

PARALELO:

#1

FECHA DE ENTREGA:

03/01/2022

PROFESORA:

ADRIANA COLLAGUAZO

TERMINO ACADEMICO:

2021-T2

INTEGRANTES:

- **LUIS SANTAMARIA**
- **ABEL GOMEZ**
- **DYLLAN BASTIDAS**
- **ARIEL GONZALEZ**

Introducción

El proyecto presente es la creación de un sistema de riego automático, permite el control de la humedad del suelo de la planta para que el crecimiento y el mantenimiento de esta sea lo más óptimo. Como idea principal se tendrá los sistemas de riego tradicional que lo que haremos es parlo al segundo nivel agregándole un sensor en la zona donde se encuentra la planta y así este junto a nuestra aplicación controlada desde el móvil maneje los datos necesarios para mantener la tierra con la húmeda que requiera cada planta.

Interfaz del riego de las plantas- Móvil

La aplicación está hecha para celulares en donde el usuario podrá agregar plantas y, podrá encender y apagar el riego de la planta desde el móvil, también tendrá la opción de poner como parámetro la humedad mínima que requiere la planta y así la aplicación estará en riego continuo y manteniendo la planta siempre a la humedad ideal.

Proceso

Será el Arduino quien determina la humedad de la tierra con ayuda de los sensores, los cuales serán exportados a una base de datos en la nube y podrán ser visibles con la aplicación desde el móvil, la cual decidirá si lo usa para un control de riego automático o manual.

Resultados


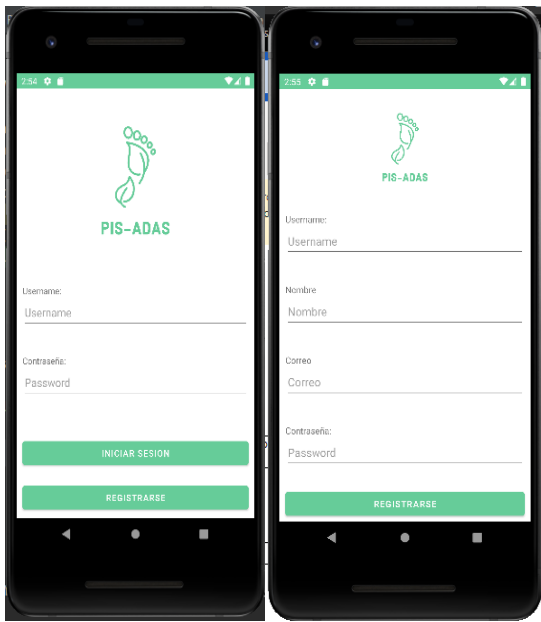
1	2
 <p>Ilustración 1 Splash art con el logo de la empresa</p>	 <p>Ilustración 2 Pantalla de inicio de sesión y registrarse</p>
Una pantalla inicial donde solo se presente el logo de la empresa.	Seguido, se presenta una pantalla para realizar inicio de sesión o registrarse.
3	4



Ilustración 3 Ventana principal o home

Una vez que se ingresó se procede a mostrar la página principal que cuenta con una lista de todas las plantas con las que se cuenta.

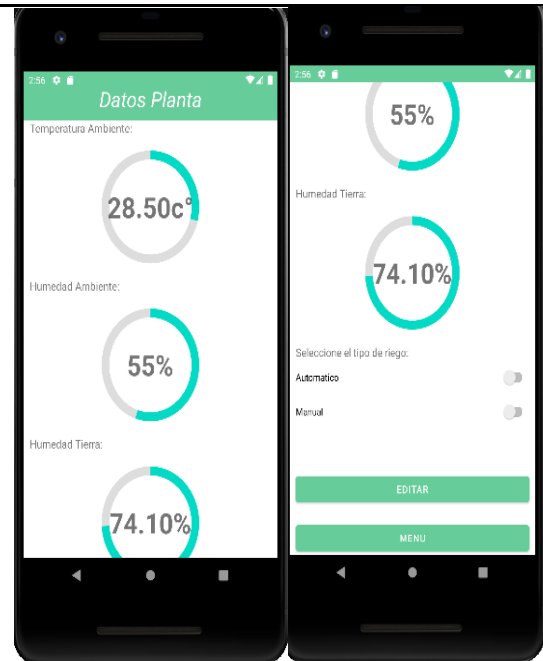
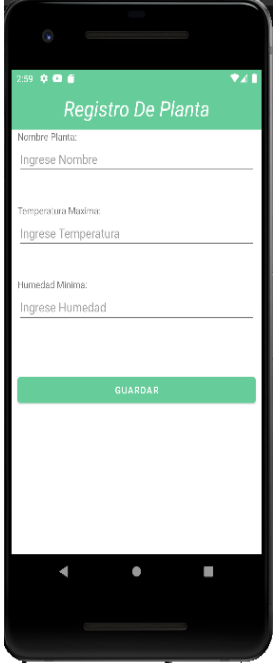
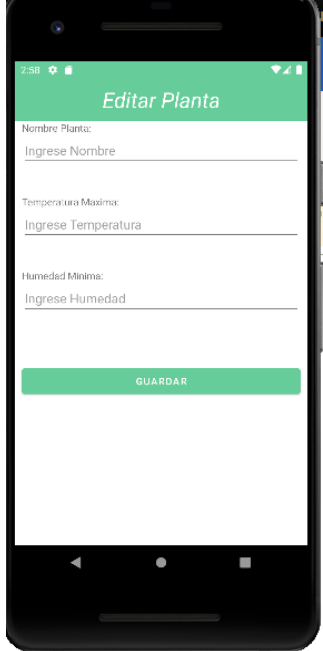


Ilustración 4 Datos de la planta

Cuando se selecciona un ítem de la lista se abrirá una nueva vista personalizada indicando los datos a tiempo real de los sensores de dicha planta.

 <p><i>Ilustración 5 Ingresar nueva planta</i></p>	 <p><i>Ilustración 6 Editar datos de la planta</i></p>
<p>Se cuenta con una vista en el cual se puede ingresar los parámetros para crear una nueva planta en el sistema</p>	<p>En la sección de categorías se muestran las categorías de libros que existen en la empresa. Una vez que se seleccione una categoría, esta debe mostrar una lista de libros de solo esa categoría como la página de home.</p>

DESARROLLO

Activities:

- Activity_splash: contiene el código usado para crear la página de arranque que contiene el logo de la aplicación y se encarga de cargar la base de datos interna.
- MainActivity: contiene el código necesario para inicio sesión del usuario y validación de que los campos este lleno y el nombre de usuario este correcto.
- RegistroActivity: Contiene código que sirve para la creación de usuario, que luego es almacenado en una base de datos.
- MenuActivity: contiene el código necesario para mostrar por medio del Recycler View todas las plantas pertenecientes a dicho usuario.
- DatosPlantaActivity: Contiene el código que se necesita para mostrar los datos a tiempo real de los sensores de cada planta registrada en la aplicación (temperatura ambiente, humedad ambiente y humedad tierra).
- IngresoPlantaActivity: Presenta el código con el cual se puede agregar una nueva planta al listado de plantas registradas, con respecto al usuario en el cual se inició sesión.

- **EditarPlantaActivity:** Este Activity es parecido al IngresoPlantaActivity con la diferencia que se mantiene la planta registrada en la base de datos y solo varían los datos que el usuario desea cambiar.

Clases:

- **Data:** Base de datos provisional del sistema.
- **ListPlanta:** clase que se encarga de crear los objetos de tipo planta para ser utilizados en la app.
- **LibroAdapter:** Adaptador de ListPlanta para poder ser utilizado en el RecyclerView de la vista del menu principal
- **Usuario:** Es la clase que permite identificar a cada usuario para la base de datos.
- **AdminSQLiteOpenHelper:** Es la clase que se encarga de crear la base interna de la planta con SQLite

Microcontrolador:

- **PIS_ADA.ino:** Es el archivo encargado de decirle al controlador su funcionamiento y que pines son los que se usaran para la lectura de los sensores.

Presupuesto:

A continuación, se detalla una tabla con los materiales usados en el proyecto y su costo en tiendas de electrónica ubicadas en el centro de la ciudad.

Servomotor	\$ 5.50
Cables de conexión	\$ 3
Sensor de humedad de tierra	\$ 4.50
Sensor DHT11	\$ 5
Modulo ESP8266	\$ 10
Total	\$ 28

Conclusiones:

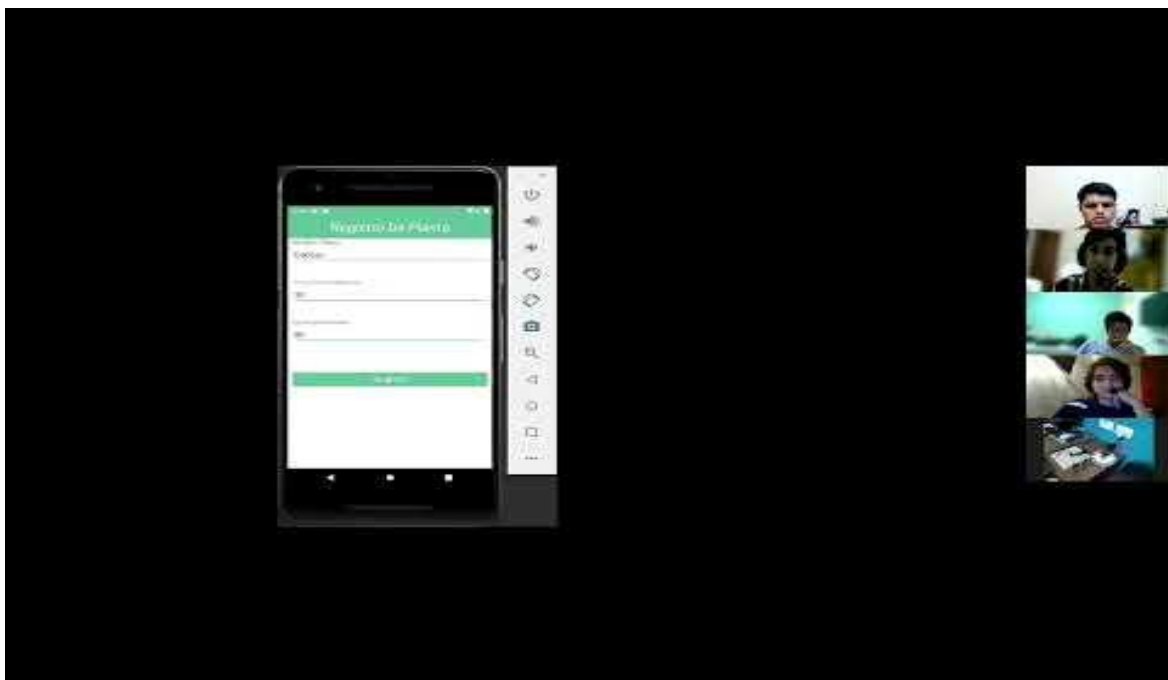
- Se logro llevar a cabo la creación de un sistema de riego automático, con ayuda de Android Studio, más la ayuda de diferentes sensores que nos permite saber el estado de la planta.
- El uso de las bases de datos nos permitió en entendimiento del almacenamiento y procesamiento de la información.
- Se logro manipular diferentes ventanas el entorno de trabajo del software Android Studio

Recomendaciones:

- La creación de base de datos, a la hora de que un usuario ingrese es indispensable dado que facilita el almacenamiento de las platas para cada usuario.
- Se recomienda hacer un estudio previo sobre la teoría de los colores, para así basarse en una paleta de colores que serán usados en la aplicación, de modo que sea agradable para el usuario.

LINK VÍDEO

[PIS-ADAS // Aplicación de riego y control automatizado](#)



ESTADÍSTICAS DEL GITHUB

December 3, 2021 – January 3, 2022

Period: 1 month ▾

Overview

0 Active Pull Requests

0 Active Issues

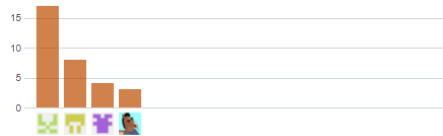
0
Merged Pull Requests

0
Open Pull Requests

0
Closed Issues

0
New Issues

Excluding merges, **4 authors** have pushed **32 commits** to main and **32 commits** to all branches. On main, **0 files** have changed and there have been **0 additions** and **0 deletions**.



REPOSITORIO GITHUB

https://github.com/afgomez20/PST_PRO_G2

Aquí se encuentra el Código fuente de hardware y software