UNIVERSIDAD TECNOLÓGICA DEL VALLE DEL MEZQUITAL

ING. DESARROLLO Y GESTIÓN DE SOFTWARE

Desarrollo Móvil Integral

DOCUMENTACIÓN FINAL VIRENS

Ángeles Muthe Luis Antonio de Jesús
Boxtha Martín Jessica
Cruz Jiménez Iván
Hernández Ávila Yareli
Morgado Abreu Isaac
Pérez Juárez Alison Yuridia
Rodríguez Corona Alelí Geraldine

10°B

ING. GARDENIA CRUZ ESCUDERO

23 de septiembre del 2025

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. Metodología Ágil

Scrum es un marco ágil para la gestión de proyectos que se centra en entregar incrementos de producto de manera iterativa y colaborativa. Divide el trabajo en ciclos cortos llamados sprints (de 2 a 4 semanas), en los cuales el equipo desarrolla, prueba y entrega funcionalidades completas.

Principios básicos de Scrum

- Transparencia: Toda la información del proyecto debe ser visible para todos los involucrados (product backlog, avance, impedimentos).
- Inspección: El equipo revisa regularmente el progreso y la calidad del trabajo.
- Adaptación: Si algo no funciona o hay cambios en las necesidades del cliente, se ajusta el plan en el siguiente sprint.

Estructura de Scrum

- Sprints: ciclos cortos (normalmente de 2 a 4 semanas) en los que el equipo desarrolla un incremento funcional del producto.
- Product Backlog: lista priorizada de todo lo que se debe implementar en el producto (épicas, features, historias de usuario).
- Sprint Backlog: conjunto de tareas seleccionadas del product backlog que se van a trabajar en un sprint.
- Incremento: resultado potencialmente entregable al final de cada sprint.

Justificación

- Permite adaptarse fácilmente a cambios en los requisitos del cliente o usuario.
- Promueve la colaboración continua entre el equipo.
- Facilita entregas frecuentes y funcionales de la aplicación, garantizando valor temprano.
- Se ajusta a proyectos donde la retroalimentación del usuario es esencial, como en el cuidado de plantas con necesidades personalizadas.

2. Definición de roles

Tabla 1 Definición de roles

Rol	Descripción	Responsable
Scrum Master	Garantiza que se cumpla el marco Scrum, elimina impedimentos y facilita la comunicación.	Rodriguez Corona Aleli Geraldine
Developers	Desarrollan las funcionalidades técnicas de la aplicación (frontend, backend, base de datos, integración de servicios).	Pérez Juárez Alison Yuridia
Developers		Hernández Ávila Yareli
Developers		Cruz Jiménez Iván
Developers		Ángeles Muthe Luis Antonio de Jesús
Documentador	Encargado de mantener la documentación actualizada (arquitectura, historias de usuario, manuales técnicos y de usuario).	Morgado Abreu Isaac
Tester	Verifica la calidad del producto, ejecuta pruebas y valida que las funcionalidades cumplan con los criterios de aceptación.	Boxtha Martin Jessica

3. Historias de Usuario

Se reúnen las épicas, features e historias de usuario del proyecto Virens, con criterios de aceptación, prioridad y asignación de sprint.

Feature: Registro e Inicio de Sesión

Tabla 2 Feature_Registro e Inicio de Sesión

ID	Historia de Usuario	Criterios de Aceptación	Prioridad	Feature/Épica	Sprint
HU-ACC-01	Como usuario, quiero registrarme en la app con correo y contraseña, para tener una cuenta personalizada.	Validar correo único. Solicitar contraseña segura. Guardar datos en la base de usuarios.	Alta	Registro e Inicio de Sesión	Sprint 0
HU-ACC-02	Como usuario, quiero iniciar sesión con mis credenciales, para acceder a mis plantas y configuraciones.	Verificar correo y contraseña. Mostrar error si credenciales son inválidas. Acceder al dashboard al iniciar sesión.	Alta	Registro e Inicio de Sesión	Sprint 0

Feature: Inicio

Tabla 3 Feature_Inicio

ID	Historia de Usuario	Criterios de Aceptación	Prioridad	Feature/Épica	Sprint
HU-IN-01	Como usuario, quiero visualizar un mensaje de bienvenida personalizado, para sentir mayor conexión con la app.	Mostrar nombre del usuario en la bienvenida. Actualizar mensaje al iniciar sesión.	Baja	Inicio	Sprint 1

Feature: Catálogo

Tabla 4 Feature_Catálogo

ID	Historia de Usuario	Criterios de Aceptación	Prioridad	Feature/Épica	Sprint
HU-CAT-01	Como usuario, quiero explorar un catálogo de especies de plantas.	Mostrar listado de especies con foto. Permitir búsqueda por nombre. Mostrar información de cuidados.	Alta	Catálogo	Sprint 2
HU-CAT-02	Como usuario, quiero filtrar el catálogo por tipo de planta, para encontrar rápido lo que necesito.	Mostrar filtros por categoría. Aplicar filtros correctamente. Actualizar lista en tiempo real.	Media	Catálogo	Sprint 2

Feature: Registro de Plantas

Tabla 5 Feature_Registro de Plantas

ID	Historia de Usuario	Criterios de Aceptación	Prioridad	Feature/Épica	Sprint
HU-01	Como usuario, quiero registrar una planta con nombre, foto y especie, para poder identificarla fácilmente.	Permitir tomar o subir una foto. Seleccionar especie desde lista Guardar y mostrar planta en la lista general.	Alta	Gestión de Plantas / Registro de Planta	Sprint 1
HU-02	Como usuario, quiero editar la información de mis plantas registradas, para mantener sus datos actualizados.	Permitir modificar nombre, foto y especie. Actualizar correctamente la base de datos. Mostrar cambios reflejados en la lista de plantas.	Media	Gestión de Plantas / Edición de Planta	Sprint 1

Feature: Perfil Planta

Tabla 6 Feature_Perfil Planta

ID	Historia de Usuario	Criterios de Aceptación	Prioridad	Feature/Épica	Sprint
HU-PP-01	Como usuario, quiero ver la ficha detallada de cada planta registrada, para consultar sus cuidados específicos.	Mostrar foto, nombre y especie. Mostrar calendario asociado. Mostrar consejos de salud.	Alta	Perfil Planta	Sprint 4

Feature: Historial de notificaciones

Tabla 7 Feature_Historial de notificaciones

ID	Historia de Usuario	Criterios de Aceptación	Prioridad	Feature/Épica	Sprint
HU-HN-01	Como usuario, quiero consultar las notificaciones pasadas de riego y fertilización, para revisar lo que ya hice.	Mostrar lista cronológica de notificaciones. Indicar fecha, hora y tarea asociada. Permitir filtrar por planta.	Media	Historial de Notificaciones	Sprint 6

Feature: Mi perfil

Tabla 8 Feature_Mi perfil

ID	Historia de Usuario	Criterios de Aceptación	Prioridad	Feature/Épica	Sprint
HU-MP-01	Como usuario, quiero ver y actualizar mis datos de perfil (nombre, correo, foto), para mantener mi cuenta al día.	Mostrar datos actuales del perfil. Permitir edición de nombre, correo y foto. Guardar cambios en la base de datos.	Media	Mi Perfil	Sprint 7

Feature: Configuración

Tabla 9 Feature_Configuración

ID	Historia de Usuario	Criterios de Aceptación	Prioridad	Feature/Épica	Sprint
HU-CONF-01	Como usuario, quiero configurar la hora en la que recibiré notificaciones, para que se ajuste a mi rutina.	Permitir elegir hora en la configuración. Guardar preferencia por usuario. Aplicar configuración a todas las notificaciones.	Media	Configuración	Sprint 7

4. Arquitectura MVC

MVC se usa inicialmente en sistemas donde se requiere el uso de interfaces de usuario. Su fundamento es la **separación del código en tres capas diferentes**, acotadas por su responsabilidad, en lo que se llaman **Modelos, Vistas y Controladores**:

1. **Model (Modelo)**

- Es la capa encargada de los datos y la lógica de negocio.
- Gestiona la base de datos, las reglas de negocio y las operaciones sobre los datos.
- No sabe nada de cómo se muestran esos datos al usuario.

2. **View (Vista)**

- Es la interfaz gráfica del usuario (UI).
- Muestra la información que viene del modelo y recibe las interacciones del usuario.
- Debe ser lo más ligera posible: solo presenta la información.

3. **Controller (Controlador)**

- Es el intermediario entre la vista y el modelo.
- Recibe las acciones del usuario desde la vista (ejemplo: dar clic en “Agregar usuario”), las procesa, pide al modelo la información necesaria y devuelve los resultados a la vista.

Aplicación de la arquitectura MVC en Virens

Model (Modelo)

- Representa los datos y la lógica de negocio.
- En Virens se encarga de definir la estructura de la información y las reglas que la gobiernan.

Ejemplos en Virens:

- Tablas de base de datos:
  - **usuarios** (datos de login, perfil).
  - **plantas** (nombre, especie, foto, luz, temperatura).
  - **cuidados** (fechas de riego, fertilización, notas, historial).
  - **notificaciones** (pendientes, enviadas).
- Lógica de negocio:
  - Calcular automáticamente la frecuencia de riego según la especie.
  - Validar horarios configurados para notificaciones.
  - Generar historial de cuidados ordenado por fecha.

View (Vista)

- Representa la interfaz gráfica que el usuario ve e interactúa.
- En Virens son las pantallas de la app móvil.

Ejemplos en Virens:

- Pantalla de Inicio → muestra próximas tareas de riego y fertilización.
- Pantalla de Calendario de cuidados → vista mensual con eventos de cada planta.
- Pantalla de Perfil de planta → ficha con foto, especie, cuidados, historial.
- Pantalla de Recordatorios → lista de notificaciones y configuración de horarios.
- Pantalla de Registro e inicio de sesión → acceso de usuarios.

Controller (Controlador)

- Maneja la lógica de aplicación y conecta la vista con el modelo.
- Recibe acciones del usuario (clics, registros, ediciones) desde la Vista, y actualiza el Modelo.

Ejemplos en Virens (métodos del controlador):

- *registrarPlanta*() → recibe datos desde la vista (foto, especie), valida y guarda en la base de datos (Modelo).
- *enviarRecordatorio*() → programa y dispara notificaciones basadas en el historial y configuraciones.
- *iniciarSesion*() → validas credenciales y carga los datos del usuario.
- *actualizarPerfil*() → modifica los datos del usuario en la base de datos y refleja el cambio en la vista.

Ejemplo en Virens:

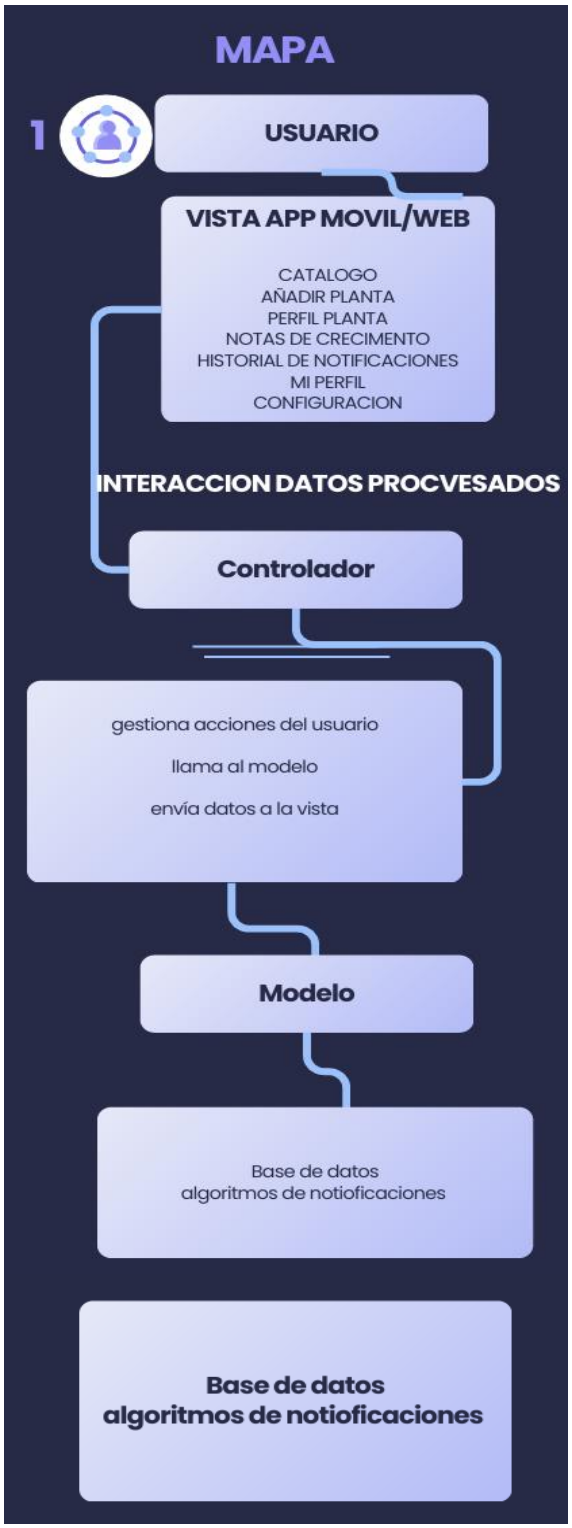

Figura 1 Arquitectura MVC

5. Sugerencia de patrones de diseño aplicables.

Se han evaluado dos opciones para Virens:

1. MVC: para que la lógica de negocio y la interfaz gráfica evolucionen por separado, así se facilita la colaboración entre equipo de desarrollo y diseño.
2. MVVM: usa un ViewModel que interactúa con el Model y notifica a la View de los cambios.

Elección de patrón de diseño:

MVC (Model–View–Controller) es una excelente elección para **Virens** porque ofrece una separación clara entre datos/lógica, interfaz y las acciones que los conectan. Esa separación facilita mantenimiento, pruebas, trabajo en equipo y escalado, todo crítico para la app que maneja datos dinámicos (plantas, recordatorios).

Beneficios:

- Separación de responsabilidades.
- Mantenibilidad y escalabilidad.
- Trabajo en equipo paralelo.
- Tests unitarios sobre Models.
- Seguridad y validación centralizada.
- Estado offline/sincronización.

Ejemplo de uso:

1. Usuario rellena formulario en la View y pulsa “Guardar”.
2. View envía datos a PlantaController.registrarPlanta(data).
3. Controller valida campos mínimos. Si falla, devuelve error a View.
4. Si OK, Controller llama a PlantaRepository.save(plantaDTO).
5. Repository/Model persiste en BD y calcula próximas fechas de riego.
6. Repository devuelve entidad guardada al Controller.
7. Controller solicita al servicio de notificaciones programar recordatorios.
8. Controller indica a la View actualizar la lista (por Observer o callback).
9. View refresca y muestra confirmación.

6. Argumentación de los frameworks de desarrollo móvil a emplear.

Se han evaluado dos opciones:

1. Flutter: por su excelente rendimiento, sistema de widgets personalizable, soporte multiplataforma.
2. React Native: ecosistema maduro, soporte de JS, integración sencilla con apps nativas.

Elección del framework:

Flutter ofrece una integración completa entre UI y lógica, con un sistema de **widgets altamente personalizable**, que permite construir interfaces dinámicas, atractivas y consistentes en todas las plataformas. Esto facilita la creación de las vistas (Views) de la aplicación, como la pantalla de perfil de planta, calendario de cuidados o recordatorios, asegurando que la experiencia de usuario sea fluida y uniforme.

Beneficios:

- Desarrollo rápido y eficiente con un solo código base para Android e iOS.
- Interfaces personalizadas, fluidas y consistentes con experiencia nativa.
- Integración sencilla con APIs y servicios backend para lógica de negocio y notificaciones.
- Compatibilidad con arquitectura MVC, separando claramente View, Controller y Model.
- Ecosistema en crecimiento con soporte activo y documentación extensa.
- Posibilidad de realizar tests unitarios y de integración fácilmente.

Ejemplo de uso en Virens:

1. Usuario interactúa con la pantalla de Registro (Widget de View).
2. View envía los datos a UsuarioController.registrarUsuario(data).
3. Controller valida la información y llama al Model (UsuarioRepository.save(usuarioDTO)).

4. Model persiste los datos y calcula configuraciones iniciales, como horarios de riego.
5. Controller notifica a la View sobre la operación exitosa.
6. View se actualiza automáticamente mostrando la confirmación al usuario.

7. Diseño del plan de pruebas

Objetivo:

Definir la estrategia y alcance de las pruebas para garantizar la calidad de la aplicación antes de su entrega.

Alcance:

Partes de la aplicación que se probarán:

- Registro de usuarios
- Login y autenticación
- Creación, edición y eliminación de elementos del menú
- Interfaz de usuario en distintos dispositivos
- Rendimiento básico

Tabla 10 Pruebas en apps móviles

Tipo de Prueba	¿Cuándo se Usa?	¿Por qué es Importante en Apps Móviles?
Pruebas Unitarias	Durante el desarrollo de cada función o componente.	Detecta errores temprano en la lógica del código y asegura que cada módulo funcione de manera independiente.

Pruebas de Integración	Al unir varios módulos	Garantiza que los módulos trabajen bien juntos y que el flujo de datos sea correcto.
Pruebas de Interfaz (UI/UX)	Antes de liberar una versión de prueba o beta.	Verifica que la app sea intuitiva, accesible y que la interfaz responda en diferentes dispositivos.
Pruebas de Rendimiento	Antes de la entrega final, en condiciones de uso real.	Evalúa velocidad, consumo de memoria y estabilidad en dispositivos de gama baja.
Pruebas de Usabilidad	Durante pruebas piloto con usuarios reales.	Confirma que los usuarios pueden cumplir tareas sin confusión y que la app es amigable.
Pruebas de Compatibilidad	Antes del lanzamiento oficial.	Asegura que funcione en distintas versiones de Android/iOS y tamaños de pantalla.
Pruebas de Regresión	Cada vez que se hace un cambio importante en el código.	Evita que se rompan funcionalidades previamente correctas.

Tipos de Pruebas Para Aplicar a Virens

Tabla 11 Pruebas en Virens

Tipo de prueba	¿Cuándo se usa?	¿Por qué es importante en Virens?
Pruebas Unitarias	Durante el desarrollo de cada módulo	Garantizan que cada componente funcione de forma aislada
Pruebas de Integración	Al unir varios módulos	Detectan errores en la comunicación entre componentes

Pruebas de UI/UX	Cuando las pantallas ya están listas	Aseguran que la app sea intuitiva y usable
Pruebas de Regresión	Después de corregir errores	Verifican que los cambios no afecten funciones anteriores

Roles y Responsables

- **Tester (Jessica Boxtha Martín):** Ejecuta pruebas, documenta resultados y reporta errores.
- **Desarrolladores (Alison, Iván, Luis):** Corrigen errores reportados y ejecutan pruebas unitarias.
- **Scrum Master (Geraldine):** Supervisa que los bugs se prioricen y se atiendan en cada sprint.
- **Documentador (Isaac):** Registra resultados en plan de pruebas y genera reporte final.

Gestión de incidencias y seguimiento de errores

La **gestión de incidencias y errores** es el proceso de **registrar, priorizar, asignar, dar seguimiento y cerrar** errores detectados durante el desarrollo y las pruebas de software.

Un buen sistema de gestión de incidencias permite:

- Identificar los errores de forma clara.
- Priorizar los más críticos primero.
- Asignar responsables de corregirlos.
- Documentar el proceso de solución.
- Mantener trazabilidad (quién lo reportó, quién lo resolvió, cuándo).

Herramientas más usadas

Tabla 12 Herramientas de Gestión de incidencias

Herramienta	Características
Jira	Ideal para equipos grandes. Soporta flujos de trabajo avanzados, priorización de incidencias, integración con repositorios y Scrum/Kanban.
Trello	Simple y visual. Útil para registrar errores en tableros Kanban con listas de "Pendiente – En progreso – Resuelto". Ideal para equipos pequeños.
GitHub Issues	Integrado con repositorios de código. Permite asociar incidencias a commits, pull requests y milestones. Perfecto si el proyecto está en GitHub.
ClickUp/Asana	Plataformas de gestión de tareas con soporte para incidencias, calendarios y notificaciones.
Bugzilla	Herramienta especializada solo en gestión de bugs, muy usada en proyectos de código abierto.

Gestión de Incidencias y Bugs para Virens

Para **Virens**, al ser un proyecto de equipo pequeño y con entrega en menos de dos meses, se recomienda un flujo **ágil y visual** para el control de errores.

Herramientas Elegidas

- **Trello**: seguimiento visual en un tablero Kanban:
    - **Listas:** Pendiente – En Progreso – En Pruebas – Resuelto
    - **Tarjetas:** Cada bug es una tarjeta con título, descripción, prioridad, responsable y fecha límite.

- **GitHub Issues**: registro técnico de errores vinculados al código.
    - Si el bug se relaciona con un commit, se vincula el Issue en GitHub para tener trazabilidad.

Flujo de Trabajo Propuesto

1. **Registro del Bug**
    - El tester (Jessica) documenta el error en Trello (tarjeta nueva).
    - Agrega prioridad (Alta, Media, Baja), descripción clara, pasos para reproducirlo y capturas.
2. **Asignación**
    - El Scrum Master (Aleli) asigna el bug a un desarrollador (Alison, Iván o Luis).
3. **Corrección**
    - El desarrollador actualiza el estado en Trello a “En Progreso” mientras lo corrige.
    - Si el bug está relacionado con código, abre un Issue en GitHub y lo vincula al commit de la solución.
4. **Prueba de Validación**
    - Jessica prueba nuevamente el flujo afectado.
    - Si pasa la prueba, cambia el estado a “Resuelto”.
    - Si falla, se regresa a “Pendiente” con nuevas observaciones.
5. **Cierre**
    - Una vez aprobado, el Scrum Master marca la tarjeta como completada y la mueve a “Cerrado”.

Plan de pruebas

Tabla 13 Plan de Pruebas

ID	Nombre de la Prueba	Objetivo	Requisitos Previos	Pasos para Ejecutar	Resultado Esperado	Estado (aprobada / falllida)	Observaciones
TC-01	Registro de Usuario	Validar que un usuario pueda registrarse correctamente.	App instalada, conexión a internet.	1. Abrir app. 2. Seleccionar "Registrarse". 3. Llenar formulario con datos válidos. 4. Presionar "Crear cuenta".	Usuario creado, mensaje de confirmación mostrado, redirección a pantalla de inicio.		
TC-02	Inicio de Sesión	Verificar que un usuario registrado pueda iniciar sesión.	Usuario creado en la base de datos.	1. Abrir app. 2. Ingresar email y contraseña válidos. 3. Presionar "Iniciar sesión".	Acceso permitido y redirección a pantalla principal.		

TC-03	Validación de Sesión Incorrecta	Asegurar que no se pueda acceder con datos inválidos.	Usuario inexistente o contraseña incorrecta.	1. Abrir app. 2. Ingresar email o contraseña incorrectos. 3. Presionar "Iniciar sesión".	Mensaje de error: "Usuario o contraseña incorrectos".		
TC-04	Agregar Planta	Confirmar que se puede registrar una nueva planta.	Usuario logueado.	1. Ir al menú "Mis Plantas". 2. Presionar "Agregar Planta". 3. Llenar datos de planta. 4. Guardar.	Planta aparece en la lista de plantas.		
TC-05	Editar Planta	Verificar que se puede modificar información de una planta existente.	Tener al menos 1 planta registrada.	1. Ir a "Mis Plantas". 2. Seleccionar planta. 3. Editar datos y guardar.	Información actualizada correctamente.		

TC-06	Eliminar Planta	Validar que una planta pueda eliminarse.	Tener al menos 1 planta registrada.	1. Ir a "Mis Plantas". 2. Seleccionar planta. 3. Presionar "Eliminar".	Planta desaparece de la lista.		
TC-07	Generación de Calendario de Riego	Comprobar que se genera calendario automático.	Tener al menos 1 planta con datos de riego.	1. Ir a "Calendario". 2. Verificar que aparecen fechas de riego generadas.	Calendario muestra correctamente las fechas.		
TC-08	Recordatorio de Riego	Hay que asegurar que el sistema envíe notificaciones.	Tener recordatorios configurados en al menos 1 planta.	1. Configurar recordatorio. 2. Esperar a la hora configurada.	Notificación push enviada.		
TC-09	Historial de Cuidados	Validar que el historial se registre.	Tener al menos 1 cuidado registrado.	1. Ir a "Historial". 2. 2. Verificar que se muestran	Historial despliega información completa.		

				eventos de riego.			
TC-10	Cerrar Sesión	Verificar que el usuario pueda cerrar sesión.	Usuario logueado.	1. Ir a menú de usuario. 2. Seleccionar "Cerrar Sesión".	Se cierra sesión y regresa a pantalla de inicio de sesión.		

8. Estrategia para la gestión de versiones.

¿Cómo se manejan las versiones para el lanzamiento en App Store y Google Play?

1. Manejo de Versiones en App Store y Google Play

Tanto App Store (iOS) como Google Play (Android) requieren que cada nueva publicación tenga un número de versión único y un número de compilación mayor que el anterior.

La convención recomendada es el formato SemVer (Semantic Versioning):

Mayor.Menor.Parche (X.Y.Z)

- X (Mayor): cambios grandes que modifican la experiencia del usuario, rompen compatibilidad o introducen nuevas funcionalidades importantes.
- Y (Menor): nuevas características pequeñas o mejoras sin romper compatibilidad.
- Z (Parche): correcciones de errores, ajustes de rendimiento, pequeñas mejoras.

En App Store:

- El Número de Versión (Version Number) se muestra a los usuarios (ej. 1.2.0).
- El Número de Build (Build Number) debe incrementarse en cada carga, aunque la versión no cambie (ej. 10 → 11).

En Google Play:

- Se utiliza el Version Name (mostrado al usuario, ej. 1.2.0)
- Y el Version Code (entero que siempre debe incrementarse, ej. 10 → 11).

¿Qué prácticas se recomiendan para versiones beta, revisiones menores, y cambios mayores?

Tabla 14 Prácticas recomendadas para versiones Beta

Tipo de Versión	Prácticas Recomendadas	Aplicación en Virens
Versión Beta	• Publicar en TestFlight (iOS) o Google Play Internal Testing. • Liberar solo a probadores internos. • Incluir notas de versión detalladas para recibir feedback.	Pruebas internas del equipo antes del lanzamiento público. Validar registro de plantas, recordatorios y calendario.
Revisiones Menores (Y)	• Añadir funciones no disruptivas. • Hacer pruebas de regresión para asegurar que lo existente no se rompe. • Documentar cambios en changelog.	Añadir mejoras como: estadísticas de riego, temas de colores, consejos extra.
Cambios Mayores (X)	• Comunicar a los usuarios (notas de versión, emails). • Actualizar documentación y tutoriales. • Realizar pruebas exhaustivas de integración.	Gran actualización como rediseño de interfaz o incorporación de cuenta multiusuario.
Parches (Z)	• Liberar lo antes posible si el bug afecta la experiencia. • Mantener el changelog actualizado.	Corregir problemas con notificaciones, errores en formularios, fallos de carga de imágenes.

Bibliografía

Admin_Donetonic. (2024, 21 octubre). User Story Mapping, qué es y cómo crearlo. DoneTonic. https://donetonic.com/es/user-story-mapping/

Atlassian. (s. f.). ¿Qué es scrum? [+ Cómo empezar] | Atlassian. https://www.atlassian.com/es/agile/scrum

Qué es MVC. (s. f.). DesarrolloWeb.com. https://desarrolloweb.com/articulos/que-es-mvc.html

A, J. F. H. (s. f.). MVVC Android. Scribd. https://www.scribd.com/presentation/883979039/MVVC-Android

Bennett, L. (2025, 28 julio). MVC frente a MVVM – Diferencia entre ellos. Guru99. https://www.guru99.com/es/mvc-vs-mvvm.html

Flutter - Build apps for any screen. (s. f.-a). https://flutter.dev/?utm_source=google&utm_medium=cpc&utm_campaign=brand_sem&utm_content=latam_latam&gclsrc=aw.ds&gad_source=1&gad_campaignid=13034410696&gbraid=0AAAAAC-INI8iMBPvvkYeSSc62TtMyrLT0&gclid=Cj0KCQjw5onGBhDeARIsAFK6QJZ_juMHeRsYmeKeBU7ZGkk-YZXo0QIPCMILeYCcupotmifJkYTt7e8aAj5kEALw_wcB

Flutter - Build apps for any screen. (s. f.-b). https://flutter.dev/?utm_source=google&utm_medium=cpc&utm_campaign=brand_sem&utm_content=latam_latam&gclsrc=aw.ds&gad_source=1&gad_campaignid=13034410696&gbraid=0AAAAAC-INI8iMBPvvkYeSSc62TtMyrLT0&gclid=Cj0KCQjw5onGBhDeARIsAFK6QJZ_juMHeRsYmeKeBU7ZGkk-YZXo0QIPCMILeYCcupotmifJkYTt7e8aAj5kEALw_wcB

Qué es MVC. (s. f.). DesarrolloWeb.com. https://desarrolloweb.com/articulos/que-es-mvc.html