Trabajo Fin de Máster
Máster en Ingeniería Industrial

Diseño y desarrollo de una App para la supervisión de un campo solar de colectores Fresnel

Autora: Luisa Piñero Alarcón

Tutora: Amparo Núñez Reyes

Departamento de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería

Sevilla, 2021

Trabajo Fin de Máster
Máster en Ingeniería Industrial

Diseño y desarrollo de una App para la supervisión de un campo solar de colectores Fresnel

Autora:

Luisa Piñero Alarcón

Tutora:

Amparo Núñez Reyes

Profesora titular

Departamento de Ingeniería de Sistemas y Automática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2021

Gracias a mis padres y a mi hermano por animarme a seguir en la distancia desde hace ya 10 años. Gracias a Paco, mi novio, sin cuyo apoyo no habría sido posible finalizar este capítulo de mi vida.

Agradecimientos

Doy las gracias a mi tutora Amparo Núñez Reyes por su paciencia, por haber sabido respetar mi ritmo y animarme en cada correo y en cada llamada, por encontrar un tema que me interesase y haber sabido orientarme para poner el broche final a mi paso por la ETSI que tantos momentos me ha dado (buenos y malos). Gracias a mi familia, que hace 10 años apostó por mí, y me apoyaron en mi idea de irme a estudiar fuera de casa (más de 400km fuera). Gracias a mi novio, que lleva aguantando a una novia ausente y estresada todo este tiempo, y siempre con una sonrisa y un hombro donde apoyarme. Pero sobre todo, gracias a Gema, la que ha estado conmigo hombro con hombro en los últimos cursos, tropezándonos con las mismas piedras que nos alejaban de terminar este máster, pero siempre juntas, gracias a ella y a su último empujón, que me motivaron a terminar este trabajo para salir las dos juntas de esta escuela en la misma convocatoria.

Resumen

En este trabajo se lleva a cabo el desarrollo de una aplicación móvil para comunicarse con el campo de colectores de tipo Fresnel ubicado en la azotea del edificio de la Escuela Técnica Superior de Ingeniería de Sevilla.

Con la aplicación se podrá monitorizar el sistema y actuar sobre el mismo dependiendo de la cualificación del personal registrado. Los estudiantes podrán acceder a la planta para obtener los datos que arroja en cada preciso momento la planta o para acceder a datos de días ya pasados. Los docentes podrán hacer lo mismo que los estudiantes y, además, podrán aceptar el registro de nuevos perfiles de acceso y añadir datos de incidencia al histórico de datos. También tendrán la posibilidad de apagar los motores de la planta con un único botón, en caso de detectar un peligro inminente.

La aplicación tendrá distintas pantallas para acceder a los datos y a las acciones a realizar que se busquen en cada momento. Se adapta a las necesidades de cada perfil y será sencilla e intuitiva para que no sea necesario un conocimiento alto en tecnología para poder utilizarla. Esta aplicación conectará con bases de datos online (alojadas en una web) para acceder al registro de usuarios que pueden acceder a ella y para acceder a los datos de la planta que se deseen obtener o actualizar en cada momento.

Esta aplicación podrá ser modificada para monitorizar cualquier otra planta de los laboratorios o para añadir otras plantas solares a la vez que esta, simplemente replicando lo que se ha desarrollado para la planta de la ETSI y seleccionando al inicio del acceso la planta a la que se desea acceder.

Abstract

In this work, the development of a mobile application is carried out to communicate with the field of Fresnel collectors located on the roof of the building of the Higher Technical School of Engineering.

With the application you can monitor the system and act on it depending on the qualifications of the registered personnel. Students will be able to access the plant to obtain the data that the plant throws at that precise moment or to access data from past days. Teachers will be able to do the same as students and, in addition, they will be able to accept the registration of new access profiles and add incident data to the data log. They will also have the possibility to switch off the plant's motors with a single button, in case of detecting an imminent danger.

The application will have different screens to access the data and the actions to be carried out that are searched at all times. It adapts to the needs of each profile and will be simple and intuitive so that a high knowledge of technology is not necessary to use it. This application will connect with online databases (hosted on a website) to access the registry of users who can access it and to access the data of the plant that they wish to obtain or update at any time.

This application can be modified to monitor any other plant in the laboratories or to add other solar plants at the same time, simply by replicating what has been developed for the plant of the Higher Technical School of Engineering and selecting the plant at the beginning of access to which you want to access.

Índice

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

1.1. Motivación y objetivos

El objetivo de este trabajo consiste en la creación de una aplicación móvil capaz de monitorizar el campo solar instalado en el tejado de la Escuela Técnica Superior de Ingeniería de Sevilla (ETSI).

Con la aplicación se podrá monitorizar la planta en todo momento, accediendo a los datos, pudiendo actuar sobre ciertas variables, recibiendo notificaciones de alarmas y obteniendo datos de históricos. Para acceder a la aplicación se deberá ser un usuario registrado, se podrá acceder desde dos perfiles diferenciados, perfil estudiante y perfil docente. El perfil estudiante accederá a los datos tanto actuales como históricos, mientras que el perfil docente, además, podrá introducir comentarios a los registros de histórico, podrá aceptar el acceso de nuevos usuarios y podrán parar los motores de la planta desde un botón en caso de ser necesario.

La aplicación está diseñada para un único campo solar, pero es fácilmente adaptable para añadir cuantas plantas sean necesarias, ya que sería simplemente replicar la planta que ya está creada e implementar en la pantalla de inicio la selección del campo solar deseado.

1.2. Estructura del documento

- *Capítulo 1*: Objetivo de este trabajo y motivaciones para llevarlo a cabo.

- *Capítulo 2*: En este capítulo se detallan las características de la planta solar de la ETSI, su funcionamiento y las ventajas de este tipo de planta frente a otras. Se comenta también el estado del arte del sistema de monitorización y control de la planta, es decir, la existencia de otras aplicaciones con el uso que se le quiere dar a la que se desarrolla en este trabajo.

- *Capítulo 3*: En este capítulo se explica con detenimiento el software utilizado en este trabajo, sus características y sus usos.

- *Capítulo 4*: En el cuarto capítulo se comenta el diseño de la aplicación, tanto el estudio previo como los requisitos y la maqueta que sirve de partida para el desarrollo.

- *Capítulo 5*: En este capítulo se especifica todo el proceso del desarrollo de la aplicación, el diseño completo en Unity de la aplicación, el acceso de usuarios a dicha aplicación y la conexión de ésta para acceder a los datos de la web.

- *Capítulo 6*: En el capítulo sexto se realiza el seguimiento de las pruebas y resultados de la aplicación, así como su implementación.

- *Capítulo 7*: En el último capítulo se abordan las conclusiones del trabajo.

2 PLANTA SOLAR

En este capítulo se realizará una breve descripción del campo solar de tipo Fresnel perteneciente a la planta de refrigeración solar ubicada en la azotea de la Escuela Técnica Superior de Ingeniería. [1] [2] [3]

Ilustración 1 - Planta solar tipo Fresnel

2.1. Descripción

La planta solar se encuentra instalada en la azotea de la ETSI. La instalación se sitúa a 6° de longitud hacia el oeste y a 37,41° de latitud. La planta solar está situada a 12°3'1" al suroeste de la fachada sur y paralelamente a ésta

Ilustración 2 - Localización de la planta solar

Esta planta de refrigeración solar (PRS) funciona a partir de un campo de captadores solares lineales de Fresnel que recoge la radiación del sol con objeto de calentar agua hasta unos 180◦C. El agua a esta temperatura se utiliza para accionar una máquina climatizadora que transforma la energía térmica en frío, el cual es apto para la climatización de las dependencias del edificio. Este tipo de funcionamiento ocurre cuando se necesita generar frio. Sin embargo, en invierno puede cambiar de funcionamiento para producir calor, lo cual lleva a cabo con un rendimiento del 95%. Como todos los equipos de este tipo, precisa de un sistema de disipación del calor generado en el absorbedor y en el condensador, para lo cual se utiliza un sistema de captación de agua que actúa como sistema de refrigeración. En concreto, se aprovecha el circuito de agua de captación del río Guadalquivir que se creó para todos los edificios de la Exposición Universal del año 1992 (Expo'92) de la Isla de la Cartuja (Sevilla). Adicionalmente, la máquina de absorción instalada incorpora un quemador de gas natural que entra en funcionamiento como fuente de energía auxiliar, de forma que se garantice la disponibilidad del servicio en momentos de carencia de potencia térmica desde el lado solar. Es decir, cuando la irradiación es insuficiente para accionar la máquina de absorción, la energía proporcionada por el gas natural sirve de apoyo para mantener el proceso en funcionamiento.

El esquema general de la planta se encuentra en la siguiente ilustración, en él podemos ver que la salida del sistema de energía solar pasa por un tanque de almacenamiento y después llega a una máquina de absorción de doble efecto. El objetivo de la instalación es refrigerar el edificio a partir del frío producido en la máquina de absorción por medio del agua calentada por la energía solar (por eso se puede llamar planta de refrigeración solar). En caso de tener exceso de energía esta se almacena en el tanque de almacenamiento, mientras que si la energía producida no es suficiente se utilizará el quemador de Gas Natural instalado para generar la energía restante.

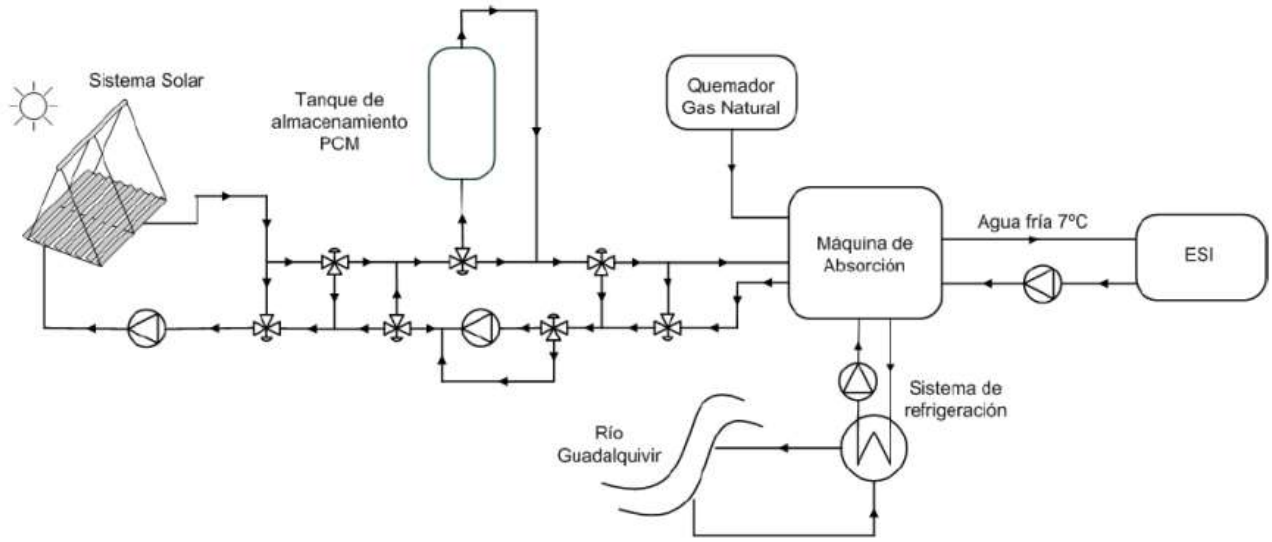

Ilustración 3 - Esquema general de la instalación

Centrándonos en la planta solar, las características se encuentran en la siguiente tabla:

Extensión de terreno ocupada	480 m²
Superficie reflectora primaria total	352 m²
Orientación de la planta	Este-Oeste
Número de líneas receptoras	1
Longitud de la línea receptora	64
Tipo de receptor	De cavidad con reflector secundario y cubierta de vidrio
Altura de la línea receptora	4 m sobre los receptores primarios
Anchura del receptor	0,3 m
Tipo de absorbedor	Tubo de acero DIN 1,4541 (AISI 321) Acero inoxidable austenítico estabilizado
Fluido de trabajo	Agua
Generación de vapor	No
Presión de diseño	13 bar
Número de filas de reflectores primarios por línea receptora	22 filas
Número total de reflectores	176
Longitud de cada módulo reflector	4 m
Anchura de los reflectores	0,5 m
Reflectividad	0,92
Relación de concentración	25
Potencia térmica nominal	120KW

Tabla 1 - Características del sistema de captación solar instalado en la ETSI

Características	Valores
Acumulador	
Peso en vacío	4100 Kg
Peso lleno de agua	8150 Kg
Capacidad de almacenamiento	291 KW h
Potencia máxima	149.9 KW
Salto de temperatura máximo	30 ◦C
Carcasa Hidroquinona	
Temperatura de diseño	200 ◦C
Presión de diseño	1 bar
Volumen	3300 L
Tubos Agua presurizada	
Temperatura de diseño	200 ◦C
Presión de diseño	15 bar Volumen

Tabla 2 - Características del acumulador de la PRS

A continuación, se detallan los componentes del colector solar:

• La estructura de acero marca los límites del área donde están situados los espejos, está recubierta con pintura en polvo y sostiene tanto los espejos como el tubo de absorción y el reflector secundario.

• Los espejos reflectores son de vidrio y están unidos a un eje que es movido por un motor impulsor que se encargar del seguimiento solar. Por seguridad son ligeramente curvados elásticamente, el radio de curvatura es de entre 8,6 y 10.6 metros.

• Los motores colocados en cada fila mueven 8 espejos de cada fila, cuatro en cada lado.

• El reflector secundario envuelve al tubo captador y refleja la radiación solar que se desvía del receptor para aumentar la eficiencia óptica del sistema. También protege al receptor.

• El tubo receptor tiene una absortividad nominal de 0.94, tiene un diámetro de 70mm y una longitud de 64 m. La temperatura máxima soportada es de 200ºC y la presión máxima en su interior de 16 bares.

• Además de los componentes anteriores, en el sistema se utilizan diversos sensores que se comunican a través de CANopen con el autómata encargado del control de la planta, estos son los siguientes:

- **Sensor solar**: se utiliza para la calibración automática detectando la reflexión de los espejos primarios no centradas en el receptor.

- **Potenciómetro**: cada fila de espejos cuenta con un sensor de este tipo que envía la posición actual de cada fila.

- **Sensor de temperatura**: a la entrada y salida del captador se encuentra un sensor PT100 instalado para monitorizar la temperatura del fluido.

La mayoría de los componentes de la instalación se pueden ver en la siguiente ilustración, esta fotografía fue tomada en una de las visitas a la planta para la comprobación de su estado.

Ilustración 4 - Campo solar de la ETSI

2.2. Campo solar

El objetivo principal de la planta Fresnel es reflejar los rayos solares sobre un largo receptor cilíndrico donde se generará calor. El receptor cilíndrico está situado a lo largo de una línea recta, y es por eso que esta disposición es conocida como sistema de foco lineal.

El objetivo de un sistema de este tipo es la generación de energía, en este caso en forma de calor dentro del tubo cilíndrico que hemos comentado, por donde circula un fluido que será el encargado de transportar ese calor a una máquina de absorción, donde se absorberá dicho calor para producir frío. La generación de este calor se produce por la reflexión de la radiación solar a través de una serie de superficies reflectoras orientadas hacia el tubo receptor. Para concentrar correctamente los rayos solares sobre la superficie receptora se utiliza un mecanismo de control como puede ser un autómata programable. El autómata será el encargado de conocer la trayectoria del sol y hacer funcionar una serie de motores para mantener el sistema de superficies reflectoras orientadas hacia el tubo receptor. Por tanto, el sistema se compone de los siguientes elementos:

- **Receptor**: formado por uno o varios tubos situados en un plano superior y paralelo al de los dispositivos reflectores. Contiene el fluido que transporta el calor por la instalación. Se encuentra rodeado parcialmente por un reflector secundario.

- **Concentradores solares**: sistema generalmente formado por espejos ligeramente curvados encargado de concentrar la energía solar en el receptor.
- **Motores**: moverán los concentradores solares a partir de las señales recibas del sistema de control para que apunten al receptor y no perder así energía solar.
- **Sistema de control**: dispositivo que controlará cada concentrador de forma independiente para orientarlos hacia el receptor.
- **Estructura**: sujetará los diferentes planos de instalación.

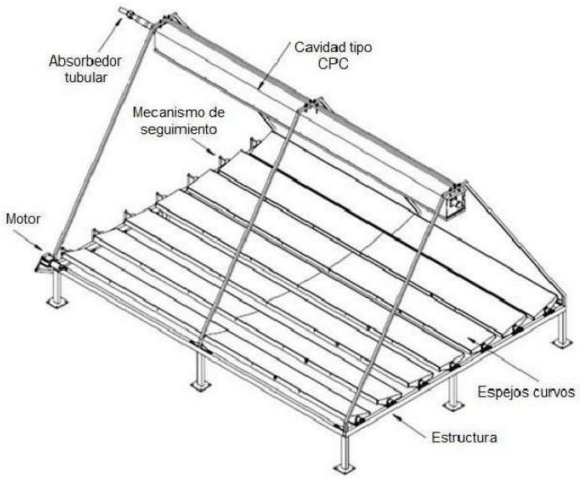

Ilustración 5 - Elementos de una planta solar tipo Fresnel

El funcionamiento completo de la planta se puede describir con el diagrama de la siguiente ilustración.

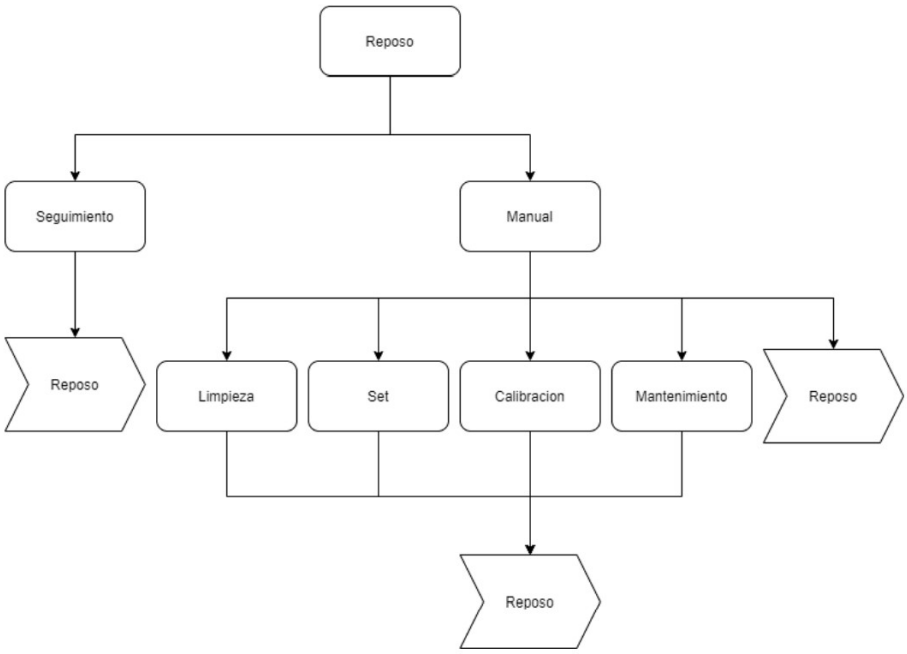

Ilustración 6 - diagrama de funcionamiento de la planta solar

En ella podemos encontrar un primer estado de "Reposo", donde se espera hasta que se le indique mediante una señal externa el modo de funcionamiento. Dentro del modo "Manual" hay varias posibilidades como puede ser la etapa de "Limpieza" donde se colocan todos los servomotores en una posición predeterminada para facilitar la limpieza de los espejos de cada fila. En el estado "Set" se colocarán los espejos en las posiciones indicadas de forma externa. En la etapa de "Mantenimiento" se colocarán los espejos en una posición vertical para facilitar el acceso al personal para comprobar el funcionamiento de los potenciómetros y servomotores fácilmente. La "Calibración" se realiza de forma manual, los datos se almacenan y se utilizan como referencia para que la etapa de "Seguimiento" funcione correctamente, esta última etapa es la encargada de que los espejos estén orientados hacia el tubo captador, tendrá en cuenta los cálculos internos para el desenfoque de algunos espejos y el error en la calibración de cada una de las filas por separado.

El proceso de calibración de la planta solar es primordial, ya que el resto del control dependerá de este paso. Como se ha mencionado anteriormente, este proceso se hace de forma manual pero no es eficiente ni tan exacto como si se realizase de forma automática. Por ello hay desarrollado un programa que se encarga de este paso y de que no se necesite de un operario para su funcionamiento, esto permite ejecutar la calibración de forma más asidua, en caso de reinicio y pérdida de los valores del sistema. Solo hay que lanzar este modo de funcionamiento sin necesidad de subir a la planta y hacer la calibración manual. El desarrollo del programa de calibración se realizó en el software Unity Pro XL por otro alumno el año pasado.

El competidor directo de este tipo de plantas solares son los captadores de tipo cilindro-parabólicos, y es por eso que a continuación, explicaremos cuales son las ventajas frente a estos:

• En el caso de la Fresnel el receptor es fijo, esto implica una mayor flexibilidad en la selección del fluido de transferencia térmica ya que no requiere de uniones móviles de alta presión.

• Mientras que el concentrador solar de un captador cilindro-parabólico es un único espejo con forma circular, en las plantas de tipo Fresnel, hay múltiples espejos que solo son ligeramente curvos. Esto abarata el precio del montaje de la instalación y del arreglo en caso de rotura de algún espejo.

• Mayor aprovechamiento del terreno y minimización de la sobra producida por los reflectores sobre sus adyacentes.

• Fácil acceso a los reflectores y partes móviles debido a su construcción cercana al suelo. Además, reduce las cargas ejercidas por el viento.

• Menores pérdidas térmicas y una mayor temperatura de funcionamiento.

• Entre los fluidos se encuentran las sales fundidas que permiten el almacenamiento de la energía.

2.3. Estado del arte del sistema de monitorización y control de una planta solar

Las aplicaciones móviles para la gestión y el control de las instalaciones fotovoltaicas han crecido en número y en calidad como la espuma. Son un aliado excelente para las empresas de servicios energéticos, industrias, administraciones públicas, así como para particulares.
Su sencilla interfaz ha convertido a estas aplicaciones en un recurso habitual para el control de equipamientos con consumo de energía. Son muy útiles para la gestión de la energía verde en

museos, teatros, hospitales, centros deportivos, educativos, centros comerciales, etc; así como en instalaciones fotovoltaicas aisladas, presentes en viviendas o explotaciones agropecuarias, por ejemplo.

El campo de las energías renovables no se ha quedado atrás en el mundo de las aplicaciones para móviles. Ya existen muchas aplicaciones sobre energías renovables puramente descriptivas e informativas. Sin embargo, hay otras más avanzadas con las que incluso se puede llevar a cabo el diseño para la implementación de las instalaciones productoras de energías renovables. Algunas aplicaciones de este tipo son: SOLARPE, CÁLCULOS ELÉCTRICOS, MyAMPERE, SOLAR CHECKER … [4]

Pero todas estas aplicaciones están enfocadas a pequeños sistemas de autoconsumo de viviendas unifamiliares o negocios, sobre todo para paneles fotovoltaicos. Si buscamos aplicaciones para campos solares de mayor envergadura vemos que se encuentra en casi todas las búsquedas la palabra "SCADA". Los Sistemas SCADA (*Supervisory, Control and Data Acquisition*) son un conjunto de aplicaciones software diseñadas para la supervisión, el control y la optimización de procesos industriales a distancia, pudiendo integrar datos recogidos desde diferentes procesos industriales y autómatas (PLCs) de manera local.

La aplicación de los sistemas SCADA en la industria, se emplea para monitorizar y controlar los procesos de producción y el funcionamiento completo de las plantas en tiempo real. Permiten acceder a las mismas mediante la comunicación digital con los distintos dispositivos de campo (controladores autónomos, autómatas programables, etc) y, mediante una interfaz gráfica (pantallas táctiles, cursores, ordenadores etc.), el operador puede controlar el proceso de forma automática, reaccionar a las alarmas y cambiar la configuración. Los activos y procesos gestionados son monitorizados en tiempo real proporcionando una visión completa del estado general de la empresa y permitiendo tomar decisiones tempranas basadas en datos.

Además, los sistemas SCADAS permiten ejecutar informes en tiempo real permitiendo visualizar, no solo lo que ocurre en cada momento, sino también lo que ha sucedido en el pasado y lo que pasará en el futuro con capacidad predictiva.

El futuro de la Industria 4.0, o industrias inteligentes, requiere de la digitalización y la interconexión de sus procesos mediante el uso de tecnologías como: Cloud monitoring, Big Data y análisis de datos, Machine Learning, IoT, ciberseguridad, Realidad Aumentada…
Solo de esta forma, las industrias podrán optimizar sus procesos y recursos, centrando los medios y un mayor esfuerzo en las tareas que requieren de inteligencia humana. [5]

Estas son algunas aplicaciones SCADA para Android [6]:

- ***OAS***: El conector de datos universal de OAS brinda un acceso incomparable a sus operaciones industriales y datos empresariales para conectividad, monitoreo, análisis y entrega. Nos podremos conectar directamente a PLC, servidores OPC, archivos, bases de datos y plataformas de IoT para crear sistemas SCADA de clase mundial o soluciones de automatización industrial. (no se especifica el precio, aunque tiene una prueba gratuita si te registras).

- ***Fernhill SCADA***: oferta SCADA escalable basada en una arquitectura cliente-servidor. Fácil de usar y configurar según defienden. Admite interfaces abiertas que incluyen OPC UA, OPC Classic, ODBC, MQTT y más. Funciona en múltiples plataformas, incluidas Windows, Linux, macOS, Android e iOS. Tiene periodo de prueba

gratuito. Se puede implementar cualquier cantidad de sistemas SCADA con una licencia de desarrollador de bajo coste.

Precio de la herramienta para desarrolladores:

$ 170 USD por licencia de 250 etiquetas
$ 380 USD por licencia de 1000 etiquetas
$ 520 USD por licencia de 2500 etiquetas

- ***CloudView***: sistema de monitoreo y administración de redes (NMS) basado en estándares universales. Se puede descubrir automáticamente, monitorear y realizar muchas funciones con dispositivos SNMP o TCP / IP de cualquier proveedor. Las funciones incluyen monitoreo / administración de redes, monitoreo de servidores, monitoreo de aplicaciones / IoT, monitoreo SCADA, automatización de operaciones de red, monitoreo de sitios web y mucho más. CloudView NMS se adapta a cualquier tamaño de red. Es compatible con múltiples plataformas, incluidos Windows, Linux, Mac OS y Raspberry Pi. Para los servidores, se admiten los modos sin agente y orientado al agente. $ 295 licencia ilimitada completa.

- ***Emerson Ovation***: Combinando el factor de forma y la robustez de un PLC con la potencia y la facilidad de integración del sistema de control Ovation, el controlador compacto es una plataforma de automatización confiable escalable para cualquier aplicación de generación de energía o agua / aguas residuales que necesite control para ubicarse cerca del equipo. La gestión remota de activos eficiente y segura desde cualquier lugar aumenta la confiabilidad y reduce los costos. Además de las aplicaciones nativas y avanzadas para optimizar las operaciones de la planta, Ovation admite monitoreo de vibración integrado, control de excitación del generador, SIS, huellas escalables para aplicaciones pequeñas o distribuidas, virtualización y simulación integrada. Sin información de precio o posible prueba gratuita.

- ***COOX***: es una solución MES + SCADA integrada y modular para la monitorización, control y optimización de la producción. En una plataforma única, entrega según sea necesario, funciones SCADA, gestión de órdenes de producción, ejecución de operaciones, gestión de recetas y rangos, trazabilidad del proceso, genealogía de productos y seguimiento del flujo de materiales, control de calidad, análisis del rendimiento de la producción e inteligencia de fabricación. Sin información de precio o posible prueba gratuita.

- ***SIMATIC SCADA***: Con su panorama de sistema abierto y escalable para la integración vertical de datos desde la producción al nivel MES / ERP hasta soluciones en la nube, SIMATIC SCADA ofrece una solución que está lista para los desafíos de hoy y mañana. Sin información de precio o posible prueba gratuita.

- ***VZRscada***: solución moderna que integra hardware, software y tecnologías de comunicaciones, con el fin de controlar y monitorear la infraestructura y los procesos críticos. Puede conectar sin problemas los sistemas de control existentes a VZRscada. Da apoyo a los clientes ya que se tiene acceso de emergencia remoto 24 horas al día, 7 días a la semana. Los cambios de programa y la optimización del proceso se pueden realizar de forma remota, lo que ahorra tiempo y costos asociados con las modificaciones en el sitio. Los sistemas PLC, la instrumentación y los controles de motores pueden conectarse a su portal de soporte seguro. VZRscada se puede conectar a sistemas existentes de varios fabricantes, por lo que si se tiene Allen-Bradley, GE, Siemens o Square-D, no hay que

reemplazar todo y empezar de nuevo, se puede conectar todo a VZRscada según defienden. Sin información de precio o posible prueba gratuita.

Estas aplicaciones suponen un coste en el caso de querer adquirirlas para la planta, y seguramente para una planta con las características de la ETSI o de cualquier otra planta de esta envergadura haría falta una aplicación más personalizada, lo que supondría un coste mayor.

3 SOFTWARE UTILIZADO

En este capítulo se presentan todos los programas que han sido necesarios para llevar a cabo el presente trabajo.

3.1. Unity

Unity es un motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, Mac OS, Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas. A partir de su versión 5.4.0 ya no permite el desarrollo de contenido para navegador a través de su plugin web, en su lugar se utiliza WebGL. Unity tiene dos versiones: Unity Professional (pro) y Unity Personal.

El éxito de Unity se debe a su enfoque en cubrir las necesidades de los desarrolladores independientes, los cuales, normalmente, no pueden permitirse crear su propio motor de juegos, ni pagar el acceso a las herramientas necesarias. La compañía busca "democratizar" el desarrollo de videojuegos y hacer que el desarrollo de contenido interactivo 2D y 3D sea lo más accesible a tantas personas en todo el mundo como sea posible. [7]

Se estudia la opción de desarrollar la aplicación en *Adroid studio* [8], pero se decide usar Unity ya que para Android es necesario tener nociones altas de java y su aprendizaje no sería cuantificable. Otro punto por el que se ha decidido usar Unity es por haberlo conocido en la ETSI, en una asignatura de control automático, en la que se nos enseñó una simulación de cintas transportadoras que despertaron nuestro interés tras saber que con ese mismo programa se podían hacer videojuegos. Supongo que, en aquel momento, era algo maravilloso y que creíamos inalcanzable, hacer videojuegos, pero la realidad, es que Unity es tan versátil que lo mismo te hace un videojuego que te simula un objeto o te programa una aplicación móvil. Pese a que cuando oímos “Unity” pensamos en videojuegos, se quiere demostrar que tiene muchas más funcionalidades y que se pueden crear aplicaciones móviles desde 0 con bajas nociones de programación, por lo que puede ser muy interesante para otras aplicaciones que puedan ser necesarias en los distintos departamentos de la ETSI.

3.2. Matlab

MATLAB es la abreviatura de MATrix LABoratory, es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, macOS y GNU/Linux. Entre sus prestaciones básicas se

hallan la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).
Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL. [8]

Tiene un hueco en esta memoria ya que se ha trabajado con él intentando conectarlo con la aplicación una vez desarrollada, y aunque sin éxito, más adelante en este documento se hablará de los intentos realizados para esta conexión y por qué han fracasado.

3.3. MYSQL y php.Myadmin

En un principio se pensó apoyarnos en una base de datos local para guardar los usuarios y claves de acceso para acceder a la aplicación, y para crear los datos en directo y el histórico que se va a mostrar. El problema que se planteaba era el de la necesidad de mantener continuamente encendido el ordenador que tuviera dicha base de datos. Este problema no sería importante para el momento de enseñar el funcionamiento de la aplicación en la defensa de este trabajo, pero sí lo es de cara a que en un futuro alguien retomase este trabajo o si tuviésemos un problema de conexión en el momento de la defensa. Por ello, y debido a las múltiples herramientas online de las que podemos disponer, se ha decidido usar una base de datos online y gratuita. La configuración es instantánea y usa phpMyAdmin para la administración.

La página nos proporciona un nombre de usuario y contraseña y una base de datos y podemos conectarnos a sus servidores MySQL de forma remota y gratuita y sin límites en la cantidad de consultas ni en el ancho de banda. La dirección y los datos son los siguientes:

https://remotemysql.com/login.php (Base de Datos):

- **usuario:** luisapial@gmail.com
- **pass:** TFMEnergyApp
- You have successfully created a new database. The details are below.
- Username: JeJBFPR3Uy
- Database name: JeJBFPR3Uy
- Password: 9CJzhSsyeh
- Server: remotemysql.com
- Port: 3306
- These are the username and password to log in to your database and phpMyAdmin

4 ESTUDIO Y DISEÑO DE LA APLICACIÓN

4.1. Requisitos

Según los estudios realizados por Jakob Nielsen, una interfaz de usuario que sea usable debe ser una interfaz que cumpla con los siguientes factores [9]:

- **Facilidad de aprendizaje**: El uso de una interfaz debe ser intuitivo, una persona de entre las que conforman su público meta debe ser capaz de entender su funcionamiento sin la necesidad de un manual.
- **Eficiencia**: El flujo de acciones debe ser corto y veloz, no requiriendo muchas etapas o mucho tiempo por parte del usuario, pero entregando los resultados esperados.
- **Sin requisición de memoria**: El usuario no debe necesitar de su capacidad de memorización. A pesar de haber dejado de utilizar la aplicación por un tiempo, es importante que logre llevar a cabo las tareas que requiere sin mayor problema o consulta adicional.
- **Índice de error**: La aplicación debe estar preparada para recuperarse en caso de errores, generar la menor cantidad posible de los mismos y ofrecer alternativas de corrección para el usuario en caso de que se presenten.
- **Experiencia satisfactoria**: El usuario debe sentirse victorioso tras haber llevado a cabo una tarea con la ayuda de la aplicación, es decir, debe sentir la satisfacción de haber logrado la meta que se proponía.
- Además, es importante acompañar la aplicación de un apartado estético atractivo y actual, acompañado de transiciones amigables y que hagan la experiencia más fluida y natural. Google pensó en ello con Android y publicó unas reglas de diseño llamadas Material Design.

Por tanto, teniendo en cuenta todo lo anterior, los requisitos para esta aplicación son los siguientes:

- La aplicación debe ser descargable gratuitamente en *play store*.
- La aplicación debe ser sencilla e intuitiva, para que cualquier persona sin altos conocimientos técnicos pueda utilizarla.
- Desde las distintas pantallas, desde todas ellas, debe haber un botón de cierre de sesión que nos lleve a la pantalla inicial previo al registro.
- Para acceder a los datos debemos tener usuario y contraseña que haya sido validado por el administrador (en este caso será la tutora de este trabajo, Amparo Núñez).

- La aplicación tendrá distintas pantallas por las que podremos ir moviéndonos para obtener los distintos datos:

    1. **Pantalla de acceso**: en ella habrá un desplegable para elegir el campo solar que se quiera observar, un recuadro para poner el usuario, otro para la contraseña, un botón para acceder y un enlace para solicitar el registro si no se tiene aún permiso para acceder. Si el registro es incorrecto se mostrará un error por pantalla.

    2. **Pantalla de inicio**: A ella se llegará tras un acceso de un usuario registrado. En ella se dará la bienvenida al campo solar solicitado, habrá un tablón de noticias reservado para poner en un futuro novedades que puedan ser interesantes, y tres botones que llevarán a 3 pestañas distintas, botón de datos en directo, botón de histórico de datos y botón de cerrar sesión.

    3. **Pantalla de registro**: A esta pantalla se accederá desde el enlace de registro de la pantalla de acceso. En dicha pantalla habrá 5 recuadros que rellenar: campo o campos solares a los que se solicita el acceso (ya que puede ser que sólo se necesite acceso a uno o a varios, pero no a todos, se seleccionarán a través de *check* en sus recuadros), correo electrónico, contraseña, confirmación de la contraseña y nombre completo y motivo por el cual se solicita el acceso. Será obligatorio rellenar todos los campos, si no se rellena saldrá un aviso y no se permitirá continuar hasta rellenar todos los huecos necesarios. Se encontrará también un botón de enviar que enviará dichos datos al administrador para que los confirme y dé de alta al usuario introduciéndolo en la tabla de la base de datos que crearemos para el registro.

    4. **Pantalla de histórico**: A esta pantalla se accederá desde el botón de histórico de datos de la pantalla de inicio. En esta pantalla habrá un botón de cierre de sesión para volver a la pantalla de registro y un botón de flecha para retroceder a la pantalla de inicio; 3 desplegables para elegir año, mes y día y un botón de recuperar datos que nos mostrará en un recuadro la información del día seleccionado (debes elegir una opción de los 3 desplegables de modo que solo se seleccione un día, si ese día no tiene datos no se mostrarán resultados).

    5. **Pantalla de datos en directo**: A esta pantalla se accederá también desde la pantalla de inicio, al pulsar el botón de datos en directo. Al igual que en la pantalla anterior, se tendrá un botón de cierre de sesión que lleve a la pantalla inicial. Habrá en esta pantalla un recuadro con los datos actuales del campo solar.

    6. **Pantalla de registro correcto**: A esta pantalla se accederá tras darle al botón de enviar de la pantalla de registro, en ella estará escrita la frase: "Su solicitud ha sido registrada con éxito. Se le comunicará por correo electrónico cuando sea aceptada ¡Gracias!". Y un botón de cerrar que llevará de nuevo a la pantalla de acceso.

- Las conexiones serán hacia una base de datos que simulará el campo solar de la ETSI. La base de datos debe crearse específicamente para este trabajo.

- Si se pulsa el botón “atrás” se irá a la pestaña abierta justamente anterior.

- Los datos de solicitud de registro llegarán al administrador por correo y éste los añadirá al listado de usuarios registrados y avisará a dicho usuario de que ya puede acceder si así lo desea.

- Se debe comprobar, al iniciar sesión en la pantalla de acceso, que el usuario y la contraseña coinciden con un usuario de la tabla que contiene todos los usuarios/contraseñas que han sido dados de alta. Dicha tabla es dinámica y puede ser modificada por parte del administrador tanto para añadir usuarios como para quitarlos.

Se debe cuidar la apariencia de la aplicación para que sea funcional pero también agradable a la vista.

La planificación de este trabajo se ha realizado con un tablero Kanban realizado en Trello. Kanban ha ido ganando popularidad durante las últimas décadas. Nació para aplicarse a los procesos de fabricación y con el tiempo se convirtió en un territorio reclamado por los desarrolladores de software. Últimamente, ha empezado a ser reconocido por las entidades empresariales de diferentes ámbitos. [10].

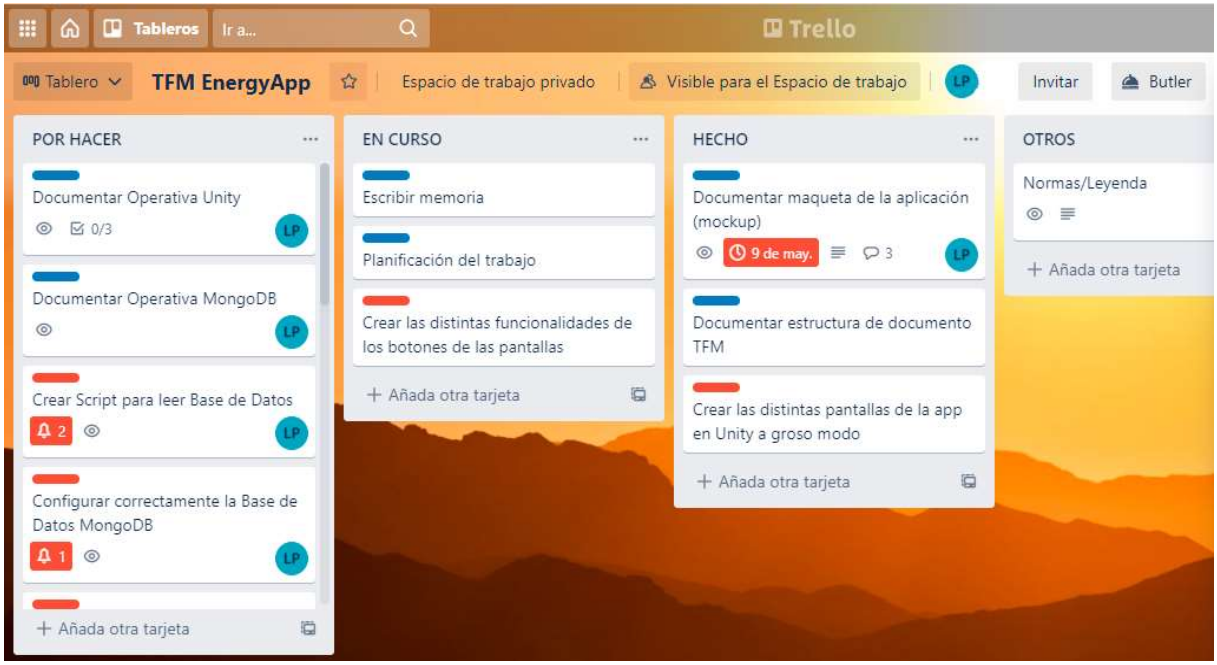

Ilustración 7 - Planificación con un tablero Kanban

Con esta herramienta se puede planificar las distintas tareas necesarias, así como pasarlas a "en curso" o terminadas cuando corresponde. También se pueden añadir etiquetas (en este caso diferenciamos entre documentación del trabajo y el desarrollo de la aplicación). También sirve para añadir comentarios a las tareas y no perder el hilo si pasa tiempo sin retomarla. Se ha querido dedicar un pequeño apartado para esto, para mostrar las distintas herramientas que se han usado en todo el trascurso de la realización de este trabajo, y no solo en la parte más técnica, sino también en la parte previa de organización, ya que somos conocedores de su gran importancia.

Buscando información acerca de cómo afrontar el desarrollo de una aplicación móvil (y de cualquier proyecto en general), se comenta la importancia de una buena planificación inicial y las herramientas que pueden ser de gran utilidad para un correcto trascurso y meta. Se ha decidido usar Trello ya que es una herramienta que se utiliza en el trabajo para organizar el día a día y no olvidar las tareas que se tienen pendientes o en curso. Las tareas pueden ser compartidas entre varios miembros y así vemos lo que otros han hecho sobre ellas y no realizan lo mismo dos personas a la vez por no saber que otro compañero ha cogido esa tarea ya. En este caso no tiene esta funcionalidad pues es únicamente para uso de la persona que desarrolla la aplicación, pero es interesante de cara a planificar este proyecto y marcar unos puntos de control en las fechas que se consideran críticas. También se ha querido mostrar otra herramienta que no se aprende en la ETSI, para tener una alternativa al *Microsoft Project* que sí se estudia en algunos grados, para que todo estudiante que lea este TFM pueda conocer otras opciones, que es más interesante que realizar la planificación con una herramienta que ya se maneja.

A lo largo de la realización de esta aplicación se han tenido que tomar diversas decisiones. En un principio se pensó hacer la aplicación con *Android Studio*, que es el entorno de desarrollo integrado oficial para la plataforma Android. Tenía varias ventajas como mucha información gratuita en la red y de fácil acceso, pero por otro lado se basa en java que no es un lenguaje que domine la estudiante que ha realizado este trabajo, por lo que sería necesario aprenderlo para poder empezar a crear aplicaciones de cierta complejidad. Como el tiempo empleado para aprender java desde cero es bastante alto y no sería cuantificable para el desarrollo de esta aplicación, objeto de Trabajo Fin de Máster, finalmente se ha decidido utilizar Unity por su versatilidad sin necesidad de grandes nociones de lenguajes informáticos para obtener una aplicación funcional en un período de tiempo asumible y poder dedicar tiempo a hacerla lo más completa posible sin perder el tiempo en la formación previa necesaria, que repetimos, sería difícil cuantificar.

4.2. Maqueta

Siguiendo los requisitos iniciales, se realiza un *mockup* (boceto) de cómo quedarían las distintas pantallas de la aplicación en un primer diseño. Se ha usado para tal fin *Drawio*, que es una aplicación online gratuita con diversas opciones desde bocetos en blanco hasta prediseños para los usos más comunes. En este caso se decidió empezar desde 0 para asemejar a la pantalla del móvil y todos sus partes que no siguen ningún patrón típico, como puede ser un bocadillo de conversación, una nube de pensamiento o unas viñetas. La imagen queda así:

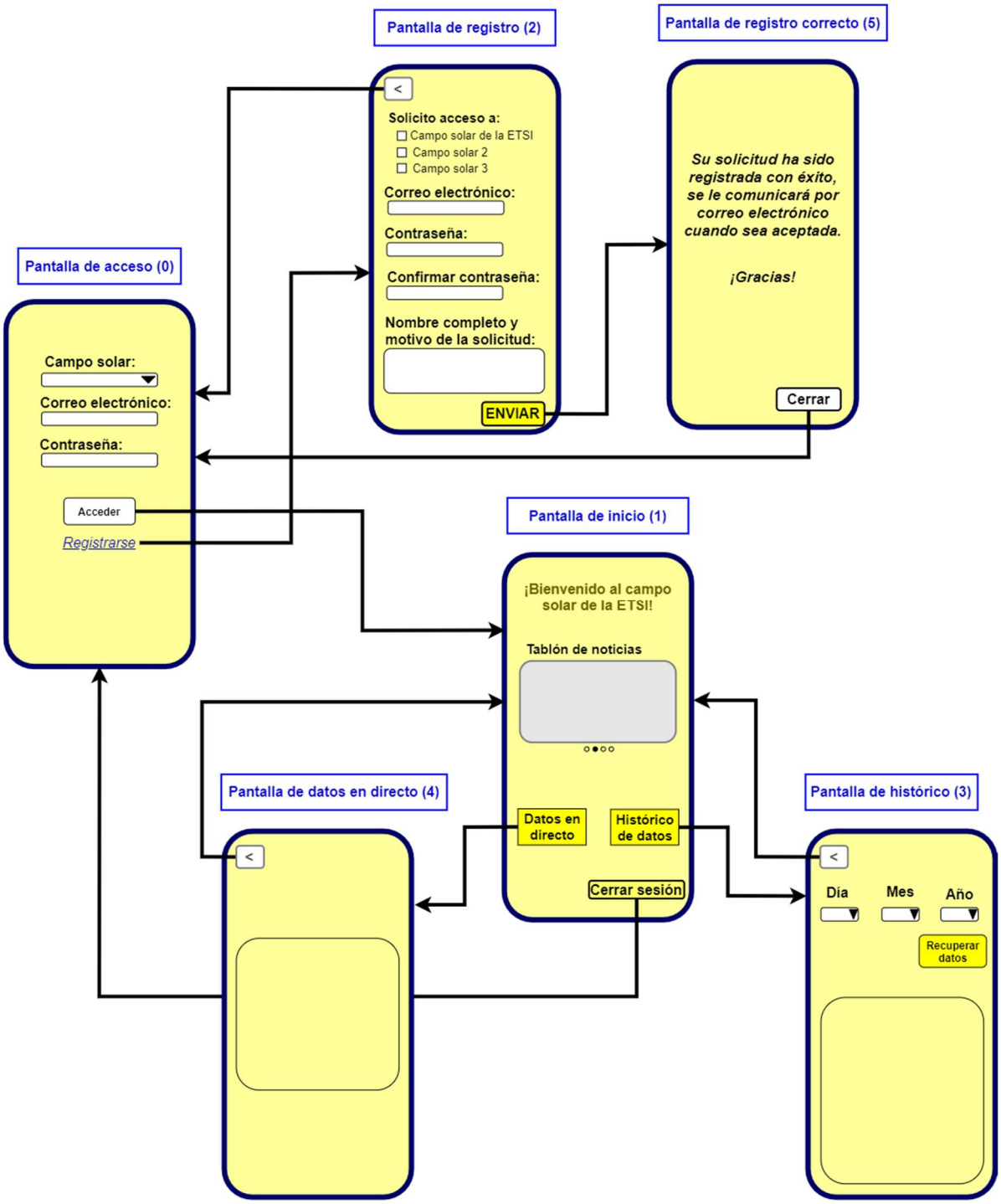

Ilustración 8 - Mockup de la aplicación

Como se puede observar, desde la pantalla de acceso podemos ir a dos pantallas, la de “inicio” si nos registramos con éxito o la de “registro” si pulsamos el botón de registrarse. Desde registrarse, y tras rellenar todos los campos y darle a enviar, se llega a la pantalla de “registro correcto” y al cerrar volvemos a la primera pantalla de acceso. Si se accede a la pantalla de inicio podemos leer las noticias, pulsar cerrar sesión para volver a la pantalla de acceso o dar a los botones de datos en directo o histórico de datos. Si pulsamos al botón de “datos en directo” accedemos a la pantalla de datos en directo, desde ella podemos dar al botón de atrás para volver a la pantalla de inicio o al botón de cerrar sesión para volver a la pantalla de acceso. De igual modo si pulsamos el botón

de “histórico de datos” accedemos a la pantalla de histórico de datos, donde podremos elegir la fecha que queramos consultar con los 3 desplegables que hay, y al dar al botón “recuperar datos” aparecerán abajo los datos solicitados, de igual modo podemos pulsar el botón de la flecha que nos lleva hacia atrás a la pantalla de inicio o el botón de cerrar sesión para volver a la de acceso.

A continuación, se muestran todas las pantallas anteriormente descritas:

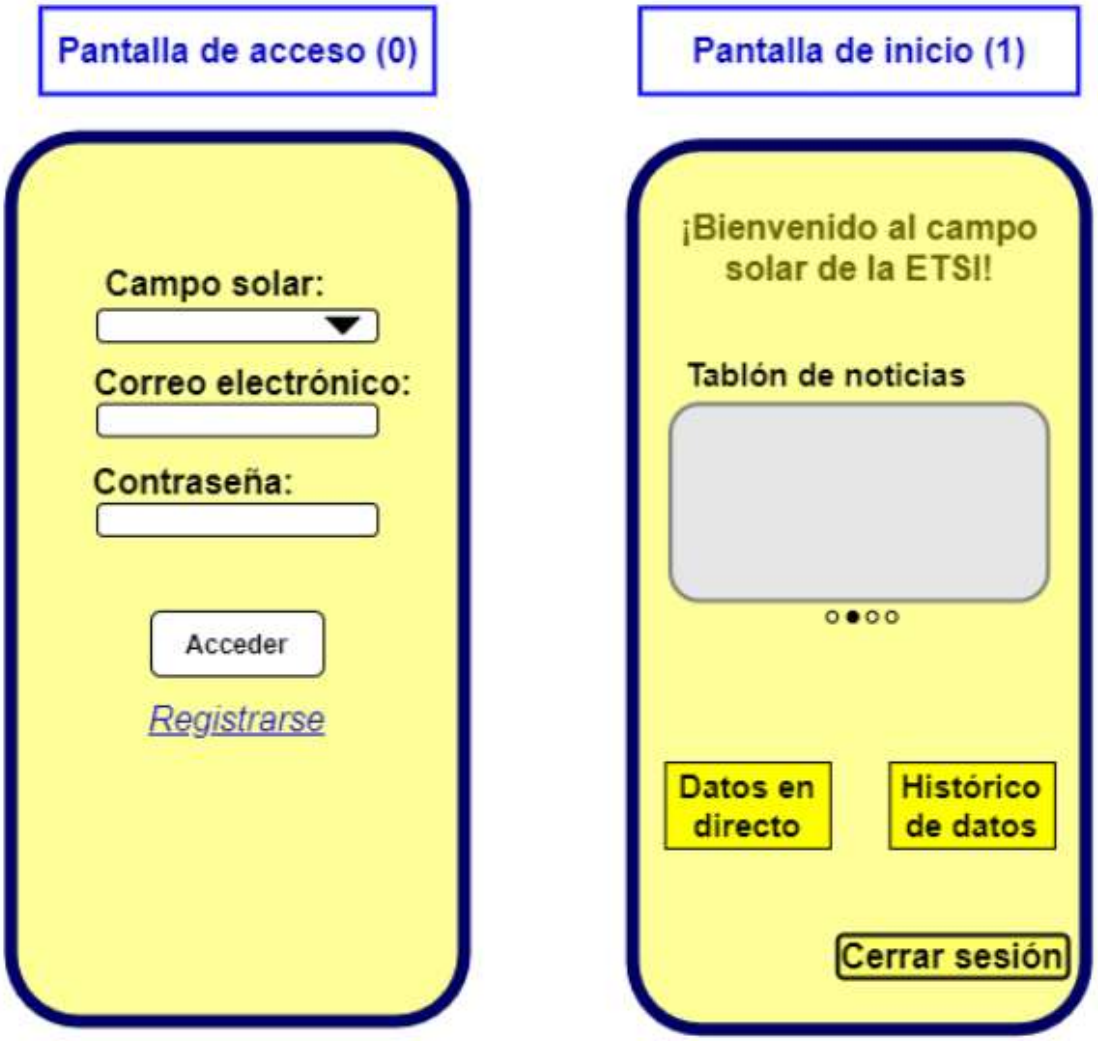

Ilustración 9 - Pantallas de acceso e inicio

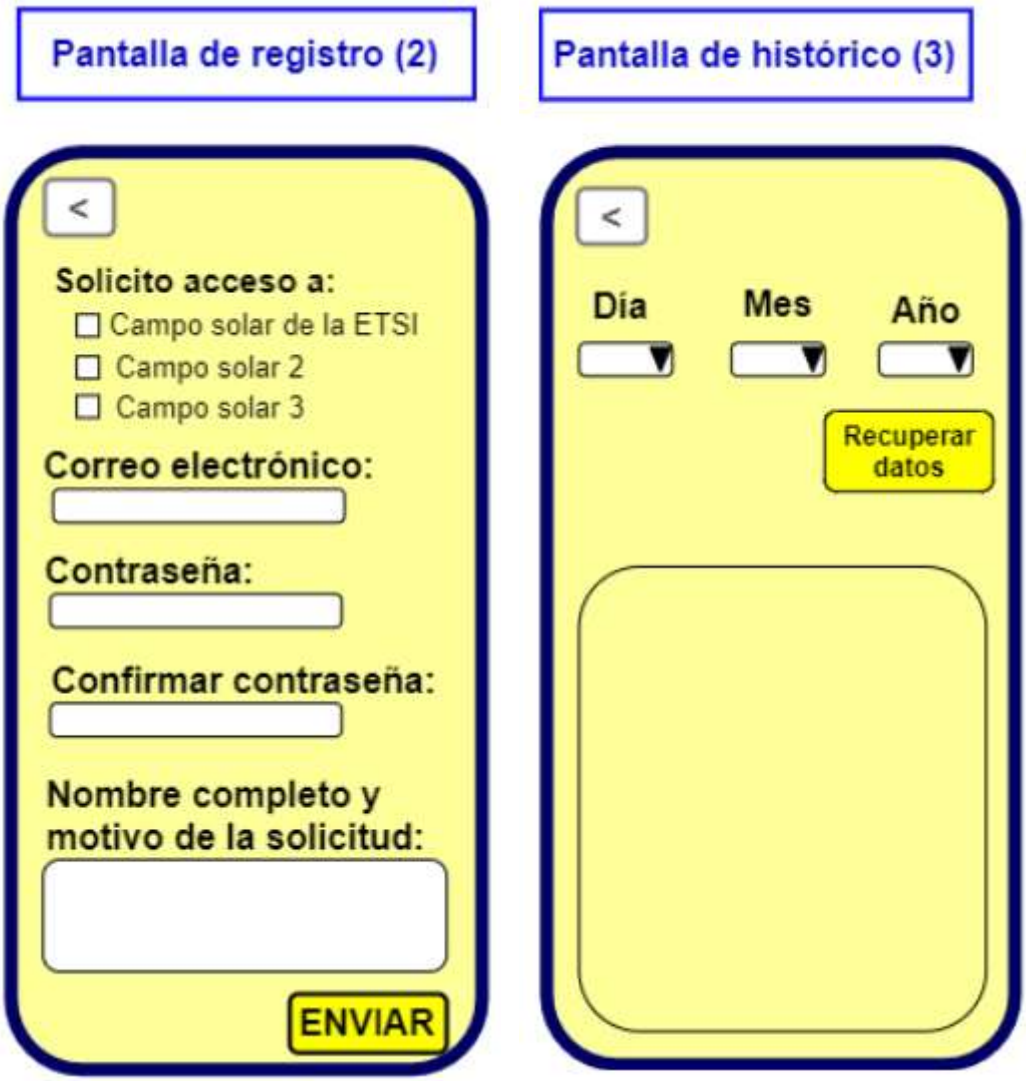

Ilustración 10 - Pantallas de registro e histórico

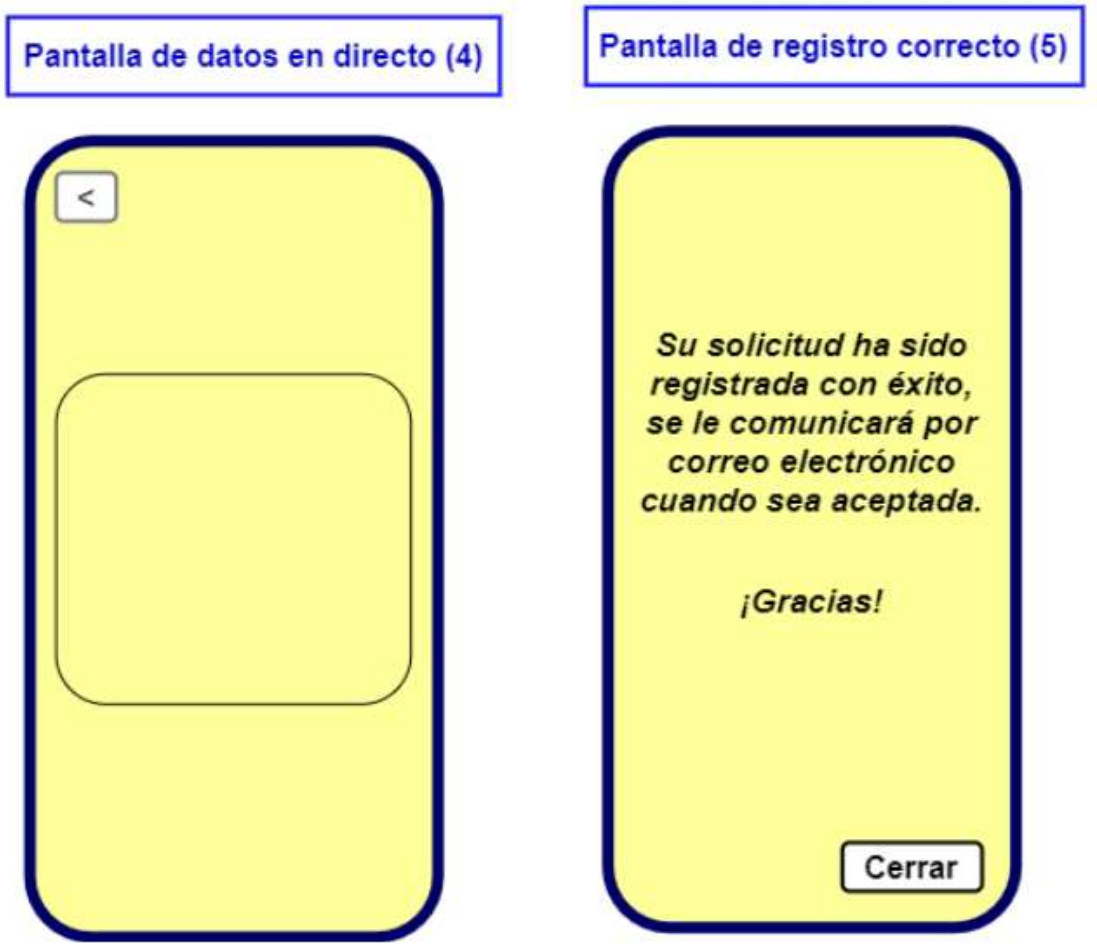

Ilustración 11 - Pantallas de directo y registro correcto

4.3. Nombre y logotipo de la aplicación

Se decide llamar a la aplicación "EnergyApp" por ser un nombre corto y fácil de recordar. También se diseña el logo siguiente:

Ilustración 12 - Logo de la aplicación EnergyApp

5 DESARROLLO DE LA APLICACIÓN

En este apartado se presenta la descripción con detalle del funcionamiento de cada una de las partes de la aplicación.

5.1. Esquema de la aplicación

El esquema de la aplicación es el siguiente:

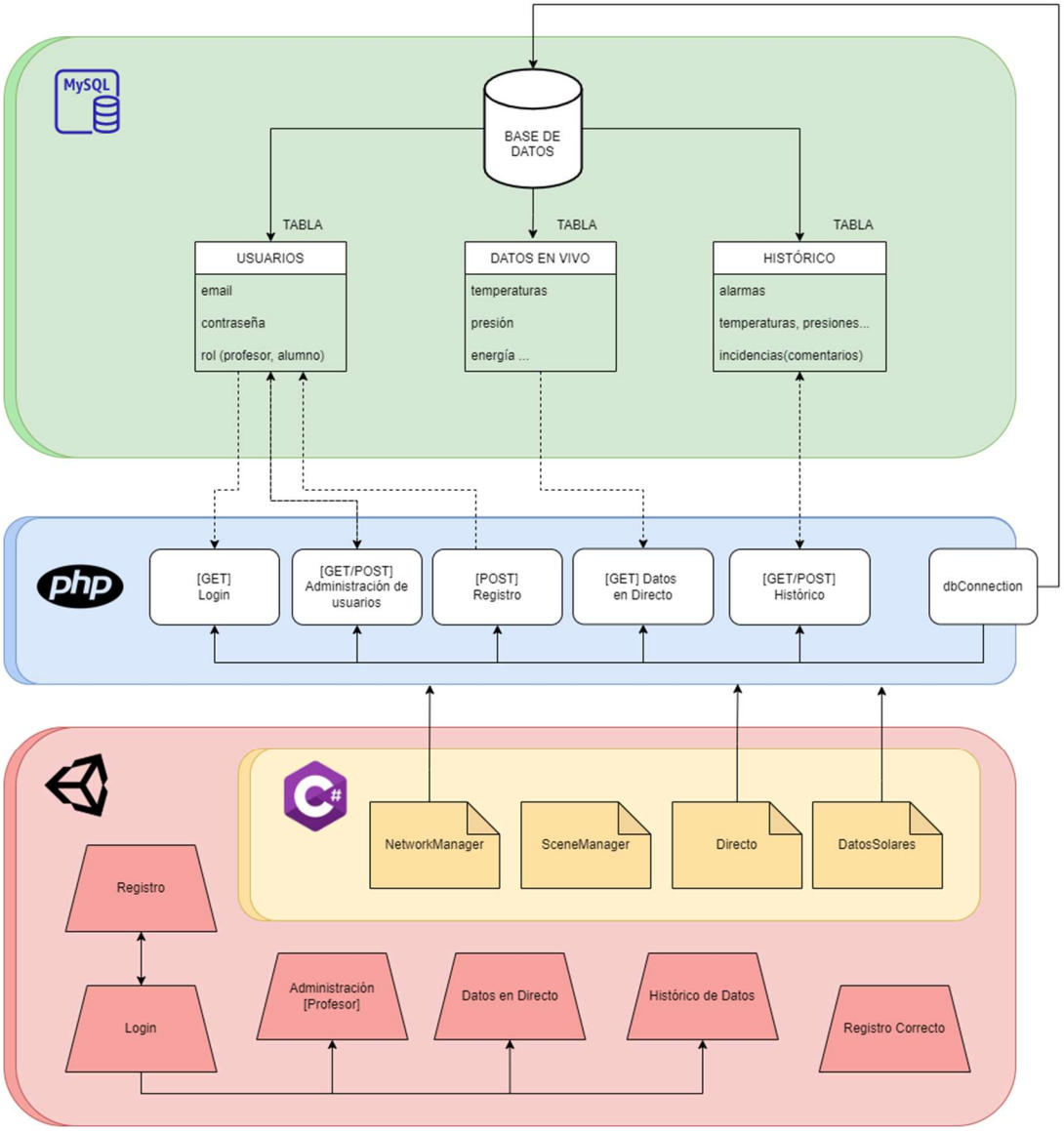

Ilustración 13 - Diagramas y relaciones de la aplicación

En el esquema se representa la división de la aplicación. La parte rosa es Unity, donde está la parte visual, con lo que el usuario interactúa. Esta parte conecta con la verde (la parte de los datos de la planta, en este caso en una base de datos de MySQL alojada en la web) gracias a php (la azul), que funciona como nexo. Gracias al fichero dbConnection, se puede conectar con dicha base de datos y gracias al resto de ficheros php, extraer los datos necesarios que mandar a la aplicación para que estos se puedan visualizar.

5.2. Capas de la aplicación

EnergyApp se ha diseñado para satisfacer el uso por parte del usuario, por lo que gran parte del contenido se centra en la interfaz gráfica y escenas de esta aplicación.

Este diseño intuitivo se consigue separando las distintas capas que lo componen, como son la de modelado de datos o **capa de datos** y el servidor de peticiones o **capa de negociación** que conforman el "backend" y las escenas con la interfaz de Unity o **capa de presentación** que conforman la parte del "frontend".

- **Capa de datos:** La capa de datos está formada por el propio servidor de base de datos externo basado en MySQL, el cual contiene diferentes tablas a modo de simulación de los datos que se desean extraer a futuro del campo solar. Las tablas son las siguientes:

    - **Tabla de usuarios**: En esta tabla se recogen los usuarios que tienen acceso (o lo han solicitado) a la aplicación. Se compone de los siguientes campos:
        - correo: Email de usuario para el acceso
        - password: Contraseña para el acceso
        - rol: campo enumerado con alternativas 'profesor' y 'alumno'
        - acceso: campo enumerado con alternativas 'si' y 'no'

    - **Tabla de datos en directo**: En esta tabla se recogen los datos en directo de la planta, es decir, la foto del momento en el que se lanza la consulta. Los campos que la componen son los siguientes:
        - TSC: Temperatura de salida del colector expresada en ºC
        - TEC: Temperatura de entrada del colector expresada en ºC
        - TST: Temperatura de salida del tubo expresada en ºC
        - TAMB: Temperatura ambiente expresada en ºC
        - motores: binario entre 0 y 1 que determina si el motor está apagado o encendido respectivamente. Se han registrado en esta tabla un total de 22 motores diferentes
        - tanque: Carga del tanque de almacenamiento expresada en kW.h
        - energia: Energía generada expresada en kW.h
        - presión: Presión expresada en bares

    - **Tabla de datos históricos**: En esta tabla se recogen los datos de la planta de días pasados. Los campos son los siguientes:
        - fecha: fecha del registro
        - salidaSol: hora de salida del Sol para la fecha indicada
        - puestaSol: hora de puesta de Sol para la fecha indicada
        - alarmaTs: binario entre 0 y 1 que determina si durante la fecha de entrada se ha activado o no la alarma de temperatura

        - alarmaP: binario entre 0 y 1 que determina si durante la fecha de entrada se ha activado o no la alarma de presión
        - alarmaM: binario entre 0 y 1 que determina si durante la fecha de entrada se ha activado o no la alarma de motores
        - TSCMedia: temperatura de salida del colector media durante el día correspondiente a la fecha expresada en ºC
        - TECMedia: temperatura de entrada del colector media durante el día correspondiente a la fecha expresada en ºC
        - TSTMedia: temperatura de salida del tubo media durante el día correspondiente a la fecha expresada en ºC
        - TAmbMedia: temperatura de ambiente media durante el día correspondiente a la fecha expresada en ºC
        - presionMed: presión media durante el día correspondiente a la fecha expresada en bares
        - acumulaciónMax: máxima carga del tanque durante el día correspondiente a la fecha expresada en kW.h

- **Capa de negociación**: La capa de negociación es la encargada, mediante peticiones de servicio, de extraer la información solicitada de la base de datos. Hace las veces de enlace con el servidor de base de datos. Para el proyecto, se encuentra alojado en un servidor externo accesible desde cualquier parte y está basado en PHP.

    - dbConnection.php: Fichero encargado de establecer conexión con la base de datos. Aquí se especifica el nombre del servidor, el nombre de la base de datos, el nombre de usuario con acceso y la contraseña.
    - loginUsuario.php: Fichero encargado de realizar las llamadas GET para la consulta y verificación de usuario y contraseña de acceso a la aplicación.
    - existeUsuario.php: Fichero encargado de realizar la llamada GET para comprobar si el usuario existe en base de datos previo a su creación y otras llamadas.
    - crearUsuario.php: Fichero encargado de realizar las llamadas POST para la introducción de datos de nuevos usuarios en la aplicación.
    - obtenerDatos.php: Fichero encargado de realizar las llamadas GET para la consulta de datos en la pantalla de históricos.
    - introducirDatos.php: Fichero encargado de realizar las llamadas POST para la introducción de comentarios de incidencia para las entradas del histórico de datos.
    - profesoresAcceso.php: Fichero encargado de realizar la llamada GET para extraer un listado de docentes con acceso a la aplicación, con la finalidad de enviar correos de notificación en las altas de nuevo usuario.
    - obtenerDirecto.php: Fichero encargado de realizar las llamadas GET para la consulta de datos en la pantalla de datos en directo.

- **Capa de presentación**: La capa de presentación está formada por los elementos de interfaz gráfica de usuario (GUI) que componen la aplicación. En Unity, motor sobre el que se ha desarrollado, se denominan “escenas” y vienen acompañadas de scripts en lenguaje C# para la realización de acciones y llamadas.

    - Login: Escena orientada a facilitar el acceso a la aplicación por parte de los usuarios (docente/estudiante).

- Registro: Escena orientada a facilitar el registro en la aplicación por parte de los usuarios (docente/estudiante).
- RegistroCorrecto: Escena orientada a informar al usuario sobre la creación exitosa de un usuario, a la espera de confirmación por agentes administradores de la aplicación.
- Administración: Escena orientada a administrar y gestionar el acceso de usuarios que aún no han sido validados. Esta pantalla sólo es accesible si el rol del usuario es docente.
- Datos en Directo: Escena orientada a facilitar los datos en vivo sobre el campo solar seleccionado en el momento de la consulta.
- Histórico de datos: Escena orientada a facilitar los datos sobre el campo solar seleccionado en una fecha seleccionada. Esta escena permite introducir una incidencia a modo de comentario a los usuarios con perfil docente.

5.3. Introducción al diseño en Unity

En este punto se explicará cómo se fue desarrollando la aplicación ordenadamente y se comentarán los cambios y mejoras que se han ido haciendo respecto al diseño inicial, que como se ha dicho, no dejaba de ser un boceto previo al desarrollo.

El Canvas es el área donde todos los elementos UI deben estar. Es un Game Object con un componente Canvas en él, y todos los elementos UI deben ser hijos de dicho Canvas.

Cuando se crea un nuevo elemento UI, tal como una Image (imagen) utilizando el menú GameObject > UI > Image, automáticamente crea un Canvas si no hay uno en la escena ya creado. El elemento UI es creado como un hijo de este Canvas.

El área Canvas se muestra como un rectángulo en la Vista de Escena (*sceneView*). Canvas utiliza el objeto EventSystem para ayudarle al sistema de mensajes. [11]

Se van a describir solo los elementos que se van a utilizar en esta aplicación para no excedernos en la formación, estos elementos, como se ha comentado, pertenecen a la interfaz de usuario (UI). Las características de cada uno son las siguientes [12]:

- **Button**: Es un botón estándar que se puede oprimir con el fin de (activar/desactivar) un evento.

- **Image**: Muestra un Sprite para el sistema de UI. Los *Sprites* son objetos gráficos 2D usados para los elementos de la aplicación o el videojuego. Los gráficos se obtienen de imágenes bitmap - Texture2D. La clase Sprite identifica principalmente la sección de la imagen que se debe usar para un sprite específico.

- **Input Field**: Devuelve una etiqueta simple a un campo de entrada con el que se puede interactuar.

- **Panel**: Define un área y sirve para unir varios elementos y poder moverlos a la vez, ya que los haces pertenecer a este panel (colgar de él).

- **Text**: El *Graphic* predeterminado para dibujar los datos fuente a la pantalla, sirve para poner un objeto texto en la pantalla.

- **Scroll Rect**: Se utiliza cuando el contenido ocupa más del espacio necesitado cuando se muestra en un área pequeña. El *Scroll Rect* proporciona la funcionalidad para desplazarse hacia este contenido. Usualmente es combinado con un *Mask* con el fin de crear una vista de *scroll*, dónde solamente el contenido desplazable dentro del Scroll Rect es visible. Éste también puede ser combinado con uno o dos *Scrollbars* que pueden ser arrastrados para desplazarse horizontal o verticalmente.

- **Dropdown:** (Desplegable) Puede ser utilizado para permitirle al usuario escoger una sola opción de la lista de opciones. El control muestra la opción actualmente escogida. Una vez es seleccionada, abre la lista de opciones paraescoger una nueva opción. Al elegir una nueva opción, la lista se cierra, y el control muestra la nueva opción seleccionada. La lista también se cierra si el usuario hace *click* en el control en sí, o cualquier otro lado del Canvas.

5.4. Detalles y relación de escenas en Unity

Las escenas en Unity contienen los entornos y menús de la aplicación. Cada archivo de escena es un nivel o módulo único. En cada escena se colocan de forma independiente los elementos, botones, textos, imágenes, script y datos que la componen, lo que hace que la aplicación se construya por partes.

A continuación, se muestra el diagrama final de relación de escenas:

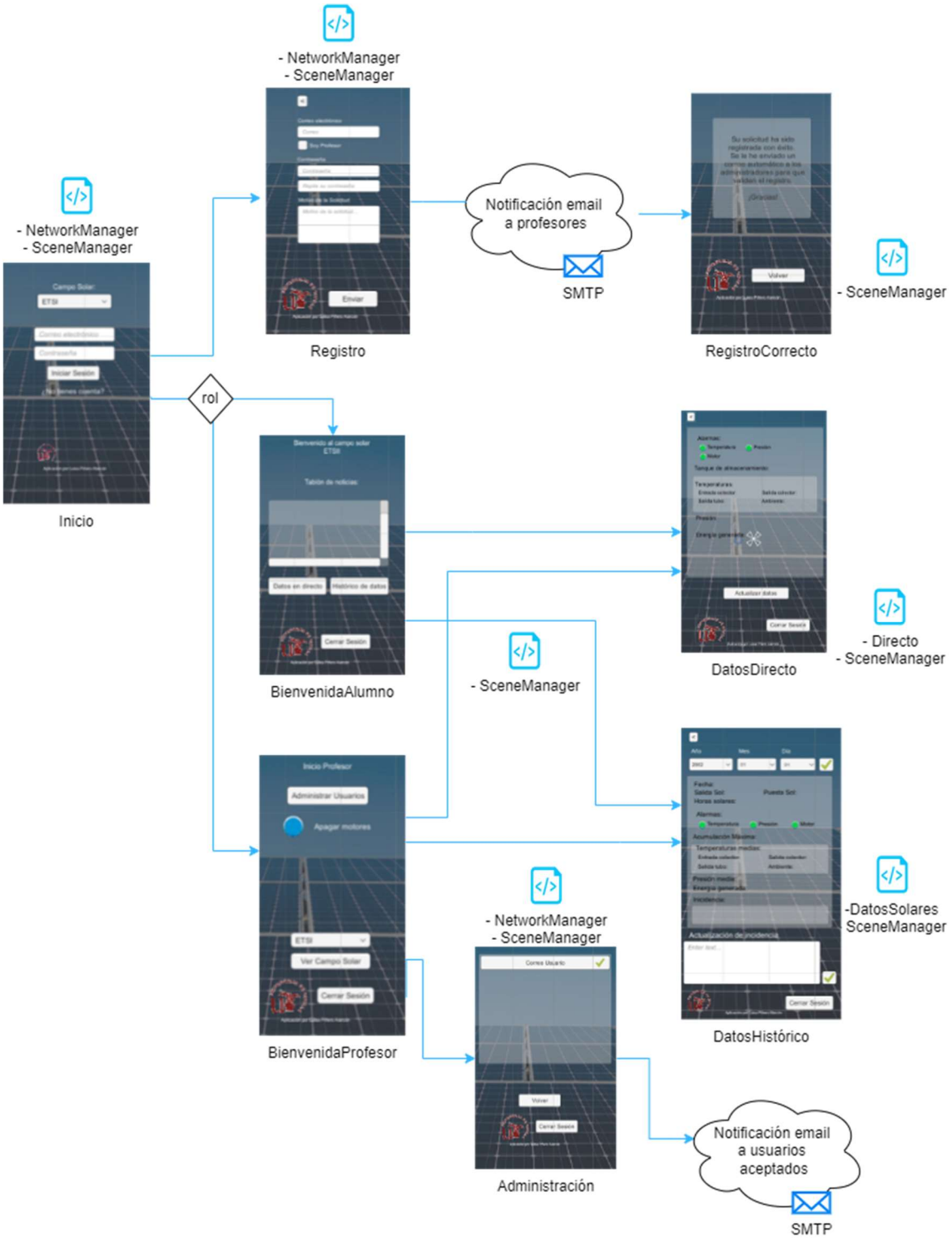

Ilustración 14- Diagrama final de la aplicación

Las escenas pueden llamar, a su vez, a otras escenas a través de acciones (en forma de botón). La carga de escenas cuando se produce el click sobre un botón la realiza el script "SceneManager.cs". Este script es transversal a la aplicación y no establece conexión con la base de datos o el servidor de negociación.

El resto de *scripts* C# que encontramos para hacer posible la lógica de *EnergyApp* son los siguientes:

- NetworkManager.cs: Este script realiza las operaciones de carga y comprobación de usuario en el inicio de sesión, así como las verificaciones e inserción de datos cuando se trata de dar de alta un usuario.

- Directo.cs: Este script se encarga de mostrar los datos en tiempo real sobre los elementos del campo solar. Por seguridad y para evitar el cierre del servidor de negociación, se ha inhabilitado la actualización de datos automática recursiva, ya que el servidor detecta múltiples consultas en breve espacio de tiempo y procede a su desactivación. En su lugar se refrescarán los datos mediante el click de un botón de actualizar datos.

- DatosSolares.cs: Este script se encarga de mostrar los datos extraídos de la tabla de históricos y reflejarlos en pantalla. También es el encargado de registrar la actualización de incidencia si la hubiera en el caso de usuarios con rol "profesor".

- Otros: La aplicación se compone también de otros scripts menores que realizan operaciones sencillas que han sido aislados para mantener el código ordenado. Es el caso de las funciones que desactivan todos los motores desde la pantalla de BienvenidaProfesor.

El detalle funcional de las escenas que componen *EnergyApp* es el siguiente:

- Login (SampleScene): Es la escena principal (de inicio) y la que se carga en el arranque de la aplicación. Tiene como objetivo el filtrar el acceso al contenido mediante un sistema simple de inicio de sesión. Los elementos que la componen son los siguientes:

    - Correo (InputField): Entrada de texto donde introducir el correo/email del usuario que desea acceder. Es clave primaria y su valor es único, es decir, no pueden coexistir dos usuarios con el mismo correo.
    - Password (InputField): Entrada de texto donde introducir la contraseña correspondiente al correo de usuario seleccionado anteriormente.
    - Iniciar Sesión (Button): Botón cuya acción es la de realizar la comprobación de correspondencia de usuario y contraseña.
    - ¿No tienes cuenta? (Button): Botón cuya acción es la de remitir al usuario a la pantalla de Registro.

- Registro: Es la escena dedicada al registro de un nuevo usuario y está compuesta por los siguientes elementos:

    - Correo (inputField): Entrada de texto donde introducir el correo/email a registrar para un nuevo usuario. El correo introducido no puede existir previamente.
    - Soy Docente (Toggle): Check seleccionable para discernir entre usuarios con rol "alumno" (inactivo) y con rol "profesor" (activo).
    - Password (InputField): Entrada de texto donde introducir la contraseña deseada para el usuario registrado en el elemento 'Correo'.
    - PasswordRepeat (InputField): Entrada de texto donde introducir de nuevo a modo de confirmación la contraseña deseada para el registro.
    - Motivo (InputField): Entrada de texto donde introducir el motivo por el que el usuario solicita acceso.
    - Enviar (Button): Botón cuya acción es la de registrar al usuario. Previamente comprueba que el 'Correo' no existe, que las contraseñas coinciden y que ambos campos cumplen las normas definidas como que el correo sea válido y que la contraseña tenga más de ocho caracteres. Una vez se ha comprobado que se cumplen los requisitos, se carga la escena de RegistroCorrecto.

- RegistroCorrecto: Escena informativa que notifica al usuario del éxito en el registro. Se compone de:

    - Mensaje (Text): Texto informativo del éxito en el registro.
    - Volver (Button): Botón cuya acción carga la escena Login.

- BienvenidaUsuario: Esta escena se corresponde con la escena de inicio tras el login para un usuario con rol “alumno”. Los elementos que aparecen son:

    - TablonNoticias (Text): Texto informativo con noticias destacadas.
    - Datos en directo (Button): Botón cuya acción carga la escena DatosDirecto.
    - Histórico de datos (Button): Botón cuya acción carga la escena HistoricoDatos.
    - Cerrar Sesión (Button): Botón destinado a cerrar el acceso al usuario y la carga de la escena Login.

- BienvenidaProfesor: Esta escena se corresponde con la escena de inicio tras el login para un usuario con rol “profesor”. Los elementos que aparecen son:

    - Administrar usuarios (Button): Botón que carga la escena de Administración.
    - Apagar motores (Button): Botón destinado a apagar todos los motores del campo en caso de emergencia.
    - Selección de Campo (DropDown): Lista seleccionable de campos disponibles para su acceso a datos.
    - Ver Campo (Button): Botón que realiza la carga de la escena de visualización del campo solar seleccionado.

- Administración: Esta escena recoge los usuarios que han sido registrados, pero aún no han sido validados. Los elementos que se muestran son:

    - UsuarioAValidar (Button): Botón con el nombre del usuario a validar para el acceso a la aplicación. Se genera una lista de botones de este tipo con tantos elementos como usuarios a validar haya en cola. Al pulsar sobre el botón, se valida el acceso para el usuario a la par que se le envía un email de confirmación a su dirección de correo electrónico.
    - Volver (Button): Botón cuya acción carga la escena BienvenidaProfesor.
    - Cerrar Sesión (Button): Botón destinado a cerrar el acceso al usuario y la carga de la escena Login.

- DatosDirecto: Esta escena se encarga de la visualización de los datos en vivo que existen actualmente en el campo solar. Los elementos que componen la escena son:

    - alarmaTs (Image): Imagen que representa el estado de la alarma de temperatura. Ésta salta (se vuelve de color rojo) si la temperatura de salida del colector es mayor de 190ºC, de lo contrario permanece en color verde.
    - alarmaM (Image): Imagen que representa el estado de la alarma de motor. Ésta salta (se vuelve color rojo) si hay algún motor apagado, indicando el texto del motor o los motores que se encuentran inoperativos. Si todos los motores están encendidos, el indicador luce de color verde.
    - alarmaP (Image): Imagen que representa el estado de la alarma de presión. Ésta salta (se vuelve de color rojo) si la presión indicada es mayor de 15 bares.
    - tanqueAlmacenamiento (Text): Texto que indica la carga actual del tanque expresada en kW.h.
    - tempTsc (Text): Texto que indica la temperatura actual de salida del colector. Se expresa en ºC.

- tempTec (Text): Texto que indica la temperatura actual de entrada del colector. Se expresa en ºC.
- tempTst (Text): Texto que indica la temperatura actual de salida del tubo. Se expresa en ºC.
- tempAmb (Text): Texto que indica la temperatura actual ambiente. Se expresa en ºC.
- presion (Text): Texto que indica la presión actual. Se expresa en 'bar'.
- energiaGenerada (Text): Texto que indica la energía generada actual. Se expresa en kW.h.
- Actualizar Datos (Button): Botón cuya acción actualiza la salida de datos en pantalla.

- DatosHistorico: Esta escena refleja los datos relativos a una fecha concreta almacenada en el histórico. Los elementos que la componen son:

  - dia (DropDown): Elemento desplegable que lista los días a elegir para la extracción de los datos.
  - mes (DropDown): Elemento desplegable que lista los meses a elegir para la extracción de los datos.
  - año (DropDown): Elemento desplegable que lista los años a elegir para la extracción de los datos.
  - aceptarFecha (Button): Botón cuya acción permite realizar la carga de datos para el combo de día-mes-año introducido en los campos anteriores.
  - alarmaTs (Image): Imagen que representa el estado de la alarma de temperatura. Ésta salta (se vuelve de color rojo) si la temperatura de salida del colector saltó durante el día elegido, de lo contrario permanece en color verde.
  - alarmaM (Image): Imagen que representa el estado de la alarma de motor. Ésta salta (se vuelve color rojo) si la alarma de algún motor apagado saltó durante el día elegido. Si todos los motores funcionaron durante todo el día elegido, el indicador lucirá de color verde.
  - alarmaP (Image): Imagen que representa el estado de la alarma de presión. Ésta salta (se vuelve de color rojo) si la presión superó los 15 bares durante el día elegido.
  - acumulacionMax (Text): Texto que indica la capacidad máxima que alcanzó el tanque durante el día elegido expresada en kW.h.
  - tempTscMed (Text): Texto que indica la temperatura media de salida del colector durante el día elegido. Se expresa en ºC.
  - tempTecMed (Text): Texto que indica la temperatura media de entrada del colector durante el día elegido. Se expresa en ºC.
  - tempTstMed (Text): Texto que indica la temperatura media al de salida del tubo durante el día elegido. Se expresa en ºC.
  - tempAmb (Text): Texto que indica la temperatura media ambiente durante el día elegido. Se expresa en ºC.
  - presionMed (Text): Texto que indica la presión media obtenida durante el día elegido. Se expresa en 'bar'.
  - energiaGenerada (Text): Texto que indica la energía media generada durante el día elegido. Se expresa en kW.h.
  - Actualizar Datos (Button): Botón cuya acción actualiza la salida de datos en pantalla.
  - Incidencia (Text): Texto que refleja si se introdujo alguna incidencia para el día seleccionado.
  - ComentarioIncidencia (InputField): Entrada de texto que recoge a modo de comentario la incidencia. Al pulsar sobre el botón Actualizar, se guarda la incidencia para el día seleccionado.

- Actualizar (Button): Botón cuya acción habilita la inserción de la incidencia comentada en ComentarioIncidencia.

5.4.1. Creación de las escenas en Unity

Para crear cada una de las pantallas de las que se compone nuestro programa, debemos hacer lo siguiente, arriba a la izquierda le damos a "*file*" y a "*new scene*", ahí le podremos dar el nombre que queramos.

Pantalla de acceso (0):

Ilustración 15 - Pantalla de acceso

Los elementos que se han añadido a esta escena son los siguientes:

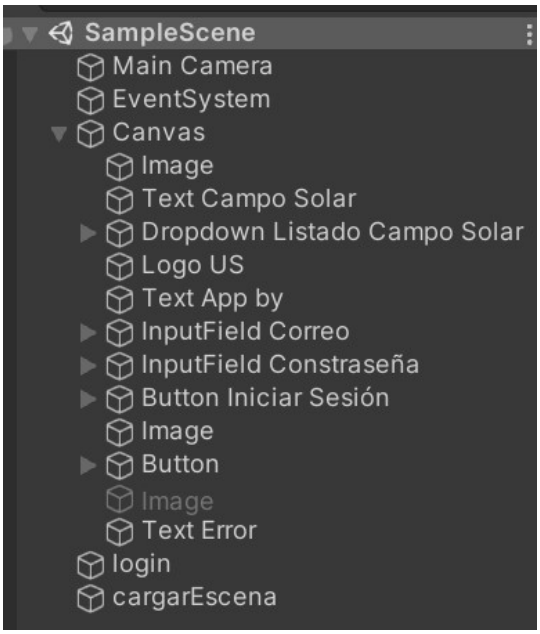

Ilustración 16 - Elementos de la pantalla de acceso

Para añadir todos estos elementos se debe hacer *click* con botón derecho (sobre el panel izquierdo en "*hierarchy*") bajo el título, seleccionamos "UI" (interfaz de usuario) y elegimos en primer lugar "Canvas", bajo este componente pondremos todos los demás colgando de él, del mismo modo, click derecho, UI y elegimos. En este caso tenemos imágenes, textos, un "dropdown" (desplegable), "inputField" (campos para introducir datos), un botón y script que llamamos cargar escena que nos sirve para acceder a otra pantalla desde ésta.

Pantalla de inicio (1):

En esta pantalla vamos a ver el primer cambio respecto al diseño original. Por un tema de seguridad y comodidad de cara al manejo de los registros de usuarios, se va a diferenciar entre perfil estudiante y perfil docente. La diferencia radica en que un perfil docente será el único perfil con permiso para poder aceptar nuevos usuarios. Por tanto, esta pantalla de inicio será distinta según el perfil que se sea. Si eres estudiante, verás la siguiente pantalla:

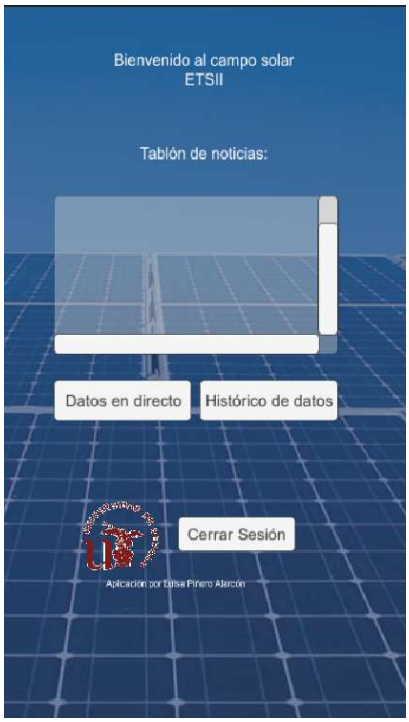

Ilustración 17 - Pantalla de inicio (estudiante)

Sin embargo, si eres docente, verás la siguiente pantalla, en la que podrás ver las solicitudes de acceso pendientes y aceptarlas:

Ilustración 18 - Pantalla de inicio (docente)

Al darle a “Ver campo solar” sí se accede a una pantalla igual a la que ve el estudiante. Es decir, esta pantalla sería una pantalla intermedia entre iniciar sesión y acceder al panel de bienvenida desde el que acceder a los datos de la planta, esta pantalla intermedia solo la tendrán los perfiles “docente” para poder admitir a nuevos usuarios.

El método de añadir los elementos es el mismo que la pantalla 0 explicada arriba, por lo que no repetiremos los objetos que ya se han mencionado para no ser redundantes y sólo se comentará algún elemento nuevo si se da el caso, como en este caso en el que tenemos un “*Scroll View*” que es simplemente un objeto que podemos ver más de su contenido haciendo *scroll* con el ratón. Y se añade aquí una interacción hacia la base de datos de apagar todos motores (ponerlos a 0) dándole a un único botón, añadiendo así una acción de control de la aplicación.

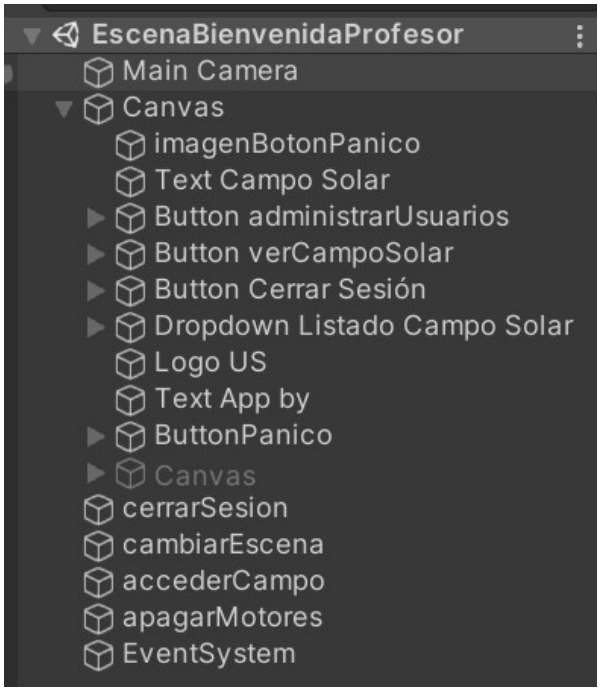

Ilustración 19 - Elementos de la pantalla de inicio de docente

Pantalla de registro (2):

En esta pantalla vamos a ver algunos cambios que se han introducido ya que, si existe un perfil estudiante y otro docente, esto se deberá especificar en la solicitud de registro. Esto lo haremos con un *check* que se seleccionará si se es docente y se dejará sin marcar en caso de ser estudiante. También se ha decidido centrarnos únicamente en el campo solar de la ETSI por lo que el registro se hará para este campo y la base de datos creada para los usuarios será sólo para este campo solar, en el caso futuro de querer ampliar para abarcar otras plantas solares será tan sencillo como replicar lo que se hace para esta planta, pero no creemos que aporte valor a este trabajo realizar lo mismo para otros campos solares inventados. La pantalla quedará así:

Ilustración 20 - Pantalla de registro

Y la forman los siguientes elementos:

Ilustración 21 - Elementos de la pantalla de registro

Pantalla de histórico (3):

Estos datos se van a obtener de una base de datos donde se han simulado registros tanto para los datos en directo como para este histórico de datos, la idea cuando la planta esté en funcionamiento será pasar los datos de los distintos sensores a una base de datos online donde podamos conectar la aplicación del mismo modo que ha sido conectada para nuestra base de datos online.

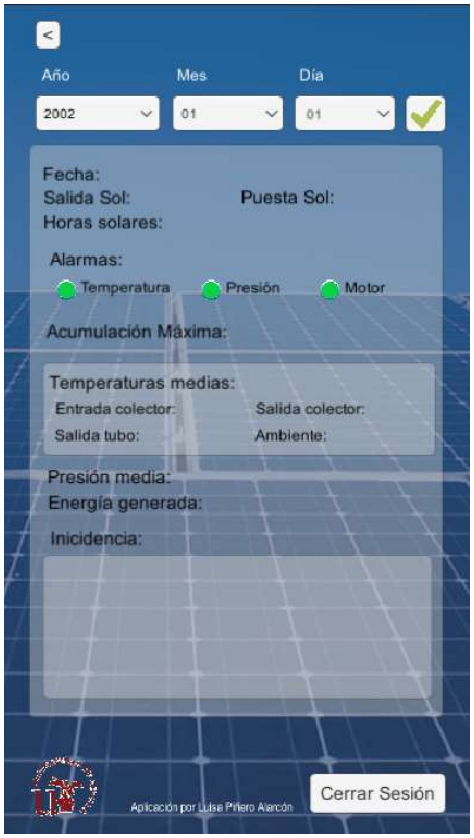

Ilustración 22 - Pantalla de histórico de datos de estudiante

Elementos que lo forman:

Ilustración 23 - Elementos de la pantalla de histórico de datos

Para esta pantalla se tiene una variante para el caso de docente, éste podrá añadir un comentario a la tabla en el campo de “incidencia”. Por ejemplo, si un día se ha parado alguno de los motores por alguna avería, el docente podrá dejar constancia desde la propia aplicación. La pantalla quedará por tanto del siguiente modo:

Ilustración 24 - Pantalla de histórico de datos para perfil docente

Y sus elementos:

Ilustración 25 - Elementos de la pantalla de histórico de datos para docente

Pantalla de datos en directo (4):

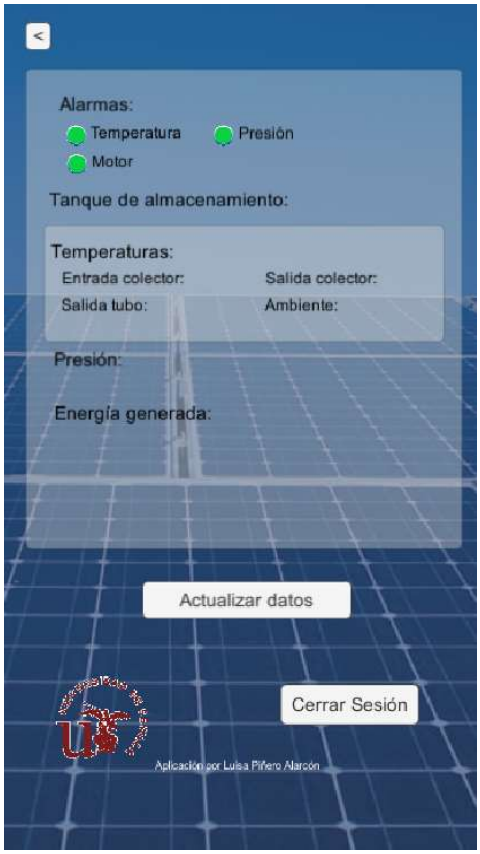

Ilustración 26 - Pantalla de datos en directo

Elementos que lo forman:

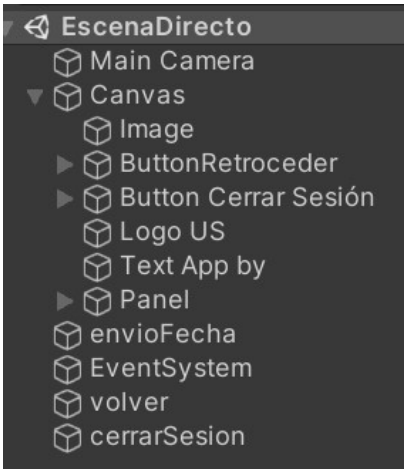

Ilustración 27 - Elementos de la pantalla de datos en directo

Pantalla de registro correcto (5):

Ilustración 28 - Pantalla de registro correcto

Y los elementos de esta pantalla son los siguientes:

Ilustración 29 - Elementos de la pantalla de registro en directo

5.4.2. Creación de scripts en Unity

Para ser ordenados, en el proyecto, en *asset*, se puede diferenciar por directorios las escenas (scenes), imágenes (images), scripts… En este caso se ha creado una carpeta "scripts" que cuelga de asset y contiene todos los scripts que se usan en la aplicación. De este modo un mismo script puede ser utilizado para distintas escenas, asociándoselo luego al objeto que lo necesite. Un caso de un script utilizado varias veces en este mismo proyecto es el script de SceneManager.cs, ya que es el encargado de cargar la pantalla que se quiera al pulsar un botón. En este caso, este script se puede asociar al botón (en la propiedad de *onClick*) de dos formas: o bien directamente arrastramos el script al botón que deseamos y seleccionamos en él la escena a la que se quiere ir al pulsar ese botón, o bien se crea un objeto vacío que asociamos al botón del mismo modo. Por tener un orden visual y poder usar el mismo script varias veces en una escena, se ha optado por la segunda opción, para no tener que entrar en las propiedades del botón para ver qué sucede al pulsarlo, ya que se tiene un objeto vacío con un nombre intuitivo que muestra qué es lo que hace. Todos los scripts creados están recogidos al final de la memoria en la parte de ANEXOS para no saturar estos puntos de la memoria, en los que no se va a entrar en el detalle de los códigos.

5.5. Acceso de usuarios (*login*)

Aunque puede parecer una parte sencilla del trabajo, el *login* puede ser la parte más compleja de este trabajo, ya que realizando esta parte nos hemos dado cuenta de todas las comprobaciones que se deberían llevar a cabo previas al acceso de esta aplicación.

En primer lugar, nos encontrábamos ante la problemática de dónde guardar los datos de registro de tal modo que se pudieran consultar en cualquier momento. La primera opción había sido una base de datos en un equipo propio de la ETSI. Esto no era operativo dado que requería de un equipo que estuviera siempre encendido y con conexión a internet, además de tener que estar siempre supervisado. Este problema se solucionó rápidamente al buscar una base de datos de MySQL online y gratuita. En segundo lugar, se había pensado que un profesor encargado de esta base de datos introdujera los nuevos usuarios manualmente, esto no era adecuado porque meter datos a mano en una base de datos tiene un riesgo alto de introducir algún error debido al factor humano y, además, la persona encargada de este paso debe tener nociones informáticas ya que manipularía los registros en una base de datos. Esto se ha solucionado introduciendo el módulo de administración en la aplicación, de tal modo que el profesor desde su sesión pueda aceptar a los nuevos usuarios con un simple *click*, siendo en ese momento cuando se registra totalmente el nuevo usuario en la base de datos.

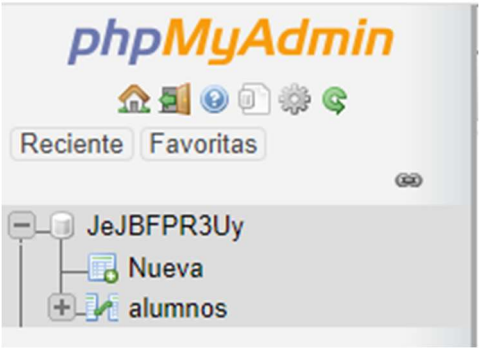

Ilustración 30 - Crear una tabla nueva

con los siguientes campos (se irán añadiendo más conforme más usuarios se vayan registrando):

correo	password	rol	acceso
anreyes@us.es	tfm2021!	profesor	si
luipinala@alum.us.es	pass1234	alumno	si
luisa_pial@hotmail.com	pass1111	alumno	no

Tabla 3 - Tabla de usuarios

El campo “rol” será el encargado de determinar si un usuario va a entrar como estudiante o como docente, es decir, si accederá a la pantalla de admitir a nuevos usuarios o no. El campo de “acceso” determina si ese usuario tiene ya permiso para acceder (si) o sigue pendiente de aceptación (no).

Cuando un usuario quiere acceder, seleccionará el campo solar al que se quiere acceder (en este caso sólo estará habilitado el de la ETSI como se ha comentado anteriormente) escribirá su dirección de correo electrónico, su contraseña y le dará a “iniciar sesión”.

Las comprobaciones que hará el programa serán las siguientes:

1. El correo electrónico está dado de alta en la tabla de base de datos (de no ser así, mostrará un error de que el usuario no está registrado).

2. Si el correo existe, se comprobará que la contraseña coincida con la contraseña definida para ese correo en la tabla de base de datos (de no ser así, nos dará un error).

3. Se comprobará el campo “estado” para ver si ese usuario ya tiene acceso a su sesión, o si por el contrario está aún pendiente de validación. En caso de no acceder por estar pendiente, también nos mostrará el aviso por pantalla.

4. Si cumple todas las condiciones anteriores se accederá a la pantalla de bienvenida de estudiante o docente, según el caso que corresponda.

El código de la parte del login en C# está definido en el script “Login.cs”, está completo en el anexo 1 al final de este trabajo.

Si un usuario no está registrado y quiere poder acceder a esta aplicación, deberá dar a “¿No tienes cuenta?” para registrarse. Al darle a ese enlace, accederá a la pantalla de registro, en el que deberá rellenar una serie de campos: correo electrónico, *check* de si se es docente, contraseña y confirmación de contraseña, nombre completo y motivo por el cual solicita el acceso. El programa de registro hace las siguientes validaciones:

1. Todos los campos deben estar informados antes de enviar la solicitud ya que, de no ser así, no se enviará la solicitud y nos aparecerá un mensaje avisándonos de que tenemos campos sin informar.

2. Comprueba si la *check* de docente está señalada o no para determinar el tipo de usuario que se va a insertar en la tabla en el caso de que el registro finalice de manera satisfactoria.

3. Los correos electrónicos deben ser de este tipo “@gmail.com”, “@hotmail.com”, “@alum.us.es” o “@us.es”.

4. El correo electrónico no debe estar ya informado en la base de datos (es una *primary key*). Si está informado, se mostrará el mensaje de que el usuario está ya registrado.

5. La contraseña debe contener al menos 8 caracteres, de no ser así, nos informará por pantalla de lo que debe cumplir una contraseña para ser aceptada.

6. Las dos contraseñas deben coincidir.

7. Una vez que se realiza el registro de manera satisfactoria, llegará un correo electrónico a todos los correos de perfil docente que haya en la base de datos en la tabla de usuarios en ese momento, para avisar de que tal usuario solicita el acceso a la aplicación. En el correo vendrán los datos de correo y motivo para acceder.

Llegados a este punto, cualquier docente puede entrar en la aplicación y aceptar al nuevo usuario, de tal modo que le llegue un aviso a su correo de que ya ha sido aceptado, y por tanto ya puede acceder a la aplicación.

El código de la parte del registro en C# está definido en el script “Registro.cs” y está en el anexo 2 al final de este trabajo.

5.6. Conexión

Para simular unos datos de la planta se ha empleado la misma base de datos que la usada para la tabla de usuarios del *login*. En este caso se tienen dos tipos de tablas. La tabla de datos en directo tiene solo un registro ya que simula los datos que arroja el campo solar en una “foto”. Se le ha añadido un campo “fecha” pensando en un futuro, ya que lo ideal sería que el proceso que se encargue de actualizar los valores de la planta en la base de datos, en lugar de *updatear* ese registro de datos en directo, introduzca un nuevo registro con la nueva hora, de modo que para obtener el dato en directo solo hará falta una pequeña modificación de código para que se extraiga la fila de fecha mayor. Además, si en un futuro se hace eso, también será mucho más rápido generar al final del día la nueva fila que se insertará en la tabla de histórico, haciendo la media de los valores del día de según que parámetro, o el valor máximo de todos o lo que corresponda. Este es el registro que se introducido para simular una foto en directo:

motor1	motor2	…	motor22	tanqueAlmacenamiento	temTec	tempTsc	tempTst	tempTamb	presión	energiaGenerada	fecha
1	1		1	100	140	180	140	160	10	450	24/11/2021 0:00

Tabla 4 - Tabla de datos en directo

Los campos que vamos a sacar para mostrar en las pantallas de la aplicación, o los que nos harán falta para mostrar algún resultado indirecto son los siguientes:

motorx: Campo del motor x de los 22 posibles (es decir, la tabla tiene 22 columnas de motor). Booleana, 0 apagada, 1 encendida.

tanqueAlmacenamiento: Carga del tanque de almacenamiento que tiene el sistema para cargarse con la energía excedente en algunos momentos del día. Su valor máximo es 292kW.h (las unidades de este campo son KW.h).

tempTec: Temperatura de entrada al captador (ºC).

tempTsc: Temperatura de salida al captador (ºC).

tempTst: Temperatura de salida de los tubos (ºC).

tempTamb: Temperatura ambiente (ºC).

presión: Presión (bar).

energíaGenerada: Energía generada por campo solar en lo que va de día (KW·h).

fecha: formato YYYY-MM-DD (en el caso de que en un futuro se inserten cada X segundos la foto del campo en la tabla de datos en directo, se debería guardar con HH:MM:SS para poder tener un registro durante todo el día de cara a generar la tabla de histórico de datos al finalizar el día.

Las alarmas de esta pantalla dependen de los valores de temperatura de salida del colector, presión o estado de los motores respectivamente. Así que se encenderán en los siguientes casos:

Alarma Ts: Alarma que avisa si la temperatura de salida del captador es mayor a un valor concreto (en este caso 190ºC).

Alarma P: Alarma que avisa si la presión es mayor a un valor concreto (en este caso 15 bar). Booleana, 0 apagada, 1 encendida.

Alarma Motor: Alarma que avisa si alguno de los 22 motores se parase. Booleana, 0 apagada, 1 encendida.

Para el caso del histórico de datos, hemos metido registros del 1 de septiembre en adelante hasta el 31 de diciembre para hacer las pruebas y para poder mostrar su funcionamiento el día de la defensa de este trabajo. Los datos de salida y puesta de sol sí son reales [13].La tabla es así:

fecha	salidaSol	puestaSol	alarmaTs	alarmaP	alarmaM	acumulacionMax	tecMed	tscMed	tstMed	TambMed	presionMed	energiaGen	incidencia
01/09/2021	7:54:00	20:53:00	0	0	1	250	145	160	120	25	10.3	658.7	El motor 8 se paró por una avería
02/09/2021	7:55:00	20:51:00	0	0	0	252	142	163	115	24	11.1	640.6	
03/09/2021	7:56:00	20:50:00	0	0	0	200	137	150	145	25	11.2	635.6	
04/09/2021	7:57:00	20:48:00	0	0	0	210	145	168	140	25	11.3	635.3	
05/09/2021	7:57:00	20:47:00	1	0	0	240	140	170	145	23	10.8	660.7	
06/09/2021	7:58:00	20:45:00	0	0	0	205	142	167	130	22	11.1	580.2	
07/09/2021	7:59:00	20:44:00	0	0	0	188	120	167	160	22	12	580.2	
08/09/2021	8:00:00	20:42:00	0	0	0	190	131	170	165	21	13.1	590.1	

09/09/2021	8:01:00	20:41:00	0	0	1	202	125	165	160	21.5	12.1	587.4	
10/09/2021	8:01:00	20:39:00	0	0	0	195	122	164	133	23	12.7	560.2	
11/09/2021	8:02:00	20:38:00	0	0	0	240	124	155	145	23	13.8	563.5	
12/09/2021	8:03:00	20:36:00	0	1	0	210	130	150	142	19	12.6	600.2	
13/09/2021	8:04:00	20:34:00	0	0	0	212	135	170	140	20	12.7	620.7	
14/09/2021	8:05:00	20:33:00	0	0	0	202	125	172	143	20.1	12.6	610.1	
15/09/2021	8:06:00	20:31:00	0	0	0	198	120	165	145	21.1	13.1	605.8	

Tabla 5 - Tabla de histórico de datos

Los campos que se van a sacar para mostrar en las pantallas de la aplicación, o los que harán falta para mostrar algún resultado indirecto son los siguientes:

fecha: formato YYYY-MM-DD (en este caso, la fecha será siempre de este formato ya que habrá un registro por día).

salidaSol: Hora de salida del sol ese día en Sevilla (son datos reales obtenidos de páginas de meteorología, por eso tenemos los datos metidos hasta el 31 de diciembre).

puestaSol: Hora de la puesta de sol de Sevilla (idem que la anterior).

alarmaTs: Alarma que avisa si la temperatura de salida del captador ha sido mayor a 190ºC en algún momento del día. Es decir, se traduce a que constará como 1 (verdadero) si ha saltado la alarma en algún momento de las comprobaciones que se hayan hecho para los datos en directo de ese día. Booleana, 0 apagada, 1 encendida.

alarmaP: Alarma que avisa si la presión ha sido mayor a un valor concreto (en este caso 15 bar). Idem que la anterior. Booleana, 0 apagada, 1 encendida.

alarmaMot: Alarma que avisa si alguno de los 22 motores se ha parado en este día (ha tomado el valor de 0). Idem que la anterior. Booleana, 0 apagada, 1 encendida.

acumulacionMax: Registra el % de carga mayor del día (entre 0 y 100) del acumulador de energía.

tecMed: Temperatura media de entrada al captador (ºC).

tscMed: Temperatura media de salida al captador (ºC).

tstMed: Temperatura media de salida de los tubos (ºC).

tambMed: Temperatura ambiente media (ºC).

presiónMed: Presión media (bar)

energíaGen: Energía generada por campo solar al final del día (KW·h).

5.7. Módulos de la aplicación

La parte funcional de la aplicación puede dividirse en dos grandes módulos: el módulo de accesos y el de datos.
A continuación, se comentan en detalle estos módulos que conforman la lógica funcional de *EnergyApp*.

5.7.1. Acceso

El módulo de acceso es el encargado de gestionar los usuarios, tanto para el registro de estos como para el inicio de sesión una vez registrados.

- Capa de datos: La capa de datos relativa a este módulo la compone la siguiente tabla:
  - Usuarios

- Capa de negociación: La capa de negociación relativa a este módulo la componen los PHP siguientes:

  - dbConnection.php
  - loginUsuario.php
  - registroUsuario.php
  - administraciónUsuarios.php

- Capa de presentación: La capa de presentación está compuesta por las escenas y los scripts siguientes:

  - Login (escena)
  - Registro (escena)
  - Administración (escena)
  - SceneManager.cs (script)
  - NetworkManager.cs (script)

La lógica del módulo a bajo nivel es la siguiente:

- En la pantalla de Login, un usuario introduce un ‘correo’ y una ‘contraseña’. Tras esto pulsa en ‘Iniciar Sesión’. En este momento, se realiza una conexión con la base de datos a través del servicio ofrecido por dbConnection.php. También se realiza una consulta para validar que el usuario y la contraseña corresponden con un usuario dado de alta y que dispone de acceso. La consulta es la siguiente:

  ```
  SELECT password,rol FROM alumnos WHERE correo = '$correo'
  ```

- Si los valores de correo y contraseña introducidos son correctos, se carga la escena BienvenidaUsuario (si el rol del usuario es “alumno”) o BienvenidaProfesor (si el rol del usuario es “profesor”).

- Desde la pantalla de Login, también es posible pulsar sobre el botón ‘¿No tienes cuenta?’. Este botón llama al script ‘SceneManager.cs’ para cargar la escena de Registro.

- En la pantalla de Registro, el usuario introduce un correo válido, una contraseña y la validación de esta contraseña (repetición). También tiene la opción de indicar si el usuario a dar de alta es o no docente mediante un *check* seleccionable. Como medida, es obligatorio indicar el motivo por el que se necesita acceso a la aplicación. Al pulsar sobre el botón ‘Enviar’ y si los datos son correctos, el script ‘NetworkManager.cs’ realiza una llamada al servicio ‘crearUsuario.php’ el cual consulta a la base de datos si no existe el usuario previamente para introducir la entrada del nuevo usuario con la siguiente query:

  ```
  INSERT INTO alumnos(correo, password,rol,acceso) VALUES(:field1,:field2,:field3,'no')
  ```

- Como consecuencia del éxito en el registro, el script se comunica con el servidor SMTP correspondiente para enviar una notificación vía email a los docentes que tienen acceso. El listado de usuarios con rol “profesor” y acceso “si” se obtiene a través del servicio que ofrece ‘profesoresAcceso.php’ mediante otra consulta a la base de datos:

  ```
  SELECT correo FROM alumnos WHERE rol LIKE 'profesor' AND acceso LIKE 'si'
  ```

- Como último paso, se carga la escena RegistroCorrecto, notificando al usuario del registro exitoso.

5.7.2. Datos

El módulo de datos es el encargado de gestionar los datos relacionados con los campos solares, tanto en directo como por histórico.

- Capa de datos: La capa de datos relativa a este módulo la componen las siguientes tablas:
  - datosDirecto
  - historicoDatos
- Capa de negociación: La capa de negociación relativa a este módulo la componen los PHP siguientes:
  - dbConnection.php
  - obtenerDato.php

    - obtenerDirecto.php
    - introducirDatos.php
- Capa de presentación: La capa de presentación está compuesta por las escenas y los scripts siguientes:
    - DatosDirecto (escena)
    - DatosHistórico (escena)
    - SceneManager.cs (script)
    - DatosSolares.cs (script)
    - Directo.cs (script)

La lógica del módulo a bajo nivel es la siguiente:

- En la pantalla de Bienvenida, el usuario puede seleccionar si desea ver los datos en directo o si prefiere consultar el histórico. Estas acciones se revelan a través de unos botones que llaman al script 'SceneManager.cs'.

- Desde la pantalla de datos en directo los datos se cargan instantáneamente gracias al servicio que ofrece 'obtenerDirecto.php' y su respectiva conexión a la tabla 'datosDirecto'. La consulta que se realiza es la siguiente:

    SELECT * FROM datosDirecto

- Esto muestra por pantalla los datos y permite volver a llamar al mismo servicio si pulsamos sobre el botón 'Actualizar Datos'.

- Desde la otra pantalla de datos que corresponde con el Histórico, el usuario debe seleccionar un día, un mes y una fecha y pulsar sobre el botçon de validar junta a esta. En este momento se realiza una consulta mediante el servicio 'obtenerDatos.php':

    SELECT * FROM historicoDatos WHERE fecha= '$fecha'

- Si la fecha coincide con la de alguna entrada en la tabla 'historicoDatos' de la base de datos con la que conecta, se pinta la información en pantalla. Si no, devuelve un error. Estas acciones las realiza el script 'DatosSolares.cs'.

- Además, en esta pantalla también encontramos un campo de texto donde es posible introducir a modo de comentario una incidencia para el día seleccionado. Si el usuario introducie texto en ese campo y pulsa sobre el botón validar que se encuentra a su derecha, la aplicación insertará a través del servicio 'introducirDatos.php' la incidencia en la entrada para la fecha seleccionada. La query que realiza la actualización de incidencia es la siguiente:

    UPDATE historicoDatos SET incidencia='$incidencia' WHERE fecha = '$fecha'.

5.8. Intento de conectar con Matlab

La idea inicial de esta aplicación no era otra que la de conectar esta aplicación directamente con la planta real. Esto no ha podido ser posible ya que la planta de la ETSI sigue parada tras la avería,

y no se sabe cuánto tiempo pasará hasta que se compren los repuestos necesarios para su puesta en marcha de nuevo. Por tanto, la segunda idea que se tuvo fue intentar conectarla a una simulación en Matlab de esta planta, que estaba desarrollando un compañero de otro máster. El problema en este punto estaba en que el ritmo de realización de los trabajos no era el mismo ni iban necesariamente en paralelo, y se necesitaban datos de esa simulación para ir avanzando con este trabajo, y esto era lo final del trabajo del compañero que no contaba con entregarlo este curso, por lo que esperar para poder intentar conectar esta aplicación a su simulación no era viable. Por tanto, y a modo avance de cara al compañero que realice la conexión de esta aplicación con la simulación o con la planta real, se decidió intentar conectar la aplicación con Matlab aunque solo fuera para recoger y mostrar algún valor que se le pasase a la aplicación. El problema principal que nos encontramos fue que la mayoría de información era bastante antigua y obsoleta para las conexiones entre Unity y Matlab. Hay que dejar claro que la mayoría de información se ha obtenido en foros en inglés entre distintos trabajadores de la comunidad informática, y prácticamente todos desembocaban en el mismo código. Tras buscar en distintos foros y vídeos de ejemplos [15] [16], todas las conexiones entre Unity y Matlab tienen el siguiente código base para el caso que necesitamos en el que Unity es el servidor (*server*) y Matlab el cliente (*client*):

Unity:

```
using UnityEngine;
using System.Collections;
using System.Net;
using System.Net.Sockets;
using System.Linq;
using System;
using System.IO;
using System.Text;
public class readSocket : MonoBehaviour {
      // Use this for initialization
      TcpListener listener;
      String msg;
      void Start () {
          listener=new TcpListener (55001);
          listener.Start ();
          print ("is listening");
      }
      // Update is called once per frame
      void Update () {
          if (!listener.Pending ())
          {
          }
          else
          {
              print ("socket comes");
              TcpClient client = listener.AcceptTcpClient ();
              NetworkStream ns = client.GetStream ();
              StreamReader reader = new StreamReader (ns);
              msg = reader.ReadToEnd();
              print (msg);
          }
      }
  }
```

Matlab:

```
clc
clear all
tcpipClient = tcpip('127.0.0.1',55001,'NetworkRole','Client');
set(tcpipClient,'Timeout',15);
fopen(tcpipClient);
a='yah!! we could make it';
fwrite(tcpipClient,a);
fclose(tcpipClient);
```

El código de Matlab empezaba dando el aviso de que la parte de "*clear all*" no era necesaria y que solía dar problemas, por lo que decidimos eliminarlo del código al ver que obteníamos el mismo resultado con él que sin él. Tras esto, llegó el error de que el comando "*tcpip*" estaba obsoleto, parece que las últimas versiones de Matlab no aceptan este comando y en su lugar te piden usar el "*tcpClient*", por lo que hicimos dicha corrección y nos encontramos con el problema de que el "*fopen*" parecía no poder abrir la variable "*tcpipClient*" ya que esta no era un fichero, por lo que probamos a realizar un paso intermedio definiendo una variable "*data*" que leyera dicha variable y se quedara con los primeros 30 caracteres (de sobra para leer la dirección ip). Con estas correcciones el código queda del siguiente modo:

```
clc
tcpipClient = tcpclient('127.0.0.1',55001,'Timeout',15);
data=read(tcpipClient,30,"string");
fopen(data);
a='yah!! we could make it';
fwrite(tcpipClient,a);
fclose(tcpipClient);
```

El problema, es que no se consigue una conexión correcta tras muchos intentos y distintas pruebas, algo ocurre en Unity pero en Matlab tras pasar los 15s del timeout falla diciendo que no ha sido capaz de conectar. Hay que tener en cuenta que en estas conexiones, primero debe ejecutarse la parte del servidor (en este caso el programa de Unity) y después la de cliente (en este caso Matlab), si no se hace así, nos aparecerá un error. El problema que nos hemos encontrado además, es que cada vez que ponemos a correr Matlab, la parte de Unity se queda pillada y el programa deja de reaccionar a nuestros intentos de pararlo o hacer cualquier modificación, de modo que hay que cerrar Unity desde el administrador de tareas del equipo y volver a abrirlo de nuevo (tarda un par de minutos en arrancar todo) de modo que todos los intentos de arreglar esto han desencadenado mucho tiempo perdido cerrando y abriendo el programa antes de revisar y volver a hacer otro intento.

En la consola de Unity vemos como se pinta el "is listening" al ejecutar el programa y llega a pintar el "socket comes" cuando se ejecuta el programa de Matlab a continuación.

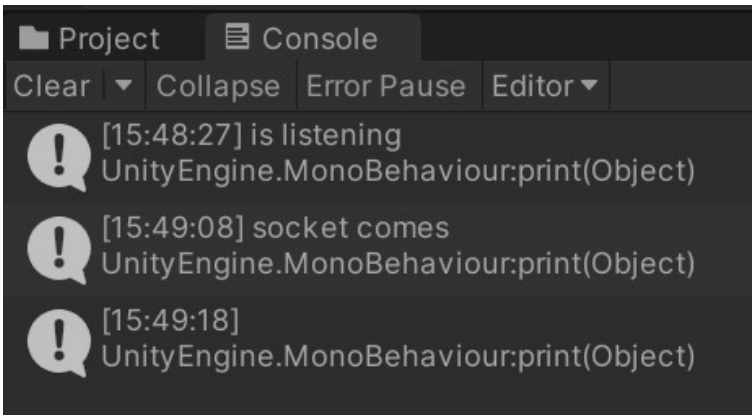

Ilustración 31 - Consola de Unity durante la ejecución con Matlab

Sin embargo, pasados los 15 segundos, Matlab pinta el siguiente fallo:

```
Error using PROGRAMA_MATLAB (line 3)
Error receiving data from the remote server.
Additional Information: Operation timed out before requested data was received.
```

Ilustración 32 - Fallo en Matlab tras intentar leer de Unity

5.9. Google play

Google Play es la tienda de aplicaciones de Android. Para subir cualquier aplicación es necesario darse de alta como desarrollador, con un coste de 30€ y cumplir con una serie de requisitos a la hora de creación del contenido de la ficha para Google Play y en el código de la aplicación.

Para la subida de una aplicación a la tienda de aplicaciones primero es necesario que ésta sea compilada y firmada se debe generar un archivo "key"(.jks). Con la *key* y habiéndose podido corregir los *warnings* y errores que este exigente compilado detecta, obtendremos nuestra apk lista para la subida.

Es importante realizar copias de seguridad de la key entregada pues no se puede volver a generar y si no se dispone en el futuro de ella, no se podrá volver a actualizar la aplicación en la tienda de aplicaciones, teniéndose que subir otra versión diferente duplicando así la aplicación en la Play Store y con el consecuente problema de actualización a los usuarios.

Google Developer Console es un portal donde los desarrolladores suben sus aplicaciones a la tienda Play Store, crean las fichas de información de dicha aplicación y consultan todo tipo de estadísticas relevantes para el desarrollo su desarrollo y seguimiento. La aplicación se clasificaría en la categoría que más encaje y una vez subida se le aplica unas clasificaciones de consumo, que suelen ser las más sencillas para una aplicación de estas características. En el *Developer Console* hay un apartado de APK que nos dirá la compatibilidad de nuestra aplicación, dándonos el número de modelos de dispositivos diferentes compatibles con la aplicación, y también veremos ahí el número de actualizaciones de ésta.

También hay un apartado de bastante interés que es el apartado de reporte de errores de usuarios, donde aparecerá si se ha detectado algún error que haya sido reportado por los usuarios que han descargado la aplicación. [14]

En el caso de esta aplicación se ha decidido no subirla por ahora a Google play ya que estamos ante una versión beta de la aplicación que carece aún de una conexión a la planta real, por lo que la funcionalidad actual sería más bien de prueba y no será hasta más adelante que será plenamente funcional, una vez que exista una web con datos reales cuando la planta esté de nuevo en funcionamiento. Por tanto, actualmente la subida a Google play solo conllevaría un gasto económico y, además, dicho perfil quedaría asociado a la autora de este trabajo, y entorpecería futuras actualizaciones realizadas por otros estudiantes que pudieran heredar este trabajo para continuarlo. Además, estaría el gran inconveniente de que cualquier persona podría descargarse la aplicación e intentar registrarse, llegándole avisos al correo a los docentes que estén registrados en la aplicación. Viendo que los contras actuales son mucho mayores que los pros, se decide informar de los pasos a seguir para la subida de una aplicación a Google play, pero sin llevarlo a cabo en este momento.

6 IMPLEMENTACIÓN, PRUEBAS Y RESULTADOS

Una vez finalizado el desarrollo de la aplicación en Unity se revisan los datos de los sistemas operativos más usados el año pasado para poder hacernos una idea del mercado actual de cara a la distribución de esta aplicación. Las estadísticas son las siguientes [15]:

Sistemas operativos para móviles más usados en 2020

Sistema Operativo	Porcentaje
Android	71,42 %
iOS	27,95 %
Samsung	0,2 %
KaiOS	0,14%
Desconocido	0,12 %
Windows	0,06 %

Ilustración 33 - Sistemas Operativos móviles más usados

El sistema operativo móvil más usado en el mundo en 2020 fue el Android, con un 71.42%, seguido de IOS con 27.95%, teniendo con éste prácticamente el 100% de los sistemas operativos utilizados, aunque con bastante diferenciados el primer y el segundo puesto.

Android es un sistema operativo móvil basado en el núcleo Linux y otros *softwares* de código abierto. Fue diseñado para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas, relojes inteligentes Wear OS, automóviles con otros sistemas a través de Android Auto, al igual los automóviles con el sistema Android Automotive y televisores Android TV.

Inicialmente fue desarrollado por Android Inc., que fue adquirido por Google en 2005. Android fue presentado en 2007 junto con la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El código fuente principal de Android se conoce como Android Open Source Project (AOSP), que se licencia principalmente bajo la Licencia Apache. Android es el sistema operativo móvil más utilizado del mundo muy por encima de IOS.

Según los documentos secretos filtrados en 2013 y 2014, el sistema operativo es uno de los objetivos de las agencias de inteligencia internacionales. [16]

Además, cabe destacar, que España está muy por encima de ese porcentaje ya que el mercado de *smartphones* en España tiene particularidades que no tiene ningún otro. El 90,8% de todos los que se vendieron entre junio y agosto de 2013 funcionaban con sistema Android, según Kantar Worldpapnel. No hay otro país con ese porcentaje, pues el siguiente China tenía solo un 72,4%. [17].

6.1. Implementación

Con intención de ser lo más eficientes posible, se plantea pues el desarrollo de la aplicación móvil en este SO, Android, que nos ofrece la capacidad de abarcar un mercado y un público más extenso programando en su plataforma.

Además, pese a no ser lo que se pensó en un principio, se decide compilar también para ordenador, de modo que la aplicación pueda ser usada también desde un ordenador si es más cómodo. Ambos casos se testean para poder comprobar su correcto funcionamiento y su visualización adaptada a cada uso. Cabe destacar que se podría compilar para IOS, pero en este caso no se puede asegurar su correcto funcionamiento y conexiones hacia las bases de datos generadas, ya que se han desarrollado buscando información para aplicaciones Android, además, no se dispone de terminales ios para poder hacer pruebas y asegurar su correcto funcionamiento en estos terminales, por lo que se decide centrarnos en asegurar un funcionamiento sin errores en Android y para pc en el caso de que algún usuario no tenga terminal Android, ya que se entiende que todos los que necesiten acceder a estos datos disponen en casa o en la universidad de ordenador.

6.2. Testing

Una de las partes más importantes de todo desarrollo es el "*testing*" o comprobación de que todo el código funciona. Durante este paso aparecen pequeños fallos que se irán corrigiendo y depurando hasta conseguir una aplicación sin errores en su ejecución. En el caso de esta aplicación se han determinado las siguientes pruebas:

6.2.1. Pantalla de acceso:

- Se muestra error cuando se intenta acceder con un usuario que no está registrado. Por ejemplo, esto_es_una_prueba@hotmail.com.

Ilustración 34 - Evidencia de usuario no registrado

- Se muestra error cuando la contraseña introducida para un usuario registrado es incorrecta. Por ejemplo “noeslacontraseña” para luisapial@gmail.com.

Ilustración 35 - Evidencia de contraseña incorrecta

- Se muestra error cuando el usuario aún está pendiente de validación por parte de algún docente.

Ilustración 36 - Evidencia de usuario pendiente de validación

- Se accede con un perfil estudiante. Por ejemplo, luisapial@gmail.com con contraseña "lpatfm2021".

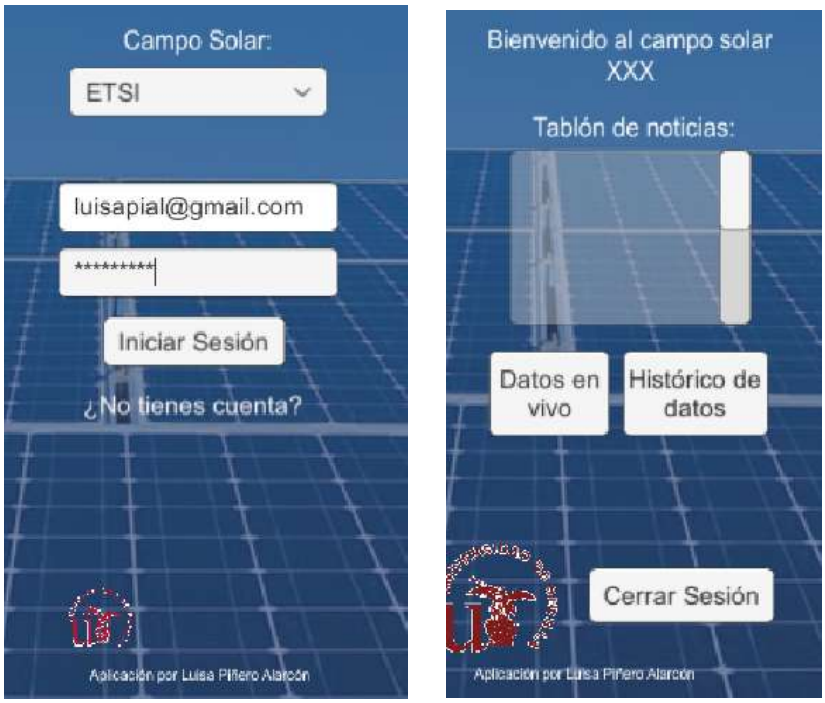

Ilustración 37 - Evidencia de acceso desde un perfil estudiante

- Se accede con un perfil docente. Por ejemplo, anreyes@us.es con contraseña tfm2021!.

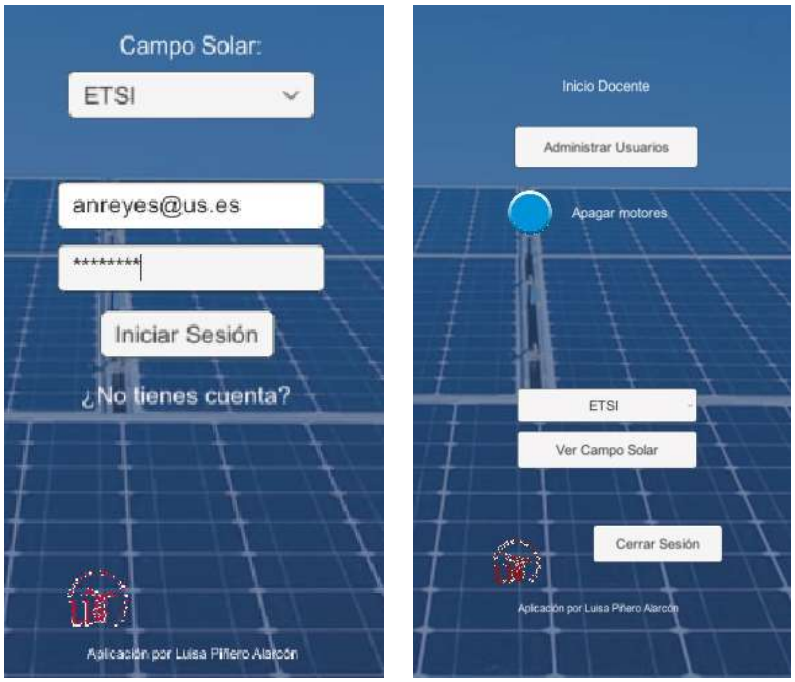

Ilustración 38 - Evidencia de acceso desde un perfil docente

6.2.2. Pantalla de inicio:

Si se accede con perfil de estudiante:

- Se cierra sesión y se vuelve a la pantalla de acceso.

Ilustración 39 - Evidencia de cierre de sesión de un perfil estudiante

- Se pulsa el botón de datos en directo y se accede a dicha pantalla.

Ilustración 40 - Evidencia de datos en directo estudiante

- Se pulsa el botón de histórico de datos y se accede a dicha pantalla.

Ilustración 41 – Evidencia de acceso a la pantalla de histórico de datos de estudiante

Si se accede con perfil docente:

- Se cierra sesión y se vuelve a la pantalla de acceso.

Ilustración 42 - Evidencia de cierre de sesión de un perfil docente

- Se aceptan los usuarios pendientes de aceptación y desaparecen de la pantalla. Por ejemplo, se va a aceptar al usuario luipinala@alum.us.es.

Ilustración 43 - Evidencia de aceptar usuarios pendientes de validación

- Llega correo de aviso a dichos usuarios avisando de que ya tienen acceso a la aplicación.

Ilustración 44 - Evidencia de correo tras dar acceso

- Se pulsa el botón del campo solar y se accede al inicio como el anterior del estudiante.

Ilustración 45 - Evidencia de acceso a campo solar de perfil docente

- Se cierra sesión y se vuelve a la pantalla de acceso.

Ilustración 46 - Evidencia de cierre de sesión de perfil docente

- Se pulsa el botón de datos en directo y se accede a dicha pantalla.

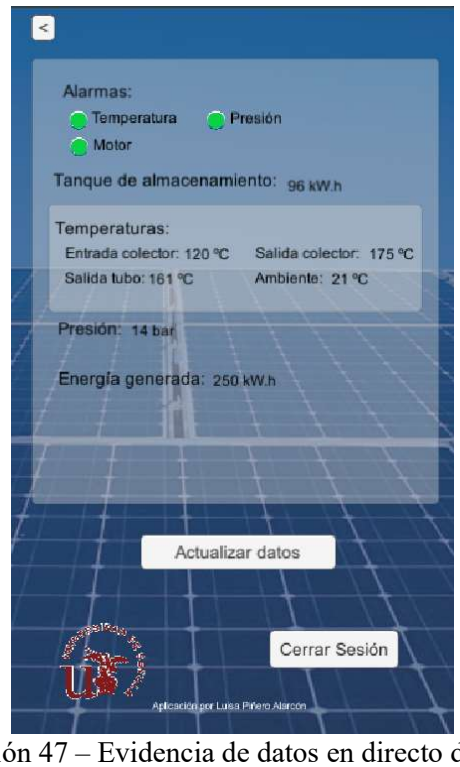

Ilustración 47 – Evidencia de datos en directo de docente

- Se pulsa el botón de histórico de datos y se accede a dicha pantalla.

Ilustración 48 – Evidencia de acceso a la pantalla de histórico de datos de docente

6.2.3. Pantalla de datos en directo:

- Se cierra sesión y se vuelve a la pantalla de acceso.

Ilustración 49 - Evidencia de cierre de sesión desde datos en directo

- Se le da a la flecha de retroceder y se vuelve a la pantalla de inicio.

Ilustración 50 - Evidencia de vuelta a pantalla de inicio desde datos en directo de estudiante

6.2.4. Pantalla de histórico de datos:

- Se selecciona una flecha con los desplegables y se da al *check* para obtener los datos si los hay.

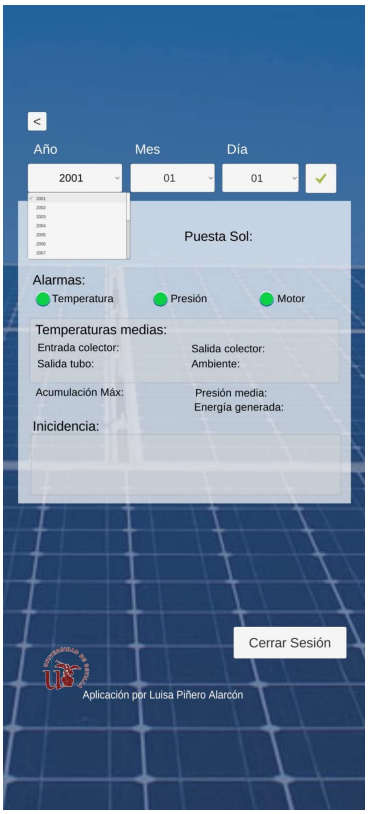

Ilustración 51 - Evidencia de desplegable de histórico de datos

- Se cierra sesión y se vuelve a la pantalla de acceso.

Ilustración 52 - Evidencia de cierre de sesión desde histórico de datos

- Se le da a la flecha de retroceder y se vuelve a la pantalla de inicio.

  Caso estudiante:

Ilustración 53 - Evidencia de vuelta a pantalla de inicio de estudiante desde histórico

Caso docente:

Ilustración 54 - Evidencia de vuelta a pantalla de inicio de docente desde histórico

- En caso de haber entrado con perfil docente, posibilidad de introducir un texto como incidencia al día seleccionado.

Ilustración 55 - Evidencia de campo incidencia en perfil docente

- Error en incidencia si se selecciona alguna fecha que no tiene datos.

Ilustración 56 – Evidencia de error en incidencia si no existen datos

- Error al intentar introducir una incidencia en un día que no tiene datos.

Ilustración 57 - Evidencia de introducir incidencia en un día sin datos

6.2.5. Pantalla de registro:

- Se muestra error si se intenta registrar un usuario ya registrado.

Ilustración 58 - Evidencia de intentar registrar un usuario ya registrado

- Se muestra error si el correo no tiene la extensión "@hotmail.com", "@gmail.com", "@alum.us.es" o "@us.es".

Ilustración 59 - Evidencia de formato de correo no válido

- Se muestra error si la contraseña no tiene 8 caracteres o más.

Ilustración 60 - Evidencia de contraseña con tamaño incorrecto

- Se muestra error si las dos contraseñas introducidas NO coinciden.

Ilustración 61 - Evidencia de contraseñas no coincidentes

- Se muestra error si no se rellena el campo de motivo.

Ilustración 62 - Evidencia de motivo sin informar

- Se accede a la pantalla de registro correcto al darle a enviar si todos los campos cumplen los requisitos. Caso de registro de un estudiante.

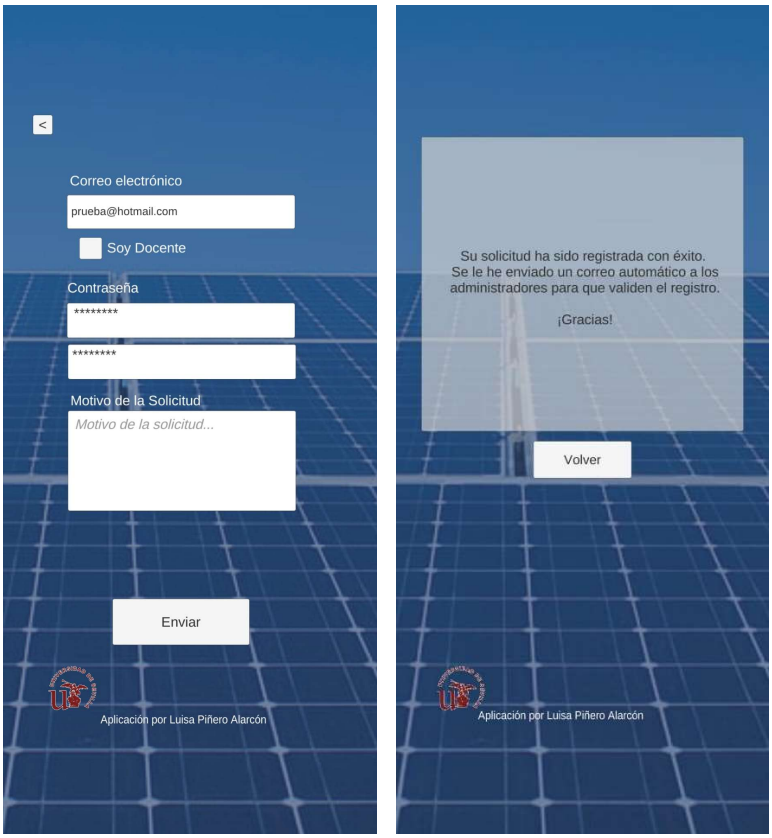

Ilustración 63 - Evidencia de registro correcto de estudiante

- Se añade a la base de datos con el campo acceso “no” ya que está pendiente.

	correo	password	rol	acceso
Editar Copiar Borrar	a@gmail.com	paquito90	alumno	no
Editar Copiar Borrar	admin	admin	profesor	si
Editar Copiar Borrar	alumno@alum.us.es	12345678	alumno	no
Editar Copiar Borrar	prueba@hotmail.com	12345678	alumno	no

Ilustración 64- Evidencia de inserción de nuevo usuario estudiante sin permisos

- Llega correo a los perfiles docentes avisando de un usuario nuevo que se ha registrado y está pendiente de confirmación.

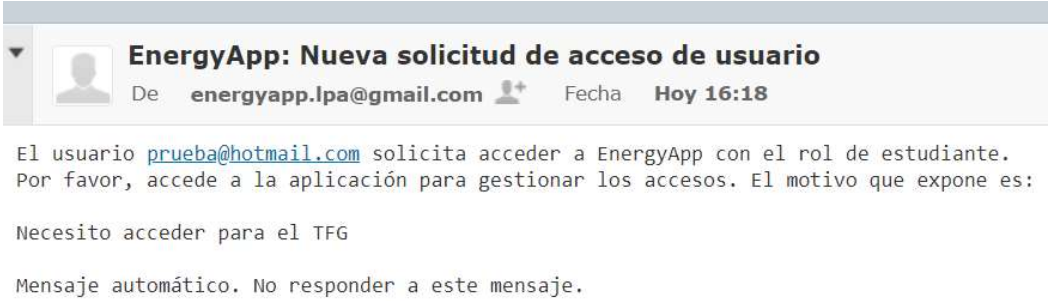

Ilustración 65 - Evidencia de correo de nuevo registro de usuario estudiante

- Se accede a la pantalla de registro correcto al darle a enviar si todos los campos cumplen los requisitos. Caso de registro de un docente.

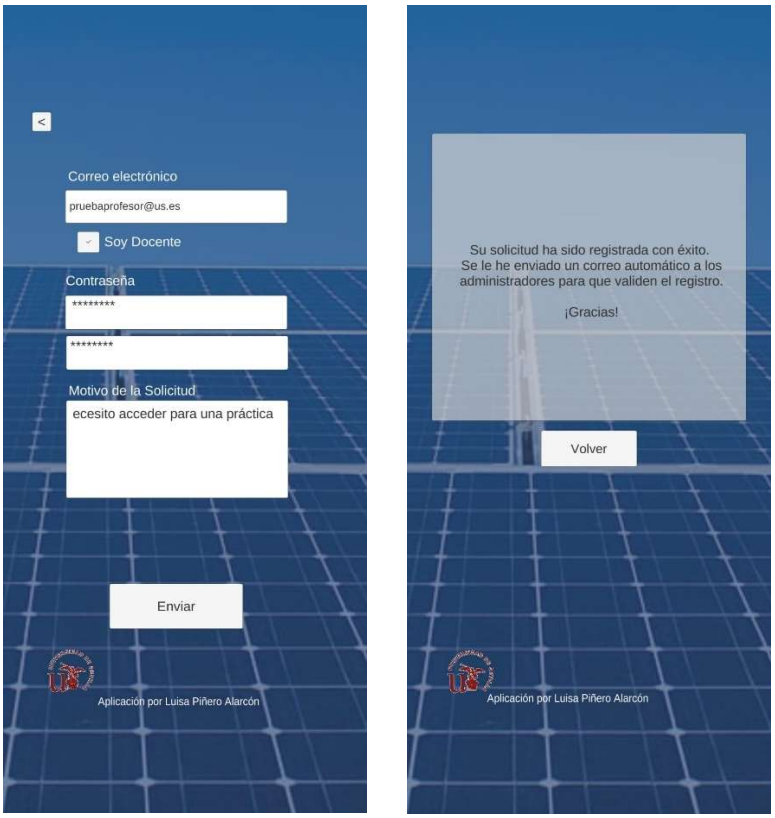

Ilustración 66 - Evidencia de registro correcto de docente

- Se añade a la base de datos con el campo acceso “no” ya que está pendiente.

Ilustración 67 - Evidencia de inserción de nuevo usuario docente sin permisos

- Llega un correo de aviso de nuevo usuario pendiente de aceptación a los perfiles “docente” que tienen acceso a la aplicación.

Ilustración 68 - Evidencia de correo de nuevo registro de usuario docente

- Una vez aceptado el perfil pasará a acceso “si”.

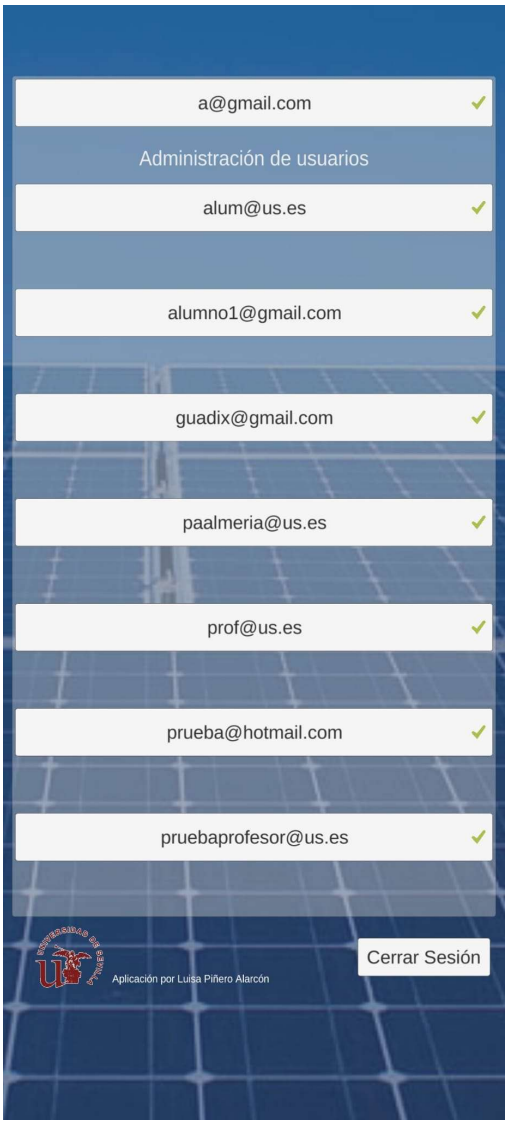

Ilustración 69 - Evidencia de usuarios pendientes de validación

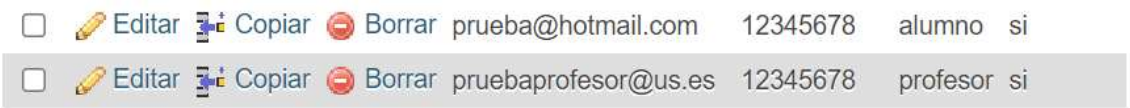

Ilustración 70 - Evidencia de permisos para acceder

- Estos nuevos usuarios ya pueden acceder a la aplicación.

Ilustración 71 - Evidencia de acceso de nuevo estudiante

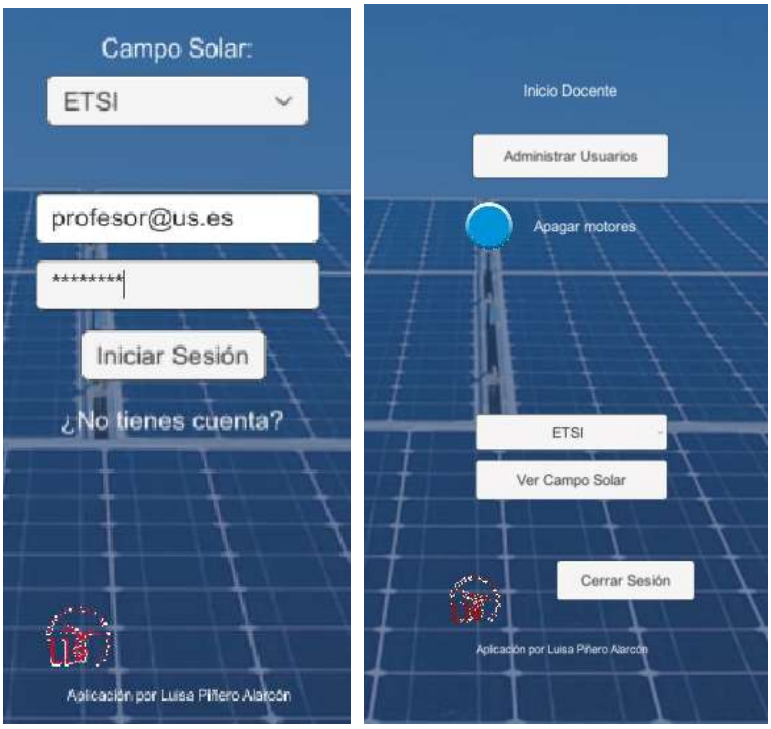

Ilustración 72 - Evidencia de acceso de nuevo docente

6.2.6. Pantalla de registro correcto:

- Se vuelve a la pantalla de acceso al darle al botón de aceptar.

Ilustración 73 - Evidencia de volver a la pantalla de acceso desde registro correcto

Se comprueba también que se visualiza correctamente cada pantalla (tanto en móvil como en ordenador), sin haber error de formato o espacios en blanco.

6.3. Resultados

Una vez realizado el *testing* exhaustivo de todas las pantallas de la aplicación podemos dar por finalizado su desarrollo. En este *testing* se ha detectado ciertos errores que se han solventado sobre la marcha, como variables con nombres incorrectos por lo cual no accedíamos a las bases de datos con el parámetro correcto, o error de selección de resolución de modo que al compilarlo para móvil se veían las pantallas achatadas con espacio libre arriba y debajo de la pantalla. También se han detectado botones de cierre de sesión o de retroceso que no tenían el script implementado y no realizaban su función, todos los errores se han solventado. Tras estas pequeñas correcciones tenemos una aplicación que aporta un nuevo medio para poder acceder a la planta solar en un futuro con pequeñas modificaciones, según el modo en el que la planta arroje sus datos.

7 CONCLUSIONES

En el presente trabajo se ha diseñado y desarrollado una aplicación para dispositivos móviles con el objeto de establecer comunicación con una planta instalada en la ETSI.
Con la aplicación desarrollada es posible monitorizar la planta en todo momento, actuar sobre ciertas variables, según la cualificación del personal registrado, así como gestionar y recibir notificaciones de alarmas y obtener datos históricos.

Todos los componentes de la planta solar se integran a través de un equipo supervisor (NAE). Dicha NAE posee una IP pública para que se pueda acceder (mediante usuario y contraseña) desde cualquier lugar al SCADA online Metasys, de la empresa Johnson Controls.
Metasys permite exportar tendencias y comunicación en tiempo real con formato .xlsx.

Dado que la planta solar se encuentra inactiva, con el objetivo de probar y validar los resultados, se ha llevado a cabo una conexión estandarizada entre la aplicación y bases de datos alojadas en una web, de manera que la aplicación desarrollada pueda comunicarse con la planta real en cuanto ésta esté operativa.

Como líneas futuras de investigación se pueden destacar las siguientes:

- Realizar pruebas con la planta real.

- Adaptar la aplicación al sistema operativo IOS para poder llegar a casi el 100% del mercado de los dispositivos móviles.

- Añadir *notificaciones push* [18]a la aplicación, de tal modo que nos salte, por ejemplo, cuando una alarma se enciende, de tal modo que podamos detectarlo sin tener que estar dentro de la aplicación en ese momento.

- Añadir la opción de poder seleccionar otros campos solares (o de cualquier otro tipo), con sus pantallas personalizadas y sus datos de interés correspondientes.

- Conseguir una conexión con Matlab, de cara a una futura conexión con una simulación de la planta, en el caso de que la conexión con la planta real no sea posible o en caso de que sea interesante conectar la aplicación a una planta en construcción que ya disponga de una planta simulada previa.

REFERENCIAS

[1] W. D. C. Salazar, *Modelado neuro-borroso de un sistema captador solar lineal tipo Fresnel como gemelo digital,* Sevilla, 2020.

[2] L. G. Parrilla, *Modelado y simulación de una planta solar,* Sevilla, 2019.

[3] J. J. L. Amores, *Desarrollo de sistema de calibrado automático de espejos de captador solar tipo Fresnel,* Sevilla, 2020.

[4] [En línea]. Available: https://www.electiaplus.es/apps-moviles-instalaciones-fotovoltaicas/.

[5] [En línea]. Available: https://nexusintegra.io/es/sistemas-scada-industria/.

[6] [En línea]. Available: https://sourceforge.net/software/scada/android/.

[7] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego).

[8] J. D. L. Castillo, Android studio Aprende a desarrollar aplicaciones, Libros RC.

[9] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/MATLAB.

[10] «SG,» [En línea]. Available: https://sg.com.mx/revista/31/la-importancia-la-claridad-y-sencillez-una-interfaz-usuario.

[11] «Kanbanize,» [En línea]. Available: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban.

[12] [En línea]. Available: https://docs.unity3d.com/es/530/Manual/UICanvas.html.

[13] [En línea]. Available: https://docs.unity3d.com/es/2018.4/ScriptReference/.

[14] «Meteogram,» [En línea]. Available: https://meteogram.es/sol/espana/sevilla/.

[15] [En línea]. Available: https://www.mathworks.com/matlabcentral/answers/196774-connection-between-matlab-and-unity3d.

[16] [En línea]. Available: https://www.youtube.com/watch?v=tz8WHs-rIf8.

[17] M. H. Núñez, *Diseño de APP Android para seguimiento de la web de Antiguos Alumnos de la ETSI,* Sevilla, 2016.

[18] «itsoftware,» [En línea]. Available: https://itsoftware.com.co/content/sistemas-operativos-mas-usados/.

[19] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Sistema_operativo_m%C3%B3vil#Android.

[20] «El Pais,» [En línea]. Available: https://elpais.com/tecnologia/2013/10/01/actualidad/1380620296_338056.html.

[21] [En línea]. Available: https://www.seoptimer.com/es/blog/notificaciones-push-que-son-por-que-usarlas/.

[22] [En línea]. Available: https://openautomationsoftware.com/?utm_source=sourceforge&utm_medium=website&utm_campaign=Sourceforge.

[23] [En línea]. Available: https://openautomationsoftware.com/?utm_source=sourceforge&utm_medium=website&utm_campaign=Sourceforge.

[24] [En línea]. Available: https://sourceforge.net/software/product/Fernhill-SCADA/.

ANEXO A: CÓDIGO SCENEMANAGER.CS (CAMBIAR ESCENA)

Este script sirve para cargar otra pantalla (scene) del programa, se le asigna a un botón al pulsarlo (en onclick), pasándole como parámetro la escena que se quiera abrir al realizar la acción.

```
using UnityEngine;

public class SceneManager : MonoBehaviour
{
    public string escena;
    // Carga una escena según el parámetro que se le pasa por Unity
    public void CargarEscena()
    {
        UnityEngine.SceneManagement.SceneManager.LoadScene(escena);
    }

}
```

ANEXO B: CÓDIGO LOGIN.CS

```
using MySql.Data.MySqlClient;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class Login : MonoBehaviour
{
    public InputField correoTxt;
    public InputField passwordTxt;
    public Text textoError;
    public Image fondoError;
    // Start is called before the first frame update

    public void Logear()
    {

        string _log = "SELECT * FROM"+" alumnos WHERE correo LIKE
'"+correoTxt.text+"' AND password LIKE '"+passwordTxt.text+"'";
        AdminMysql _adminMYSQL =
```

```
GameObject.Find("AdministradorBaseDatos").GetComponent<AdminMysql>();
        MySqlDataReader resultado = _adminMYSQL.Select(_log);

        if (resultado.HasRows){

            resultado.Close();
            string _profesor = "SELECT *"+" FROM alumnos WHERE correo LIKE
'"+correoTxt.text+"' AND rol LIKE 'profesor'";
            MySqlDataReader profesor = _adminMYSQL.Select(_profesor);
            if(profesor.HasRows){
                profesor.Close();
                string _acceso = "SELECT *"+" FROM alumnos WHERE correo LIKE
'"+correoTxt.text+"' AND acceso LIKE 'si'";
                MySqlDataReader acceso = _adminMYSQL.Select(_acceso);
                if (acceso.HasRows)
                {
                    Debug.Log("El profesor con correo "+correoTxt.text+"
tiene acceso");

UnityEngine.SceneManagement.SceneManager.LoadScene("EscenaBienvenidaProfesor"
);

                }
                else
                {
                    Debug.Log("El usuario profesor está pendiente de ser
validado por un profesor");
                    textoError.text = "Este usuario profesor se encuentra
pendiente de ser aceptado en la app";
                    fondoError.gameObject.SetActive(true);

                }
                acceso.Close();
            }
            else
            {
                profesor.Close();
                string _acceso = "SELECT *"+" FROM alumnos WHERE correo LIKE
'"+correoTxt.text+"' AND acceso LIKE 'si'";
                MySqlDataReader acceso = _adminMYSQL.Select(_acceso);
                if (acceso.HasRows)
                {
                    Debug.Log("El usuario con correo "+correoTxt.text+" tiene
acceso");

UnityEngine.SceneManagement.SceneManager.LoadScene("EscenaBienvenida");

                }
                else
                {
                    Debug.Log("El usuario está pendiente de ser validado por
un profesor");
                    textoError.text = "Este usuario se encuentra pendiente de
ser aceptado en la app";
                    fondoError.gameObject.SetActive(true);

                }
                acceso.Close();
            }
```

```
        }
        else
        {
            Debug.Log("Correo y/o contraseña incorrectos");
            textoError.text = "Correo y/o contraseña incorrectos";
            fondoError.gameObject.SetActive(true);
        }
        resultado.Close();

    }
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

ANEXO C: CÓDIGO REGISTRO.CS

```
using System.Collections.Generic;
using System.Data;
using MySql.Data.MySqlClient;
using UnityEngine;
using System.Net.Mail;
using UnityEditor;
using UnityEngine.UI;

public class Registro : MonoBehaviour
{
    public InputField correoTxt;
    public InputField passwordTxt;
    public InputField passwordTxt2;
    public InputField mensaje;
    public Text textoError;
    public Image fondoError;
    public Toggle checkProfesor;

    public void EnviarMail(List<string> receptores)
    {
        MailMessage mail = new MailMessage();
        SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com");
```

```
        mail.From = new MailAddress("energyapp.lpa@gmail.com");
        foreach (string rec in receptores)
        {
            mail.To.Add(rec);
        }


        mail.Subject = "Nueva solicitud de acceso de usuario";
        mail.Body = "El usuario " + correoTxt.text + " solicita acceder a
EnergyApp.\n" +
                    "Por favor, accede a la aplicación para gestionar los
accesos. El motivo que expone es: \n\n" +
                    mensaje.text +
                    "\n\nMensaje automático. No responder a este mensaje.";

        //System.Net.Mail.Attachment attachment;
        //attachment = new System.Net.Mail.Attachment("c:/textfile.txt");
        //mail.Attachments.Add(attachment);

        SmtpServer.Port = 587;
        SmtpServer.Credentials = new
System.Net.NetworkCredential("energyapp.lpa@gmail.com", "iticona15");
        SmtpServer.EnableSsl = true;

        SmtpServer.Send(mail);
    }

    private bool compruebaPass(string pass)
    {
        bool res = false;
        if (pass.Length >= 8)
        {
            Debug.Log("RESTRICCIÓN 1 CORRECTA: La contraseña insertada tiene
8 o más caracteres.");
            foreach (char c in pass)
            {
                if (char.IsDigit(c))
                {
                    Debug.Log("RESTRICCIÓN 2 CORRECTA: La contraseña
insertada contiene al menos un dígito.");
                    res = true;
                }
                else
                {
                    Debug.Log("RESTRICCIÓN 2 ERRÓNEA: La contraseña insertada
no contiene dígitos.");
                    textoError.text = "La contraseña debe tener al menos un
dígito";
                    fondoError.gameObject.SetActive(true);
                }
            }
        }
        else
        {
            Debug.Log("RESTRICCIÓN 1 ERRÓNEA: La contraseña insertada tiene
menos de 8 caracteres.");
            textoError.text = "La contraseña debe tener al menos 8
caracteres";
            fondoError.gameObject.SetActive(true);
```

```
        }

        return res;
    }

    private bool compruebaMotivo(string msg)
    {
        return msg.Length != 0;
    }

    public void Registrar()
    {
        string _log = "SELECT * FROM" + " alumnos WHERE correo LIKE '" +
correoTxt.text + "'";
        string _logAlumno = "INSERT INTO" + " alumnos (correo, password, rol,
acceso) values ('" + correoTxt.text +
                            "','" + passwordTxt.text + "','alumno','no')";
        string _logProfesor = "INSERT INTO" + " alumnos (correo, password,
rol, acceso) values ('" + correoTxt.text +
                                "','" + passwordTxt.text +
"','profesor','no')";
        AdminMysql _adminMYSQL =
GameObject.Find("AdministradorBaseDatos").GetComponent<AdminMysql>();
        MySqlDataReader resultado = _adminMYSQL.Select(_log);
        MySqlDataReader res;

        if (resultado.HasRows)
        {
            Debug.Log("El usuario con correo " + correoTxt.text + " ya
existe");
            textoError.text = "Existe un usuario con esta dirección de
correo. Inténtelo con otro diferente.";
            fondoError.gameObject.SetActive(true);
            resultado.Close();
        }
        else
        {
            if (correoTxt.text.EndsWith("@gmail.com") ||
correoTxt.text.EndsWith("@hotmail.com") ||
                correoTxt.text.EndsWith("@us.es") ||
correoTxt.text.EndsWith("@alum.us.es"))
            {
                if (passwordTxt.text != passwordTxt2.text)
                {
                    Debug.Log("Las contraseñás no coinciden");
                    textoError.text = "Las contraseñás no coinciden";
                    fondoError.gameObject.SetActive(true);
                    resultado.Close();
                }
                else
                {
                    if (compruebaPass(passwordTxt.text))
                    {
                        if (compruebaMotivo(mensaje.text))
                        {
                            Debug.Log("Se cumplen todas las restricciones
para el registro.");
                            if (checkProfesor.isOn)
                            {
```

```
                                resultado.Close();
                                Debug.Log("Insertando profesor " +
correoTxt.text + " en la Base de Datos.");
                                res = _adminMYSQL.Insert(_logProfesor);
                                res.Close();
                            }
                            else
                            {
                                resultado.Close();
                                Debug.Log("Insertando alumno " +
correoTxt.text + " en la Base de Datos.");
                                res = _adminMYSQL.Insert(_logAlumno);
                                res.Close();
                            }

                            string _correosProfesores =
                                "SELECT * FROM" + " alumnos WHERE rol LIKE
'profesor' AND acceso LIKE 'si'";

                            MySqlDataReader profs =
_adminMYSQL.Select(_correosProfesores);
                            List<string> correosProf = new List<string>();
                            while (profs.Read())
                            {
                                string email = (string) profs["correo"];
                                if (email.EndsWith("@gmail.com") ||
email.EndsWith("@hotmail.com") ||
                                    email.EndsWith("@us.es") ||
email.EndsWith("@alum.us.es"))
                                {
                                    correosProf.Add(email);
                                    Debug.Log("Correo destino: " + email);
                                }
                            }

                            //string[] profesores = new string[]{};
                            EnviarMail(correosProf);
                            Debug.Log("Correo informativo enviado a los
profesores.");

UnityEngine.SceneManagement.SceneManager.LoadScene("EscenaRegistroCorrecto");
                            Debug.Log("Cambio de escena a
EscenaRegistroCorrecto.");
                        }
                        else
                        {
                            Debug.Log("No está relleno el campo de texto
'Motivo'.");
                            textoError.text =
                                "Es obligatorio indicar un motivo";
                            fondoError.gameObject.SetActive(true);
                            resultado.Close();
                        }
                    }
                    else
                    {
                        resultado.Close();
                    }
                }
```

```
            }
            else
            {
                Debug.Log("El dominio de correo no es '@alum.us.es',
'@us.es', '@gmail.com' ni '@hotmail.com'.");
                textoError.text =
                    "Solo se permiten los siguientes correos de dominio:
'@alum.us.es', '@us.es', '@gmail.com', '@hotmail.com'.";
                fondoError.gameObject.SetActive(true);
                resultado.Close();
            }
        }
    }
}
```

ANEXO D: CÓDIGO DIRECTO.CS

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Net.Mail;
using UnityEngine;
using UnityEngine.Rendering;
using UnityEngine.UI;

public class Directo : MonoBehaviour
{
    //Variables para la comunicación PHP
    public string getURL = "http://energyapp.eu5.org/obtenerDirecto.php";

 //Variables para la interpretación en pantalla

    public Image alarmaTs;
    public Image alarmaP;
    public Image alarmaM;
    public Text acumulacionMax;
    public Text tecMed;
    public Text tscMed;
    public Text tstMed;
    public Text tAmb;
    public Text presionMed;
    public Text energiaGen;
    public Sprite verde;
    public Sprite rojo;
    public Text motoresTxt;

   /* void Start()
    {
        StartCoroutine(cUpdate());
```

```
    }

    IEnumerator cUpdate()
    {
        while(true)
        {
            //Codigo
            StartCoroutine("LoadData");

            //Esperamos el tiempo definido que tenga 'intervalo'
            yield return new WaitForSeconds(5);
        }
    }
*/
    //Los botones llaman a las Coroutines
   public void Recibir()
    {
        StartCoroutine("LoadData");
    }


    //Coroutines para la comunicación con la web y mostrar datos recogidos de
la BD
    private IEnumerator LoadData()
    {
        Debug.Log("Entro en LoadData para cargar DATOS");


        //Debug.Log("fecha seleccionada en unity: "+_dia+"/"+_mes+"/"+_ano);
        //mostrar datos de entorno aquí
        string urlString2 = getURL;
        WWW getFecha = new WWW(urlString2);

        //Esperamos hasta que haya una respuesta (por eso usamos Coroutines)
        yield return getFecha;
        Debug.Log("BD me devuelve " + getFecha.text);


            string[] cadena = getFecha.text.Split(';');

            string motor1 = cadena[0];
            string motor2 = cadena[1];
            string motor3 = cadena[2];
            string motor4 = cadena[3];
            string motor5 = cadena[4];
            string motor6 = cadena[5];
            string motor7 = cadena[6];
            string motor8 = cadena[7];
            string motor9 = cadena[8];
            string motor10 = cadena[9];
            string motor11= cadena[10];
            string motor12= cadena[11];
            string motor13 = cadena[12];
            string motor14= cadena[13];
            string motor15= cadena[14];
            string motor16= cadena[15];
            string motor17= cadena[16];
            string motor18= cadena[17];
            string motor19= cadena[18];
```

```
            string motor20= cadena[19];
            string motor21= cadena[20];
            string motor22= cadena[21];
            string[] motores =
{motor1,motor2,motor3,motor4,motor5,motor6,motor7,motor8,motor9,motor10,motor
11,motor12,motor13,motor14,motor15,motor16,motor17,motor18,motor19,motor20,mo
tor21,motor22};
            string alM = "0";
            string motoresApagados = "";
            for (int i=0;i<motores.Length;i++)
            {
                Debug.Log("Motor"+(i+1)+" está a "+motores[i]);
                if (motores[i] == "0")
                {
                    alM = "1";
                    motoresApagados += "->"+(i+1)+" ";

                }
            }

            motoresTxt.text = motoresApagados;
            acumulacionMax.text = cadena[22]+ " kW.h";
            tecMed.text = cadena[23] + " ºC";
            tscMed.text = cadena[24] + " ºC";
            tstMed.text = cadena[25] + " ºC";
            tAmb.text = cadena[26] + " ºC";
            presionMed.text = cadena[27] + " bar";
            energiaGen.text = cadena[28] + "kW.h";

            string alTs = "0";

            string alP = "0";
            double tsc = Double.Parse(cadena[24]);
            double pres = Double.Parse(cadena[27]);
            Debug.Log("LA presion es: "+pres+ " y la temp es: "+ tsc);
            if (tsc > 190.0)
            {
                alTs = "1";
            }

            if (pres>15.0)
            {
                alP = "1";
            }

            Debug.Log("La alarma TS es: "+alTs);
            Debug.Log("La alarma M es:" + alM);
            Debug.Log("La alarma P es: "+alP);
            coloresAlarma(alTs,alP,alM);




    }


    private void coloresAlarma(string alarmaTsTxt, string alarmaPTxt, string
alarmaMTxt)
```

```
    {
        if (alarmaTsTxt=="0")
        {
            alarmaTs.sprite = verde;
        }
        else
        {
            alarmaTs.sprite = rojo;
        }

        if (alarmaPTxt=="0")
        {
            alarmaP.sprite = verde;
        }
        else
        {
            alarmaP.sprite = rojo;
        }

        if (alarmaMTxt=="0")
        {
            alarmaM.sprite = verde;
        }
        else
        {
            alarmaM.sprite = rojo;
        }
    }


}
```

ANEXO E: CÓDIGO DATOS.CS

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Net.Mail;
using UnityEngine;
using UnityEngine.Rendering;
using UnityEngine.UI;

public class DatosSolares : MonoBehaviour
{
    //Variables para la comunicación PHP
    public string postURL = "http://energyapp.eu5.org/introducirDatos.php";
    public string getURL = "http://energyapp.eu5.org/obtenerDatos.php";
```

```
    private string _comentario = "";
    private string _ano = "";
    private string _mes = "";
    private string _dia = "";


    //Variables para la interpretación en pantalla
    public InputField comentario;
    public Text dia;
    public Text mes;
    public Text ano;
    public Text fecha;
    public Text salidaSol;
    public Text puestaSol;
    public Text horasSolares;
    public Image alarmaTs;
    public Image alarmaP;
    public Image alarmaM;
    public Text acumulacionMax;
    public Text tecMed;
    public Text tscMed;
    public Text tstMed;
    public Text tAmb;
    public Text presionMed;
    public Text energiaGen;
    public Text incidencia;
    public Sprite verde;
    public Sprite rojo;
    public InputField incidenciaActualizada;



    //Los botones llaman a las Coroutines
    public void Enviar()
    {
        StartCoroutine("SaveData");
    }

    public void Recibir()
    {
        StartCoroutine("LoadData");
    }


    private IEnumerator SaveData()
    {
        //Debug.Log("Entro en SaveData");


        string urlString2 = getURL + "?" + "fecha=" + WWW.EscapeURL(_ano) +
"-" + WWW.EscapeURL(_mes) + "-" +
                            WWW.EscapeURL(_dia);
        Debug.Log("Llamando a URL: " + urlString2);
        WWW getFecha = new WWW(urlString2);

        //Esperamos hasta que haya una respuesta (por eso usamos Coroutines)
        yield return getFecha;
        if (getFecha.text.Trim() == "")
```

```
        {
            incidencia.color=Color.red;
            incidencia.text = "NO ES POSIBLE REGISTRAR LA NUEVA INCIDENCIA.
ES POSIBLE QUE EL DÍA ESPECIFICADO NO ESTÉ REGISTRADO.";
        }
        else
        {
            if (incidenciaActualizada.text.Trim() == "")
            {
                incidencia.color=Color.red;
                incidencia.text = "DEBE INSERTAR UN COMENTARIO.";
            }
            else
            {
                _comentario = incidenciaActualizada.text;
                string url = postURL + "?" + "incidencia=" +
WWW.EscapeURL(_comentario)+"&"+"fecha="+WWW.EscapeURL(fecha.text);
                Debug.Log("Enviando URL NUEVA:  "+url);
                incidencia.color=Color.red;
                incidencia.text = "Procesando...";
                WWW usuarioExiste = new WWW(url);
                yield return usuarioExiste;
                Debug.Log("BD devuelve: "+usuarioExiste.text);
                incidencia.color=Color.black;
                incidencia.text = _comentario;
            }


        }



    }









    //Coroutines para la comunicación con la web y mostrar datos recogidos de
la BD
    private IEnumerator LoadData()
    {
        Debug.Log("Entro en LoadData para cargar DATOS");

        _dia = dia.text;
        _mes = mes.text;
        _ano = ano.text;
        //Debug.Log("fecha seleccionada en unity: "+_dia+"/"+_mes+"/"+_ano);
        //mostrar datos de entorno aquí
        string urlString2 = getURL + "?" + "fecha=" + WWW.EscapeURL(_ano) +
"-" + WWW.EscapeURL(_mes) + "-" +
                            WWW.EscapeURL(_dia);
        Debug.Log("Llamando a URL: " + urlString2);
        WWW getFecha = new WWW(urlString2);

        //Esperamos hasta que haya una respuesta (por eso usamos Coroutines)
```

```
        yield return getFecha;
        Debug.Log("BD me devuelve " + getFecha.text);
        if (getFecha.text.Trim() == "")
        {
            incidencia.color=Color.red;
            incidencia.text = "ERROR AL CARGAR LOS DATOS. ES POSIBLE QUE EL
DÍA ESPECIFICADO NO ESTÉ REGISTRADO.";

        }
        else
        {
            string[] cadena = getFecha.text.Split(';');
            fecha.text = cadena[0];
            salidaSol.text = cadena[1];
            puestaSol.text = cadena[2];
            horasSolares.text =
calcularHorasEntre(salidaSol.text,puestaSol.text);
            string alarmaTsTxt = cadena[3];
            string alarmaPTxt = cadena[4];
            string alarmaMTxt = cadena[5];
            coloresAlarma(alarmaTsTxt,alarmaPTxt,alarmaMTxt);
            acumulacionMax.text = calcularPorcentajeAcum(cadena[6]);
            tecMed.text = cadena[7] + " ºC";
            tscMed.text = cadena[8] + " ºC";
            tstMed.text = cadena[9] + " ºC";
            tAmb.text = cadena[10] + " ºC";
            presionMed.text = cadena[11] + " bar";
            energiaGen.text = cadena[12] + "kW.h";
            incidencia.color=Color.black;
            incidencia.text = cadena[13];

        }

    }

    private string calcularPorcentajeAcum(string s)
    {
        double num=Double.Parse(s);
        double res = num * 100 / 292;
        return res.ToString().Substring(0,4)+" %";
    }

    private void coloresAlarma(string alarmaTsTxt, string alarmaPTxt, string
alarmaMTxt)
    {
        if (alarmaTsTxt=="0")
        {
            alarmaTs.sprite = verde;
        }
        else
        {
            alarmaTs.sprite = rojo;
        }

        if (alarmaPTxt=="0")
        {
            alarmaP.sprite = verde;
        }
        else
```

```
        {
            alarmaP.sprite = rojo;
        }

        if (alarmaMTxt=="0")
        {
            alarmaM.sprite = verde;
        }
        else
        {
            alarmaM.sprite = rojo;
        }
    }

    private string calcularHorasEntre(string horaSalida, string horaPuesta)
    {
        Debug.Log("Entrando en calcularHorasEntre");
        DateTime hSalida=DateTime.Parse(horaSalida);
        DateTime hPuesta=DateTime.Parse(horaPuesta);
        TimeSpan result = hPuesta.Subtract(hSalida);
        int horaEntera=result.Hours;
        int minutosEntero = result.Minutes % 60;
        Debug.Log("HORA: "+horaEntera+ " MINUTOS: "+minutosEntero);
        string horas= Convert.ToString(result.TotalHours);
       // string[] cadena = horas.Split(',');
        //Debug.Log("cadena1 (minutos) son: "+cadena[1].Substring(0,2));
        //double minutos = Double.Parse(cadena[1].Substring(0,2))/100 * 60;

        //string min = minutos.ToString().Substring(0,2);
        return horaEntera.ToString() + "h y " + minutosEntero.ToString() +
"min";
    }


}
```

ANEXO F: CÓDIGO LISTARUSUARIOS.CS

```
using System.Collections.Generic;
using MySql.Data.MySqlClient;
using UnityEngine;
using UnityEngine.UI;
using Button = UnityEngine.UIElements.Button;

public class ListarUsuarios : MonoBehaviour
{
    private string datosConexion;
    private MySqlConnection conexion;
```

```
    // Función que lista los usuarios pendientes de acceso
    private List<string> Listar()
    {
        string _log = "SELECT * FROM"+" alumnos WHERE acceso LIKE 'no'";
        Debug.Log("QUERY: "+_log);
        MySqlCommand cmd = conexion.CreateCommand();
        cmd.CommandText = _log;
        MySqlDataReader listado = cmd.ExecuteReader();




        List<string> listadoUsuariosPendientes = new List<string>();
        while (listado.Read())
        {
            string correo = (string) listado["correo"];
            listadoUsuariosPendientes.Add(correo);
            Debug.Log("Se añade el correo "+correo+" a la lista de usuarios
sin acceso.");
        }

        listado.Close();
        return listadoUsuariosPendientes;
    }


    private void ConectarConServidorBaseDatos()
    {
        conexion = new MySqlConnection(datosConexion);

        try
        {
            conexion.Open();
            Debug.Log("Conexión con BD correcta!");
        }
        catch(MySqlException error)
        {
            Debug.LogError("Imposible conectar con la Base de Datos:
"+error);
        }
    }
    void Start()
    {
        datosConexion = "Server=remotemysql.com"
                                    + ";Database=JeJBFPR3Uy"
                                    + ";Uid=JeJBFPR3Uy"
                                    + ";Pwd=9CJzhSsyeh"
                                    + ";";
        ConectarConServidorBaseDatos();
        Debug.Log("Estamos en el START");
        GameObject buttonTemplate = transform.GetChild(0).gameObject;
        GameObject g;
        List<string> usuariosPendientes = Listar();
        for (int i = 0; i < usuariosPendientes.Count; i++)
        {
            g = Instantiate(buttonTemplate, transform);
            g.transform.GetChild(0).GetComponent<Text>().text =
usuariosPendientes[i];
```

```
        }
        Destroy(buttonTemplate);

    }
}
```

ANEXO G: CÓDIGO NETWORKMANAGER.CS

```
using System.Collections;
using System.Collections.Generic;
using System.Net.Mail;
using UnityEngine;
using UnityEngine.UI;

public class NetworkManager : MonoBehaviour
{
    //Variables para la comunicación PHP
    public string postURL = "http://energyapp.eu5.org/crearUsuario.php";
    public string getURL = "http://energyapp.eu5.org/loginUsuario.php";
    public string getCorreo = "http://energyapp.eu5.org/existeUsuario.php";
    public string getProfesoresURL =
"http://energyapp.eu5.org/profesoresAcceso.php";

    private string _correo = "";
    private string _password = "";
    private string _rol = "";

    //Variables para la interpretación en pantalla
    public InputField correo;
    public InputField contraseña;
    public InputField contraseñaRepetida;
    public InputField motivo;
    public Text textoError;
    public Toggle checkProfesor;

    //Los botones llaman a las Coroutines
    public void Enviar()
    {
        StartCoroutine("SaveUser");
    }

    public void Recibir()
    {
        StartCoroutine("LoadUser");
    }
```

```
    //Coroutines para la comunicación con la web y mostrar datos recogidos de
la BD
    private IEnumerator LoadUser()
    {
      Debug.Log("Entro en LoadUser");
      Debug.Log("Cogiendo pass desde la URL "+getURL);
      textoError.text = "Procesando...";

      _correo = correo.text;
      //mostrar datos de entorno aquí
      string urlString = getURL + "?" + "correo=" + WWW.EscapeURL(_correo);
      WWW getCorreo = new WWW(urlString);

      //Esperamos hasta que haya una respuesta (por eso usamos Coroutines)
      yield return getCorreo;
      Debug.Log("La pass es en BD: "+getCorreo.text);
      Debug.Log("La pass escrita en la app es: "+contraseña.text);
      string[] cadenaDB = new string[3];
      string passDB = "";
      string rolDB = "";
      string accesoDB = "";
      if (getCorreo.text.Split().Length==3)
      {
          cadenaDB = getCorreo.text.Split(' ');
          passDB = cadenaDB.GetValue(1).ToString();
          rolDB = cadenaDB.GetValue(0).ToString();
          accesoDB = cadenaDB.GetValue(2).ToString();

          if (contraseña.text==passDB)
          {
              if (accesoDB == "no")
              {
                  textoError.text = "Este usuario se encuentra pendiente de
validación de acceso";
              }
              else
              {
                  Debug.Log("La pass en BD es: "+passDB);
                  Debug.Log("El rol en BD es: "+rolDB);
                  Debug.Log("Entro en el if");
                  Debug.Log("Las contraseñas son iguales");
                  if (contraseña.text=="")
                  {
                      Debug.Log("La contraseña está vacía");
                      textoError.text = "Debe introducir una contraseña";

                  }
                  else
                  {
                      Debug.Log("La contraseña no está vacía");
                      Debug.Log("El valor de la variable _log es: "+ _rol);
                      if (rolDB=="profesor")
                      {
                          Debug.Log("PROFESOR");

UnityEngine.SceneManagement.SceneManager.LoadScene("EscenaBienvenidaProfesor"
);
                      }
                      else
```

```
                    {
                        Debug.Log("ALUMNO");
UnityEngine.SceneManagement.SceneManager.LoadScene("EscenaBienvenida");
                    }

                }
            }



        }
        else
        {
            Debug.Log("Las contraseñas son distintas o no encuentra
correo");
            textoError.text = "Correo y/o contraseña incorrectos";
        }
    }
    else
    {
        textoError.text = "Correo y/o contraseña incorrectos";
    }




}


public void EnviarMail(List<string> receptores)
{
    MailMessage mail = new MailMessage();
    SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com");
    mail.From = new MailAddress("energyapp.lpa@gmail.com");
    foreach (string rec in receptores)
    {
        mail.To.Add(rec);
    }


    mail.Subject = "EnergyApp: Nueva solicitud de acceso de usuario";
    mail.Body = "El usuario " + correo.text + " solicita acceder a
EnergyApp.\n" +
                "Por favor, accede a la aplicación para gestionar los
accesos. El motivo que expone es: \n\n" +
                motivo.text +
                "\n\nMensaje automático. No responder a este mensaje.";

    //System.Net.Mail.Attachment attachment;
    //attachment = new System.Net.Mail.Attachment("c:/textfile.txt");
    //mail.Attachments.Add(attachment);

    SmtpServer.Port = 587;
    SmtpServer.Credentials = new
System.Net.NetworkCredential("energyapp.lpa@gmail.com", "iticona15");
    SmtpServer.EnableSsl = true;

    SmtpServer.Send(mail);
```

```
    }


    private bool correoCorrecto(string correo)
    {
        bool res = false;
        if (correo.EndsWith("@gmail.com") || correo.EndsWith("@alum.us.es")
|| correo.EndsWith("@us.es") || correo.EndsWith("hotmail.com"))
        {
            if (!correo.StartsWith("@"))
            {
                res = true;
            }
            else
            {
                textoError.text = "Introduzca un correo válido";
            }
        }
        else
        {
            textoError.text="Los dominios válidos son '@alum.us.es',
'@us.es', 'gmail.com' y 'hotmail.com'";
        }
        return res;
    }

    //Coroutine para comunicarse con la BD e introducir los datos que se
envian desde Unity
    //Lo único que hace Unity desde aquí es enviar los datos a la página web
    private IEnumerator SaveUser()
    {
        //Debug.Log("Entro en SaveUser");
        textoError.text = "Procesando...";
        _correo = correo.text;
        Debug.Log("_correo es LO SIGUIENTE: " + _correo);
        string url=getCorreo + "?" + "correo=" + WWW.EscapeURL(_correo);
        Debug.Log("Enviando URL NUEVA:  "+url);
        WWW usuarioExiste = new WWW(url);
        yield return usuarioExiste;
        Debug.Log("usuarioExiste.text devuelve: "+usuarioExiste.text);
        if (usuarioExiste.text != "")
        {
            textoError.text = "El usuario con el correo introducido ya
existe";
        }
        else
        {
            if (correoCorrecto(correo.text))
        {
            if (contraseña.text.Length>=8 && !contraseña.text.Contains(" "))
            {
                if (contraseña.text == contraseñaRepetida.text)
                {
                    //Debug.Log("Las contraseñas coinciden");
                    _correo = correo.text;
                    _password = contraseña.text;
                    if (checkProfesor.isOn)
                    {
```

```
                        _rol = "profesor";
                    }
                    else
                    {

                        _rol = "alumno";
                    }

                    string urlString = postURL + "?" + "correo=" +
WWW.EscapeURL(_correo) + "&" + "password=" +
                                        WWW.EscapeURL(_password) + "&" +
"rol=" + WWW.EscapeURL(_rol);

                    //Debugpara verde donde viene la data
                    Debug.Log("Enviando: "+urlString);



                    // pasamos la data al servidor

                    WWW postUser = new WWW(urlString);

                    yield return postUser;

                    //INICIO CODIGO ANTIGUO
                    WWW getProfesoresActivos = new WWW(getProfesoresURL);
                    yield return getProfesoresActivos;


                    string[] arrayCorreos = getProfesoresActivos.text.Split('
');
                    List<string> correosProf = new List<string>();


                    foreach (string s in arrayCorreos)
                    {
                        if (s.EndsWith("@gmail.com") ||
s.EndsWith("@hotmail.com") ||
                            s.EndsWith("@us.es") ||
s.EndsWith("@alum.us.es"))
                        {
                            correosProf.Add(s);
                            //Debug.Log("Correo destino: " + s);
                        }
                    }

                    EnviarMail(correosProf);
                    //Debug.Log("Correo informativo enviado a los
profesores.");


                    /*
                    List<string> correosProf = new List<string>();
                    while (correosProfesores.Read())
                    {
                        string email = (string) profs["correo"];
                        if (email.EndsWith("@gmail.com") ||
email.EndsWith("@hotmail.com") ||
                            email.EndsWith("@us.es") ||
```

```
email.EndsWith("@alum.us.es"))
                        {
                            correosProf.Add(email);
                            Debug.Log("Correo destino: " + email);
                        }
                    }

                    //string[] profesores = new string[]{};
                    EnviarMail(correosProf);
                    Debug.Log("Correo informativo enviado a los
profesores.");

UnityEngine.SceneManagement.SceneManager.LoadScene("EscenaRegistroCorrecto");
                    Debug.Log("Cambio de escena a
EscenaRegistroCorrecto.");*/
                    //FIN CODIGO ANTIGUO


UnityEngine.SceneManagement.SceneManager.LoadScene("EscenaRegistroCorrecto");
                }
                else
                {
                    textoError.text = "Las contraseñas no coinciden";
                }

            }
            else
            {
                textoError.text = "La contraseña debe contener al menos 8
carcateres. Los espacios no están permitidos";
            }
        }
        }









    }
}
```

ANEXO H: CÓDIGO BOTON.CS

```
using System;
using System.Collections.Generic;
using System.Net.Mail;
using MySql.Data.MySqlClient;
using UnityEngine;
using UnityEngine.UI;
```

```
public class Boton : MonoBehaviour
{
    public void EnviarMail(string correo)
    {
        Debug.Log("SE VA A ENVIAR UN CORREO A: "+correo);
        MailMessage mail = new MailMessage();
        SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com");
        mail.From = new MailAddress("energyapp.lpa@gmail.com");
        mail.To.Add(correo);


        mail.Subject = "EnergyApp: Solicitud de acceso aceptada";
        mail.Body = "El usuario " + correo + " ha sido aceptado.\n" +
                    "Por favor, comprueba el acceso en la aplicación." +
                    "\n\nMensaje automático. No responder a este mensaje.";

        //System.Net.Mail.Attachment attachment;
        //attachment = new System.Net.Mail.Attachment("c:/textfile.txt");
        //mail.Attachments.Add(attachment);

        SmtpServer.Port = 587;
        SmtpServer.Credentials = new
System.Net.NetworkCredential("energyapp.lpa@gmail.com", "iticona15");
        SmtpServer.EnableSsl = true;

        SmtpServer.Send(mail);
    }
    public void AceptarUsuario()
    {
        GameObject corr = transform.GetChild(0).gameObject;
        string correo = corr.GetComponent<Text>().text;
        Debug.Log("Has seleccionado aceptar a "+correo);


        string _logAceptar = "UPDATE alumnos" + " SET acceso='si' WHERE
correo='" + correo+"'";
        AdminMysql _adminMYSQL =
GameObject.Find("AdministradorBaseDatos").GetComponent<AdminMysql>();
        MySqlDataReader resultado = _adminMYSQL.Select(_logAceptar);
        EnviarMail(correo);
        resultado.Close();

UnityEngine.SceneManagement.SceneManager.LoadScene("EscenaAdministracion");

    }

}
```

ANEXO I: CÓDIGO ADMINMYSQL.CS

```
using MySql.Data.MySqlClient;
using UnityEngine;

public class AdminMysql : MonoBehaviour
{
    public string servidorBaseDatos;
    public string nombreBaseDatos;
    public string usuarioBaseDatos;
    public string contraseñaBaseDatos;
    private string datosConexion;
    private MySqlConnection conexion;

    // Start is called before the first frame update
    void Start()
    {
        datosConexion = "Server=" + servidorBaseDatos
            + ";Database="+nombreBaseDatos
            + ";Uid=" + usuarioBaseDatos
            + ";Pwd=" + contraseñaBaseDatos
            + ";";
        ConectarConServidorBaseDatos();
    }

    private void ConectarConServidorBaseDatos()
    {
        conexion = new MySqlConnection(datosConexion);

        try
        {
            conexion.Open();
            Debug.Log("Conexión con BD correcta!");
        }
        catch(MySqlException error)
        {
            Debug.LogError("Imposible conectar con la Base de Datos:
"+error);
        }
    }

    public MySqlDataReader Select(string _select)
    {
        MySqlCommand cmd = conexion.CreateCommand();
        Debug.Log("conexion.CreateCommand()...");
        cmd.CommandText = _select;
        Debug.Log("cmd.CommandText = _select;...");
        MySqlDataReader resultado = cmd.ExecuteReader();
        Debug.Log("cmd.ExecuteReader()...");
        return resultado;
    }

    public MySqlDataReader Insert(string _select)
    {
        MySqlCommand cmd = conexion.CreateCommand();
        cmd.CommandText = _select;
        MySqlDataReader res = cmd.ExecuteReader();
        return res;
```

```
    }
}
```

ANEXO I: CÓDIGOS PHP

crearUsuario.php:

```
<?php
    //Incluimos archivo para conectar
            include "dbConnection.php";

    //Cogemos el valor de estas variables que nos envía Unity
    $correo = $_GET['correo'];
    $password= $_GET['password'];
    $rol = $_GET['rol'];

    //Mostramos el valor en el Log
    echo 'Correo = '.$correo;
    echo 'Pass = '.$password;
    echo 'Rol = '.$rol;

    //Insertamos valores en DB
    $stmt = $pdo->prepare("INSERT INTO alumnos(correo, password,rol,acceso)
VALUES(:field1,:field2,:field3,'no')");
    $stmt->execute(array(':field1' => $correo, ':field2' => $password,':field3' => $rol));

    //Para comprobar la introducción de datos desde la página a modo de log, mostraremos los
datos de la BD
    foreach($pdo->query('SELECT * FROM alumnos') as $row){
        echo $row['correo'].' '.$row['password'].'</br>' ;
```

```
    }

?>
```

dbConnection.php:

```
<?php

        $pdo = new PDO('mysql:host=remotemysql.com;
dbname=JeJBFPR3Uy;charset=utf8','JeJBFPR3Uy','9CJzhSsyeh');
        $pdo->setAttribute(PDO::ATTR_ERRMODE , PDO:: ERRMODE_EXCEPTION);
        $pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES,false);

?>
```

existeUsuario.php:

```
<?php
    include("dbConnection.php");

    $correo = $_GET['correo'];
    //PARA MOSTRAR LOS RESULTADOS

    foreach($pdo->query("SELECT correo FROM alumnos WHERE correo = '$correo'") as
$row){
        echo $row['correo'];
    }

?>
```

introducirDatos.php:

```
<?php
        include("dbConnection.php");
        $fecha=$_GET['fecha'];
        $incidencia= $_GET['incidencia'];
        //PARA MOSTRAR LOS RESULTADOS

    //echo $incidencia;
    //echo $fecha


        $pdo->query("UPDATE historicoDatos SET incidencia='$incidencia' WHERE fecha =
'$fecha'");
        //  foreach($pdo->query("UPDATE historicoDatos(incidencia) VALUES($incidencia)
WHERE fecha = $fecha") as $row){
        //    echo $row['rol'].' '.$row['password'];
        //}

?>
```

loginUsuario.php:

```
<?php
        include("dbConnection.php");

        $correo = $_GET['correo'];
        //$password = $_GET['password'];
        //PARA MOSTRAR LOS RESULTADOS

        foreach($pdo->query("SELECT password,rol FROM alumnos WHERE correo =
'$correo'") as $row){
```

```
            echo $row['rol'].' '.$row['password'];
        }


?>
```

obtenerDatos.php:

```
<?php
        include("dbConnection.php");


        $fecha= $_GET['fecha'];
        //PARA MOSTRAR LOS RESULTADOS
      //echo $fecha;



        foreach($pdo->query("SELECT * FROM historicoDatos WHERE fecha= '$fecha'") as
$row){
            echo
$row['fecha'].';'.$row['salidaSol'].';'.$row['puestaSol'].';'.$row['alarmaTs'].';'.$row['alarmaP'].';'.$r
ow['alarmaM'].';'.$row['acumulacionMax'].';'.$row['tecMed'].';'.$row['tscMed'].';'.$row['tstMed'].
';'.$row['TambMed'].';'.$row['presionMed'].';'.$row['energiaGen'].';'.$row['incidencia'];
        }


?>
```

obtenerDirecto.php:

```
<?php
        include("dbConnection.php");
```

```
        foreach($pdo->query("SELECT * FROM datosDirecto") as $row){

            echo
$row['motor1'].';'.$row['motor2'].';'.$row['motor3'].';'.$row['motor4'].';'.$row['motor5'].';'.$row['
motor6'].';'.$row['motor7'].';'.$row['motor8'].';'.$row['motor9'].';'.$row['motor10'].';'.$row['motor
11'].';'.$row['motor12'].';'.$row['motor13'].';'.$row['motor14'].';'.$row['motor15'].';'.$row['motor1
6'].';'.$row['motor17'].';'.$row['motor18'].';'.$row['motor19'].';'.$row['motor20'].';'.$row['motor21
'].';'.$row['motor22'].';'.$row['tanqueAlmacenamiento'].';'.$row['tempTec'].';'.$row['tempTsc'].';'.
$row['tempTst'].';'.$row['tempTamb'].';'.$row['presion'].';'.$row['energiaGenerada'];

        }


?>
```

profesoresAcceso.php:

```
<?php
        include("dbConnection.php");


        $correo = $_GET['correo'];

        //$password = $_GET['password'];

        //PARA MOSTRAR LOS RESULTADOS


        foreach($pdo->query("SELECT correo FROM alumnos WHERE rol LIKE 'profesor'
AND acceso LIKE 'si'") as $row){

            echo $row['correo'].' ';

        }



/*<?php
        include("dbConnection.php");


        $correo = $_GET['correo'];

        $password = $_GET['password'];

        //PARA MOSTRAR LOS RESULTADOS
```

```
        foreach($pdo->query("SELECT password FROM alumnos WHERE correo =
'$correo'") as $row){
            echo $row['correo'].' '.$row['password'];
        }



 ?>*/
?>
```