Universidad Politécnica de Chiapas.

Cuatrimestre.	Septiembre- Diciembre 2025
Grupo.	Noveno B
Asignatura.	Programación para Moviles 2.
Corte	C1
Actividad.	Proyecto 2-> Aplicación móvil
Fecha de entrega.	___
Matricula.	221189
Nombre.	Luis Alberto Batalla Gonzàlez.

1. Introducción.

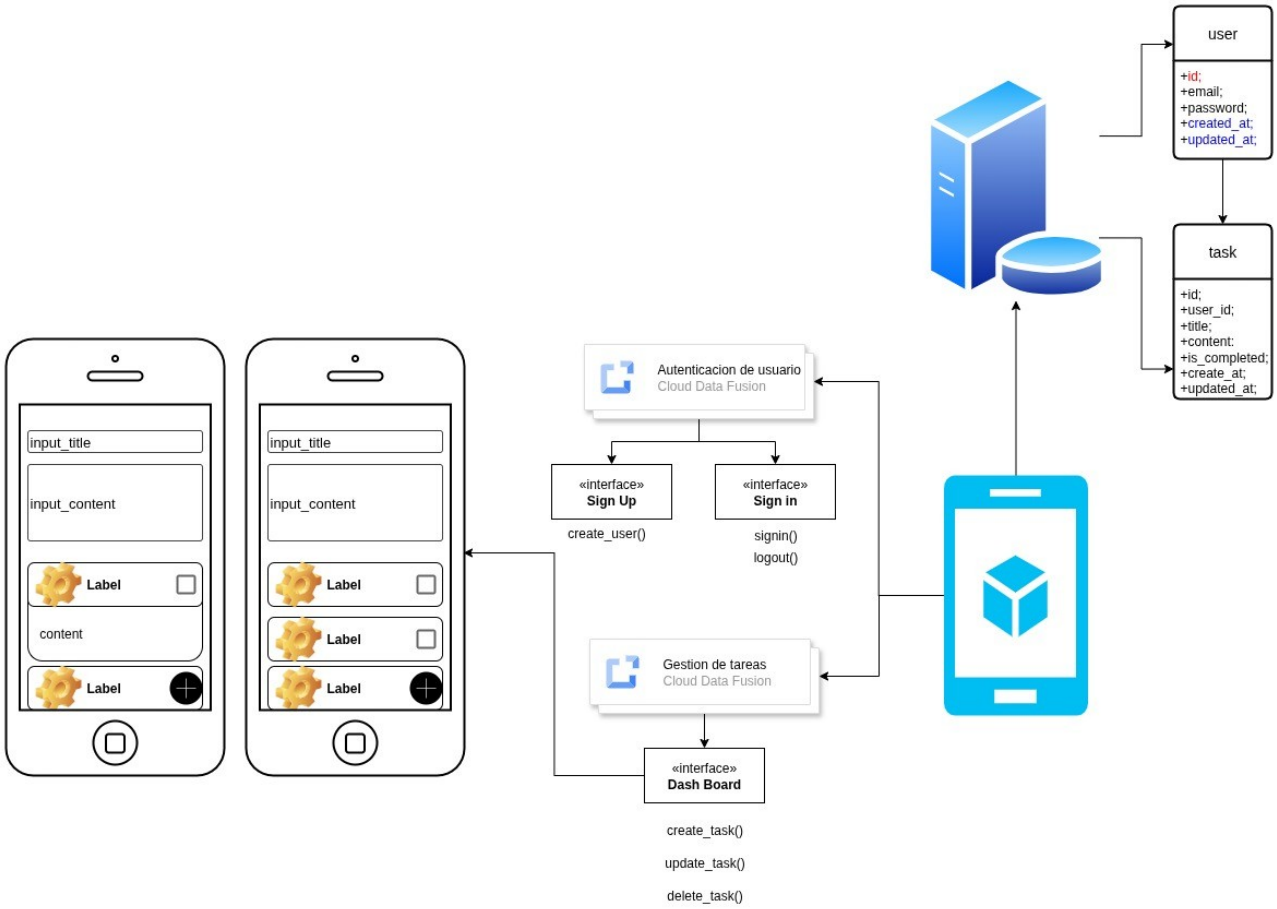
La aplicación es un Todo List Básico que interactua con un API echa a medida.

El Api incluye los metodos CRUD de las entidades Usuarios y Tareas. Es una Base de datos que agrega persistencia a la apicacion.

La aplicación cubre los servicios de gestion de usuarios y gestion de tareas, adicional a esto cuenta con una pantalla de progreso. La aplicación fué modificada para cumplir con el requerimiento tecnico: Gestion de Capturas de pantalla. Un requrimiento academico agregádo para la materia “Seguridad de la información”.

2. Arquitectura

Prinsipalmente, el proyecto es un cliente-servidor con dos dominios: “Autenticacion de usuario” y “Gestion de tareas”.



La Aplicacion móvil se desarrollo en arquitectura MVVM, y se relaciona con Clean Architecture, Donde vista cunple el rol de capa de presentacion y View model remplaza la capa de Datos. Su estilo la vuelve escalable y adaptable a otras a otras arquitecturas, como un Clean Architecture completo o un Screaming Architecture, que incluye dos funcionalidades Authentication y Task. Usa Provider para realizar inyecciones y consumir datos.

Gestión de Estado.

Provider es la herramienta principal para gestion de estados. Se basa en una idiologia de de separacion de responsabilidades, básada en MVVM. La lista de tareas en el Dashboard principal se usando un consumer, que recibe la lista del ViewModel.

Provider funciona como un sistema de inyeccion de Dependencias.

4. Cliente HTTP.

El cliente HTTP se implemento con la librería dio. Las llamadas al API se encuentran en ViewModel, manteniendo la vista libre de responsabilidades.

Para obtener la lista de tareas, se realiza una petición GET al endpoint /tasks. Es crucial incluir el token de autenticación en las cabeceras para que el servidor sepa a qué usuario pertenecen las tareas.

Para crear una nueva tarea, se envía una petición POST al endpoint /tasks. El cuerpo (data) de la petición contiene la información de la nueva tarea (título y contenido).

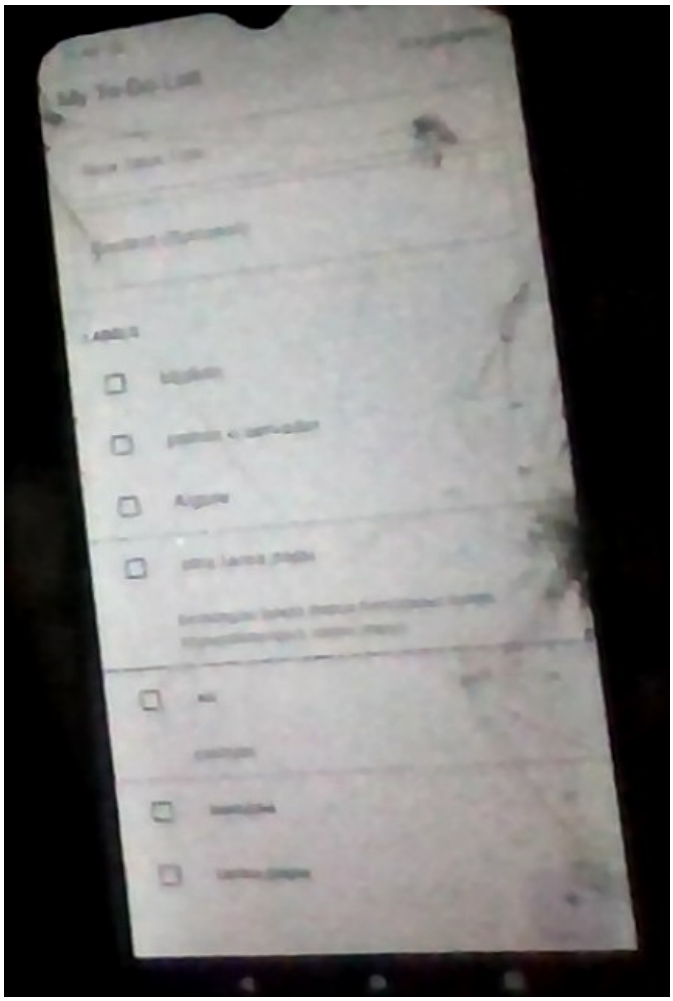
Para eliminar tareas, se realiza una petición DELETE al endpoint /tasks/{id} por cada tarea seleccionada. El ID de la tarea se pasa como parte de la URL.

Desde Android 9, el sistema bloquea el trafico http, para permitirlo se añade el permiso:

android:usesCleartextTraffic="true" en el archivo AndroidManifest.xml.

5. Interfaz Gráfica.

Se adjuntan capturas de pantalla.



No es posible tomar capturas directas por el requerimientro academico añadido.

MultiProvider es el widget más importante para la arquitectura de la aplicación. Ubicado en main.dart, su función es inyectar los ViewModels (AuthViewModel, TaskViewModel) en la parte superior del árbol de widgets, haciéndolos accesibles para cualquier pantalla que los necesite.

El widget Consumer se utiliza en todas las pantallas para escuchar los cambios en un ViewModel específico y reconstruir solo la parte necesaria de la interfaz, lo cual es mucho más eficiente que reconstruir toda la pantalla.

El Consumer envuelve la lista de tareas. Cuando se añade, elimina o actualiza una tarea en TaskViewModel, solo este ListView se redibuja.

En SignUpScreen, se utiliza un patrón interesante para dar feedback al usuario. Dentro de un Consumer, se usa WidgetsBinding.instance.addPostFrameCallback para mostrar un SnackBar o navegar a otra pantalla después de que el frame actual se haya construido. Esto evita errores comunes

de "setState() or markNeedsBuild() called during build". Este enfoque permite que la UI reaccione a los cambios de estado de una manera segura.

Conclusiones.

Evidentemente es un reto intelectualmente logico pero se logra si se prepara correctamente la preproduccion o planeacion del sistema.

Es un un sitema que consta de dos piezas de software echas a medida. Lo que facilita su manejo/mdificacion, mantenimiento y expancion.

Aprendi a usar inteligentemente el lenguaje Dart con sus bases “Wiged”. Aprendi y aplique la arquitectura MVVM.

No hai muchas mejoras futuras porque el proyecto es pequeño, pero se puede terminar de aterrizar la arquitectura en una más solida como una completa con Clean Rchitecture o una separacion de responsabilidades con Vertical Slicing.