Sistema de Gestión de Datos de Pacientes Diabéticos

Data management system for diabetic patients

Trabajo de fin de grado

Ingeniería Informática

Cristóbal Ramos Laina

Jaime Viejo Martínez

Dirigido por

José Ignacio Hidalgo Pérez

Facultad de informática Universidad Complutense de Madrid

Febrero 2020

Resumen:

La diabetes es una enfermedad crónica que aparece cuando el páncreas no produce insulina suficiente o cuando el organismo no utiliza eficazmente la insulina que produce. La insulina es una hormona que regula el azúcar en la sangre. Los efectos de la diabetes mal controlada se hacen visibles tanto a largo como a corto plazo. Mantener niveles de glucosa elevados (hiperglucemia) durante largos periodos de tiempo puede dañar o afectar gravemente a varios órganos y sistemas, especialmente a los nervios y los vasos sanguíneos. Por otra parte, situaciones en las que los valores de glucosa estén muy bajos (hipoglucemia), pueden desencadenar episodios peligrosos que conduzcan incluso a la muerte.

Para mantener el nivel de glucosa controlado, el paciente diabético necesita en algunos casos inyectarse insulina y saber cuál es la cantidad de insulina adecuada no es una tarea fácil. El grupo ABSYS (Adaptive and Bioinspired sytems) tiene un proyecto de investigación en marcha entre cuyos objetivos está generar algoritmos predictivos para calcular la cantidad de insulina a inyectar en los pacientes de una manera rápida, fiable y segura. Estos algoritmos toman como entrada para su entrenamiento una gran cantidad de datos, que se han recogido usando un conjunto de dispositivos de medición.

La motivación de este trabajo de fin de grado ha sido la necesidad de gestionar esta gran cantidad de datos de los pacientes. El objetivo principal ha sido crear una aplicación web que puedan usar todo tipo de usuarios, desde pacientes sin experiencia informática hasta investigadores con un buen conocimiento del entorno, pasando por el profesional de la salud.

La aplicación web desarrollada permite gestionar varios tipos de datos exportados de los dispositivos de medición utilizados por los pacientes. Entre los dispositivos soportados por la misma están pulseras de actividad, medidores continuos de glucosa y sistemas de inyección subcutánea de insulina, también conocidas como bombas de insulina. El sistema desarrollado permite obtener datos preparados para su utilización en distintos entornos de modelado y predicción, visualizar los mismos en forma de tablas y estudiar de forma visual las tendencias de las medidas almacenadas, tanto de manera individual como de forma conjunta.

Palabras clave: Diabetes, Insulina, Glucosa, pandas, DataFrame, phpMyAdmin, Django, Python, Médico, Paciente

Abstract:

Diabetes is a chronical disease that appears when the pancreas does not produce enough insulin or when the organism does not use properly the insulin it produces. Insulin is a hormone that controls the sugar in our blood. The effects of poorly controlled diabetes become visible at both long and short term. Maintaining high levels of glucose (hyperglycemia) during long periods of time can damage or affect organs and systems, specially nerves and blood vessels. On the other hand, situations where glucose values are very low (hypoglycemia) can evolve in dangerous episodes that can even lead to death.

In order to maintain controlled levels of glucose, the diabetic patient sometimes needs to inject insulin and knowing the right quantity of insulin is not an easy task. ABSYS group (Adaptive and Bioinspired systems) has an ongoing research project whose objectives are generating predictive algorithms to calculate the quantity of insulin to inject in a fast, reliable and safe way. Those algorithms are trained with large amounts of data that has been collected using a set of measuring devices.

The motivation of this project has been the need to manage the large amount of data that these patients generate. The main objective was to create a web application that can be used by any kind of user, from patients with very little computer knowledge to researchers with a good strong background, through health experts.

The web application that has been developed allows users to manage various types of data exported from the devices that the patients use. Among the supported devices are activity bracelets, continuous glucose monitoring systems and subcutaneous insulin injection systems, also known as insulin pumps. The system developed permits to obtain data ready to be used in different modeling and predictive environments, visualize them in tables and study graphically the tendencies of the measurements stored.

Key words: Diabetes, Insulin, Glucose, pandas, DataFrame, phpMyAdmin, Django, Python, Doctor, Patient

1. Introducción

La diabetes es una enfermedad crónica que afecta cada vez a más personas [1] en los países desarrollados. La enfermedad es una compleja afección producida o bien por un defecto en la producción de insulina o bien por la mala acción de esta. La insulina es una hormona producida por el sistema endocrino; esta es necesaria para regular la absorción de glucosa de algunas células importantes del cuerpo humano. La mayoría de los pacientes de diabetes se pueden clasificar en dos tipos: Tipo 1 Diabetes Mellitus (T1DM) y Tipo 2 (T2DM) Diabetes Mellitus [1]. La gente que padece T1DM sufre de una enfermedad autoinmune que ataca a las células del páncreas encargadas de la producción de insulina. Dicho de otra manera, el páncreas de la gente con T1DM no produce insulina suficiente para procesar el azúcar en sangre. Por otro lado, los pacientes con T2DM sí que producen insulina, pero su cuerpo genera una resistencia a ella por lo que tiene muy poco o ningún efecto. En ambos casos el efecto es una subida de los niveles de glucosa en el torrente sanguíneo. Mantener altos niveles de glucemia durante largos periodos de tiempo puede causar otras enfermedades crónicas como pueden ser la ceguera o lesiones cerebrales o de riñón. Las ya mencionadas consecuencias a largo plazo pueden ser fatales, pero también hay riesgos a corto plazo debido a niveles bajos de glucosa: la hipoglucemia ocurre cuando la glucosa baja de un límite, esto puede causar la inconsciencia y puede que hasta la muerte. Por estas razones los pacientes diabéticos tienen que mantener un control muy exhaustivo de los niveles de glucosa, intentando siempre mantener unos niveles razonables, similares a los de una persona sana. Normalmente el rango es [70-180]*mg/dl* [2]. Cuando una persona que padece diabetes va a comer tiene que estimar las unidades que se va a inyectar para que tras la comida la glucosa se mantenga dentro de los límites. Esta estimación la tienen que hacer teniendo en cuenta muchos factores, pero principalmente, los pacientes conocen su nivel de glucosa en el momento de la ingesta y hacen una estimación de la cantidad de alimentos que van a comer (normalmente se mide en unidades de carbohidratos) [1]. Para los pacientes de T1DM se tiene que tener en cuenta el pronóstico de valores futuros de glucosa basándose en los valores de alimentos ingeridos, inyección de insulina, y/o su glucagón. Por lo tanto, como se puede deducir, este proceso requiere de muchas estimaciones y no es un proceso claramente definido en las variables involucradas.

Por todo lo citado anteriormente, se puede asegurar que la diabetes es una enfermedad grave que según la Organización Mundial de la Salud [3] más de 420 millones de personas sufrían esta enfermedad en 2014. Para poder realizar una vida lo más normal posible, se necesita controlarla y para ello hay que tener muchos factores en cuenta.

Hoy en día con ayuda de la tecnología es más sencilla la recopilación de los datos y la toma de decisiones. Para poder utilizar los datos de manera efectiva, estos tienen que estar bien organizados. En la actualidad la mayoría de los pacientes diabéticos utilizan una serie de dispositivos que registran toda su actividad y distintas variables físicas. En concreto es muy común utilizar pulseras de actividad y en menor medida bombas de inyección y medidores continuos de glucosa.

Este proyecto trata de poner la tecnología al servicio del paciente diabético. Para ello hemos desarrollado una aplicación web que permite gestionar todos los datos que recogen los dispositivos mencionados anteriormente.

Esta aplicación ayuda a los usuarios de varias formas. Por un lado, los pacientes podrán llevar de forma sencilla un seguimiento de todos los datos recopilados. A su vez, los médicos

podrán acceder a los datos sin que los pacientes tengan que ir a consulta favoreciendo asi un control más exhaustivo de los mismos. Por último, los investigadores podrán acceder a los datos de los usuarios para generar modelos de predicción o ver qué margen de error generan los dispositivos de medición.

Por todo lo anterior y el interés que tenemos ambos en el desarrollo de aplicaciones y el tratamiento de datos decidimos embarcarnos en este proyecto. Tras un mes de investigación y 5 meses de desarrollo hemos conseguido obtener una versión funcional de la aplicación, válida para que médicos, investigadores y pacientes la usen. Hasta el momento podemos gestionar diez tipos de archivos diferentes que provienen de 5 dispositivos de medición.

Resulta muy gratificante sentir que aplicar nuestros conocimientos en un proyecto relacionado con la medicina pueda ayudar a médicos a trabajar de una manera más útil para el beneficio de los pacientes.

1.1. Introduction

Diabetes is a chronic disease that affects more and more people [1] in developed countries. The disease is a complex condition produced either by a defect in the production of insulin or by the bad action of this. Insulin is a hormone produced by the endocrine system; This is necessary to regulate the glucose uptake of some important cells of the human body. Most diabetes patients can be classified into two types: Type 1 Diabetes Mellitus (T1DM) and Type 2 (T2DM) Diabetes Mellitus [1]. People with T1DM suffer from an autoimmune disease that attacks pancreas's cells, those are responsible of insulin production. In other words, the pancreas of people with T1DM does not produce enough insulin to process sugar in the bloodstream. On the other hand, patients with T2DM do produce insulin, but their body generates resistance to it, so it has little or no effect. In both cases the effect is a rise of glucose levels. Maintaining high glucose levels for long periods of time can cause other chronic diseases such as blindness or brain or kidney damage. The long-term consequences can be fatal, but there are also short-term risks due to low glucose levels: hypoglycaemia occurs when glucose drops below a limit, this can cause unconsciousness and even death. For these reasons diabetic patients must maintain a very thorough control of glucose levels, always trying to maintain reasonable levels, similar to those of a healthy person. Normally the range is [70-180] mg / dl [2]. When a person suffering from diabetes is going to eat, they have to estimate the units to be injected so that after the meal the glucose stays within the limits. This estimate has to be made taking into account many factors, but mainly, patients know their glucose level at the time of intake and make an estimate of the amount of food they will eat (usually measured in units of carbohydrates) [1]. For patients with T1DM, the prognosis of future glucose values must be considered based on the values of ingested food, insulin injection, and / or its glucagon. Therefore, as it can be deduced this requires many estimates and is not a clearly defined process in the variables involved.

For these reasons, it can be ensured that diabetes is a serious disease that according to the World Health Organization [3] more than 420 million people suffered from this disease in 2014. In order to live as normal a life as possible, they need to control it and for this they must take many factors into account.

Nowadays, with the help of technology, data collection and decision making are easier. In order to effectively use data, it has to be well organized. Currently, most diabetic patients use a series of devices that record all their activity along with different physical variables. It is

very common to use activity wristbands and to a lesser extent injection pumps and continuous glucose meters.

This project tries to put technology at the service of the diabetic patients. For this we have developed a web application that allows managing all the data collected by the devices mentioned above.

This application helps users in several ways. On the one hand, patients can easily keep track of all data collected, on the other, doctors will be able to access the data without patients having to go to the clinic, thus favouring a more exhaustive control of them. Finally, researchers will be able to access data so that they can generate prediction models or see what margin of error the measuring devices generate.

For all the above and the interest we both have in the development of applications and data processing we decided to embark on this project. After a month of research and 5 months of development we have managed to obtain a functional version of the application, valid for doctors, researchers and patients. So far, we can manage ten different types of files that come from 5 measuring devices.

It is very gratifying to feel that applying our knowledge in a medical-related project can help doctors work in a more useful way for the benefit of patients.

1.2. Objetivo

Nuestro objetivo principal en este trabajo de fin de grado ha sido crear una aplicación web que sea capaz de gestionar la subida, tratamiento y bajada de datos de los dispositivos de medición más habituales, que generan archivos de tipo CSV (archivos separados por comas). Además de la funcionalidad, la aplicación tiene que ser amigable para el usuario y que no dé lugar a errores.

Para llegar a estos objetivos hemos planteado varios pasos a seguir:

- Análisis de todas las fuentes de datos.
- Diseño de una base de datos que pudiese darnos la funcionalidad que deseábamos.
- Estudio de las diferentes formas de implementar scripts de Python que interactúen con bases de datos en una web.
- Investigación de los diferentes framework de aplicaciones web disponibles en la actualidad.
- Aprendizaje de Python y revisión de los conocimientos en HTML y CSS.
- Aprendizaje sobre el funcionamiento de Django (El framework de aplicaciones web seleccionado).
- Aprendizaje de la librería Pandas de Python.

1.3. Plan de trabajo

Para este trabajo hemos decidido utilizar la metodología ágil de SCRUM ya que creemos que era la que mejor nos venía para conseguir un producto final válido. Para ello lo primero que hicimos fue definir una serie de tareas en un tablón online llamado Trello. Separamos estas en 3 distintas: Pendientes, En proceso y Finalizadas, además contábamos con una extra en la que íbamos añadiendo toda la información que considerábamos que podría ser de utilidad en un

futuro. Respecto al apartado de control de versiones hemos usado un repositorio privado en Git Hub, gracias a esto hemos podido trabajar en paralelo de forma cómoda y segura.

Lo primero y más esencial era generar una plantilla base.html para poder trabajar de forma correcta con el framework de aplicaciones web Django. Por ello la primera versión solo tenía una página main, un registro de usuario y la base.html desarrollada con la cabecera y el pie de página. En la segunda versión queríamos implementar toda la estructura de usuarios, es decir, queríamos que hubiese distintos tipos de usuarios y que hubiese una jerarquía entre ellos. Para ello creamos un registro de médicos e investigadores dentro de la página de administración de Django. Por último, una vez teníamos la estructura montada, había que añadir funcionalidades a la aplicación. En nuestro caso, permitir la subida de archivos a la base de datos y posteriormente la bajada de estos.

Con cada iteración generada hacíamos un análisis de requisitos cumplidos, no cumplidos y necesarios para implementar en la siguiente fase. De esta manera no nos alejábamos nunca del producto final.

1.4. Resultado

La aplicación web desarrollada permite subir todos los tipos de archivo que se habían planteado al inicio del proyecto. Los datos provienen de la pulsera Fitbit Ionic, la bomba de insulina de Medtronic y su medidor de glucosa, el medidor de glucosa de Free Style y la bomba de insulina de Roche. En cuanto a las funcionalidades de descarga y visualización, hemos conseguido alcanzar todos los objetivos planteados. Podemos seleccionar cualquier dato de los subidos a la base de datos y descargarlos de forma rápida en un archivo csv o visualizarlos tanto en una tabla como en un gráfico (estas dos últimas opciones se visualizan en la misma aplicación web).

La gestión por parte del administrador es más sencilla de lo esperado gracias al panel de administración de Django y a los formularios personalizados que hemos creado. También, gracias a este panel, la gestión de los centros médicos y de investigación es muy cómoda ya que te permite crearlos, editarlos y eliminarlos pulsando unos pocos botones. En la parte visual, hemos conseguido crear lo que es a nuestro parecer una interfaz de usuario bastante amigable. Al tener menús personalizadas para cada tipo usuario facilitamos la usabilidad de la página ya que no se sobrecarga con información innecesaria.

1.5. Organización de la memoria

El resto de la memoria está organizada de la siguiente manera:

- En el capítulo 2 explicamos las fuentes de datos de los dispositivos de medición.
- En el capítulo 3 definimos las características técnicas, es decir, qué herramientas software, base de datos, framework de aplicaciones web y lenguajes utilizamos.
- El capítulo 4 lo dedicamos al desarrollo del software de nuestra aplicación: análisis de datos, código Python y parte HTML y CSS.
- En el capítulo 5 explicamos los resultados finales del proyecto.
- En el capítulo 6 se encuentran las conclusiones.
- En el capítulo 7 están las aportaciones de cada miembro del proyecto.
- En el capítulo 8 se encuentran las referencias.

2. Fuentes de datos

2.1. Dispositivos inteligentes seleccionados

La información recogida en este apartado está relacionada con los siguientes objetivos del proyecto HERACLES-II [4]:

- O1.1. Estudiar los diferentes Dispositivos Inteligentes (DI) del mercado tanto de uso médico como de uso general para determinar el tipo de datos que pueden proporcionar.
- O1.2. Mejorar la base de datos de pacientes creada en HERACLES incorporando las nuevas variables obtenidas con los DI.

Estos objetivos a fecha de 1 de diciembre de 2019 han sido completados total o parcialmente. En el caso del O.1.1. además del estudio bibliográfico, se ha realizado una búsqueda de dispositivos que fueran fáciles de utilizar, fiables, con base de datos accesibles mediante API y cuyos datos sean exportables a formatos csv, xml o JSON.

Van den Bulck [5] hace algunas observaciones y comentarios sobre aplicaciones móviles y dispositivos portátiles para monitorizar el sueño que son aplicables a todas las formas de tecnologías de salud del consumidor y que resumimos a continuación:

- La mayoría de estas tecnologías no están etiquetadas como dispositivos médicos, pero sí transmiten declaraciones de valor explícitas o implícitas sobre nuestro estándar de salud.
- Es necesario determinar si el uso de la tecnología influye en el conocimiento y la actitud de las personas sobre su propia salud y cómo lo hace.
- El autodiagnóstico basado en datos recopilados por los propios profesionales de la medicina podría ser incoherente con los diagnósticos clínicos proporcionados por los profesionales de la medicina.
- Aunque la auto monitorización puede revelar problemas de salud no diagnosticados, es probable que dicha monitorización a nivel de una gran población dé lugar a muchos falsos positivos.
- El uso de tecnologías puede crear una obsesión malsana (o incluso dañina) con la salud personal para las personas o los miembros de sus familias que utilizan dichas tecnologías. Esto puede ser especialmente problemático en familiares de pacientes con diabetes en el que el miedo a hipoglucemias inadvertidas puede provocar un comportamiento obsesivo y sobre corrector de las pautas insulínicas.

La mayoría de las tecnologías de salud y rendimiento que hemos revisado se han desarrollado sobre la base de las necesidades del mundo real, pero sólo una pequeña proporción ha demostrado ser eficaz a través de una validación rigurosa e independiente. La validación científica independiente proporciona el nivel más alto de apoyo a la tecnología. Sin embargo, no siempre es posible alcanzar los niveles más altos de validación. Desde una perspectiva de investigación, las tecnologías de salud del consumidor pueden clasificarse en aquellas que se han utilizado en estudios de validación, estudios observacionales, detección de trastornos de salud y estudios de intervención [6]. En ausencia de una validación independiente, las tecnologías que no han sido validadas con arreglo a un patrón ”golden” (pero que se utilizan regularmente en investigación) se consideran ”bien soportadas”. Otros factores técnicos que los usuarios deben tener en cuenta son si los dispositivos requieren calibración o formación

especializada para configurar e interpretar los datos, la portabilidad y el alcance físico de la transmisión/grabación de señales, las capacidades de transferencia de datos Bluetooth/ANT+ y en tiempo real, y la capacidad y seguridad de almacenamiento de datos a bordo o en la nube.

Con estas premisas se han seleccionado el dispositivo Fitbit-Ionic para obtener las señales de actividad, ritmo cardiaco y sueño y el sensor Free-Style Libre de Abbot para monitorización continua de glucosa. También se utiliza la información proporcionada por los dispositivos de infusión subcutánea continua de insulina (bombas) que portan algunos de los pacientes seleccionados, en concreto Minimed de la firma Medtronic y el dispositivo de Roche.

2.2. Descripción de los datos

En este apartado se explica el formato de los ficheros que contienen los datos obtenidos de los pacientes del Hospital Universitario Príncipe de Asturias de Alcalá de Henares en Madrid desde el 13 de junio de 2018 hasta el 17 de julio de 2019. Los datos pueden provenir de las siguientes fuentes y dispositivos:

- Free Style Libre- Abbott
- Minimed Medtronic CGM
- Medtronic Insulin Pump
- Roche Insulin Pump
- Fitbit Ionic

Hay que tener en cuenta que no todos los pacientes usaron todos los dispositivos y que en ocasiones no es posible recuperar todos los registros. Con estos dispositivos hemos registrado los siguientes datos:

- Glucosa intersticial
- Glucosa en sangre
- Insulina inyectada con múltiples dosis de insulina (MDI)
- Notas de estimación de unidades de carbohidratos ingeridos, tomadas por cada paciente.
- Insulina inyectada mediante un dispositivo de infusión de insulina de Medtronic y/o Roche, que registra las inyecciones de insulina basal y en bolo cada cinco minutos.
- Ritmo cardíaco
- Estado del sueño y de la siesta
- Calorías quemadas
- Número de pasos
- Notas de los pacientes
- Información adicional

2.3. Fuentes de datos

A continuación, describimos la información recogida en cada uno de los ficheros de los distintos dispositivos inteligentes. Explicaremos el contenido de cada uno de los ficheros y explicaremos con detalle aquellas columnas que nos proporcionan información útil para almacenar en nuestra base de datos y posterior uso en la investigación.

2.3.1. Free Style Libre- Abbott

El dispositivo Free Style Libre- Abbott es un monitor continuo de glucosa que mide la glucosa intersticial. La Figura 1 muestra una imagen en la que se puede ver la diferencia entre la medida de la glucosa intersticial y la medida de la glucosa en sangre que se realiza habitualmente con una tira y un glucómetro. Los medidores utilizados para leer los dispositivos Free Style Libre - Abbott disponen también de una ranura para la lectura de la tira. Normalmente existe un decalaje temporal entre la medida en sangre y la medida de glucosa intersticial de unos 10 a 15 minutos. La información recogida en el fichero exportado en formato txt o csv, es en realidad un csv que contiene las siguientes columnas. Cada fila contiene un único tipo de registro.

- ID: Identificador del evento, lo genera automáticamente el medidor
- Hora: Fecha y hora del registro
- Tipo de registro: Identifica el registro medido y cuyo valor contiene la línea
    - Tipo 0: Histórico glucosa (mg/dL)
    - Tipo 1: Glucosa leída (mg/dL)
    - Tipo 2: Glucosa de la tira (mg/dL)
    - Tipo 4: Evento de insulina: puede ser una medida de Insulina de acción rápida (unidades) o insulina de acción rápida sin valor numérico
    - Tipo 5: Evento de Alimentos: puede ser alimentos sin valor numérico o valor de carbohidratos (gramos)
    - Tipo 6: Evento de Cambio de hora: La fila contiene hora anterior y hora actualizada
- Histórico de glucosa (mg/dL): Es la glucosa medida automáticamente, cada 15 minutos, con el sensor.
- Glucosa leída (mg/dL) Glucosa intersticial medida con el sensor de forma manual, puede haber registro en cualquier instante de tiempo, ya que lo decide el usuario.
- Insulina de acción rápida sin valor numérico: Registro de un evento de inyección de insulina, informativo no contiene el número de unidades inyectadas.
- Insulina de acción rápida (unidades): Registro de un evento de inyección de insulina, contiene el número de unidades inyectadas
- Alimentos sin valor numérico: Registro de un evento de ingestión de alimentos, informativo no contiene el número de gramos ingeridos.
- Carbohidratos (gramos): Registro de un evento de ingestión de alimentos, contiene el número de carbohidratos (en gramos) ingeridos.
- Insulina de acción lenta sin valor numérico: Registro de un evento de inyección de insulina de acción lenta, informativo no contiene el número de unidades inyectadas.
- Insulina de acción lenta (unidades): Registro de un evento de inyección de insulina de acción lenta, contiene el número de unidades inyectadas.
- Notas: Cualquier anotación que el usuario quiera recoger.

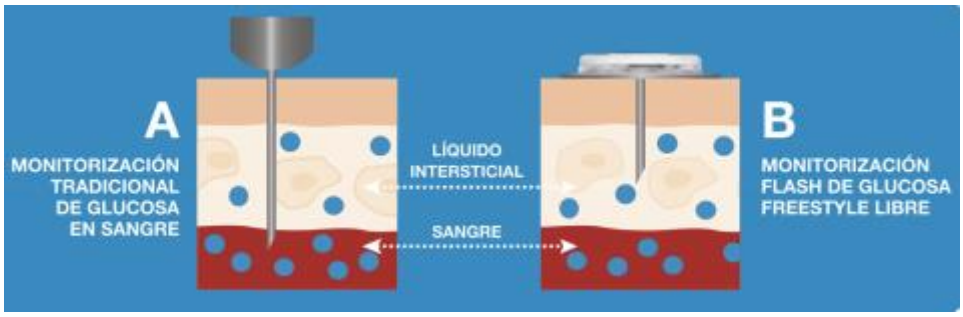


Figura 1: Diferencia entre medida de la glucosa intersticial y medida de la glucosa en sangre [7]

- Glucosa de la tira (mg/dL): Glucosa en sangre, medida con la tira. Ver la Figura 1 para ver la diferencia entre glucosa intersticial y glucosa en sangre.
- Cetonas (mmol/L)
- Insulina comida (unidades): Configuración del paciente
- Insulina corrección (unidades): Configuración del paciente
- Insulina cambio usuario (unidades): Configuración del paciente
- Hora anterior: registro de cambio de hora del lector
- Hora actualizada: registro de cambio de hora del lector

hupa

ID	Hora	Tipo de registro	Histórico glucosa (mg/dL)	Glucosa leída (mg/dL)	Insulina de acción rápida sin valor numérico	Insulina de acción rápida (unidades)	Alimentos sin valor numérico	Carbohidratos (raciones)	Insulina de acción lenta sin valor numérico	Insulina de acción lenta (unidades)	Notas	Glucosa de la tira (mg/dL)	Cetonas (mmol/L)	Insulina comida (unidades)	Insulina corrección (unidades)	Insulina cambio usuario (unidades)	Hora anterior	Hora actualizada
0	2018/09/12 20:28	6															2018/09/12 20:21	2018/09/12 20:28
36377	2016/04/04 14:21	4															1	
36377	2016/04/04 14:21	5														1		
36377	2016/04/04 14:21	1		187														
36444	2016/04/04 14:22	1		188														
36461	2016/04/04 14:17	0	190															
36462	2016/04/04 14:36	1		159														
36475	2016/04/04 14:36	1		159														
36496	2016/04/04 14:33	0	161															
36497	2016/04/04 14:48	0	115															

Figura 2: Ejemplo de un fichero csv de FreeStyle Libre

La Figura 2 muestra el aspecto de un fichero de datos de FreeStyle Libre de Abbot abierto con un editor de texto plano.

2.3.2. Minimed Medtronic CGM y Medtronic Insulin Pump

Los dispositivos de Medtronic permiten recoger en un único fichero toda la información tanto de la bomba de Insulina como del medido continuo de glucosa. El registro se realiza cada 5 minutos y la información que se recoge es la siguiente:

- Index: Es un índice autoasignado automáticamente para cada registro
- Date: Día del registro
- Time: Hora de toma del registro

- New Device Time: Cuando se realiza la sustitución del sensor o del catéter de la bomba, se registra la hora.
- BG Reading (mg/dL): Medida de glucosa en sangre realizada con un glucómetro de la firma Báyer que se enlaza por conexión inalámbrica y que sirve además para calibrar el medidor. Para una correcta medida es necesario realizar una medida en sangre cada 8 horas.
- Linked BG Meter ID: Identificador del dispositivo de medida de glucosa en Sangre.
- Basal Rate (U/h): Las bombas de Medtronic utilizan un solo tipo de insulina, por lo que la insulina de acción lenta se sustituye por una serie de microinyecciones de insulina, realizadas cada 5 minutos a un ritmo indicado por este Basal rate. La bomba anota cada cambio de valor y sigue inyectando a ese ritmo hasta que aparece registrado un valor diferente.
- Bolus Type: Tipo de Bolo, puede ser Normal o cuadrado
- Bolus Volume Selected (U): Cantidad seleccionada de insulina para el bolo.
- Bolus Volume Delivered (U): Cantidad realmente inyectada, en ocasiones no es la misma que la seleccionada porque se producen errores o fallos en la inyección, por ejemplo, debido a obstrucciones del catéter.
- Bolus Duration (h:mm:ss): Duración del bolo
- Prime Type: Tipo de Bolus
- Prime Volume Delivered (U): Volumen inyectado real
- Alarm: Registro de una señal de alarma.
- Suspend: Registro de una suspensión de inyección de insulina
- Rewind: Registro de reinicio del sensor
- BWZ Estimate (U): Carbohidratos estimados
- BWZ Target High BG (mg/dL): Datos del Paciente, Glucosa Objetivo máxima. Normalmente a las dos horas de la comida. Se anota en ocasiones para indicar que hay un cambio.
- BWZ Target Low BG (mg/dL): Datos del Paciente, Glucosa Objetivo mínima. Normalmente a las dos horas de la comida. Se anota en ocasiones para indicar que hay un cambio.
- BWZ Carb Ratio (U/Ex): Proporción de Insulina / Carbohidratos
- BWZ Insulin Sensitivity (mg/dL/U): Sensibilidad a la insulina
- BWZ Carb Input (exchanges): Carbohidratos Ingeridos
- BWZ BG Input (mg/dL): Glucosa en sangre
- BWZ Correction Estimate
- BWZ Food Estimate (U): Unidades de Carbohidratos
- BWZ Active Insulin (U): Insulina activa
- Sensor Calibration BG (mg/dL): Medida de glucosa en sangre, es la misma que la de la columna BG Reading.
- Sensor Glucose (mg/dL): Medida de glucosa intersticial mediante el medidor continuo de glucosa de Medtronic.
- ISIG Value: Valor de la señal eléctrica que nos da la glucosa
- Event Marker: Marca de evento
- Bolus Number: Número de bolo
- Bolus Cancellation Reason: Cuando se cancela el bolo, se anota la razón de esa cancelación.
- BWZ Unabsorbed Insulin Total (U): Cantidad de insulina no absorbida

- Final Bolus Estimate: Valor estimado de final de Bolus.
- Scroll Step Size: No utilizado.
- Insulin Action Curve Time: No utilizado.
- Sensor Calibration Rejected Reason: Si se rechaza el valor de calibración se anota el motivo.

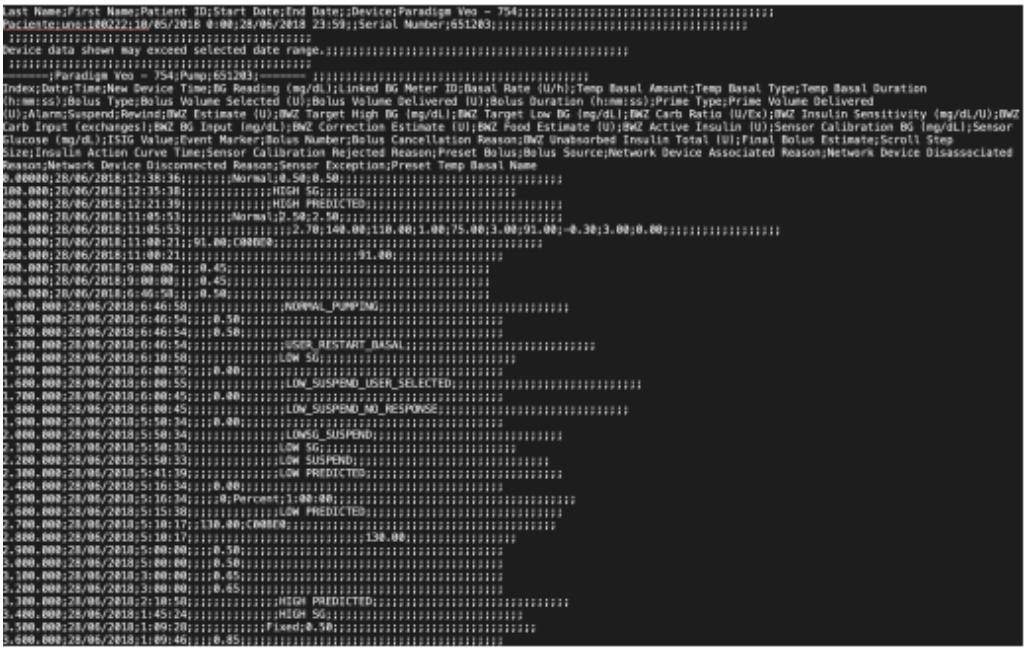


Figura 3: Ejemplo de un fichero csv de Freestyle Libre

Los ficheros de Medtronic vienen divididos en dos partes. Por un lado, vienen los registros de la bomba de insulina, y por otro lado los del sensor en el caso de que existan ambos. Hay que tener cuidado porque las fechas y horas vienen duplicadas, es decir existe un registro de la bomba con un día y hora y puede existir uno con esa misma fecha y hora del medidor continuo de glucosa. El primero con información en cualquiera de las columnas descritas anteriormente excepto en la de Sensor Glucose y el segundo mostrará únicamente valor en las columnas Date, Time y Sensor Glucose. La Figura 3 muestra un ejemplo de un archivo de datos de Medtronic abierto con un editor de texto plano.

2.3.3. Roche Insulim Pump

Los dispositivos de Roche permiten recoger en un único fichero toda la información de la bomba de Insulina. El registro se realiza cada 5 minutos y la información que se recoge es la siguiente:

- Nombre: Nombre del paciente
- Fecha: Día del registro
- Hora: Hora del registro
- Hito: Momento del registro
- Glucemia: Glucosa en sangre medida con glucómetro externo
- Unidad Glucemia: Unidad de medida
- Hidratos de Carbono

- Cantidad de insulina
- Tipo de Bolo: Tipo de Bolo
- Dosis Basal (UI/H)
- Total, insulina inyectada por Bolo
- Total, insulina inyectada por Basal y Bolo
- Eventos de arranque y parada de la bomba
- Comentarios

A	B	C	D	E	F	G	H	I	J	M	N	O	P	Q	R	S	T	U	V
Nombre	NHC	Fecha	Hora	Hito	Glucemi	Unidad (	Pre&P	Hidratc	Insulina	Tipo de Bol	Unidades	Dosis Basal	Perfil	Total Bolo	Total Basal y	Eventos S.I.	Evento	Comentarios	
Paciente 2		28/6/18	9:31									0.44	1			Parada			
Paciente 2		28/6/18	9:22							Std	3.3								
Paciente 2		28/6/18	9:21	Antes D	201	mg/dL		10											
Paciente 2		28/6/18	9:00									0.85	1						
Paciente 2		28/6/18	8:08									0.91	1			Arranque			
Paciente 2		28/6/18	7:58									1.45	1			Parada			
Paciente 2		28/6/18	7:00									1.5	1						
Paciente 2		28/6/18	6:00									1.65	1						
Paciente 2		28/6/18	5:00									1.4	1						
Paciente 2		28/6/18	4:00									1.35	1						
Paciente 2		28/6/18	3:00									0.95	1						
Paciente 2		28/6/18	2:00									0.9	1						
Paciente 2		28/6/18	0:33							Std	1.5								
Paciente 2		28/6/18	0:32	Antes D	141	mg/dL		15											
Paciente 2		28/6/18												4.8	15.25				
Paciente 2		28/6/18	0:00									0.7	1						
Paciente 2		27/6/18	23:00									0.9	1						
Paciente 2		27/6/18	22:00									1.3	1						

Figura 4: Fichero de Roche abierto con excel

En la Figura 4 podemos observar un ejemplo de archivo de datos de Roche abierto con Excel.

2.3.4. Fitbit Ionic

El dispositivo Fitbit Ionic (ver Figura 5) incorpora los siguientes sensores y motores:

- Un acelerómetro de tres ejes MEMS, que monitoriza los patrones de movimiento.
- Un altímetro, que monitoriza los cambios de altitud.
- Un receptor GPS con GLONASS, que monitoriza tu ubicación durante un entrenamiento.
- Un monitor óptico de ritmo cardiaco.
- Un sensor de luz ambiental.
- Un motor de vibración.
- Un transceptor de radio Bluetooth 4.0.
- Un chip wifi.
- Un chip NFC.

Ionic almacena datos, como estadísticas diarias, información de sueño e historial de ejercicios, durante siete días y es necesario sincronizarlo de manera que no se pierdan datos. Puede realizarse una sincronización automática con el teléfono móvil. Los datos almacenados pueden exportarse a una serie de ficheros csv con información separada. El proceso es complejo y ha sido necesario desarrollar un script específico para recuperar la información. La mayoría de los datos se registran en un fichero diario, por lo que obtendremos un fichero para cada tipo de dato y para cada día.

Figura 5: Aspecto exterior del dispositivo Ionic de Fitbit [8]

2.3.4.1. Fitbit registro de calorías

El Ionic recoge información de las calorías gastadas cada minuto. Los datos almacenados son:

- dateTime: Hora del registro (el día está en el nombre del fichero). El formato es HH:MM:SSS
- value: Valor del consumo de calorías en unidades, ejemplo 0.8050000071525574.

Si queremos almacenar el valor cada 5 minutos, hay que sumar los valores correspondientes. La Figura 6 muestra un ejemplo de un archivo de datos de calorías consumidas abierto con un editor de texto plano.

paciente1_absys_cals_2018-06-13.csv

```
dateTime,value
00:00:00,0.8050000071525574
00:01:00,0.8050000071525574
00:02:00,0.8050000071525574
00:03:00,0.8050000071525574
00:04:00,0.8050000071525574
00:05:00,0.8050000071525574
00:06:00,0.8050000071525574
00:07:00,0.8050000071525574
00:08:00,0.8050000071525574
00:09:00,0.8050000071525574
00:10:00,0.8050000071525574
00:11:00,0.8050000071525574
00:12:00,0.8050000071525574
00:13:00,0.8050000071525574
00:14:00,0.8050000071525574
```

Figura 6: Ejemplo de archivo de registro de calorías con Ionic

2.3.5. Fitbit registro de pasos

El Ionic recoge información de los pasos caminados cada minuto. Los datos almacenados son

- dateTime: Hora del registro, el día está en el nombre del fichero. El formato es HH:MM:SSS
- value: Valor en unidades, ejemplo 3.

Si queremos almacenar el valor cada 5 minutos, hay que sumar los valores correspondientes. La Figura 7 muestra un ejemplo de un archivo de datos de pasos abierto con un editor de texto plano.

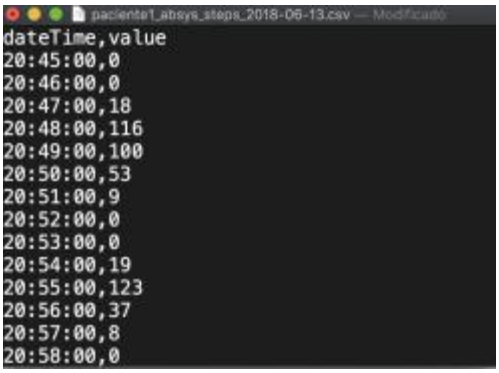


Figura 7: Ejemplo de archivo de registro de pasos con Ionic

2.3.6. Fitbit registro de ritmo cardiaco

El Ionic también recoge información del ritmo cardiaco mediante un sensor óptico con información cada 3 segundos. Los datos almacenados son

- Time: Hora del registro, el día está en el nombre del fichero. El formato es HH:MM:SSS
- Heart Rate: Valor en unidades, ejemplo 70.

Si queremos almacenar el valor cada 5 minutos, hay que hacer una media de los valores correspondientes. La Figura 8 muestra un ejemplo de un archivo de datos de ritmo cardiaco con un editor de texto plano.

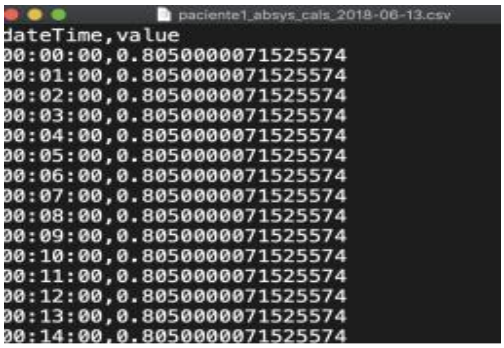


Figura 8: Ejemplo de archivo de ritmo cardiaco con Fitbit Ionic.

2.3.7. Fitbit registro de sueño nocturno

Fitbit proporciona dos tipos de ficheros para el sueño; unos con el resumen diario y otros con el detalle de cada día con registro de información cada 1 minuto. En el caso del resumen se recogen los datos siguientes:

- Date: fecha en formato 2018-06-14
- Start Time: Hora de comienzo del sueño nocturno en formato 2018-06- 14T02:50:00.000
- End Time: Hora de final del sueño nocturno en formato 2018-06- 14T10:06:00.000
- MainSleep: Tipo de sueño (True o False)
- Efficiency: Eficiencia del sueño en porcentaje
- Duration: Duración en segundos (26160000)
- Minutes Asleep: Minutos dormido.
- Minutes Light: minutos en sueño ligero.
- Minutes Deep: minutos en sueño profundo.
- Minutes REM: minutos en fase REM.
- Minutes Awake: Minutos despierto.
- Minutes in Bed: Minutos acostado (minutos dormido + minutos despiertos).

El fichero de la información con el detalle del sueño nocturno proporciona un registro con cada cambio de estado del sueño. La Figura 9 muestra en detalle esta información en el que se ve el registro de los cambios en un fichero de ejemplo. La información registrada es la siguiente:

- dateTime: Día y Hora en el formato 2018-06-14T02:50:00.000 15
- Duration: Duración de esa fase en segundos
- State: Estado, puede ser
  - light
  - wake
  - rem
  - Deep

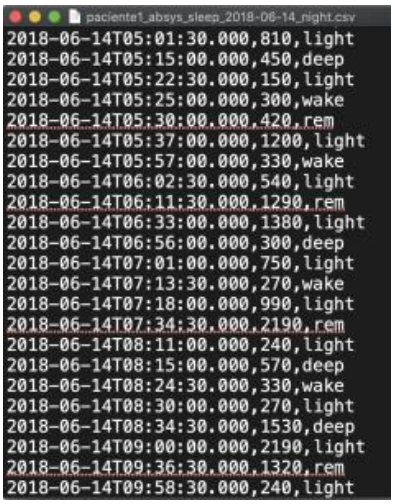


Figura 9: Ejemplo de archivo de detalle del sueño nocturno con Fitbit Ionic.

2.3.8. Fitbit registro de siesta

Al igual que en el caso del sueño nocturno se registran dos tipos de ficheros; unos con el resumen diario y otros con el detalle de cada siesta con registro de información cada 1 minuto. En el caso del resumen se recogen los datos siguientes:

- Date: fecha en formato 2018-06-14
- Start Time: Hora de comienzo del sueño nocturno en formato 2018-06-14T02:50:00.000
- End Time: Hora de final del sueño nocturno en formato 2018-06-14T10:06:00.000
- MainSleep: Tipo de sueño (True o False)
- Efficiency: Eficiencia del sueño en porcentaje
- Duration: Duración en segundos (26160000)
- Minutes Asleep: Minutos dormido.
- Minutes Light: minutos en sueño ligero.
- Minutes Deep: minutos en sueño profundo.
- Minutes REM: minutos en fase REM.
- Minutes Awake: Minutos despierto.
- Minutes in Bed: Minutos acostado (minutos dormido + minutos despierto). Date,Start Time,End

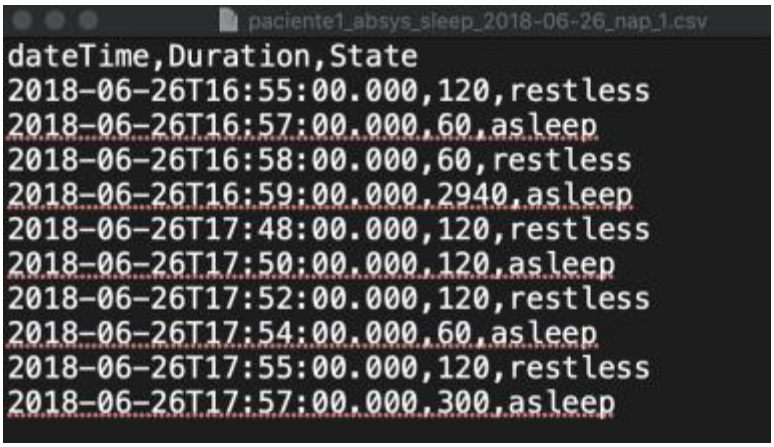


Figura 10: Ejemplo de archivo de detalle del sueño en la siesta con Fitbit Ionic.

Por lo que respecta a los ficheros con el detalle de la siesta, se registra un fichero por cada siesta, es decir, que, si una persona realiza tres descansos, se registran tres ficheros con la siguiente información:

- dateTime, en el formato 2018-06-26T16:55:00.000
- Duration, en segundos
- State : puede ser
  - restless
  - asleep
  - awake

3. Características técnicas

Una de las partes más importantes al empezar un proyecto es decidir, viendo las características que se quieren obtener, qué herramientas se van a utilizar. Para ello lo primero que se tiene que hacer es definir los requisitos principales del proyecto.

3.1. Captura de requisitos

Se tendrán que especificar los requisitos funcionales de la web; estos se deben especificar de forma explícita por la persona que quiere realizar la web ya que es la que tiene la idea principal del funcionamiento deseado. También se tendrán que definir los requisitos técnicos de la aplicación; estos se tendrán que discutir teniendo en cuanta las funcionalidades especificadas anteriormente.

3.1.1. Requisitos funcionales

Para generar los mejores requisitos funcionales lo que decidimos hacer fue hablar tanto con el responsable de la aplicación como con los usuarios de esta. En nuestro caso solo teníamos acceso al responsable por lo que solo hablamos con él. Una vez reunidos con él se le plantearon tres preguntas principales.

1. ¿Cuál es el objetivo del sitio web?
2. ¿Qué tipo de usuarios tendrá?
3. ¿Qué tareas llevarán a cabo los distintos tipos de usuarios?

El objetivo del sitio es crear una web que sea capaz de gestionar archivos de tipo csv. En este caso la gestión abarca la subida, tratamiento, almacenado y descarga de los datos. Lo que se quiere hacer es generar una base de datos que sea capaz de almacenar datos de múltiples dispositivos de medición de salud. Los dispositivos se pueden dividir en dos secciones, por un lado, la pulsera de actividad Fitbit y por otro los dispositivos relacionados directamente con el tratamiento de la diabetes (bombas de insulina y sensores de glucosa). Todos estos archivos toman mediciones en momentos en el tiempo precisos, es decir un día, una hora, un minuto y un segundo concreto. Para facilitar el análisis de los datos en la descarga de estos se quiere establecer un estándar de tiempos para todos los archivos, en nuestro caso las mediciones tienen que guardarse con intervalos de 5 minutos (ej.: 10:00, 10:05, 10:10…). También es necesario dar la posibilidad de visualizar los datos, tanto en tabla como en gráfico.

Respecto a los tipos de usuario se va a crear una jerarquía de usuarios para que cada tipo de usuario tenga la capacidad de hacer lo que necesite. Los tipos de usuario son 4; en la parte más alta están los administradores, seguidos de médicos e investigadores y por último están los pacientes.

Los administradores podrán dar de alta al resto de usuarios siendo su trabajo clave el alta de médicos e investigadores. Los médicos e investigadores son usuarios muy similares, ambos pueden dar de alta a usuarios subir información de los pacientes mediante la subida de archivos de tipo CSV (coma separated values o archivos separados por comas) y descargar información de los pacientes. Por último, están los pacientes, estos solo podrán subir información mediante la subida de archivos de tipo CSV y descargarla.

3.1.2. Requisitos técnicos

Estos requisitos son los que garantizan la calidad del desarrollo informático de la web. Lo primero que hicimos fue pensar en cómo crear un sitio que pudiese gestionar web, scripts y base de datos. Nuestra primera opción fue crear una web con html y css usando bootsrap, crear la base de datos con php myadmin y gestionar los scripts con php. Pero tras realizar una búsqueda vimos que existían los framework de aplicaciones web. Estos son muy utilizados en los entornos de aplicaciones web que requieren algo más que código html. Un framework de aplicaciones web es una aplicación genérica incompleta que se puede configurar a nuestro gusto añadiendo líneas de código. Por ello nos decidimos a usar un framework de aplicaciones web en vez de generar todos los archivos nosotros.

3.2. Herramientas utilizadas

En nuestro caso las herramientas utilizadas se dividen en 3 ramas principales. Lo primero y más esencial es el framework de aplicaciones web ya que a partir de ahí se decidiría que lenguaje utilizaríamos para los scripts y como organizaríamos la parte de html y css. Una vez decidido el framework de aplicaciones web deberíamos decidir qué gestor de bases de datos usar ya que para empezar a desarrollar la web necesitaríamos almacenar información en la aplicación. Por último, tendríamos que decidir qué programa usaríamos para gestionarlo todo. Además de lo mencionado anteriormente usaremos un repositorio privado de Git Hub para tener un control de versiones y poder trabajar mejor en pareja paralelizando el trabajo.

3.2.1. Framework de aplicaciones web

Para decidirnos por el framework de aplicaciones web correcto nos enfocamos en buscar cuál nos proporcionaba las características que estábamos buscando. En el caso de Django usa el modelo de vista controlador, usa medidas de seguridad y por lo que pudimos leer era bastante rápido y escalable. La decisión final fue usar Django 2.2.7 que era la versión que estaba disponible en ese momento (posteriormente durante el desarrollo la actualizaríamos a la 2.2.8). Django es uno de los framework de aplicaciones web más utilizados en lo que a aplicaciones web se refiere [9]. Tiene la comodidad de que te genera un panel de administración automáticamente. Además, cuenta con una amplia lista de beneficios a nivel de seguridad (cross site scripting, clickjacking, host header validation, cifrado de contraseñas, etc.) [10].

Como lenguaje de programación utiliza Python. Esto para nosotros ha sido un reto ya que ninguno de los dos había programado en este lenguaje anteriormente.

Para el apartado web utiliza un sistema de plantillas html estas se guardan en una carpeta que defines previamente en el archivo settings.py. Dentro de este sistema tenemos una función muy útil: las etiquetas. Mediante las etiquetas podemos ahorrarnos muchísimo código repetido, además de poder añadir partes de código Python en el interior de las plantillas. Una de las partes en las que se usan mucho las etiquetas es en la platilla base, en nuestro caso esta plantilla se llama base.html y contiene la cabecera y el pie de página de la web [11]. Para beneficiarte de ella solo tienes que llamarla usando una llamada de Django y usar las etiquetas que han creado dentro del archivo para modificar cosas como el title o las importaciones de css.

3.2.2. Base de datos

El gestor de bases de datos que viene por defecto con Django es SQLite3. Ya que ninguno de los dos estaba familiarizado con él decidimos eliminarlo y trabajar con algo con lo que si estábamos familiarizados. Al ver que Django soportaba phpMyAdmin decidimos utilizarlo; es un gestor con el que ambos habíamos trabajado con anterioridad y que permite hacer modificaciones en la base de datos con bastante facilidad ya que cuenta con una interfaz gráfica bastante intuitiva.

3.2.3. Software de programación

En este apartado no teníamos mucha idea de cuál utilizar ya que ninguno de los dos había programado con anterioridad en Python. Tras una breve búsqueda encontramos muchos editores gratuitos entre los cuales estaba Visual Studio Code. De los programas gratuitos es el que más nos convenció por su amplia cantidad de plugins útiles, además la interacción con Git Hub es bastante buena. Más adelante nos dimos cuenta de que con la universidad tenemos derecho a licencias de JetBrains PyCharm 2019. Nada más probarlo nos decidimos a cambiar de entorno de desarrollo porque este proporciona más facilidades a la hora de programar [13]. El sistema de auto completado es más eficaz, las propuestas de importaciones son perfectas, los errores los detecta con mayor eficacia y el sistema de depuración es muy similar a los que hemos usado con anterioridad. La integración de PyCharm con Git Hub es muy buena [12], te permite gestionar las versiones y usar todas las características de Git con mucha facilidad y rapidez.

4. Desarrollo del software

Para realizar el Trabajo de Fin de Grado hemos decidido utilizar el lenguaje de programación Python y un framework de aplicaciones web llamado Django. Este framework de aplicaciones web se instala en el proyecto de manera que se crean por defecto unos archivos que están preparados para trabajar directamente sobre ellos, simplificando el trabajo. Django está preparado para que estos archivos estén comunicados entre ellos de manera que, por ejemplo, el archivo urls.py creado por defecto (archivo que contiene las urls) comunique con el archivo views.py (archivo que contiene las funciones) ejecutando la función en dicho archivo. Más adelante hay una explicación más detallada de esto. También ofrece la posibilidad de trabajar con los HTML de una manera cómoda, creando una plantilla padre llamada base.html de la que heredan las demás evitándonos tener que repetir código.

La gran ventaja de utilizar Django es que la mayoría de las cosas que se necesitan implementar ya están incorporadas, solo hay que importarlas y adaptarlas a nuestro proyecto.

4.1. Bases de Django

Django cuenta con los siguientes archivos predeterminados:

- Models.py: En la Figura 11 se declaran las clases del modelo, es decir, las tablas de la base de datos se definen aquí y al ejecutar el comando *Python manage.py migrate* se realiza una migración que crea las tablas [14].

```python
class Paciente(User):
    TYPES = (
        ('Tipo 1', 'Diabetes de tipo 1'),
        ('Tipo 2', 'Diabetes de tipo 2'),
        ('Otro', 'Diabetes de otro tipo')
    )
    birth_date = models.DateField()
    diabetes_type = models.CharField(max_length=10, choices=TYPES)
    treatment = models.ForeignKey(Tratamiento, on_delete=models.PROTECT, null = True)
    start_date = models.DateField(null = True, blank = True)
    doctor_id = models.ForeignKey(Medico, on_delete=models.PROTECT)
    first_date = models.DateTimeField(null = True, blank= True, help_text="Primera fecha con datos del paciente")
    last_date = models.DateTimeField(null=True, blank=True, help_text="Ultima fecha con datos del paciente")

    def __str__(self):
        return self.username

    class Meta:
        db_table = 'paciente'


class Centro_investigacion(models.Model):
    name = models.CharField(max_length=250)
    #postal = models.IntegerField(max_length=250, blank = True, null=True)

    def __str__(self):
        return self.name

    class Meta:
        db_table = 'centro_investigacion'
```

Figura 11: Ejemplo del archivo models.py de Django

- Views.py: En la Figura 12 podemos ver una de las funciones que se declaran dentro de la vista. Es sin duda el archivo en el que más hemos trabajado, ya que cuenta con toda la lógica del programa.
  Cuenta con las funciones:
  - Enviar_correo: Se envía un correo al paciente registrado por el médico.
  - Welcome: Si estás logueado, te dirige a la página principal, en caso contrario te dirige al login.
  - New_password: Cambio de contraseñas para los pacientes.
  - Register: Registro de pacientes por parte de los médicos.
  - Logout: Cierra la sesión.
  - Download: Descarga un archivo csv, muestra una tabla o muestra un gráfico
  - Upload: Subida de datos de pacientes.
  - Roche: Script para tratar los datos de Roche
  - Medtronic: Script para tratar los datos de Medtronic
  - Sleep_nap_resumen: Subida de datos en caso de que sea de tipo resumen
  - Sleep_nap: Subida de datos en el caso de noche o siesta.
  - Free_style_sensor: Script para tratar los datos de Free Style
  - Juntar_glu: Combina los datos de glucosa leída e histórico de glucosa haciendo una media en el caso en el que coincidan.
  - Fitbit: Script para tratar los datos de Fitbit

```python
def login(request):
    try:
        # Creamos el formulario de autenticación vacío
        if request.method == "POST":

                # Recuperamos las credenciales validadas
                username = request.POST['username']
                password = request.POST['password']

                # Verificamos las credenciales del usuario
                user = authenticate(username=username, password=password)

                # Si existe un usuario con ese nombre y contraseña
                if user is not None:
                    # Hacemos el login manualmente
                    do_login(request, user)
                    # Y le redireccionamos a la portada
                    return redirect('/')
                else: #si el nom de usuario o contraseña es incorrecto
                    msg = "Error en el usuario o contraseña"
                    return render(request, "login.html",{'msg':msg})

        # Si llegamos al final renderizamos el formulario

        return render(request, "login.html")
    except:
        msg = "Error en el usuario o contraseña"
        return render(request, "login.html", {'msg': msg})

def logout(request):
    try:
        do_logout(request)
        # Redireccionamos a la portada
        return redirect('/')
    except:
        return redirect('/')
```

Figura 12: Ejemplo del archivo views.py de Django

- Urls.py: Como podemos ver en la Figura 13 el archivo relaciona las url con las funciones, es decir, el usuario al insertar una dirección url, se ejecuta su función asociada en el views.py.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.welcome),
    path('login/', views.login),
    path('logout/', views.logout),
    path('new_password/', views.new_password),
    path('register/', views.register),
    path('download/', views.download),
    path('upload/', views.upload),
]
```

Figura 13: url patterns dentro del archivo urls.py

- Settings.py: Es la configuración central del proyecto en Django. Lo hemos utilizado para realizar la conexión con la base de datos (ver Figura 14), establecer la ruta de las plantillas HTML y poder utilizar el correo de glucmodel@ucm.es

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': '',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',

    }
}
```

Figura 14: Configuración de la base de datos en el archivo settings.py

Django crea por defecto un panel de administración [15], en el que solo los administradores de la aplicación pueden entrar para dar de alta a médicos, investigadores, centros médicos, centros de investigación, pacientes y tipos de tratamiento. Los formularios de usuario que no son los básicos hay que crearlos y decirle a Django de alguna manera que los inserte en el panel de administración. Tanto para médicos como para investigadores hemos creado formularios personalizados en el archivo admin.py. En la Figura 15, se muestra un ejemplo de registro de un médico.

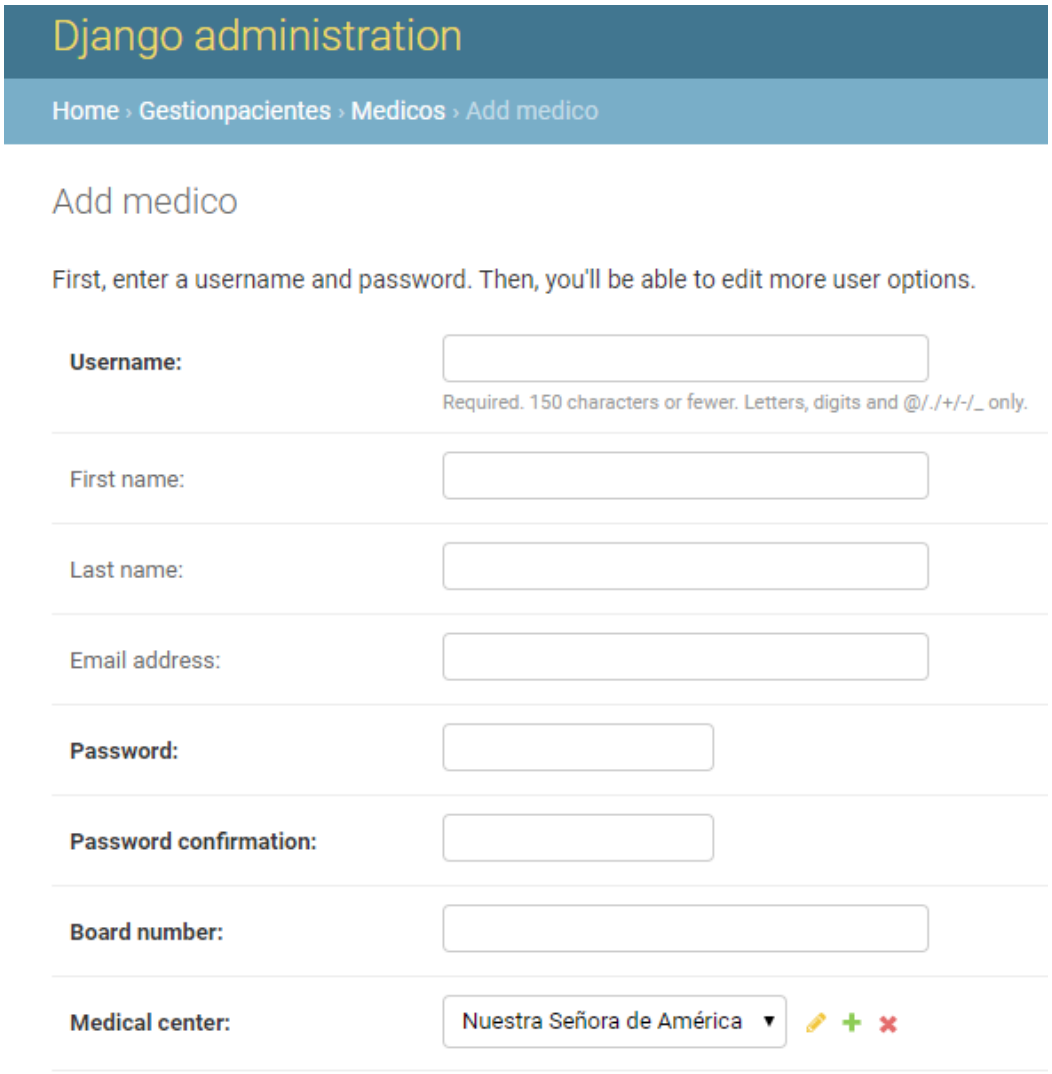


Figura 15: Formulario de registro de un médico, visto desde el panel de administración de Django

4.2. Creación de la plantilla base.html

Django cuenta con la extensión de plantillas. Esto significa que puedes usar las mismas partes de tu HTML para diferentes páginas de tu sitio web.

Las plantillas son útiles cuando quieres usar una misma información o un mismo esquema en más de un lugar. De esta forma no tienes que repetir código en cada uno de los archivos y si deseas modificar algo, solo tienes que hacerlo en un lugar.

En la Figura 16 podemos ver parte de nuestro archivo base.html que cuenta con la cabecera, que dependiendo del tipo de usuario mostrará opciones diferentes, y con el pie de página, que es el mismo para todos los usuarios.

```
</head>
<body id="page-top">
<nav class="navbar navbar-expand-lg bg-secondary text-uppercase fixed-top" id="mainNav">
    <div class="container">
        <a class="navbar-brand js-scroll-trigger" href="/">Glucmodel</a>
        <button class="navbar-toggler navbar-toggler-right text-uppercase font-weight-bold bg-primary text-white rounded" type="button" data-toggle=
            Menu
            <i class="fas fa-bars"></i>
        </button>
        <div class="collapse navbar-collapse" id="navbarResponsive">
            <ul class="navbar-nav ml-auto">
                {% if request.user.is_authenticated %}
                    {% if request.user.id == request.user.medico.user_ptr_id %}
                        <li class="nav-item mx-0 mx-lg-1">
                            <a class="nav-link py-3 px-0 px-lg-3 rounded js-scroll-trigger" href="/register">Añadir Paciente</a>
                        </li>
                    {% endif %}

                    <li class="nav-item mx-0 mx-lg-1">
                        <a class="nav-link py-3 px-0 px-lg-3 rounded js-scroll-trigger" href="/download">Descargar o ver datos</a>
                    </li>

                    <li class="nav-item mx-0 mx-lg-1">
                        <a class="nav-link py-3 px-0 px-lg-3 rounded js-scroll-trigger" href="/upload">Subir datos</a>
                    </li>

                    <li class="nav-item mx-0 mx-lg-1">
                        <a class="nav-link py-3 px-0 px-lg-3 rounded js-scroll-trigger" href="/logout">Logout</a>
                    </li>
                {% endif %}
                {% if not request.user.is_authenticated %}
                    <li class="nav-item mx-0 mx-lg-1">
                        <a class="nav-link py-3 px-0 px-lg-3 rounded js-scroll-trigger" href="/logout">Login</a>
                    </li>
                {% endif %}
            </ul>
        </div>
    </div>
</nav>
{# Navigation Menu #}
</header>
```

Figura 16: Cabecera de nuestro archivo base.html

```
{% extends "base.html" %}
{% load staticfiles i18n %}

{% block title %}Ejemplo{% endblock title %}

{% block css %}

    <!-- Icons font CSS-->
    <link href="{% static 'vendor/mdi-font/css/material-design-iconic-font.min.css' %}" rel="stylesheet" media="all">

{% endblock css %}

{% block content %}
   <h1>Hola Mundo!</h1>
{% endblock content %}

{% block js %}
    <!-- Jquery JS-->
    <script src="{% static 'vendor/jquery/jquery.min.js' %}"></script>
    <!-- Vendor JS-->
    <script src="{% static 'vendor/select2/select2.min.js' %}"></script>
    <script src="{% static 'vendor/datepicker/moment.min.js' %}"></script>
    <script src="{% static 'vendor/datepicker/daterangepicker.js' %}"></script>
    <script src="{% static 'vendor/datepicker/checkbox.js' %}"></script>


    <!-- Main JS-->
    <script src="{% static 'js/global.js' %}"></script>

{% endblock js %}
```

Figura 17: Ejemplo de utilización de la plantilla base.html en Django

En la Figura 17 podemos ver un ejemplo de lo que sería un HTML utilizando el base.html. Como se puede observar, está separado por bloques predefinidos en el base.html en los cuales se puede añadir información extra para cada archivo.

4.3. Base de datos

En este apartado vamos a explicar cómo está estructurada nuestra base de datos. Veremos en primer lugar las tablas que genera automáticamente Django y posteriormente las que hemos creado nosotros teniendo en cuanta las claves y relaciones de estas.

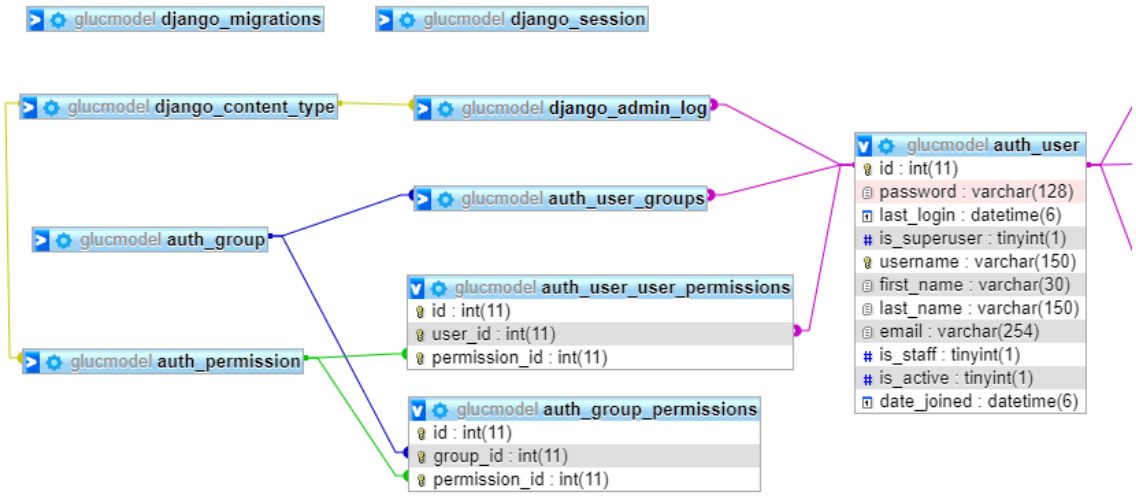


Figura 18: Diagrama de la base de datos generada automáticamente por Django

Como podemos ver en la Figura 18 se generan varias tablas que son esenciales para el buen funcionamiento del framework de aplicaciones web. Explicaremos a continuación las principales y de más utilidad. La tabla django_migrations [16] es la encargada de guardar todas las migraciones de la base de datos, cada creación de tabla, modificación de campo y eliminación de tabla se guarda en una migración y por cada migración se genera una fila en esta tabla. La tabla auth_user es la tabla principal en la cual se guarda toda la información de todos los usuarios de la web. El resto de las tablas son esenciales para el buen funcionamiento, pero no hemos tenido que hacer modificaciones en ellas.

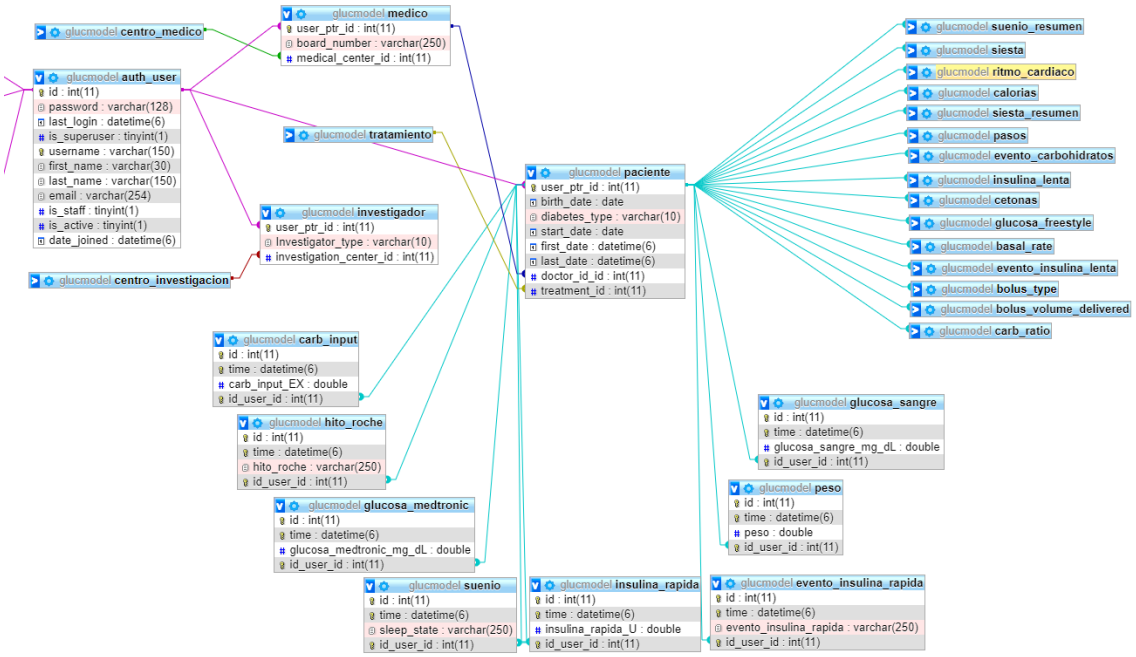


Figura 19: Diagrama de las tablas que hemos generado nosotros

En la Figura 19 podemos ver la estructura básica de nuestra base de datos. A la izquierda esta la tabla principal de usuarios de Django, esta está relacionada con los tipos de usuarios que hemos creado nosotros, médicos, investigadores y pacientes. Cada investigador y médico tienen asignado un centro de investigación o un centro médico dependiendo de qué tipo sea. Como podemos observar existe una relación entre el médico y los pacientes ya que se debe tener un seguimiento de qué médico trata a qué pacientes. En el caso de los investigadores no existen relación ya que tienen acceso total a los datos de todos los pacientes. Los pacientes están relacionados con un tratamiento y con un médico, estos además tienen una relación con cada una de las tablas de datos de los dispositivos de medición. Las tablas "centro_medico", "centro_investigacion" y "tratamiento" se han creado para tener uniformidad en los datos ya que se pueden añadir hospitales, universidades o nuevos tratamientos en un futuro.

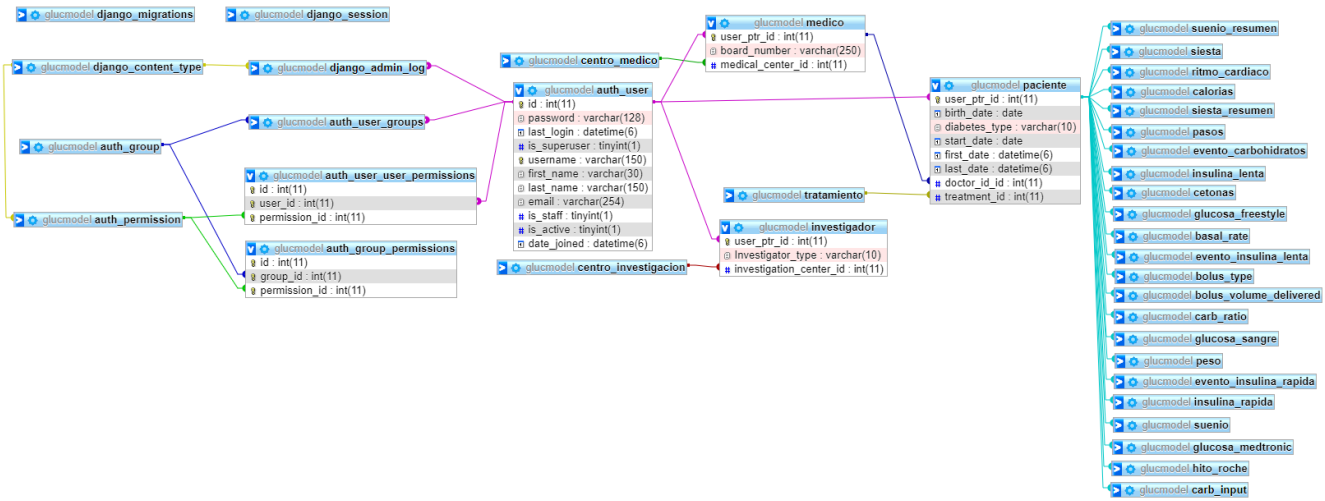


Figura 20: Tabla completa con algunas tablas minimizadas de la base de datos

En la Figura 20 podemos ver como queda la estructura final de la base de datos, juntando las tablas de Django y las nuestras.

4.4. Scripts de subida de datos

Para poder subir los archivos a la base de datos lo primero que hay que hacer es tratarlos y generar unos registros que se puedan subir a la base de datos y que sigan la lógica que se ha decidido implementar. Como se ha comentado anteriormente todos los datos deben tener una marca de tiempo común, en nuestro caso y para que los datos sean más fáciles de interpretar se tienen que modificar los datos para que los registros salgan cada 5 minutos partiendo siempre desde las horas en punto (ej. 10:00, 10:05, 10:10). Para conseguirlo hemos tenido que tratar los datos de forma diferente para cada caso de los expuestos a continuación. En algunos de los dispositivos se tienen que generar registros adicionales ya que hay datos de los cuales se requiere información de forma continua.

4.4.1. Free Style Libre- Abbott

El dispositivo Free Style Libre es un medidor que monitoriza de forma continua la glucosa intersticial.

Para los archivos que genera este dispositivo hay que hacer tres pasos principales:

1. Redondeo de la marca de tiempo a 5 minutos.
2. Juntar todas las filas que coinciden en esos periodos de 5 minutos haciendo las medias.
3. Calcular un solo dato de glucosa usando los datos de dos columnas, una que lee la glucosa automáticamente cada 15 minutos y otro que lee la glucosa cada vez que el usuario decide leerla.

```
def free_style_sensor(request,csv_file,usuario):
    try:
        gluc_data = pd.read_csv(csv_file, skiprows=1, sep=';', usecols=['Hora', 'Glucosa leída (mg/dL)', 'Histórico glucosa (mg/dL)', 'Insulina de acción rápida sin valor numérico',
                                                                        'Insulina de acción rápida (unidades)', 'Alimentos sin valor numérico', 'Carbohidratos (raciones)',
                                                                        'Insulina de acción lenta sin valor numérico',
                                                                        'Insulina de acción lenta (unidades)', 'Glucosa de la tira (mg/dL)', 'Cetonas (mmol/L)'])

        # Convert time to time series
        gluc_data['Hora'] = pd.to_datetime(gluc_data['Hora'])

        # Sort data by date
        gluc_data = gluc_data.sort_values(by=['Hora'])

        # Round data to the closest five minutes
        gluc_data['Hora'] = gluc_data['Hora'].dt.round('5min')
        gluc_data.set_index('Hora', inplace=True)

        # 5 minutes sampling
        gluc_data = gluc_data.resample('5min').mean()

        gluc_data_1 = gluc_data[["Glucosa leída (mg/dL)", "Histórico glucosa (mg/dL)"]].fillna(0)
        gluc_data_1 = gluc_data_1.assign(Glucosa_total=0).apply(juntar_glu, axis='columns')
        gluc_data["Glucosa_total"] = gluc_data_1["Glucosa_total"]
        gluc_data.pop("Glucosa leída (mg/dL)")
        gluc_data.pop("Histórico glucosa (mg/dL)")

        fecha_min = gluc_data.index.min()
        fecha_max = gluc_data.index.max()
```

Figura 21: Código en el cual tratamos los datos de un archivo de Free Style Libre

Como podemos observar en la Figura 21 lo primero que hacemos es leer el csv que se ha mandado a través del formulario de subida de datos de la página web. Lo siguiente que se hace es cambiarle el tipo al campo “hora” ya que viene en formato de String, justo debajo se ordena todo el DataFrame por ese mismo campo. Lo siguiente que se hace es hacer el redondeo de la hora para que todas las horas salgan en el formato de 5 minutos que queremos. Luego justo antes de juntar todas las filas que coincidan en la misma marca de tiempo definimos el campo “hora” como índice. Por último, con la ayuda de la función juntar_glu (Figura 22) generamos un campo llamado “Glucosa_total” que tiene, en el caso de que ambos datos estén presentes en la misma línea, la media de ambos.

```
def juntar_glu(r):
    # En esta funcion cogemos los datos de las columnas glucosa leida e historico de glucosa
    # En el caso en el que coincidan ambas en la misma fehca/hora (ej: 2017/02/02 10:05) se hace la media de ambas y se guarda
    # Si solo hay una de las dos nos quedamos con aquella que este informada
    # y si no hay ninguna devolvemos un np.nan
    # Este proceso se hace usando un apply que recorre todas las filas del DF, es decir que a la funcion le llegan filas (rows --> r)
    if (r["Glucosa leída (mg/dL)"] > 0) & (r["Histórico glucosa (mg/dL)"] > 0):
        r.Glucosa_total = (r["Glucosa leída (mg/dL)"] + r["Histórico glucosa (mg/dL)"])/2
    elif r["Glucosa leída (mg/dL)"] > 0:
        r.Glucosa_total = r["Glucosa leída (mg/dL)"]
    elif r["Histórico glucosa (mg/dL)"] > 0:
        r.Glucosa_total = r["Histórico glucosa (mg/dL)"]
    else:
        r.Glucosa_total = np.nan
    return r
```

Figura 22: Código de la función juntar_glu

La función representada en la Figura 22 recorre mediante un apply todas las filas del DataFrame haciendo comparaciones entre los dos campos deseados (“Glucosa leída (mg/dL”, “Histórico glucosa (mg/dL)”). En los casos en los que solo hay uno de los dos datos se queda con el dato que esté informado, si están los dos se crea la media y si no hay ninguno lo pone a nan.

```
for col in gluc_data.columns:
    col_csv = gluc_data[col].dropna()
    col_csv.to_csv('free.csv')  # crea csv con los resultados del script
    csv_file = open('free.csv', 'rb')  # importa datos del csv
    data_set = csv_file.read().decode('UTF-8')  # lee los datos
    csv_file.close()  # cierra el archivo para poder eliminarlo
    os.remove('free.csv')  # elimina el archivo
    io_string = io.StringIO(data_set)
    try:
        next(io_string)
        if col == "Insulina de acción rápida (unidades)":
            for column in csv.reader(io_string, delimiter=',', quotechar="|"):
                _, created = Insulina_rapida.objects.update_or_create(
                    time=column[0],
                    id_user_id=usuario,
                    defaults={"insulina_rapida_U": column[1], }
                )
        elif col == "Carbohidratos (raciones)":
```

Figura 23: Subida de los datos tratados de Free Style Libre a la base de datos

En la Figura 23 podemos ver como gestionamos la subida de información a la base de datos. Recorremos todas las filas del DataFrame y separamos en cada vuelta el DataFrame seleccionando la columna. Eliminamos todos los registros que no estén informados y procedemos a subirlos en la tabla que les corresponda.

4.4.2. Minimed Medtronic CGM y Medtronic Insulin Pump

Los dispositivos de Medtronic permiten recoger todos los datos de la bomba y el sensor de glucosa en un mismo archivo. El hecho de que vengan en un mismo archivo ambas tablas da lugar a complicaciones en el script de tratamiento de los datos. Estos archivos pueden venir de dos formas, o bien con la información de la bomba de insulina rellena o bien con la información de la bomba y del sensor de glucosa. Ambas tablas tienen su propia cabecera por lo que para detectar si hay datos habrá que tomar como referencia la segunda cabecera, esta está siempre presente haya o no datos.

Como podemos observar en la Figura 24 lo primero que hacemos es leer el csv completamente saltándonos las 6 primeras filas ya que no nos sirven de nada. Luego para encontrar la segunda cabecera buscamos que en la columna "Date" en vez de una fecha haya un String llamado "Date" que sería la columna de fecha correspondiente a la segunda tabla. Una vez obtenido el índice en el cual se encuentra la cabecera del sensor de glucosa comprobamos el número de índice con el tamaño del DataFrame. Si el tamaño del DataFrame es igual al índice + 1 significa que debajo de la segunda cabecera no hay nada, por lo que procederíamos a ejecutar el script que gestiona solo la primera tabla.

```
def medtronic(request, csv_file,usuario):
    # Obtain the row of data division if exists
    medtron_data = pd.read_csv(csv_file, skiprows=6, sep=';')
    medtron_data.to_csv('medtron.csv')#creo para poder leer mas abajo
    medtron_data.drop('Index', axis=1, inplace=True)
    str_index = medtron_data['Date'].str.find('Date').idxmax()

    if len(medtron_data) != (str_index + 1):
        # Load first data set
        medtron_data_1 = pd.read_csv('medtron.csv', engine='python',
```

Figura 24: Código con el cual decidimos si el archivo evaluado tiene las 2 tablas informadas

```
medtron_data_1 = pd.read_csv('medtron.csv', engine='python',
                             skipfooter=medtron_data.shape[0] - str_index + 3,
                             usecols=['Date', 'Time', 'BG Reading (mg/dL)', 'Basal Rate (U/h)',
                                      'Bolus Volume Delivered (U)', 'BWZ Carb Ratio (U/Ex)',
                                      'BWZ Carb Input (exchanges)', 'Sensor Calibration BG (mg/dL)'],
                             parse_dates={'Hora': ['Date', 'Time']}, dayfirst=True)
# Round data to the closest five minutes
medtron_data_1['Hora'] = medtron_data_1['Hora'].dt.round('5min')
# Convert time to time series and order DataFrame
medtron_data_1.set_index('Hora', inplace=True)
medtron_data_1.sort_index(inplace=True)
# Join equal data time only on a row
medtron_data_1 = medtron_data_1.groupby(level=0).sum(min_count=1)
# Delete repeated data
medtron_data_1.drop_duplicates(inplace=True)
# 5 minutes resampling
medtron_data_1 = medtron_data_1.resample('5T').mean()
# Add data to basal rate
medtron_data_1['Basal Rate (U/h)'] = medtron_data_1['Basal Rate (U/h)']/12
aux_br = medtron_data_1['Basal Rate (U/h)'][0]
for i in range(1, medtron_data_1.shape[0]):
    if np.isnan(medtron_data_1['Basal Rate (U/h)'][i]):
        medtron_data_1.iloc[i, medtron_data_1.columns.get_loc('Basal Rate (U/h)')] = aux_br
    else:
        aux_br = medtron_data_1.iloc[i, medtron_data_1.columns.get_loc('Basal Rate (U/h)')]
```

Figura 25: Código del tratamiento de los datos principales de la bomba de insulina Minimed de Medtronic

En la Figura 25 se ve el tratamiento de datos para la bomba de Medtronic sin tener en cuenta los bolos. Lo primero que se hace es la lectura del csv, leemos la primera cabecera y cogemos solo las columnas que nos interesan, además juntamos en un solo campo la fecha y la hora para poder trabajar mejor con ello más adelante. Luego al igual que en otros scripts, redondeamos los tiempos a 5 minutos. En este caso el ratio basal es necesario tenerlo siempre informado, por lo que generamos un bucle for que recorre todo el DataFrame, posteriormente sustituimos los ratios basales informados con un nan por el dato de ratio basal válido anterior.

```
bolus = pd.read_csv('medtron.csv', engine='python',
                    skipfooter=medtron_data.shape[0] - str_index + 3,
                    usecols=['Date', 'Time', 'Bolus Type'],
                    parse_dates={'Hora': ['Date', 'Time']}, dayfirst=True)
# Convert time to time series and order DataFrame
bolus['Hora'] = bolus['Hora'].dt.round('5T')
bolus.drop_duplicates(inplace=True)
mapping = {'Normal': 1, np.nan: 0}
bolus = bolus.replace({'Bolus Type': mapping})
bolus.set_index('Hora', inplace=True)
bolus.sort_index(inplace=True)
bolus = bolus.resample('5T').sum()
mapping = {0: np.nan, 1: 'Normal', 2: 'Normal', 3: 'Normal', 4: 'Normal', 5: 'Normal'}
bolus = bolus.replace({'Bolus Type': mapping})
# Merge bolus type and Medtronic data
medtron_data_1 = pd.merge(medtron_data_1, bolus, left_index=True, right_index=True, how='outer')
col = medtron_data_1['Bolus Type']
medtron_data_1.pop('Bolus Type')
medtron_data_1.insert(2, col.name, col)
```

Figura 26: Código del tratamiento de los bolos dentro de la bomba de insulina Minimed de Medtronic

En la Figura 26 vemos el tratamiento de los bolos. Lo único que hacemos es coger los datos de bolos y sustituirlos por 1 y 0 para poder juntar los datos que coincidan en la misma marca de tiempo. Lo último que se hace es juntarlos con los datos anteriores usando como índice de unión la fecha y la hora.

Lo último que se hace es tratar los datos del sensor de glucosa (Figura 27). Aquí leemos la segunda cabecera y solo seleccionamos la columna de glucosa. El único tratamiento de datos que se hace es redondear las horas a 5 minutos.

```
medtron_data_2 = pd.read_csv('medtron.csv', engine='python', skiprows=str_index + 1,
                             usecols=['Date', 'Time', 'Sensor Glucose (mg/dL)'],
                             parse_dates={'Hora': ['Date', 'Time']}, dayfirst=True)
medtron_data_2.set_index('Hora', inplace=True)
medtron_data_2.sort_index(inplace=True)
# 5 minutes resampling
medtron_data_2 = medtron_data_2.resample('5T').mean()
```

Figura 27: Código del tratamiento de los datos de la segunda tabla de la bomba de Minimed de Medtronic; estos datos corresponden al sensor de glucosa

Al final de esta parte obtenemos dos DataFrames con información, en el caso de no cumplir la condición del if solo gestionaría la parte de la bomba de insulina e igualaría el segundo DataFrame a None para así poder distinguir ambas casuísticas a la hora de subir los datos a la base de datos.

En la Figura 28 podemos ver como gestionamos la subida de información a la base de datos. Recorremos todas las filas del DataFrame y separamos en cada vuelta el DataFrame seleccionando la columna. Eliminamos todos los registros que no estén informados y procedemos a subirlos en la tabla que les corresponda.

```
for col in medtron_data_1.columns:
    col_csv = medtron_data_1[col].dropna()
    col_csv.to_csv('medtron.csv')  # crea csv con los resultados del script
    csv_file = open('medtron.csv', 'rb')  # importa datos del csv
    data_set = csv_file.read().decode('UTF-8')  # lee los datos
    csv_file.close()  # cierra el archivo para poder eliminarlo
    os.remove('medtron.csv')  # elimina el archivo
    io_string = io.StringIO(data_set)
    next(io_string)
    if col == "BG Reading (mg/dL)":
        for column in csv.reader(io_string, delimiter=',', quotechar="|"):
            _, created = Glucosa_sangre.objects.update_or_create(
                time=column[0],
                id_user_id=usuario,
                defaults={"glucosa_sangre_mg_dL": column[1], }
            )
    elif col == "Basal Rate (U/h)":
```

Figura 28: Parte del código que se encarga de subir los datos de la bomba de insulina de Medtronic a la base de datos

4.4.3. Roche Insulin Pump

Roche es una bomba de insulina que mide datos como la glucemia los tipos de bolos o la dosis basal. Los datos de esta bomba de insulina vienen en un solo archivo con la cabecera en la primera línea. Para tratar estos archivos tenemos que hacer algo muy similar a lo que se hace con los de Medtronic. Como tenemos datos de dosis basal tendremos que generar datos para todos los intervalos de tiempo con la dosis basal informada.

```
def roche(request, csv_file,usuario):
    try:
        roche = pd.read_csv(csv_file, sep=";", encoding="ISO-8859-1", usecols=["Fecha", "Hora", "Hito", "Glucemia",
                                                                               "Tipo de Bolo", "Unidades", "Dosis Basal (UI/H)",
                                                                               "Eventos S.I."],
                            parse_dates={'Time': ['Fecha', 'Hora']}, dayfirst=True)

        roche['Time'] = roche['Time'].dt.round('5min')

        roche.set_index('Time', inplace=True)

        roche.sort_index(inplace=True)

        roche.drop_duplicates(inplace=True)

        roche["Dosis Basal (UI/H)"].loc[roche["Eventos S.I."] == "Parada"] = 0

        roche_str = roche[["Hito", "Tipo de Bolo", "Eventos S.I."]]
        roche_int = roche[["Glucemia", "Unidades", "Dosis Basal (UI/H)"]]
        # Cuando el valor de "Eventos S.I." es Parada dosis basal a 0
        roche_int = roche.resample('5T').mean()

        roche = roche_int.merge(roche_str, left_index=True, right_index=True, how="outer")

        roche['Dosis Basal (UI/H)'] = roche['Dosis Basal (UI/H)'] / 12
        aux_br = roche['Dosis Basal (UI/H)'][0]
        for i in range(1, roche.shape[0]):
            if np.isnan(roche['Dosis Basal (UI/H)'][i]):
                roche.iloc[i, roche.columns.get_loc('Dosis Basal (UI/H)')] = aux_br
            else:
                aux_br = roche.iloc[i, roche.columns.get_loc('Dosis Basal (UI/H)')]
        roche.pop("Eventos S.I.")
```

Figura 29: Código del tratamiento de datos de la bomba de insulina de Roche

Como vemos en la Figura 29 a la hora de gestionar los datos de la bomba de insulina de Roche leemos solo las columnas que necesitamos, además juntamos ya la hora y el día para generar una columna llamada time que tenga ya esas dos variables como una sola. Luego ordenamos por la fecha y simplificamos todas las horas para que estén con el formato de 5 minutos que queremos. En este caso separamos el DataFrame en dos, por un lado, los datos que sean numéricos y por otro los datos que son Strings. Una vez separado hacemos que los datos repetidos se simplifiquen en una sola fila haciendo medias de estos. En este caso el ratio basal es necesario tenerlo siempre informado, por lo que generamos un bucle for que recorre todo el DataFrame, posteriormente sustituimos los ratios basales informados con un nan por el dato de ratio basal válido anterior.

```
for column in csv.reader(io_string, delimiter=',', quotechar="|"):
    if column[1] != "":
        _, created = Glucosa_sangre.objects.update_or_create(
            time=column[0],
            id_user_id=usuario,
            defaults={"glucosa_sangre_mg_dL": column[1], }
        )
    if column[2] != "":
```

Figura 30: Código de la subida de datos de la bomba de insulina de Roche

En este caso como podemos ver en la Figura 30 para la subida de datos lo que hacemos es recorrer todo el csv viendo fila a fila cada dato. Si el dato no está informado, es decir que tiene “” como dato no subimos nada, en caso contrario subimos los datos a la columna que corresponda.

4.4.4. Fitbit Ionic

Fitbit Ionic es un smartwatch Deportivo diseñado para tener un registro de calorías quemadas, pasos realizados, ritmo cardíaco y análisis del sueño.

4.4.4.1. Fitbit registro de calorías, pasos y ritmo cardiaco

Para la subida de este tipo de datos, actualmente debido al formato de los csv exportados del dispositivo Fitbit, se analiza el nombre del archivo para obtener la fecha del acontecimiento y añadirlo al DataFrame junto con los valores del csv para posteriormente realizar la subida a la base de datos.

Tras reunirnos con el tutor del trabajo, coincidimos en que sería mejor que el archivo exportado de Fitbit contase con una columna con la fecha del acontecimiento, para así no depender del formato del nombre del archivo y además poder cargar distintos días en un solo csv. Hemos dejado esta solución implementada para que funcione al realizar las mejoras en los archivos csv. De esta manera se puede realizar la subida de datos de ambas formas.

```
nom = csv_file.name.split('_')  # sacamos la fecha del nombre del archivo
nom = nom[3].split('.')
nom = nom[0]
cal_data = pd.read_csv("calorias.csv", skiprows=1, sep=',', names=['time', 'calories'])
cal_data['time'] = pd.to_datetime(nom + ' ' + cal_data['time'])
# Convert time to time series
cal_data.set_index('time', inplace=True)
# 5 minutes resampling
if(tipo_archivo == "FITBIT RITMO CARDÍACO"):
    cal_data = cal_data.resample('5T').mean()
else:
    cal_data = cal_data.resample('5T').sum()
```

Figura 31: Código en el que tratamos los datos de calorías, pasos o ritmo cardiaco

En la Figura 31 podemos observar como extraemos la fecha del nombre del archivo. El tratamiento de datos es muy sencillo simplemente leemos el csv, simplificamos las horas a 5 minutos y dependiendo del tipo de archivo juntamos las horas que coincidan haciendo la media o sumando. La subida de datos es muy sencilla, recorremos el csv y subimos los datos a las tablas que correspondan.

4.4.4.2. Fitbit registro de sueño nocturno y diurno

Para los datos del sueño, se exporta la información del csv a un DataFrame, en el cual se modifican los datos para que aparezcan cada 5 minutos con valor el estado de sueño de ese intervalo de tiempo. Una vez realizado los cambios, dependiendo de si se trata de sueño nocturno o diurno, se sube a la base de datos.

4.4.4.3. Fitbit registro de resumen de sueño nocturno y diurno

Para los datos de resumen de sueño diurno y nocturno, simplemente se exporta a un DataFrame y se realiza la subida a la base de datos sin realizar modificaciones de los datos.

4.5. Scripts de descarga de datos

En la descarga de datos se tiene en cuenta el tipo de usuario que haya iniciado sesión, es decir, si el usuario es un médico o investigador, se le ofrece una lista de pacientes para seleccionar al que desea realizar la consulta de datos. En caso de que el usuario sea un paciente, no se le ofrece ninguna lista de pacientes ya que solo puede consultar sus datos.

En el momento en que se selecciona un paciente de la lista, se realiza una búsqueda en la base de datos para consultar cuál es su primer y último registro y se muestra en un recuadro informativo con el rango de fechas obtenido para facilitar la búsqueda de los datos al usuario.

A continuación, se selecciona la fecha inicial y final para realizar la búsqueda en ese periodo de tiempo, se seleccionan los datos médicos que se quieren consultar y en la manera en que se desea visualizar: descarga en formato csv, en una tabla o en un gráfico.

Al seleccionarse una de estas opciones, se crea un DataFrame utilizando la librería pandas.

A continuación, en la Figura 32 mostramos el código generado para gestionar la creación del DataFrame tras seleccionar los campos en la página de descarga de datos.

```
for tabla in campos:
    items = eval(tabla).objects.filter(id_user_id=id_usuario,time__gte=first_date, time__lte=final_date)
    with open('items.csv', 'wb') as csv_file:
        write_csv(items, csv_file)
    df_aux = pd.read_csv("items.csv")
    df_aux = df_aux.drop(columns=["ID"])
    df = df.merge(df_aux, on=['time','id_user_id'], how='outer')
    csv_file.close()  # cierra el archivo calorias para poder eliminarlo
    os.remove('items.csv')
df = df.set_index("time", drop=True)
df = df.sort_index()
```

Figura 32: Código en el cual generamos un DataFrame que posteriormente podremos mostrar o descargar

La idea principal en este código es ir haciendo consultas a la base de datos con todos los campos que se han seleccionado con los check boxes de la página de descarga, a medida que vamos haciendo iteraciones en el bucle vamos juntando los datos en un mismo DataFrame. Los campos seleccionados llegan en forma de lista (“campos”), para cada iteración la guardamos en una variable que llamamos “tabla”. Evaluamos la variable “tabla” para que tenga las propiedades correctas del objeto y no las de un String y generamos la consulta pasándole por parámetro las fechas objetivo. Una vez obtenidos los datos de la consulta los introducimos en un DataFrame que hemos creado anteriormente; la forma de juntarlos es mediante un merge con la opción how igual a outer (outer join) usando como variables de cruce el id del paciente y la fecha y hora. De esta manera conseguimos que el DF vaya haciéndose más grande sin perder ninguna fila de información y consiguiendo uniformidad en la clave de los datos. Por último y una vez recorrido todas las tablas seleccionamos como índice la fecha y hora y la ordenamos para que la tabla, grafico o csv sean más fáciles de interpretar.

5. Resultados

Tras realizar la subida de datos de los pacientes, los médicos y los investigadores pueden consultar los resultados de cualquier paciente, a diferencia de los pacientes, que solo pueden consultar sus datos. Seleccionando un paciente, marcando los datos que se quieren consultar y eligiendo el rango de fechas de los datos, se pueden visualizar los datos de tres maneras: descarga de archivo csv, una tabla o un gráfico, como se aprecia en la Figura 33. A continuación, se explica con detalle estas tres maneras.

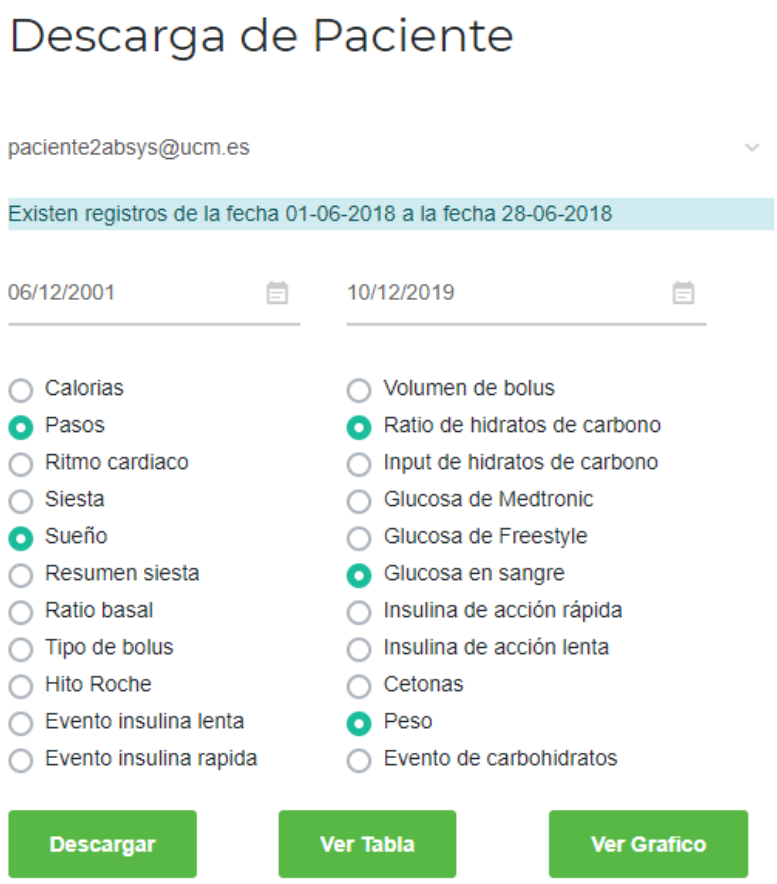


Figura 33: Página download.html

5.1. Descarga en CSV

Si se selecciona el botón Descargar, se crea un archivo .csv con la información del DataFrame creado anteriormente y se realiza una descarga de este archivo con nombre: el id del paciente y el nombre del usuario (Figura 35). Este archivo contiene toda la información de los datos deseados del usuario, cuenta con una cabecera con el nombre de los datos y a continuación la información ordenada por fecha y separado por comas como se puede apreciar en la Figura 34.

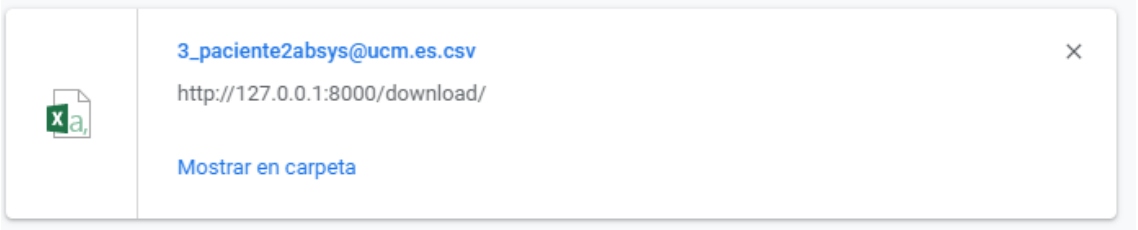


Figura 35: Ejemplo de archivo de descarga

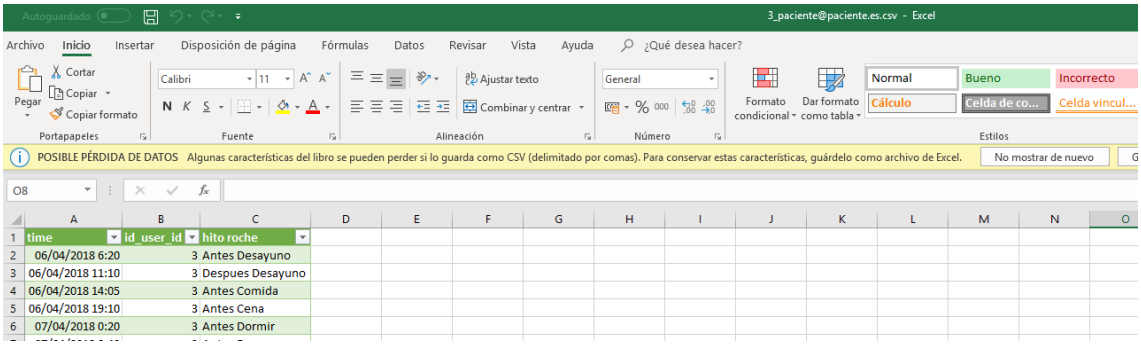


Figura 34: Ejemplo de archivo descargado y mostrado en excel

5.2. Tabla online de datos

Si se selecciona el botón ver datos, se crea una tabla con los campos seleccionados utilizando el DataFrame con la función to_html de la librería pandas. Se escribe un HTML en ese momento y se muestra como aparece en la Figura 36.

time	id_user_id	calories	heart rate	glucosa medtronic mg dL
2018-06-15T00:00:00	3	4.830000	91.357798	151.0
2018-06-15T00:05:00	3	7.325500	93.924242	159.0
2018-06-15T00:10:00	3	4.347000	94.159292	169.0
2018-06-15T00:15:00	3	4.427500	91.677966	174.0
2018-06-15T00:20:00	3	4.347000	92.899160	176.0
2018-06-15T00:25:00	3	4.427500	95.516393	186.0
2018-06-15T00:30:00	3	5.071500	95.943089	198.0
2018-06-15T00:35:00	3	4.025000	96.015385	209.0
2018-06-15T00:40:00	3	4.186000	95.948718	218.0
2018-06-15T00:45:00	3	4.025000	95.068966	228.0
2018-06-15T00:50:00	3	4.025000	97.526718	237.0
2018-06-15T00:55:00	3	4.025000	100.716312	245.0
2018-06-15T01:00:00	3	4.186000	101.529412	250.0

Figura 36: Ejemplo de tabla mostrada en la página web

5.3. Gráfico online de datos

En el caso de seleccionar el botón ver gráfico, utilizando la librería matplotlib se crea un gráfico con los campos seleccionados, se convierte en imagen y se muestra en un HTML que se escribe en ese momento. Es recomendable visualizar los datos de los gráficos por separado para

tener una mayor claridad, ya que consultar muchos datos a la vez puede generar un gráfico poco legible como se puede apreciar en la Figura 37.

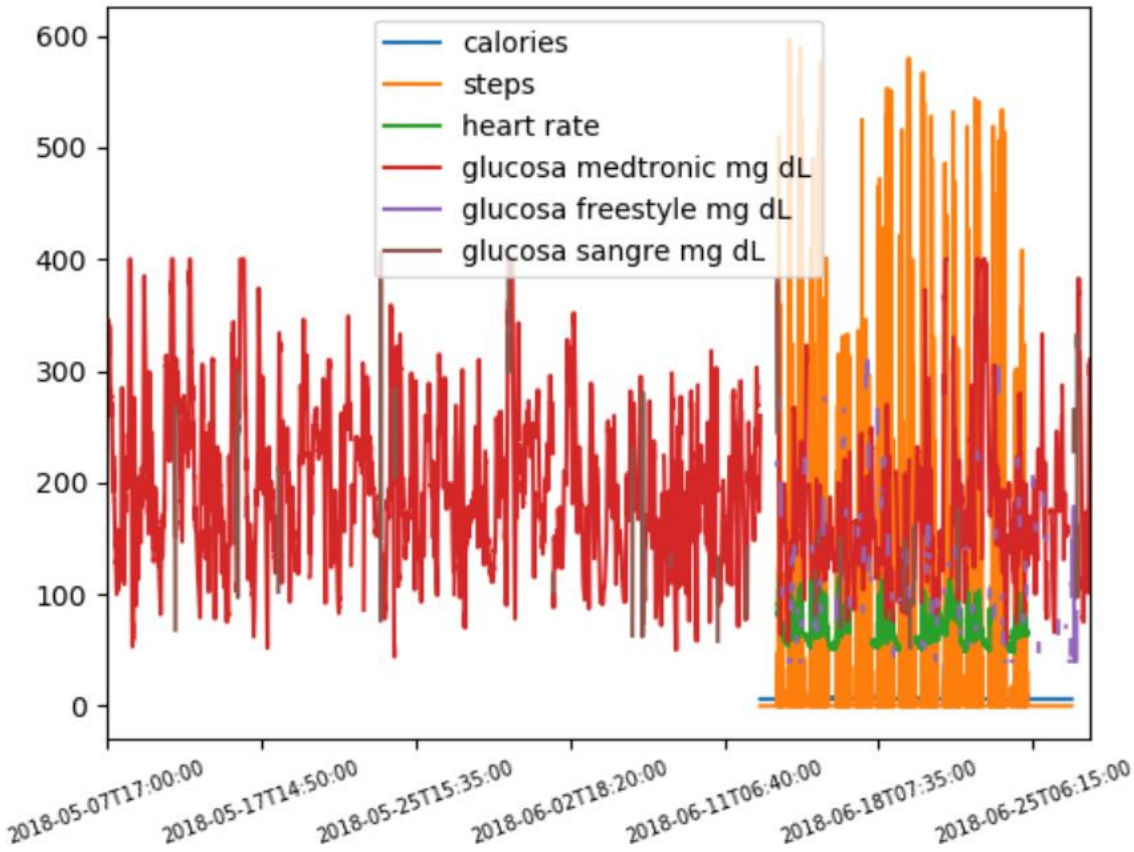


Figura 37: Ejemplo de posible grafico que puede dar lugar a confusiones

En el caso de seleccionar un solo dato para ver en el gráfico, queda mucho más claro para poder obtener conclusiones rápidas viendo brevemente el gráfico, como se ve puede ver en la Figura 38.

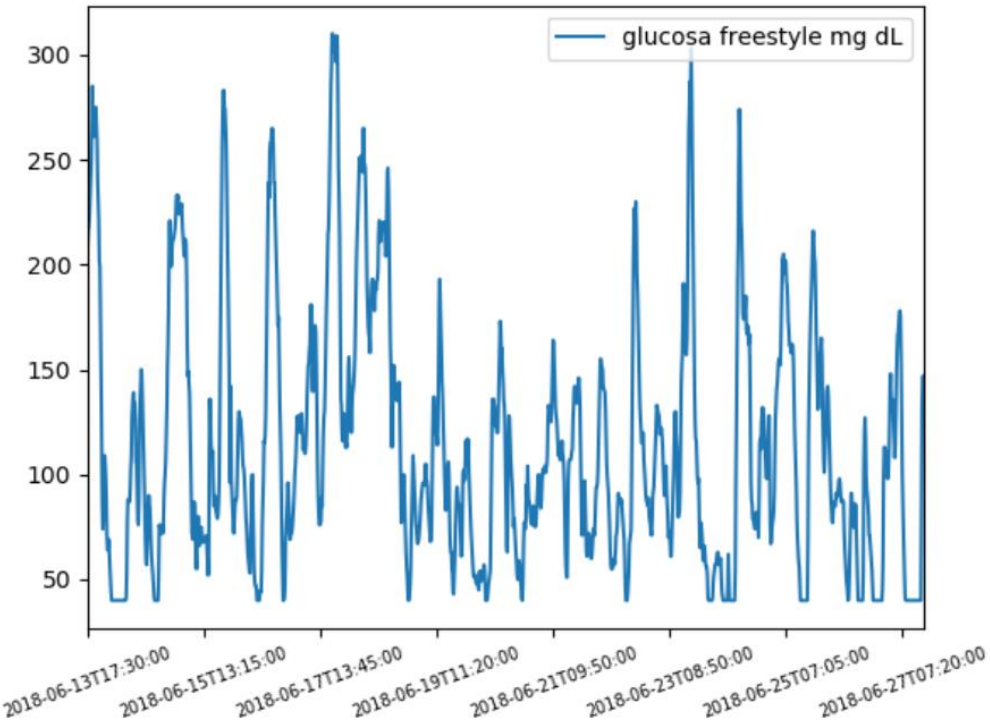


Figura 38: Ejemplo de un gráfico comprensible

5.4. Registro de Pacientes

Los pacientes son registrados únicamente por los médicos pulsando sobre la pestaña Añadir Paciente. El médico asigna al paciente un nombre de usuario (su correo electrónico), su fecha de nacimiento, la fecha en la que se comenzó a tratar su enfermedad, el tipo de diabetes (Tipo 1, Tipo 2 u Otro), el tipo de tratamiento que está recibiendo, una contraseña con un mínimo de seis caracteres y no obligatoriamente el peso. A continuación, en la Figura 39 mostramos un ejemplo del registro de un paciente.

Figura 39: Formulario de registro de paciente

Tras registrar al usuario se le asigna el número de colegiado del médico que le ha registrado y se le envía un correo informándole de que ha sido dado de alta en Glucmodel, indicándole su usuario y contraseña, que puede ser cambiada desde la aplicación. Como se muestra en la Figura 40.

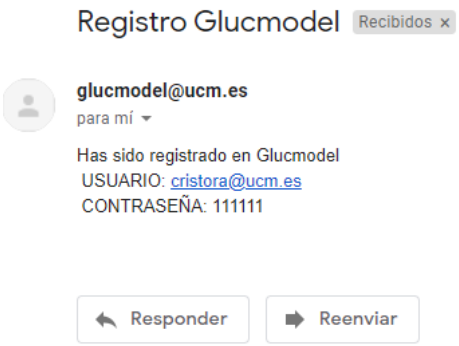


Figura 40: Ejemplo de correo electrónico enviado tras ser registrado en la página

6. Conclusiones/Conclusions

6.1. Conclusiones

Una vez finalizado el trabajo presentado en esta memoria podemos afirmar que se han cumplido los objetivos marcados.

La aplicación web que hemos creado va a ser de gran ayuda para que las personas que padecen de diabetes tengan más facilidad a la hora de tratar su enfermedad, tanto en la comunicación con el médico como para visualizar sus datos de una manera sencilla y clara. También creemos que el programa ayudará a los médicos a poder desarrollar su trabajo de una forma más cómoda, lo que acaba beneficiando a los pacientes.

Los pacientes pueden iniciar sesión, subir sus resultados y consultarlos sin problemas. Los médicos pueden registrar a pacientes cómodamente y subir datos de cualquier paciente registrado en la aplicación. Todos los usuarios pueden descargar o ver en una tabla o en un gráfico los datos sin necesidad de tener conocimientos informáticos. La única diferencia es que los pacientes solo pueden ver sus datos y los médicos e investigadores los de todos los pacientes.

Hemos sido capaces de adaptarnos a las peticiones y modificaciones que se nos han pedido tras las reuniones con el tutor del proyecto, que siempre aportaba ideas y consejos que han sido fundamentales para el buen resultado del trabajo.

En lo personal, gracias al Trabajo de Fin de Grado hemos aprendido a programar en el lenguaje Python y a utilizar librerías como pandas y matplotlib que nos han resultado muy útiles a la hora de tratar los resultados y mostrarlos tanto en tablas como en gráficos. No teníamos ningún conocimiento previo de este lenguaje y como muestran los últimos estudios, va a ser uno de los lenguaje más usados en los próximos años [17][18], lo que ha significado una motivación más para realizar un buen trabajo. Además, hemos aprendido a utilizar Django, una herramienta que nos ha parecido muy cómoda para desarrollar la aplicación de una manera diferente a lo que estábamos acostumbrados y sin ninguna duda más eficaz. También hemos mejorado nuestros conocimientos en HTML y bases de datos, ampliando el conocimiento obtenido en asignaturas como Aplicaciones Web, Bases de Datos, Tecnologías de la Programación, etc.

Como hemos citado anteriormente, el mayor orgullo es sentir que este trabajo puede ser de ayuda para otras personas, tanto para mejorar la calidad de vida de los que padecen de diabetes, como para facilitar el trabajo a los médicos e investigadores que hacen una labor tan importante.

6.2. Conclusions

Once finished the project presented in this report, we can affirm that the assigned objectives have been fulfilled.

The web application we have created is going to be very helpful so that any person that suffers from diabetes has more ease in the treatment of their illness, both in communicating with the doctor and in visualizing their data in an easy and clear way. We also believe that the program is going to help doctors develop their work in a more comfortable way, which ends up benefiting patients.

Patients can log in, upload their results and consult them without any issues. Doctors can comfortably register patients and upload data from any of those who are registered in the application. All users can download or visualize data in a table or graphic without any computer knowledge. The only difference is that patients can only see their data and doctors and researchers can see information of all patients.

We were able to adapt to the project's request's and change's after meetings with our tutor. He always contributed with ideas and tips that have been fundamental for the completion of the project.

Personally, thanks to the final degree project we have learned to program in python and to use libraries such as pandas and matplotlib that have resulted to be very useful in data treatment and data visualization in both, tables and graphics. We did not have any knowledge of this programming language and as shown in the latest studies it's going to become one the most popular programming languages in the coming years [17][18], this has been an added motivation to develop a good work. Also, we have learned to use Django, a tool that we have found very useful to develop a web application in a different, and without a doubt, more efficient way. On top of that, we have increased our knowledge in HTML and data bases, expanding the acquaintance acquired in lectures such as Web Applications, Data Bases, Programming Technologies etc.

As we have previously quoted, the greatest pride is feeling that this project can be helpful to other people, both to improve the living conditions of people suffering from diabetes and to facilitate the indispensable work that doctors and investigators do.

7. Aportaciones

En este apartado se encuentran las aportaciones de cada uno de los participantes del proyecto.

7.1. Aportaciones de Cristóbal Ramos Laina

Para poder empezar a trabajar en el proyecto de una manera efectiva, lo primero que hice fue crear un git en común con mi compañero para poder realizar cambios en el proyecto de una manera cómoda y sobre todo, sin estropear el trabajo realizado del otro. Como se puede observar en Figura 41, cuenta con 95 commits, lo que quiere decir que ha sido de gran utilidad.

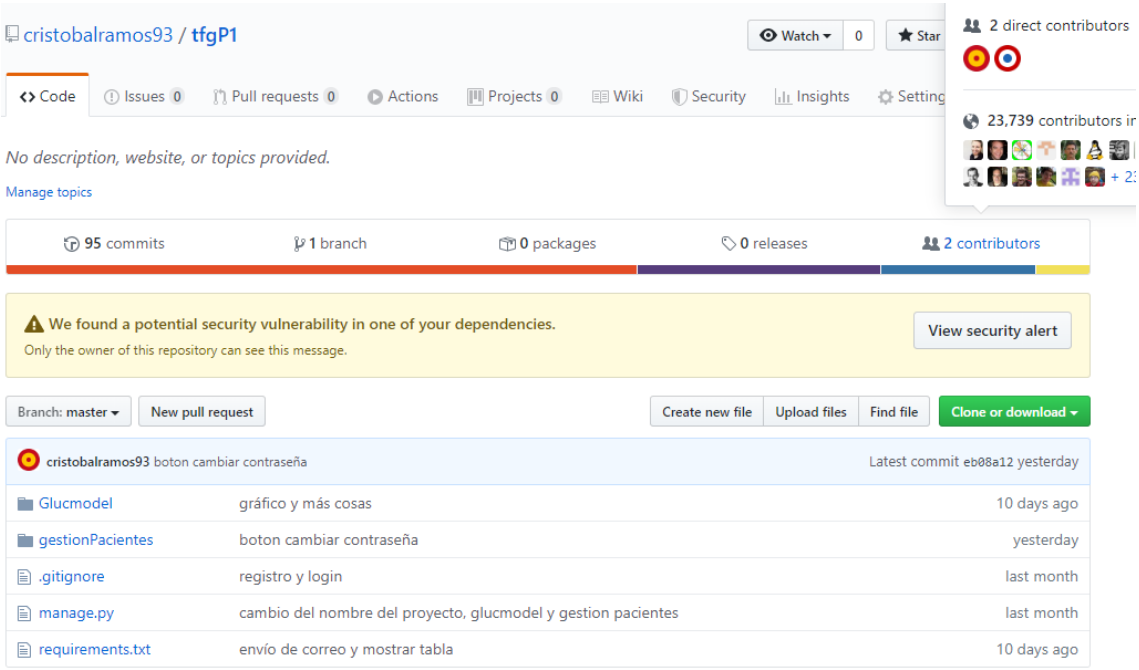


Figura 41: Github del proyecto

Una vez creado el Github, ya podíamos empezar a desarrollar el proyecto y lo primero fue buscar toda la información posible sobre el framework de aplicaciones web que encontró mi compañero, ya que no contábamos con ningún conocimiento previo de Django y parece algo complejo al principio.

Investigando sobre este funcionamiento, empecé creando las cosas más sencillas, como es la navegación de la aplicación web, es decir, la conexión de las urls del archivo urls.py con las funciones del archivo views.py.

Al crear la navegación ya se pueden crear funciones en relación con un HTML, con lo que creé una página de bienvenida muy simple, una página de registro y una de inicio de sesión con sus respectivas funciones en el archivo views.py para poder empezar a realizar pruebas reales.

Sin ninguna duda, lo que más trabajo ha supuesto a la hora de desarrollar el proyecto, ha sido gestionar la interacción entre los formularios de bajada y subida de datos y las funciones

del archivo views.py en el caso en el que el usuario fuese un médico o un investigador, ya que a la hora de seleccionarse un paciente de la lista, se tenía que cargar un recuadro informativo con la fecha del primer y último registro del paciente y esto ha supuesto que se cargue información dinámicamente sobre las consultas de la base de datos sin que resulte incómodo para el usuario.

Una parte fundamental ha sido la creación de las tablas y los gráficos, para conseguir su funcionamiento, gran parte del tiempo lo he dedicado al estudio de la librería matplotlib, encargada de realizar los gráficos. La complejidad de crear las tablas fue desarrollar un algoritmo que escribiese un HTML cada vez que se ejecutase la acción de ver tabla. Como se puede observar en Figura 42 al seleccionar la opción ver, se guarda en un String el código HTML necesario para crear una página junto con un conversor de un DataFrame a HTML que luego es escrito en el archivo HTML llamado table.html y redireccionado a este para mostrarlo.

```
if 'ver' in request.POST:
    disenio = '{% extends "base.html" %}\n'
    disenio += "{% load staticfiles i18n %}\n"
    disenio += '{% block css %}\n'
    disenio += '<link href="{% static "gestionPacientes/static/css/freelancer.css" %}" rel="stylesheet" type="text/css">\n'
    disenio += '{% endblock css %}\n'
    disenio += "{% block title %}Tabla{% endblock title %}\n"
    disenio += "{% block content %}\n"
    tabla = df.to_html(classes='table table-striped table-hover')
    disenio += tabla
    disenio += "\n"
    disenio += "{% endblock content %}"
    with open("gestionPacientes/plantillas/table.html", "w") as file:
        file.write(disenio)
    return render(request,"table.html")
```

Figura 42: Algoritmo creación de tabla

Con todas las funcionalidades hechas en el proyecto, llegó el momento de realizar los ajustes estéticos para que la aplicación web diese una buena impresión a primera vista. Para ello incluí el CSS y los ajustes para que todo quedase bien cuadrado.

7.2. Aportaciones de Jaime Viejo Martínez

Como hemos mencionado anteriormente una de las herramientas utilizadas en este proyecta ha sido el Trello. Esta es una herramienta que gestiona las tareas de un proyecto mediante un tablón de tareas online. Uno de mis cometidos en este proyecto ha sido el de crearlo y mantenerlo a lo largo de los meses de desarrollo del proyecto ya sea añadiendo tareas o actualizando las que ya estaban. Como se puede ver en la Figura 43 el tablón está dividido en 4 apartados: información, to do, doing y completado. Como era de esperar prácticamente todas las tareas están en la columna completado.

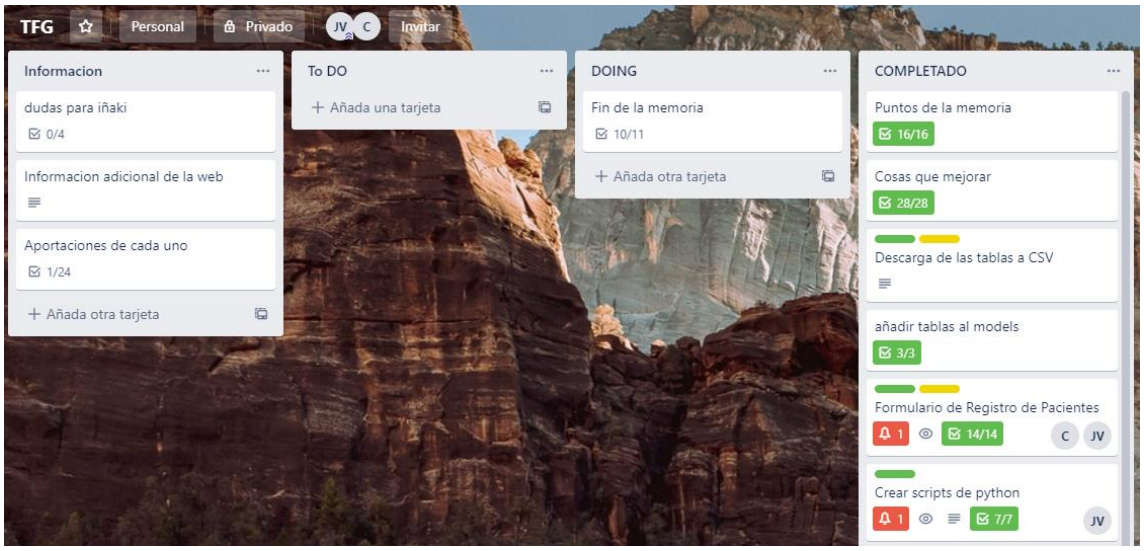


Figura 43: Tabla de trello que hemos usado para organizarnos a lo largo del proyecto

Una de las partes más importantes en este proyecto es el uso de la librería pandas para gestionar los datos. Sin esta librería la gestión de datos no sería posible por ello yo me encargué de hacer una investigación inicial sobre la misma haciendo cursos online como por ejemplo el de kaggle [19].

Como se puede observar en la memoria el framework de aplicaciones web es un pilar indispensable en nuestra aplicación. Uno de mis compromisos con este proyecto fue el de investigar posibles formas de llevar a cabo la creación de la aplicación tal y como nos la planteó el tutor del trabajo de fin de grado. Tras preguntar a varios compañeros e investigar online descubrí que los frameworks podían sernos de gran ayuda.

Los datos son la razón de ser de este proyecto y por ende esenciales para el mismo. Para poder hacer un buen análisis y tratamiento de los datos es necesario tener muy claro cuál es el estado de los estos cuando nos llegan a la aplicación. Hay que tener en cuenta principalmente las columnas, los tipos de los datos y los separadores que utilizan los archivos.

Para que la aplicación funcione de forma adecuada es esencial tener una buena estructura en la base de datos. Esto en Django se hace mediante la creación de modelos en el archivo de models.py. Las primeras tablas que se crearon fueron las de los tipos de usuarios las cuales las creamos juntos. Las siguientes tablan son todas tablas relacionadas con los datos que gestionamos.

```python
class Glucosa_medtronic(models.Model):
    id_user = models.ForeignKey(Paciente, on_delete=models.PROTECT)
    time = models.DateTimeField(null=False, blank=False)
    glucosa_medtronic_mg_dL = models.FloatField(max_length=250)

    class Meta:
        db_table = 'glucosa_medtronic'
        unique_together = (('id_user', 'time'),)
```

Figura 44: Estructura de la tabla de glucosa de Medtronic

En la Figura 44 se puede observar la estructura básica de una tabla que guarda datos dentro de nuestra base de datos. Lo principal es la última línea la sentencia unique_together indica que dentro de esa tabla los valores id_user y time son consideramos como una clave compuesta única. Esto significa que no puede haber dos valores con esa misma clave, de esta manera evitamos tener datos repetidos en nuestra base de datos.

Para poder gestionar los tipos de usuarios decidimos crear una jerarquía para ellos. En la parte más alta de la pirámide está el administrador seguido por médicos e investigadores y en la cola los pacientes. En esta aplicación web los usuarios no pueden crearse sus propias cuentas por lo que alguien de un nivel superior tiene que encargarse de darles de alta. En el caso del administrador puede dar de alta a cualquier usuario. Para facilitar su labor una de las cosas que he creado son unos formularios de creación y edición de médicos, investigadores y pacientes. Estos formularios están en el panel de administración de Django, al que se accede añadiendo /admin al final de la url raíz del proyecto.

Para facilitar la creación de páginas dentro del entorno Django una de las claves es la plantilla html base.html. Esta te permite de una forma muy sencilla reutilizar código. Con el uso de esta plantilla la creación de páginas en nuestro proyecto se hace de forma fácil y rápida. Uno de mis cometidos ha sido la creación de esta plantilla base.

Para obtener la descarga de datos en csv o la visualización de datos tanto en tabla como en gráfico es necesario descargarnos datos de la base datos. Para ellos usamos un script que recoge todos los datos de la consulta proporcionada por el usuario para crear un DataFrame. Yo me he encargado de gestionar la creación de ese DataFrame. Recogemos los datos que nos van sacando las consultas a la base de datos en un bucle for y para cada iteración juntamos los datos con los de las iteraciones anteriores.

Otra de las funcionalidades esenciales es la subida de los datos. Ya que cada fuente de datos tiene unas características únicas el tratamiento de los datos a través de los scripts de subida de datos tiene que ser único en la mayoría de los casos. Yo me he encargado de gestionar la implementación de los scripts de subida de la mayoría de las fuentes de datos de la aplicación. Casi todas las fuentes tienen unas cabeceras y número de datos distintos.

Uno de los objetivos que se querían conseguir con la creación de esta aplicación es el de facilitar el acceso a los datos a los investigadores para que puedan alimentar sus algoritmos con la mayor cantidad de datos posible. Para ello es imprescindible facilitar la descarga de los datos en un formato que ellos puedan utilizar. En nuestra aplicación permitimos la descarga de datos en archivos de tipo csv. Esta funcionalidad en concreto ha sido mi labor.

8. Referencias

[1] Hidalgo, J. I., Colmenar, J. M., Kronberger, G., Winkler, S. M., Garnica, O., & Lanchares, J. (2017). Data based prediction of blood glucose concentrations using evolutionary methods. *Journal of medical systems*, *41*(9), 142.Asda

[2] Sparacino, G., Zanderigo, F., Corazza, S., Maran, A., Facchinetti, A., and Cobelli, C., Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series. IEEE Trans. Biomed. Eng. 54(5):931–937, 2007.

[3] https://www.who.int/es/news-room/fact-sheets/detail/diabetes

[4] http://absys.dacya.ucm.es/doku.php

[5] J. Van de Bulck, Sleep apps and the quantified self: blessing or curse?, Journal of sleep research 24 (2015) 121-123

[6] K.G. Baron, J Duffecy, M.A. Berendsen, I.C. Mason, E. G. Lattie, N.C. Manalo, Feeling validated yet? A scoping review of the use of consumertargeted wearable and mobile technology to measure and improve sleep, Sleep medicine reviews 40 (2018) 151-159

[7] http://www.freestylediabetes.es

[8] https://www.fitbit.com/es/ionic

[9] https://djangostars.com/blog/why-we-use-django-framework/

[10] https://docs.djangoproject.com/en/3.0/topics/security/

[11] https://tutorial.djangogirls.org/es/template_extending/

[12]https://www.jetbrains.com/help/pycharm/manage-projects-hosted-on-github.html

[13]https://dev.to/mokkapps/why-i-switched-from-visual-studio-code-to-jetbrains-webstorm-939

[14]https://docs.djangoproject.com/en/3.0/topics/db/models/

[15]https://docs.djangoproject.com/en/3.0/ref/contrib/admin/

[16]https://docs.djangoproject.com/en/3.0/topics/migrations/

[17] https://medium.com/datadriveninvestor/7-top-programming-languages-to-look-forward-in-2020-514c25da2ce1

[18] https://iglu.net/programming-languages-future/

[19]https://www.kaggle.com/learn/pandas

Apéndice: Manual de usuario

El código fuente y todo los archivos necesarios están disponibles en el Google drive proporcionado por la facultad de informática.

Lo principal para que la aplicación funcione, es instalar Python 3, los paquetes y las librerías. Dichos paquetes se encuentran en el archivo requirements.txt que está en la carpeta raíz del proyecto. Este archivo tiene los nombres y las versiones de los paquetes utilizados.

Lo primero que debes hacer como administrador de la página es acceder al menú de administración de Django introduciendo al final de la url de la página */admin* e introducir su usuario y contraseña como se aprecia en la Figura 45. En este caso las credenciales serían las siguientes:

Username: admin

Password: admin

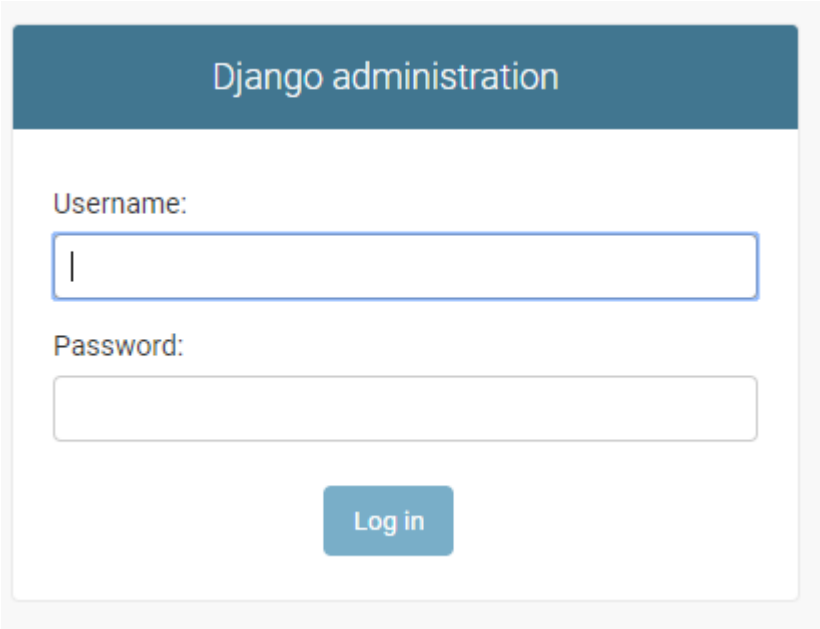


Figura 45: Login administrador

Habiendo iniciado sesión podrá dar de alta a médicos, investigadores, centros médicos, centros de investigación y tipos de tratamiento.

Figura 46: Menú de administración

Para ello solo hay que clicar en el botón *+Add* en la fila de lo que se quiere añadir y rellenar los campos solicitados. En el caso de los médicos no es necesario crear centros médicos con antelación, ya que se pueden crear en ese momento pulsando en el símbolo *+* como se aprecia en la Figura 47

Figura 47: Posibilidad de añadir centro médico

Ocurre lo mismo para los investigadores, en el caso de registrar un investigador, y no existir ningún centro de investigación creado o querer seleccionar uno nuevo, solo hay que presionar sobre el botón *+*.

A continuación, vamos a explicar todas las acciones posibles dentro de la aplicación web seas el usuario que seas:

Las credenciales de los usuarios creados en la base de datos proporcionada son los siguientes:

Usuario Administrador: admin | Contraseña Administrador: admin

Usuario Paciente 1: paciente1@ucm.es | Contraseña Paciente 1: 1234567

Usuario Paciente 2: paciente2@ucm.es | Contraseña Paciente 2: 1234567

Usuario Medico: medico | Contraseña Medico: 1234567

Usuario Investigador: investigador | Contraseña Investigador: 1234567

Los pacientes, tras recibir el correo electrónico con su usuario y contraseña, en caso de querer iniciar sesión, debe dirigirse a la dirección del sitio web (en el caso de estar en local la dirección local que aparezca en la consola ej. http://127.0.0.1:8000/) e insertar su usuario y contraseña como se muestra en la Figura 48

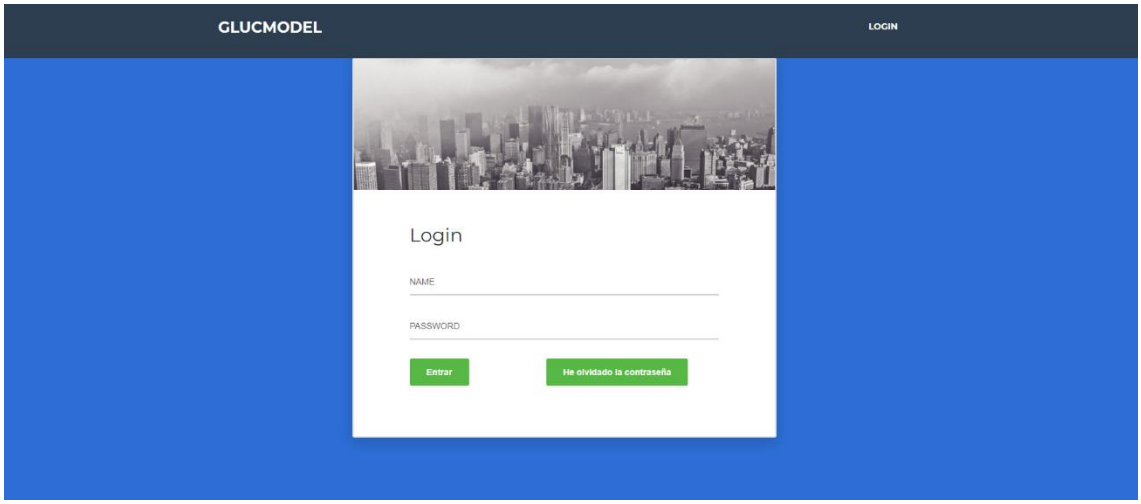


Figura 48: Inicio de sesión

En caso de olvido de la contraseña, pulsando sobre el botón *He olvidado la contraseña* en el login, son informados de que escriban un correo a la dirección glucmodel@ucm.es como se muestra en la Figura 49

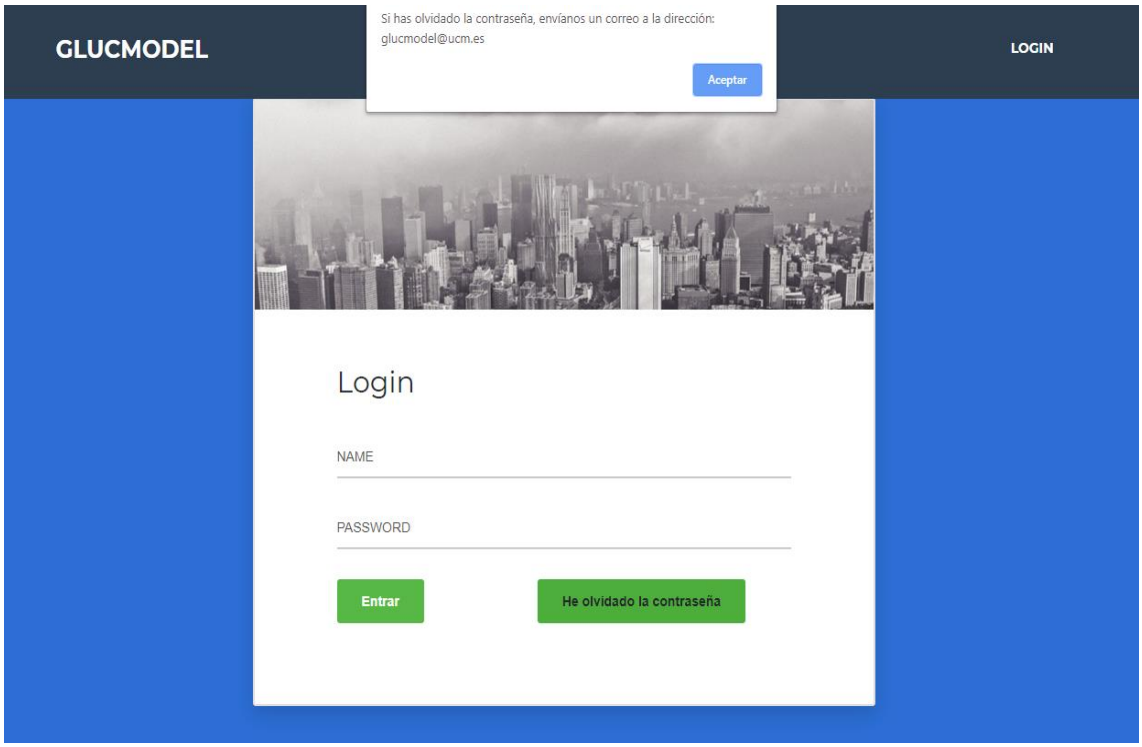


Figura 49: He olvidado mi contraseña

Una vez habiendo iniciado la sesión, son dirigidos a la página principal, en el caso de que el usuario sea un paciente, esta página tiene el aspecto de la Figura 50.

Figura 50: Página principal de los pacientes

En el caso en el que el paciente quisiese cambiar su contraseña solo tendría que clicar en el botón de Cambiar Contraseña que se encuentra en la Figura 51.

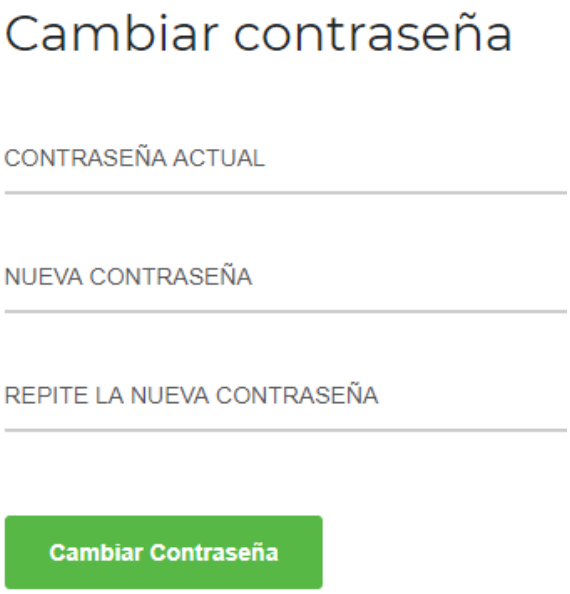


Figura 51: Ejemplo cambio de contraseña

En el caso de que el usuario sea un médico, la página tendrá otro aspecto como se muestra en la Figura 52.

Figura 52: Página principal de los médicos

En el caso de que el usuario sea un investigador, la página tiene el aspecto que se ve en la Figura 53

Figura 53: Página principal de los investigadores

Como se puede observar en la Figura 52, el médico es el único usuario que tiene opción de añadir a un paciente, en el caso de clicar sobre esta pestaña, será dirigido a la página de registro de un paciente que se aprecia como en la Figura 54.

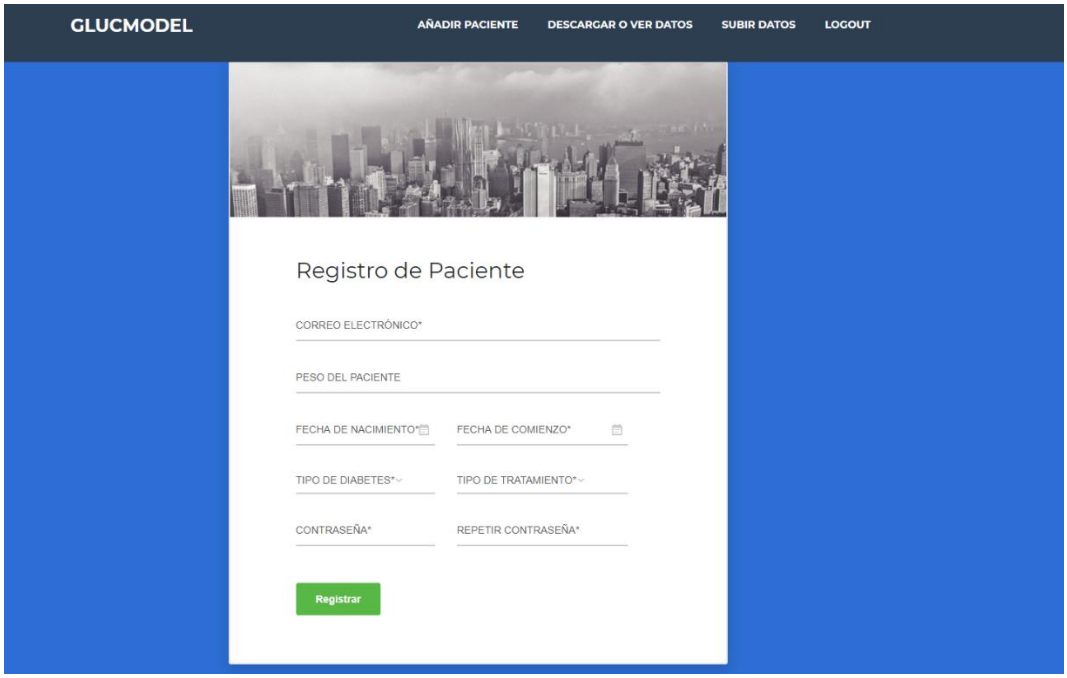


Figura 54: Página de registro de un paciente de los médicos

Para descargar o ver datos no existen diferencias en las interfaces de los médicos y de los investigadores, es decir, es la misma página como se aprecia en la Figura 55, sin embargo, el paciente sí que tiene una pequeña variación, no tiene la opción de elegir un paciente ya que solo puede descargar o ver datos de sí mismo como se ve en la Figura 56.

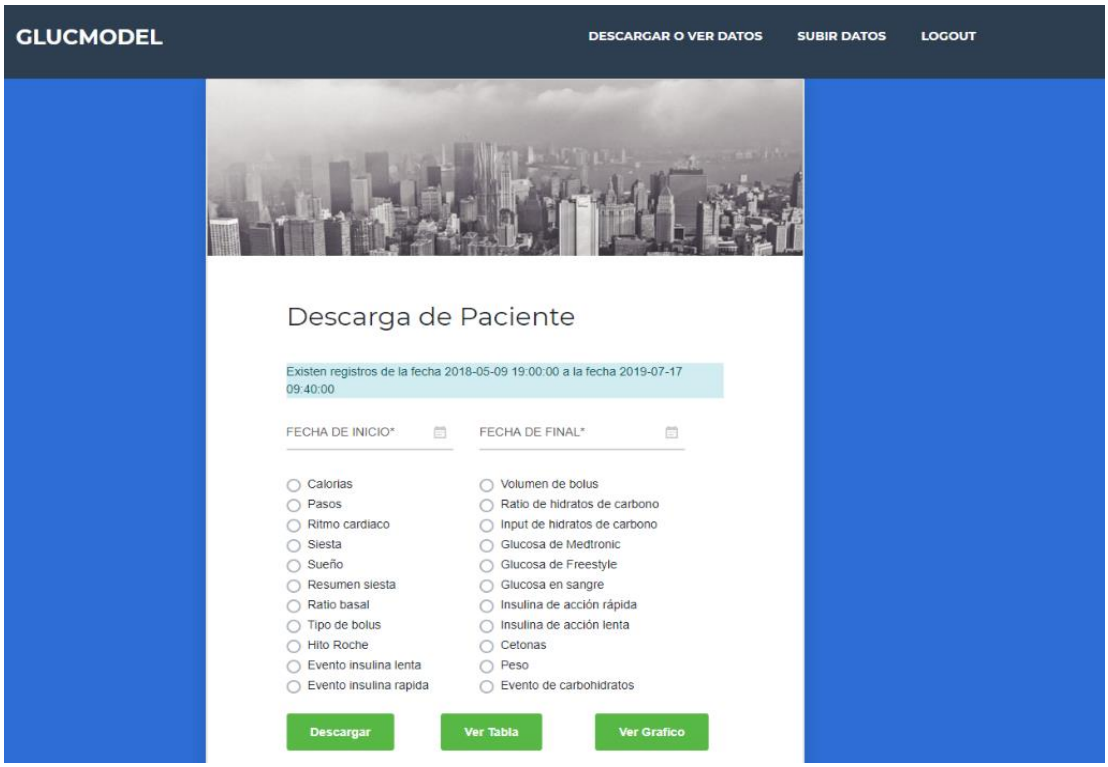


Figura 55: Página descargar o ver datos para médicos e investigadores

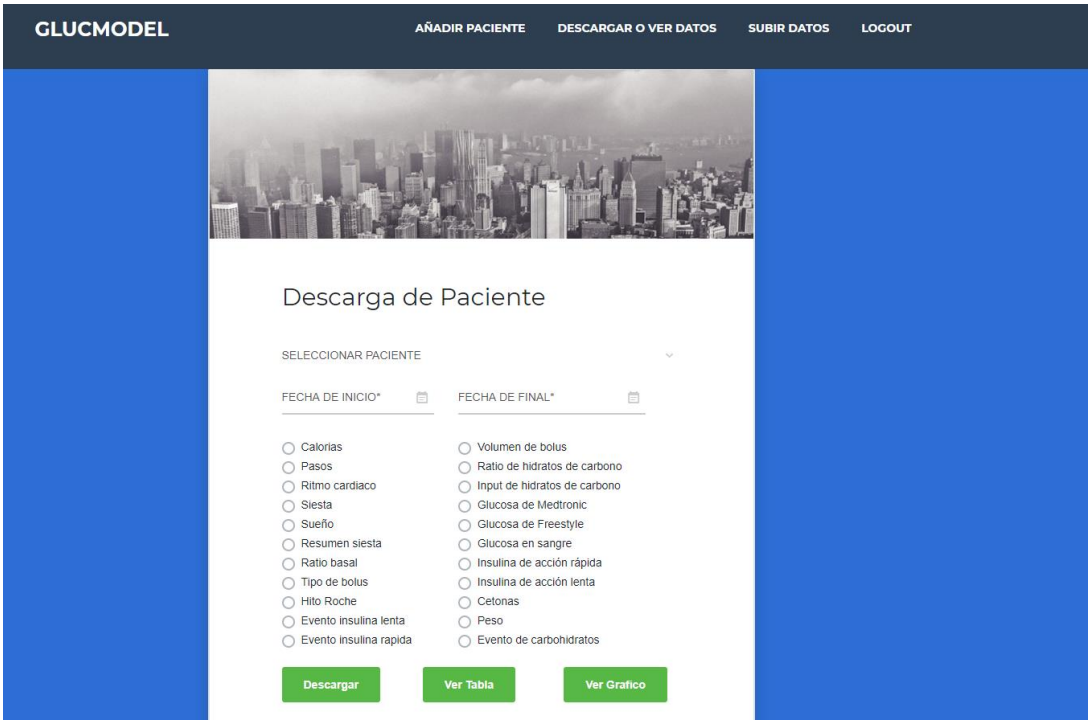
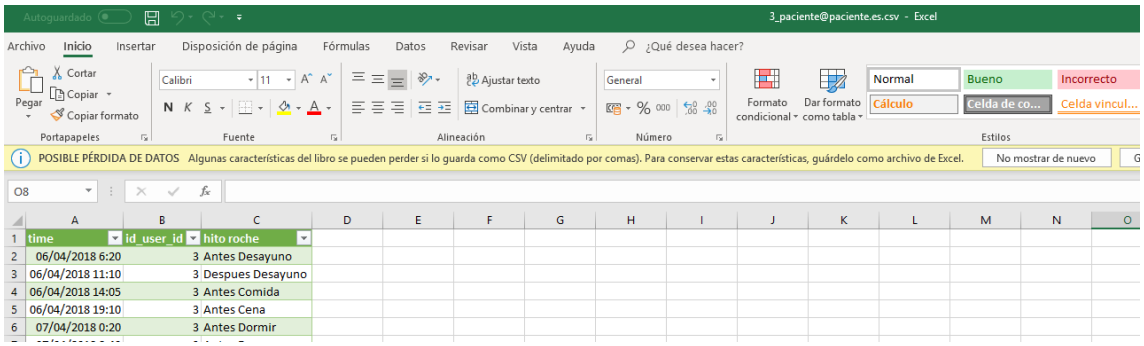


Figura 56: Página de descargar o ver datos de los pacientes

A la hora de ver los datos no existen diferencias entre los usuarios, es decir, todos los tipos de usuarios visualizan el archivo csv descargado, la tabla y el gráfico de la misma forma. En la Figura 57 se muestra un ejemplo de un archivo csv, en la Figura 58 se muestra un ejemplo de una tabla y en la Figura 59 se muestra un ejemplo de un gráfico.



	A	B	C
1	time	id_user_id	hito roche
2	06/04/2018 6:20	3	Antes Desayuno
3	06/04/2018 11:10	3	Despues Desayuno
4	06/04/2018 14:05	3	Antes Comida
5	06/04/2018 19:10	3	Antes Cena
6	07/04/2018 0:20	3	Antes Dormir

Figura 57: Ejemplo de archivo csv

time	id_user_id	calories	heart rate	glucosa medtronic mg dL
2018-06-15T00:00:00	3	4.830000	91.357798	151.0
2018-06-15T00:05:00	3	7.325500	93.924242	159.0
2018-06-15T00:10:00	3	4.347000	94.159292	169.0
2018-06-15T00:15:00	3	4.427500	91.677966	174.0
2018-06-15T00:20:00	3	4.347000	92.899160	176.0
2018-06-15T00:25:00	3	4.427500	95.516393	186.0
2018-06-15T00:30:00	3	5.071500	95.943089	198.0
2018-06-15T00:35:00	3	4.025000	96.015385	209.0
2018-06-15T00:40:00	3	4.186000	95.948718	218.0
2018-06-15T00:45:00	3	4.025000	95.068966	228.0
2018-06-15T00:50:00	3	4.025000	97.526718	237.0
2018-06-15T00:55:00	3	4.025000	100.716312	245.0
2018-06-15T01:00:00	3	4.186000	101.529412	250.0

Figura 58: Ejemplo de datos mostrados en una tabla

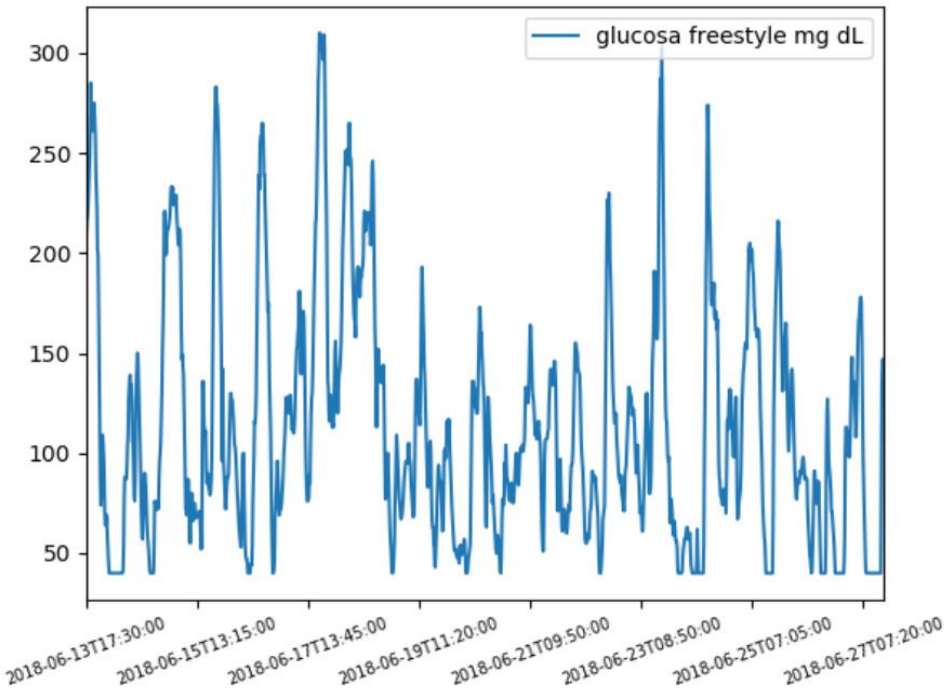


Figura 59: Ejemplo de un gráfico

Lo mismo que ocurre para descargar o ver datos, ocurre para subir archivos, es decir, la página es la misma para los investigadores y médicos, que se les muestra un listado de pacientes a elegir a diferencia de al usuario de tipo paciente. En la Figura 61 se muestra la subida de datos para pacientes y en la Figura 60 se muestra para los investigadores y médicos.

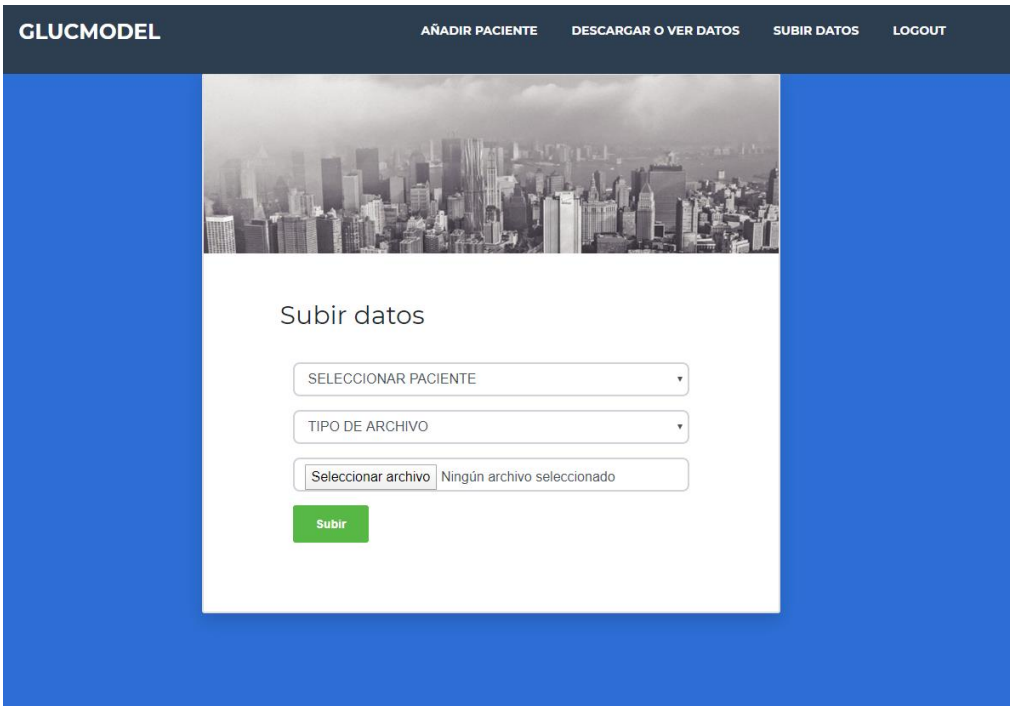


Figura 60: Página subir datos de médicos e investigadores

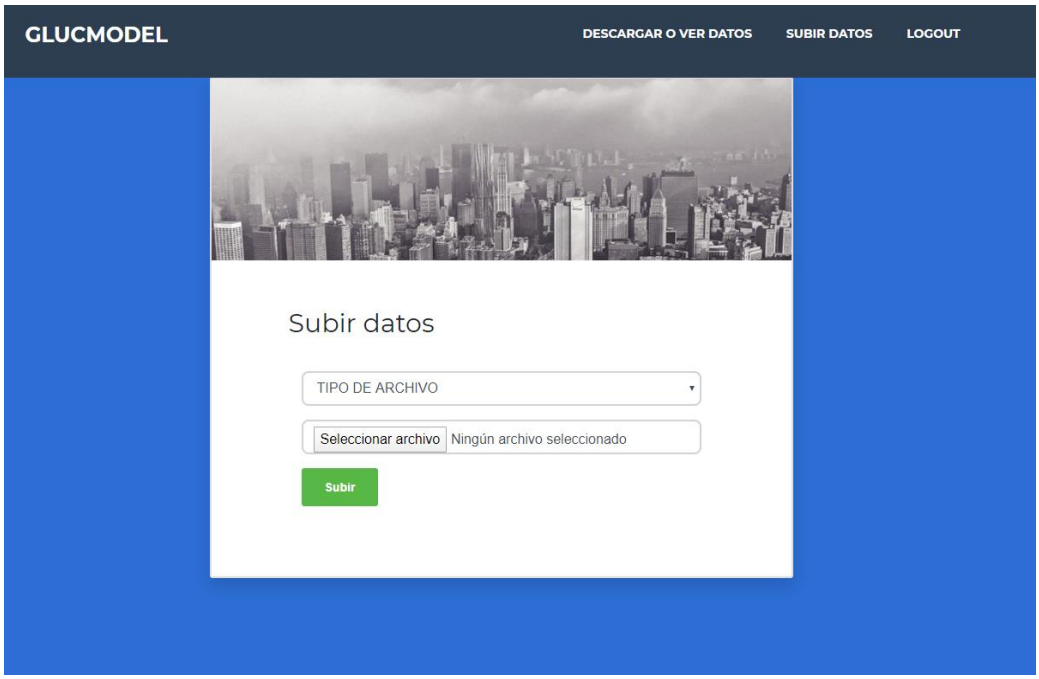


Figura 61: Página subir datos de pacientes

Si la subida de archivo se ha realizado con éxito, aparece un mensaje informando de ello como se ve en la Figura 62

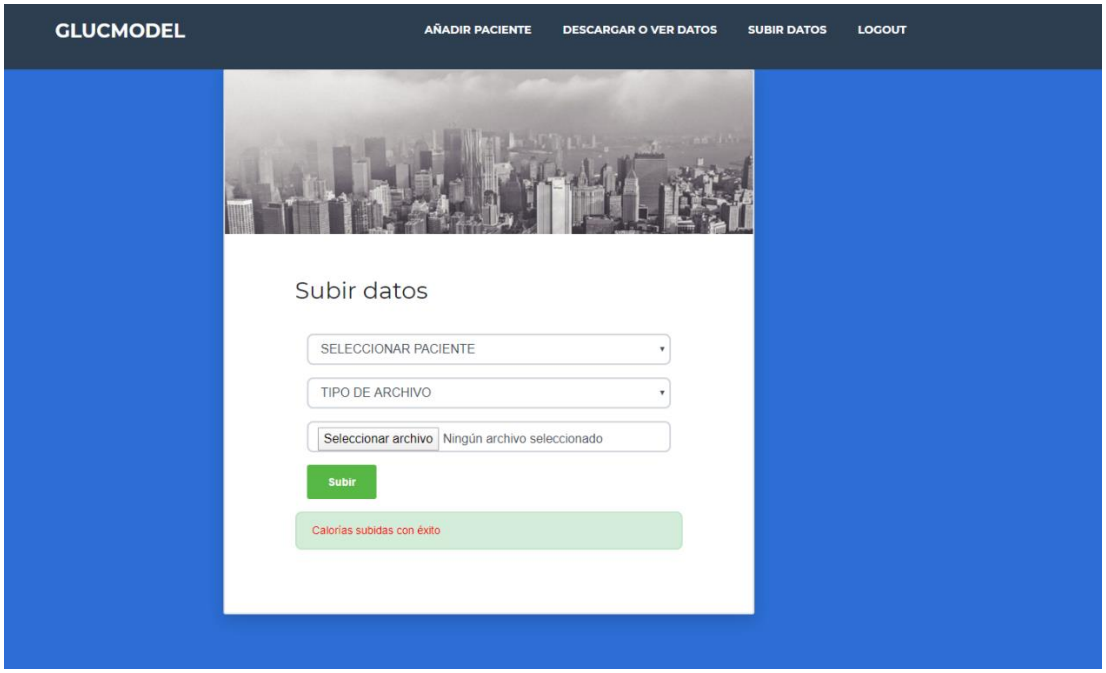


Figura 62: Mensaje de subida con éxito

En caso de que la subida fracase por algún error se mostrara el mensaje como aparece en la Figura 63

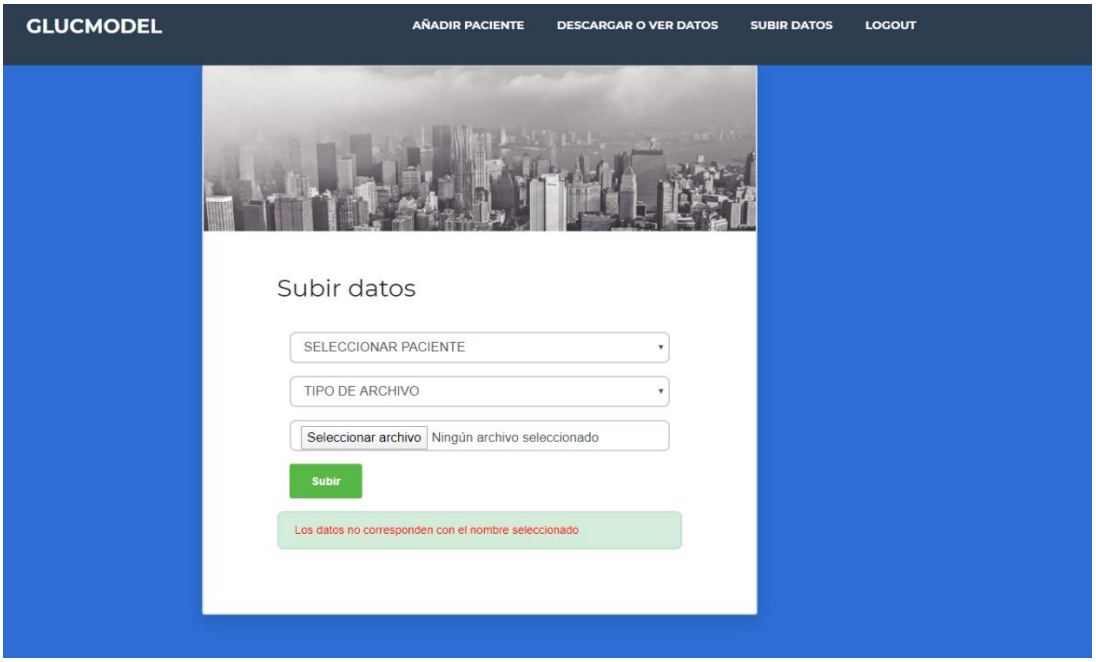


Figura 63: Mensaje de error en la subida