

**PROYECTO DE GRADO**

**APLICACIÓN MOVIL PARA EL CONTROL Y MONITOREO DE HUERTOS HIDROPONICOS**

**Raíces, Huertos Hidropónicos**

**POSTULANTE: Jorge Luis Yampara Manzaneda**

**Karen Quiroz Diaz**

**TUTOR: GRADO Y NOMBRE COMPLETO DEL TUTOR**

**Proyecto de Grado para optar al Grado Académico de Técnico**

**Superior en Sistemas Informáticos**

**COCHABAMBA- BOLIVIA**

**GESTIÓN 2022**

## RESUMEN

**TITULO: “APLICACIÓN MOVIL PARA EL CONTROL Y MONITOREO PARA EL CONTROL Y MONITOREO DE HUERTOS HIDROPONICOS.**

**RAICES, HUERTOS HIDROPONICOS”**

### AUTOR (ES): JORGE LUIS YAMPARA MANZANEDA

**KAREN QUIROZ DIAZ**

## PROBLEMATICA

DESCRIBIR BREVEMENTE LA PROBLEMÁTICA DEL PROYECTO A IMPLEMENTAR

## OBJETIVO GENERAL

Controlar de forma remota la hidroponía para interiores, que pueda dar los datos primordiales la solución nutritiva, la calidad del agua y manejar de forma remota incluso a la distancia

## CONTENIDO

La aplicación móvil “RAICES” esta dirigida para usuarios que cuenten con los huertos hidropónicos de la empresa “Raíces, productos hidropónicos”, con el objetivo principal de poder controlar y monitorear sus huertos hidropónicos.

Pensando en la fiabilidad de la aplicación se implementa la opción de inicio de sesión para guardar los datos del cliente y que ninguna otra persona ajena pueda acceder al control de los mismos.

La aplicación móvil ofrece el control y monitoreo a través de un teléfono móvil de:

- Medir la temperatura

- Medir el pH del agua

- Encender, apagar y programar las luces

- Encender, apagar y programar la mezcla de la bomba

- Ver tutoriales de la empresa

- Conexión con el personal de la empresa

- Notificaciones por parte de la aplicación

|  |  |  |
| --- | --- | --- |
| CARRERA |  | Sistemas Informáticos |
| PROFESOR GUIA |  | NOMBRE COMPLETO TUTOR |
| PERIODO DE INVESTIGACIÓN |  | EN NRO DE MESES |
| E – MAIL |  |  |
| CELULAR |  |  |

**DEDICATORIA**

En este apartado describe brevemente un homenaje muy especial a una o varias personas.

Ejemplo: “A mis padres, hijos y amigos, por el apoyo incondicional, porque sin ellos esto no hubiese sido posible”.

**AGRADECIMIENTOS**

Describe brevemente un agradecimiento a las instancias que colaboraron en la realización de tu proyecto, muestra gratitud a las personas que apoyaron incondicionalmente en tu proyecto, también puedes defender tu credo (es opcional).

**INDICE**

## CAPÍTULO I – MARCO REFERENCIAL

### TITULO DEL PROYECTO

### 1.1 INTRODUCCIÓN

La degradación de los suelos es una realidad, deteriorados gracias al agotamiento de nutrientes y otras amenazas directas, por estos motivos nace un nuevo tipo de agricultura que está tomando fuerza y se presenta como una opción para ayudar a combatir la escasez de suelos para la agricultura tradicional. La agricultura hidropónica es una forma de producción alimenticia que gracias a las nuevas tecnologías, está comenzando a popularizarse.

Al día de hoy, existen varias empresas reconocidas que están haciendo lo necesario y testeando este tipo de agricultura revolucionaria. La empresa de productos y servicios Hidroponicos Raices es una de ellas, la cual quiere implementar mejoras tecnologicas para el recopilando y analizis grandes volúmenes de información producida por cuantiosos sensores dispuestos estratégicamente en los invernaderos. De esta manera, y tras el análisis de los datos, es posible que los ventiladores, las unidades de calefacción y otros equipos sean controlados de manera remota para este tipo de agricultura.

El ambiente controlado hasta el más mínimo detalle, la muy baja o nula concentración de pesticidas, y su gran sostenibilidad son elementos que transforman a la agricultura hidropónica en una alternativa más que importante para el futuro próximo. Lo que se espera es que este tipo de técnicas e infraestructuras tecnológicas se generalice para otorgar una alternativa para ayudar y cuidar a nuestros suelos y así mismo poder intentar resolver los problemas de seguridad alimentaria en el mundo.

### 1.2 ANTECEDENTES

Los productos y servicios ofrecidos por raíces se realizan bajo un estándar de calidad. Estas prácticas permiten reforzar la confianza con los clientes y consumidores otorgándoles un precio justo. En cuanto a los productos comestibles, son cultivados sin el uso de pesticidas de origen químico para evitar el daño a la salud de las personas en general y el medio ambiente. Se puso a germinar las primeras semillas un 29 de Diciembre del 2018 usando el método de cultivo hidropónico. MISIÓN “Raíces” se destaca por brindar servicios para satisfacer la necesidad de productos y conocimientos hidropónicos al público general.

Los productos ofrecidos por raíces se realizan bajo la mayor calidad e inocuidad para satisfacer al consumidor. VISIÓN Llegar a ser una empresa consolidada a nivel departamental que otorgue servicios de asesoramiento, venta de productos y desarrollo de nuevas tecnologías para la mejora de cultivos hidropónicos. VALORES Los productos y servicios ofrecidos por raíces se realizan bajo un estándar de calidad. Estas prácticas nos permiten reforzar la confianza con los clientes y consumidores otorgándoles un precio justo.

En cuanto a los productos comestibles, son cultivados sin el uso de pesticidas de origen químico para evitar el daño a la salud de las personas en general y el medio ambiente. FILOSOFÍA Cada miembro de raíces es motivado por un sentido de unidad en el crecimiento de la empresa y bajo un entorno laboral de trato igualitario, al margen del respeto y amistad. Esto para ofrecer productos de calidad y siempre buscando la innovación en todos los procesos que conlleva la hidroponía de manera transparente

### 1.3 PLANTEAMIENTO Y FORMULACIÓN DEL PROBLEMA

**1.3.1 Descripción del Problema**

Los productos y servicios ofrecidos por raíces se realizan bajo un estándar de calidad. Estas prácticas permiten reforzar la confianza con los clientes y consumidores otorgándoles un precio justo. En cuanto a los productos comestibles, son cultivados sin el uso de pesticidas de origen químico para evitar el daño a la salud de las personas en general y el medio ambiente. Se puso a germinar las primeras semillas un 29 de Diciembre del 2018 usando el método de cultivo hidropónico.

MISIÓN

“Raíces” se destaca por brindar servicios para satisfacer la necesidad de productos y conocimientos hidropónicos al público general.Los productos ofrecidos por raíces se realizan bajo la mayor calidad e inocuidad para satisfacer al consumidor.

VISIÓN

Llegar a ser una empresa consolidada a nivel departamental que otorgue servicios de asesoramiento, venta de productos y desarrollo de nuevas tecnologías para la mejora de cultivos hidropónicos.

VALORES Los productos y servicios ofrecidos por raíces se realizan bajo un estándar de calidad. Estas prácticas nos permiten reforzar la confianza con los clientes y consumidores otorgándoles un precio justo. En cuanto a los productos comestibles, son cultivados sin el uso de pesticidas de origen químico para evitar el daño a la salud de las personas en general y el medio ambiente.

FILOSOFÍA Cada miembro de raíces es motivado por un sentido de unidad en el crecimiento de la empresa y bajo un entorno laboral de trato igualitario, al margen del respeto y amistad. Esto para ofrecer productos de calidad y siempre buscando la innovación en todos los procesos que conlleva la hidroponía de manera transparente

**1.3.2 Identificación del Problema**

**FALTA**

**1.3.3 Formulación del Problema**

La aplicación “Raíces” podrá recibir los datos del nivel de pH, Temperatura y estado de las bombas de agua, ¿al mismo tiempo que enviará notificaciones si se identifica algún problema como también podrá realizar riego de forma remota?

### 1.4 OBJETIVOS

**1.4.1 Objetivo General**

Controlar de forma remota la hidroponía para interiores, que pueda dar los datos primordiales la solución nutritiva, la calidad del agua y manejar de forma remota incluso a la distancia

**1.4.2 Objetivos Específicos**

Mantener control sobre el estado del huerto hidropónico mediante la implementación de sensores que controlen diferentes aspectos en el huerto hidropónico para que el ambiente sea óptimo para los cultivos Implementación de arduino que tiene todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador

**1.5 JUSTIFICACION**

**1.5.1 Justificación social**

La aplicación “Raíces” nos permitirá tener un gran control sobre el huerto hidropónico sin la necesidad de estar constantemente presente en el lugar, de esta manera se logrará una mayor eficiencia y teniendo mayor control de este, de esta manera se mejorará la calidad de cultivos y facilitando su proceso de cultivo y cosecha.

**1.5.2 Justificación técnica**

Todos los sensores primeramente serán configurados y conectados al Arduino para su correcto funcionamiento, una vez finalizada esa parte se instalará un módulo wi-fi para arduino el cual nos permitirá recibir y enviar información mediante conexión a internet y enviar los datos a la aplicación La aplicación se desarrollará para dispositivos Android, al mismo tiempo se utilizará la implementación de Arduino en el huerto hidropónico para ciertas funciones que tendrá a la aplicación, también se usara la instalación de sensores de temperatura, pH y sensores de nivel para medir la cantidad de agua.

**1.5.3 Justificación económica (solo si corresponde)**

**FALTA**

**1.6 ALCANCES**

• Encendido y apagado de las bombas de agua para el riego • Encendido y apagado de las luces led • Programar hora de riego • Mandar información constante de la temperatura • Mandar información constante del nivel de agua • Enviar notificaciones en caso de algún fallo • Apartado con tutoriales y contactos para usuarios de la empresa

**1.7 LIMITES**

• La aplicación no podrá modificar el pH • La aplicación no podrá arreglar errores solo enviar notificaciones de los mismos • La aplicación no rellenara el agua del tanque • La aplicación no se conectará de forma automática con cualquier tipo de sensores

### 1.8 ENFOQUE METODOLÓGICO

**1.8.1 Metodología de Desarrollo**

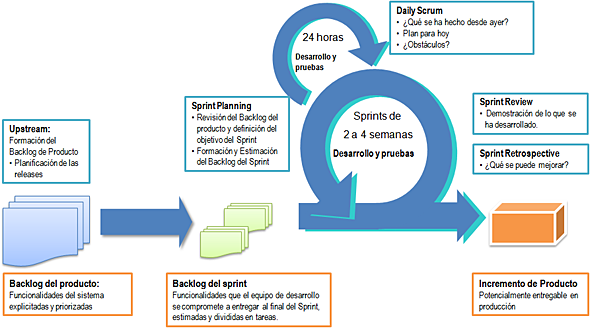
**1.8.1.1 INTRODUCCION**

Scrum es un marco de trabajo para desarrollo ágil de software que se ha expandido a otras industrias. Es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible de proyectos, caracterizado por:

• Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.

• Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.

• Solapar las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o en cascada

**1.8.1.2 CICLO DE VIDA DE SCRUM**

El ciclo de vida es el siguiente:

a. El Product Owner redacta las User Stories y las sitúa en el Product Backlog.

b. A continuación, el Product Owner prioriza estas User Stories y ordena el Product Backlog en consecuencia.

c. El equipo Scrum se junta en la reunión de planificación del Sprint, con el objetivo de establecer la lista de las User Stories que se tratarán durante el Sprint. Esto forma el Sprint Backlog y a continuación se descomponen en tareas por el equipo de desarrollo.

d. Entonces el Sprint puede comenzar con una iteración de 2, 3 o 4 semanas.

e. El equipo se reúne diariamente para realizar la Melé diaria.

f. Como consecuencia del Sprint, obtenemos un producto potencialmente entregable que forma parte de una demostración durante la revisión del Sprint.

g. El ciclo termina con la retrospectiva del Sprint.

Y a continuación, solo hay que repetir todo de nuevo.

**1.8.1.3 HISTORIA DE SCRUM**

Este modelo fue identificado y definido por Ikujiro Nonaka y Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica: Fuji-Xerox, Canon, Honda, NEC, Epson, Brother, 3M y Hewlett-Packard (Nonaka & Takeuchi, The New Product Development Game, 1986). En su estudio, Nonaka y Takeuchi compararon la nueva forma de trabajo en equipo, con el avance en formación de melé (scrum en inglés) de los jugadores de Rugby, a raíz de lo cual quedó acuñado el término “scrum” para referirse a ella.

Aunque esta forma de trabajo surgió en empresas de productos tecnológicos, es apropiada para cualquier tipo de proyecto con requisitos inestables y para los que requieren rapidez y flexibilidad, situaciones frecuentes en el desarrollo de determinados sistemas de software.

En 1995, Ken Schwaber presentó “Scrum Development Process” en OOPSLA 95 (Object-Oriented Programming Systems & Applications conference)(SCRUM Development Process), un marco de reglas para desarrollo de software, basado en los principios de Scrum, y que él había empleado en el desarrollo de Delphi, y Jeff Sutherland en su empresa Easel Corporation (compañía que en los macrojuegos de compras y fusiones, se integraría en VMARK, y luego en Informix y finalmente en Ascential Software Corporation)

**1.8.1.4 CARACTERISTICAS DE SCRUM**

Scrum es un marco de trabajo que define un conjunto de eventos, prácticas y roles,2 y que puede tomarse como conjunto base para definir el proceso de producción que usará un equipo de trabajo o dentro de un proyecto.

Los roles principales en Scrum son el Scrum Master, que procura facilitar la aplicación de Scrum y gestionar cambios, el Product Owner, que representa a los stakeholders (interesados externos o internos), y el Team (equipo) que ejecuta el desarrollo y demás elementos relacionados con él. Durante cada sprint, un periodo entre una y cuatro semanas (la magnitud es definida por el equipo y debe ser lo más corta posible), el equipo crea un incremento de software potencialmente entregable (utilizable).

El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar (PBI, Product Backlog Item). Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de Sprint Planning. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los da a conocer al equipo. Entonces, el equipo conversa con el Product Owner buscando la claridad y magnitud adecuadas (Cumpliendo el INVEST) para luego determinar la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint.4 Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint.5 Scrum permite la creación de equipos auto organizados impulsando la co-localización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.

La metodología se basa en:

• El desarrollo incremental de los requisitos del proyecto en bloques temporales cortos y fijos.

• Se da prioridad a lo que tiene más valor para el cliente.

• El equipo se sincroniza diariamente y se realizan las adaptaciones necesarias.

• Tras cada iteración (un mes o menos entre cada una) se muestra al cliente el resultado real obtenido, para que este tome las decisiones necesarias en relación a lo observado.

• Se le da la autoridad necesaria al equipo para poder cumplir los requisitos. • Fijar tiempos máximos para lograr objetivos.

• Equipos pequeños (de 3 a 9 personas cada uno).

**1.8.1.5 ROLES DE SCRUM**

Roles Principales

• Product Owner (o Propietario del producto)

El Product Owner se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio. El Product Owner ayuda al usuario a escribir las historias de usuario, las prioriza, y las coloca en el Product Backlog.

• Scrum Master (o Facilitador)

Es el responsable del cumplimiento de las reglas del marco scrum. Se asegura que estas son entendidas por la organización y de que se realiza el trabajo conforme a ellas. Elimina los obstáculos que impiden que se desarrolle el objetivo del sprint. Asesora y da la formación necesaria al propietario del producto y al equipo de desarrolladores.

• Desarrollador

Cada uno de los profesionales que realizan la entrega del incremento de producto generado en cada sprint (denominado incremento). Es recomendable un pequeño equipo de 3 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc). Roles Auxiliares Los roles auxiliares en los "equipos Scrums" son aquellos que no tienen un rol formal y no se involucran frecuentemente en el "proceso Scrum", sin embargo, deben ser tomados en cuenta. Un aspecto importante de una aproximación ágil es la práctica de involucrar en el proceso a los usuarios, expertos del negocio y otros interesados ("pandemoldes"). Es importante que esa gente participe y entregue retroalimentación con respecto a la salida del proceso a fin de revisar y planear cada sprint.

• Stakeholders (Clientes, Proveedores, Vendedores, etc) Son las personas que hacen posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su desarrollo. Solo participan directamente durante las revisiones del "sprint".

**1.8.1.6 ARTEFACTOS**

• Pila del producto (o product backlog) Registra y prioriza los requisitos desde el punto del vista del cliente. Empieza con una visión inicial del producto y crece y evoluciona durante el desarrollo del producto. Los requisitos suelen denominarse "historias de usuario"

• Pila del sprint (o sprint backlog) Registro de los requisitos desde el punto de vista de los desarrolladores. Es la lista de tareas que se deben realizar durante un sprint para lograr el incremento previsto.

• Incremento Resultado de cada sprint

**1.8.1.7 FLUJO DE TRABAJO**

**Sprint**

El Sprint es el período en el cual se lleva a cabo el trabajo en sí. Es recomendado que la duración de los sprints sea constante y definida por el equipo con base en su propia experiencia. Se puede comenzar con una duración de sprint en particular (2 o 3 semanas) e ir ajustándolo con base en el ritmo del equipo, aunque sin relajarlo demasiado.

Al final de cada sprint, el equipo deberá presentar los avances logrados, y el resultado obtenido es un producto que, potencialmente, se puede entregar al cliente.

Así mismo, se recomienda no agregar objetivos al sprint o sprint backlog a menos que su falta amenace al éxito del proyecto.

La constancia permite la concentración y mejora la productividad del equipo de trabajo. El tiempo mínimo de un Sprint es de dos (2) semanas y el máximo es de cuatro (4) semanas.

**Planificación de sprint**

• Al comienzo de un sprint, el equipo de scrum tiene un evento de planificación de sprint

• Uno de los objetivos de la reunión es identificar y comunicar cuánto del trabajo es probable que se realice durante el actual Sprint. Scrum diario También llamado Daily Standup. Cada día durante la iteración, tiene lugar una reunión de estado del proyecto. Su objetivo es que los miembros del equipo se mantengan actualizados unos a otros sobre el trabajo de cada uno desde el último standup, qué problemas han encontrado o prevén encontrar, y qué planean hacer.

• La reunión tiene una duración fija de entre 5 y 15 minutos.

• Se recomienda hacerla de pie para recordar que debe ser una reunión breve y centrada en su objetivo, sin divagaciones. Es obligatorio parar todo lo que se está haciendo para concentrarse en la reunión.

• Si se requiere ampliar un tema, se hará tras el Daily Standup, pero no se interrumpe la dinámica del Standup para elaborar una discusión.

• Se hace siempre a la misma hora y en el mismo lugar. Si falta alguien, no se pospone la reunión. Revisión de sprint Al final de un sprint, el equipo realiza dos eventos: la revisión del sprint y la retrospectiva del sprint.

En la reunión de revisión de sprint se presentan los trabajos completados y su duración no debería ser superior a 4 horas para un Sprint de 1 mes.

**Retrospectiva del sprint**

Después de cada sprint, se lleva a cabo una retrospectiva del sprint, en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado. El propósito de la retrospectiva es realizar una mejora continua de la implementación de Scrum. Esta reunión tiene un tiempo fijo de cuatro horas.

**1.8.1.8 DOCUMENTACION**

**Product backlog**

El product backlog se trata como un documento de alto nivel para todo el proyecto. Es el conjunto de todos los requisitos de proyecto, el cual contiene descripciones genéricas de funcionalidades deseables, priorizadas según su retorno sobre la inversión (ROI) . Representa el qué va a ser construido en su totalidad. Es abierto y solo puede ser modificado por el product owner. Contiene estimaciones realizadas a grandes rasgos, tanto del valor para el negocio, como del esfuerzo de desarrollo requerido. Esta estimación ayuda al product owner a ajustar la línea temporal (KEV) y, de manera limitada, la prioridad de las diferentes tareas. Por ejemplo, si dos características tienen el mismo valor de negocio la que requiera menor tiempo de desarrollo tendrá probablemente más prioridad, debido a que su ROI será más alto.

**Sprint backlog**

El sprint backlog es el subconjunto de requisitos que serán desarrollados durante el siguiente sprint. Al definir el sprint backlog, se describe el cómo el equipo va a implementar los requisitos durante el sprint. Por lo general los requisitos se subdividen en tareas, a las cuales se asignan ciertas horas de trabajo, pero ninguna tarea con una duración superior a 16 horas. Si una tarea es mayor de 16 horas, deberá ser dividida en otras menores. Las tareas en el sprint backlog nunca son asignadas, son tomadas por los miembros del equipo del modo que les parezca adecuado.

**Burn down chart**

La burn down chart es una gráfica mostrada públicamente que mide la cantidad de requisitos en el Backlog del proyecto pendientes al comienzo de cada Sprint. Dibujando una línea que conecte los puntos de todos los Sprints completados, podremos ver el progreso del proyecto. Lo normal es que esta línea sea descendente (en casos en que todo va bien en el sentido de que los requisitos están bien definidos desde el principio y no varían nunca) hasta llegar al eje horizontal, momento en el cual el proyecto se ha terminado (no hay más requisitos pendientes de ser completados en el Backlog). Si durante el proceso se añaden nuevos requisitos la recta tendrá pendiente ascendente en determinados segmentos, y si se modifican algunos requisitos la pendiente variará o incluso valdrá cero en algunos tramos.

**Definition of Done**

El Definition of Done es un documento con una serie de criterios comunes para determinar cuando una tarea está completamente hecha.

1.8.1.9. BENEFICIOS SCRUM

• Flexibilidad a cambios. Gran capacidad de reacción ante los cambiantes requerimientos generados por las necesidades del cliente o la evolución del mercado. El marco de trabajo está diseñado para adecuarse a las nuevas exigencias que implican proyectos complejos.

• Reducción del Time to Market. El cliente puede empezar a utilizar las características más importantes del proyecto antes de que esté completamente terminado.

• Mayor calidad del software. El trabajo metódico y la necesidad de obtener una versión de trabajo funcional después de cada iteración, ayuda a la obtención de un software de alta calidad.

• Mayor productividad. Se logra, entre otras razones, debido a la eliminación de la burocracia y la motivación del equipo proporcionado por el hecho de que pueden estructurarse de manera autónoma.

• Maximiza el retorno de la inversión (ROI). Creación de software solamente con las prestaciones que contribuyen a un mayor valor de negocio gracias a la priorización por retorno de inversión.

• Predicciones de tiempos. A través de este marco de trabajo se conoce la velocidad media del equipo por sprint, con lo que es posible estimar de manera fácil cuando se podrá hacer uso de una determinada funcionalidad que todavía está en el Backlog.

• Reducción de riesgos. El hecho de desarrollar, en primer lugar, las funcionalidades de mayor valor y de saber la velocidad a la que el equipo avanza en el proyecto, permite despejar riesgos efectivamente de manera anticipada.

**1.8.2 Técnicas de Recolección de Información**

**FALTA**

**1.8.3 Población del Objeto de Estudio**

**FALTA**

## CAPÍTULO II – MARCO TEORICO

### 2.1 INTRODUCCIÓN

Recopilacion de los antecedentes del proyecto, investigaciones previas respecto a la automatización, control, monitoreo, hidroponía, huertos hidropónicos, aplicaciones moviles y las herramientas que se utilizan en el desarrollo del proyecto, tomando en cuenta las consideraciones teóricas lo que se sustenta al proyecto

### 2.2 SISTEMA DE INFORMACIÓN

Falyta

### 2.3 HERRAMIENTAS PARA EL DESARROLLO DE SOFTWARE

**2.2.1 Herramientas para la implementación del FRONT-END**

**ANDROID STUDIO:**

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android y está basado en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android, como las siguientes:

• Un sistema de compilación flexible basado en Gradle

• Un emulador rápido y cargado de funciones

• Un entorno unificado donde puedes desarrollar para todos los dispositivos Android

• Aplicación de cambios para insertar cambios de código y recursos a la app en ejecución sin reiniciarla

• Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra

• Variedad de marcos de trabajo y herramientas de prueba

• Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros

• Compatibilidad con C++ y NDK

• Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine

Cada proyecto de Android Studio incluye uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

• Módulos de apps para Android

• Módulos de biblioteca

• Módulos de Google App Engine

Puedes ver todos los archivos de compilación en el nivel superior de Secuencias de comando de Gradle y cada módulo de app contiene las siguientes carpetas:

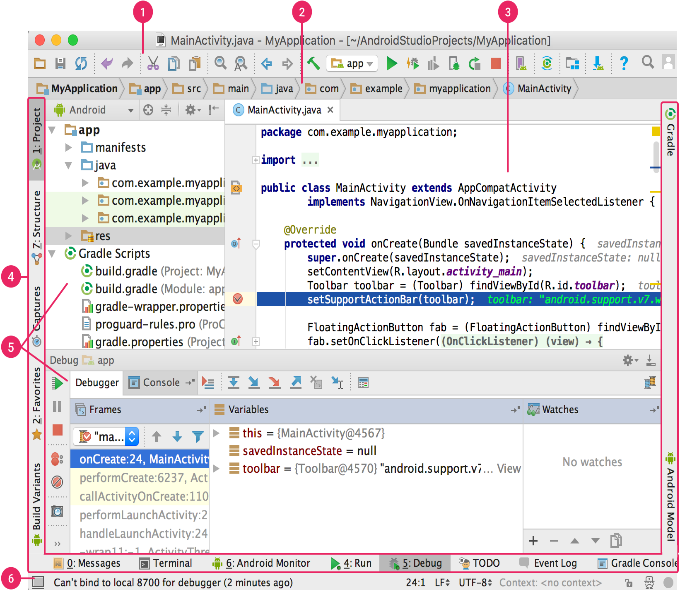
• Manifests: contiene el archivo AndroidManifest.xml.

• Java: contiene los archivos de código fuente Java, incluido el código de prueba de JUnit.

• Res: contiene todos los recursos sin código, como diseños XML, strings de IU e imágenes de mapa de bits.

La estructura del proyecto de Android en el disco difiere de esta representación plana. Para ver la estructura real de archivos del proyecto, selecciona Project en el menú desplegable Project (en la figura 1, se muestra como Android).

**Interfaz de usuario**



**Variantes de compilación**

El sistema de compilación puede ayudarte a crear diferentes versiones de la misma app a partir de un solo proyecto. Esto resulta útil cuando tienes una versión gratuita o una versión paga de tu app, o si quieres distribuir múltiples APK para diferentes configuraciones de dispositivos en Google Play.

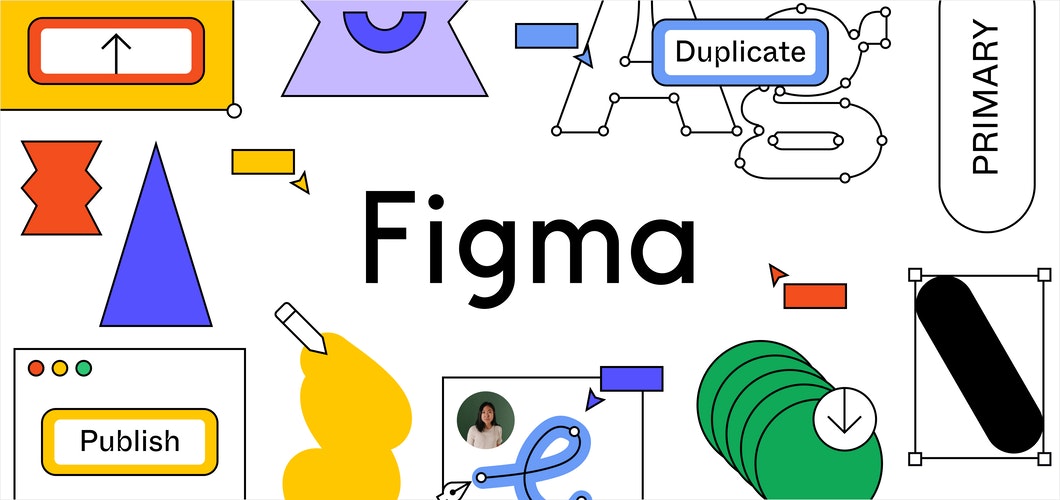
**Compatibilidad con varios APK**

La compatibilidad con varios APK te permite crear de manera eficiente varios APK basados en la densidad de la pantalla o en ABI. Por ejemplo, puedes crear APK individuales de una app para las densidades de pantalla hdpi y mdpi, y considerarlos una misma variante de modo que compartan la configuración de APK, javac, dx y ProGuard para la prueba.

**• FIGMA:**

Figma está especialmente pensada para el diseño de interfaces de usuario y experiencia de usuario, con énfasis en el trabajo colaborativo: ¡se puede diseñar con otras personas en tiempo real, en el mismo archivo! Con Figma podemos crear todo tipo de productos digitales: podemos diseñar prototipos para webs y apps, y también crear posts para redes sociales, e incluso exportar en formato PDF para imprimir

Es una herramienta ideal para bocetar antes de pasar a la fase de implementación o desarrollo de una web.

La comunidad Figma Figma tiene una comunidad de personas (los llamados creadores, Creators) que aportan gratuitamente archivos y plugins. Y los plugins son extensiones que amplían las funcionalidades de Figma: descargar imágenes e iconos para tus diseños, comprobar si dos colores tienen suficiente contraste, importar contenido de prueba, eliminar el fondo de las imágenes.

**2.2.2 Herramientas para la implementación del BACK-END**

**• BASE DE DATOS EN TIEMPO REAL:**

Firebase de Google es una plataforma en la nube para el desarrollo de aplicaciones web y móvil. Está disponible para distintas plataformas (iOS, Android y web), con lo que es más rápido trabajar en el desarrollo. Aunque fue creada en 2011 pasó a ser parte de Google en 2014, comenzando como una base de datos en tiempo real. Sin embargo, se añadieron más y más funciones que, en parte, permitieron agrupar los SDK de productos de Google con distintos fines, facilitando su uso.

**¿Para qué sirve Firebase?**

Su función esencial es hacer más sencilla la creación de tanto aplicaciones webs como móviles y su desarrollo, procurando que el trabajo sea más rápido, pero sin renunciar a la calidad requerida.

Sus herramientas son variadas y de fácil uso, considerando que su agrupación simplifica las tareas de gestión a una misma plataforma. Las finalidades de las mismas se pueden dividir en cuatro grupos: desarrollo, crecimiento, monetización y análisis. Es especialmente interesante para que los desarrolladores no necesiten dedicarle tanto tiempo al backend, tanto en cuestiones de desarrollo como de mantenimiento.

**Desarrollo**

El primer grupo de funciones es conocido como Desarrollo o Develop en Firebase. Como su nombre indica, incluye los servicios necesarios para el desarrollo de un proyecto de aplicación móvil o web. Estos contribuyen a que el proceso sea más rápido, puesto que se dejan determinadas actividades a mano de Firebase, mientras que otras permiten optimizar diversos aspectos para conseguir la calidad deseada.

**Realtime database**

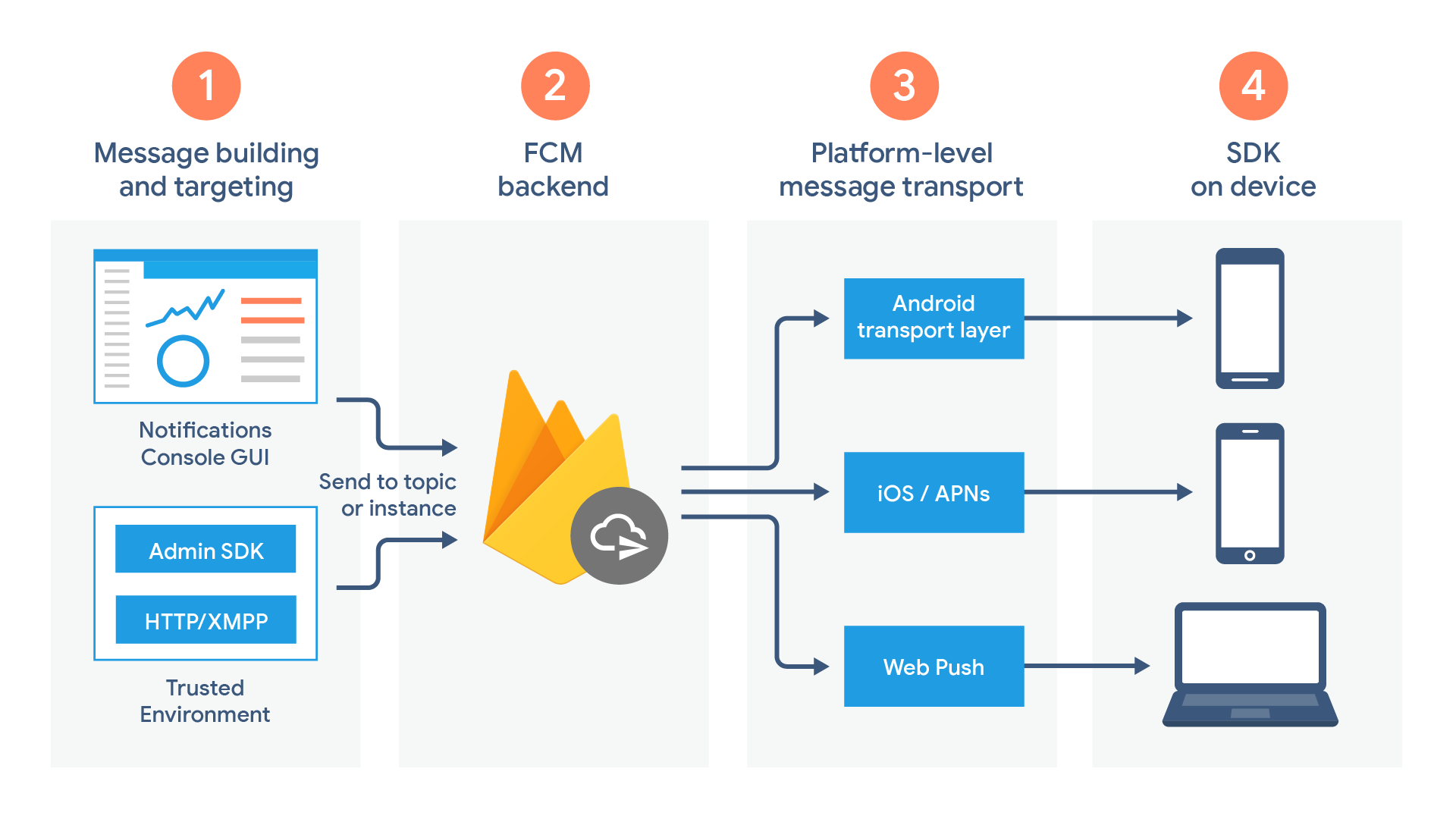
Una de las herramientas más destacadas y esenciales de Firebase son las bases de datos en tiempo real. Estas se alojan en la nube, son No SQL y almacenan los datos como JSON. Permiten alojar y disponer de los datos e información de la aplicación en tiempo real, manteniéndolos actualizados, aunque el usuario no realice ninguna acción. Firebase envía automáticamente eventos a las aplicaciones cuando los datos cambian, almacenando los datos nuevos en el disco. Aunque no hubiera conexión por parte de un usuario, sus datos estarían disponibles para el resto y los cambios realizados se sincronizarían una vez restablecida la conexión.

**Autenticación de usuarios**

La identificación de los usuarios de una app es necesaria en la mayoría de los casos si estos quieren acceder a todas sus características. Firebase ofrece un sistema de autenticación que permite tanto el registro propiamente dicho (mediante email y contraseña) como el acceso utilizando perfiles de otras plataformas externas (por ejemplo, de Facebook, Google o Twitter), una alternativa muy cómoda para usuarios reacios a completar el proceso. Así, este tipo de tareas se ven simplificadas, considerando también que desde aquí se gestionan los accesos y se consigue una mayor seguridad y protección de los datos. Se debe mencionar que Firebase puede guardar en la nube los datos de inicio de sesión con total seguridad, evitando que una persona tenga que identificarse cada vez que abra la aplicación.

**Almacenamiento en la nube**

Firebase cuenta con un sistema de almacenamiento, donde los desarrolladores pueden guardar los ficheros de sus aplicaciones (y vinculándolos con referencias a un árbol de ficheros para mejorar el rendimiento de la app) y sincronizarlos. Al igual que la mayoría de herramientas de Firebase, es personalizable mediante determinadas reglas. Este almacenamiento es de gran ayuda para tratar archivos de los usuarios (por ejemplo, fotografías que hayan subido), que se pueden servir de forma más rápida y fácil. También hace la descarga de referencias a ficheros más segura.



**• ARDUINO:**

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.

Para poder entender este concepto, primero hay que entender los conceptos de hardware libre y el software libre.

El **hardware libre** son los dispositivos cuyas especificaciones y diagramas son de acceso público, de manera que cualquiera puede replicarlos. Esto quiere decir que Arduino ofrece las bases para que cualquier otra persona o empresa pueda crear sus propias placas, pudiendo ser diferentes entre ellas, pero igualmente funcionales al partir de la misma base.

El **software libre** son los programas informáticos cuyo código es accesible por cualquiera para que quien quiera pueda utilizarlo y modificarlo. Arduino ofrece la plataforma Arduino IDE (Entorno de Desarrollo Integrado), que es un entorno de programación con el que cualquiera puede crear aplicaciones para las placas Arduino, de manera que se les puede dar todo tipo de utilidades.

**¿Cómo funciona Arduino?**

El Arduino es una placa basada en un microcontrolador ATMEL. Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, las cuales las escribes con el lenguaje de programación que puedes utilizar en el entorno Arduino IDE. Estas instrucciones permiten crear programas que interactúan con los circuitos de la placa.

El microcontrolador de Arduino posee lo que se llama una interfaz de entrada, que es una conexión en la que podemos conectar en la placa diferentes tipos de periféricos. La información de estos periféricos que conectes se trasladará al microcontrolador, el cual se encargará de procesar los datos que le lleguen a través de ellos.

El tipo de periféricos que puedas utilizar para enviar datos al microcontrolador depende en gran medida de qué uso le estés pensando dar. Pueden ser cámaras para obtener imágenes, teclados para introducir datos, o diferentes tipos de sensores.

También cuenta con una interfaz de salida, que es la que se encarga de llevar la información que se ha procesado en el Arduino a otros periféricos. Estos periféricos pueden ser pantallas o altavoces en los que reproducir los datos procesados, pero también pueden ser otras placas o controladores.

Además, las placas Arduino también cuentan con otro tipo de componentes llamados Escudos (Shields) o mochilas. Se trata de una especie de placas que se conectan a la placa principal para añadirle una infinidad de funciones, como GPS, relojes en tiempo real, conectividad por radio, pantallas táctiles LCD, placas de desarrollo, y un larguísimo etcétera de elementos. Incluso hay tiendas con secciones especializadas en dichos elementos.



**2.2.3 SGBD**

**Faltaaa**

**2.2.4 Otra herramienta adicional (ejemplo control de versiones u otro)**

**La descripción de las herramientas dentro de este apartado puede realizarse en base a: HISTORIA, CARACTERISTICAS, VENTAJAS, DESVENTAJAS, TABLA DE EVOLUCION DE LA HERRAMIENTA (si corresponde), aproximadamente de unas 20 a 30 páginas. Ya que pueden incluir imágenes, tablas, etc.)**

### 2.4 METODOLOGIAS ÁGILES

**2.4.1 Definición de metodologías ágiles**

Las **metodologías ágiles** son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

**2.4.2 Principios del Manifiesto ágil**

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. [Los procesos Ágiles](https://www.pragma.com.co/blog/que-es-el-agilismo-y-cuales-son-sus-beneficios) aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. [Los procesos Ágiles](https://www.pragma.com.co/blog/estrategia-y-cultura-agil)promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de [equipos autoorganizados.](https://www.pragma.com.co/blog/scrum-retrospectivas-enfocadas-en-equipos)
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

**2.4.3 Características de las metodologías ágiles**

* Basadas en heurísticas provenientes de prácticas de producción de código
* Especialmente preparados para cambios durante el proyecto
* Impuestas internamente (por el equipo)
* Proceso menos controlado, con pocos principios
* No existe contrato tradicional o al menos es bastante flexible
* El cliente es parte del equipo de desarrollo
* Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio
* Pocos artefactos
* Pocos roles
* Menos énfasis en la arquitectura del software

**2.4.4 Metodología ágiles vs. Metodología tradicionales**

En la metodología tradicional: priorizamos en base a la visión propia de la dirección del proyecto y no siempre en base a la visión del negocio, mientras que en las metodologías ágiles priorizamos más en base a las necesidades del cliente del proyecto en cada momento.

**2.5 METODOLOGIA SCRUM**

**2.5.1 Historia**

Este modelo fue identificado y definido por Ikujiro Nonaka y Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica: Fuji-Xerox, Canon, Honda, NEC, Epson, Brother, 3M y Hewlett-Packard (Nonaka & Takeuchi, The New Product Development Game, 1986).

En su estudio, Nonaka y Takeuchi compararon la nueva forma de trabajo en equipo, con el avance en formación de [melé](https://es.wikipedia.org/wiki/Scrum_(rugby)) (scrum en inglés) de los jugadores de Rugby, a raíz de lo cual quedó acuñado el término “scrum” para referirse a ella.

Aunque esta forma de trabajo surgió en empresas de productos tecnológicos, es apropiada para cualquier tipo de proyecto con requisitos inestables y para los que requieren rapidez y flexibilidad, situaciones frecuentes en el desarrollo de determinados sistemas de software.

En 1995, Ken Schwaber presentó “Scrum Development Process” en OOPSLA 95 (Object-Oriented Programming Systems & Applications conference)(SCRUM Development Process), un marco de reglas para desarrollo de software, basado en los principios de Scrum, y que él había empleado en el desarrollo de [Delphi](https://es.wikipedia.org/wiki/Embarcadero_Delphi), y Jeff Sutherland en su empresa Easel Corporation (compañía que en los macrojuegos de compras y fusiones, se integraría en VMARK, y luego en Informix y finalmente en Ascential Software Corporation)

**2.5.2 Forma de trabajo**

**Sprint**

El Sprint es el período en el cual se lleva a cabo el trabajo en sí. Es recomendado que la duración de los sprints sea constante y definida por el equipo con base en su propia experiencia. Se puede comenzar con una duración de sprint en particular (2 o 3 semanas) e ir ajustándolo con base en el ritmo del equipo, aunque sin relajarlo demasiado.

Al final de cada sprint, el equipo deberá presentar los avances logrados, y el resultado obtenido es un producto que, potencialmente, se puede entregar al cliente.[4](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)#cite_note-schwaber-4)​

Así mismo, se recomienda no agregar objetivos al sprint o *sprint backlog* a menos que su falta amenace al éxito del proyecto. La constancia permite la concentración y mejora la productividad del equipo de trabajo.

El tiempo mínimo de un Sprint es de dos (2) semanas y el máximo es de cuatro (4) semanas.

**Planificación de sprint**

Al comienzo de un sprint, el equipo de scrum tiene un evento de planificación de sprint

Uno de los objetivos de la reunión es identificar y comunicar cuánto del trabajo es probable que se realice durante el actual Sprint.

**Scrum diario**

También llamado Daily Standup. Cada día durante la iteración, tiene lugar una reunión de estado del proyecto. Su objetivo es que los miembros del equipo se mantengan actualizados unos a otros sobre el trabajo de cada uno desde el último standup, qué problemas han encontrado o prevén encontrar, y qué planean hacer.[4](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)#cite_note-schwaber-4)​

La reunión tiene una duración fija de entre 5 y 15 minutos.

Se recomienda hacerla de pie para recordar que debe ser una reunión breve y centrada en su objetivo, sin divagaciones. Es obligatorio parar todo lo que se está haciendo para concentrarse en la reunión.

Si se requiere ampliar un tema, se hará tras el Daily Standup, pero no se interrumpe la dinámica del Standup para elaborar una discusión.

Se hace siempre a la misma hora y en el mismo lugar. Si falta alguien, no se pospone la reunión.

**Revisión de sprint**

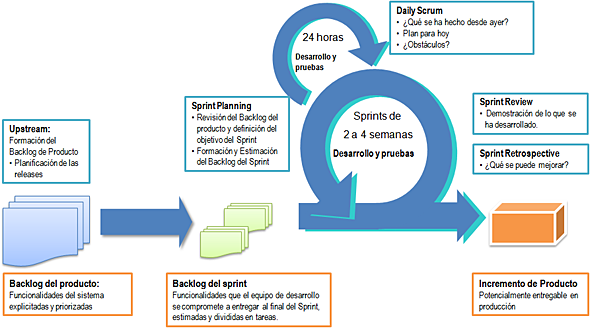
Al final de un sprint, el equipo realiza dos eventos: la revisión del sprint y la retrospectiva del sprint.

En la reunión de revisión de sprint se presentan los trabajos completados y su duración no debería ser superior a 4 horas para un Sprint de 1 mes.

**Retrospectiva del sprint**

Después de cada sprint, se lleva a cabo una retrospectiva del sprint, en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado. El propósito de la retrospectiva es realizar una mejora continua de la implementación de Scrum. Esta reunión tiene un tiempo fijo de cuatro horas.

**2.5.3 Ciclo de Vida**

****

El ciclo de vida es el siguiente:

* El Product Owner redacta las User Stories y las sitúa en el Product Backlog.
* A continuación, el Product Owner prioriza estas User Stories y ordena el Product Backlog en consecuencia.
* El equipo Scrum se junta en la reunión de planificación del Sprint, con el objetivo de establecer la lista de las User Stories que se tratarán durante el Sprint. Esto forma el Sprint Backlog y a continuación se descomponen en tareas por el equipo de desarrollo.
* Entonces el Sprint puede comenzar con una iteración de 2, 3 o 4 semanas.
* El equipo se reúne diariamente para realizar la Melé diaria.
* Como consecuencia del Sprint, obtenemos un producto potencialmente entregable que forma parte de una demostración durante la revisión del Sprint.
* El ciclo termina con la retrospectiva del Sprint.

Y a continuación, solo hay que repetir todo de nuevo.

**2.5.4 Roles de usuario**

* **Product Owner** (o Propietario del producto)
* **Scrum Master** (o Facilitador)
* **Desarrollador**
* **Roles Auxiliares**
* **Stakeholders** (Clientes, Proveedores, Vendedores, etc)

1. **Product Owner** (o Propietario del producto)

El *Product Owner* se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio. El Product Owner ayuda al usuario a escribir las [historias de usuario](https://es.wikipedia.org/wiki/Historias_de_usuario), las prioriza, y las coloca en el [Product Backlog](https://es.wikipedia.org/w/index.php?title=Scrum_(development)&action=edit&redlink=1" \o "Scrum (development) (aún no redactado)).

1. **Scrum Master** (o Facilitador)

Es el responsable del cumplimiento de las reglas del marco scrum. Se asegura que estas son entendidas por la organización y de que se realiza el trabajo conforme a ellas. Elimina los obstáculos que impiden que se desarrolle el objetivo del *sprint*. Asesora y da la formación necesaria al propietario del producto y al equipo de desarrolladores.

1. **Desarrollador**

Cada uno de los profesionales que realizan la entrega del incremento de producto generado en cada sprint (denominado incremento). Es recomendable un pequeño equipo de 3 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc).

1. **Roles Auxiliares**

Los roles auxiliares en los "equipos Scrums" son aquellos que no tienen un rol formal y no se involucran frecuentemente en el "proceso Scrum", sin embargo, deben ser tomados en cuenta. Un aspecto importante de una aproximación ágil es la práctica de involucrar en el proceso a los usuarios, expertos del negocio y otros interesados ("[pandemoldes](https://es.wikipedia.org/w/index.php?title=Pandemolde&action=edit&redlink=1" \o "Pandemolde (aún no redactado))"). Es importante que esa gente participe y entregue retroalimentación con respecto a la salida del proceso a fin de revisar y planear cada sprint.

1. **Stakeholders** (Clientes, Proveedores, Vendedores, etc)

Son las personas que hacen posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su desarrollo. Solo participan directamente durante las revisiones del "sprint".

**2.3.5 Características**

* Gestión regular de las expectativas del cliente, resultados anticipados, flexibilidad y adaptación, retorno de inversión, mitigación de riesgos, productividad y calidad, o, equipo motivado.
* Se hace uso de equipos auto-dirigidos y auto-organizados.
* Se realiza a diario una reunión de Scrum, que es una reunión de avance diaria que no dura más de 15 minutos con el objetivo de obtener realimentación sobre las tareas del equipo y los obstáculos que se presentan.
* Cada uno de estos puntos mencionados hacen que el Scrum sea utilizado de manera regular en un conjunto de buenas prácticas para el trabajo en equipo y de esa manera obtener resultados posibles.

**La metodología se basa en:**

* El desarrollo incremental de los requisitos del proyecto en bloques temporales cortos y fijos.
* Se da prioridad a lo que tiene más valor para el cliente.
* El equipo se sincroniza diariamente y se realizan las adaptaciones necesarias.
* Tras cada iteración (un mes o menos entre cada una) se muestra al cliente el resultado real obtenido, para que este tome las decisiones necesarias en relación a lo observado.
* Se le da la autoridad necesaria al equipo para poder cumplir los requisitos.
* Fijar tiempos máximos para lograr objetivos.
* Equipos pequeños (de 3 a 9 personas cada uno).

**2.3.6 Ventajas**

* **Flexibilidad a cambios.** Gran capacidad de reacción ante los cambiantes requerimientos generados por las necesidades del cliente o la evolución del mercado. El marco de trabajo está diseñado para adecuarse a las nuevas exigencias que implican proyectos complejos.
* **Reducción del Time to Market.** El cliente puede empezar a utilizar las características más importantes del proyecto antes de que esté completamente terminado.
* **Mayor calidad del software.** El trabajo metódico y la necesidad de obtener una versión de trabajo funcional después de cada iteración, ayuda a la obtención de un software de alta calidad.
* **Mayor productividad.** Se logra, entre otras razones, debido a la eliminación de la burocracia y la motivación del equipo proporcionado por el hecho de que pueden estructurarse de manera autónoma.
* **Maximiza el retorno de la inversión (ROI).** Creación de software solamente con las prestaciones que contribuyen a un mayor valor de negocio gracias a la priorización por retorno de inversión.
* **Predicciones de tiempos.** A través de este marco de trabajo se conoce la velocidad media del equipo por sprint, con lo que es posible estimar de manera fácil cuando se podrá hacer uso de una determinada funcionalidad que todavía está en el Backlog.
* **Reducción de riesgos.** El hecho de desarrollar, en primer lugar, las funcionalidades de mayor valor y de saber la velocidad a la que el equipo avanza en el proyecto, permite despejar riesgos efectivamente de manera anticipada.

**2.3.7 Desventajas**

* **Se aplica a equipos reducidos:**Scrum es exitosa cuando se trabaja con grupos de pocos colaboradores. En una empresa grande, por ejemplo, se debe sectorizar o dividir en grupos que cuenten con objetivos concretos. De lo contrario, el efecto de la técnica no será el mismo.
* **Requiere una exhaustiva definición de las tareas y sus plazos:**Scrum funciona correctamente cuando tanto las tareas como el tiempo en que se ejecutará cada una se encuentran definidos. La esencia de esta metodología reside en la división del trabajo de cada etapa y de sus tareas específicas.
* **Requiere de perfiles senior en su aplicación:**Quienes aplican Scrum cuentan con una alta cualificación, por lo que no es una modalidad de gestión propia de grupos junior o que estén en formación. Podemos decir que el éxito de Scrum radica en la experiencia profesional de equipos que cuentan con años de experiencia y se encuentran capacitados para estas exigencias. El éxito de un proyecto que trabaje con metodología Scrum dependerá en gran parte del liderazgo del Scrum Master.
* **Difícil escalabilidad:** Aplicar un enfoque Scrum para grandes proyectos se establece un reto ya que puede fallar la coordinación precisa, por lo que no garantiza que sea escalable a largo plazo.
* **Puede necesitar de transformaciones dentro de la organización:** En ocasiones, para trabajar con Scrum la empresa debe pasar por ciertas transformaciones organizativas en sus departamentos y áreas. Es la empresa quien debe gestionar y organizarse para que las colaboraciones sean exitosas.
* **No se integra fácilmente con enfoque clásico de gestión de proyectos:**El enfoque de Scrum no suele ser el adecuado para proyectos que requieren previsibilidad y un plan bien definido.
* En conclusión,**la metodología Scrum es una metodología ágil que hace énfasis en el trabajo en equipo** donde la claridad de los objetivos es crucial para avanzar hacia una versión cada vez mejor. El éxito de su aplicación dependerá, en gran medida, de contemplar estas ventajas y desventajas según lo requiera el proyecto.

**2.3.8 Artefactos**

**Pila del producto** (o product backlog)

Registra y prioriza los requisitos desde el punto del vista del cliente. Empieza con una visión inicial del producto y crece y evoluciona durante el desarrollo del producto. Los requisitos suelen denominarse "[historias de usuario](https://es.wikipedia.org/wiki/Historias_de_usuario)"

**Pila del sprint** (o sprint backlog)

Registro de los requisitos desde el punto de vista de los desarrolladores. Es la lista de tareas que se deben realizar durante un sprint para lograr el incremento previsto.

**Incremento**

Resultado de cada sprint.

**2.3.9 Flujo de trabajo**

**Sprint**

El Sprint es el período en el cual se lleva a cabo el trabajo en sí. Es recomendado que la duración de los sprints sea constante y definida por el equipo con base en su propia experiencia. Se puede comenzar con una duración de sprint en particular (2 o 3 semanas) e ir ajustándolo con base en el ritmo del equipo, aunque sin relajarlo demasiado.

Al final de cada sprint, el equipo deberá presentar los avances logrados, y el resultado obtenido es un producto que, potencialmente, se puede entregar al cliente.[4](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)#cite_note-schwaber-4)​

Así mismo, se recomienda no agregar objetivos al sprint o *sprint backlog* a menos que su falta amenace al éxito del proyecto. La constancia permite la concentración y mejora la productividad del equipo de trabajo.

El tiempo mínimo de un Sprint es de dos (2) semanas y el máximo es de cuatro (4) semanas.

**Planificación de sprint**

Al comienzo de un sprint, el equipo de scrum tiene un evento de planificación de sprint

Uno de los objetivos de la reunión es identificar y comunicar cuánto del trabajo es probable que se realice durante el actual Sprint.

Scrum diario

También llamado Daily Standup. Cada día durante la iteración, tiene lugar una reunión de estado del proyecto. Su objetivo es que los miembros del equipo se mantengan actualizados unos a otros sobre el trabajo de cada uno desde el último standup, qué problemas han encontrado o prevén encontrar, y qué planean hacer.[4](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)#cite_note-schwaber-4)​

La reunión tiene una duración fija de entre 5 y 15 minutos.

Se recomienda hacerla de pie para recordar que debe ser una reunión breve y centrada en su objetivo, sin divagaciones. Es obligatorio parar todo lo que se está haciendo para concentrarse en la reunión.

Si se requiere ampliar un tema, se hará tras el Daily Standup, pero no se interrumpe la dinámica del Standup para elaborar una discusión.

Se hace siempre a la misma hora y en el mismo lugar. Si falta alguien, no se pospone la reunión.

**Revisión de sprint**

Al final de un sprint, el equipo realiza dos eventos: la revisión del sprint y la retrospectiva del sprint.

En la reunión de revisión de sprint se presentan los trabajos completados y su duración no debería ser superior a 4 horas para un Sprint de 1 mes.

**Retrospectiva del sprint**

Después de cada sprint, se lleva a cabo una retrospectiva del sprint, en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado. El propósito de la retrospectiva es realizar una mejora continua de la implementación de Scrum. Esta reunión tiene un tiempo fijo de cuatro horas.

**2.3.10 Documentacion**

**Product backlog**

El *product backlog* se trata como un documento de alto nivel para todo el proyecto. Es el conjunto de todos los requisitos de proyecto, el cual contiene descripciones genéricas de funcionalidades deseables, priorizadas según su retorno sobre la inversión ([ROI](https://es.wikipedia.org/wiki/Retorno_de_la_inversi%C3%B3n)). Representa el *qué* va a ser construido en su totalidad. Es abierto y solo puede ser modificado por el *product owner*. Contiene estimaciones realizadas a grandes rasgos, tanto del valor para el negocio, como del esfuerzo de desarrollo requerido. Esta estimación ayuda al *product owner* a ajustar la línea temporal (KEV) y, de manera limitada, la prioridad de las diferentes tareas. Por ejemplo, si dos características tienen el mismo valor de negocio la que requiera menor tiempo de desarrollo tendrá probablemente más prioridad, debido a que su [ROI](https://es.wikipedia.org/wiki/Retorno_de_la_inversi%C3%B3n) será más alto.

**Sprint backlog**

El sprint backlog es el subconjunto de requisitos que serán desarrollados durante el siguiente sprint. Al definir el sprint backlog, se describe el *cómo* el equipo va a implementar los requisitos durante el sprint. Por lo general los requisitos se subdividen en tareas, a las cuales se asignan ciertas *horas de trabajo,* pero ninguna tarea con una duración superior a 16 horas. Si una tarea es mayor de 16 horas, deberá ser dividida en otras menores. Las tareas en el *sprint backlog* nunca son asignadas, son tomadas por los miembros del equipo del modo que les parezca adecuado.

**Burn down chart**

La **[burn down chart](https://es.wikipedia.org/wiki/Burn_down_chart" \o "Burn down chart)**es una gráfica mostrada públicamente que mide la cantidad de requisitos en el Backlog del proyecto pendientes al comienzo de cada Sprint. Dibujando una línea que conecte los puntos de todos los Sprints completados, podremos ver el progreso del proyecto. Lo normal es que esta línea sea descendente (en casos en que todo va bien en el sentido de que los requisitos están bien definidos desde el principio y no varían nunca) hasta llegar al eje horizontal, momento en el cual el proyecto se ha terminado (no hay más requisitos pendientes de ser completados en el Backlog). Si durante el proceso se añaden nuevos requisitos la recta tendrá pendiente ascendente en determinados segmentos, y si se modifican algunos requisitos la pendiente variará o incluso valdrá cero en algunos tramos.

**Definition of Done**

El **[Definition of Done](https://es.wikipedia.org/w/index.php?title=Definition_of_Done&action=edit&redlink=1" \o "Definition of Done (aún no redactado))**es un documento con una serie de criterios comunes para determinar cuando una tarea está completamente hecha.

**(Los puntos dentro este apartado son solo referenciales ya que se puede ampliar o reducir en función a la metodología ágil elegida, aunque se recomienda utilizar SCRUM, aproximadamente de unas 10 a 15 páginas. Ya que deben incluir imágenes, tablas, etc.)**

**2.4 FUNDAMENTOS TEORICOS DEL SISTEMA (depende del sistema)**

**2.4.1 Definición de conceptos**

**2.4.2 Terminología del área de inventario (es solo un ejemplo)**

**2.4.3 Manejo de información**

**Los puntos a describir dentro de este apartado son solo referenciales ya que en función al tipo de sistema a desarrollar se especificarán los términos y la información que corresponda, aproximadamente de unas 5 a 10 páginas. Ya que pueden incluir imágenes, tablas, etc.)**

## CAPÍTULO III – MARCO APLICATIVO O PROPUESTA DE SOLUCION

### 3.1 INTRODUCCIÓN

### 3.2 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

**3.2.1 Requerimientos Funcionales**

|  |  |
| --- | --- |
| N° | FUNCIONALES |
| 1 | La aplicación móvil permitirá realizar la programación de la hora del riego |
| 2 | La aplicación móvil permitirá recibir los datos de los sensores para medir la cantidad de agua |
| 3 | La aplicación móvil permitirá recibir los datos de los sensores para medir la cantidad del nivel del pH |
| 4 | La aplicación móvil permitirá recibir los datos de los sensores para medir la cantidad de la temperatura |
| 5 | La aplicación móvil se encargara de enviar notificaciones en caso de que la temperatura no sea adecuada |
| 6 | La aplicación móvil se encargara de enviar notificaciones en caso de que el pH no sea adecuado |
| 7 | La aplicación móvil tendrá un apartado de información de contacto para conectarse con la empresa |
| 8 | La aplicación móvil tendrá un apartado de información de ayuda sobre el uso de la aplicación |
| 9 | La aplicación móvil permitirá el encendido y apagado de las luces led |
| 10 | La aplicación móvil se conectara al arduino de los huertos hidropónicos |

**3.2.2 Requerimientos no funcionales**

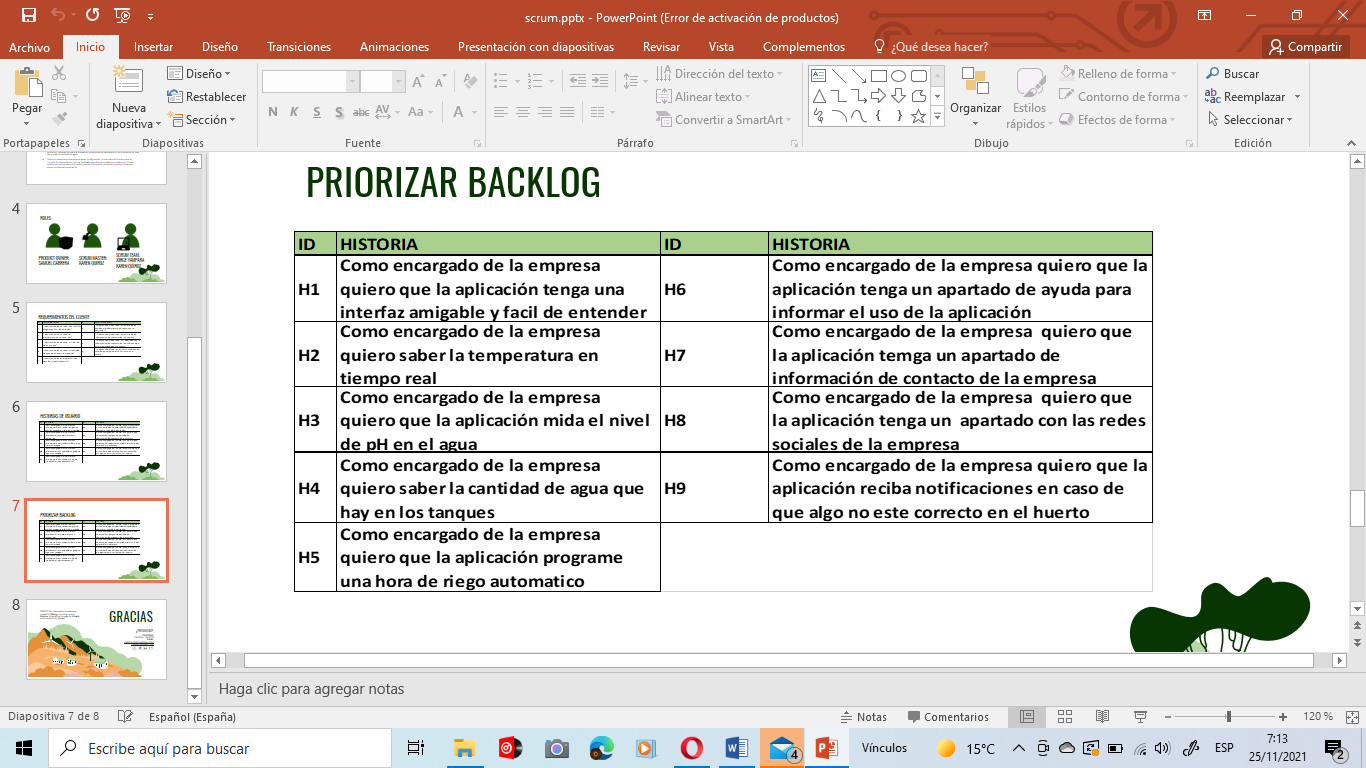
|  |  |
| --- | --- |
| N° | NO FUNCIONALES |
| 1 | La aplicación móvil tendrá una interfaz amigable con el usuario |
| 2 | El tiempo de respuesta entre la aplicación móvil y el arduino serán menor a 3000 m/s |
| 3 | La aplicación móvil requerirá menos de 100mb en el almacenamiento en el dispositivo |
| 4 | La aplicación móvil funcionara con conexión a internet |

**3.3 APLICACIÓN DE LA METODOLOGIA SCRUM**

**3.3.1 Historias de usuario**

|  |  |
| --- | --- |
| N° | Rol de usuario: Encargado de la empresa |
| H1 | Como encargado de la empresa quiero realizar el riego de forma remota |
| H2 | Como encargado de la empresa quiero saber el nivel de pH en el agua |
| H3 | Como encargado de la empresa quiero saber la temperatura en tiempo real |
| H4 | Como encargado de la empresa quiero saber la cantidad de agua que hay en los tanques |
| H5 | Como encargado de la empresa quiero programar una hora de riego automático |
| H6 | Como encargado de la empresa quiero recibir notificaciones en caso de que algo no este correcto en el huerto |
| H7 | Como encargado de la empresa quiero tener un apartado de información de contacto de la empresa “Raíces” |
| H8 | Como encargado de la empresa quiero un apartado de ayuda para informarme de cómo usar la aplicación |
| H9 | Como encargado de la empresa me gustaría tener un apartado con las redes sociales de la empresa para estar actualizado con la misma |
| H10 | Como encargado de la empresa me gustaría modificar los límites de temperatura para que la aplicación me envié notificaciones |
| H11 | Como encargado de la empresa me gustaría modificar los límites de pH para que la aplicación me envié notificaciones |
| H12 | Como encargado de la empresa me gustaría que la aplicación tenga una interfaz amigable y fácil de entender |

**3.3.2 Product Backlog**

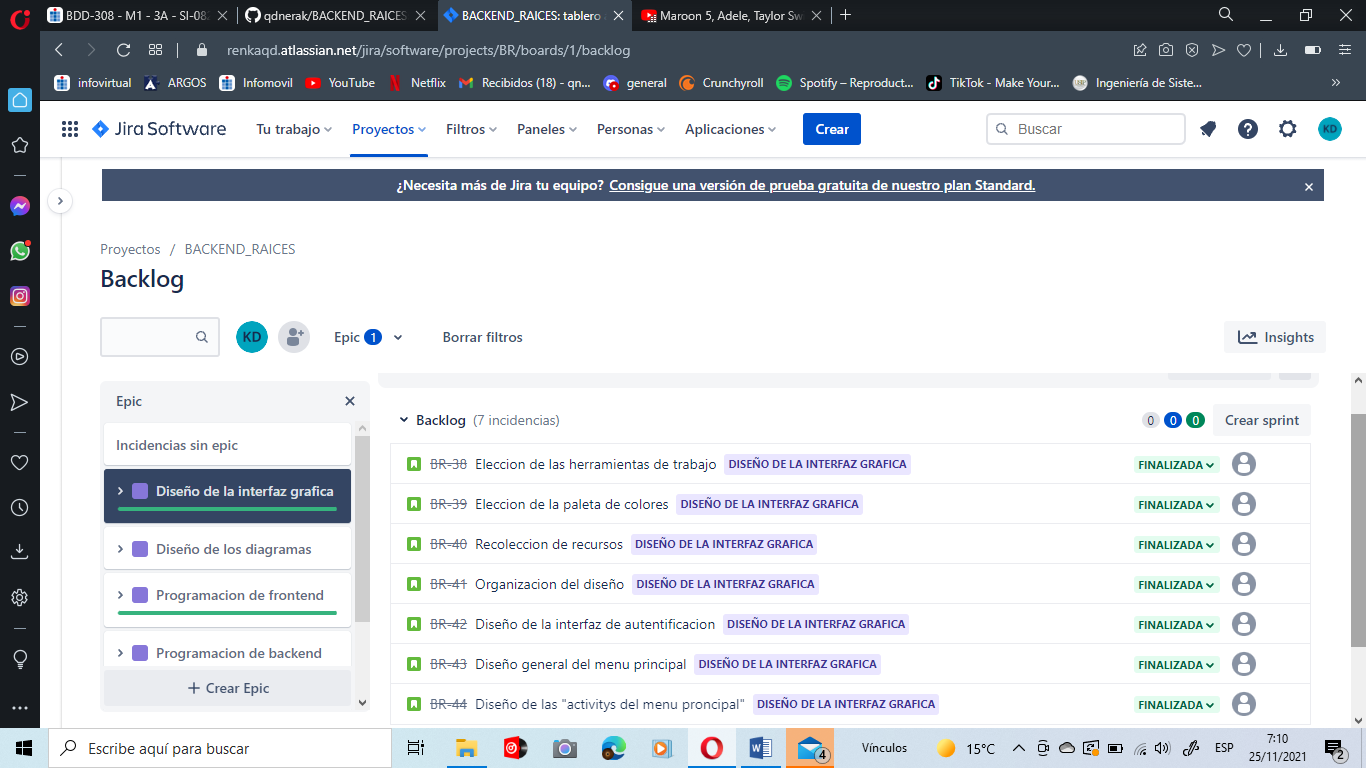


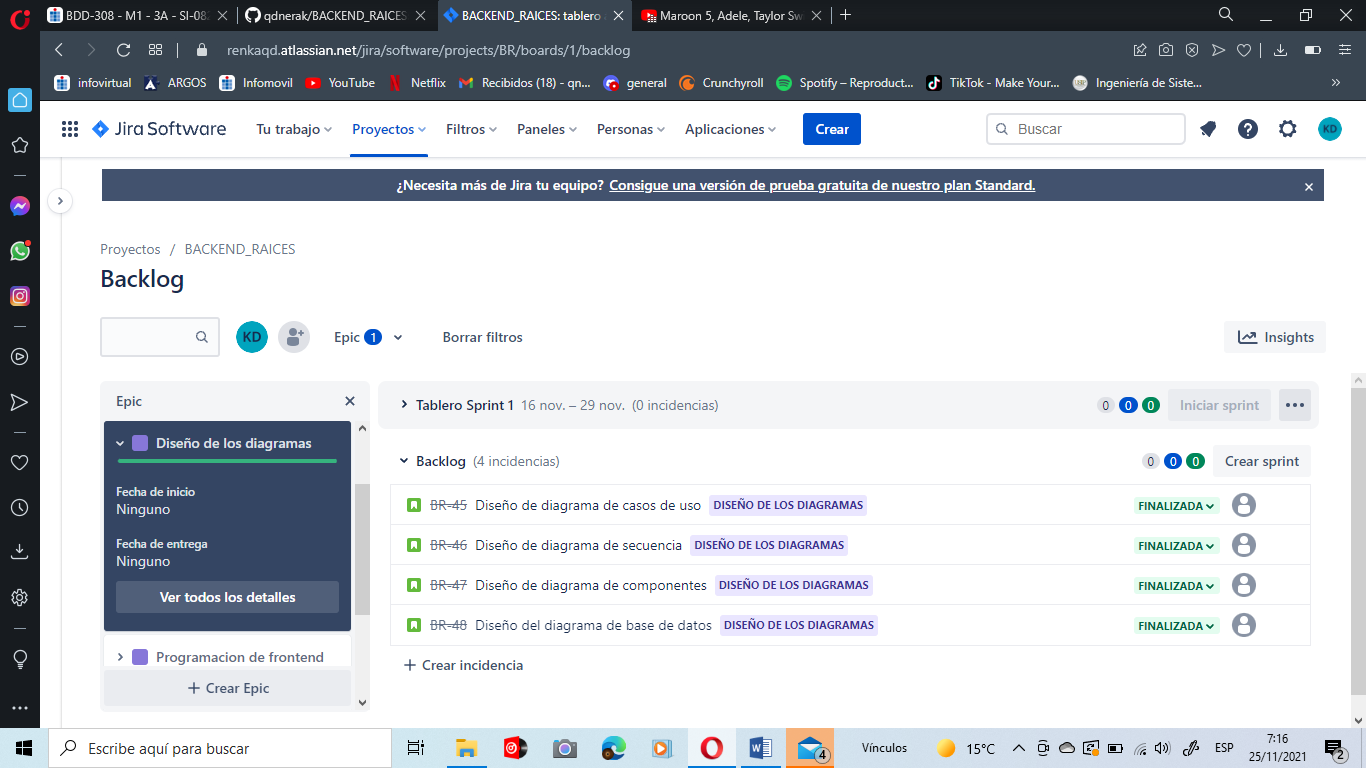
**3.3.3 Sprint Backlog**

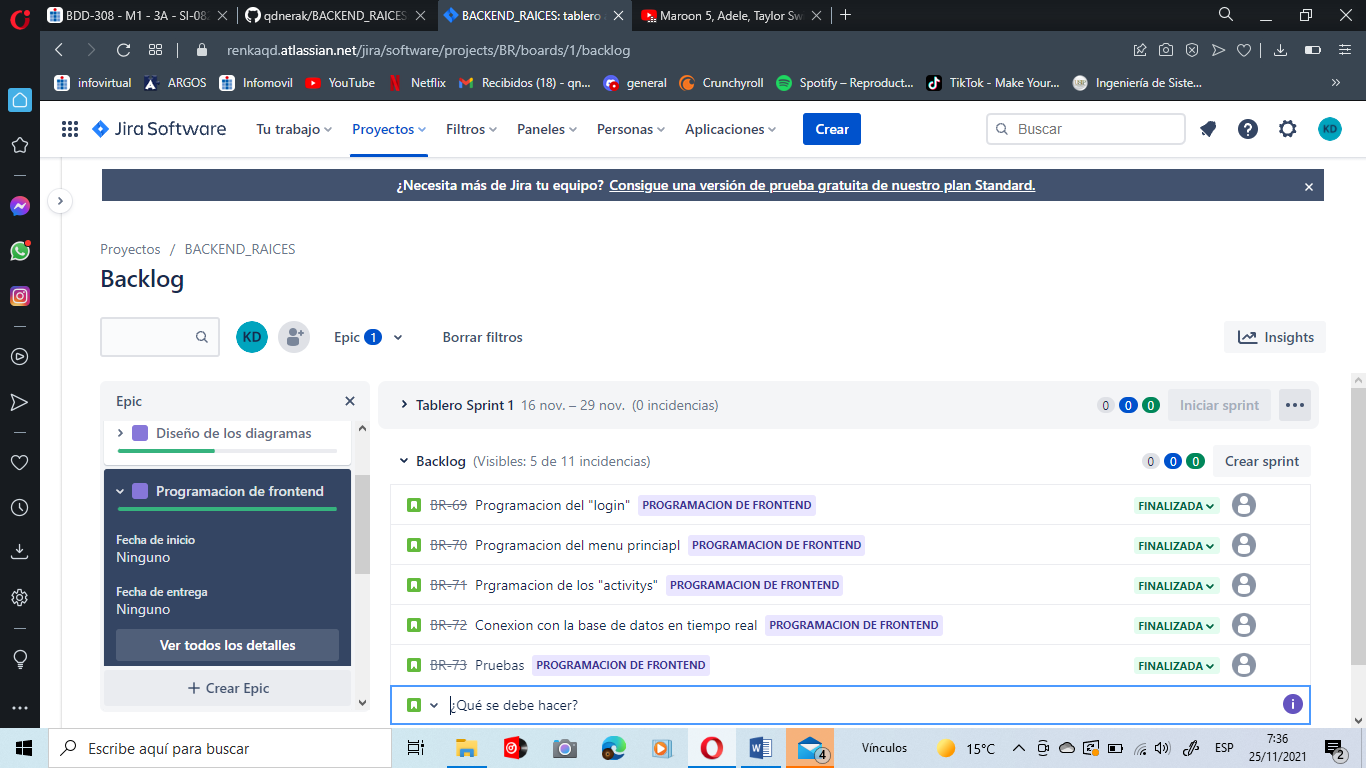
**3.3.4 Reuniones**

**3.3.5 Roles de usuario**

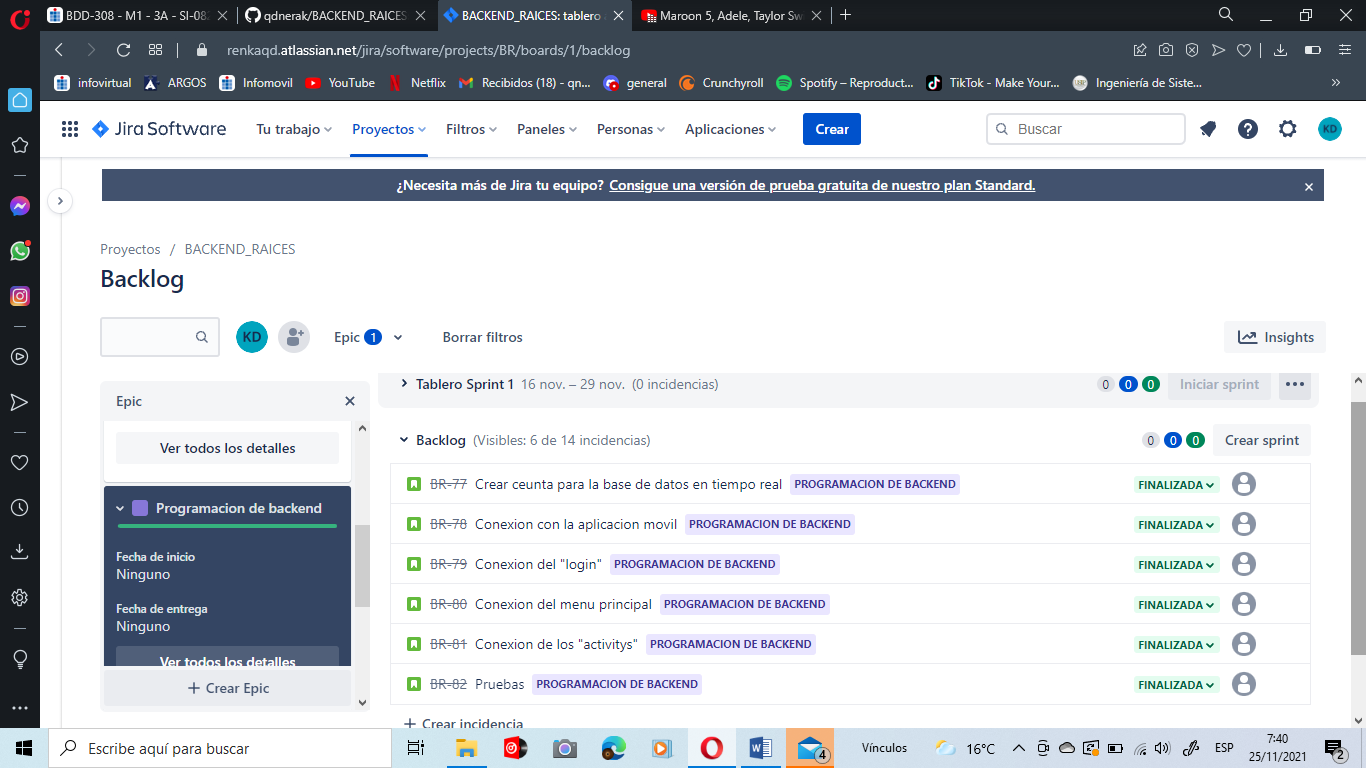
**3.3.6 Planificación de Sprints**

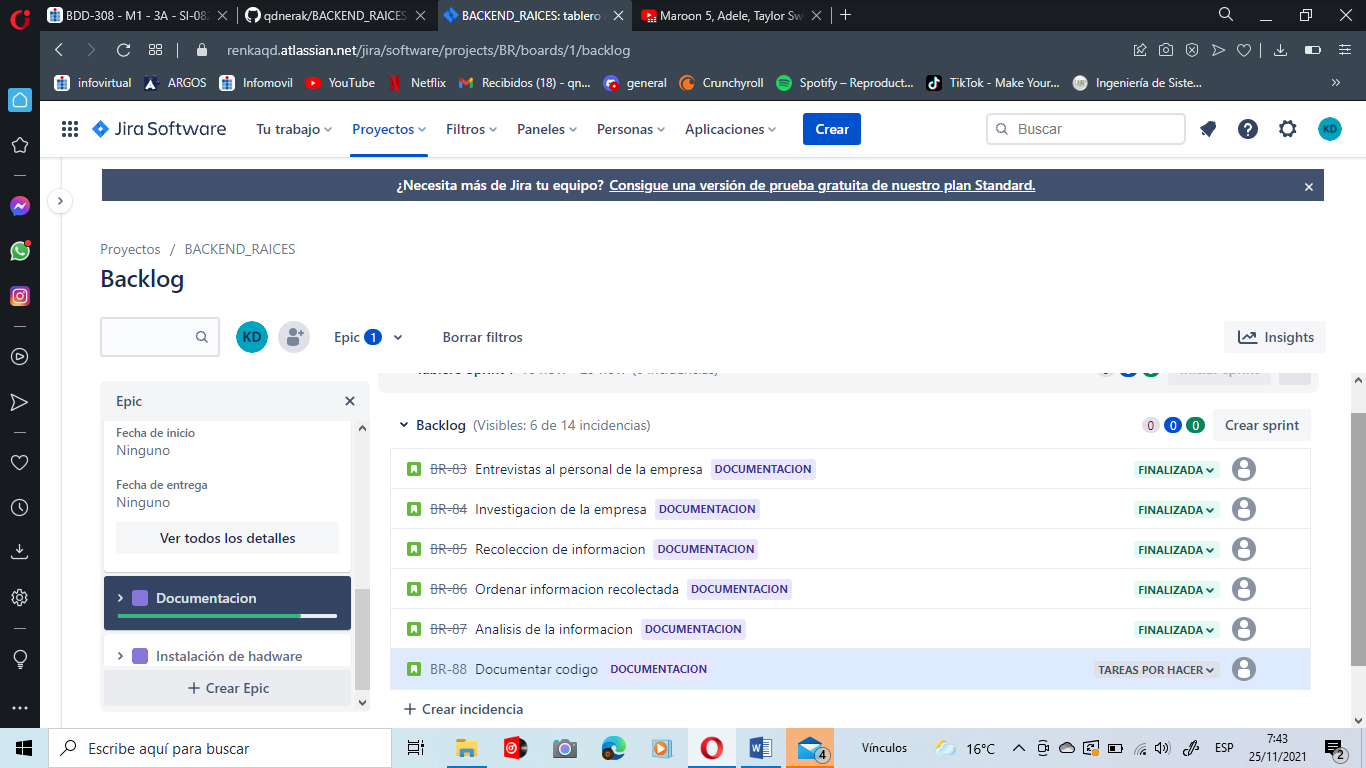
**3.3.6.1 Primer Sprint**

**3.3.6.2 Segundo Sprint**

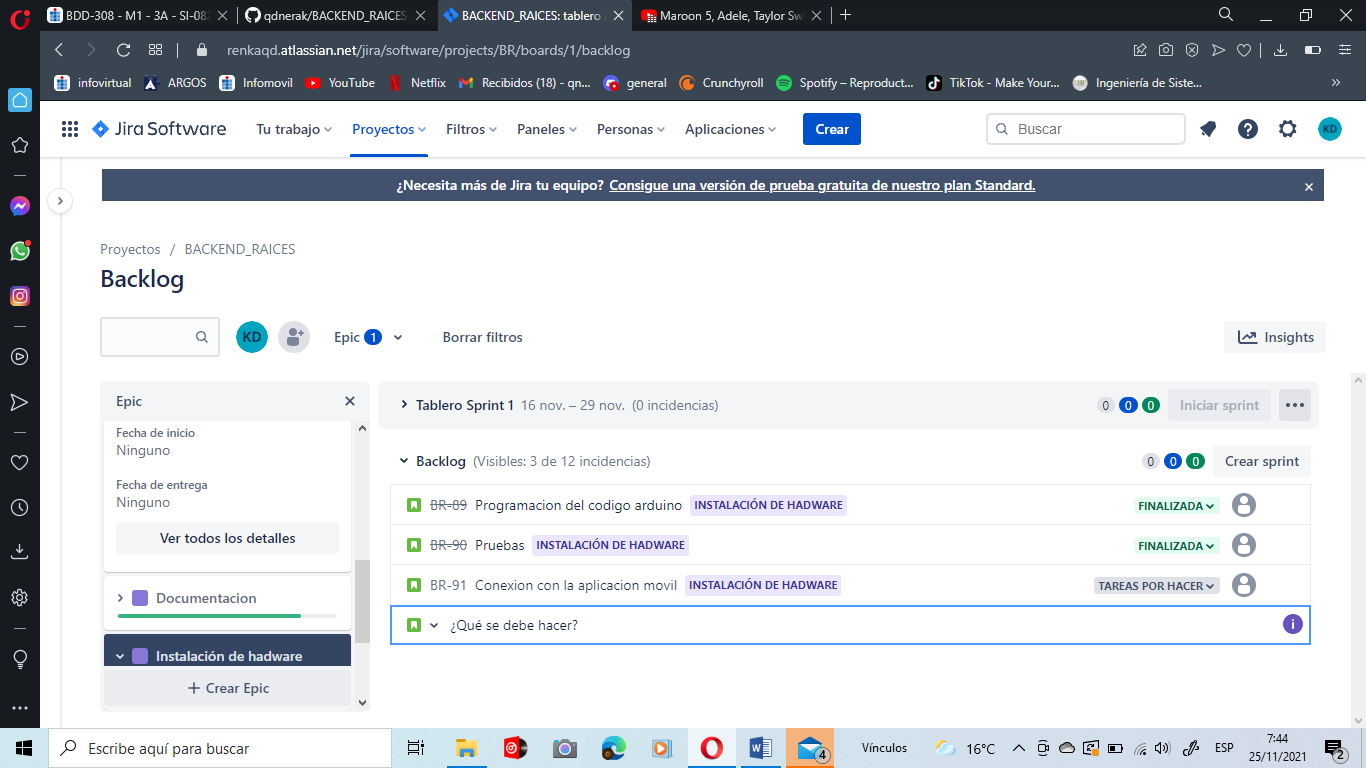
**3.3.6.3 Tercer Sprint**

**3.3.6.4 Cuarto Sprint**

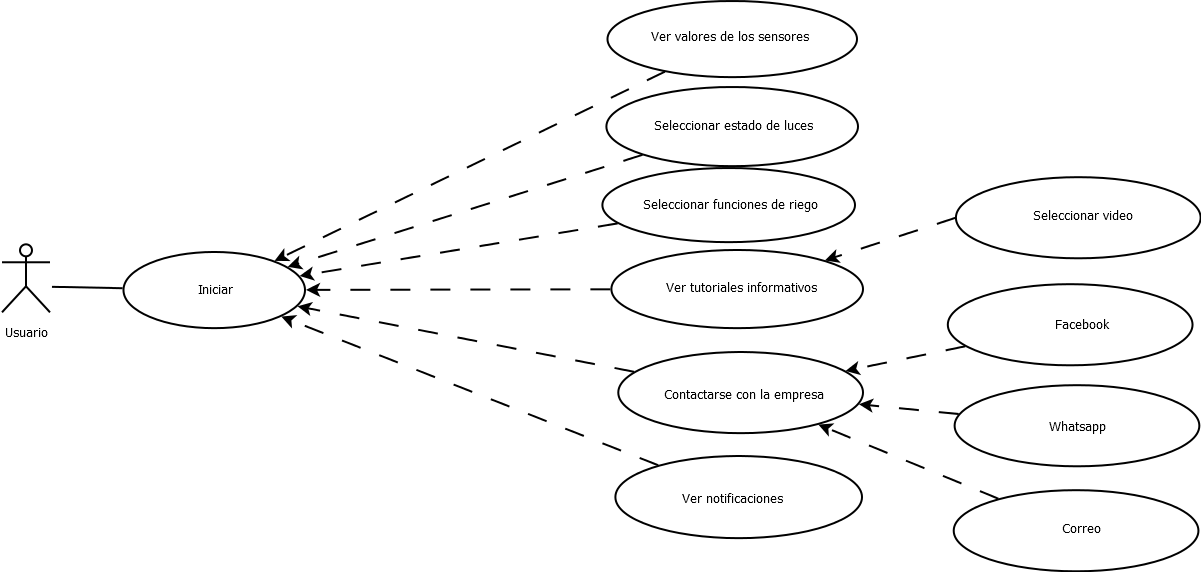


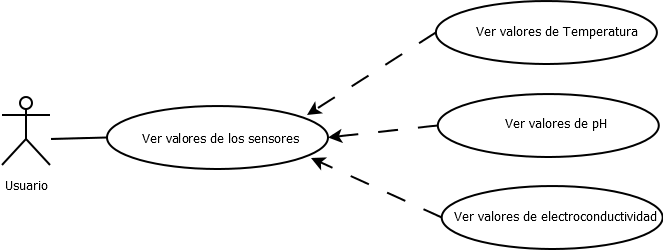
**3.3.6.5 Quinto Sprint**

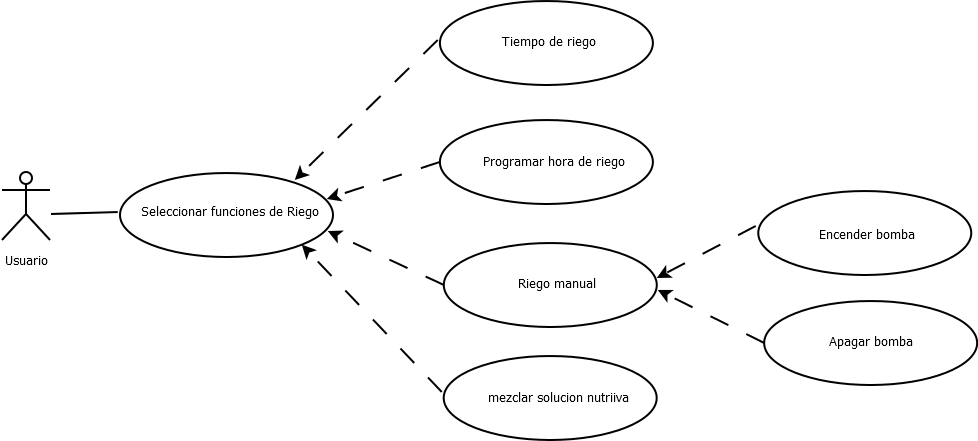
**3.3.6.6 Sexto Sprint**

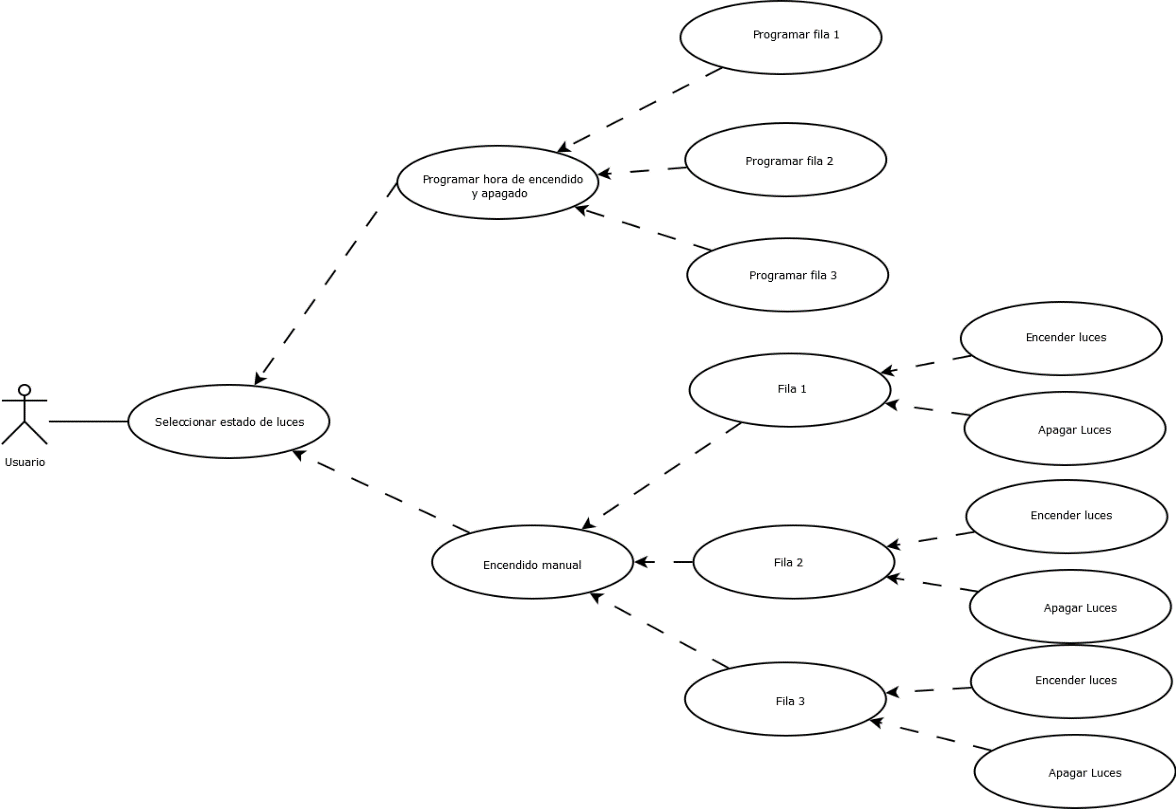


**3.3.7 Diagramas de Casos de Uso**

 **3.3.7.1 General**

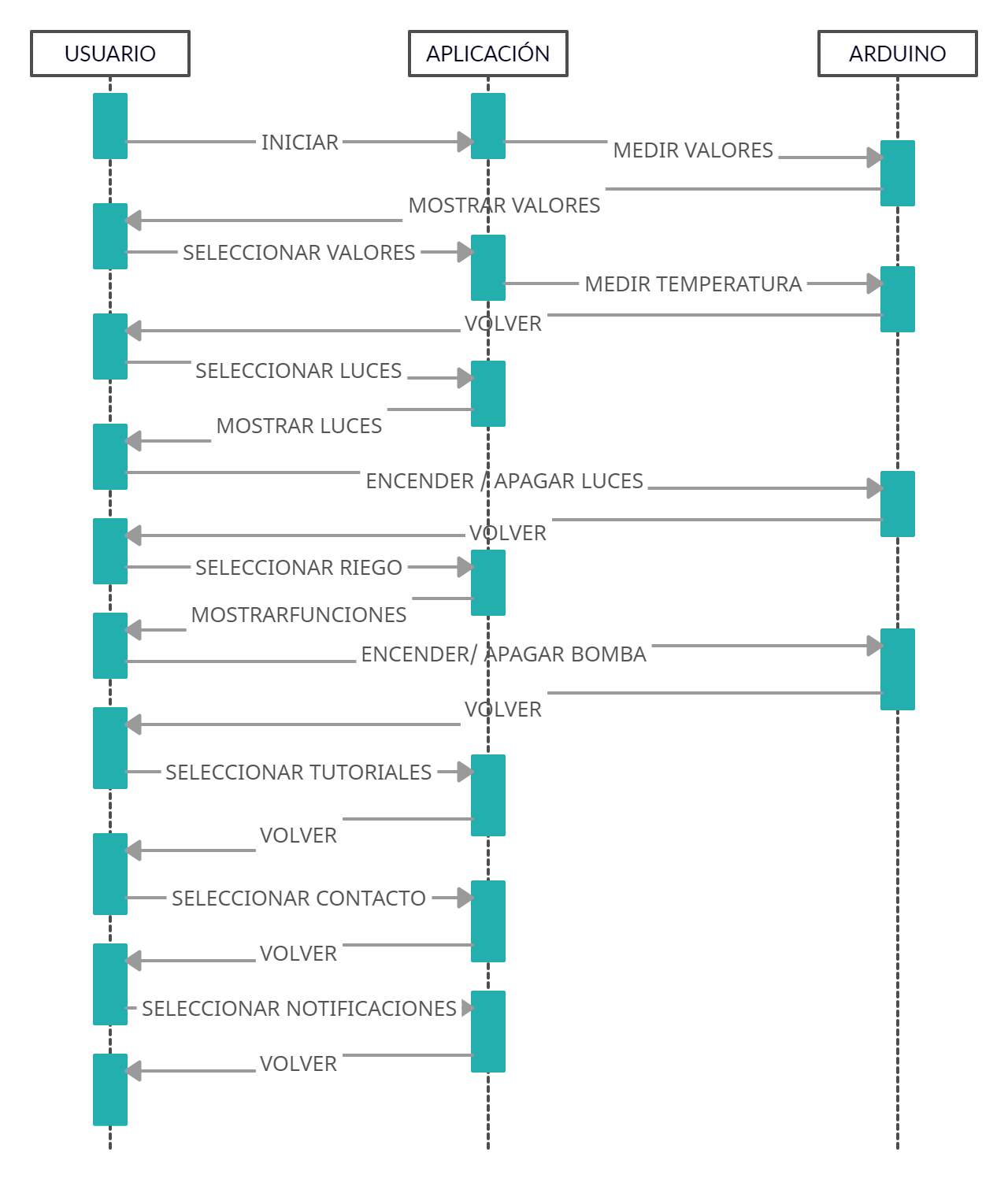
****

**3.3.7.1 Especifico**

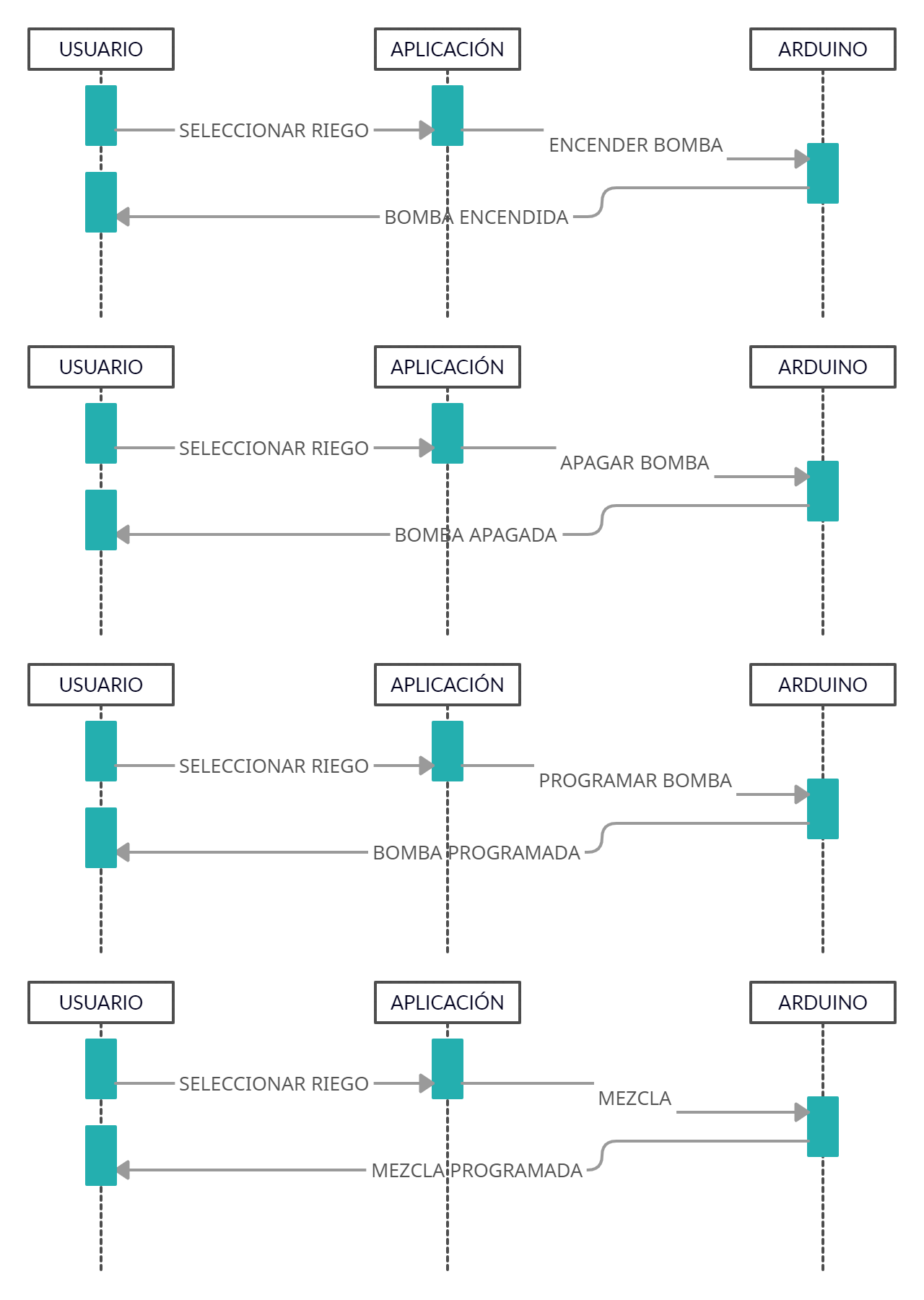
****

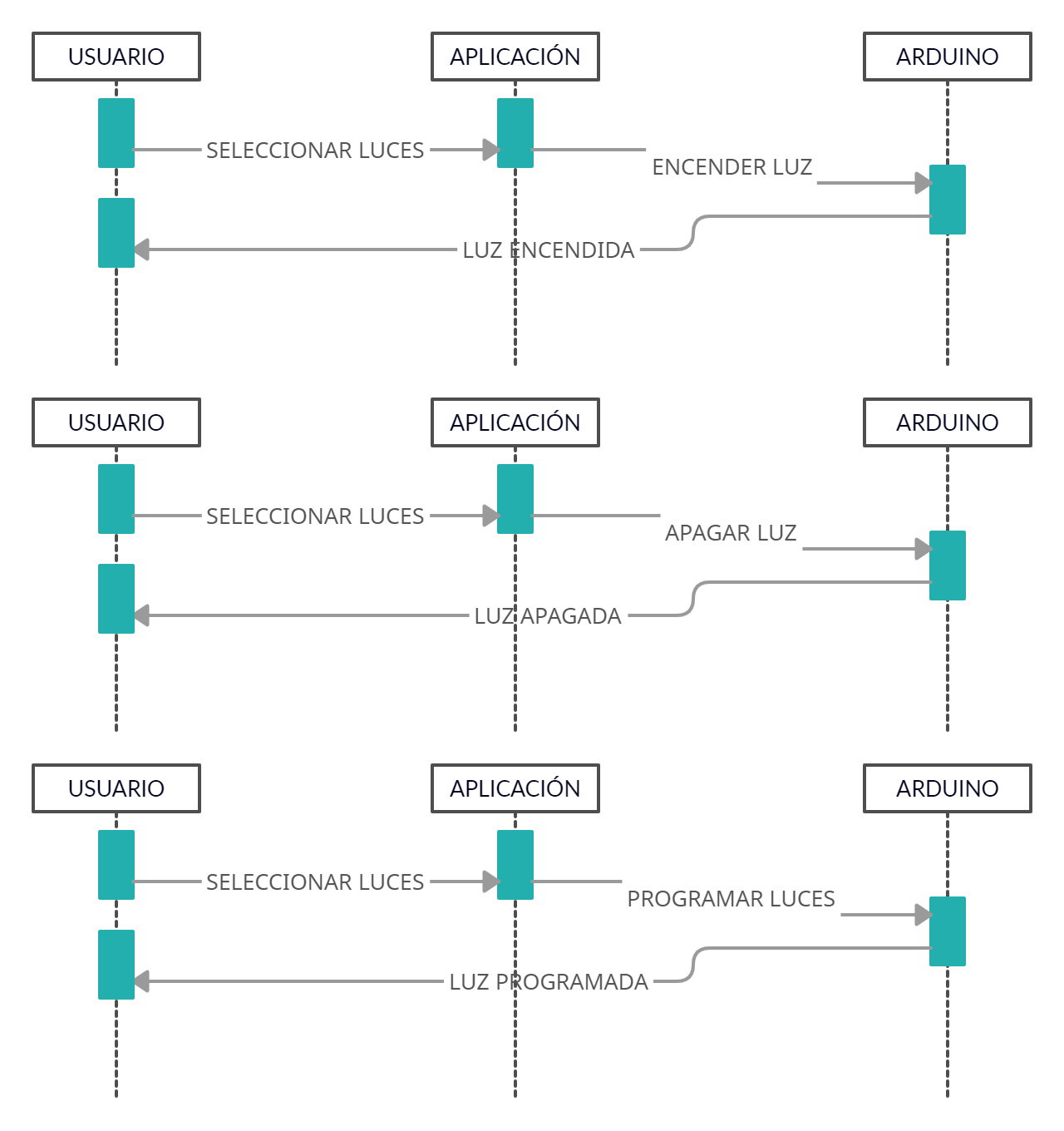
**3.3.8 Diagramas de Secuencia**

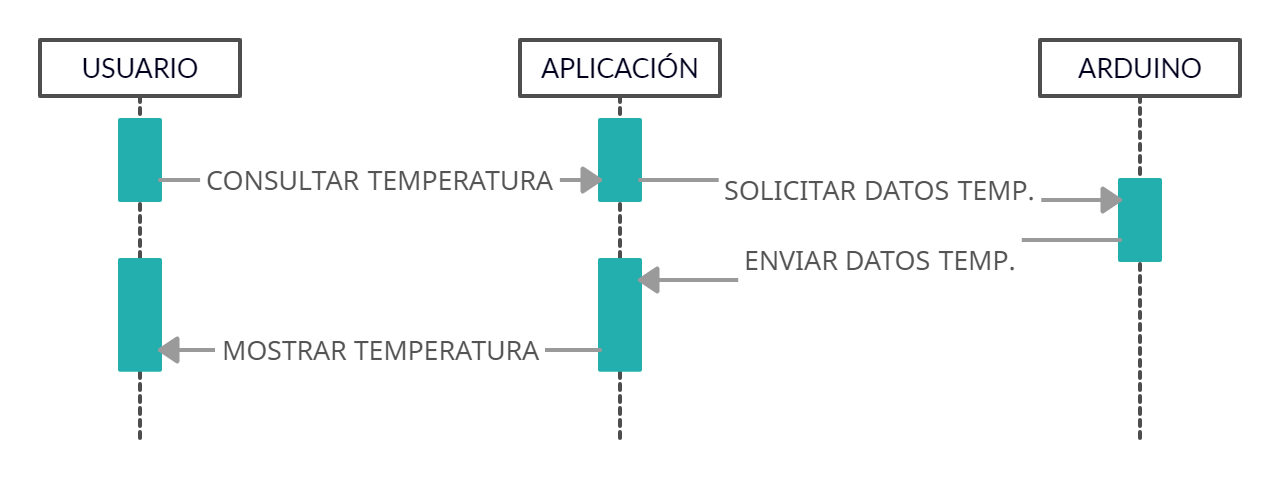
**3.3.8.1 General**

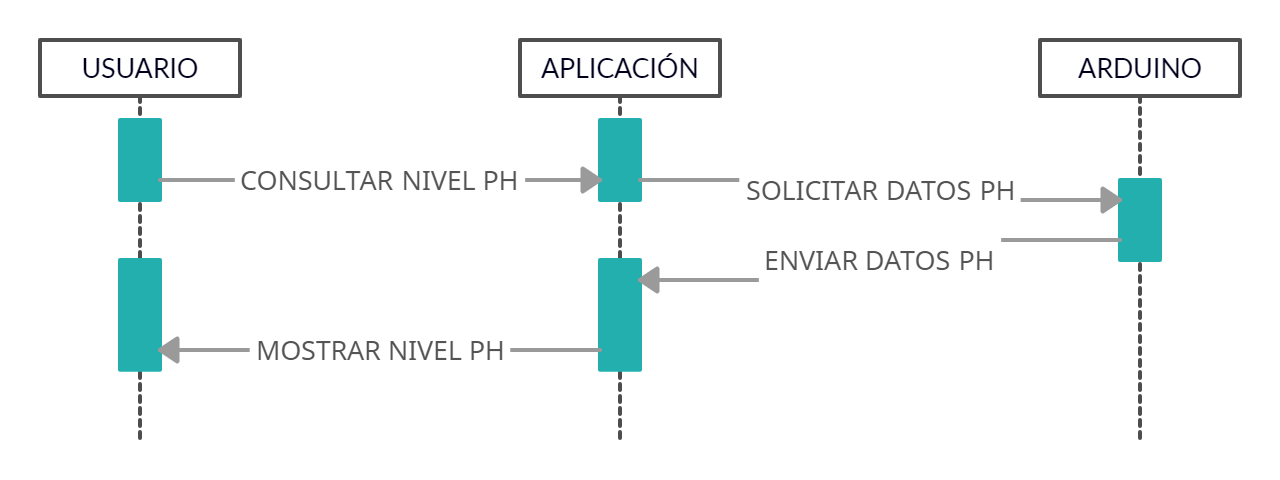
****

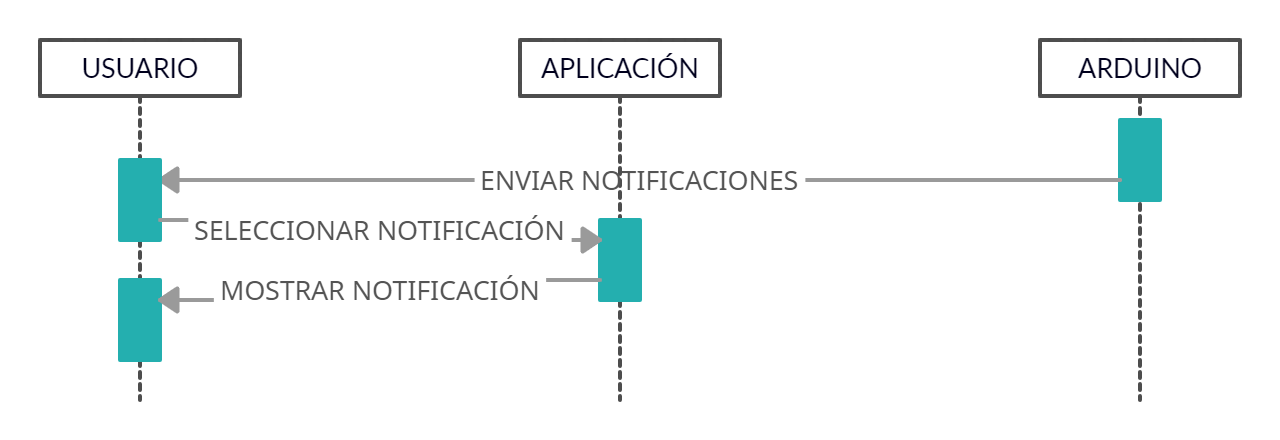
**3.3.7.1 Especifico**

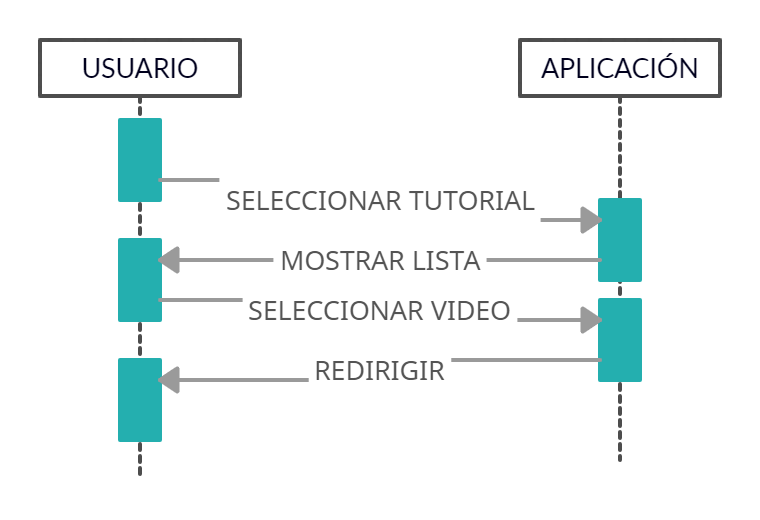
****

****

****

****

****

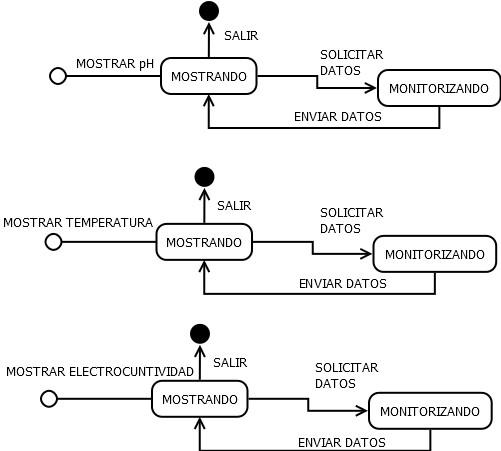


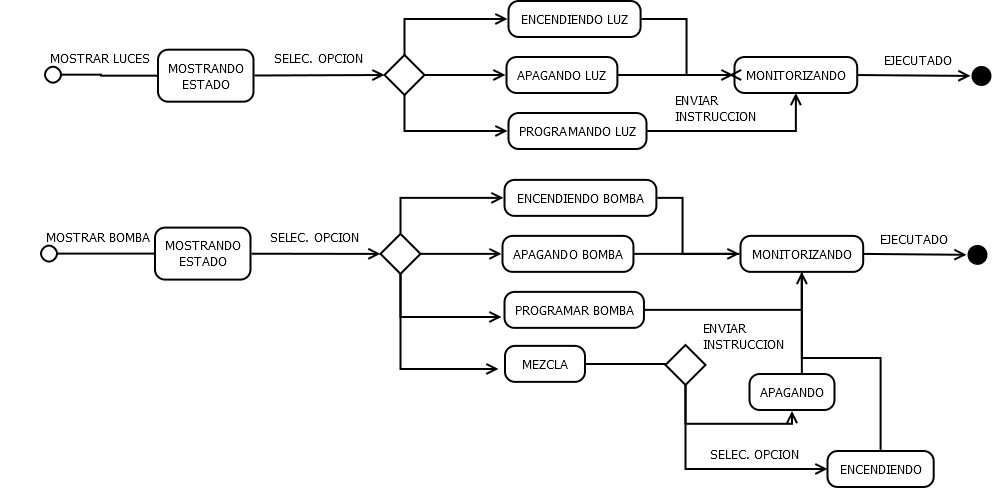
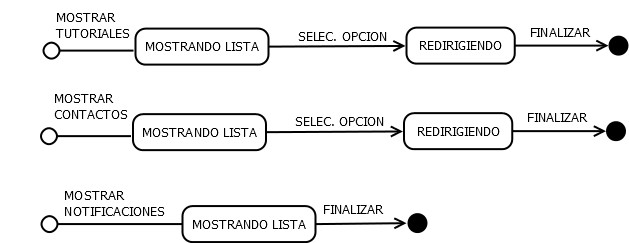
**3.3.9 Diagramas de estados**

**3.3.9.1 General**

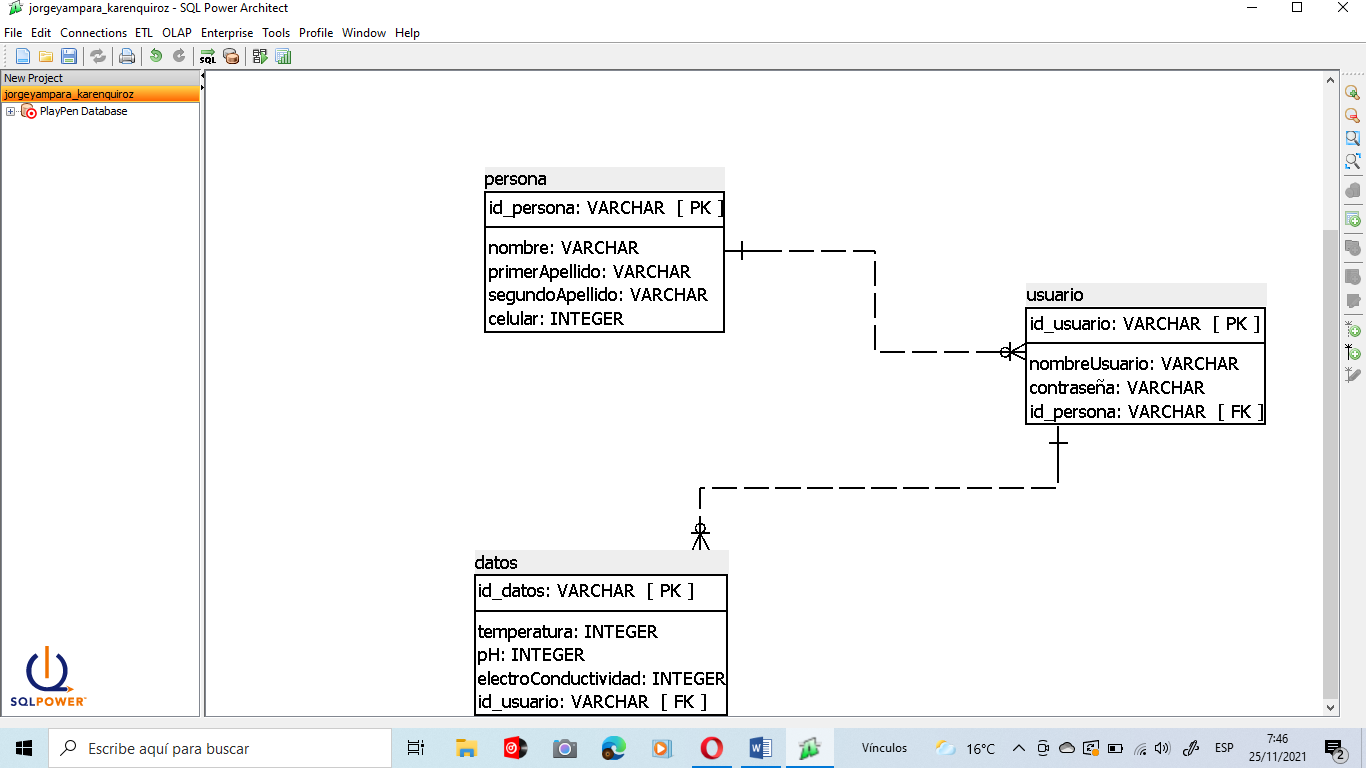


**3.3.9.2 Especifico**





**3.3.10 Diagrama de la base de datos**

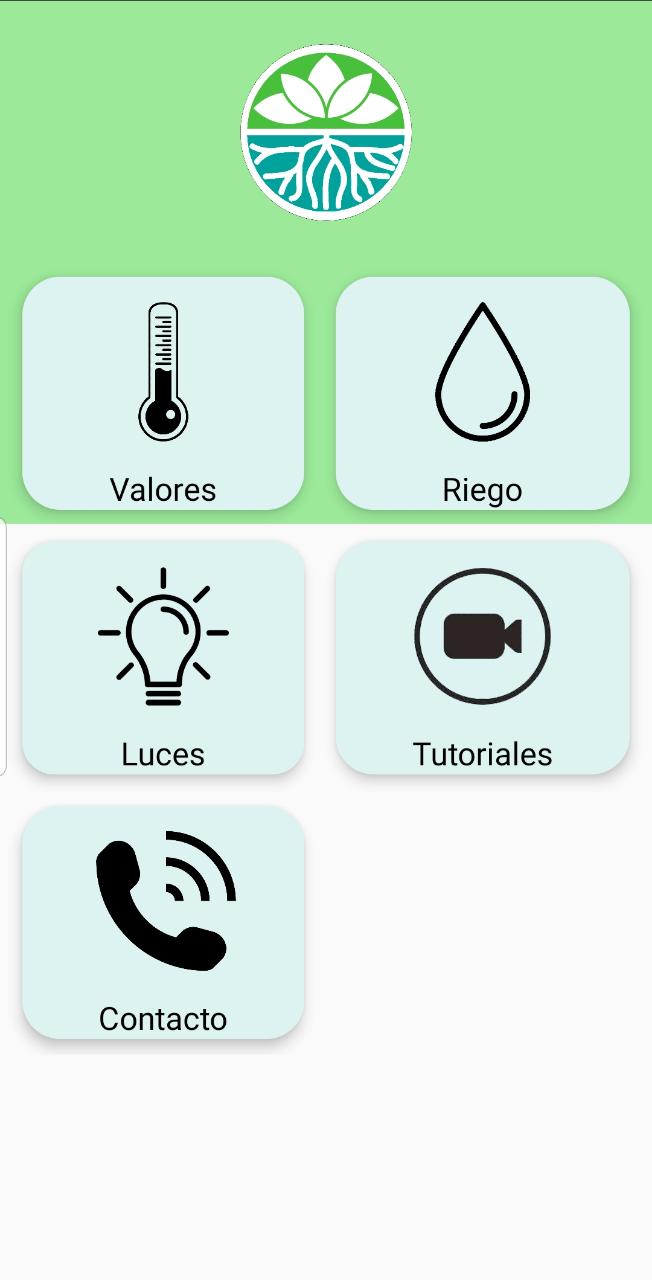


**3.3.10 Diseno final de la aplicación**

**3.3.10.1 Login**

****

**3.3.10.2 Menu Principal**



**La descripción de este apartado se desarrollará en función a la metodología ágil elegida, aproximadamente de unas 20 a 60 páginas. Ya que deben incluir imágenes, tablas, gráficos, diagramas, etc. Cada uno de los diagramas o tablas deben estar detalladamente descritos.**

**3.4 DISEÑO DE LA BASE DE DATOS**

**3.4.1 Descripción del tipo de diagrama a utilizar (sustentar con bibliografía de investigación)**

**3.4.2 Simbología – Elementos – etc.**

**3.4.3 Diagrama ER del proyecto**

**3.4.4 Descripción de cada una de las tablas de la BD**

**3.4.5 Diccionario de Datos, ETC.**

**3.5 FASE DE CONSTRUCCIÓN**

**3.5.1. Herramientas de Implementación**

**3.5.2 Herramientas de Diseño**

En función a la metodología ágil elegida desarrollar cada uno de los sprints ó módulos del sistema

**3.5.3 Entorno de Desarrollo**

**3.5.3.1 Implementación del Acceso del Sistema**

**3.5.3.2 Implementación del módulo de usuario**

**3.5.3.3**

**3.5.3.4**

**3.5.4 Diagrama de Actividades Administración de Usuarios**

**3.5.5 Fase de Pruebas**

**3.5.6 Requerimientos del sistema**

**3.5.6.1 Requerimientos de Hardware**

**3.5.6.2 Requerimientos de Software**

**3.5.7 Instalación del sistema**

**3.5.7.1 Instalación de los Programas**

**3.5.7.2 Instalación de los archivos**

**3.5.7.3 Utilización del Sistema**

**3.6 CONCLUSIONES Y RECOMENDACIÓN**

**3.6.1 CONCLUSIONES**

**3.6.1.1 Conclusiones acerca de los objetivos**

**3.6.1.2 Conclusiones acerca de las herramientas**

**3.6.1.3 Conclusiones acerca de la metodología**

**3.6.2 RECOMENDACIONES**

**3.6.2.1 Recomendaciones acerca de la instalación**

**3.6.2.2 Recomendaciones acerca del uso del sistema**

**3.6.2.3 Recomendaciones acerca de las actualizaciones**

**BIBLIOGRAFIA**

**Índice de figuras**

Figura 1: Modelo CVS........................................................................................................16

Figura 2: Ciclo de vida Struts.............................................................................................18

Figura 3: Ciclo de vida JSF..................................................................................................20

Figura 4: Ciclo de vida WebWork ......................................................................................21

Figura 5: Ciclo de vida Tapestry ........................................................................................23

Figura 6: Secuencia de request Struts...............................................................................29

**Índice de tablas**

Tabla 1: Comparación metodologías agiles vs metodologías tradicionales....................................26

Tabla 2: Elaboración de Product Backlog .......................................................................................38

Tabla 3: Ciclo de vida Iterativo-Incremental…………………………………………........................................61

**ANEXOS**

**MANUAL DE USUARIO**

**MANUAL TECNICO**