UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

título del TFG
Documentación Técnica

Presentado por Jorge Martínez Martín
en Universidad de Burgos — 4 de julio de 2024
Tutor: Álvar Arnaiz González

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

El propósito de este documento es establecer una guía de forma clara y organizada para la ejecución de un proyecto. En él se definen los objetivos y la forma en la que se van a alcanzar.

Scrum

Para la planificación del proyecto ha sido empleado el marco de gestión de proyectos de metodología ágil [3]. Scrum es una metodología ágil para la gestión de proyectos complejos en entornos cambiantes. Se basa en tres pilares: eventos, roles y artefactos, y se trabaja en sprints de una duración determinada, generalmente entre una semana y un mes. Los equipos de Scrum se autoorganizan y se comprometen a entregar resultados de alta calidad de manera eficiente y creativa.

Roles

En este marco de gestión de proyectos, se destacan tres roles principales:

- *Scrum Master*: Líder del equipo encargado de eliminar obstáculos y facilitar la auto-organización y coordinación del equipo.
- *Product Owner*: Representa la voz del cliente, lidera el desarrollo del producto y busca el valor para los usuarios.

- Equipo de desarrollo: Responsable de ejecutar las acciones previstas para el éxito del proceso

Eventos

Los eventos clave en Scrum son:

- *Sprint*: Reunión para planificar el trabajo del próximo sprint.
- Scrum diario: Breve reunión diaria para sincronizar actividades.
- Revisión del *sprint*: Revisión del producto al final del *sprint*.
- Retrospectiva del *sprint* : Análisis de los éxitos y fallos del *sprint*

Artefactos

Scrum utiliza tres artefactos principales:

- Pila de producto o *product backlog*: Inventario que contiene el trabajo pendiente en el producto.
- Pila del *sprint* o *sprint backlog*: Lista de tareas a realizar durante el *sprint*.
- Incremento: Versión mejorada y funcional del producto al final de cada *sprint*.

A.2. Planificación temporal

Planificación mediante *sprints*

Se ha decido que la planificación del proyecto se haga mediante *sprints* debido al tamaño del equipo (un desarrollador).

Sprint 1

- **Objetivos**
  1. Configuración inicial: creación del repositorio y configuración de cuenta de ZenHub.
  2. Memoria: redacción de la introducción y familarización con las aplicaciones para edición de archivos TEX.

3. Lectura de papers: «Supervised classification of bradykinesia in Parkinson's disease from smartphone videos» [5], «The discerning eye of computer vision: Can it measure Parkinson's finger tap bradykinesia?» [4] y «A computer vision framework for finger-tapping evaluation in Parkinson's disease» [2].

- **Periodo** Este *sprint* se desarrolló entre el 2 de octubre del 2023 y el 16 de octubre del 2023.
- ***Review*** En la revisión se resolvieron dudas sobre la documentación y se decidió cambiar de herramienta para la realización de la metodología scrum del proyecto a zube.

Sprint 2

- **Objetivos**

1. Documentación de trabajos previos: lectura del trabajo realizado por Catalin para ver la extracción de datos.
2. Memoria: redacción del apartado de trabajos relacionados.
3. Curso de flask: realización de los tutoriales del framework de flask para la realización del apartado web.

- **Periodo** Este *sprint* se desarrolló entre el 16 de octubre del 2023 y el 29 de octubre del 2023.
- ***Review*** Se resolvieron dudas sobre dónde encontrar la información dentro del proyecto de Catalin así como problemas a la hora de compilar los documentos de LaTeX.

Sprint 3

- **Objetivos**

1. Documentación de trabajos previos: lectura de los documentos para la extracción de datos usando el paquete de python TSFresh.
2. Memoria: corrección de errores en la documentación y añadida biografía.
3. Curso de flask: continuada la realización de los tutoriales del framework de flask para la realización del apartado web.

4. Repositorio: añadido el archivo gitignore para evitar la subida de archivos temporales.

- **Periodo** Este *sprint* se desarrolló entre el 30 de octubre del 2023 y el 12 de noviembre del 2023.
- ***Review*** Problemas a la hora de la realización de los objetivos propuestos agregados en el siguiente *sprint*.

Sprint 4

- **Objetivos**
  1. Creación del *mockup* de la aplicación.
  2. Documentación de los anexos.
- **Periodo** Este *sprint* se desarrolló entre el 13 de noviembre del 2023 y el 30 de diciembre del 2023.
- ***Review*** Se revisó el *mockup* y se propuso una modificación del mismo.

Sprint 5

- **Objetivos**
  1. Finalización del curso de flask.
  2. Modificación del *mockup*.
  3. Creación de diagramas entidad relación y diagramas de casos de uso.
- **Periodo** Este *sprint* se desarrolló entre el 30 de noviembre del 2023 y el 12 de diciembre del 2023.
- ***Review*** Se revisaron tanto los diagramas como el *mockup* y se propusieron unos cambios en los diagramas.

Sprint 6

- **Objetivos**
  1. Documentación general añadiendo técnicas y herramientas.
  2. Búsqueda de trabajos parecidos o aplicaciones médicas similares.

3. Modificación de los diagramas de casos de uso y entidad-relación.

- **Periodo** Este *sprint* se desarrolló entre el 13 de diciembre del 2023 y el 19 de enero de 2024.
- ***Review*** Se cumplieron todos los objetivos del sprint.

Sprint 7

- **Objetivos**

  1. Documentación general añadiendo técnicas y herramientas.
  2. Búsqueda de trabajos parecidos o aplicaciones médicas similares.
  3. Modificación del diagramas de casos de uso y entidad-relación.

- **Periodo** Este *sprint* se desarrolló entre el 20 de enero de 2024 y el 15 de febrero de 2024.
- ***Review*** Se cumplieron todos los objetivos del sprint. Se detectaron ciertos fallos en la documentación que se añadieron como tarea para el siguiente sprint.

Sprint 8

- **Objetivos**

  1. Comienzo de la página web.
  2. Conceptos teóricos de la aplicación.
  3. Solución de errores detectados en la anterior revisión.
  4. Conexión con la base de datos.

- **Periodo** Este *sprint* se desarrolló entre el 16 de febrero del 2024 y el 29 de febrero de 2024.
- ***Review*** Se cumplieron todos los objetivos del sprint.

Sprint 9

- **Objetivos**

  1. Continuación de la redacción de los conceptos teóricos.

2. Creación de la página de registro.

- **Periodo** Este *sprint* se desarrolló entre el 1 de marzo del 2024 y el 14 de marzo de 2024.
- ***Review*** No se cumplió el objetivo de la documentación que se añadió al siguiente sprint.

Sprint 10

- **Objetivos**
  1. Continuación de la redacción de los conceptos teóricos.
  2. Creación de la página de inicio de sesión.
- **Periodo** Este *sprint* se desarrolló entre el 15 de marzo del 2024 y el 8 de abril de 2024.
- ***Review*** Se cumplieron todos los objetivos del sprint.

Sprint 11

- **Objetivos**
  1. Documentación de técnicas y herramientas.
  2. Creación de la página principal de administrador.
  3. Solución de errores relacionados con la página de registro.
  4. Creación de la página de modificación de usuarios.
- **Periodo** Este *sprint* se desarrolló entre el 9 de abril de 2024 y el 25 de abril de 2024.
- ***Review*** Se cumplieron todos los objetivos del sprint.

Sprint 12

- **Objetivos**
  1. Creación de la página principal del paciente.
  2. Añadir la funcionalidad de actualizar el usuario en la base de datos una vez modificado.

3. Iniciar entrenamiento de los modelos para la predicción de características.

- **Periodo** Este *sprint* se desarrolló entre el 26 de abril de 2024 y el 8 de mayo de 2024.
- ***Review*** No se pudieron realizar los experimentos debido a problemas de compatibilidad de versión.

Sprint 13

- **Objetivos**
  1. Creación de la página principal del doctor.
  2. Continuar con el entrenamiento de los modelos.
- **Periodo** Este *sprint* se desarrolló entre el 9 de mayo de 2024 y el 15 de mayo de 2024.
- ***Review*** Debido a problemas durante el entrenamiento de los modelos, no se pudo completar este objetivo correctamente.

Sprint 14

- **Objetivos**
  1. Continuar con el entrenamiento de los modelos.
- **Periodo** Este *sprint* se desarrolló entre el 16 de mayo de 2024 y el 22 de mayo de 2024.
- ***Review*** Se cumplieron todos los objetivos del sprint y se consiguió resolver el problema con el entrenamiento de modelos.

Sprint 15

- **Objetivos**
  1. Agregar funcionalidad de gestionar medicinas de los pacientes.
  2. Agregar funcionalidad de ver medicinas del paciente.
  3. Modificar la base de datos para añadir medicinas a la aplicación.
  4. Agregar funcionalidad de gestionar vídeos de los pacientes.

5. Agregar funcionalidad de ver vídeos del paciente.

- **Periodo** Este *sprint* se desarrolló entre el 23 de mayo de 2024 y el 31 de mayo de 2024.
- ***Review*** Se cumplieron todos los objetivos del sprint.

Sprint 16

- **Objetivos**

1. Refactorizar la aplicación para aplicar un Modelo-vista-controlador.
2. Solucionar fallos de funcionalidades del doctor a la hora de gestionar vídeos y medicinas.
3. Implementar extracción de características de los vídeos.
4. Realizar comparación de modelos.

- **Periodo** Este *sprint* se desarrolló entre el 1 de junio de 2024 y el 13 de junio de 2024.
- ***Review*** Se cumplieron todos los objetivos del sprint.

A.3. Estudio de viabilidad

En este apartado se estudian los criterios a tener en cuenta a nivel económico y legal para determinar si es factible y si cumple los objetivos propuestos.

Viabilidad económica

En el siguiente subapartado se expone la viabilidad del proyecto a nivel económico.

Costes

Se entiende por coste todo aquel gasto necesario para poder llevar acabo un proyecto desde su inicio hasta su finalización. Para el análisis de los costes del proyecto y el calculo de la amortización se ha tenido en cuenta la duración total del proyecto (40 semanas o entorno a 9 meses).

1. **Costes de empleados**: en este caso se cuenta con dos empleados que han sido el desarrollador (el alumno) y el *product owner* (tutor académico). Se estima que el tiempo real invertido por parte del desarrollador es de entorno a las 550 horas a lo largo de 9 meses. Teniendo en cuenta que el sueldo promedio de un programador junior en España es de entorno a 23.700€[1] y la jornada laboral completa son aproximadamente 1820 horas, el salario medio bruto por hora es de 13€. Por tanto el salario del desarrollador estará entorno a 550 horas x 13 $\frac{€}{\text{hora}}$ = 7150€ o aproximadamente 795$\frac{€}{\text{mes}}$ A esta cantidad se le ha de añadir los impuestos que se han de pagar como empresa por el empleado[2]

   - **Contingencias Comunes**: correspondiente al 23,60 %.
   - **Desempleo**: correspondiente al 6,70 %.
   - **FOGASA**: fondo de garantía salarial. Correspondiente al 0,20 %.
   - **Formación profesional**: correspondiente al 0,60 %.
   - **Accidentes de Trabajo y Enfermedades Profesionales**: correspondiente al 1,50 %.

   Por tanto, tras realizar los cálculos añadiendo los impuestos correspondientes, la empresa debe pagar 1.179,52 $\frac{€}{mes}$ por el desarrollador.

$$\frac{795\frac{€}{mes}}{(1 - (0{,}236 + 0{,}067 + 0{,}002 + 0{,}006 + 0{,}015))} = 1.179{,}52\frac{€}{mes} \tag{A.1}$$

   Por otro lado el *product owner*, suponiendo un sueldo bruto anual de 39000€[3] , se calcula que el salario bruto hora es de 21,43€. Teniendo en cuenta que el tutor ha dedicado entorno a 3 horas por semana al proyecto, cobraría 257,16€ brutos al mes. Aplicando la formula al igual que para el desarrollador:

$$\frac{257,16\frac{€}{mes}}{(1 - (0{,}236 + 0{,}067 + 0{,}002 + 0{,}006 + 0{,}015))} = 381{,}54\frac{€}{mes} \tag{A.2}$$

   Por lo tanto, y teniendo en cuenta que la duración del proyecto ha sido de 9 meses, el total que ha de pagar la empresa es de 14.049,54€ por los empleados.

[1] Fuente: https://es.indeed.com/career/programador-junior/salaries

[2] Fuente: https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/10721/10957/9932/4315

[3] Fuente: https://www.glassdoor.es/Sueldos/product-owner-sueldo-SRCH_KO0,13.htm

$$381{,}54\frac{€}{\text{mes}} \times 9\,\text{meses} + 1179{,}52\frac{€}{\text{mes}} \times 9\,\text{meses} = 14049{,}54\,€$$

2. **Hardware**: para el hardware solo se ha hecho uso del equipo del alumno. Este equipo se trata de un PcCom Gold Élite con un Intel Core i5-11400F, 16 GB de RAM y NVIDIA Geforce RTX 3070 con un P.V.P [4] de 1199,18€. Se calcula que el ordenador será amortizado en 4 años por lo que se debe calcular el precio amortizado para la duración del proyecto. En total son 224,84€.

$$\frac{1199,18€ \times 9\text{meses}}{4\text{años} \times 12\text{meses}} = 224,85€$$

3. **Software**: la mayoría de programas y *sofware* utilizado durante el desarrollo son gratuitos a excepción de GitHub Copilot cuyo coste de la versión *entreprise* es de 36,35€ al mes. Teniendo en cuenta la duración del proyecto, esto nos da un coste total de 327,15€.

Costes totales

A continuación se muestra la tabla con los costes totales.

Concepto	**Coste(€)**
Empleados	14.049,54
Hardware	224,84
Software	327,15
Total	14.601,53

Tabla A.1: Tabla de costes totales

Ingresos

Dado que este proyecto pretende ser accesible, su uso va a ser completamente gratuito. Debido a esto una de las principales formas en las que se podría obtener rédito económico de esta aplicación sería mediante la implementación de anuncios. Dado que es proyecto pretende ser una herramienta para facilitar el trabajo a pacientes y médicos no se ha considerado oportuno y por tanto no se ha estudiado esta opción. Otra forma en la que se podría

[4]Fuente: https://www.pccomponentes.com/pccom-gold-elite-intel-core-i5-11400f-16gb-1tb-ssd-rtx-3070

sacar ingresos sería mediante donaciones o ayudas que permitiesen cubrir los gastos de mantenimiento y desarrollo del proyecto.

Viabilidad legal

A continuación se muestra una tabla recopilando las licencias de las dependencias utilizadas durante el desarrollo del proyecto.

Dependencia	**Versión**	**Licencia**
`Flask`	3.0.2	BSD License
`Flask-Login`	0.6.3	MIT License
`Flask-SQLAlchemy`	3.1.1	BSD License
`Flask-WTF`	1.2.1	BSD License
`Font-Awesome-Flask`	0.1.4	GNU General Public License v3 or later (GPLv3+)
`Jinja2`	3.1.3	BSD License
`MarkupSafe`	2.1.5	BSD License
`PyYAML`	6.0.1	MIT License
`Pygments`	2.18.0	BSD License
`SQLAlchemy`	2.0.27	MIT License
`WTForms`	3.1.2	BSD License
`Werkzeug`	3.0.1	BSD License
`absl-py`	2.1.0	Apache Software License
`annotated-types`	0.6.0	MIT License
`attrs`	23.2.0	MIT License
`blinker`	1.7.0	MIT License
`certifi`	2024.2.2	Mozilla Public License 2.0 (MPL 2.0)
`cffi`	1.16.0	MIT License
`charset-normalizer`	3.3.2	MIT License
`click`	8.1.7	BSD License
`cloudpickle`	3.0.0	BSD License
`colorama`	0.4.6	BSD License
`contourpy`	1.2.1	BSD License
`cycler`	0.12.1	BSD License
`dask`	2024.5.0	BSD License
`dask-expr`	1.1.0	BSD License
`distributed`	2024.5.0	BSD License
`flatbuffers`	24.3.25	Apache Software License
`fonttools`	4.51.0	MIT License

Continúa en la siguiente página

Tabla A.2 – *Continuación*

Dependencia	**Versión**	**Licencia**
fsspec	2024.3.1	BSD License
greenlet	3.0.3	MIT License
gunicorn	22.0.0	MIT License
idna	3.7	BSD License
importlib_metadata	7.1.0	Apache Software License
itsdangerous	2.1.2	BSD License
jax	0.4.27	Apache-2.0
jaxlib	0.4.27	Apache-2.0
joblib	1.4.2	BSD License
kiwisolver	1.4.5	BSD License
llvmlite	0.42.0	BSD
locket	1.0.0	BSD License
matplotlib	3.8.4	Python Software Foundation License
mediapipe	0.10.13	Apache Software License
ml-dtypes	0.4.0	Apache Software License
msgpack	1.0.8	Apache Software License
numba	0.59.1	BSD License
numpy	1.26.4	BSD License
opencv-contrib-python	4.9.0.80	Apache Software License
opencv-python	4.9.0.80	Apache Software License
opt-einsum	3.3.0	MIT
packaging	24.0	Apache Software License; BSD License
pandas	2.2.2	BSD License
partd	1.4.2	BSD
patsy	0.5.6	BSD License
pillow	10.3.0	Historical Permission Notice and Disclaimer (HPND)
pip-licenses	4.4.0	MIT License
protobuf	4.25.3	3-Clause BSD License
psutil	5.9.8	BSD License
pure-eval	0.2.2	MIT License
pyarrow	16.0.0	Apache Software License
pycparser	2.22	BSD License
pydantic	2.7.1	MIT License
pydantic-settings	2.2.1	MIT License
pydantic_core	2.18.2	MIT License
pyparsing	3.1.2	MIT License

Continúa en la siguiente página

Tabla A.2 – *Continuación*

Dependencia	**Versión**	**Licencia**
`python-dateutil`	2.9.0.post0	Apache Software License; BSD License
`python-dotenv`	1.0.1	BSD License
`pytz`	2024.1	MIT License
`pywin32`	306	Python Software Foundation License
`pyzmq`	26.0.3	BSD License
`requests`	2.31.0	Apache Software License
`scikit-learn`	1.4.2	BSD License
`scipy`	1.13.0	BSD License
`six`	1.16.0	MIT License
`sortedcontainers`	2.4.0	Apache Software License
`sounddevice`	0.4.6	MIT License
`statsmodels`	0.14.2	BSD License
`stumpy`	1.12.0	BSD License
`tblib`	3.0.0	BSD License
`threadpoolctl`	3.5.0	BSD License
`toolz`	0.12.1	BSD License
`tornado`	6.4	Apache Software License
`tqdm`	4.66.4	MIT License; Mozilla Public License 2.0 (MPL 2.0)
`traitlets`	5.14.3	BSD License
`tsfresh`	0.20.2	MIT
`typing_extensions`	4.9.0	Python Software Foundation License
`tzdata`	2024.1	Apache Software License
`urllib3`	2.2.1	MIT License
`zict`	3.0.0	BSD License
`zipp`	3.18.1	MIT License

Tabla A.2: Dependencias del proyecto y licencias bajo las cuales están registradas. Extraídas mediante la librería pip-licenses[5].

Todas las librerías, dependencias y herramientas utilizadas licencias encontradas en programas de código abierto. Además, el Trabajo de Fin de Grado PADDEL [1] que ha sido utilizado para el desarrollo de este proyecto tiene una licencia MIT la cual permite a cualquier persona que obtenga una copia del proyecto modificarlo, copiarlo y redistribuirlo libremente. Es

[5] https://pypi.org/project/pip-licenses/

por ello que se ha optado por utilizar la licencia MIT para este proyecto también.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Previo a la creación de un programa se debe especificar las características concretas que ha de tener. En este caso caso ha definido estas características utilizando UML como estándar.

B.2. Objetivos generales

El objetivo principal de esta aplicación es mostrar, tanto a los pacientes como a sus respectivos doctores, su evolución mediante el uso de inteligencia artificial.

Además se busca que el usuario doctor sea capaz de realizar comparaciones de múltiples pacientes de manera simultánea.

B.3. Catálogo de requisitos

A continuación se presentan los requisitos funcionales y no funcionales de la aplicación.

Requisitos funcionales

RF1: El sistema debe permitir diferenciar entre tres tipos de roles: administrador, paciente y doctor.

RF2: El doctor ha de ser capaz de agregar vídeos a sus pacientes.

RF3: El doctor ha de ser capaz de eliminar vídeos a sus pacientes.

RF4: El doctor ha de ser capaz de visualizar el listado de vídeos de sus pacientes.

RF5: El doctor ha de ser capaces de clasificar los vídeos de sus pacientes.

RF6: Los usuarios han de ser capaces de iniciar sesión con sus correspondientes credenciales.

RF7: Los usuarios deben ser capaces de poder cerrar sesión.

RF8: El administrador ha de ser capaz de añadir usuarios.

RF9: El administrador ha de ser capaz de eliminar usuarios.

RF10: El administrador ha de ser capaz de modificar usuarios.

RF11: El administrador ha de ser capaz de visualizar a los usuarios.

RF12: El administrador ha de ser capaz de asignar doctor a los pacientes.

RF13: El administrador ha de ser capaz de desasignar doctor a los pacientes.

RF14: El paciente y el doctor han de ser capaces de ver la evolución del paciente.

RF15: El doctor ha de ser capaz de añadir medicinas a sus pacientes.

RF16: El doctor ha de ser capaz de eliminar medicinas a sus pacientes.

RF17: El doctor ha de ser capaz de visualizar el listado de medicinas de sus pacientes.

Requisitos no funcionales

RNF1: Todos los datos sensibles, incluyendo credenciales de usuario deben ser encriptados utilizando algoritmos seguros como SHA-256.

RNF2: Las operaciones básicas, como el inicio de sesión, la carga y visualización de vídeos, y la gestión de usuarios, deben completarse en menos de 2 segundos bajo condiciones normales de uso.

RNF3: La interfaz de usuario ha de ser diseñada siguiendo principios de diseño centrado en el usuario para facilitar la usabilidad de esta.

RNF4: La aplicación debe ser totalmente funcional en los navegadores más utilizados (Chrome, Firefox, Safari y Edge) en sus versiones más recientes. Debe tener un diseño responsivo que asegure una experiencia de usuario óptima en dispositivos de escritorio, tablets y móviles.

B.4. Especificación de requisitos

En este apartado se especifican los casos de uso correspondientes a los requisitos funcionales previamente expuestos así como los actores de la aplicación.

Actores

En esta aplicación se pueden distinguir tres actores:

- Administrador: Usuario con sesión de administrador iniciada que tiene acceso a funcionalidades de añadir, modificar y eliminar usuarios.
- Doctor: Usuario con sesión de doctor que tiene acceso a funcionalidades de gestión de pacientes y medicinas así como de visualización de evolución del paciente.
- Paciente: Usuario con sesión de paciente acceso a la funcionalidad de visualizar su evolución.

Además en este apartado se muestra el diagrama de casos de uso (ver figura B.1) y las siguientes tablas:

- La tabla B.1 contiene el caso de uso Iniciar sesión.
- La tabla B.2 contiene el caso de uso Cerrar sesión.
- La tabla B.3 contiene el caso de uso Visualizar evolución.
- La tabla B.4 contiene el caso de uso Dar de alta a usuarios.
- La tabla B.5 contiene el caso de uso Dar de baja a usuarios.
- La tabla B.6 contiene el caso de uso Modificar usuarios.
- La tabla B.7 contiene el caso de uso Visualizar listado de usuarios.
- La tabla B.8 contiene el caso de uso Añadir vídeos.

- La tabla B.9 contiene el caso de uso Eliminar vídeos.
- La tabla B.10 contiene el caso de uso Clasificar vídeos.
- La tabla B.11 contiene el caso de uso Asignar paciente.
- La tabla B.12 contiene el caso de uso Desasignar paciente.
- La tabla B.13 contiene el caso de uso Añadir medicina.
- La tabla B.14 contiene el caso de uso Eliminar medicina.
- La tabla B.15 contiene el caso de uso Visualizar listado de medicinas de un paciente.

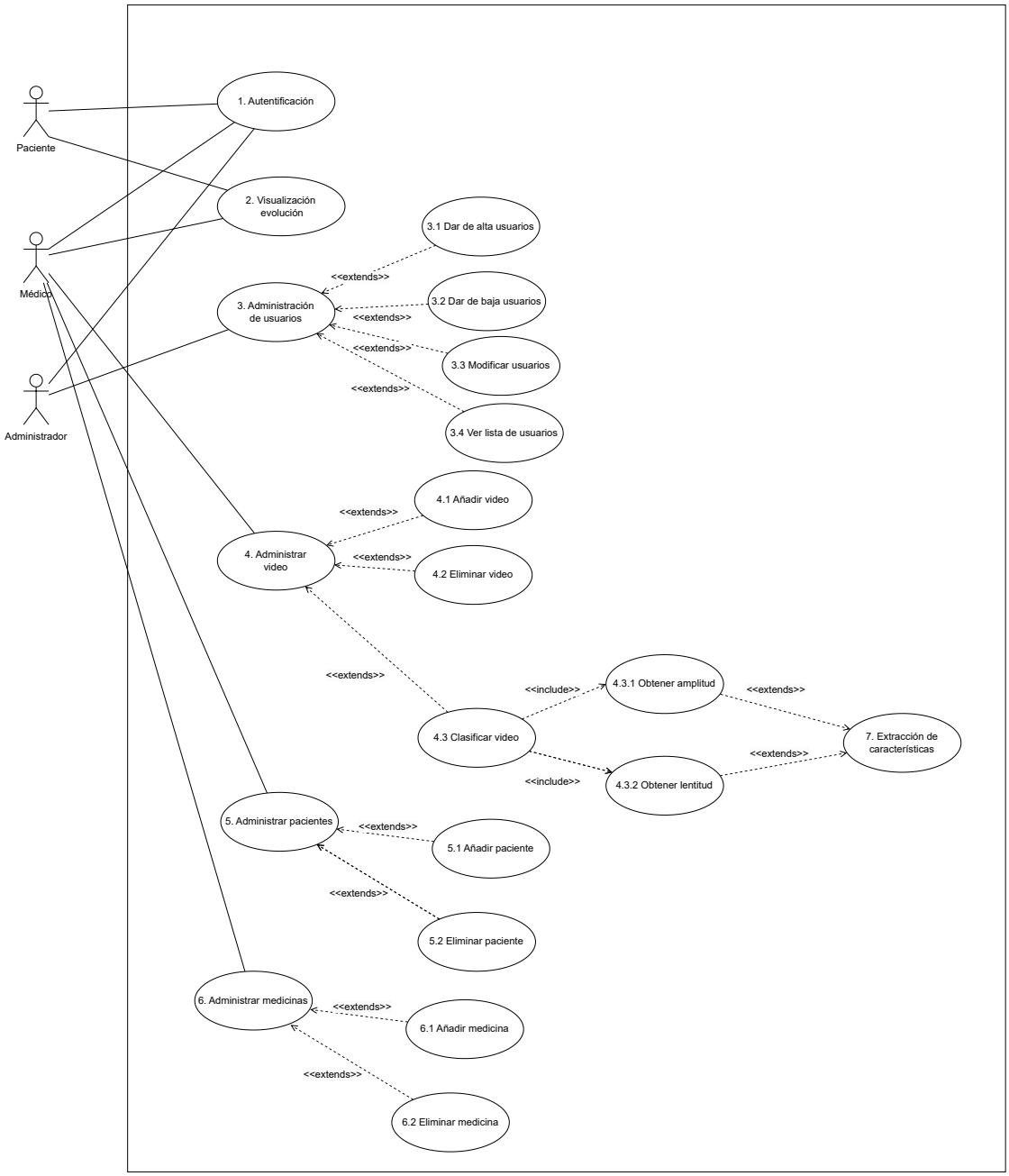

Figura B.1: Diagrama de casos de uso

CU-1	**Iniciar sesión**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-1,RF-6
Descripción	Los usuarios deben ser capaces de iniciar sesión
Precondición	▪ El usuario debe estar dado de alta en la base de datos. ▪ El usuario debe estar en la página de inicio de sesión. ▪ El usuario no debe tener una sesión iniciada.
Acciones	▪ El usuario introduce su nombre de usuario. ▪ El usuario introduce su contraseña. ▪ El usuario selecciona su rol. ▪ El usuario hace clic en el botón de iniciar sesión.
Postcondición	El usuario será redirijo a su página de perfil correspondiente
Excepciones	▪ Las credenciales no son correctas. ▪ El usuario no está dado de alta.
Importancia	Alta

Tabla B.1: CU-1 Iniciar sesión.

CU-2	**Cerrar sesión**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-1,RF-7
Descripción	Los usuarios deben ser capaces de cerrar sesión
Precondición	▪ El usuario debe tener la sesión iniciada.
Acciones	▪ El usuario hace clic en el botón de finalizar sesión.
Postcondición	El usuario será redirijo a la página de inicio
Excepciones	
Importancia	Alta

Tabla B.2: CU-2 Cerrar sesión.

CU-3	**Visualizar evolución**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-14
Descripción	El paciente y el doctor han de ser capaces de visualizar la evolución del paciente
Precondición	▪ El usuario debe de tener una sesión de doctor o paciente iniciada. ▪ El paciente debe estar dado de alta en la base de datos. ▪ El doctor debe ser el que esta asignado al paciente.
Acciones	▪ El usuario accede a su perfil. ▪ Se visualiza la gráfica de la evolución del paciente.
Postcondición	Se muestra al usuario su gráfico correspondiente
Excepciones	▪ El doctor no corresponde con el del paciente.
Importancia	Alta

Tabla B.3: CU-3 Visualizar evolución.

CU-4.1	**Dar de alta a usuarios**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-8
Descripción	El administrador debe poder dar de alta a nuevos usuarios.
Precondición	▪ El usuario debe de tener una sesión de administrador iniciada. ▪ El usuario que se desea añadir no debe estar dado de alta.
Acciones	▪ El administrador pulsa el botón de «añadir usuario». ▪ El administrador introduce los datos del nuevo usuario. ▪ El administrador confirma los datos y pulsa el botón «añadir».
Postcondición	Se muestra el nuevo usuario en la lista de usuarios
Excepciones	▪ El usuario ya existe. ▪ Los datos introducidos son erróneos o incompletos.
Importancia	Alta

Tabla B.4: CU-4.1 Dar de alta a usuarios.

CU-4.2	**Dar de baja a usuarios**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-9
Descripción	El administrador debe poder dar de baja a usuarios.
Precondición	▪ El usuario debe de tener una sesión de administrador iniciada. ▪ El usuario que se desea eliminar debe estar dado de alta.
Acciones	▪ El administrador pulsa el botón de «eliminar usuario». ▪ El administrador confirma el usuario y pulsa el botón «eliminar».
Postcondición	Ya no se muestra al usuario en la lista de usuarios
Excepciones	▪ El usuario no existe. ▪ El usuario es un doctor y tiene pacientes asignados. ▪ El usuario que se desea eliminar es el mismo que el usuario de la sesión.
Importancia	Alta

Tabla B.5: CU-4.2 Dar de baja a usuarios.

CU-4.3	**Modificar usuarios**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-10
Descripción	El administrador debe poder modificar usuarios.
Precondición	▪ El usuario debe de tener una sesión de administrador iniciada. ▪ El usuario debe estar dado de alta.
Acciones	▪ El administrador pulsa el botón de «modificar usuario». ▪ El administrador realiza los cambios del usuario y pulsa el botón «confirmar».
Postcondición	Se muestra la modificación del usuario.
Excepciones	▪ El usuario modificado coincide con otro que ya existe.
Importancia	Alta

Tabla B.6: CU-4.3 Modificar usuarios.

CU-4.4	**Visualizar listado de usuarios**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-11
Descripción	El administrador debe poder visualizar la lista de usuarios.
Precondición	▪ El usuario debe de tener una sesión de administrador iniciada. ▪ Debe existir al menos 1 usuario.
Acciones	▪ El administrador visualiza la lista de usuarios.
Postcondición	Se muestra la lista de usuarios.
Excepciones	
Importancia	Media

Tabla B.7: CU-4.4 Visualizar listado de usuarios.

CU-5.1	**Añadir vídeos**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-2
Descripción	Los doctores han de ser capaces de añadir nuevos vídeos a sus pacientes.
Precondición	▪ El usuario debe de tener una sesión de doctor iniciada. ▪ El doctor del paciente ha de ser el mismo que el doctor de la sesión. ▪ Debe existir el paciente al que se le quiere añadir el vídeo.
Acciones	▪ El usuario pulsa el botón «añadir vídeo». ▪ Se rellenan los datos correspondientes del vídeo como la mano. ▪ Se añade el fichero con el vídeo que se quiere subir. ▪ Se pulsa el botón «añadir».
Postcondición	El nuevo vídeo debe estar en la lista de vídeos del paciente correspondiente.
Excepciones	▪ No se han rellenado los datos correctamente. ▪ El fichero subido no corresponde con formato solicitado.
Importancia	Alta

Tabla B.8: CU-5.1 Añadir vídeos.

CU-5.2	**Eliminar vídeos**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-3
Descripción	Los doctores han de ser capaces de eliminar vídeos de sus pacientes.
Precondición	▪ El usuario debe de tener una sesión de doctor iniciada. ▪ El doctor del paciente ha de ser el mismo que el doctor de la sesión. ▪ Debe existir el paciente al que se le quiere eliminar el vídeo. ▪ El paciente debe tener al menos un vídeo.
Acciones	▪ El usuario pulsa el botón «eliminar vídeo» al lado del vídeo que desea eliminar. ▪ El usuario confirma que se desea eliminar el vídeo.
Postcondición	El vídeo ya no esta en la lista de vídeos del paciente correspondiente.
Excepciones	
Importancia	Media

Tabla B.9: CU-5.2 Eliminar vídeos.

CU-5.3	**Clasificar vídeos**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-5
Descripción	Los doctores han de ser capaces de clasificar los vídeos del paciente.
Precondición	▪ El usuario debe de tener una sesión de doctor iniciada. ▪ El doctor del paciente ha de ser el mismo que el doctor de la sesión. ▪ Debe existir el paciente al que se le quiere clasificar el vídeo.
Acciones	▪ El doctor pulsa el botón «añadir vídeo». ▪ Se clasificará el vídeo correspondiente.
Postcondición	Aparecerá la clasificación del vídeo.
Excepciones	
Importancia	Alta

Tabla B.10: CU-5.3 Clasificar vídeos.

CU-6.1	**Asignar paciente**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-12
Descripción	El administrador ha de ser capaz de asignar pacientes a la lista de pacientes de un doctor.
Precondición	▪ El usuario debe de tener una sesión de administrador iniciada. ▪ Debe existir al menos un doctor.
Acciones	▪ El administrador registra un nuevo paciente. ▪ El administrador selecciona al doctor correspondiente. ▪ El administrador registra al paciente.
Postcondición	Aparecerá el paciente en la lista del doctor.
Excepciones	▪ El paciente no existe.
Importancia	Alta

Tabla B.11: CU-6.1 Asignar paciente.

CU-6.2	**Desasignar paciente**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-13
Descripción	El administrador ha de ser capaz de desasignar pacientes de la lista de un doctor.
Precondición	▪ Debe existir el paciente que se desea eliminar. ▪ El paciente debe estar incluido en la lista de pacientes del doctor.
Acciones	▪ El administrador pulsa el botón «eliminar paciente». ▪ Se selecciona el paciente que se desea eliminar. ▪ Se confirma que se desea eliminar a ese paciente.
Postcondición	Ya no aparecerá el paciente en la lista del doctor.
Excepciones	▪ El paciente no se encuentra en la lista del doctor.
Importancia	Media

Tabla B.12: CU-6.2 Desasignar paciente.

CU-7.1	**Añadir medicina**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-15
Descripción	El administrador debe poder dar de alta a nuevos usuarios.
Precondición	▪ El usuario debe de tener una sesión de doctor iniciada. ▪ El doctor del paciente ha de ser el mismo que el doctor de la sesión. ▪ Debe existir el paciente al que se le quiere añadir la medicina.
Acciones	▪ El doctor pulsa el botón de «añadir medicina». ▪ El doctor introduce los datos de la nueva medicina. ▪ El doctor confirma los datos y pulsa el botón «añadir».
Postcondición	Se muestra la nueva medicina en la lista de medicinas del paciente.
Excepciones	▪ Los datos introducidos son erróneos o incompletos.
Importancia	Media

Tabla B.13: CU-7.1 Añadir medicina.

CU-7.2	**Eliminar medicina**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-9
Descripción	El administrador debe poder dar de baja a usuarios.
Precondición	▪ El usuario debe de tener una sesión de doctor iniciada. ▪ El doctor del paciente ha de ser el mismo que el doctor de la sesión. ▪ Debe existir el paciente al que se le quiere añadir la medicina. ▪ El paciente ha de tener al menos una medicina.
Acciones	▪ El doctor pulsa el botón de «eliminar usuario». ▪ El doctor confirma el usuario y pulsa el botón «eliminar».
Postcondición	Ya no se muestra la medicina en la lista de medicinas
Excepciones	
Importancia	Media

Tabla B.14: CU-7.2 Eliminar medicina.

CU-7.3	**Visualizar listado de medicinas de un paciente**
Versión	1.0
Autor	Jorge Martínez Martín
Requisitos asociados	RF-11
Descripción	El administrador debe poder visualizar la lista de usuarios.
Precondición	▪ El usuario debe de tener una sesión de doctor iniciada. ▪ El doctor del paciente ha de ser el mismo que el doctor de la sesión. ▪ Debe existir el paciente al que se le quiere añadir la medicina.
Acciones	▪ El doctor visualiza la lista de medicinas del paciente seleccionado.
Postcondición	Se muestra la lista de medicinas del paciente.
Excepciones	
Importancia	Media

Tabla B.15: CU-7.3 Visualizar listado de medicinas de un paciente.

Apéndice C

Especificación de diseño

C.1. Introducción

En el siguiente apartado se describen los detalles de la estructura del *software*: el comportamiento y la interacción entre los distintos componentes.

C.2. Diseño de datos

Para implementar la persistencia de los datos se ha hecho uso de una base de datos SQLite debido a que no requiere instalación separada y es fácil de integrar con el framework de Flask. Este diseño se ha dividido en varias fases: diseño del diagrama entidad-relación, diseño del diagrama relacional y diseño del diccionario de datos.

Modelo entidad-relación

Para representar gráficamente las relaciones entre los datos, se ha creado un diagrama E-R (ver figura C.1). En el se puede observar una ISA exclusiva total. Esto se debe a que, tras consultarlo con el profesor Jesús Maudes, se llegó a la conclusión de que crear una ISA exclusiva total era una solución válida debido a que se quiere evitar que un mismo usuario tenga acceso a varias funcionalidades. En caso de que esto se requiriese (si un doctor necesita permisos de administración) se crearía otra cuenta de forma más controlada.

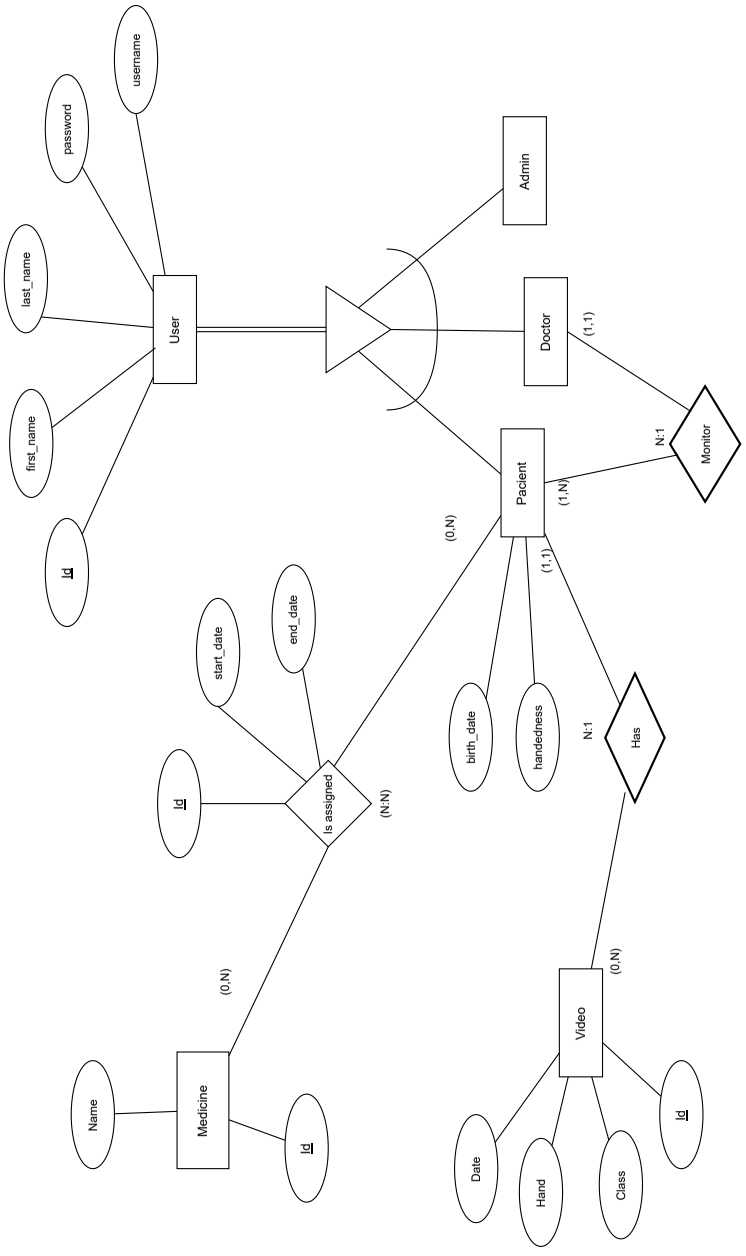

Figura C.1: Diagrama entidad-relación

Modelo relacional

Partiendo del modelo entidad-relación (ver figura C.1) se ha creado el diagrama relacional (ver figura C.2). A continuación se mencionan algunas de las decisiones más importantes que se han decido tomar a la hora de crear este diagrama. Con respecto a la ISA se ha decidido dividir en tres tablas distintas ya que en un principio un usuario solo va a poder tener uno de los tres roles asignados y en caso de que se necesitase que un doctor tuviera el rol de administrador, que es el un caso excepcional, se le otorgaría otra cuenta distinta con los permisos de administración. De esta manera podremos controlar los accesos de cada usuario de manera más limpia y exclusiva. El rol de administrador es un rol que pese a que no se relacione con ninguna otra entidad, se ha visto que es necesario para poder permitir el acceso a los controles de administrador.

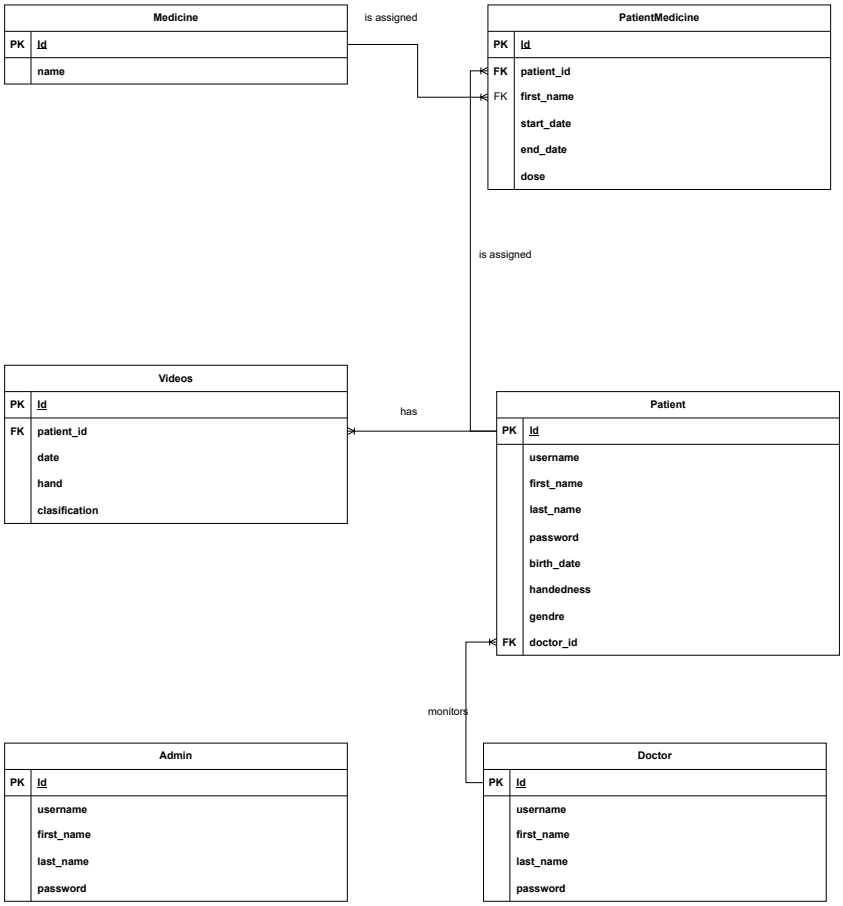

Figura C.2: Diagrama relacional de la aplicación

Diccionario de datos

Se adjunta a continuación, para cada tabla, el diccionario de datos correspondiente.

- **Administradores**: la tabla C.1 contiene la información correspondiente a los administradores de la aplicación.
- **Doctores**: la tabla C.2 contiene la información correspondiente a los doctores.
- **Pacientes**: la tabla C.3 contiene la información correspondiente a los pacientes.

Nombre	Tipo	Columna	Descripción
id	INTEGER	PK	Identificador único del administrador.
username	VARCHAR(100)	UNIQUE NOT NULL	Nombre de usuario del administrador. Debe ser único.
password	VARCHAR(100)	NOT NULL	Contraseña del administrador, cifrada usando sha-256.
first_name	VARCHAR(100)	NOT NULL	Primer nombre del administrador.
last_name	VARCHAR(100)	NOT NULL	Apellidos del administrador.

Tabla C.1: Diccionario de datos. Tabla correspondiente a la clase `Admin`.

Nombre	Tipo	Columna	Descripción
id	INTEGER	PK	Identificador único del doctor.
username	VARCHAR(100)	UNIQUE NOT NULL	Nombre de usuario del doctor. Debe ser único.
password	VARCHAR(100)	NOT NULL	Contraseña del doctor, cifrada usando sha-256.
first_name	VARCHAR(100)	NOT NULL	Primer nombre del doctor.
last_name	VARCHAR(100)	NOT NULL	Apellidos del doctor.

Tabla C.2: Diccionario de datos. Tabla correspondiente a la clase `Doctor`.

Nombre	Tipo	Columna	Descripción
id	INTEGER	PK	Identificador único del paciente.
username	VARCHAR(100)	UNIQUE NOT NULL	Nombre de usuario del paciente. Debe ser único.
password	VARCHAR(100)	NOT NULL	Contraseña del paciente, cifrada usando sha-256.
first_name	VARCHAR(100)	NOT NULL	Primer nombre del paciente.
last_name	VARCHAR(100)	NOT NULL	Apellidos del paciente.
birth_date	DATE	NOT NULL	Fecha de nacimiento del paciente.
handedness	ENUM('right', 'left')	NOT NULL	Mano dominante del paciente (derecha o izquierda).
gender	ENUM('M', 'F')	NOT NULL	Género del paciente (masculino o femenino).
doctor_id	INTEGER	FK(doctor.id)	Clave foránea a la tabla Doctor. Indica el doctor a cargo del paciente.

Tabla C.3: Diccionario de datos. Tabla correspondiente a la clase `Patient`.

Nombre	Tipo	Columna	Descripción
`id`	`INTEGER`	`PK`	Identificador único del video.
`patient_id`	`INTEGER`	`FK (patient.id)`	Clave foránea a la tabla Paciente. Indica el paciente asociado al video.
`hand`	`ENUM('left', 'right')`	`NOT NULL`	Mano usada en el vídeo (izquierda o derecha).
`date`	`DATE`	`NOT NULL`	Fecha de grabación del video.
`video_data`	`LargeBinary`	`NOT NULL`	Datos binarios del vídeo.
`amplitude`	`VARCHAR(100)`	`NOT NULL`	Amplitud clasificada para el video.
`slowness`	`VARCHAR(100)`	`NOT NULL`	Lentitud clasificada para el video.

Tabla C.4: Diccionario de datos. Tabla correspondiente a la clase `Video`.

Nombre	Tipo	Columna	Descripción
`id`	`INTEGER`	`PK`	Identificador único del medicamento.
`name`	`VARCHAR(100)`	`NOT NULL`	Nombre del medicamento.

Tabla C.5: Diccionario de datos. Tabla correspondiente a la clase `Medicine`.

Nombre	Tipo	Columna	Descripción
`id`	`INTEGER`	`PK`	Identificador único de la asociación entre paciente y medicamento.
`patient_id`	`INTEGER`	`FK (patient.id)`	Clave foránea a la tabla Paciente. Indica el paciente que toma el medicamento.
`medicine_id`	`INTEGER`	`FK (medicine.id)`	Clave foránea a la tabla Medicamento. Indica el medicamento que toma el paciente.
`dosage`	`VARCHAR(100)`	`NOT NULL`	Dosis del medicamento para el paciente.
`start_date`	`DATE`	`NOT NULL`	Fecha de inicio del tratamiento con el medicamento.
`end_date`	`DATE`	`NULL`	Fecha de fin del tratamiento con el medicamento (opcional).

Tabla C.6: Diccionario de datos. Tabla correspondiente a la clase `PatientMedicine`.

- **Vídeos**: la tabla C.4 contiene la información correspondiente a los vídeos de los pacientes pacientes.
- **Medicinas**: la tabla C.5 contiene la información de las medicinas.
- **PacienteMedicinas**: la tabla C.6 contiene la información de las medicinas administradas a cada paciente incluyendo la fecha de inicio del tratamiento así como la fecha de finalización.

C.3. Diseño procedimental

El diseño procedimental en informática es una metodología de desarrollo de software que se centra en la creación de procedimientos o funciones para llevar a cabo tareas específicas dentro de un programa. Durante el desarrollo del proyecto se ha hecho uso de la herramienta Draw.io[1] para representar los procedimientos. Dentro del sistema podemos distinguir los siguientes componentes:

- Doctor: representado por el monigote, representa a un usuario que haya iniciado sesión como doctor.
- Navegador: se refiere al apartado de *frontend* de la aplicación.
- Servidor web: se refiere a la parte de *backend*.
- Paddel: se refiere a la librería de Paddel utilizada durante el proyecto.
- Base de datos: se refiere a la base de datos de la aplicación.

Debido a que los diagramas secuenciales de la aplicación son muy básicos exceptuando el de la clasificación de los vídeos, se ha tomado la decisión de omitirlos.

Clasificación de los videos

Para esta funcionalidad, el doctor ha de tener al menos un paciente para poder realizar la acción. El doctor solicita la vista de añadir vídeos al usuario. Tras recibirla, el doctor envía la información del vídeo junto con el archivo del mismo a la aplicación. La aplicación lo envía a la librería Paddel para extraer las características y estas características se envían a los modelos entrenados para obtener las clasificaciones. Una vez obtenidas el vídeo es guardado en la base de datos y posteriormente, se envía al usuario de vuelta a la página de gestión de vídeos actualizada del paciente al que se le ha añadido. La figura C.3 muestra el diagrama de secuencia del proceso.

[1] Fuente: draw.io

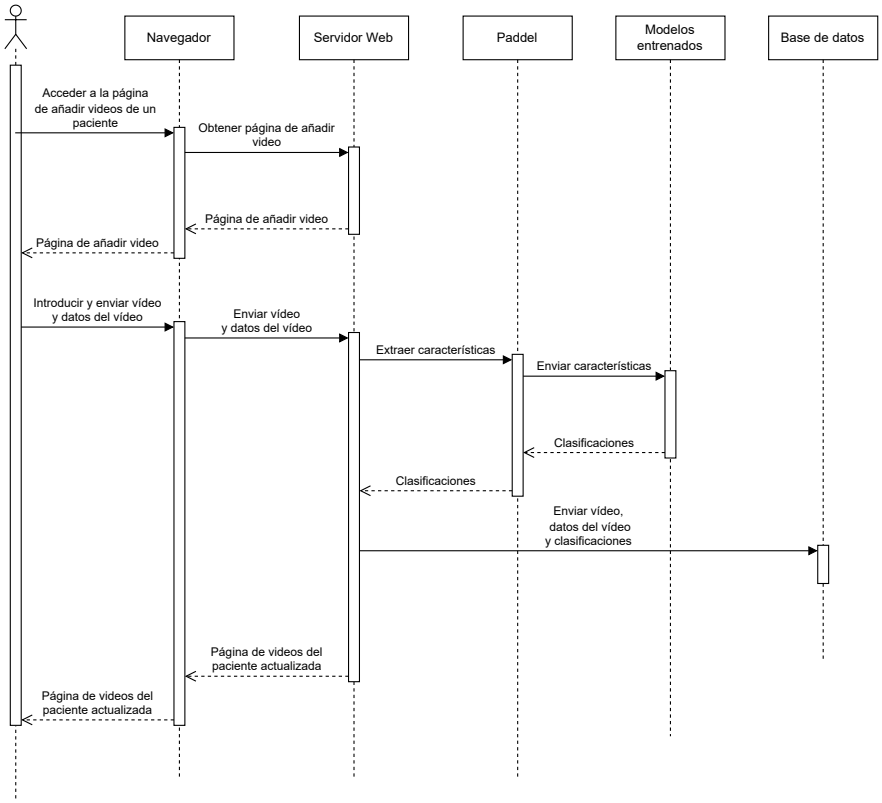

Figura C.3: Diagrama de secuencia de clasificación de los vídeos

C.4. Diseño arquitectónico

Para ilustrar la arquitectura de la aplicación se emplea el diagrama de despliegue de la figura C.4. Este tipo de diagrama UML muestra cómo están organizados físicamente los componentes del software en el sistema.

La aplicación no se ha desplegado debido al gran tamaño de los archivos, incluidos los vídeos y los datos de los sensores de los pacientes, por lo que la sección del Web Server no es precisa, ya que la lógica de la aplicación se ejecuta localmente.

El Data Base Server se encarga de gestionar la capa de persistencia del proyecto. Para ello, se ha utilizado SQLite como herramienta de gestión de la base de datos.

El usuario o cliente actúa como el controlador de la interfaz de usuario, interactuando con las páginas HTML a través de su navegador.

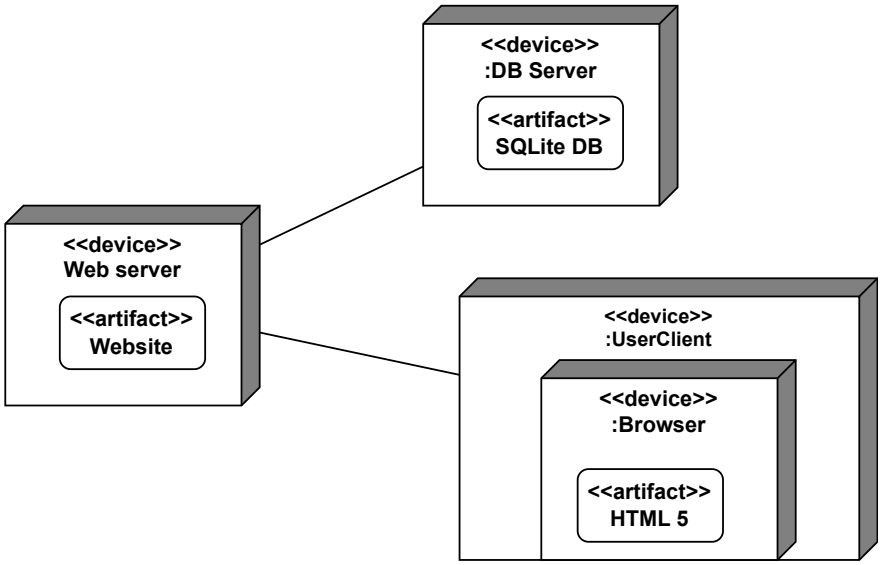

Figura C.4: Diagrama de despliegue

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Este anexo incluirá una reflexión personal del alumnado sobre los aspectos de la sostenibilidad que se abordan en el trabajo. Se pueden incluir tantas subsecciones como sean necesarias con la intención de explicar las competencias de sostenibilidad adquiridas durante el alumnado y aplicadas al Trabajo de Fin de Grado.

Más información en el documento de la CRUE https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sosteniblidad_Crue2012.pdf.

Este anexo tendrá una extensión comprendida entre 600 y 800 palabras.

Bibliografía

[1] Catalin Andrei Cacuci. Github:paddel. https://github.com/cataand/tfg-paddel, 2024.

[2] Taha Khan, Dag Nyholm, Jerker Westin, and Mark Dougherty. A computer vision framework for finger-tapping evaluation in parkinson's disease. *Artificial intelligence in medicine*, 60(1):27–40, 2014.

[3] Marta Palacio. *Scrum Master. Temario troncal 1.* 02 2022.

[4] Stefan Williams, Samuel D Relton, Hui Fang, Jane Alty, Rami Qahwaji, Christopher D Graham, and David C Wong. Supervised classification of bradykinesia in parkinson's disease from smartphone videos. *Artificial Intelligence in Medicine*, 110:101966, 2020.

[5] Stefan Williams, Zhibin Zhao, Awais Hafeez, David C Wong, Samuel D Relton, Hui Fang, and Jane E Alty. The discerning eye of computer vision: Can it measure parkinson's finger tap bradykinesia? *Journal of the Neurological Sciences*, 416:117003, 2020.