**“Documento de Arquitectura”**

PROYECTO: “Desarrollo de un sistema de gestión para pequeños productores agrícolas”

Integrantes – Año 2023

| Legajo | Nombre | E-Mail |
| --- | --- | --- |
| 1639201 | Cervantes Ortiz, Marisol Carolina | mari\_967@frba.utn.edu.ar |
| 1443331 | Gonzalez, Lourdes Ayelen | lourdesgonzalez@frba.utn.edu.ar |
| 1670475 | Iakantas, Gabriel Maximiliano | giakantas@frba.utn.edu.ar |
| 1633818 | Mazzarella, Chiara Betiana | cmazzarella@frba.utn.edu.ar |
| 1522220 | Olmedo Paco, Jhon Daniel | jhonpaco@frba.utn.edu.ar |

Profesores:

***Director de Cátedra:*** *Mag. Ing. Gabriela Salem*

***Profesor a cargo del curso:*** *Mag. Ing. Gabriela Salem, Ing. Claudio Crescentini.*

***Profesor a cargo del proyecto:*** *Mag. Ing. Gabriela Salem, Ing. Claudio Crescentini, Ing. Emiliano Cortez.*

Contenido

[**1. Objetivo 4**](#_heading=h.3dy6vkm)

[**2. Diagrama de Contexto 4**](#_heading=h.1t3h5sf)

[**3. Diagrama de Arquitectura 5**](#_heading=h.x1itlns7eipj)

[3.1 Comparación entre Firebase y arquitectura tradicional 5](#_heading=h.d0aamrxlbbnn)

[**4. Modelo de capas y componentes del stack. 6**](#_heading=h.ggp4np8jbc7h)

[4.1. Detalles de la capa Cliente. 6](#_heading=h.jmavvm3bkc1d)

[4.2. Detalles de las capas de Integración, aplicación y persistencia 6](#_heading=h.98sl2zlrtmia)

[**5. Atributos de calidad – Requisitos No Funcionales 7**](#_heading=h.6rey0lfcxlc6)

[**6. Tácticas para garantizar RNF 8**](#_heading=h.i6zyl3b20eu7)

[**7. Políticas de backup / Continuidad de negocio. 8**](#_heading=h.s5dd9know6iy)

[**8. APIs de integración 8**](#_heading=h.pk1qk3tnp7jv)

[**9. Patrones de diseño y arquitectura 8**](#_heading=h.elwv5of6epb4)

[9.1. Arquitectura para la aplicación móvil 8](#_heading=h.44z9ggegp4jo)

[9.2. Arquitectura para el backend de la aplicación 9](#_heading=h.jp66ccvik04q)

Historial de Revisión

| Fecha | Versión | Descripción | Rol | Autor |
| --- | --- | --- | --- | --- |
| 25/05/23 | 1.0 | Creación del documento. | Desarrollador Senior;  Desarrollador Junior  Product Owner | Marisol Cervantes;  Chiara Mazzarella;  Jhon Olmedo;  Gabriel Iakantas; |
| 05/06/23 | 2.0 | Cambios en el diagrama de contexto. | Desarrollador Senior;  Desarrollador Junior  Product Owner | Marisol Cervantes;  Chiara Mazzarella;  Jhon Olmedo;  Gabriel Iakantas; |

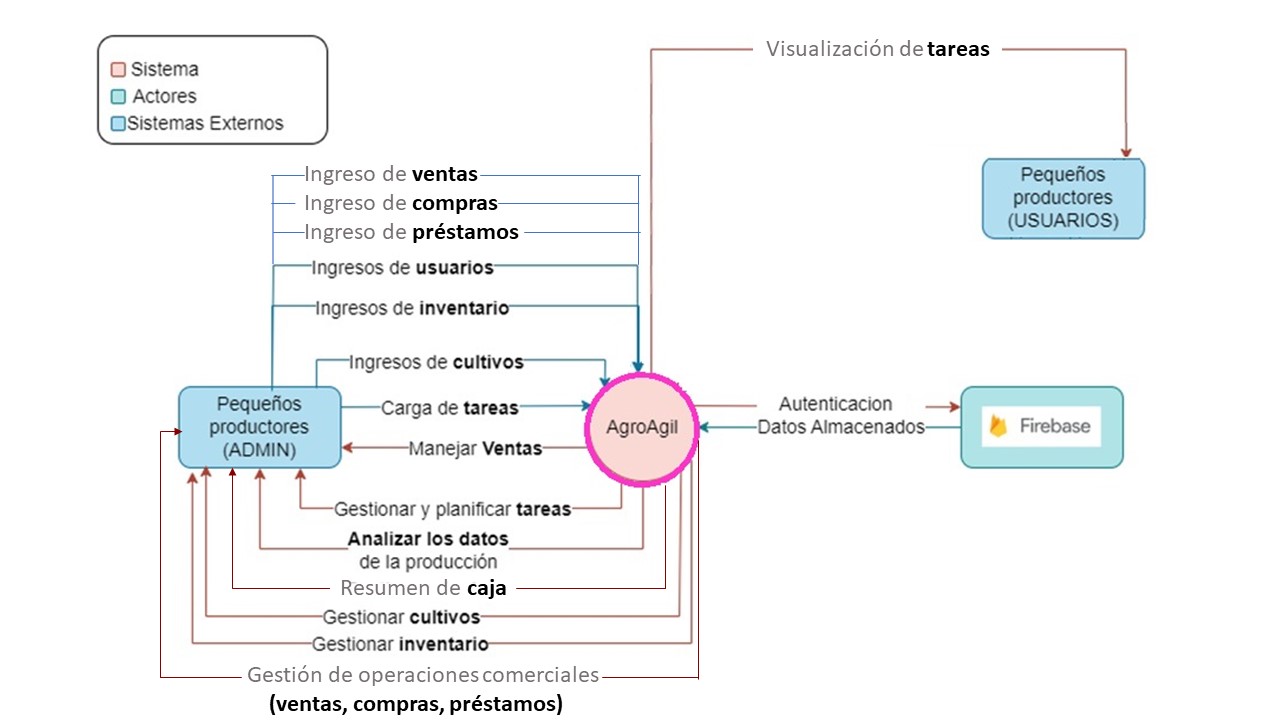
# Objetivo

Este documento tiene como objetivo describir la arquitectura general de la solución que se va a desarrollar en el proyecto. Se van a indicar los componentes individuales que forman parte de la solución y cómo interactúan.

También, se va a incluir el diagrama de contexto y el de arquitectura.

# Diagrama de Contexto

Este diagrama muestra las entidades que interactúan con el sistema determinando los límites y el ambiente del mismo.



En el caso de nuestro sistema, las acciones a realizar por parte de los usuarios dependen de sus roles (administrador de la granja y trabajadores), por lo que se implementarán distintos grados de acceso a la información.

El sistema permite al administrador de la granja gestionar tareas del campo, operaciones comerciales (compras, ventas, préstamos) e inventario. También, podrá visualizar un resumen de sus ingresos y egresos tanto en el almacén como en la caja.

Por otro lado, los trabajadores tendrán solamente permiso para ver las tareas que les asignó el administrador de la granja.

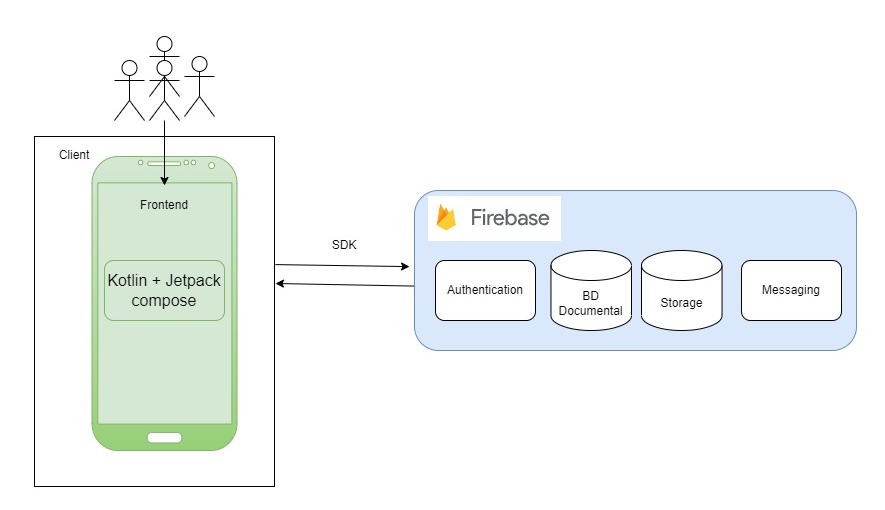
Como entidad externa tenemos el proveedor de Firebase que permite diferentes formas de autenticación, ya sea con una cuenta de Google o de otro tipo.

En el proyecto, se va a poder iniciar sesión en la aplicación utilizando únicamente la cuenta de Google.

# Diagrama de Arquitectura

Este diagrama pretende mostrar la interacción lógica y física de los diversos componentes que conforman el sistema.

Los agricultores interactúan con una aplicación móvil como Frontend que se comunica con el backend de Firebase a través de solicitudes HTTP.



## 3.1 Comparación entre Firebase y arquitectura tradicional

Firebase es un servicio de Backend as a Service de Google (BaaS), es decir, que ofrece soluciones para las funciones de infraestructura, como la configuración del servidor, la autenticación de usuarios, la integración de bases de datos, los servicios de notificación, la escalabilidad, entre otras.

Por lo mencionado en el párrafo anterior, fue elegido Firebase, ya que permite acelerar el desarrollo. Un backend tradicional requiere más trabajo de codificación y configuración.

|  |
| --- |

# Modelo de capas

## Detalles de la capa Cliente.

La capa Cliente se enfoca en la interfaz de usuario. También se centra en la interacción con los usuarios a través de dispositivos móviles.

Se desarrollará una aplicación móvil específica para pequeños agricultores familiares, que estará disponible en Android.

Será desarrollada con el lenguaje de programación Kotlin y el framework para construir componentes visuales será Jetpack compose.

Proporcionará las funcionalidades de gestión de : tareas, almacén, y operaciones comerciales (compras, ventas, préstamos) . Además, se podrán visualizar los ingresos y egresos en caja e información relevante en la pantalla principal de la aplicación para tomar decisiones.

La aplicación se comunicará con el backend a través de una API RESTful para realizar operaciones y acceder así a los datos.

## Detalles de las capas de Integración, aplicación y persistencia

Kotlin interactúa con los servicios de Firebase directamente, ya que ambos están integrados de manera nativa en el ecosistema de desarrollo de Android. Se deben configurar algunas bibliotecas para aprovechar al máximo la integración. No necesitan ningún intermediario.

El almacenamiento de datos se hará con la base de datos Firebase Realtime Database.

También, se definirán reglas de seguridad y acceso para controlar quién puede realizar qué acciones sobre los datos.

# Atributos de calidad – Requisitos No Funcionales

Los principales atributos no funcionales del sistema son:

| **Atributo de calidad** | |
| --- | --- |
| **Estímulo** | Origen del Estímulo | | Pequeños Productores |
| Estímulo | | Uso de la aplicación sin internet. |
| **Ambiente** | Ambiente | | Uso normal de la aplicación en el campo |
| Componente | | Carga o actualizaciones de tareas. Carga o actualizaciones en el módulo de cultivo |
| **Respuesta** | Respuesta | | Sitio disponible, poder realizar la operación deseada con una posterior sincronización de datos. |
| Medida de Respuesta | | Disponibilidad el 99.9% del tiempo. |

| **Atributo de calidad** | |
| --- | --- |
| **Estímulo** | Origen del Estímulo | | Pequeños Productores |
| Estímulo | | Entendimiento intuitivo y sencillo de la aplicación. |
| **Ambiente** | Ambiente | | Uso normal del aplicativo. |
| Componente | | Tareas, cultivos,almacén |
| **Respuesta** | Respuesta | | La interfaz que se les presente debe ser clara, intuitiva y práctica |
| Medida de Respuesta | | El usuario debe poder ingresar a las diferentes actividades rutinarias con no más de 3 clicks. |

| **Atributo de calidad** | |
| --- | --- |
| **Estímulo** | Origen del Estímulo | | Pequeños Productores |
| Estímulo | | Protección de la confidencialidad de los datos |
| **Ambiente** | Ambiente | | Uso normal del aplicativo. |
| Componente | | Tareas, cultivos, almacén, administración de usuario |
| **Respuesta** | Respuesta | | Solo el usuario autorizado puede ver los datos de los módulos de la aplicación |
| Medida de Respuesta | | Para poder usar la aplicación el usuario deberá autenticarse. Además se implementará autorización por roles (administrador y trabajador) |

# 

# Tácticas para garantizar RNF

* **Táctica para garantizar la disponibilidad el 99.9% del tiempo:** El backend de Firebase tiene una SLA para la disponibilidad de 99,95% por lo que usándolo garantizamos este nivel de disponibilidad. Además permitimos a los usuarios usar la aplicación sin conexión y que estos se sincronicen al recuperar la conectividad.
* **Táctica para garantizar la usabilidad de la aplicación**: se establece que el acceso a las funcionalidades deben realizarse con 3 o menos clicks. A partir de relevamientos se identificarán las funciones claves de la aplicación, se realizarán prototipos de las mismas y se harán pruebas de usabilidad con usuarios.
* **Táctica para garantizar autenticación y autorización:** se realizará integración con el servicio de Firebase Authentication para permitir el logueo con utilizando su correo de google.

# Políticas de backup / Continuidad de negocio.

Para la realización de los respaldos se utilizará la opción de copias de seguridad automáticas de Firebase Realtime Database, que permite crear copias de seguridad diarias de los datos y almacenarlos en un bucket de Cloud Storage*[[1]](#footnote-0)*.

Esto se hará a partir de que la aplicación se encuentre productiva.

# APIs de integración

La aplicación se conectará con las apis de AccuWeather*[[2]](#footnote-1)* para ver datos del clima.

# Patrones de diseño y arquitectura

## Arquitectura para la aplicación móvil

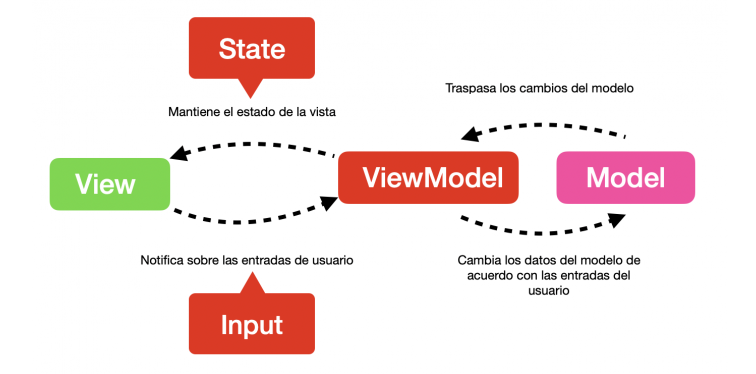
Como patrón de arquitectura para la aplicación móvil en android decidimos usar MVVM (Model, View y ViewModel).  
Separa la lógica de negocio de la interfaz de usuario (UI), haciendo que el código sea más fácil de leer, de mantener y de probar. MVVM es especialmente popular en el desarrollo de aplicaciones móviles, incluyendo plataformas como Android y iOS.

Los componentes de la arquitectura son los siguientes:

* **Model (Modelo):** Representa los datos y la lógica de negocio. Es el estado subyacente y los datos que la aplicación gestiona.
* **View (Vista):** Es la interfaz de usuario. Presenta la información al usuario y gestiona las interacciones del usuario con la aplicación.
* **ViewModel (Modelo de Vista):** Encapsula la lógica de la interfaz de usuario. Proporciona datos a la vista y mantiene el estado de la interfaz de usuario.

Ventajas

* La lógica de negocio está desacoplada de la interfaz de usuario.
* Es más fácil de mantener y probar. Se pueden hacer pruebas unitarias para el modelo y para vista-modelo, sin necesidad de hacer referencia a la vista.
* Los componentes pueden ser reutilizados
* El mantenimiento de los sistemas es simplificado.



## Arquitectura para el backend de la aplicación

Para el backend de la aplicación no usaremos una arquitectura para construirlo, ya que utilizamos un BaaS.

Aprobaciones

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Fecha:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Aprobación Técnica de la Arquitectura

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Fecha:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Profesor a cargo del Proyecto

1. Firebase Backups: [https://firebase.google.com/docs/database/backups](https://firebase.google.com/docs/database/backups?hl=es-419) [↑](#footnote-ref-0)
2. AccuWeather API: <https://developer.accuweather.com/apis> [↑](#footnote-ref-1)