# **TITULO PORTADA TRABAJO**

# Ejemplo de subtítulo

NOMBRE:

CARRERA:

ASIGNATURA:

PROFESOR:

FECHA:

# Introducción

Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo Este es un texto de ejemplo

# Arquitectura Cliente-Servidor en Android Studio con Java

1. **Cliente (Frontend):**

**Responsabilidad**: Manejar la lógica de negocio, la interfaz de usuario, y las interacciones con el usuario. La mayor parte del procesamiento se realizará localmente en el dispositivo.

**Implementación**:

**Actividades y Fragmentos**: Utiliza **Activities** y **Fragments** para definir las diferentes pantallas y la interacción del usuario con los datos de animales y plantas.

**Almacenamiento local**: Los datos temporales o no críticos se pueden almacenar en el dispositivo usando **SQLite** o **Room** (una librería de persistencia de Android) para mantener la información accesible sin necesidad de conexión continua.

**Conectividad con la nube**: Cuando se necesite guardar o recuperar datos de animales y plantas, la aplicación enviará peticiones HTTP al servidor mediante bibliotecas como **Retrofit** o **OkHttp**.

**Interfaz gráfica**: El diseño de la interfaz se llevará a cabo usando los layouts y vistas nativas de Android, que se definirán en XML o directamente en código Java.

1. **Servidor (Backend en la nube)**

**Responsabilidad**: Gestionar la base de datos y proporcionar la API para la sincronización de los datos. La lógica pesada y el almacenamiento de datos se manejarán en el servidor.

**Implementación**:

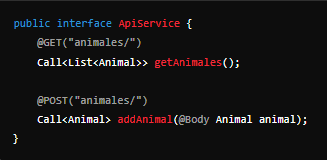
**API RESTful**: El servidor, desarrollado con tecnologías como Node.js, Django o Flask, proporcionará los endpoints necesarios para almacenar y recuperar datos.

**Base de datos**:**Firebase** para guardar la información sobre animales y plantas.

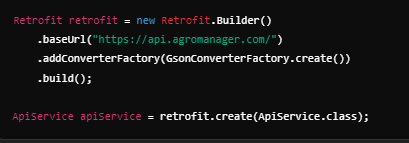
1. **Comunicación Cliente-Servidor**

**Retrofit/OkHttp**: En Java, se recomienda usar **Retrofit** para realizar las llamadas HTTP desde la app hacia el servidor. Esta librería permite manejar las peticiones y respuestas de manera eficiente y estructurada.

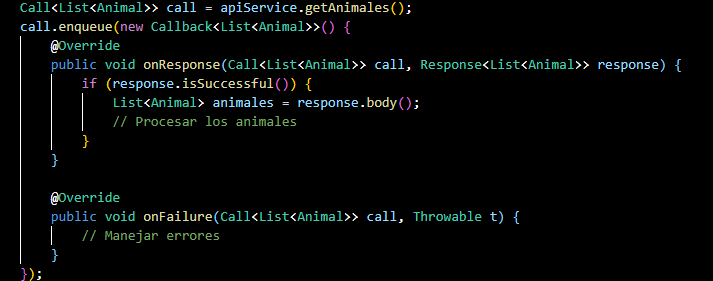
Ejemplo de uso de Retrofit en Java:



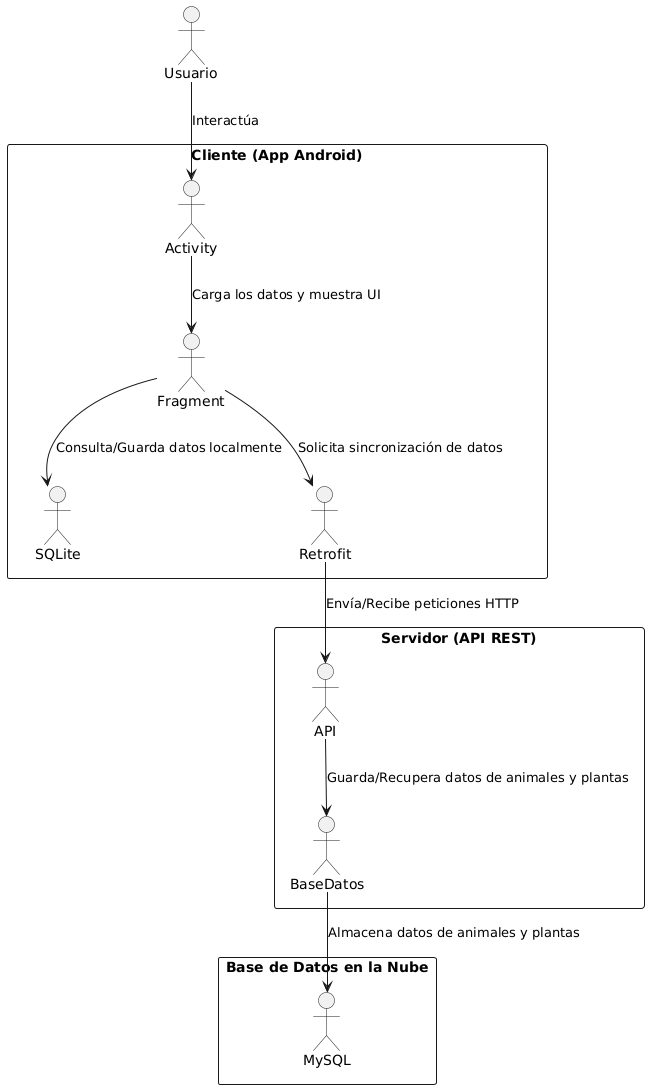
Confiuguracion de Retrofit:



Llamada desde la app:



Ejemplo de implementación:



### **Flujo de Interacción:**

1. **Pantalla de gestión de animales y plantas**: El usuario interactúa con la app en su dispositivo Android para visualizar o actualizar datos.
2. **Lógica local**: La aplicación realiza las acciones internas localmente, como cálculos o gestión de recordatorios, usando **SQLite** o **Room**.
3. **Sincronización con la nube**: Los datos de animales y plantas se sincronizan con el servidor a través de la API REST, utilizando Retrofit.
4. **Servidor**: Recibe las solicitudes de la app, procesa los datos y actualiza la base de datos en la nube.

### **Ventajas de Usar Android Studio con Java:**

* **Compatibilidad nativa**: Desarrollar directamente en Android Studio con Java asegura una integración nativa con todas las capacidades de Android, optimizando el rendimiento.
* **Amplia documentación y soporte**: Java y Android Studio son herramientas maduras y bien documentadas, lo que facilita el desarrollo y el soporte a largo plazo.
* **Gestión eficiente del ciclo de vida**: Android Studio permite manejar fácilmente el ciclo de vida de las aplicaciones móviles, asegurando que los datos y los estados de la aplicación se mantengan a través de interrupciones.

### **Herramientas y Tecnologías Recomendadas:**

* **Retrofit o OkHttp**: Para la comunicación HTTP con el servidor.
* **Gson**: Para la conversión de datos JSON entre el servidor y la app.
* **SQLite/Room**: Para el almacenamiento de datos local en el dispositivo.
* **LiveData y ViewModel (opcional)**: Para manejar los datos de manera más eficiente y con un ciclo de vida mejorado.

### **Propuesta de Próximos Pasos:**

1. **Configurar Retrofit en la app** para manejar las solicitudes a la API en la nube.
2. **Diseñar la base de datos local** usando SQLite o Room para gestionar los datos cuando no haya conexión.
3. **Definir la interfaz de usuario en XML** y estructurar las actividades y fragmentos según las necesidades de la aplicación.
4. **Implementar la sincronización con el servidor** para guardar y recuperar datos de animales y plantas.

# 3 Estandares de programación segura

### **Perfiles y Roles en el Sistema**

El sistema define diferentes roles con responsabilidades específicas y niveles de acceso diferenciados para garantizar la seguridad de la información. A continuación, se detallan los perfiles y roles principales:

#### **1. Administrador**

* **Responsabilidades**:
  + Crear, editar y eliminar usuarios.
  + Acceso completo a todos los módulos y funciones del sistema.
  + Gestión de contenido, incluyendo la creación y eliminación de libros y artículos.
  + Monitoreo de la actividad del sistema y generación de informes.
* **Seguridad**:
  + Implementar autenticación multifactor (MFA) para proteger las cuentas de los administradores.
  + Limitar el acceso administrativo solo a direcciones IP específicas, cuando sea posible.
  + Realizar auditorías regulares de las acciones ejecutadas por los administradores.

#### **2. Usuario Regular**

* **Responsabilidades**:
  + Visualizar información sobre libros y revistas disponibles en el sistema.
  + Modificar su propio perfil, incluyendo datos personales y preferencias.
  + Participar en intercambios de libros, como solicitar y ofrecer ejemplares.
* **Seguridad**:
  + Validar todas las entradas de datos para prevenir ataques como la inyección SQL.
  + Implementar controles de acceso que aseguren que los usuarios solo puedan modificar sus propios datos.
  + Proteger los datos sensibles, como contraseñas, mediante técnicas de hashing seguro (por ejemplo, bcrypt).

#### **3. Invitado**

* **Responsabilidades**:
  + Acceso restringido al contenido público del sistema, como una lista de libros disponibles y descripciones generales.
* **Seguridad**:
  + Garantizar que los invitados no tengan acceso a ninguna información confidencial.
  + Implementar limitaciones de tasa de solicitudes para prevenir ataques de denegación de servicio (DoS).
  + Deshabilitar cualquier funcionalidad que permita a los invitados realizar modificaciones en el sistema.

### **Privilegios de Acceso por Perfil**

Cada perfil tiene privilegios específicos que limitan sus capacidades dentro del sistema:

* **Administrador**:
  + CRUD completo (crear, leer, actualizar, eliminar) en todos los módulos del sistema.
  + Acceso total a todas las funcionalidades del sistema.
* **Usuario Regular**:
  + Acceso limitado a la lectura y modificación de su propio perfil.
  + No tiene acceso a la gestión de otros usuarios ni a la creación o eliminación de contenido general.
* **Invitado**:
  + Solo puede visualizar información pública, sin acceso a funciones de modificación.

### **Prácticas de Seguridad Implementadas**

Se han adoptado diversas prácticas para garantizar que el sistema cumpla con altos estándares de seguridad:

#### **1. Validaciones de Entrada**

* Se validan todos los datos que el usuario introduce para prevenir vulnerabilidades como inyección SQL y ataques de cross-site scripting (XSS). Las consultas parametrizadas se utilizan en todas las interacciones con la base de datos.

#### **2. Cifrado de Datos**

* Las contraseñas se almacenan utilizando algoritmos de hashing seguro, como bcrypt. Además, los datos en tránsito entre la app y la base de datos están protegidos mediante SSL/TLS para garantizar la privacidad de la información.

#### **3. Gestión de Errores**

* Los errores se manejan de forma segura, sin exponer detalles técnicos al usuario. Los registros (logs) de errores se mantienen internamente para análisis y diagnóstico, sin comprometer la seguridad del sistema.

#### **4. Gestión de Sesiones**

* Se han implementado tiempos de expiración para las sesiones inactivas. Las cookies de sesión están marcadas como seguras (Secure) y accesibles solo por el servidor (HttpOnly) para protegerlas de ataques de cross-site scripting (XSS).

#### **5. Control de Acceso Basado en Roles (RBAC)**

* El sistema está diseñado para que los usuarios solo puedan realizar acciones permitidas según su rol. Esto garantiza que las acciones más sensibles estén limitadas a los administradores, mientras que los usuarios regulares y los invitados tienen acceso restringido a sus respectivos ámbitos.

### **Pruebas de Seguridad**

* Se realizarán pruebas de penetración periódicas para identificar posibles vulnerabilidades en la gestión de roles y accesos.
* El código será auditado regularmente para asegurar que se sigan las mejores prácticas de seguridad recomendadas por OWASP.

# 4

# 5 Plan de Prueba AgroManager App

### **Introducción**

Este documento presenta el Plan de Pruebas para la AgroManager App, diseñada para facilitar la gestión de animales de granja y plantas agronómicas. El objetivo del plan es asegurar que todas las funcionalidades de la aplicación cumplan con los requisitos establecidos y las expectativas de los usuarios, garantizando calidad, seguridad y funcionamiento.

#### **Objetivo del Plan de Pruebas**

El objetivo principal de este plan es definir el enfoque, el alcance y las estrategias de pruebas necesarias para verificar que la AgroManager App cumpla con las expectativas y necesidades del usuario final. A través de pruebas funcionales, de seguridad, de integración y de rendimiento, se garantizará la correcta implementación de todas las funcionalidades asociadas.

#### **Alcance del Plan de Pruebas**

El alcance cubre las principales funcionalidades de la aplicación:

* Registro y gestión de animales.
* Registro y gestión de plantas.
* Cálculo de necesidades para animales.
* Recordatorios automáticos para cuidados.
* Acceso a información desde la base de datos (solo modificable por desarrolladores).

### **Historias de Usuario y Criterios de Aceptación**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Rol** | **Característica / Funcionalidad** | **Razón / Resultado** | **Criterios de Aceptación** |
| H01 | Estudiante | Ver una lista de todas las especies de animales y plantas. | Para obtener una visión general de las especies gestionadas. | El usuario puede ver la lista de animales y plantas.  El usuario puede seleccionar una especie para ver más información. |
| H02 | Estudiante | Filtrar las especies por categoría (animal o planta). | Para encontrar más fácilmente la información que necesita. | El estudiante puede aplicar un filtro para ver ciertos animales o ciertas plantas en la lista. |
| H03 | Estudiante | Acceder a la información detallada de una especie seleccionada. | Para conocer la alimentación, cuidados y espacio necesario. | El estudiante puede ver detalles como imágenes, cuidados, alimentación y necesidades de espacio. |
| H04 | Estudiante | Acceder al apartado de plagas relacionadas con las plantas. | Para identificar amenazas y combatirlas de manera efectiva. | El estudiante puede ver un apartado de plagas asociado a cada planta con sus características. |
| H05 | Estudiante | Ver la época recomendada de siembra para cada planta. | Para planificar mejor los cultivos. | El estudiante puede ver la época recomendada de siembra en la pantalla de detalles de la planta. |
| H06 | Estudiante | Ver imágenes adicionales de las especies. | Para identificar visualmente plantas o animales en la granja. | El estudiante puede ver varias imágenes de las especies en la pantalla de información. |
| H07 | Estudiante | Agregar recordatorios automáticos para tareas comunes. | Para mantener una rutina eficiente sin olvidar tareas importantes. | El estudiante puede seleccionar tareas predefinidas como "alimentación", "control de plagas" y "riego" para generar recordatorios automáticos. |
| H09 | Estudiante | Ingresar recordatorios personalizados para tareas adicionales. | Para recordar tareas específicas como limpieza de establos. | El estudiante puede agregar recordatorios personalizados con un campo de texto libre. |
| H10 | Estudiante | Visualizar todos los eventos programados de la semana. | Para planificar mejor las actividades diarias. | El estudiante puede ver todos los eventos programados en una vista de calendario semanal. |
| H11 | Estudiante | Calcular la cantidad de comida, agua, luz y espacio necesarios. | Para optimizar recursos y asegurar el cuidado adecuado. | El estudiante puede ingresar la cantidad de animales o plantas, y la calculadora muestra la cantidad de recursos necesarios. |
| H12 | Estudiante | Editar o eliminar recordatorios previamente creados. | Para actualizar o corregir tareas. | El estudiante puede editar o eliminar recordatorios en el calendario. |
| H13 | Estudiante | Recibir alertas sobre recordatorios que están por vencer. | Para recordar que debe realizar una tarea. | El estudiante recibe alertas de tareas pendientes antes de que se venzan en el calendario. |
| H14 | Estudiante | Consultar el historial de eventos completados. | Para tener un registro de actividades realizadas. | El estudiante puede acceder al historial de eventos completados y ver detalles de cada tarea realizada. |
| H15 | Estudiante | Acceder a un listado de enfermedades comunes en plantas y animales. | Para identificar síntomas y tomar medidas preventivas. | El estudiante puede ver una lista de enfermedades comunes y acceder a la información detallada de cada una. |
| H16 | Estudiante | Acceder a una lista de consejos sobre el manejo de granjas. | Para mejorar sus habilidades en gestión. | El estudiante puede acceder a una lista de consejos y seleccionar uno para ver información detallada. |

### **3. Funcionalidades Asociadas a las Historias de Usuario**

A continuación, se mapean las funcionalidades desarrolladas para cubrir las historias de usuario:

#### **Historia de Usuario 1: Ver lista de especies**

* Funcionalidad para mostrar todas las especies de animales y plantas disponibles en la app.

#### **Historia de Usuario 2: Filtrar especies**

* Implementación de filtros para que los estudiantes puedan buscar animales o plantas por categoría.

#### **Historia de Usuario 3: Acceder a información detallada**

* Pantalla que muestra detalles completos de cada especie seleccionada.

#### **Historia de Usuario 4: Acceder a plagas**

* Apartado dedicado a las plagas asociadas a cada planta, incluyendo características y medidas de control.

#### **Historia de Usuario 5: Época de siembra**

* Información que muestra la época recomendada de siembra para cada planta.

#### **Historia de Usuario 6: Imágenes adicionales**

* Funcionalidad que permite al usuario ver varias imágenes de cada especie.

#### **Historia de Usuario 7: Recordatorios automáticos**

* Sistema de configuración para generar recordatorios automáticos para tareas comunes.

#### **Historia de Usuario 8: Recordatorios personalizados**

* Interfaz para que los usuarios ingresen recordatorios específicos.

#### **Historia de Usuario 9: Visualización de eventos**

* Pantalla que muestra todos los eventos programados en un calendario semanal.

#### **Historia de Usuario 10: Calcular necesidades**

* Calculadora que determina los recursos necesarios según la cantidad de animales o plantas.

#### **Historia de Usuario 11: Editar o eliminar recordatorios**

* Funcionalidad para que los usuarios editen o eliminen recordatorios en el calendario.

#### **Historia de Usuario 12: Alertas sobre recordatorios**

* Sistema de alertas que notifica a los usuarios sobre tareas pendientes.

#### **Historia de Usuario 13: Consultar historial**

* Pantalla para acceder al historial de eventos completados.

#### **Historia de Usuario 14: Listado de enfermedades**

* Funcionalidad que permite a los usuarios acceder a un listado de enfermedades comunes.

#### **Historia de Usuario 15: Consejos de manejo**

* Acceso a una lista de consejos sobre el manejo efectivo de la granja.

### **4. Estrategia de Pruebas**

#### **4.1 Pruebas Funcionales**

**Objetivo**: Validar que las funcionalidades implementadas cumplan con los criterios de aceptación de las historias de usuario.

**Áreas de Pruebas**:

* **Ver lista de especies**: Se verificará que el usuario pueda visualizar correctamente la lista de animales y plantas.
* **Filtrar especies**: Comprobar que los filtros se apliquen correctamente y que los resultados sean precisos.
* **Acceder a información detallada**: Validar que la información mostrada sea completa y clara.
* **Acceder a plagas**: Probar que los usuarios puedan ver información sobre plagas asociadas a cada planta.
* **Época de siembra**: Verificar que la información de siembra se muestre correctamente.
* **Imágenes adicionales**: Confirmar que se presenten varias imágenes de las especies.
* **Recordatorios automáticos**: Comprobar que se generen recordatorios según las configuraciones del usuario.
* **Recordatorios personalizados**: Validar que los recordatorios se creen y se guarden correctamente.
* **Visualización de eventos**: Asegurar que los eventos programados se muestren adecuadamente.
* **Calcular necesidades**: Probar que la calculadora entregue resultados precisos basados en la entrada del usuario.
* **Editar o eliminar recordatorios**: Verificar que el usuario pueda modificar y eliminar recordatorios.
* **Alertas sobre recordatorios**: Confirmar que se envíen alertas a tiempo.
* **Consultar historial**: Validar que el historial de eventos se muestre con detalles precisos.
* **Listado de enfermedades**: Asegurar que la lista de enfermedades sea accesible y que cada entrada sea informativa.
* **Consejos de manejo**: Probar que los consejos se muestren adecuadamente y sean seleccionables para más detalles.

#### **4.2 Pruebas de Integración**

**Objetivo**: Asegurar que los diferentes módulos del sistema trabajen correctamente juntos.

**Áreas de Pruebas**:

* **Integración entre registros y recordatorios**: Verificar que la información registrada sobre animales y plantas se utilice para generar recordatorios.
* **Integración de la calculadora con datos de especies**: Asegurar que los datos de especies se reflejen correctamente en los cálculos de necesidades.

#### **4.3 Pruebas de Seguridad**

**Objetivo**: Verificar que la aplicación esté protegida contra vulnerabilidades y que la información de los usuarios esté asegurada.

**Áreas de Pruebas**:

* **Validación de datos de entrada**: Asegurar que todos los datos ingresados sean validados para prevenir vulnerabilidades como inyección SQL.
* **Cifrado de datos sensibles**: Verificar que las contraseñas y otros datos críticos se cifren adecuadamente.

#### **4.4 Pruebas de Rendimiento**

**Objetivo**: Evaluar el rendimiento del sistema bajo diferentes condiciones de carga.

**Áreas de Pruebas**:

* **Pruebas de carga**: Simular múltiples usuarios interactuando con la aplicación simultáneamente para validar la estabilidad.
* **Pruebas de estrés**: Evaluar cómo se comporta la aplicación bajo condiciones de carga extremas.

### **5. Criterios de Salida**

Un ciclo de pruebas se considerará exitoso si se cumplen los siguientes criterios:

* Todas las pruebas funcionales se completan sin errores críticos.
* Se han corregido los defectos de seguridad detectados.
* Los resultados de las pruebas de rendimiento están dentro de los límites aceptables.
* La integración entre los módulos clave funciona sin problemas.

### **6. Recomendaciones**

* **Automatización de Pruebas**: Se recomienda implementar pruebas automatizadas para facilitar las pruebas de regresión y asegurar la consistencia en cada ciclo de desarrollo.
* **Revisión Continua**: Debido a la evolución de los requisitos del proyecto, el plan de pruebas debe revisarse periódicamente para ajustarse a cualquier cambio en las historias de usuario o en el Product Backlog.