PROYECTO DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

“CONTROL DE PRESENCIA”

MIGUEL RUBIO MUÑOZ

CURSO 2020/21

Identificación del autor, estudios y centro.

Este proyecto ha sido realizado por Miguel Rubio Muñoz, alumno de 2º curso del Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Multiplataforma en el centro I.E.S. Virgen del Carmen, de la ciudad de Jaén.

El presente proyecto integrado ha sido supervisado por Don Manuel Molino Milla, profesor del departamento de Informática del centro.

Introducción.

Se hace cada vez más necesario la automatización de tareas en todos los niveles de nuestra sociedad, como nos ha demostrado la pandemia que estamos sufriendo, evitando el uso compartido de cualquier objeto, e impedir de esta forma los contagios y hacer más segura cualquier actividad que haya que realizar. Además, la automatización y digitalización de los datos, nos permitirá agilizar y manejar la información de forma más eficiente. Por estos motivos, este proyecto pretende realizar un control de presencia en determinados espacios, donde es necesario registrar las entradas, salidas y tiempos de permanencia de los trabajadores en empresas, universidades, institutos, …, de forma que cada persona, usuaria del sistema, pueda almacenar sus propias acciones de forma autónoma, rápida y segura, utilizando los dispositivos que automatizan el mismo.

Objetivos.

- El principal objetivo es crear un sistema capaz de leer la identidad de un usuario mediante un sistema de identificación por radiofrecuencia (RFID). Este sistema permitirá almacenar y recuperar datos que usan dispositivos tales como tarjetas, llaveros, …, dispositivos estos, de identificación automática.

- El sistema debe ser capaz de leer una etiqueta RDIF, identificar al usuario asociado a la misma y almacenar en una base de datos la fecha y hora del registro, así como el identificador del usuario que ha realizado la acción.

- El tratamiento de los datos deberá permitir saber qué usuarios y a qué hora han estado en una dependencia determinada.

- Crear un servicio REST con sus operaciones más importantes (GET, POST, PUT y DELETE.

- Permitir al administrador de la aplicación obtener los registros realizados en un año determinado.

Hipótesis de Trabajo.

Partimos de la necesidad durante la pandemia, de controlar la asistencia de los trabajadores de cualquier empresa, por medios informáticos.

Actualmente, en muchas empresas, el control de asistencia y el cumplimiento del horario, se refleja mediante firma en documentos en papel, siendo este el caso en la mayoría de los centros educativos, empresas, etc. Existe la necesidad de minimizar los riesgos de contagio, eliminando la manipulación de un mismo documento por muchas personas.

Espero que este proyecto cumpla los objetivos que me he marcado y cubrir las necesidades descritas anteriormente.

Aunque en la actualidad, existen un gran número de productos en el mercado destinados a este fin, con mi proyecto pretendo crear un producto asequible debido a su bajo coste y fácil de administrar y de utilizar por los usuarios.

Agilizar tareas cotidianas como es el control de asistencia mediante medios automáticos, me ha llevado a plantear este proyecto fin de Ciclo.

Material.

Tecnologías empleadas en el Backend.

- Firestore: Base de datos más reciente de Google. Pensada para el desarrollo de aplicaciones en móviles. Esta es en tiempo real, es decir, los datos se sincronizan tras la actualización de los mismo. Este hecho es el que me ha llevado a su elección ante el desarrollo de uno propio con las tecnologías vistas durante el curso o buscar y aprender una nueva.

- Python 3: Lenguaje elegido para el script del transceptor por su gran cantidad de librerías externas. Lenguaje interpretado filosofía hace hincapié en la legibilidad de su código.

  Las librerías que he usado son:

  - Firebase_admin: SDK propio de Google, permite la conexión con Firestore desde una aplicación en Python.

  - I2CPN532: Librería usada para la detección y el uso de un RFID PN532

Tecnologías empleadas en el Frontend.

- Ionic: SDK de código abierto usado para el desarrollo de aplicaciones hibridas. Desarrolla sobre Angular en su momento, a día de hoy también se puede usar sobre otras tecnologías como Real. En mi caso uso Ionic sobre Angular.

Hardware y software utilizado.

El material que he empleado es el siguiente:

- Raspberry PI (dispositivo facilitado por el centro):
  Placa base muy elemental y de bajo coste, que soporta multitud de componentes y con un sistema operativo de código abierto.
- Lector RFID PN532 (transceptor) (dispositivo facilitado por el centro):
  Dispositivo electrónico, cuya función es la recepción y transmisión de datos convirtiendo las ondas de radio en un formato legible por la Raspberry PI.

- Tarjetas y llaveros RFID (transpondedores) (dispositivos facilitados por el centro):
  Cada vez más usados en el mundo empresarial, las tarjetas y llaveros RFID son soportes que permiten leer, transmitir y capturar información utilizando la radiofrecuencia.
- Ordenador portátil. Características principales:
  - HP Envy TS 15
    - Procesador Intel Core i7-4700 2.40GHz (8 Cores)
    - Memoria RAM: 16GB
    - Tarjeta gradica Nvidia GeForce GT 740 2GB
    - HDD 1TB
    - Tarjeta de red Ralink RT3290 802.11bng
- Sistema operativo:
  - Ubuntu 18.04.
  - Raspbian GNU/Linux 10.
- Herramientas de desarrollo:
  - Visual Studio Code: IDE más usado durante el ciclo. Tiene la capacidad de añadir distintas extensiones para cada uno de los distintos lenguajes y tecnologías del mercado. En mi caso he usado extensiones para Typescript e Ionic.
  - Thonny: IDE usado en las Raspberry para el desarrollo en Python. Simple de usar al ser un editor de texto con salida de consola y posibilidad de depurar el código y ejecutarlo.
- Navegador Firefox y Google Chrome.
- Visme. Herramienta para la creación de diagramas.
- Microsoft Visio.

El tiempo empleado en el desarrollo del proyecto ha sido de 63 horas.

Método.

SEMANA	TAREAS REALIZADAS
12/10 al 18/10	• Planificación del proyecto. • Instalación de VNC para la conexión a escritorio remoto de la Raspberry PI (RPI). • Configuración del protocolo I2C. • Actualización de la RPI.
19/10 al 25/10	• Ver documentación para la conexión del transceptor a la RPI. • Instalación de librerías para la conexión. • Pruebas con script para la comprobación de la conexión y funcionamiento.
26/10 al 01/11	• Desarrollo de la primera versión del script: lectura de una tarjeta y visualización de su ID en bucle.

02/11 al 08/11	• Creación de la base de datos en Firestore.
	• Añadir la librería firebase_admin al script.
	• Creación de nuevos documentos en la colección:
	  o Mediante la creación de un JSON con los datos de un individuo, vacíos, excepto el ID de su llave para inicializar el documento, que será posteriormente completado desde la aplicación.
	• Desarrollo de la primera versión de la aplicación móvil.
	  o Creación de una página encargada de mostrarnos los ID de las llaves que han sido guardados en Firestore y sus datos no han sido completados.
	• Formulario para completar los datos de una persona. Para evitar el tener que desarrollar un Cliente-Servidor para el rellenado de ese documento.

09/11 al 15/11	• Creación de un formulario para poder completar la información de cada uno de los documentos generados a través del JSON. • Comprobar mediante un filtrado, que los datos de cada individuo están completados. • Mostrar en distintas páginas de la aplicación cada uno de los filtrados realizados. • Desarrollo de la documentación del proyecto.

16/11 al 22/11	• Modificación del script (ver 2.0) en el cual, mediante una petición GET se buscará en Firestore el documento cuya ID sea igual a la leída con el transceptor.
	  o Si el documento recibido es nulo, se añade, inicializando el documento igual que en su primera versión.
	  o En caso de reciba datos (no nulo), se almacenará en otra colección, la hora, el día y el ID de la llave.
	• Implementación en el script (ver 3.0):

SEMANA	TAREAS REALIZADAS
	○ Al recibir los datos de la persona, obtener el e-mail del usuario, para enviarlo a Firestore, añadiéndolo al JSON a través del cual, se envían las horas. La finalidad es tener una relación más simple para el administrador, en la que se relaciona e-mail con cada persona, en lugar de la relación antigua ID de la tarjeta con individuo. • Desarrollo de las nuevas funcionalidades de la aplicación. ○ Comprobación del almacenamiento correcto de los datos en los documentos. La forma en la que almacenaba la hora era incorrecta, ya que no almacenaba los "0" en los minutos, impidiendo una correcta lectura por parte de cualquier persona. ○ Creación de un filtrado para las personas por apellidos y para las horas por email de dicha persona • Desarrollo de la documentación del proyecto.
23/11 al 29/11	• Solución de error en el script (ver 3.1): ○ Cuando una persona pasaba la llave sin estar completada, el script generaba la hora y el día sin el e-mail del individuo. Este procedimiento, tenía que ser controlado para que se almacenaran de forma correcta los datos en Firestore. Para ello, el script controla todos los casos posibles: ▪ Que la llave no esté almacenada. ▪ Que esté almacenada, pero sus datos no hayan sido completados. ▪ Que se encuentre almacenada y completada. • Implementar una búsqueda en las horas desde el detalle de las personas. Esta funcionalidad la implemente para casos en los que el administrador no recuerde el email de la persona pero si sus apellidos, de esta forma será más sencilla la búsqueda de sus fichas. • Desarrollo de la documentación del proyecto.
30/11 al 06/12	• Finalización del script (ver 4.0), cambiando los JSON por objetos. • Mejora de las vistas de la lista de horas, añadiendo un ActionSheet en el botón de borrar. • Centrar los botones de todas las vistas. • Desarrollo de la documentación del proyecto.
07/12 al 13/12	• Desarrollo de la documentación del proyecto. • Desarrollo de un histórico. Para ello usando una pagina en Ionic el administrador introduce el año que buscar. Una vez elegido se cargara una lista con todos los registros de hora y fecha que se encuentren en dicho histórico. • El histórico se rellenará al borrar la base de datos. Antes de borrarla se guardan en dicho histórico y tras ello, procede a borrarlo todo. Esto se debe a no repetir datos, de esta manera los mas recientes estarán en el apartado "Fichas" de la app, mientras que el resto se encontraran en "Histórico
14/12 al 18/12	• Desarrollo de la documentación del proyecto. • Ultimar los detalles de la aplicación. • Solucionado un error en el histórico ○ Para crear o guardar en una colección las horas, miraba el primer elemento de la colección y creaba, acorde a su año dicha base de

SEMANA	TAREAS REALIZADAS
	datos, en caso de que esta no estuviese creada, y almacenaba los documentos. Al acercarse fin de año me he dado cuenta de que, si la colección “Horas” contuviese como primer elemento un documento con una fecha, por ejemplo, del 2020. Si se creaba otro registro durante el 2021 y se borraba, este último también acabaría en el histórico del 2020. La solución correcta pasa por mirar cada año de los distintos registros.

Arquitectura.

Este proyecto está basado en una solución de tres capas a dos niveles. Por un lado, tenemos la presentación, que sería la aplicación de administración. Este nivel está conectado con la capa de lógica, que en mi caso corresponde con Firestore.

El siguiente nivel corresponde con la RPI y Firestore. La RPI debe realizar una petición GET para buscar en Firestore a la persona y ver si se encuentra almacenada y con sus datos completados. En caso de que estos lo estén, se ocupará de hacer una petición POST con la fecha, hora y email del usuario.

Modelo de carpetas de la aplicación.

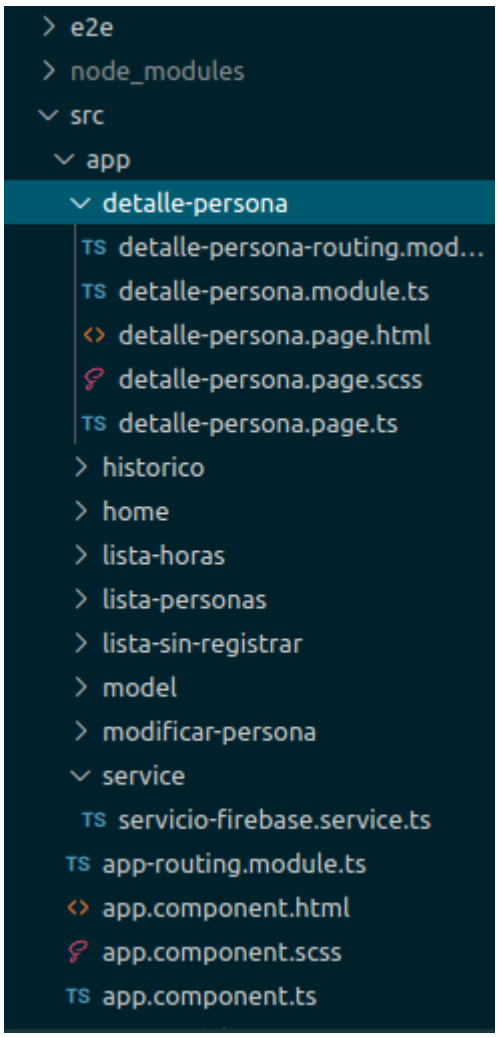

Ilustración 1. Carpetas de la aplicación.

Ionic cuenta con su cliente para añadir elementos nuevos a una aplicación. Cada página se generará dentro de una carpeta con el nombre que le demos al componente.

Dentro de esta carpeta encontraremos los siguientes ficheros:

- Routing-module: Fichero que nos permitirá configurar las rutas a las que podremos acceder desde esa vista
- Module: Un módulo en angular es un contenedor para almacenar componentes, páginas o servicios
- Page.html: Vista del componente en html. Aquí codificaremos la vista que el usuario verá.
- SCSS: Fichero de estilos de Ionic.
- Page.ts: Archivo donde se encuentra la lógica del componente

Este proyecto cuenta con siete componentes, cada uno detallando en su nombre la vista que el administrador verá, es decir, el componente "detalle-persona" se encargará de mostrar el detalle de esta, en "Lista-sin-registrar", en el menú de la aplicación, nombrado como "Llaves sin almacenar", (*ver manual de referencia*), mostrará los IDs de las llaves cuyos datos no estén completos. Esto se obtiene mediante un filtrado de la colección "Persona" de Firestore.

En el directorio service se encuentra el código del servicio encargado de recibir y enviar la información a Firestore.

También cuenta la carpeta "Model", en la cual se encuentran los modelos de los objetos que usaremos (Persona y Hora).

Base de Datos Firestore.

En Firestore guardaremos las colecciones de las personas, las horas de registro y el histórico de cada año.

- *Persona*: El ID del documento será el ID de la llave electrónica de esa persona. En primer lugar, será almacenado completando el campo "IDLlave" y desde la aplicación se podrán modificar y actualizar los datos del documento. (*véase Ilustración 2*).

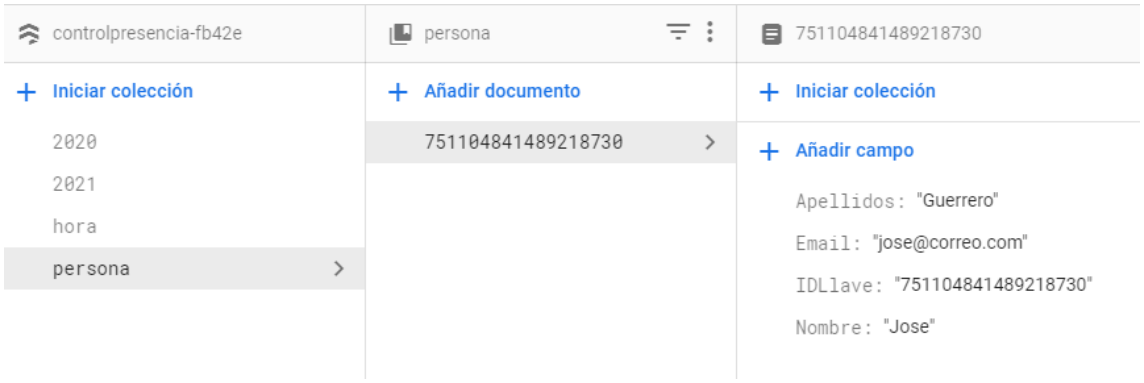

Ilustración 2. Colección Persona.

- *Hora*: Aquí se almacenará cada registro de la hora, fecha y el correo electrónico de la persona que pase la llave. Cabe añadir que en el caso de que los datos de la persona no estén rellenos en la tabla "Persona", la RPI no mandará la petición POST a Firestore. Desde la aplicación se podrán almacenar estos documentos en el histórico del año (ver manual de usuario). Su relación con la tabla "Persona" viene dado por el email.

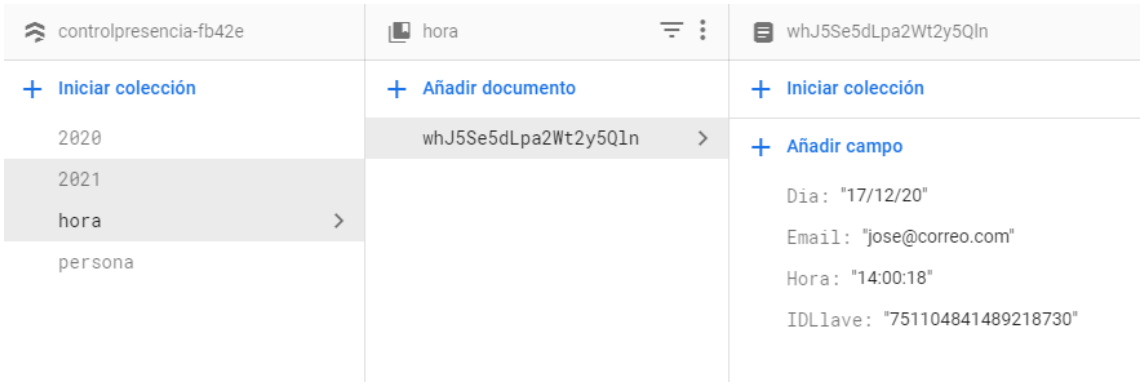

Ilustración 3. Colección Hora.

- *Colecciones histórico*: En esta colección se almacenarán los documentos eliminados mediante las posibles opciones de la aplicación (*véase Ilustración 4*).

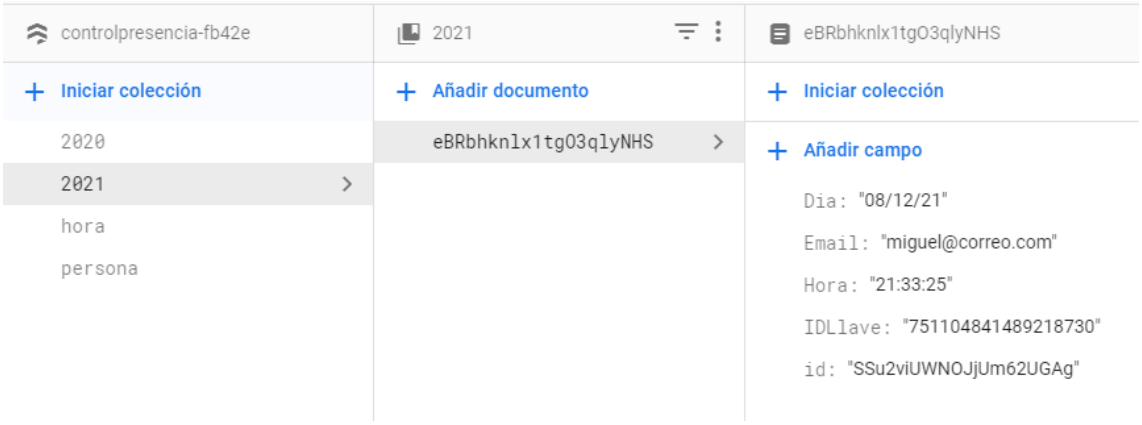

Ilustración 4. Colección Histórico 2021.

Modelo de Objetos del script.

Aquí muestro un ejemplo de uno de los modelos usados para la programación orientada a objetos que se usa en la RPI.

La función "__init__" actúa como constructor de este modelo. En este caso solo le pasamos el email y el id de la llave para construirlo. Su método "to_dict" convierte a JSON el objeto, esto es necesario ya que la librería firebase_admin trabaja de esta forma. La hora será obtenida del sistema en ese momento ya que al estar obligada la RPI a encontrarse constantemente en línea, la hora siempre será la correcta. En el momento de crear el JSON a partir del objeto, su campo Hora y Dia se verá rellenado con los datos del sistema.

```python
import time
class hora:
    hora = ''
    dia = ''
    email = ''
    IDLlave = ''
    def __init__(self, email, IDLlave):
        self.email = email
        self.IDLlave = IDLlave

    def to_dict(self):
         dest = {
         u'Hora': time.strftime("%H:%M:%S"),
         u'Dia' : time.strftime("%d/%m/%y"),
         u'Email': self.email,
         u'IDLlave': self.IDLlave
         }
         if self.email:
             dest[u'Email'] = self.email
         if self.IDLlave:
            dest[u'IDLlave'] = self.IDLlave
         return dest
```

Ilustración 5. Clase Hora.py

Tabla API REST.

ACCIÓN	URL	MÉTODO HTTP	BODY
-	/	GET	
Obtener lista personas	/listaCompleta	GET	
Obtener lista de Llaves sin almacenar	/listaLlaves	GET	
Listar registro de horas	/horas	GET	
Ver el histórico de un año	/histórico	GET	
Obtener detalle de Persona	/detallePersona/:id	GET	
Buscar registros de horas de persona especifica	/listaCompleta/:id	GET	
Completar/Modificar datos de persona	/modificar/:id	PUT	Los datos actualizados o completados
Borrar persona	/listaPersonas	DELETE	Persona a borrar
Borrar todas las horas	/horas	DELETE	
Borrar hora única	/hora	DELETE	La hora a borrar
Añadir al histórico todas las horas	/hora	POST	Todas las horas
Añadir horas de una persona	/listaPersonas	POST	Las horas de esa persona en concreto

Planificación.

Ilustración 6. Planificación del proyecto.

Diseño.

Diagrama de casos de uso.

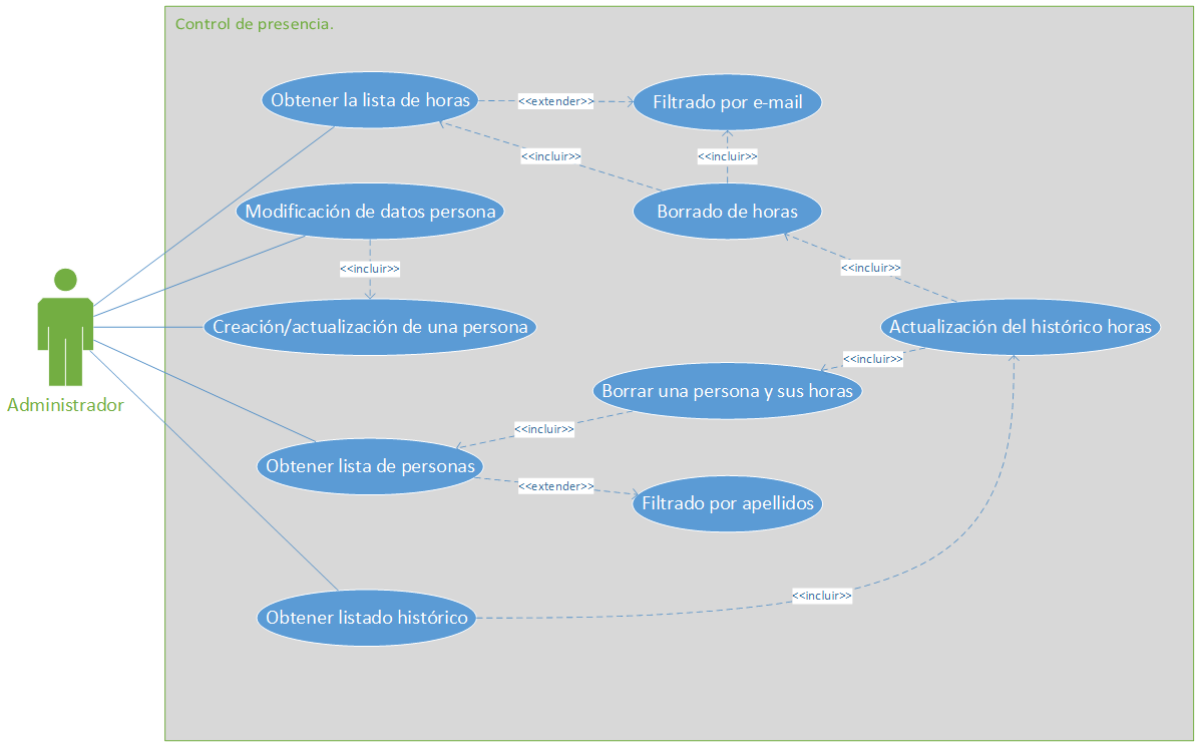

Ilustración 7. Diagrama de uso.

Resultados.

He confeccionado un script de Python que se encarga de leer con un transceptor RFID, una llave transpondedora, almacenando ID de la llave en un documento de Firestore (Persona) para su posterior tratamiento. Este mismo script, se encarga de almacenar la fecha, hora y e-mail del usuario en un documento en la colección Horas.

La aplicación en Ionic permite obtener la lista de las personas, modificar sus datos, y buscar sus registros en la colección Horas. También nos da la posibilidad de modificar los datos de usuario, cuando este pasa por primera vez la llave, permite el borrado de la colección Horas, para evitar saturación de datos, iniciar un nuevo periodo, etc.… y, por último, también da acceso a un histórico de los registros de Horas, en el cual se almacenan los documentos antes de ser borrados de Horas.

Conclusiones.

Valoración de tecnologías usadas.

Como se ha visto en el documento, las tecnologías que he usado son Ionic para el Frontend, Python para el desarrollo del script encargado de leer y enviar los datos a Firestore y, obviamente, esta última.

Para el Frontend en un principio pensé usar Xamarin.Forms, tecnología usada durante la primera parte de mi FCT. Es un framework de Microsoft pensado para realizar aplicaciones tanto en IOS, Android como para escritorio de Window con .NET.

Mi decisión final fue quedarme con Ionic, estudiado en el módulo de Desarrollo de Interfaces. Aunque durante el curso vi distintas tecnologías para el desarrollo de un frontend, como React, Android nativo o Angular, preferí hacerlo con Ionic ya que es un framework sencillo uy que cuenta dentro de su documentación con todo lo necesario para el desarrollo. No puedo negar que el diseño sigue siendo mi “asignatura pendiente”, pero gracias a esta facilidad, he podido realizar una app de administración, en la cual prevalece la funcionalidad al diseño a través de un desarrollo rápido y sencillo.

La parte de la conexión del RFID con Firestore la he realizado usando Python. En este lenguaje he encontrado las librerías necesarias tanto para su conexión con Firestore como para la conexión del RFID por protocolo I2C. Python cuenta con una amplia gama de librerías externas, de terceros o de empresas, que nos permite desarrollar casi cualquier aplicación con este lenguaje.

Durante el curso vimos Python en el módulo de Acceso a Datos. Mi primer acercamiento he de decir que no me agrado. La tabulación para diferenciar los distintos ámbitos después de estar aprendiendo durante más tiempo C++, Java, Kotlin y Javascipt/Typescript, lenguajes que abren y cierran ámbitos con llaves, hizo que chocase un poco con este lenguaje.

Tras estos meses, he terminado por cogerle cierto gusto. Permite, en menos líneas de código, realizar una mayor cantidad de acciones que otros lenguajes y aludiendo a la primera parte, su gran cantidad de librerías ayuda mucho a los desarrolladores, sobre todo a estudiantes y a juniors.

Sobre Firestore he de añadir un comentario, y es que descubrí demasiado tarde las ayudas que Google tiene a disposición de los alumnos. Podría haber sacado más provecho a esta base de datos en tiempo real. El hecho de que la haya elegido frente a una base de datos local y un backend ha sido su "tiempo real". Gracias a esto el administrador puede ver al instante a cualquier persona que haya registrado su ficha. Además, la gran ayuda que ofrece al no tener que programar un backend nos permite a día de hoy a los programadores dedicar más tiempo a otras labores. Esta tecnología de Google la utilizamos en el Módulo de Desarrollo de Interfaces.

No puedo terminar este apartado sin comentar un último detalle. Utilizar algunos conocimientos de electrónica al conectar el RFID con la RPI, sabiendo interpretar unos diagramas que ni amigos ni familiares con conocimientos en el campo entendían y comprobar que yo sí, modificar la tabla de verdad la cual se encuentra dentro del propio lector para cambiar de protocolo, instalar las librerías necesarias y activar el protocolo I2C, me ha servido para recordar una de las asignaturas que más me gustó cuando estudiaba el Grado de Ingeniería Informática.

Investigación.

Lo primero que investigue fue como conectarme mediante escritorio remoto a la RPI. El trabajar con el mismo ordenador cambiando de ventana sería una ventaja frente al tenerla conectada a un monitor con un teclado y un ratón. Tras ello encontré VNC Viewer y Server, instalando el Server en la RPI y en mi ordenador el Viewer, fijando la ip manualmente me permitió realizar este tipo de conexión.

Esta ha sido la primera vez que he trabajado con una RPI, lo cual me ha llevado a investigar cómo conectar los circuitos auxiliares en los conectores GPIO, así como los protocolos para conectar dichos circuitos (I2C o SPI). Una vez realizada, lo siguiente fue buscar que pines

corresponden al voltaje, la entrada de tierra, el reloj,…Tras ello, viendo diversos diagramas, pude fácilmente conectar cada pin del RFID532 en el lugar que corresponde de la RPI.

Lo poco que había visto de Python fue para temas totalmente distintos. Esto me llevo a buscar cómo usar la librería firebase_admin, crear un servicio REST con Firestore y como obtener la hora correctamente del sistema, ya que, al principio, la forma en la que la obtenía omitía los ceros e impedía saber si eran las 18:01 o las 18:10 por ejemplo.

Para Ionic las investigaciones fueron para detalles puntuales como centrar elementos en pantalla, el filtrado de colecciones y la expresión regular para comprobar que un email esté correctamente.

Aspectos a mejorar.

Tras realizar el proyecto, me he dado cuenta de tres aspectos a mejorar del mismo y en mi persona:

El primero es el más obvio, el uso de GIT. Tarde demasiado en subirlo, lo que impide ver la evolución del proyecto y la trayectoria de este desde que comencé hasta el día de entrega. Además, durante la trayectoria de mi FCT, mi primer tutor me enseñó el uso de las ramas y como realizar los "merges" entre ellas. Hubiese sido un punto a favor haber hecho uso de este aprendizaje tanto para el proyecto como para mi iniciación en el mundo de la programación como Desarrollador Junior.

El segundo es la organización en carpetas de la parte de la RPI. La estructura la creé yo mismo sin uso de herramientas externas y debería de haber sido gestionada de otra manera, separando tanto la librería de encargada de la lectura de las llaves mediante el RFID como la parte del modelo.

Añadir también que al no poder obtener un monitor LCD, los mensajes del script son mostrados en la consola propia del sistema operativo. Esto nos lleva a que no existe la comunicación entre el usuario que pasa su llave y la RPI.

Comentar el hecho de no haber podido ejecutar este código al encender la RPI. Demasiados intentos fallidos entre que no reconocía las librerías de Python instaladas o no encontraba algunos ficheros necesarios para su ejecución.

Por último, pero no menos importante, seria hablar del propio diseño de la aplicación. Diseñar sigue siendo mi talón de Aquiles. Siempre ha sido la parte del desarrollo que más me ha costado. Aunque la aplicación sea de administración, donde la funcionalidad debe prevalecer frente al diseño, esto no implica que debiera ser una de las mejoras que debería tener este proyecto.

Bibliografía.

- Firebase_Admin:
  - Administración de Firebase en Google.
- Guía rápida de servicio REST con Python y vídeo de ejemplo:
  - Firebase en Google.
  - Canal Youtube Firebase.
- Obtener hora de Python:
  - Pythondiario.
- Durante el documento he hablado de un punto en el que no se guardaba bien la hora actual, ya que omitía ceros. Para realizar esto correctamente lo hice como sale en la siguiente página.
  - Obtención de hora de forma correcta.
- Preguntas para solucionar detalles concretos.
  - Stackoverflow.
- Librería para RFID:
  - Github.
- Conexión de la RFID a la RPI:
  - Canal de Youtube Canal HQ.
- Conexión del PC mediante VNC a la RPI:
  - Canal de Youtube JadsaTV.
- Documentación de IONIC (seleccionando Angular):
  - Ionicframework.

Anexos.

Manual de usuario.

Esta App se basa en una aplicación de administración de los registros realizados por un usuario cuando, haciendo uso de su llave electrónica, se registra a la entrada de un lugar determinado. Los registros pueden ser efectivos en cualquier lugar y pueden variar según su uso: empresas, universidades, institutos de educación secundaria, ...

La aplicación consta de las siguientes vistas disponibles desde el menú:

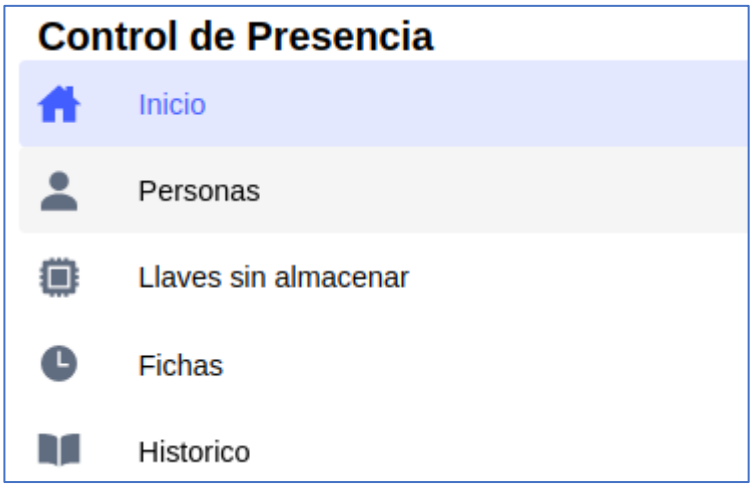

Ilustración 8. Menú de la aplicación.

Personas.

La primera pestaña "Personas", nos lleva a la vista donde se muestran la lista de las personas registradas en la BBDD (*véase Ilustración 8*). Seleccionando cualquier elemento de la lista abriremos el detalle de esa persona, esto es, el nombre, apellidos, el identificador de su llave electrónica (ID) y el correo electrónico. Dentro de este mismo cuadro de diálogo nos da la posibilidad de eliminar una persona, desplazando hacia la izquierda el elemento. Cuando una persona es borrada de la BBDD, automáticamente, la aplicación añadirá al histórico los registros correspondientes a la misma, de forma que podremos saber quien estuvo en todo momento en un lugar determinado.

También cuenta con un buscador, que nos permitirá filtrar las búsquedas por los apellidos de una persona y así, agilizar y hacer la exploración de la BBDD más simple y sencilla.

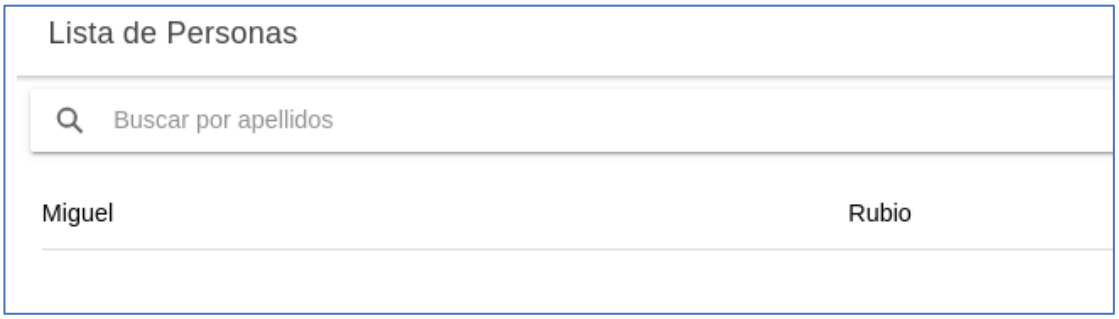

Ilustración 9. Lista de personas.

Pasando al detalle podremos ver el nombre, los apellidos, el correo y el ID de la llave de esa persona (*Ilustración 10*). También tenemos dos botones centrados en el detalle: buscar y editar.

Ilustración 10. Detalle de una persona.

Buscar, nos llevará a la pestaña de "Fichas" (Consultar su apartado correspondiente donde se explicará esta función de la aplicación) mostrando en ella solo los registros de las horas que haya tenido esta persona y de nadie más.

El botón Editar nos permitirá cambiar los datos de esa persona en caso de que haya habido algún error al ser creados (*véase Ilustración 10*).

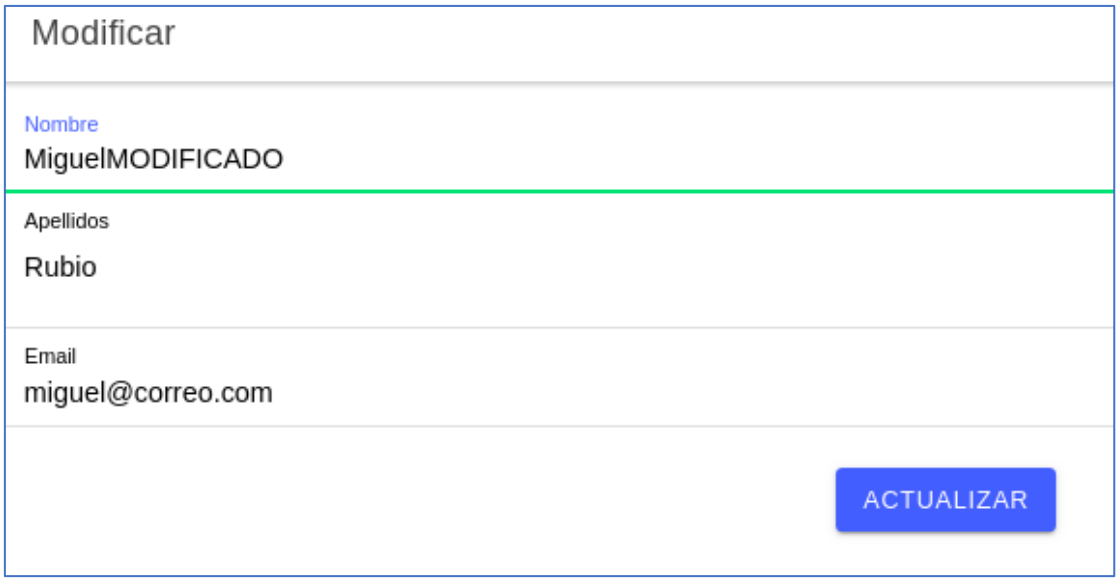

Ilustración 11. Modificación Persona.

Llaves sin Almacenar.

Esta interfaz, nos permite visualizar todas las llaves que hayan sido registradas por primera vez (*Ilustración 12*). Una vez que se muestren, es recomendable, y casi obligatorio, completar los datos al instante, para que, de esta manera, no haya confusiones al registrar cada llave electrónica (ID de la misma) con los datos del usuario o persona a quien pertenece ésta.

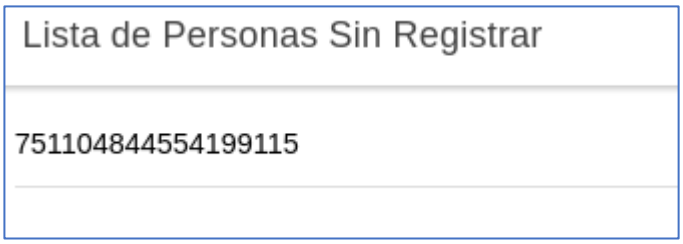

Ilustración 12. Laves sin registrar.

Al seleccionar el ID de una llave, la aplicación abrirá la siguiente ventana (*Ilustración 13*), en la cual debemos rellenar los datos pertenecientes a la persona: Nombre, Apellido y correo electrónico.

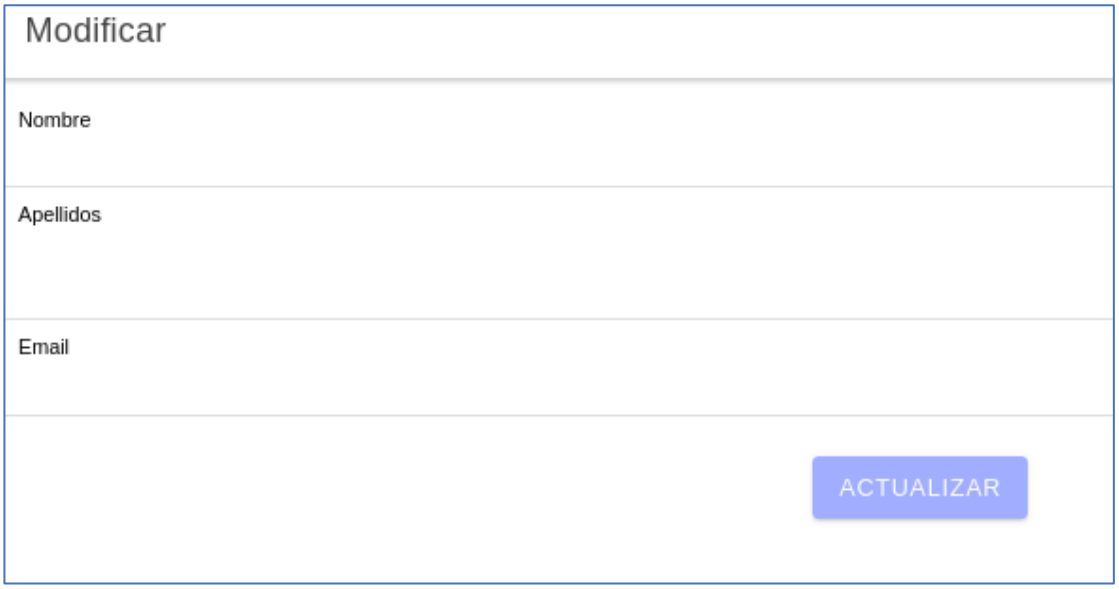

Ilustración 13. Modificación de datos de una persona.

Fichas.

En esta interfaz (*Ilustración 14*), podremos visualizar la fecha y hora, asociadas al correo electrónico de la persona, en las que esta, ha estado en una estancia. Hacer la salvedad de que en esta lista aparecerán todos los registros que no hayan sido pasados al histórico. En la parte superior podremos filtrar por email para buscar los registros de una persona en concreto. Si queremos volver a mostrar todo, solo debemos borrar lo escrito en el en la barra de búsqueda.

Otra forma de acceder a esta pestaña seria mediante el botón "Buscar" dentro del detalle de una persona (*véase Ilustración 10*). Al pulsar ese botón la aplicación mostrara las fichas que haya realizado la persona. Se puede también reiniciar desde aquí para mostrar todo, escribiendo cualquier cosa en la barra de búsqueda y borrándolo lo que reiniciará el filtrado, volviendo a mostrar todo.

Ilustración 14. Lista de horas.

Si pulsa el botón Borrar BD (*véase Ilustración 14*), los datos, antes de ser eliminados, serán almacenados en el histórico. Por recomendación personal, le diría que cada semana o quincena realice esta acción. Esto evitará que se acumulen los registros en la base de datos y, por consiguiente, sean más fáciles de visualizar e interpretar. Como ya he comentado, estos registros, cuando son borrados, se almacenan en un histórico (*véase Ilustración 15*) y, por lo tanto, siguen siendo accesibles para su consulta. Al ser una mera recomendación, lo dejo a su elección.

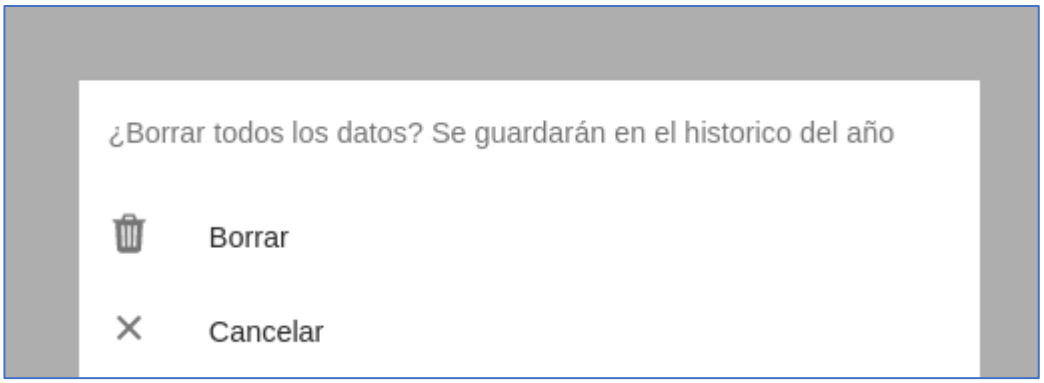

Ilustración 15. Borrado de horas y paso al histórico.

Los registros también pueden ser borrados uno por uno (*véase Ilustración 16*). Esto debe hacerse al encontrarse con un error a la hora de pasar la tarjeta como, por ejemplo, pasar dos veces la tarjera o que una persona ya hubiera fichado con anterioridad. Es importante hacer saber que esta acción no guardará los registros borrados en el histórico del año, por lo que la información se perderá de forma permanente.

Ilustración 16. Eliminación de un registro de hora.

Histórico.

Esta interfaz nos permitirá buscar los registros por años de todas las fichas de las personas. Para ello, basta con introducir el año en el buscador. El año debe estar completo, por ejemplo, 2020 o 2021 (*véase Ilustración 17*).

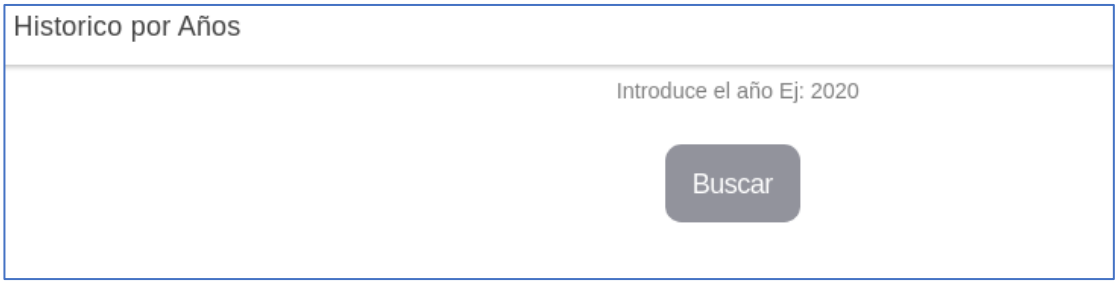

Ilustración 17. Histórico de horas. Búsqueda.

En la siguiente imagen (*Ilustración 18*), podemos ver un ejemplo de visualización de los registros almacenados en el histórico, correspondientes al año 2020.

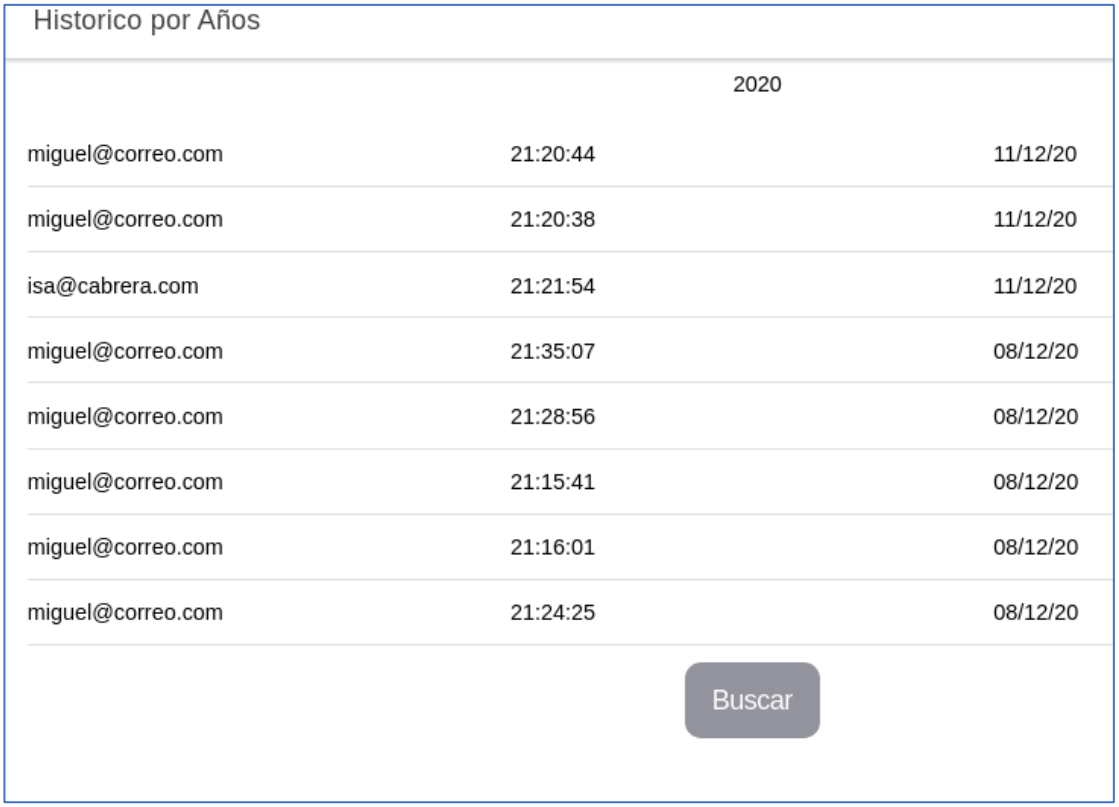

Ilustración 18. Histórico de horas. Listado.

Manual de uso del lector transceptor.

El usuario debe pasar una llave transpondedora por el transceptor. Existen tres posibles casos una vez sea pasada:

A. La llave es la primera vez que es pasada (*Ilustración 19*): En este caso el ID se guardará en la base de datos y será el administrador el que se encargue de forma inmediata de completar los datos de la persona. Recuerde hacer esto uno por uno. Para ello desde la app vaya al apartado de "Llaves sin almacenar" y complete el formulario con los datos del usuario.

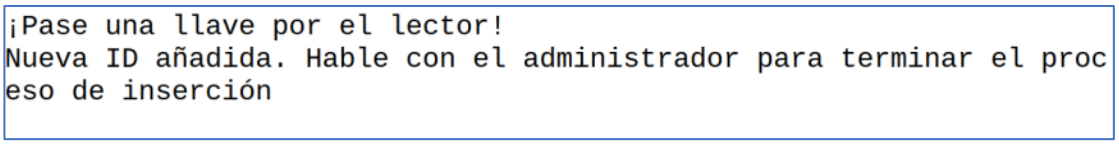

Ilustración 19. Paso de una llave por primera vez.

B. La llave es pasada pero los datos no han sido completados (*Ilustración 20*): Para poder almacenar la hora de entrada de una persona, los datos de esta deben primero estar completados.

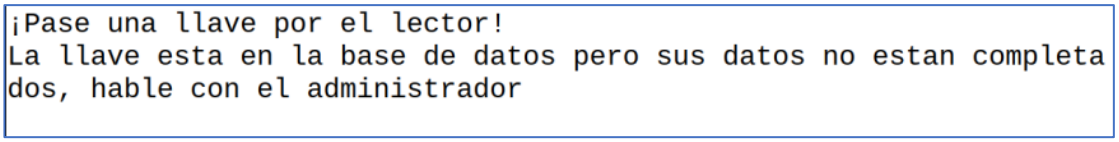

Ilustración 20. Datos de la persona sin registrar.

C. La llave es pasada con el registro terminado (*Ilustración 21*): Se almacenará la hora, la fecha y el email de la persona. Este se podrá consultar en el apartado Fichas de la aplicación o en el histórico.

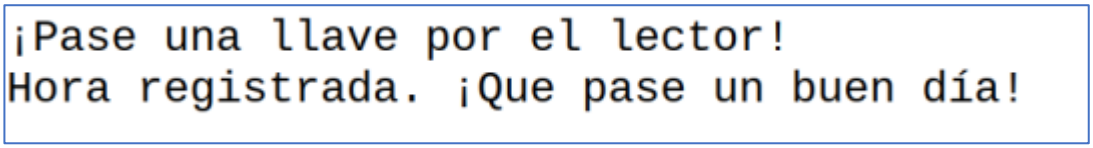

Ilustración 21. Hora registrada.

Agradecimientos.

Desde un principio tenía pensado escribir este apartado, pero las dudas me hicieron replanteármelo. Tras hablar con cierta persona y saber que él sí lo hizo al final me decidí lanzarme con esto y hablar de estos dos años que he pasado bajo su tutela.

Llegué a este ciclo sin un camino que seguir, abandonando un grado que por temas personales fui incapaz de completar y por este motivo, me sentía sin rumbo y sin motivación para estudiar o trabajar.

Me encuentro en estos momentos a punto de terminar esta etapa, faltando solo este proyecto y puedo decir, después de estos dos cursos y medio, que he encontrado mi camino y mi vocación profesional.

Quiero daros las gracias desde lo más profundo de mí, por todo el apoyo que he recibido desde que llegué, por haber sacado a un chaval adelante que perdió la confianza en sí mismo varios años atrás.

A Elvira, Carmen, Santiago y Guadalupe en primero. A Manolo y Juan Gualberto en ambos cursos y a Jacinto, Luis, Cándido y Juan José en segundo. Mil gracias por todo, desde el primer día hasta el último de mi paso por el centro y para siempre. Gracias por entenderme, por apoyarme y por todo lo que he aprendido de vosotros. Sois grandísimos profesores y me siento afortunado de haber sido vuestro alumno.

Tampoco puedo olvidarme de mi familia, de mi madre, mi padre y mi hermana, su apoyo incondicional y más en este último tramo de mis estudios. Sin ellos, este último curso hubiese sido imposible terminarlo.

Por último, pero no menos importante, quiero agradecer a mis compañeros por estos dos años magníficos cargados de risas. A Juan Carlos, David, Daniel, Alejandra, José Luis, Rubén, Juan Pedro y así cada uno de los compañeros y amigos que he hecho.

Pero en especial a José, por su apoyo incondicional y porque que sin él no hubiese conocido a dos personas imprescindibles hoy en mi vida, Dulce e Isabel. Mil gracias a los tres.

No quería terminar este documento sin estas palabras: GRACIAS POR TODO.

Índice de ilustraciones.