ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

MODELADO DE DATOS Y APLICACIÓN WEB PARA EL ESTUDIO DE LA ENFERMEDAD ARTERIAL PERIFÉRICA

Autor: Jorge Simón Gil

Tutor: Óscar Barquero Pérez
Cotutor: Rebeca Goya Esteban

Curso académico 2017/2018

"Tú eres el dueño de tu suerte"(Galleta de la fortuna)
"La suerte es el esfuerzo de los tontos"(Anónimo)

II

Agradecimientos

Este es un apartado dedicado a echar la vista atrás, a reflexionar sobre las personas que han pasado a nuestro lado y a las que han permanizo en él. No podía empezar los agradacimientos sino es por mis padres, lejos del tópico, gracias a ellos he podido cursar mis estudios, a nuestro esfuerzo conjunto me han abierto las puertas del mundo desde una posición privilegiada, me han acompañado y enseñado a lo largo de este trayecto, y a pesar de cada vez irme convirtiendo en una persona adulta ellos siguen ahí enseñandome y guiandome ante las incertidumbres que se me puedan presentar y presentan.

Desde luego y gran medida también puedo considerar como a mis padres a mis profesores, mis padres profesionales aquellos sin los que hoy no estaría en la empresa en la que estoy, sin los que no habría aprendido la gran mayoría de lo que sé, profesionalmente hablando. A pesar de los malos ratos que también por su culpa se pasan hasta llegar a este punto, hasta llegar a cerrar una etapa en la cual aprendes para aprobar y abrir una nueva en la cual aprendes por gusto. Gracias a vosotros los profesores que habeís hecho que me pique el gusanillo de la curiosidad, que despierte en mi el hambre de conocimiento y que habeís hecho de mi un pequeño ingeniero con una gran proyección.

Y no menos importante es sin duda mi pareja, aquella que más me ha sufrido durante los cuatro años de carrera. Aquella sin la que en alguna que otra ocasión seguro hubiera tirado la toalla, sin la que seguro que estaría menos cuerdo aún de lo que estoy y aquella que supone un gran apoyo para lo que me queda por afrontar. Porque nadie la dijo que era fácil aguantar a un proyecto de ingeniero en época de exámenes, y lejos de abandonarme a mi suerte y sabiendo lo que viene, me sigue apoyando para que continúe mis aventuras ahora relacionadas con el máster. Por todo esto y más gracias.

Y finalmente agradecer a todas las personas que me han acompañado a lo largo de este proceso. Amigos sin los cuales las tardes en los laboratorios

o biblioteca se hubieran hecho insufribles, sin los cuales la mesa del comedor hubiera estado vacía y sin los cuales no sería hoy un poco más sabio. Nacho, Alex, Eva... hemos pasado grandes tardes juntos, me habéis sacado de mis casillas y yo a vosotros, hemos tenido discusiones filosóficas sobre que hacíamos y porque nos gustaba esta tortura pero finalmente esta tortura se acaba y las cicatrices que tenemos las hemos cosido juntos.

Por todo esto y seguramente más personas y enseñanzas que me pueda estar dejando en el tintero, gracias a todos sin los que hoy no estaría donde estoy.

Gracias.

Resumen

La Enfermedad Arterial Periférica (EAP) consiste en la disminución del flujo circulatorio arterial al miembro inferior, por lo que es catalogada como cardiovascular oclusiva. La actividad física moderada, incluso caminar, provoca dolor en las zonas afectadas, por lo que es una enfermedad con un alto impacto en la calidad de vida de los pacientes que la sufren. Un análisis de los parámetros clínicos, y de las pruebas físicas de los pacientes podría ayudar a esclarecer los factores de riesgo, permitir una detección precoz y un mejor tratamiento de los pacientes. [4]

Uno de los grupos de personas más afectadas por esta enfermedad son los diabéticos, los cuales según estudios de la OMS son ya 1 de cada 11 personas, y dicha población se está incrementando [13]. También aquellas personas con enfermedades cardiovasculares forman parte del grupo de riesgo de la EAP, y estas corresponden al 31 % de las muertes registradas anualmente [14], por encima de las muertes por tabaquismo o accidentes de tráfico. Y otro de los grupos de riesgo es el formado por quienes sufren problemas respiratorios, los cuales son más de 235 millones sobre los 7000 millones de personas en el mundo, entorno al 30 % [15]. Por lo que la EAP es una enfermedad de riesgo para cualquier habitante del planeta.

Para ello como Trabajo de Fin de Grado he desarrollado una herramienta de trabajo común para los estudios médicos, en relación con los estadísticos. Que busca generar un incremento en la calidad de los estudios, eliminando los problemas típicos del ámbito relacionados con la toma de datos, la gestión y evaluación de pacientes. Además la aplicación proporciona un módulo de integración de Maching Learning (ML) gracias al cuál en tiempo real se pueden generar, probar y comparar modelos. El trabajo parte como base del estudio realizado durante el periodo de 01–01–2014 a 31–12–2016 sobre la EAP por parte de nuestra universidad a cargo del Dr. José Luis Rojo [4].

Aunque hayamos partido de un estudio cerrado sobre la EAP, la herra-

mienta permite ser utilizada para otros estudios, una vez ajustados sus parámetros de configuración. Es decir dicha herramienta no es específica, sino que pretende sentar bases para la automatización en relación a la realización de estudios médicos que necesiten de la integración de principios estadísticos avanzados como el ML.

Índice general

Agradecimientos **III**

Resumen **V**

Lista de figuras **IX**

Lista de tablas **XI**

1. Introducción **1**
1.1. Motivación 1
1.2. Objetivos 2
1.3. Metodología y estructura de la memoria 2

2. Estado del arte **3**
2.1. Contexto de estudio 3
2.2. Tecnologías aplicadas al desarrollo 4

3. Estudio de los datos proporcionados **5**
3.1. Formalirio de inclusión de pacientes 5
3.2. Enfermedades 6
3.3. Medicamentos 10
3.4. Pruebas complementarias 16
3.5. Conclusiones sobre el estudio de los datos 17
3.6. Modelado de los datos 18
3.7. Estructura de datos 19
3.7.1. Medicamentos 19
3.7.2. Enfermedades 20
3.7.3. Pruebas 20
3.7.4. Pacientes 21

4. Creación del servidor **23**
4.1. Configuración del servidor 23
4.2. Inserción de los datos de los pacientes 24
4.2.1. Automatización datos pacientes 24
4.2.2. Inserción de nuevos pacientes 25
4.3. Interacción Servidor Usuario 25
4.4. Interacción Servidor Base de Datos MongoDB 26
4.4.1. API creada para el manejo de la base de datos: 26
4.5. Código colores depuración 28

5. Desarrollo de la página web **33**
5.1. Responsividad página web 33
5.2. Obtención de datos del servidor 34
5.3. Insercción de gráficos 34
5.4. Interacción del usuario vía WEB 34

6. Estimación **45**

7. Conclusiones y líneas futuras **47**
7.1. Conclusiones 47
7.2. Líneas futuras 49

Índice de figuras

3.1. Formulario de los pacientes. 5
3.2. Esquema base de datos. 19

5.1. Index HTML. 35
5.2. searchPatient HTML. 36
5.3. riskPatient HTML. 36
5.4. medicamentos HTML. 37
5.5. enfermedades HTML. 37
5.6. tipos de paciente HTML. 38
5.7. boxplot edadessexo HTML. 38
5.8. histogramas de edades HTML. 39
5.9. histograma de alturas HTML. 39
5.10. histograma de pesos HTML. 40
5.11. singupPatient 1 HTML. 40
5.12. singupPatient 2 HTML. 41
5.13. patient HTML. 41
5.14. Página con el riesgo del paciente. 42

7.1. Arquitectura sistema. 49

Índice de tablas

3.1. Medicamentos 16

4.1. Urls que resuelve el servidor 26
4.2. API pacientes 29
4.3. API medicamentos 30
4.4. API enfermedades 31
4.5. API pruebas 32

5.1. API gráficos 43

Capítulo 1

Introducción

En este capítulo se explica la motivación que ha dado lugar a este trabajo fin de grado. A continuación, se detallan los objetivos del trabajo. Finalmente, se describe la metodología seguida, para la consecución de los objetivos y la estructura de la memoria.

1.1. Motivación

La Enfermedad Arterial Periférica o EAP es una enfermedad que afecta al sistema cardiovascular obstruyendo tanto el riego como el flujo sangíneo de las extremidades inferiores. Una de sus consecuencias es que durante la actividad física los pacientes padecen dolor en las zonas afectadas y la necesidad de detenerse, esto se conoce como claudicación intermitente, y cuando hablamos de actividad física nos referimos incluso caminar. [4]

Además los grupos de riesgo del estudio de la EAP forman un gran porcentaje de la población mundial:

- Diabéticos: 1 de cada 11 personas. [13]
- Enfermedades Cardiovasculares: 31 % de la población. [14]
- Enfermedades Respiratorias: 235 millones de los 7000 millones de habitantes en el mundo. [15]

Por lo que un análisis de los parámetros clínicos de los pacientes podría ayudar a esclarecer los factores de riesgo, permitir una detección precoz y un mejor tratamiento de los pacientes.

1.2. Objetivos

El objetivo global de este trabajo de fin de grado es proporcionar un entorno único para la realización de estudios clínicos. Para ello, se ha abordado el diseño de un entorno que permite el almacenamiento estructurado de los datos de los pacientes, y en el que se pueden realizar un análisis estadístico de los mismos. Para ello he estructurado el trabajo en varias etapas:

- Estudio de los datos proporcionados.
- Creación de un servidor para albergar los datos de los pacientes.
- Módulo de integración de aplicaciones básadas Maching Learning o Deep Learning con el fin predecir el grado de enfermedad en un paciente.
- Desarrollo de la interfaz de usuario a través de tecnologías web.
- Implementación de los módulos estadísticos de monitorización del estudio.

1.3. Metodología y estructura de la memoria

La metodología seguida en el presente proyecto consta de las siguientes etapas:

- Revisión bibliográfica sobre el estudio de la EAP.
- Exposición de las enfermedades y los medicamentos observados durante el estudio de los datos.
- Presentación de la base de datos generada a partir de los datos observados.
- Creación del servidor que albergue los datos y permita la interacción con ellos.
- Finalmente se plantea un módelo de maching learning el cual nos permita estimar el grado de enfermedad de un paciente.

Capítulo 2

Estado del arte

En esta sección se realiza una pequeña introducción sobre la EAP y las tecnologías en las que se basa la aplicación desarrollada. Comenzando por una visión general de la EAP y finalizando con una presentación más detallada de las tecnologías en las que se basa la aplicación.

2.1. Contexto de estudio

La EAP es una enfermedad circulatorio oclusiva, es decir a lo largo del desarrollo de algunas de las enfermedades causantes de la EAP (diabetes, colesterol, otros problemas circulatorios), el paciente sufre de dolores en las extremidades inferiores provocadas por la falta de irrigación sanguínea. Dependiendo del grado de desarrollo de la enfermedad el dolor se puede presentar de forma continua en labores del día a día, normalmente caminando, o en situaciones de estrés físico.

Los datos utilizados en este trabajo fin de grado, fueron obtenidos en un proyecto de investigación conjunto entre la Universidad Rey Juan Carlos y la Universidad Pablo Olavide de Sevilla. El proyecto incluía la adquisición de datos clínicos de pacientes con diferente grado de EAP, así como la realización de pruebas clínicas y físicas. El objetivo era reunir un conjunto de datos que permitiesen ayudar a esclarecer los factores de riesgo, permitir una detección precoz y un mejor tratamiento de los pacientes. [4]

Por ello surgió la idea de crear una herramienta que permitiese el almacenamiento y análisis estadístico de los datos. No obstante, esta herramienta se ha diseñado de forma que posteriormente puedan integrarse otros estudios diferentes.

La idea de integrar estudios estadísticos surge a partir de dos preguntas fundamentales.

- ¿Cómo podemos conseguir datos de calidad para el estudio a bajo coste? La respuesta principal y que inspirá toda la plataforma es: El laboratorio de herramientas Physio (PhysioBank: Banco de bases de datos de señales fisiológicas, PhysioToolkit: Banco de herramientas que proporciona, y PhysioNet: Portal Web para la gestión y trabajo sobre los datos de Physio) que es un portal de señales fisiológicas complejas sobre diversos campos de la bioingeniería. [16]

- ¿Cómo mejorar Physio? Consiguiendo homogeneizar los datos que pone a disposición, y elaborar una herramienta de integración sobre los modelos sin limitaciones en el servidor. La idea de mejora, nos lleva a proporcionar servicios que integrén y monitoreen los estudios, como los que pone a su disposición la plataforma de Luca propiedad de Telefonica, para el campo y los servicios relacionados con las telecomunicaciones.

2.2. Tecnologías aplicadas al desarrollo

En base a esas dos preguntas, surgió la integración y el uso de Python, Django y MongoDB.

- Python: Por el soporte y apoyo que encuentra en la comunidad, por las librerías de ML que tiene ya implementadas y que se usan de forma extensa. La otra opción era R, pero debido a que python montaba en el mismo lenguaje su propio servidor fue descartado. Además Python ya integra una herramienta de desarrollo a través del software que proporciona Anaconda. [18]

- Django: La elección de Python y Django fue conjunta, necesitábamos un servidor el cual pudiera funcionar de forma directa y ya integrada con las librerías de ML más usadas. [17]

- MongoDB: Quedaba de una ddbb que tuviera soporte para el lenguaje Python, que además pudiera manejar con facilidad y rapidez grandes volúmenes de datos y permitir modificaciones en los modelos sin mucho esfuerzo, lo que nos llevo a una ddbb no relacional y en su momento por el estado de madurez que tenía, a MongoDB. [19]

Capítulo 3

Estudio de los datos proporcionados

En este capítulo se muestra el formulario para la inclusión de pacientes en el estudio. A continuación, se realiza una recopilación de enfermedades que padecen los pacientes y de los medicamentos que toman.

3.1. Formalirio de inclusión de pacientes

El formulario de pacientes proporcionado por el estudio es el siguiente:

Codigo de paciente		Nombre		Apellido		
Fecha de nacimiento		Talla (cm)		Peso (kg)		
Tipo de paciente	Control		I Brazo/tobillo izq		I Brazo/tobillo der	
	Asintomático					
	Claudicante		Pierna Izquierda		Pierna Derecha	
Tiempo de diagnóstico		¿Fumador o no?		Distancia claudicación (m)		
Otras enfermedades						
Medicación						
Actividad física ¿Si o No?		Dias		Actividad		
Pies planos/cavos		¿Usa plantillas?		Pierna Dominante		
Correo Postal						

Figura 3.1: Formulario de los pacientes

Como podemos observar consta de campos que por la ley de protección de datos debemos anonimizar, que son:

- Nombre.
- Apellidos.
- Residencia del paciente (Los formularios que proporcionen dicha información).

Y por contra el resto de datos son los que proporcionaran información al estudio. Para ello primero se mostrará la recopilación de enfermedades y medicamentos de los pacientes.

3.2. Enfermedades

A través de los formularios de datos de los pacientes se han recopilado todas las enfermedades padecidas por los pacientes, posteriormente se ha hecho un filtrado de las mismas para eliminar las que estuvieran en diversos pacientes a la vez, o que fueran la misma enfermedad pero afectasen a diferentes partes del cuerpo(ejemplo: By-pass brazo derecho, By-pass pierna izquierda). Para ello ha habido que indagar en el significado de las diversas enfermedades y sus principales afecciones. Finalmente he conseguido quedarme con la siguiente lista de enfermedades:

- HTA: Hipertensión Arterial. La presión arterial consiste en la medición de la fuerza ejercida contra las paredes de las arterias en cada bombeo del corazón. Se considera hipertensión cuando la presión arterial es de 140/90 mmHg o mayor.
- Ácido Úrico: Esta enfermedad consiste en un exceso de $C_5H_4N_4O_3$ o Ácido Úrico en sangre. Normalmente el cuerpo elimina este exceso a través de la orina al ser filtrado por los riñones, pero cuando no se elimina suficiente se sufre dicha enfermedad. Cuando esto sucede se denominan hiperuricemia.
- Colesterol: El Colesterol es una grasa del organismo la cual se encarga de lubricar las arterias para la correcta distribución de los nutrientes y la sangre por las diferentes zonas del cuerpo. Cuando se sufre de colesterol se puede sufrir en exceso o en defecto, en ambos casos deja de cumplir su funcionamiento y causa un efecto dañino en el organismo, por obstrucción o disfunción de las arterias.
- Diabetes Tipo II: Es una enfermedad crónica la cual provoca un alto nivel de glucosa en sangre. Esto es causado o un incorrecto funcionamiento por parte del pancreas durante el proceso digestivo de los alimentos, lo que también conlleva posibles bajadas de glucosa en el organiso. Dicha enfermedad conlleva problemas musculares(y por musculares no debemos entender solo las partes del cuerpo llamadas músculos como los gemelos o el biceps, sino también los organos como el corazón o cerebro) ya que estos necesitan del azucar para su correcto funcionamiento.

- Neuropatía Pefiférica: Se denomina a la atrofía o insuficiencia de nervios sanos entre el cerebro y la medula espinal. Esto provoca dolor, insensibilización e incapacidad en los musculos.

- Infarto De Miocardio: Un infarto en general es la necrosis o muerte de los tejidos que forman un órgano. Cuando hablamos de infarto de miocardio nos referimos a la necrosis de la cavidad aortica del corazón, puediendo llegar a provocar incluso la parada del mismo. La causa es la hipoxia en las ceculas del órgano en cuestión, suele estar relacionada con otras enfermedades circulatorias como el colesterol o HTA.

- By-pass: El By-Pass es una práctica médica por la cual se recontruye una sección de un sistema (Ejemplo: El circulatorio) del cuerpo humano, atrofiando por contra otra sección menos vital de dicho sistema. Normalmente se usa como método para devolver la circulación a un organo afectado (Ejemplo: El corazón), o en operaciones para mantener la respiración del paciente.

- Pié Diabético: Es una infección que puede llegar a ulceración o destrucción de los tejidos oseos, musculares y nerviosos, dependiendo del grado de la infección. Por norma afecta a pacientes con Diabetes Tipo II y está relacionado con el incorrecto tratamiento de la Diabetes. En el peor de los casos puede llevar a la amputación del miembro afectado.

- Pié Sin Falanges: Pié que carece de las falanges (dedos) debido a accidente o relacionado con alguna enfermedad (Ejemplo: Pié diabético)

- Marcapasos: Mecanismo diseñado para la regulación y mantenimiento del ritmo cardiaco, a través de la electroestimulación del corazón. Funciona analizando los propios latidos y corrigiendolos cuando sea necesario.

- Stent: Es una malla que se introduce dentro de los vasos sanguineos para evitar la constricción de los mismos. Suele estar relacionada con la cirugía angioplastica, la cual consiste en la limpieza y desobstrucción de los vasos sanguineos.

- Operación De Cataratas: Es una operación la cual consiste en retirar el cristalino del ojo, porque se ha vuelto opaco, y sustituirlo por una lente artificial transparente.

- EPOC: Es un transtorno que consiste en la obstrucción de las vías respiratorias progresiva e irreversible. Está relacionada con la alta exposición

a humos tóxicos como el CO o monóxido de carbono, el $(NH_4)(NO_3)$ o nitrato de amonio, etc, procedente del tabaco, fábricas o coches. Trae consigo otras enfermedades relacionadas como: el Enfisema Pulmonar o agrandamientos de los bronquios de forma permanente, y la Bronquitis Crónica que es una inflamación constante de los bronquios. Y su tratamiento puede tener efectos secundarios como la hipertensión pulmonar.

- Apnea Del sueño: La apnea consiste en la falta de respiración durante un periodo del tiempo. En el caso de la apnea del sueño o apnea-hipopnea durante el sueño esto se da durante el periodo de sueño del paciente. Es una patología del sistema respiratorio que afecta a la faringe que afecta a todas las personas pero que cuando excede de cierto número en el transcurso del sueño es considerado como un factor de riesgo de la salud. Según el grado de apnea puede causar fatiga diurna, fatiga crónica o incluso alteraciones respiratorias y cardiovasculares.

- PCR: El PCR consiste en episodios de crisis durante los cuales ni el corazón ni los pulmones ejercen su normal funcionamiento, es decir colapsan y entran en parada. Esto provoca la hipoxia celular en el organismo del paciente que lo sufre.

- Angina De Pecho: Es una enfermedad de las arterias coronarias del paciente, las cuales se obstruyen debida a la hipoxia celular en la zona coronaria del corazón. Se suele confundir con el infarto de miocardio, pero la zona del corazón afectada es diferente.

- Monoparesia (Secuela De Polio): Parálisis parcial o total de una extremidad, en función del grado de la enfermedad la parálisis puede manifestarse como un hormigueo constante en la extremidad afectada.

- Cancer De Próstata: Proceso de disgregación celular de forma másiva, de células anómalas, que puede formar grandes masas denominadas tumores. Concretamente el cancer de próstata se lócaliza en la próstata(órgano encargado de la segregación del semen) pero puede extenderse en caso de no se tratado hasta el punto de llegar a metástasis.

- Parálisis Facial: Daño en los nervios faciales lo que impide el movimiento de uno o más músculos de la cara. Puede ser total o parcial, temporal o permanente dependiendo de los nervios afectados y el grado de daño sufrido por estos. Está relacionado con Ictus, estrés, etc.

- Pólipos En El Colon: Los pólipos son crecimientos anormales de téjido que surgen del capa interior o mucosa, en forma de ramificaciones. Pueden darse en las cavidades sinuosas, o como en este caso en el colon. Pueden derivar en cancer.

- Cardio Isquémica: o Cardiopatía isquémica es causada por la arteriosclerosis de las arterias coronarias, que consiste en que la formación de colágeno y lípidos en el órgano en cuestión (el corazón) no es eliminada o se acumula hasta el punto de ser peligroso para el correcto funcionamiento de dicho órgano. Está directamente relacionada con el colesterol, ser fumador, tener diabetes, ser obeso o tener antecedentes familiares. También tener un cuadro clínico previo de angina de pecho o infarto. Causa un incorrecto funcionamiento del corazón.

- Ictus (cerebral): o accidente cerebrovascular (ACV) es una enfermedad cerebro vascular que afectas a las venas encargadas de sumistrar al cerebro tanto los nutrientes necesarios para su funcionamiento, como oxígeno. Puede darse debido a una embolia o trombosis cerebral. Cuanto esto sucede las celulas del cerebro mueren por hipoxia cerebral.

- Dedos En Garras: Los dedos en garra suelen producirse por una agravación de la desviación del hallux valgus, que es una deformidad que afecta al primer segmento metatarsodigital del pié excepto en los pulgares. Lo que causas deformidades en la pisada y posteriores molestias en diferentes articulaciones del aparato motor del paciente(rodillas, cadera, tobillos, etc).

- Embolia Pulmonar: Es la obstrucción o bloqueo de forma súbita de una arteria pulmonar, causada por un trombo o coágulo sanguíneo. Los síntomas y posteriores efectos permanentes pueden ser: falta de aire, dolor torácico, tos con sangre e incluso la muerte. El problema reside en la formación de los trombos, normalmente en regiones musculares grandes como el cuadriceps.

- Endoprótesis Aorta Abdominal: Una endoprótesis aortica es un tipo de prótesis introducida en el interior del organismo (Ejemplo:stents) que se utilizan en las estenosis de las arterias coronarias o al que se emplea para evitar la rotura del aneurisma de aorta. En el caso de ser abdominal quiere decir que se ha realizado se realiza en la zona aortica correspondiente al estómago para evitar un futuro deterioro de la aorta.

- Escoliosis: Curvatura de la espina dorsal en forma de 'Só de 'C'.

- Neumonía: Inflamación de los pulmones debido a una causa vírica o basteriana. Produce insuficiencia respiratoria, fiebre alta y dolor torácico.
- Coma Inducido: Periodo de estado de inconsciencia controlado por el suministro de barbitúricos, para proteger alguno de los sistemas vitales del paciente como el neuronal o el cardiovascular. También puede usarse para tratar epilepsia.
- Operado de Estenosis Aórtica: Operación para reparar la válvula aórtica y restaurar el flujo normal de sangre en el corazón. Es una enfermedad de riesgo para sufrir infartos o angina de pecho.

[12]

3.3. Medicamentos

Al igual que se hizo con las enfermedades, también he creado un listado de los medicamentos primero basandonos en los descritos en el estudio y posteriormente los he filtrado en función de su fin médico. Para ello a través de la página de la agencia española del medicamento he sacado una breve descricción de los medicamentos permitiendonos así eliminar duplicados o generalidades que no aporten información:

Medicamento	**Enfermedad**	**Efectos Secundarios**
Pastillas para HTA	HTA	Eliminación de elementos como sodio y potasio, importantes en el mantenimiento de funciones musculares y nerviosas.
Pastillas para colesterol	Colesterol en sangre	Hepatitis, debilidad e inflamación muscular, estreñimiento, náuseas o diarreas
Antidiabéticos orales	Diabetes	hipoglucemia, intolerancia gastrointestinal

Medicamento	**Enfermedad**	**Efectos Secundarios**
Insulina	Diabetes	enrojecimiento, hinchazón o irritación en el sitio de la inyección. Cambios en la sensación de su piel, engrosamiento de la piel (acumulación de grasa) o un poco de depresión en la piel (irregularidad de la grasa) aumento de peso.
Metformina 850 mg	Diabetes o SOP	acidosis láctica
Simbastatina	Colesterol en sangre	Miopatía, Astenia, Diarrea, Náuseas, Vómitos, Cefaleas, Pancreatitis aguda o Neuropatía periférica
Parches para memoria	Perdida de memoria	náuseas, vómitos, pérdida de hambre, fatiga y las reacciones cutáneas
Pastilla (orina)	Prostatítis	somnolencia, náuseas y mareos
Bisoprolol 5 mg	hipertensión, angina de pecho crónica estable, insuficiencia cardiaca crónica estable	cansancio excesivo, vómitos, diarrea, dolores musculares, secreción nasal, respiración entrecortada, inflamación (hinchazón) de las manos, los pies, los tobillos o las piernas

Medicamento	**Enfermedad**	**Efectos Secundarios**
Torasemida 5mg	Edema, enfermedad renal o hepática, hipertensión	dolor de cabeza, vértigo, cansancio, debilidad, calambres musculares, y molestias gastrointestinales
Ramipril 5mg	hipertensión, infarto de miocardio, insuficiencia cardiaca	Dolor de cabeza o sensación de cansancio Sensación de mareo, hipotensión, sinusitis, calambres o dolor en sus músculos, vértigo, Visión borrosa
Xarelto 20 mg	prevenir la formación de coágulos y tratarlos	anemia, vértigo, dolor de cabeza, hemorragia en diversos puntos del organismo, hipotensión
Adiro 100 mg	infarto de miocardio o una angina de pecho, accidente cerebrovascular no hemorrágico transitorio o permanente, angioplastia coronaria o by-pass coronario	Trastornos gastrointestinales, como úlcera gástrica, úlcera duodenal, sangrado gastrointestinal, dolor abdominal, molestias gástricas, náuseas, vómitos
Pantoprazol 20 mg	reflujo gastroesofágico	dolor de articulaciones, urticaria, dificultad para respirar o tragar, ritmo cardíaco fuerte, rápido o irregular cansancio excesivo, espasmos musculares.

Medicamento	**Enfermedad**	**Efectos Secundarios**
Nitrofix 5 mg	angina de pecho	hipotensión arterial, debilidad, taquicardia, desmayos, palpitaciones, sofocos, mareos, náuseas, vómitos y dermatitis
Concor 10	insuficiencia cardiaca	Lasitud, mareos, cefalea ligera, empeoramiento de la claudicación intermitente, hipotensión erupciones cutáneas y hipoglucemia
Tramadol retad NORMON	analgésico	hipoglucemia, arritmia, hipotensión, ataques epilépticos, tirones musculares, disnea
Gliclazida	Diabetes	Dolor abdominal, náuseas, vómitos, indigestión, diarrea y estreñimiento
Enalapril	hipertensión, infarto de miocardio, insuficiencia cardiaca	inflamación de la cara, la garganta, la lengua, los labios, los ojos, las manos, los pies, los tobillos o las pantorrillas
Pravastatina	angina de pecho, colesterol	acidez dolor de cabeza, dolor, sensibilidad o debilidad muscular falta de energía, cansancio extremo, sangrado o moretones inusuales
idalprem	Trastornos del sueño, Insomnio, Hiperemotividad, Neurosis	fatiga, dolor de cabeza, mareo, debilidad muscular, ataxia

Medicamento	**Enfermedad**	**Efectos Secundarios**
pantecta	enfermedades relacionadas con el ácido del estómago e intestino	dolor en la parte superior del abdomen, diarrea, estreñimiento, flatulencias, dolor de cabeza, mareo, visión borrosa, náuseas, vómitos, reacciones alérgicas
Lyrica	epilepsia	Fibromialgia, Dolor neuropático diabético, Dolor neuropático por lesión de la médula espinal, Dolor posterior al herpes zóster, Convulsiones de inicio parcial en adultos con epilepsia
Versatis	dolores	enrojecimiento, erupción, picor, quemazón, dermatitis y pequeñas vesículas
Clopidogrel	problemas circulatorios	efectos aterotrombóticos (como infarto cerebral, infarto de miocardio o muerte)
Hemovas	trastornos de la circulación periférica	sofocos, náuseas, vómitos o diarreas, arritmias, alergias, colestasis intrahepática y elevación de las transaminasas, hipotensión
Dolocat	analgésico muscular	hipotensión, transaminasas en sangre elevadas

Medicamento	**Enfermedad**	**Efectos Secundarios**
Pastillas para la circulación	Problemas de circulación	náuseas, vómitos o diarreas, arritmias, angina de pecho, alergias, vertigos
Lixumia	Diabetes mellitus tipo 2	náuseas, vómitos, diarrea y dolor de cabeza, alergias
Omeprazol	Problemas de estomago	cefaleas, diarrea, dolor de estómago, náuseas, mareos, dificultad para despertar y pérdida de sueño
Vasaltan	antihipertensivo	desgaste del tejido muscular
Furosimida	insuficiencia cardíaca congestiva, hipertensión y edemas	hiperglicemia y glucosuria, náusea, anorexia, vómitos, diarrea, y constipación
Losartan	hipertensión	problemas músculo-esqueléticos
Atomastatina	colesterol	debilidad muscular, dolor muscular, malestares digestivos, problemas de vista
Tranziliun	ansiolítico, hipnótico, anticonvulsivante, sedante, relajante muscular y amnésico	somnolencia, mareo, debilidad muscula
Arcosia	reducir el dolor y la hinchazón (inflamación) en las articulaciones y músculos	dificultad para respirar, dolor torácico o hinchazón de tobillo, ictericia, alergias

Medicamento	**Enfermedad**	**Efectos Secundarios**
Carduran	Déficit de atención	accidente cerebrovascular, hipoestesia, síncope, acúfenos, angina de pecho, infarto de miocardio, epistaxis, estreñimiento, diarrea, flatulencia, vómitos, gastroenteritis, pruebas de función hepática anormales, erupción cutánea, artralgia
Pletal	claudicación intermitente	aritmias, inflamación de los tobillos, los pies o la cara, infarto de miocardio, diabetes y aumento de la concentración de azúcar en la sangre
Amlodipino	hipertensión y la enfermedad de las arterias coronarias	inflamación de las manos, pies, tobillos o piernas y cefalea
Sintrom	anticoagulante	hemorragias, alergias, hematomas con ampollas en la piel, ictericia

Tabla 3.1: Medicamentos [10] [12]

3.4. Pruebas complementarias

A los pacientes del estudio se les realizaron una serie de pruebas médicas que proporcionan información sobre su estado de salud.

- Ánalisis de la marcha: Estudio sobre el modo de locomoción bípeda con actividad alternada de los miembros inferiores, consistente en exploración muscular y articular en camilla, análisis de la postura y equilibrio (estabilometría), análisis estático y dinámico de presiones plantares con plataforma baropodométrica (presión ejercida por las diferentes zonas de la planta del pié durante la marcha). [9]

- Motas de la marcha: Pequeños sensores colocados en ambas extremidades del paciente, lo cual nos permite realizar diversas mediciones sobre la cadena cinemática en el momento de andar. Las mediciones que tomará serán: tiempo de vuelo (periodo durante el cual el pié se encuentra en el aire), tiempo de apoyo (periodo durante el cual se encuentra en contacto el pié con el suelo), tiempo de apoyo doble (periodo en el cual ambos piés se encuentran en contacto con el suelo) frecuencia del ciclo (cada cuanto se repite una cadena cinemática), longitud del ciclo (cuanto se tarda en realizar al cadena cinemática). [7]

- VFC: La prueba de la variabilidad de la frecuencia cardíaca se basa en medir de forma no invasiva, a través de una señal de ritmo cardíaco, el estado de sistema nervioso autónomo. Se han propuesto en la literatura numerosos métodos para cuantificar la VFC [3]. Finalmente es una herramienta preventiva para evitar riesgos en las diversas patologías, y en medicina deportiva lo que se pretende es facilitar la adaptación de los deportistas a los entrenamientos. [1] [2]

- NIRS: La prueba NIRS consiste en la medición del oxigeno en sangre durante la ejecución de algún ejercicio, a atravéś de un espectroscopio. Este, a través de la longitud de onda captada a través de la piel al atravesar el torrente sangíneo determina el nivel de óxigeno en sangre. Es una técnica parecida a la que se utiliza para determinar si las bolsas de aire contienen petroleo. [20]

- ITB: Índice tobillo brazo, consiste en medir la variación de la PAS entre dos sensores. Uno de ellos ubicado en el tobillo y el otro en el brazo del paciente. Se debe realizar para las dos extremidades. Esta prueba servirá dentro del estudio de la EAP para determinar el tipo de paciente: riesgo, control, claudicante. [5]

3.5. Conclusiones sobre el estudio de los datos

Tras la observación y documentación sobre el estudio de la EAP, y en comparación con otros estudios médicos como el Informe Anual del Sistema Nacional de Salud 2015. Se llega a la conslusión de que hay factores comúnes de estudio entre las diferentes ramas de la medicina, que atienden al patrón utilizado en medicina del modelo científico: Se parte de una hipótesis condicionada a una serie de premisas-condiciones, se fuerzan las condiciones y

se observa en que porcentaje se cumple la hipótesis. Estos factores comunes fueron:

- Enfermedad: Es la hipótesis de partida, un paciente sufre una enfermedad en medida a sus síntomas.
- Causas o síntomas de la enfermedad: Como se manifiesta la enfermedad en los sujetos de estudio
- Medicamentos asociados a la enfermedad: Control de los agentes externos que puedan interferir o dar falsos positivos.
- Pruebas asociadas a la enfermedad: Una enfermedad puede tener una o varias formas de manifestarse, por lo que las pruebas son las diferentes comprobaciones a realizar sobre las diferentes síntomas de la enfermedad.
- Pacientes enfermos: Pacientes que se sabe que padecen la enfermedad.
- Pacientes de riesgo: Pacientes con posibilidades de sufrir la enfermedad.
- Pacientes de control: Pacientes sin aparentes posibilidades de sufrir la enfermedad.

3.6. Modelado de los datos

El manejo de los pacientes, tras la observación de los datos que de ellos poseemos, lo haremos a través de la base de datos MongoDB, es una base de datos no relacional especializada en guardar grandes volúmenes de datos. Por este motivo es por el que es la base de datos que usaremos para el desarrollo del estudio, debido a que hay pruebas que son grandes volúmenes de datos.

Principalmente vamos a manejar 4 tablas relacinadas entre sí: Medicamentos, Enfermedades, Pacientes y Pruebas. Este manejo de las cuatro tablas también es con la finalidad de hacer las *querys* más ágiles para los tiempos de carga de la aplicación web. Pero aún así y para mantener la filosofía de las bases de datos no relacionales y a pesar de crear duplicidades en la base de datos, el esquema de pacientes mantendrá a través de campos ocultos los valores del resto de los esquemas. Esto además nos proporciona la posibilidad de exportar el estudio solo con una tabla.

A pesar de que estemos viendo el esquema como un esquema clásico ER, por debajo MongoDB no lo está usando para su modelado, por lo que las restricciones derivadas de las bases de datos relacionales no se nos presentarán. En MongoDB el formato que en el que se guardan los datos, no es el

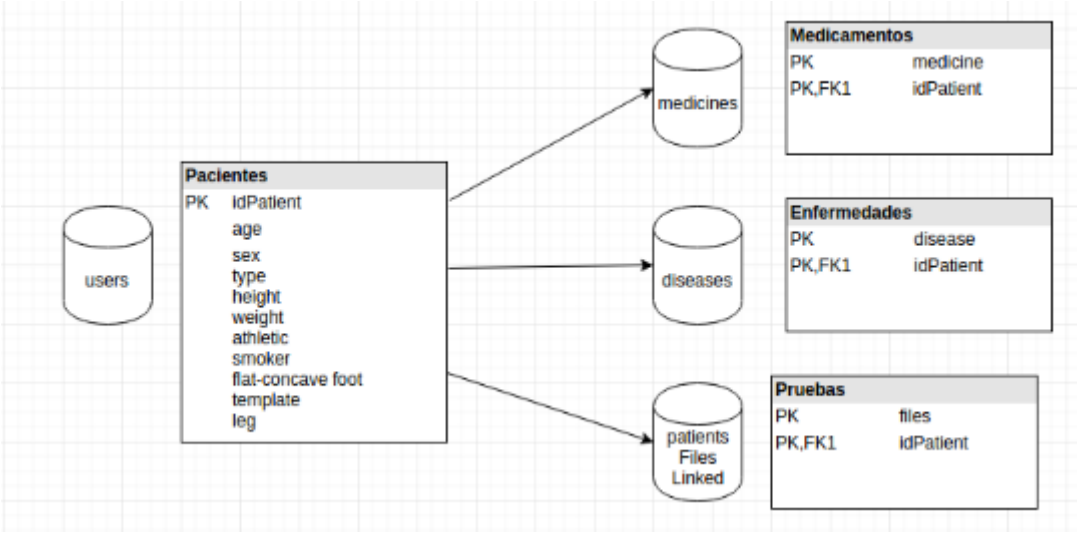

Figura 3.2: Esquema base de datos

clásico de SQL sino que usa un formato propio parecido a JSON, por lo que la referencia al diseño en SQL es solo para hacerlo más inteligible.

3.7. Estructura de datos

3.7.1. Medicamentos

La tabla de medicamentos, no existirá como tal en MongoDB sino que serán todos los documentos guardados con el formato:

```
medicine_doc = {
"medicine": "",
"username": ""
}
```

En SQL, para el manejo de la tabla de medicamentos tendríamos la tabla definida de la siguiente forma:

```
CREATE TABLE medicines(
medicine VARCHAR(20),
username VARCHAR(20),
PRIMARY KEY(medicine,username));
```

Si realizamos una comparación en el código, según el tipo de base de datos, se ve que al usar la tecnología de MongoDB el formato que usamos es ya directamente un JSON. La ventaja principal es que al ser directamente

JSON podemos exportarlo más fácilmente y a más plataformas, y con menos tiempo de procesamiento, por lo que ganamos en rápidez y audiencia. Además también podremos usar *string* ilimitados, mientras que para SQL tendremos que definir la longitud previamente.

3.7.2. Enfermedades

La tabla de enfermedades, no existirá como tal en MondoDB sino que serán todos los documentos guardados con el formato:

```
disease_doc = {
"disease": "",
"username": ""
}
```

En SQL, para el manejo de la tabla de medicamentos tendríamos la tabla definida de la siguiente forma:

```
CREATE TABLE diseases(
disease VARCHAR(20),
username VARCHAR(20),
PRIMARY KEY(disease,username));
```

Al igual que con el apartado de los medicamentos, tenemos la misma ventaja debida al uso del JSON.

3.7.3. Pruebas

La tabla de pruebas, no existirá como tal en MondoDB sino que serán todos los documentos guardados con el formato:

```
test_doc = {
"test": "",
"username": ""
}
```

En SQL, para el manejo de la tabla de pruebas tendríamos la tabla definida de la siguiente forma:

```
CREATE TABLE tests(
test VARCHAR(20),
username VARCHAR(20),
PRIMARY KEY(test,username));
```

Al igual que en los apartados anteriores, tenemos la misma ventaja debida al uso del JSON.

3.7.4. Pacientes

La tabla de pacientes, no existirá como tal en MondoDB sino que serán todos los documentos guardados con el formato:

```
patient_doc = {
"idPatient": "",
"age": "",
"sex": "",
"type": "",
"height": "",
"weight": "",
"athletic": "",
"smoker": "",
"flat-concave foot": "",
"template": "",
"leg": "",
"_medicines":[],
"_diseases":[],
"_test":[]
}
```

En SQL, para el manejo de la tabla de medicamentos tendríamos la tabla definida de la siguiente forma:

```
CREATE TABLE patients(
idPatient VARCHAR(20),
age VARCHAR(20),
sex VARCHAR(20),
type VARCHAR(20),
height INT,
weight INT,
athletic BOOLEAN,
smoker BOOLEAN,
flat-concave foot VARCHAR(20),
template BOOLEAN,
leg VARCHAR(20),
PRIMARY KEY(idPatient);
```

Al igual que en los apartados anteriores, tenemos la misma ventaja debida al uso del JSON. Además en este caso en particular al ser una tabla más compleja con diferentes tipos de datos, gracias al JSON podremos usar tipos más convencionales en los lenguajes de programación a alto nivel, sin necesidad de tener que hacer adaptadores entre el programador en Python y el lenguaje SQL.

Capítulo 4

Creación del servidor

Aquí vamos a describir la elección del servidor y su funcionalidad. En nuestro caso he elegido Django server porque para el manejo de MongoDB se usa Python, y Django server está escrito en Python. Por lo que usando el mismo lenguaje y la misma versión de este: 2.7 evitamos posibles problemas de compatibilidad.

4.1. Configuración del servidor

Con el fin de que la aplicación sea *portable* en la configuración de los ajustes del servidor (settings.py) tendremos una serie de variables las cuales habrá que ajustar si migramos de máquina. Ejemplo de configuración local de las variables:

```
TEMPLATE_DIRS = ('templates/',)
API_MONGO_DIR = '/home/jorge/tfg/connect_python_to_mongo/'
HOST_MONGO = 'localhost'
PORT_MONGO = 27017
FILES_PATIENT_UPLOAD = '/home/jorge/tfg/uploaded_files'
API_STATISTICS_DIR = '/home/jorge/tfg/statistics_functions/'
DISEASES_TXT = '/home/jorge/tfg/diseases.txt'
MEDICINES_TXT = '/home/jorge/tfg/medicines.txt'

STATIC_URL = '/static/'
STATIC_URL_CSS = 'templates/css/'
STATIC_URL_JS = 'templates/js/'
STATIC_URL_IMAGES = 'templates/images/'

MEDIA_ROOT = '/home/jorge/tfg/uploaded_files'
```

Significado de las variables para su uso en la exportación:

- TEMPLATE_DIRS: Ruta donde se encuentren las plantillas usadas para la presentación de la aplicación de cara al usuario, el código HTML.
- API_MONGO_DIR: Ruta donde se encuentre la librería creada para el manejo de la base de datos creada para la aplicación.
- HOST_MONGO: Máquina donde esté ubicada la base de datos de la aplicación.
- PORT_MONGO: Puerto habilitado para atender peticiones en la base de datos.
- API_STATISTICS_DIR: Ruta donde se encuentre la librería creada para el manejo de la obtención de valores estadísticos en los gráficos para el servidor.
- DISEASES_TXT: Ruta completa, incluyendo nombre del archivo donde se encuentre el listado de enfermedades para la aplicación.
- MEDICINES_TXT: Ruta completa, incluyendo nombre del archivo donde se encuentre el listado de medicamentos para la aplicación.
- STATIC_URL: Por defecto en los servidores de Django, para sus propias aplicaciones de monitoreo. Para esta aplicación no es necesaria su modificación. Valor por defecto: '/static/'
- STATIC_URL_CSS: Ruta donde se encuentran todos las librerías de estilo del código HTML del servidor, es decir la ruta del código CSS.
- STATIC_URL_JS: Ruta donde se encuentran todos las librerías de código JavaScript del servidor.
- STATIC_URL_IMAGES: Ruta donde se encuantran las imágenes necesarias para la renderización del servidor.
- MEDIA_ROOT: Ruta donde se guardarán los archivos subidos al servidor.

4.2. Inserción de los datos de los pacientes

4.2.1. Automatización datos pacientes

Para la automatización de la carga de datos de los pacientes ya existentes, se han convertido los documentos de estilo excel proporcionados por el estudio

al formato csv, ya que dicho formato es el más usado para la exportación de datos y en competiciones de maching learning.

4.2.2. Inserción de nuevos pacientes

La insercción de nuevos pacientes se hace a través de la interación entre servidor y usuario. Esto es posible porque el servidor proporciona un formulario con los datos necesarios que se explica en el apartado WEB.

4.3. Interacción Servidor Usuario

Para ver como interactuá el usuario con el servidor, primero vamos a ver que recursos es capaz de resolver el servidor y posteriormente a través de que interfaz hace uso de ellos.

Primero describimos las URL a las que puede acceder el usuario a lo largo de la navegación y la función de cada una. Las URLs están disponibles en urls.py, que es la parte encargada de gestionar las urls a las que da acceso en el servidor:

```
urlpatterns = [
url(r'^admin/', include(admin.site.urls)),
url(r'^css/(?P<path>.*)$', 'django.views.static.serve', {'
    document_root' : settings.STATIC_URL_CSS}),
url(r'^images/(?P<path>.*)$', 'django.views.static.serve', {'
    document_root' : settings.STATIC_URL_IMAGES}),
url(r'^js/(?P<path>.*)$', 'django.views.static.serve', {'
    document_root' : settings.STATIC_URL_JS}),
url(r'^csv/(.*)$', 'serverapps.views.downloadCsv'),
url(r'^test/(.*)$', 'serverapps.views.downloadMedicalTest'),
url(r'^formSingupPatient(.*)$', 'serverapps.views.
    formSingupPatient'),
url(r'^formSearchPatient(.*)$', 'serverapps.views.
    formSearchPatient'),
url(r'^formCalculateRisk(.*)$', 'serverapps.views.
    formCalculateRisk'),
url(r'^formDeleteUser(.*)$', 'serverapps.views.formDeleteUser'),
url(r'^restartUsers(.*)$', 'serverapps.views.formRestartUsers'),
url(r'^$', 'serverapps.views.index'),
]
```

URL	Función
admin/	Permite gestionar los parámetros del servidor a través de la página de admin de django
css/<css file>	Gestiona las peticiones de los css al servidor
images/<image>	Gestiona las peticiones de imágenes al servidor
js/<js file>	Gestiona las peticiones de los javascripts al servidor
csv/<user Id>	Url a la cual se nos redigirá a la hora de pedir un csv, la cual nos devolverá el csv correspondiente
formSingupPatient	Url la cual gestiona el alta de un paciente en la aplicación
formSearchPatient	Url que nos devolverá el paciente del cual queremos que se nos muestre su información
formCalculateRisk	Url que nos llevará a la parte de la aplicación destinada a la aplicación de los modelos de Maching Learning diseñados
formDeleteUser	Permite borrar usuarios de la base de datos
restartUsers	Permite resetear la base de datos de usuarios
/	Recurso principal

Tabla 4.1: Urls que resuelve el servidor

4.4. Interacción Servidor Base de Datos MongoDB

A continuación describimos como interactuá el servidor con la base de datos de MongoDB, en este caso no he decidido integrar directamente la base de datos en el servidor por varios motivos:

- Queremos que la aplicación sea *portable* independientemente del servidor, lo que nos obliga a definir una API para hablar con la base de datos.
- Que la base de datos ella por si sola pueda migrarse a cualquier equipo, si estuviera integrada nos obligaría a llevarnos el servidor entero.

4.4.1. API creada para el manejo de la base de datos:

Se han implementado las siguientes funciones para el manejo de la base de datos. En las tablas 4.2, 4.3, 4.4 y 4.5 se detalla el cometido de cada función, los parámetros de entrada y el tipo de esto, y en caso de esperar un resultado cual será y su formato.

- openConectionMDB(host,port)
- tabla de pacientes:
  - addPatientMDB(dbh,username,dictDoc)
  - checkPatientMDB(dbh,username)
  - deletePatientMDB(dbh,username)
  - deleteAllPatientsMDB(dbh)
  - getPatientMDB(dbh,username)
  - getNextIdPatientMDB(dbh)
  - getAllPatientWithMDB(dbh,query)
  - updatePatientMDB(dbh,username,updates)
- tabla de medicamentos:
  - addMedicineMDB(dbh,idPatient,dictDoc)
  - deleteAllMedicinesMDB(dbh)
  - getAllMedicinesMDB(dbh)
  - getPatientMedicinesMDB(dbh,idPatient)
  - getNumberOfPatientsMedicineMDB(dbh,medicine)
  - loadindAllMedicinesMDB(dbh,medicinesDoc)
- tabla de enfermedades:
  - addDiseaseMDB(dbh,idPatient,dictDoc)
  - deleteAllDiseasesMDB(dbh)
  - getAllDiseasesMDB(dbh)
  - getPatientDiseasesMDB(dbh,idPatient)
  - getNumberOfPatientsDiseaseMDB(dbh,disease)
  - loadindAllDiseasesMDB(dbh,diseasesDoc)
- tabla de pruebas:
  - addFileToPatientMDB(dbh,idPatient,filename, idFile)
  - getFileMDB(dbh,file_id)
  - getDictFilesIdsMDB(dbh,idPatient)
  - insertFileMDB(idPatient, filename,dbh)

4.5. Código colores depuración

Para mantener un orden a la hora de saber que está pasando en el servidor durante el procesamiento de las peticiones entre cliente y servidor, servidor y base de datos, y servidor y modelos ML, he desarrollado un código de colores usando la librería de termcolor para python. Con esto lo que conseguimos es que dependiendo del tipo de petición o proceso que estemos atendiendo podamos saber rápidamente de que se trata. Para ello lo primero que he hecho ha sido la integración de la librería colored:

```
from termcolor import colored
```

A continuación se detallan los colores usados en relación con su origen:

- Amarillo: Cualquier mensaje de debug que venga de los métodos para la integración de MongoDB en el servidor.
- Rojo: Para los mensajes de debug que vengan de los métodos que integren los modelos ML en el servidor.
- Aqua: Aparecerá cuando el servidor recibe una petición sobre algún recurso web de carga de la página como los JS o CSS, además de las imágenes.
- Morado: En este color se mostrarán los errores que se produzcan en el servidor referentes a la plataforma que lo integra que es Django.
- Verde: Procesamiento de las peticiones en el servidor
- Por Defecto: django

Método o función	**Entradas**	**Return**	**Cometido**
addPatientMDB	• dbh • username • dictDoc	not return	añadir un paciente a la base de datos
checkPatientMDB	• dbh • username	boolean	comprobar si existe un paciente por su id
deletePatientMDB	• dbh • username	not return	borrar un paciente
deleteAllPatientsMDB	• dbh	not return	borrar todos los pacientes de la base de datos
getPatientMDB	• dbh • username	JSON paciente	devuelve la información de un paciente
getNextIdPatientMDB	• dbh	string	devuelve el siguiente id correspondiente para la tablad e pacientes
getAllPatientWithMDB	• dbh • query	[JSON pacientes]	devuelve todos los pacientes coincidentes a una query
updatePatientMDB	• dbh • username • updates	not return	actualiza la información de un paciente

Tabla 4.2: API pacientes

Nombre del método o función	**Parámetros de entrada**	**Return**	**Cometido**
addMedicineMDB	• dbh • idPatient • dictDoc	not return	añade la relación entre paciente y medicamento
deleteAllMedicinesMDB	• dbh	not return	vacía la tabla de datos de medicamentos
getAllMedicinesMDB	• dbh	[JSON Medicamentos]	devuelve todas medicinas diferentes ingresadas en la base de datos
getPatientMedicinesMDB	• dbh • idPatient	[JSON Medicamentos]	devuelve los medicamentos asociados a un paciente
getNumberOfPatientsMedicineMDB	• dbh • medicine	int	devuelve cuantos pacientes usan un medicamento
loadindAllMedicinesMDB	• dbh • medicinesDoc	not return	permite la carga de medicamentos a través de un fichero

Tabla 4.3: API medicamentos

Nombre del método o función	**Parámetros de entrada**	**Return**	**Cometido**
addDiseaseMDB	• dbh • idPatient • dictDoc	not return	añade la relación entre un usuario y una enfermedad
deleteAllDiseasesMDB	• dbh	not return	vacía las relaciones entre pacientes y enfermedades
getAllDiseasesMDB	• dbh	[JSON Enfermedades]	devuelve todas enfermedades diferentes ingresadas en la base de datos
getPatientDiseasesMDB	• dbh • idPatient	[JSON Enfermedades]	devuelve las enfermedades asociadas a un paciente
getNumberOfPatientsDiseaseMDB	• dbh • disease	int	devuelve el número de paciente que sufre una enfermedad
loadindAllDiseasesMDB	• dbh • diseasesDoc	not return	permite la carga de medicamentos a través de un fichero

Tabla 4.4: API enfermedades

Nombre del método o función	**Parámetros de entrada**	**Return**	**Cometido**
addFileToPatientMDB	• dbh • idPatient • filename • idFile	not return	añade una prueba asociada a un paciente
getFileMDB	• dbh • file_id	Prueba Médica	Devuelve una prueba médica
getDictFilesIdsMDB	• dbh • idPatient	ids:filesName	devuelve las pruebas asociadas a un usuario
insertFileMDB	• idPatient • filename • dbh	int	inserta un fichero en la base de datos

Tabla 4.5: API pruebas

Capítulo 5

Desarrollo de la página web

El desarrollo de la parte visual o web del servidor se ha hecho con HTML5, CSS3 y JavaScript, ya que nos permite ajustarnos a las demandas actuales sobre que la Web sea responsiva de las páginas web, y además para la representación de las gráficas de los resultados de los pacientes es más limpio. Para la realicación de la parte visual me he apoyado en las librerías de Bootstrap, JQuery y JQuery-UI.

5.1. Responsividad página web

La página web se ha diseñado conforme a la visión actual sobre los servicios web, es decir: "Los móviles primero", ya que hoy día la gente tiene o usa más el móvil que el ordenador. Esto nos lleva a que el diseño de la página esté adaptado a los móviles y sus diferentes tamaños de pantalla, para ello he usado las diferentes librerías nombradas al principio del capítulo de la siguiente forma:

- La librería de Bootstrap se ha usado con la finalidad de mantener la página ordenada para los posibles tamaños de pantalla. Gracias a su principal función de división de la página en una cuadricula de doce elementos, conseguimos que cuando el número de elemento que caben en pantalla sea menor su distrubución se lleve a cabo.

- Las librerías de JQuery o JQuery-UI se han usado para generar animaciones de transito entre eventos relacionados con el uso de la página web, de manera que según el tamaño también tiene en cuenta como redistribuir la información o en este caso las animaciones.

5.2. Obtención de datos del servidor

La obtención de datos del servidor para la representación de los gráfico o la muestra de datos de un usuario se hace de forma empotrada en el HTML, con el fin de evitar crear grandes documentos de descargas adicionales debído a que el manejo de datos que hacemos es excesivamente grande como para que la descargar a parte de dichos datos no suponga un aumento de la latencia en el servicio a proveer.

5.3. Insercción de gráficos

Para la insercción de gráficos en el servidor he usado la librería de highcharts, que nos pérmite la creación de nuestra propia API de gráficos. Se han implementado las siguientes funciones. En la tabla 5.1 se detalla el cometido y parámetros de las funciones.

- boxplotGraph = function(id,title,xCategories,xTitle,yAxis,mean)
- barGraph = function(id,title,subtitle,yTitle,seriesVal)
- histoGraph = function(id,title,dataArray)

5.4. Interacción del usuario vía WEB

Como el usuario a la hora de interactuar con el servidor lo hará a través de los formularios y de la página principal, esto le permitirá una navegación por las diferentes secciones de la misma, se describen a continuación.

- Página principal:

Figura 5.1: Página principal del interfaz del usuario. Recurso: Index.html

Aunque nos de la sensación de estar en otra página diferente seguimos en la página principal, pero se irán mostrando las diferentes pestañas según la necesidad del usuario. La página principal nos permite una visión global de las funcionalidades a través del carrousel disponible en la parte inferior.

- Buscar paciente: La página de buscar paciente nos permite diversas funcionalidades, la primera y por la cual recibe el nombre es la de obtener todos los datos del paciente en cuestión que queramos relacionado con el estudio. Como funcionalidades añadidas nos permite también obtener un CSV con todos los pacientes del estudio, de tal forma que podamos exportar para trabajar, los datos del mismo de una forma cómoda y versátil. Y por último nos permite reiniciar los datos por si queremos reutilizar la estructura de este.

Figura 5.2: Página de busqueda de pacientes. Recurso: searchPatient.html

- Calcular riesgo: La página del calcular riesgo, nos permite seleccionar un modelo de ML previamente desarrollado, con el cual evaluar la probabilidad de que un paciente sufra o no EAP. Además y posteriormente nos permitirá redirigirnos a la página del paciente en cuestión para observar sus condicionantes.

Figura 5.3: Página de selección de evaluación de los pacientes. Recurso:riskPatient HTML

- Gráficos del estudio: En esta pestaña podemos seleccionar entre diferentes gráficos que proporcionamos para tanto la monitorización de los datos del estudio en tiempo real, como para posteriormente la toma de decisiones sobre el estudio y formulación de hipótesis.
  - Gráfico de barra sobre los medicamentos:

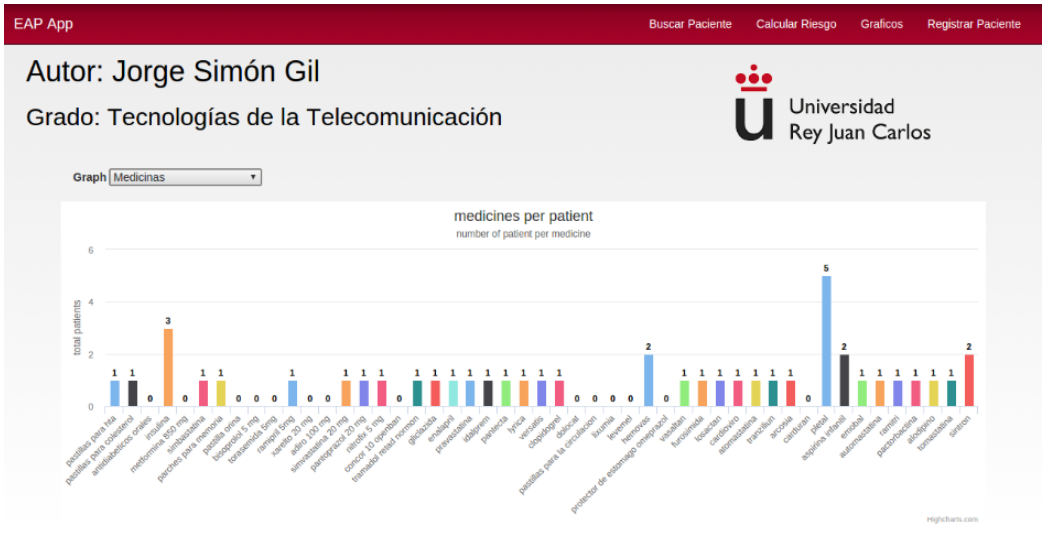

Figura 5.4: Gráfico sobre los medicamentos del estudio.

  - Gráfico de barra sobre las enfermedades:

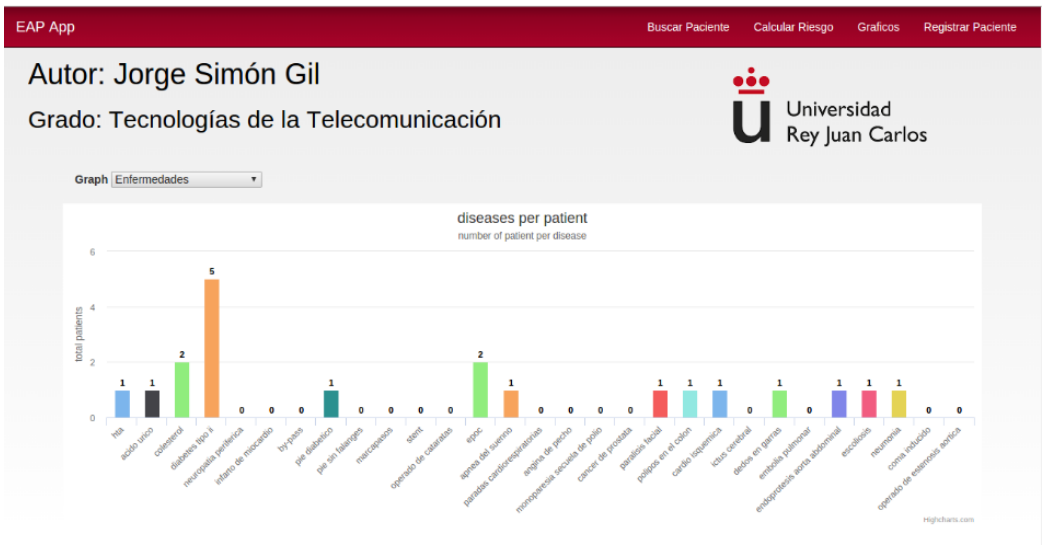

Figura 5.5: Gráfico sobre las enfermedades del estudio.

- Gráfico de barras sobre los tipos de pacientes:

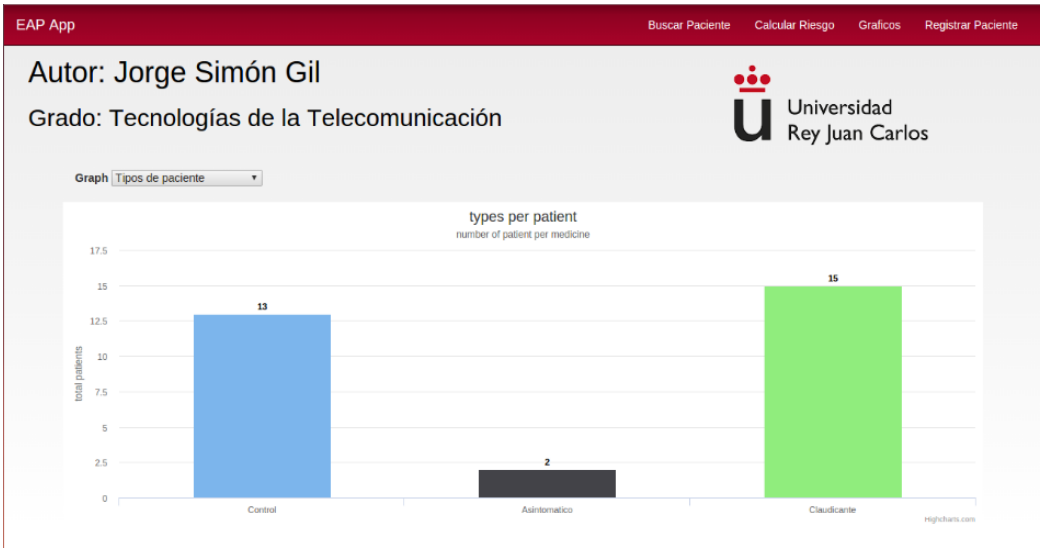

Figura 5.6: Gráfico sobre los pacientes del estudio.

- Boxplot sobre la edad y el sexo de los pacientes:

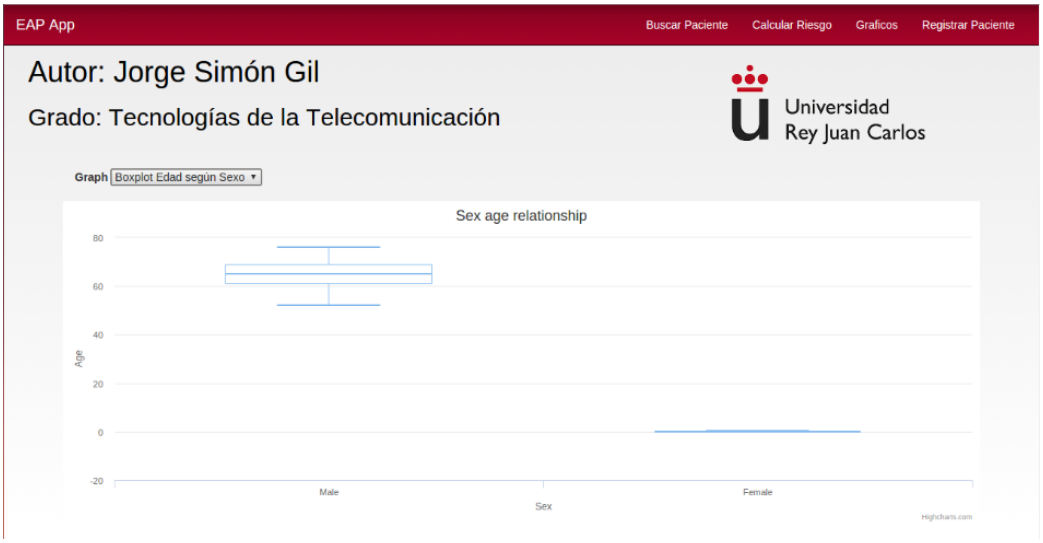

Figura 5.7: Boxplot sobre la edad en función del sexo

- Histograma de edades:

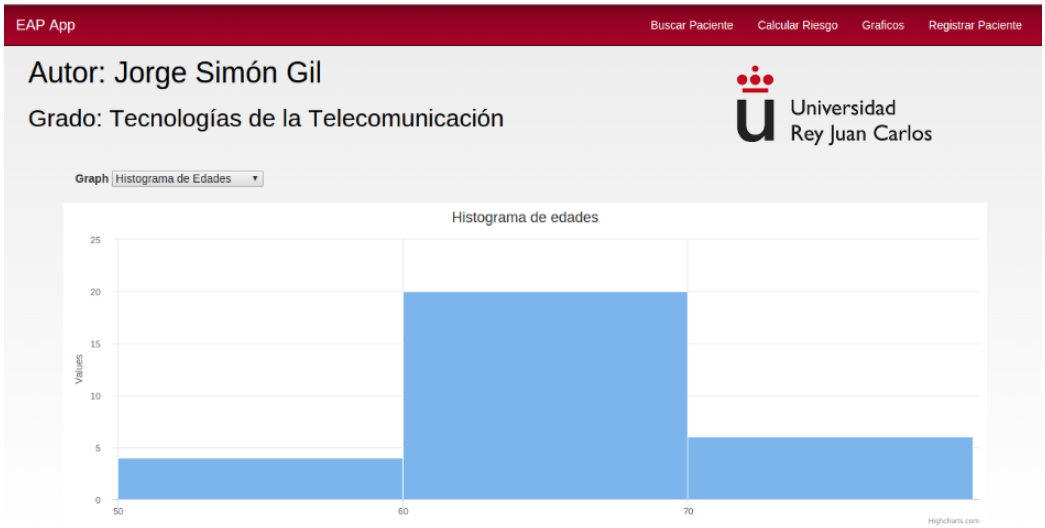

Figura 5.8: Histograma de edades

- Histograma de alturas:

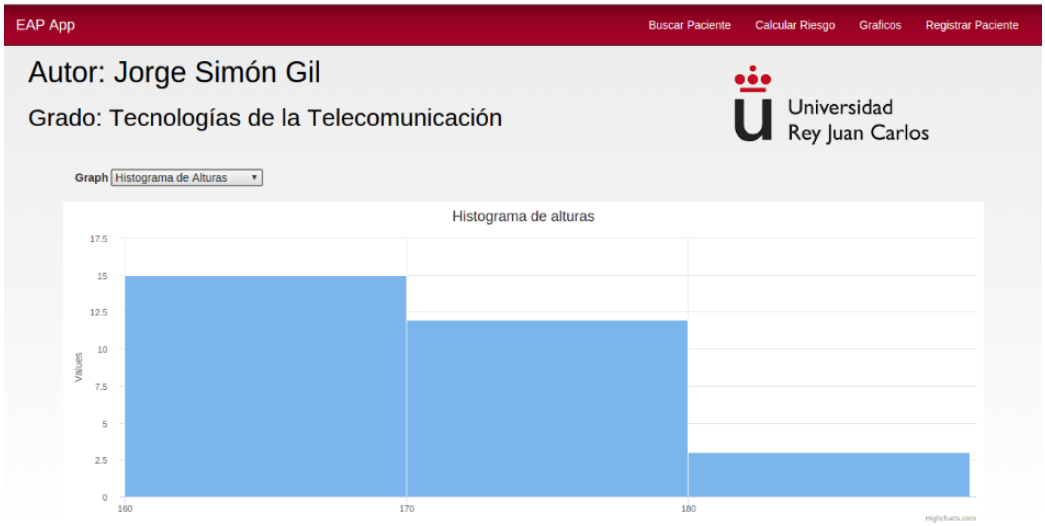

Figura 5.9: Histograma de alturas

- Histograma de pesos:

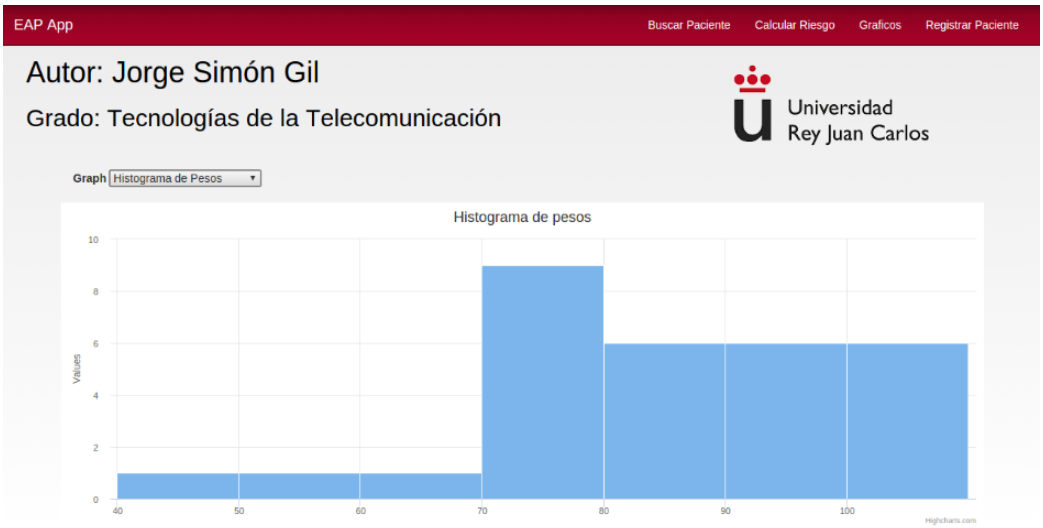

Figura 5.10: Histograma de pesos

- Registrar paciente: Pestaña que nos permite registrar un paciente para el estudio. El formulario está dividido en dos partes, una con datos más generales y una segunda con las enfermedades, medicamentos y datos relativos al estudio de la EAP.

Figura 5.11: Primera parte del formulario. Recurso singupPatient.html

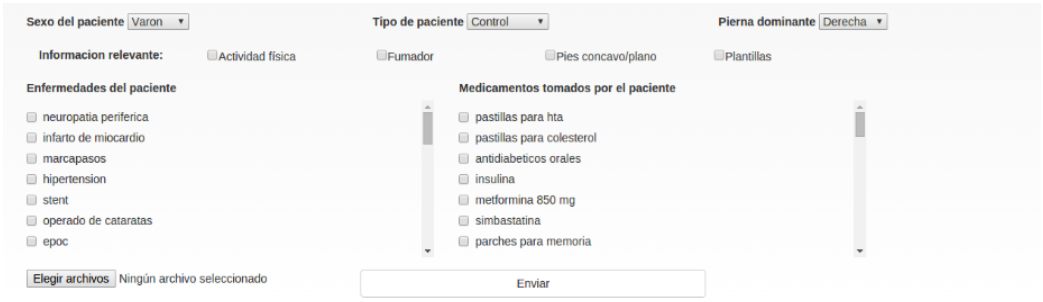

Figura 5.12: Segunda parte del formulario. Recurso singupPatient.html

- Página de un paciente: Página a la cual accedemos tras rellenar el formulario de búsqueda de paciente. Nos proporciona una visión global de los datos del paciente, además de exportarlo en CSV, adjuntar más pruebas o borrar el usuario.

Figura 5.13: Página resumen sobre los datos del paciente. Recurso: Patient.html

- Riesgo de un paciente: Página que en función del modelo de ML seleccionado nos dirá la probabilidad de sufrir EAP.

Figura 5.14: Recurso: Risk.HTML

Nombre del método o función	**Parámetros de entrada**	**Cometido**
boxplotGraph	▪ id ▪ title ▪ xCategories ▪ xTitle ▪ yAxis ▪ mean	Dibuja un gráfico del tipo boxplot
barGraph	▪ id ▪ title ▪ subtitle ▪ yTitle ▪ seriesVal	Dibuja un gráfico de barras
histoGraph	▪ id ▪ title ▪ dataArray	Dibuja un gráfico del tipo histograma

Tabla 5.1: API gráficos

Capítulo 6

Estimación

Se proporciona la aplicación de herramienta de ML o Deep Learning integrada en el servidor, es un apartado en el cual pueda integrar los diferentes modelos para posteriormente.

La manera de integrar un modelo de ML con el servidor es de la siguiente forma:

Primero desarrollar el modelo en cuestión y un CSV con los datos del usuario. Dicho CSV será el previamente configurado para el servidor, el de por defecto configurado para la EAP dispone de lo siguiente:

- ID: Será el ID del paciente en cuestión.
- Age: Este valor corresponderá a la edad del paciente.
- Sex: Está columna hace referencia al género biológico del paciente. Será una variable categórica entre varón y hembra.
- Type: Representa la variable categórica correspondiente a la parte de población a la que pertenece el paciente. Control, Asintomático y Claudicante.
- Weight: Hace referencia al peso del paciente.
- Height: Hace referencia a la altura del paciente.
- Athletic: Representa una variable booleana la cual representa si el paciente se encuentra en un buen estado de forma física o no.
- Smoker: Representa una variable booleana entre si fuma o no.
- Foot: Representa la variable categórica correspondiente al tipo de pie del paciente. Cóncavo o plano.

- Template: Representa una variable booleana entre si usa plantillas al caminar o no.
- Leg: Representa la variable categórica correspondiente a si el paciente es zurdo o diestro. Izquierda o Derecha.

Devolverá un único valor, que será la probabilidad del estimador o decisor. En caso de necesitar analizar valores de las pruebas en concreto, tendrá que realizarlo a través de la petición del navegador o trabajando directamente con la API.

Posteriormente añadir el modelo como si fuera un método en el archivo *modelsML.py* ubicado en *<ruta de la descarga del proyecto>/statistics_functions/*, y dar de alta su nombre en el método getModelsName, además de en el calculateRisk. Sino se da de alta el nombre del modelo no se podrá consultar a través del navegador.

Ejemplo básico de inserción de un modelo:

```
def getModelsName():
models = "<option value='dummyModel1' selected='selected'>
    dummyModel1</option>"
models += "<option value='dummyModel2'>dummyModel2</option>"
return models

def caculateRisk(userDoc,modelMLName):
if(modelMLName == "dummyModel1"):
return dummyModel1(userDoc)
else:
return dummyModel()

def dummyModel():
return 0.5

def dummyModel1(userDoc):
return 0.1
```

Los mensajes de debug de modelsML.py deberán ir en color rojo usando la librería de colored de termcolor para python. Ejemplo de mensaje de depuración:

```
msg_dubug = "esto es un mensaje de depuraci'on"
print colored(msg_dubug,'red')
```

Capítulo 7

Conclusiones y líneas futuras

En este capítulo vamos a recapitular lo que he aprendido a lo largo del desarrollo del Trabajo de Fin de Grado, la arquitectura del sistema desarrollada y cuales van a ser los siguientes puntos a abordar como mejoras.

7.1. Conclusiones

Sí retomamos el punto de partida de partida al comienzo del TFG y evaluamos el desarrollo obtenido sobre la aplicación, podemos considerar que el proceso hasta finalizar la aplicación ha sido satisfactorio. Ha sido satisfactorio tanto por lo aprendido en proceso como por los requisitos que he logrado hasta tener una aplicación estable.

En cuanto a lo aprendido, la base tecnológica constaba del conocimiento de Python, Django, Tecnologías Web, diseño básico de datos y Maching Learning, es decir del lenguaje, la base del servidor y como hacer interactuar al servidor con el usuario. A lo largo del desarrollo de la plicación y por las necesidades para implementar una solución ágil al problema que se nos planteó al inicio: Construir una aplicación que sirviera para el desarrollo de estudios estadísticos, he aprendido no solo relacionado con los elementos que constituyen la aplicación sino que también he aprendido:

- Conocimiento biomédico adquirido: Durante el desarrollo de la plicación he aprendido a observar, recabar, analizar, filtrar y trabajar, información adquirida de pacientes.

- Conocimiento médico adquirido: El conocimiento médico adquirido está relacionado con los sistemas endocrino, vascular y respiratorio, ya que durante todo el proceso de observación de los datos, estos estaban

relacionados con personas que padecían enfermedades en los diferentes sistemas.

- Conocimiento en bases de datos no relacionales adquirido: Al principio se nos planteó el problema de como crear y manejar datos de forma ágil, lo que nos llevo tras la etapa de investigación y observación de otras soluciones para problemas similares a la decisión de que teníamos que usar bases de datos no relacionales. Posteriormente tuvimos que elegir una y en concreto fue MongoDB. Durante este proceso se obtuvo el conocimiento necesario para manegar MongoDB e integrarlo con Python para su uso en el servidor, además de como diseñar el modelo de datos en base de datos no relacionales.

- Conocimiento adquirido sobre ML: Una de las partes fundamentales era la integración de ML dentro del servidor para poder evaluar los datos, para ello fué necesario ampliar conocimiento sobre como interactúan las librerías más usadas: pandas, sklearn y nunpy, principalmente. Saber que datos esperan y como los trabajan. Para ello y aprovechando la situación se asistió al curso impartido por la URJC: *Datatón URJC 2017*

- Conocimiento sobre diseño HTML: Por último se debía presentar la herramienta de una forma fácil de usar y que además resultase amigable, para ello se usó HTML5, CSS3 y Javascript. Sobretodo hubo que aprender en relación con la integración y diseño de los gráficos, y proporcionar una interfaz útil y completa para la monitorización y gestión de usuario.

La adquisición de este conocimiento me ha llevado al desarrollo de la arquitectura del sistema que vemos a continuación:

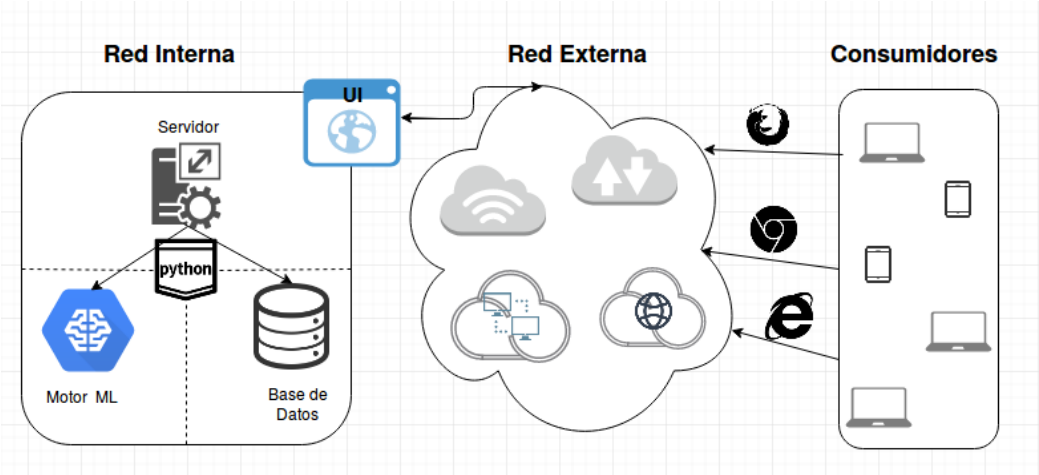

Figura 7.1: Arquitectura sistema.

Como se puede observar la arquitectura generada consta de tres partes:

- Consumidores: Tendremos una serie de usuarios de nuestra aplicación, que a través de sus dispositivos haciendo uso de un navegador podrán consumir nuestros recursos.

- Red Externa: Esta será la red en la que expongamos nuestros recursos para ser consumidos. Puede ser tanto inalámbrica como cableada, local como externa.

- Red Interna: Esta será la red que nosotros generaremos con el despliegue de la aplicación. Donde tendremos el servidor, la base de datos y el motor de ML, destacar que los tres módulos no tienen porque estar en la misma máquina física.

En definitiva el desarrollo de la plicación ha sido mi propio desarrollo como creador de ella. Durante la presentación del TFG se pretende además realizar una demo abordando todas las capacidades actuales de la misma.

7.2. Líneas futuras

Debido a la extensión del proyecto ha habido temas relacionados que no se han podido abordar para el Trabajo de Fin de Grado, y que se van a usar como punto de partida para el Trabajo de Fin de Máster. Estos son:

- Pruebas masivas: Como se ha realizado todo el desarrollo con un entorno controlado y cerrado, no se ha probado la estabilidad y funcionalidad con cargas masivas de datos. Esto puede hacer que a la hora de ser usado como herramienta, de manera abierta y global, ocurran problemas con los procesamientos asíncronos o múltiples entre las peticiones y cargas en la base de datos.

- Desarrollo de un estudio de ML sobre la EAP: A causa del desarrollo de toda la aplicación, no se ha podido realizar un estudio de ML sobre la EAP por lo que para el Trabajo de Fin de Máster se quiere abordar este punto, además así se pueden aprovechar los conocimientos adquiridos en este área durante el máster.

Bibliografía

[1] Óscar Barquero Pérez, *HEART RATE VARIABILITY:A FRACTAL ANALYSIS*, FEUP Universidad de Oporto , Junio 2008

[2] Rebeca Goya Esteban, *Heart Rate Variability Characterization Using Entropy Measures*, FEUP Universidad de Oporto , Junio 2008

[3] J. Thomas Bigger, GÃijnter Breithardt, Sergio Cerutti, Richard J. Cohen y Philippe Coumel, *Guidlines Heart rate variability*, European Heart Journal , 1996

[4] José Luis Rojo, J Naranjo Orellana,, *PROCESADO DIGITAL NO LINEAL Y APRENDIZAJE ESTADISTICO CON NUCLEOS AUTOCORRELACION PARA APLICACIONES EN EL SECTOR SALUD*, Ministerio de Ciencia e Innovación España, 2013

[5] Lucrecia Herranz de la Morena, *ÃDndice tobillo brazo para la evaluación de la enfermedad arterial periférica*, Unidad de Diabetes, Servicio de Endocrinología y Nutrición, Hospital Universitario âĂIJLa PazâĂİ , Madrid , 2005

[6] Leicht A.S., Crowther R.G., Golledge J., *Influence of peripheral arterial disease and supervised walking on heart rate variability.*, J Vasc Surg. , 2011

[7] Gavin R.H. Sanderock, Lynette D. Hodges, Saroj K. Das y David A. Brodie, *THE IMPACT OF SHORT TERM SUPERVISED AND HOME-BASED WALKING PROGRAMMES ON HEART RATE VARIABILITY IN PATIENTS WITH PERIPHERAL ARTERIAL DISEASE*, Journal of Sports Science and Medicine , 2007

[8] Dra. Marco Sanz, Carmen. Profesora Titular de Cinesiología *Marcha humana.pdf*,Universidad de Zaragoza Cinesiología cap. 6: Análisis cinesiológico de las destrezas motoras básicas. Tema 29: Cinesiología de la Marcha Humana Normal,2008

[9] Podoactiva: Clínica de Podología y Biomecánica *www.podoactiva.com*,Clínica Especialidada en el análisis de la marcha

[10] Agencia Europea de Medicamentos *medlineplus.gov* Agencia Europea de Medicamentos

[11] Agencia Española de Medicamentos y Productos Sanitarios *www.aemps.gob.es* Agencia Nacional de Sanidad

[12] Informe Anual Sistema Nacional de Salud *Inf_Anual_SNS_2015.1.pdf*, Agencia Nacional de Sanidad,2015

[13] Informe Anual de la OMS sobre diabetes *http://www.who.int/mediacentre/diabetes/*, Organización Mundial de la Salud,2016

[14] Informe Anual de la OMS sobre enfermedades Cardiovasculares *http://www.who.int/mediacentre/factsheets/fs317/es/*, Organización Mundial de la Salud,2016

[15] Informe Anual de la OMS sobre enfermedades Respiratorias *http://www.who.int/respiratory/es/*, Organización Mundial de la Salud,2016

[16] Origins and Objectives of Physionet *http://ieeexplore.ieee.org/abstract/document/932728/?part=1*, IEEE,2001

[17] Towards a Contextualization Solution for Platform Services, Django *http://ieeexplore.ieee.org/abstract/document/6133160*, IEEE,2011

[18] Python programming language) *colenak.ptkpt.net*, ptkpt, 29 September 2012

[19] Patterns and Processes for the Popular Document-Oriented Database *colenak.ptkpt.net*, Niall O'Higgins, 2011

[20] Near Infrared Spectroscopy (NIRS) Brain Imaging Laboratory *http://www.researchimaging.pitt.edu/content/near-infrared-spectroscopy-nirs-brain-imaging-laboratory* University of Pittsburgh,

Librerías programación

- xlrd:Para la lectura de los Xmls y conversión a CSV. *https://pypi.python.org/pypi/xlrd*
- highcharts: Para la insercción de gráficos en la página web *https://www.highcharts.com/*
- python 2.7.x *https://www.python.com/*
- django: Para la implementación del servidor *https://docs.djangoproject.com*
- termcolor: Mensajes de depuración *https://pypi.python.org/pypi/termcolor*

Glosario de términos

Términos médicos

- EAP: Enfermedad Arterial Periférica
- EPOC: Enfermedad Pulmonar Obstructiva Crónica
- VFC: Variabilidad de la Frecuencia Cardíaca
- HTA: Hipertensión Arterial
- PCR: Parada Cardiorespiratoria
- ACV: Accidente Cerebrovascular
- ITB: Índice tobillo brazo
- SOP: Síndrome de ovario poliquístico
- Edema: Retención de líquidos
- PAS: Presión arterial sistólica.
- NIRS: Near-infrared spectroscopy.
- AEMPS: Agencia Española de Medicamentos y Productos Sanitarios
- ERC: Enfermedad respiratoria crónica.
- PAS: Presión arterial sistólica.

Términos de la telecomunicación

- API: Application Programming Interface
- UI: User Interface
- DB: Data Base
- dbh: Data Base Handling
- ML: Maching Learning
- DBH: Data Base Handler

Apéndice

views.py

Este archivo es en el que se ha desarrollado toda la inteligencia del servidor, donde se encuentra el núcleo de la herramienta.

```
import sys
sys.path.append("/home/jorge/tfg/connect_python_to_mongo/")
sys.path.append("/home/jorge/tfg/statistics_functions/")

import csv
import zipfile
import StringIO
import os

from django.shortcuts import render
from django.http import HttpResponse,HttpResponseRedirect
from django.template.loader import get_template
from django.template import Context
from django.conf import settings
from django.views.decorators.csrf import csrf_exempt

from django.core.files.storage import FileSystemStorage
from django.core.files.base import ContentFile
from django.utils.encoding import smart_str

from api_MDB_server_django import *
from statistics_api import *
from modelsML import *

from termcolor import colored

# Create your views here.
################################## Others Methods
    ##################################

# change Form to python Dictionary
def parsedForm(dataForm):
```

```
form = dataForm.split("&")
Dictionary = {}
for f in form:
tupla = f.split("=")
Dictionary[str(tupla[0])] = str(tupla[1])
return Dictionary

def deleteDuplicates(array,key):
arrayNotDuplicated = []
for a in array:
disease = a[key]
#print colored(a[key],"green")
try:
arrayNotDuplicated.index(disease)
except (ValueError):
arrayNotDuplicated.append(disease)
return arrayNotDuplicated

def getLabelSelectForm(typeOption,index,value):
SelectedForm = '<div class="checkbox">'
SelectedForm += '<label><input type="checkbox" value="' + str(
    value) +'"'
SelectedForm += 'name="'+ str(typeOption) + str(index) +'">' +
    str(value) +'</label></div>'
return SelectedForm

def getEmptyParagraph(id,value):
paragraph = "<p id='" + id.replace(" ","_")
paragraph += "'>" + str(value)
paragraph += "</p>"
return paragraph

def getItemize(array):
itemize = "<ul>"
for a in array:
itemize += "<li>" + str(a) + "</li>"
itemize += "</ul>"
return itemize

def getForm(form,typeForm):
notDuplicated = deleteDuplicates(form,typeForm)
form = ""
index = 0
for nd in notDuplicated:
index += 1
form += getLabelSelectForm(typeForm,index,nd)
return form

def getValues(pForm,key):
```

```
values = []
for f in pForm:
if (str(f).find(key) != -1):
values.append(pForm[f].replace("+","␣"))
return values

def selectValue(arrayDic,key):
aux = []
for a in arrayDic:
aux.append(a[key])
return aux

def checkSingForm(pForm):
try:
pForm['smoker']
except KeyError:
pForm['smoker'] = False
try:
pForm['athletic']
except KeyError:
pForm['athletic'] = False
try:
pForm['flat-concave␣foot']
except KeyError:
pForm['flat-concave␣foot'] = False
try:
pForm['template']
except KeyError:
pForm['template'] = False
return pForm

def patientsToCsv(patients,writer):
for patient in patients:
writer.writerow([patient['idPatient'], patient['age'], patient['
    sex'], patient['type'], patient['weight'], patient['height'],
     patient['athletic'], patient['smoker'], patient['flat-
    concave␣foot'], patient['template'], patient['leg']])

def getTypesPatientsValues():
queryControlType = {'type':'Control'}
patientsControlType = getAllPatients(queryControlType)
controlTypeCount = len(patientsControlType)
querySintomaticType = {'type':'Asintomatico'}
patientsSintomaticType = getAllPatients(querySintomaticType)
sintomaticTypeCount = len(patientsSintomaticType)
queryClauType = {'type':'Claudicante'}
patientsClauType = getAllPatients(queryClauType)
clauTypeCount = len(patientsClauType)
paragraphs = getEmptyParagraph("Control",controlTypeCount) +
```

```
    getEmptyParagraph("Asintomatico",sintomaticTypeCount) +
    getEmptyParagraph("Claudicante",clauTypeCount)
return paragraphs

def getAllOneAtrib(query,typeAtrib):
patients = getAllPatients(query)
paragraphs = ""
for patient in patients:
paragraphs += getEmptyParagraph(patient['idPatient'],patient[
    typeAtrib])
return paragraphs

def getAgesPatientValues():
query = {}
atrib = 'age'
paragraphs = getAllOneAtrib(query,atrib)
return paragraphs

def getWeigthPatientValues():
query = {}
atrib = 'weight'
paragraphs = getAllOneAtrib(query,atrib)
return paragraphs

def getHeightPatientValues():
query = {}
atrib = 'height'
paragraphs = getAllOneAtrib(query,atrib)
return paragraphs

def handlerFiles(request):
for afile in request.FILES.getlist('patientFiles'):
myfile = afile
fs = FileSystemStorage()
filename = settings.MEDIA_ROOT + "/" + myfile.name
f = open(filename,'w')
f.write(myfile.read())
f.close()

################################## MDB Methods
    ###################################
def checkInPatient(idPatient):
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
return checkPatientMDB(dbh,idPatient)

def getIdPatient():
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
nextId = "id" + str(getNextIdPatientMDB(dbh))
return nextId
```

```
def addMedicinesPatient(dbh,idPatient,pForm):
medicines = getValues(pForm,"medicine")
for medicine in medicines:
mForm = {"medicine": medicine}
addMedicineMDB(dbh,idPatient,mForm)
print colored("proceso para anadir las medicinas que toma un
    paciente", "green")

def addDiseasesPatient(dbh,idPatient,pForm):
diseases = getValues(pForm,"disease")
for disease in diseases:
dForm = {"disease":disease}
addDiseaseMDB(dbh,idPatient,dForm)
print colored("proceso para anadir las enfermedades que tiene un
    paciente", "green")

def addPatient(pForm):
idPatient = getIdPatient()
#print (idPatient)
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
addPatientMDB(dbh,idPatient,pForm)
addMedicinesPatient(dbh,idPatient,pForm)
addDiseasesPatient(dbh,idPatient,pForm)
print colored("Added " + str(idPatient), "green")
return idPatient

def getPatient(idPatient):
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
p = getPatientMDB(dbh,idPatient)
print colored("patient search " + str(idPatient) + ",result: %s"
    %p, "green")
return(p)

def getAllPatients(query):
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
patients = getAllPatientWithMDB(dbh,query)
return patients

def getAllMedicines():
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
medicines = getAllMedicinesMDB(dbh)
return getForm(medicines,"medicine")

def getPatientMedicines(idPatient):
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
medicines = getPatientMedicinesMDB(dbh,idPatient)
return getItemize(medicines)
```

```
def getMedicinesValues():
dbh = openConectionMDB(settings.HOST_MONGO, settings.PORT_MONGO)
medicines = getAllMedicinesMDB(dbh)
paragraphs = ""
for medicine in medicines:
c = getNumberOfPatientsMedicineMDB(dbh, medicine['medicine'])
paragraphs += getEmptyParagraph(medicine['medicine'],c)
return paragraphs

def getAllDiseases():
dbh = openConectionMDB(settings.HOST_MONGO, settings.PORT_MONGO)
diseases = getAllDiseasesMDB(dbh)
return getForm(diseases,"disease")

def getPatientDiseases(idPatient):
dbh = openConectionMDB(settings.HOST_MONGO, settings.PORT_MONGO)
diseases = getPatientDiseasesMDB(dbh,idPatient)
return getItemize(diseases)

def getDiseasesValues():
dbh = openConectionMDB(settings.HOST_MONGO, settings.PORT_MONGO)
diseases = getAllDiseasesMDB(dbh)
paragraphs = ""
for disease in diseases:
c = getNumberOfPatientsDiseaseMDB(dbh, disease['disease'])
paragraphs += getEmptyParagraph(disease['disease'],c)
return paragraphs

def getSexAgeRelationValues():
dbh = openConectionMDB(settings.HOST_MONGO, settings.PORT_MONGO)
maleQuery = {"sex":'Varon'}
femaleQuery = {"sex":'Hembra'}
maleAges = selectValue(getAllPatientWithMDB(dbh,maleQuery),'age'
    )
print(maleAges)
femaleAges = selectValue(getAllPatientWithMDB(dbh,femaleQuery),'
    age')
print(femaleAges)
paragraphs = ""
paragraphs += getEmptyParagraph('maleAgeVal',maleAges)
paragraphs += getEmptyParagraph('femaleAgeVal',femaleAges)
return paragraphs

def deletePatient(idPatient):
dbh = openConectionMDB(settings.HOST_MONGO, settings.PORT_MONGO)
deletePatientMDB(dbh,idPatient)

def updatePatient(idPatient,pForm):
dbh = openConectionMDB(settings.HOST_MONGO, settings.PORT_MONGO)
```

```
updatePatientMDB(dbh,idPatient,"cosas")

def restartUsers():
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
deleteAllPatientsMDB(dbh)
deleteAllMedicinesMDB(dbh)
deleteAllDiseasesMDB(dbh)
loadindAllDiseasesMDB(dbh,settings.DISEASES_TXT)
loadindAllMedicinesMDB(dbh,settings.MEDICINES_TXT)

def addFileToPatient(idPatient,fileName):
print colored("----------------- eadimos en la ddbb
    ----------------------" + fileName,'blue')
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
idFile = insertFileMDB(idPatient, fileName,dbh)
addFileToPatientMDB(dbh,idPatient,fileName,idFile)
print colored("----------------- eadido en la ddbb:" + str(
    idPatient) + ":" + str(idFile),'blue')

def getDictFilesIds(idPatient):
dbh = openConectionMDB(settings.HOST_MONGO,settings.PORT_MONGO)
filesId = getDictFilesIdsMDB(dbh,idPatient)
if not filesId:
return None
files = []
for f in filesId:
files.append(getFileMDB(dbh,f))
return [files,filesId]

################################## Process request
    ##################################
def index(request):
r = renderIndex()
return HttpResponse(r)

def manejoficheros(request,idPatient):
try:
print colored("----------------- empezamos con el manejo de
    ficheros----------------------",'blue')
print colored(request.FILES,'blue')
for afile in request.FILES.getlist('patientFiles'):
myfile = afile
print colored("*-1-*" + str(myfile),'blue')
fs = FileSystemStorage()
print colored("*-2-*" + str(fs),'blue')
filename = settings.MEDIA_ROOT + "/" + myfile.name
f = open(filename,'w')
print colored(f,'blue')
f.write(myfile.read())
```

```
f.close()
filename = fs.save(myfile.name, myfile)
"""print colored("*-3-*" + str(filename),'blue')
uploaded_file_url = fs.url(filename)
print colored("*-4-*" + str(uploaded_file_url),'blue')
filename = settings.MEDIA_ROOT + "/" + myfile.name"""

for afile in request.FILES.getlist('patientFiles'):
filename = settings.MEDIA_ROOT + "/" + afile.name
addFileToPatient(idPatient,filename)

print colored("
    ----------------------------------------------------------FIN
    ----------------------------------","blue')
except:
print colored("without files","blue')

@csrf_exempt
def formSingupPatient(request,formu):
print colored(request,'blue')
print colored(formu,'blue')
pForm = request.POST
pForm = checkSingForm(pForm)
idPatient = addPatient(pForm)
redirect = "/formSearchPatient?idPatient=" + str(idPatient)
manejoficheros(request,idPatient)
return HttpResponseRedirect(redirect)

def formSearchPatient(request,formu):
form = str(request).split(" ")[2].split("?")[1].split("'")[0]
pForm = parsedForm(form)
print colored(str(pForm),'red')
isPatient = checkInPatient(pForm)
print colored(isPatient,"green")
if not(isPatient):
return HttpResponseRedirect("/")
patient = getPatient(pForm)
print colored(patient,'red')
r = renderUserPage(patient)
return HttpResponse(r)

def formCalculateRisk(request,formu):
form = str(request).split(" ")[2].split("?")[1].split("'")[0]
idP = form.split("&")[0];
modelMLName = form.split("&")[1].split("=")[1]
pForm = parsedForm(idP)
print colored(form,"green")
print colored(idP,"green")
print colored(modelMLName,"green")
```

```
print colored(pForm,"green")
isPatient = checkInPatient(pForm)
print colored(isPatient,"green")
if not(isPatient):
return HttpResponseRedirect("/")
patient = getPatient(pForm)
r = renderRiskUserPage(patient,modelMLName)
return HttpResponse(r)

def formDeleteUser(request,formu):
form = str(request).split("␣")[2].split("?")[1].split("'")[0]
pForm = parsedForm(form)
isPatient = checkInPatient(pForm)
print colored(isPatient,"green")
if not(isPatient):
return HttpResponseRedirect("/")
deletePatient(pForm)
print colored("user␣delete:" + str(pForm), "green")
return HttpResponseRedirect("/")

def formRestartUsers(request,formu):
restartUsers()
print colored("restart␣users", "green")
return HttpResponseRedirect("/")

@csrf_exempt
def formUpdateTest(request,idPatient):
print colored(request,'blue')
print colored(request.FILES,'blue')
print colored(idPatient,'blue')
manejoficheros(request,idPatient)
return HttpResponseRedirect('/formSearchPatient?idPatient=' +
    idPatient)



def downloadCsv(request,user):
print colored(user,'red')
response = HttpResponse(content_type='text/csv')
response['Content-Disposition'] = 'attachment;␣filename=' + user
writer = csv.writer(response)
writer.writerow(['id','age','sex','type','weight', 'height','
    athletic','smoker','foot','template','leg'])
if user.split(".")[0] == "users":
query = {}
patients = getAllPatients(query)
patientsToCsv(patients,writer)
else:
query = {'idPatient':user.split(".")[0]}
```

```
patient = getPatient(query)
writer.writerow([patient['idPatient'], patient['age'], patient['
    sex'], patient['type'], patient['weight'], patient['height'],
     patient['athletic'], patient['smoker'], patient['flat-
    concave foot'], patient['template'], patient['leg']])
return response

def downloadMedicalTest(request,user):

#response['Content-Disposition'] = 'attachment; filename=' +
    user
tupla = getDictFilesIds(user)
if not tupla:
resp = HttpResponse(content_type='text/json')
resp['Content-Disposition'] = 'attachment; filename=%s' % user
return resp

files = tupla[1]
filenames = files.keys()

# Folder name in ZIP archive which contains the above files
# E.g [thearchive.zip]/somefiles/file2.txt
# FIXME: Set this to something better
zip_subdir = str(user)
zip_filename = "%s.zip" % zip_subdir

# Open StringIO to grab in-memory ZIP contents
s = StringIO.StringIO()

# The zip compressor
zf = zipfile.ZipFile(s, "w")

for fpath in filenames:
fpath = fpath.replace('\u002E','.')
# Calculate path for file in zip
fdir, fname = os.path.split(fpath)
zip_path = os.path.join(zip_subdir, fname)

# Add file, at correct path
zf.write(fpath, zip_path)

# Must close zip for all contents to be written
zf.close()

# Grab ZIP file from in-memory, make response with correct MIME-
    type
resp = HttpResponse(s.getvalue(), content_type = "application/x-
    zip-compressed")
# ..and correct content-disposition
```

```
resp['Content-Disposition'] = 'attachment; filename=%s' %
    zip_filename

return resp

#path = settings.MEDIA_ROOT + "/" + afile.name
#response = HttpResponse(content_type='application/force-
    download') # mimetype is replaced by content_type for django
    1.7
#for f in filesKeys:
#        f = f.replace('\u002E','.')
#        response['Content-Disposition'] = 'attachment; filename
    =%s' %smart_str(f)
#        response['X-Sendfile'] = str(f)
#        # It's usually a good idea to set the 'Content-Length'
    header too.
#        # You can also set any other required headers: Cache-
    Control, etc.
#        print colored(f,"red")
#        return response

############################## Render Pages
    ######################################
def renderRiskUserPage(userDoc,modelMLName):
risk = caculateRisk(userDoc,modelMLName)
template = get_template('userRisk.html')
c = Context({'idPatient':userDoc['idPatient'],'risk':risk,'
    modelName':modelMLName})
render = template.render(c)
return render

def renderUserPage(userDoc):
template = get_template('user.html')
medicines = getPatientMedicines(userDoc['idPatient'])
diseases = getPatientDiseases(userDoc['idPatient'])
c = Context({'idPatient':userDoc['idPatient'],'age':userDoc['age
    '],'sex':userDoc['sex'], 'weight':userDoc['weight'],'height':
    userDoc['height'],'type':userDoc['type'], 'athletic':userDoc[
    'athletic'], 'smoker':userDoc['smoker'],'foot':userDoc['flat-
    concave foot'],'template':userDoc['template'],'diseases':
    diseases, 'medicines':medicines})
render = template.render(c)
return render

def renderIndex():
template = get_template('index.html')
diseasesForm = getAllDiseases()
diseasesVal = getDiseasesValues()
medicinesForm = getAllMedicines()
```

```
medicinesVal = getMedicinesValues()
typeVal = getTypesPatientsValues()
ageVal = getAgesPatientValues()
weightVal = getWeigthPatientValues()
heightVal = getHeightPatientValues()
sexAgeRelationVal = getSexAgeRelationValues()
modelsMLVal = getModelsName()
c = Context({"diseases":diseasesForm,"medicines":medicinesForm,"
    diseasesVal":diseasesVal,"medicinesVal":medicinesVal,"typeVal
    ":typeVal,'ageVal':ageVal,'weightVal':weightVal, 'heightVal':
    heightVal,'sexVal':sexAgeRelationVal, 'modelsML':modelsMLVal
    })
render = template.render(c)
return render
```

triggerXlsxMongo

Este archivo es el encargado de la obtención de todos los datos de los pacientes a través del formulario de excel proporcionado, para la automatización de carga de usuarios en la base de datos.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import xlrd
import datetime
import sys
import os

sys.path.append("/home/jorge/tfg/connect_python_to_mongo/")
from api_MDB_server_django import *

def getDiseases(wsheet):
diseases = wsheet.cell(7, 2).value
diseases = diseases.split(",")
lastDiseases = diseases[-1].split('y')
if (len(lastDiseases) > 1) :
diseases[-1] = lastDiseases[0]
diseases.append(lastDiseases[-1])
return diseases

def getMedicines(wsheet):
medicines = wsheet.cell(8, 2).value
medicines = medicines.split(",")
lastMedicines = medicines[-1].split('y')
if (len(lastMedicines) > 1) :
medicines[-1] = lastMedicines[0]
medicines.append(lastMedicines[-1])
return medicines

def getTypePatient(wsheet):
control = wsheet.cell(3, 3).value
asintomatico = wsheet.cell(4, 3).value
claudicante = wsheet.cell(5, 3).value
if(control):
return "Control"
if(asintomatico):
return "Asintomatico"
if(claudicante):
return "Claudicante"

def getAge(birthdate):
now = int(datetime.datetime.now().strftime("%Y-%m-%d␣%H:%M:%S").
    split("-")[0])
year = int(birthdate[0])
```

```
return (now - year)

def getPatientDic(codePatient,sex,age,hight,weight,typePatient,
    smoker,athletic,foot,template,leg):
patientDic = {}
patientDic['idPatient'] = codePatient
patientDic['age'] = age
patientDic['sex'] = sex
patientDic['height'] = hight
patientDic['weight'] = weight
patientDic['typePatient'] = typePatient
patientDic['smoker'] = smoker
patientDic['athletic'] = athletic
patientDic['flat-concave foot'] = foot
patientDic['template'] = template
patientDic['leg'] = leg
return patientDic

def getPatientToXlsx(nameFile):
wb = xlrd.open_workbook(nameFile)
wsheet = wb.sheet_by_index(0)
codigo_paciente = wsheet.cell(1, 2).value
nombre = wsheet.cell(1, 4).value
apellido = wsheet.cell(1, 6).value
fecha_de_nacimiento = xlrd.xldate_as_tuple(wsheet.cell(2, 2).
    value,0)
age = getAge(fecha_de_nacimiento)
altura = wsheet.cell(2, 4).value
peso = wsheet.cell(2, 6).value
tipo_de_paciente = getTypePatient(wsheet)
fumador = wsheet.cell(6, 4).value.encode("utf-8").replace('\xc3\
    xb1','nn').replace('\xc3\xb3','o').replace('\xc3\xad','i').
    replace('\xc3\xa9','e').replace('\xc3\xba','u').replace('\xc3
    \xa1','a')
atletico = wsheet.cell(9, 2).value.encode("utf-8").replace('\xc3
    \xb1','nn').replace('\xc3\xb3','o').replace('\xc3\xad','i').
    replace('\xc3\xa9','e').replace('\xc3\xba','u').replace('\xc3
    \xa1','a')
pies_planoscavos = wsheet.cell(10, 2).value.encode("utf-8").
    replace('\xc3\xb1','nn').replace('\xc3\xb3','o').replace('\
    xc3\xad','i').replace('\xc3\xa9','e').replace('\xc3\xba','u')
    .replace('\xc3\xa1','a')
plantillas = wsheet.cell(10, 4).value.encode("utf-8").replace('\
    xc3\xb1','nn').replace('\xc3\xb3','o').replace('\xc3\xad','i'
    ).replace('\xc3\xa9','e').replace('\xc3\xba','u').replace('\
    xc3\xa1','a')
pierna_dominante = wsheet.cell(10, 7).value.encode("utf-8").
    replace('\xc3\xb1','nn').replace('\xc3\xb3','o').replace('\
    xc3\xad','i').replace('\xc3\xa9','e').replace('\xc3\xba','u')
```

```
    .replace('\xc3\xa1','a')
if(pierna_dominante == "D"):
pierna_dominante = "Derecha"
elif(pierna_dominante == "I"):
pierna_dominante = "Izquierda"
enfermedades = getDiseases(wsheet)
medicinas = getMedicines(wsheet)

dbh = openConectionMDB('localhost',27017)
idPatient = 'id'+str(getNextIdPatientMDB(dbh))
dictDoc = getPatientDic(idPatient,'Varon',age,altura,peso,
    tipo_de_paciente,fumador,atletico,pies_planoscavos,plantillas
    ,pierna_dominante)
addPatientMDB(dbh,idPatient,dictDoc)

dictEnf = {}
for e in enfermedades:
dictEnf['disease'] = e.lower().strip().encode("utf-8").replace('
    \xc3\xb1','nn').replace('\xc3\xb3','o').replace('\xc3\xad','i
    ').replace('\xc3\xa9','e').replace('\xc3\xba','u').replace('\
    xc3\xa1','a')
addDiseaseMDB(dbh,idPatient,dictEnf)

dictMed = {}
for m in medicinas:
dictMed['medicine'] = m.lower().strip().encode("utf-8").replace(
    '\xc3\xb1','nn').replace('\xc3\xb3','o').replace('\xc3\xad','
    i').replace('\xc3\xa9','e').replace('\xc3\xba','u').replace('
    \xc3\xa1','a')
addMedicineMDB(dbh,idPatient,dictMed)


def addAllXlsInformation():
directory = os.walk('.')

for dirname, dirnames, filenames in os.walk('./xlsData/'):
for filename in filenames:

f = os.path.join(dirname, filename)
print(f)
getPatientToXlsx(f)


def main():
addAllXlsInformation()


if __name__ == "__main__":
main()
```

api_MDB_server_django.py

Este archivo es el encargado de conectar las peticiones del servidor con la base de datos.

```
import sys
import gridfs
from termcolor import colored

from pymongo import Connection
from pymongo.errors import ConnectionFailure

def __getPatientFilesDoc(idPatient,dictIdDoc):
patient_file_doc = {
"idPatient": idPatient,
"files": dictIdDoc
}
return patient_file_doc

def __getPatientDoc__(idPatient,dictDoc):
patient_doc = {
"idPatient": idPatient,
"age": dictDoc['age'],
"sex": dictDoc['sex'],
"type": dictDoc['typePatient'],
"height": dictDoc['height'],
"weight": dictDoc['weight'],
"athletic": dictDoc['athletic'],
"smoker": dictDoc['smoker'],
"flat-concave foot": dictDoc['flat-concave foot'],
"template": dictDoc['template'],
"leg": dictDoc['leg']
}
return patient_doc

def __getMedicineDoc__(idPatient,dictDoc):
medicine_doc = {
"medicine" : dictDoc['medicine'],
"idPatient": idPatient
}
return medicine_doc

def __getDiseaseDoc__(idPatient,dictDoc):
disease_doc = {
"disease" : dictDoc['disease'],
"idPatient": idPatient
}
return disease_doc
```

```
############################## Public Methods API
    ######################################

def openConectionMDB(h,p):
""" Connect to MongoDb """
try:
c= Connection(host=h, port=p) # be careful if you change the
    port, it doesn't work
print colored("Connection successfully","yellow")
except ConnectionFailure,e:
sys.stderr.write("Could not connect to MongoDb: %s" % e)
sys.exit(1)

""" to can talk to the database you need know the name and
    password"""
dbh = c["mydb"] #this create a hadle to
assert dbh.connection == c
print colored("Successfully set up a database handle","yellow")
return dbh


########################################## Files Table
    ########################################################
def addFileToPatientMDB(dbh,idPatient,filename, idFile):
print colored(idPatient,"red")
print colored(idFile,"red")
patients_filesCursor = dbh.patientsFilesLinked.find_one({"
    idPatient":idPatient})
print colored(patients_filesCursor,"red")
filename = filename.replace('.','\u002E')
if not patients_filesCursor:
dictIdDoc = {}
dictIdDoc[filename] = idFile
new_patient_files = __getPatientFilesDoc(idPatient,dictIdDoc)
dbh.patientsFilesLinked.insert(new_patient_files,safe=True)
print colored("add a files test patient: %s" %new_patient_files
    ,"yellow")
else:
print colored(patients_filesCursor,"red")
new_files = patients_filesCursor.get("files")
print colored(new_files,'red')
new_files[filename] = idFile
new_patient_files = __getPatientFilesDoc(idPatient,new_files)
dbh.patientsFilesLinked.update({"idPatient":idPatient},
    new_patient_files)
print colored("update a files test patient:" + str(idPatient) +
    ":" + str(new_files),"yellow")

def getFileMDB(dbh,file_id):
```

```
file_saved = dbh.patientsFiles.chunks.find_one({'files_id':
    file_id})
print colored(file_saved,'red')
return file_saved

def getDictFilesIdsMDB(dbh,idPatient):
patients_filesCursor = dbh.patientsFilesLinked.find_one({"
    idPatient":idPatient})
if not patients_filesCursor:
return None
dictIdObject = patients_filesCursor.get("files")
return dictIdObject

def insertFileMDB(idPatient, filename,dbh):
print "----------insert----------"
print "Patient --> " + idPatient
print "File    --> " + filename

patient = dbh.users.find_one({'idPatient': idPatient})
print colored(patient,'green')

grid = gridfs.GridFS(dbh,"patientsFiles")
print colored(grid,'blue')
with open(filename, "r") as fin:
sdf_id = grid.put(fin)
print colored(sdf_id,'yellow')
return sdf_id

########################################## Medicines Table
    ################################################
def addMedicineMDB(dbh,idPatient,dictDoc):
medicine_doc = __getMedicineDoc__(idPatient,dictDoc)
dbh.medicines.insert(medicine_doc,safe=True)
print colored("add a medicine: %s" %medicine_doc,"yellow")

def getAllMedicinesMDB(dbh):
medicines = dbh.medicines.find({"idPatient":"root"})
#print colored("medicines docs founded: %s" %medicines,"yellow
    ")
rMedicines = []
for record in medicines:
rMedicines.append(record)
return rMedicines

def getPatientMedicinesMDB(dbh,idPatient):
medicinesCursor = dbh.medicines.find({'idPatient':idPatient})
if not medicinesCursor:
return ""
medicines = []
```

```
for medicine in medicinesCursor:
medicines.append(medicine['medicine'])
return medicines

def getNumberOfPatientsMedicineMDB(dbh,medicine):
medicinesCursor = dbh.medicines.find({'medicine':medicine})
c = medicinesCursor.count() - 1
return c

def deleteAllMedicinesMDB(dbh):
dbh.medicines.remove()
print colored("all medicines remove","yellow")

def loadindAllMedicinesMDB(dbh,medicinesDoc):
f = open(medicinesDoc,"r")
for line in f:
line = line.split("\n")[0].lower()
print line
dictDoc = {"medicine": line}
addMedicineMDB(dbh,"root",dictDoc)
print colored("loaded medicines table","yellow")

########################################## Diseases Table
    ################################################
def addDiseaseMDB(dbh,idPatient,dictDoc):
disease_doc = __getDiseaseDoc__(idPatient,dictDoc)
dbh.diseases.insert(disease_doc,safe=True)
print colored("add a disease: %s" %disease_doc,"yellow")

def getAllDiseasesMDB(dbh):
diseases = dbh.diseases.find({"idPatient":"root"})
#print colored("diseases docs founded: %s" %diseases,"yellow")
rDiseases = []
for record in diseases:
rDiseases.append(record)
return rDiseases

def getPatientDiseasesMDB(dbh,idPatient):
diseasesCursor = dbh.diseases.find({'idPatient':idPatient})
if not diseasesCursor:
return ""
diseases = []
for disease in diseasesCursor:
diseases.append(disease['disease'])
return diseases

def deleteAllDiseasesMDB(dbh):
dbh.diseases.remove()
print colored("all patients remove","yellow")
```

```
def getNumberOfPatientsDiseaseMDB(dbh,disease):
diseasesCursor = dbh.diseases.find({'disease':disease})
c = diseasesCursor.count() - 1
return c

def loadindAllDiseasesMDB(dbh,diseasesDoc):
f = open(diseasesDoc,"r")
for line in f:
print line
line = line.split("\n")[0].lower()
print line
dictDoc = {"disease": line}
addDiseaseMDB(dbh,"root",dictDoc)
print colored("loaded diseases table","yellow")

########################################## Patients Table
   ##################################################
def checkPatientMDB(dbh,idP):
user_doc = dbh.users.find_one(idP)
if not user_doc:
print colored("no document found for idPatient: " + str(idP),"
    yellow")
return (False)
print colored("document found for idPatient: " + str(idP),"
    yellow")
return (True)

def addPatientMDB(dbh,idPatient,dictDoc):
patient_doc = __getPatientDoc__(idPatient,dictDoc)
dbh.users.insert(patient_doc,safe=True)
print colored("add a patient: %s" %patient_doc,"yellow")

def getPatientMDB(dbh,idP):
user_doc = dbh.users.find_one(idP)
return(user_doc)

def deletePatientMDB(dbh,idPatient):
dbh.users.remove(idPatient,safe=True)
print colored("delete a patient:" + str(idPatient),"yellow")

def updatePatientMDB(dbh,idPatient,updates):
print colored("updates","yellow")

def deleteAllPatientsMDB(dbh):
dbh.users.remove()
print colored("all patients remove","yellow")

def getNextIdPatientMDB(dbh):
```

```
next = dbh.users.count() + 1
return str(next)

def getAllPatientWithMDB(dbh,query):
queryCursor = dbh.users.find(query)
if not queryCursor:
return ""
results = []
for result in queryCursor:
results.append(result)
return results
```