPN)

Unidad Profesional Interdisciplinaria de Ingeniería

y Ciencias Sociales y Administrativas – UPIICSA

– Ingeniería en Informática –

Programación Móvil ·

Martínez Vázquez Gustavo

6NM60

Manual Técnico:

"Automatización de la papelería Catalunya"

Equipo 5:

- Berlanga Nepita Santiago
- Chávez Sánchez José Enrique
- Cruz Aguilar Omar
- Martínez Aburto David
- Santiago Villegas Erik Alejandro

Contenido

Introducción:

Objetivo del manual

El objetivo de este manual es principalmente ser una guía detallada para los desarrolladores y personal técnico involucrado en el proyecto “Automatización de una papelería”. Su propósito es documentar la arquitectura, componentes y funcionamiento interno del software, siendo frontend y backend, con el fin de facilitar las siguientes actividades:

- Mantenimiento: Realizar actualizaciones y correcciones de errores de forma eficiente.

- Ampliación: Agregar nuevas funcionalidades y características de la aplicación sin comprometer aquellas que ya se encuentren implementadas.

- Solución de problemas: Identificar y resolver rápidamente cualquier incidencia que pueda surgir.

Alcance del manual

El manual abarca la totalidad del desarrollo de la aplicación implementada para la papelería Catalunya, desde la concepción inicial hasta su implementación. Se contemplan los siguientes aspectos:

- Frontend: Diseño y desarrollo de la interfaz de usuario, utilizando

- Backend: Desarrollo de la lógica y control de la aplicación tomando en cuenta las funcionalidades que debe presentar, utilizando Kotlin Multiplataforma para el desarrollo de la aplicación y el servicio en línea de Firebase para las bases de datos.

- Funcionalidades

    - Sistema de usuarios: Creación y registro de usuarios para acceder a la aplicación.
    - Sistema de inventario: Creación, registro y despliegue de los productos encontrados en el inventario.
    - Sistema de ventas: Registro y a

Definiciones y acrónimos:

- **Frontend:** Es la interfaz de un sitio web o aplicación, desde su estructura hasta los estilos, como pueden ser la definición de los colores, texturas, tipografías, etcétera.

- **Backend:** Es el encargado de procesar toda la información que alimenta a un frontend. Se compone de marcos, bases de datos o códigos.

- **Base de datos no relacionales:** Las bases de datos no relacionales, o no SQL, están diseñadas específicamente para modelos de datos específicos y almacenan los datos en esquemas flexibles que se escalan con facilidad para aplicaciones modernas.

- **Rol de usuario:** Una categoría asignada a los usuarios que define sus permisos y acceso dentro del sistema. En este proyecto se maneja un rom general.

- **Jetpack Compose:** Es un kit de herramientas moderno diseñado para simplificar el desarrollo de las interfaces de usuario.

- **Kotlin:** Lenguaje de programación de código abierto y tipado estático.

- **Firebase:** Plataforma en la nube para el desarrollo de aplicaciones web y móvil.

- **Android Studio:** Entorno de desarrollo integrado oficial que se usa en el desarrollo de apps para Android.

- **Graddle:** Herramienta que permite la automatización de compilación de código abierto.

Requisitos del sistema:

Requisitos de Hardware

- Memoria RAM: 4.00GB
- Memoria ROM: 64.00GB
- Dimensión de la pantalla: 1512x720px

Requisitos de software

- Sistema Operativo: Android 10 o superior.
- Conexión a Internet.

Dependencias:

En el desarrollo de software, las dependencias son componentes externos, como librerías, frameworks o herramientas, que un proyecto necesita para funcionar correctamente. Estas proporcionan funcionalidades esenciales, optimizan el desarrollo y aseguran que el software cumpla con los estándares de calidad y rendimiento.

En este proyecto, las dependencias se gestionan mediante Gradle y están centralizadas en un archivo lib.versions.toml. Este enfoque permite mantener consistencia en las versiones, facilita la actualización y mejora la organización del código.

¿Por qué son importantes las dependencias?

1. Eficiencia: Permiten reutilizar código probado y optimizado, reduciendo el tiempo de desarrollo.
2. Compatibilidad: Aseguran que todas las partes del sistema trabajen juntas sin conflictos.
3. Actualización centralizada: Usar un archivo de configuración como lib.versions.toml simplifica el mantenimiento al controlar las versiones desde un único lugar.

Dependencias principales

El proyecto incluye diversas librerías y herramientas, clasificadas según su funcionalidad:

- Desarrollo multiplataforma:
    - Kotlin (2.0.21): Lenguaje principal del proyecto.
    - Compose Multiplatform (1.7.0): Framework para crear interfaces gráficas multiplataforma.
- Navegación:
    - Voyager (1.1.0-beta02): Librería para gestionar la navegación en la aplicación, incluyendo transiciones y pestañas.
- Servicios en la nube:
    - Firebase BOM (33.7.0): Gestiona versiones de servicios Firebase, como autenticación y Firestore.
- Interfaces de usuario:
    - Material Design 3 (1.3.1): Librería para implementar componentes de UI modernos y consistentes.
- Herramientas de Android:
    - AndroidX Core (1.13.1): Funcionalidades esenciales para aplicaciones Android.
    - Room KTX (2.6.1): Gestión de bases de datos local.

Entorno de desarrollo:

- **Kotlin Multiplataforma:** Versión 1.9.0. Lenguaje en que se desarrollo el ´proyecto.

- **Editor de código:** Android Studio. Herramienta para escribir, editar y simular el código fuente.

- **Base de datos:** Firebase. Es el sistema de gestión de bases de datos no SQ utilizado para almacenar la información de la aplicación.

Arquitectura del Software

1) Modulo de autenticación y gestión de usuarios

- Componentes:
  - Inicio de Sesión: Valida las credenciales del usuario. Si la autenticación es exitosa, se da acceso al sistema.
  - Registro: Crea nuevos usuarios con el rol de empelado y guarda la información en la base de datos.

- Interacciones:
  - El Inicio de sesión autentica al usuario contra la base de datos y luego permite acceder a información específica del perfil a través de la

2) Modulo de bases de datos:

- Componentes:
  - Usuarios: Tabla que almacena la información de los usuarios.
  - Productos: Información sobre los productos ofrecidos en el inventario.
  - Ventas: Es el registro que se lleva a cabo al momento de realizar una venta.
  - Detalle_Ventas: Es el desglose de la información especifica de las ventas realizadas.

- Interacciones:
  - El Backend se comunica con Firebase para validar credenciales de usuario, registrar nuevos usuarios, productos y ventas, y recuperar la información sobre los productos encontrados en Inventario.

3) Frontend (Interfaz de Usuario)

- Componentes utilizados:
  - Jetpack Compose: Se utilizó para diseñar y construir las interfaces de usuario de forma declarativa y moderna. Cada pantalla está diseñada como un Composable, asegurando un código limpio y legible.
  - Voyager: Responsable de manejar la navegación entre pantallas de forma eficiente. Proporciona transiciones fluidas y una gestión de la pila de navegación simplificada.

Interacciones entre módulos

El proyecto sigue el patrón de diseño MVC (Model-View-Controller), que organiza la lógica en tres capas principales para una separación clara de responsabilidades:

1. Modelo (Model):
    - Contiene la lógica de negocio y las interacciones con los datos.
    - En este proyecto, se encarga de validar datos (como credenciales de usuario) y gestionar estados relacionados con las operaciones.
2. Vista (View):
    - Representa la interfaz de usuario, diseñada con Jetpack Compose.
    - Cada pantalla se define como una vista que utiliza los datos proporcionados por el controlador y responde a las acciones del usuario.
3. Controlador (Controller):
    - Gestiona la lógica de la aplicación.
    - Recibe las acciones del usuario desde la vista, interactúa con el modelo para procesar datos, y actualiza la vista según sea necesario.
    - Por ejemplo, valida credenciales de usuario y maneja errores, como mostrar mensajes cuando las credenciales son inválidas.

Flujo de interacción en MVC

1. Cuando un usuario interactúa con la interfaz (View), como ingresar un correo y contraseña, estas acciones son gestionadas por el controlador (Controller).
2. El controlador verifica los datos a través del modelo (Model) y toma decisiones, como permitir el inicio de sesión o mostrar un mensaje de error.
3. La vista (View) se actualiza dinámicamente con los resultados del controlador, como transitar a otra pantalla o mostrar un mensaje de error en caso de credenciales inválidas.

Diseño de bases de datos:

Modelo de datos

Entidades:

- Usuarios:
  Representa a todos los empleados registrado en la papelería ala vez que guarda su información, así como sus credenciales.

- Productos:
  Contiene información sobre los productos ofrecidos al cliente, incluyendo el nombre del producto, su código, precio de compra, precio de venta y la cantidad disponible en inventario.

- Ventas:
  Es donde se registra la información generada al momento de generar una venta, tal como lo es el ID de la venta, Tipo de pago, numero de documento, el monto a pagar, monto recibido, el cambio a dar, etcétera.

- Detalle Ventas:
  Es un despliegue más detallado de las ventas realizadas, en donde se puede observar

Diccionario de datos:

Tabla de usuarios

Tipo de llave	Nombre	Tipo de dato	Tamaño	Descripción	Valor Nulos
PK	idusario	Integer		Código de identificación del empleado	No
	nombrecompleto	Charvar	100	Nombre del empleado	No
	correo	Charvar	100	Correo del empleado	No
	clave	Charvar	100	Clave de acceso del empelado	No
	fecharegistro	Timestamp		Fecha de registro del empleado	No

Tabla de Productos

Tipo de llave	Nombre	Tipo de dato	Tamaño	Descripción	Valor Nulos

	codigo	Charvar	100	Código de barras del producto	No
FK	idcategoria	Charvar		Código de identificación de la categoría	No
	descripcion	Charvar	100	Descripción del producto	No
	preciocompra	Numeric	10,2	Precio de compra del producto	No
	precioventa	Numeric	10,2	Precio de venta del producto	No
	inventario	Integer		Cantidad disponible en inventario	No
	fecharegistro	Timestamp		Fecha de registro del Stock	No
PK	idproducto	Charvar	5	Código de identificación del producto	No

Tabla de Ventas

Tipo de llave	Nombre	Tipo de dato	Tamaño	Descripción	Valor Nulos
PK	idventa	Integer		Código de identificación de la venta	No
	tipopago	Charvar	50	Método de pago en que se realizó la venta	No
	numerodocumento	Charvar	50	Numero de documento que genera la venta	No
	documentocliente	Charvar	50	Número de ticket de venta	No
	nombrecliente	Charvar	100	Nombre del cliente	No
	montopagocon	Numeric	10,2	Cantidad con la que paga el cliente	No
	montocambio	Numeric	10,2	Cambio que se le da al cliente	No
	montosubtotal	Numeric	10,2	Precio original del producto	No
	montoigv	Numeric	10,2	IVA agregado	No
	montototal	Numeric	10,2	Cantidad total a cobrar	No
	fecharegistro	Timestamp		Fecha en que se registró la venta	No

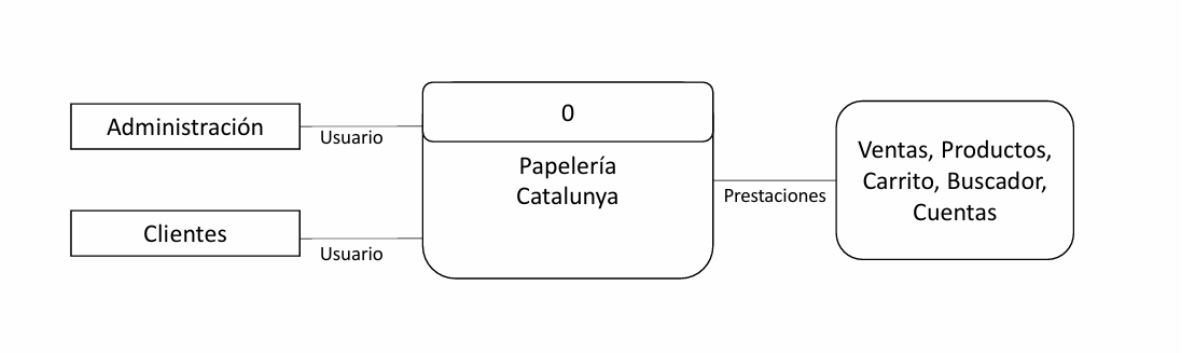
Tabla de Detalle de ventas

Tipo de llave	Nombre	Tipo de dato	Tamaño	Descripción	Valor Nulos
PK	iddetalleventa	Integer		Código de identificación del reporte de venta	No
FK	idventa	Integer		Código de identificación de la venta	No
FK	idproducto	Integer		Código de identificación del producto	No
FK	precioventa	Numeric	10,2	Precio de venta del producto	No
FK	cantidad	Integer		Cantidad vendida de un producto	No
FK	total	Numeric	10,2	Total a cobrar	No
	fecharegistro	Timestamp		Fecha de registro de la entrada	No

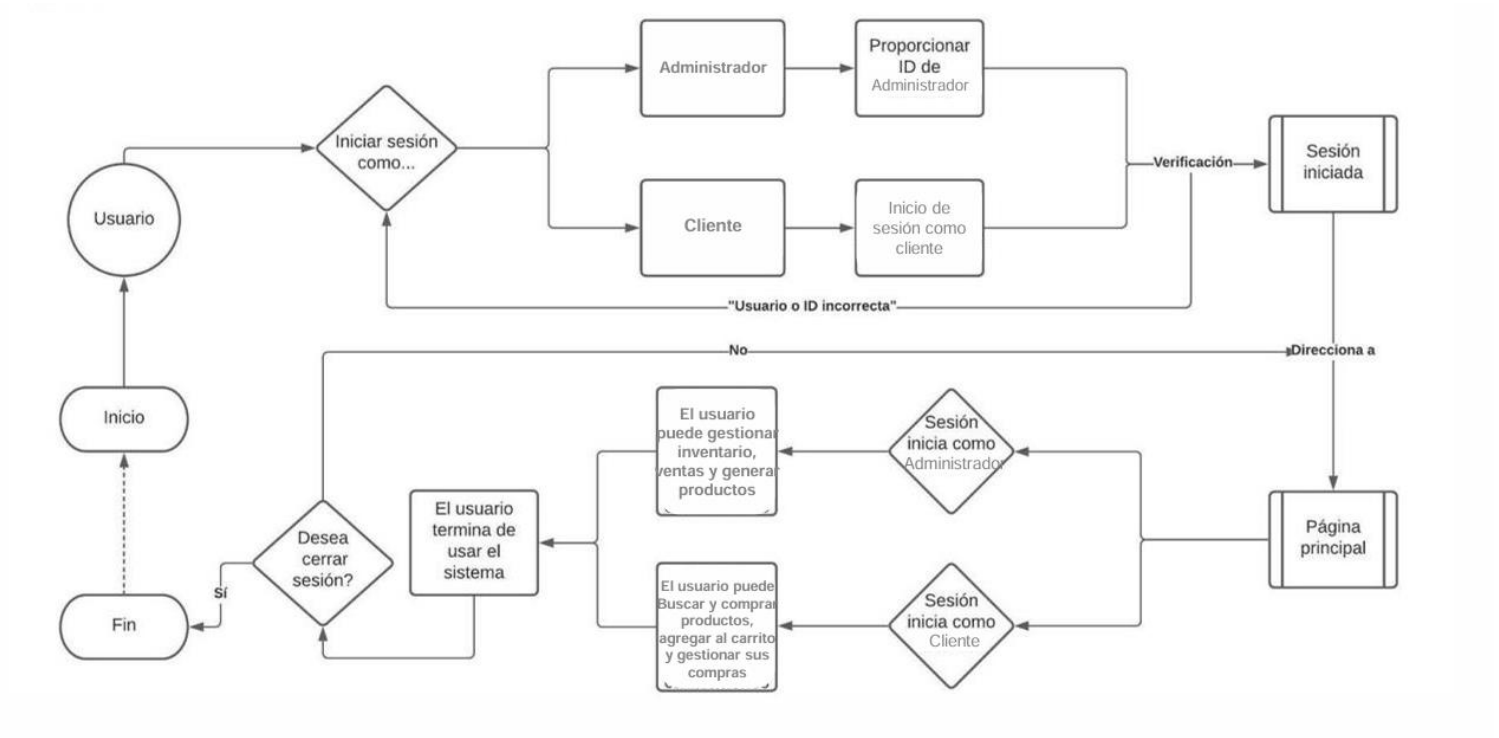
Diagramas

Diagramas de Fujo de Datos

Nivel 0



Nivel 1



Nivel 2

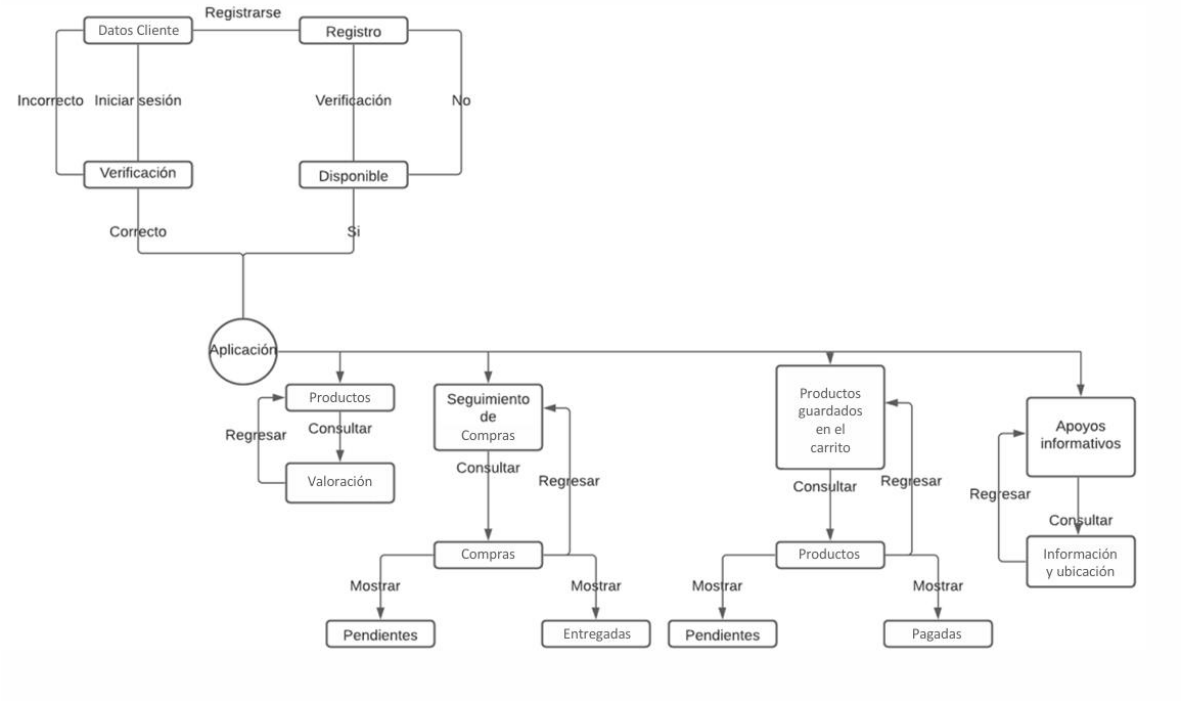


Diagrama de clases