UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Identificación de Parkinson por visión artificial

Documentación Técnica

Presentado por Catalin Andrei Cacuci
en Universidad de Burgos — 20 de mayo de 2023
Tutora: Alicia Olivares Gil
Cotutor: Álvar Arnaiz González

Índice general

Índice de figuras

Índice de tablas

Apéndice A. Plan de Proyecto Software

A.1. Introducción

El propósito del plan de proyecto es definir de forma clara cuales son los objetivos del proyecto y de qué manera van a ser alcanzados.

A.2. Planificación temporal

La planificación del proyecto se llevó a cabo mediante una metodología ágil, en concreto Scrum, en un principio se utilizó la herramienta ZenHub, pero debido a cambios en la política de la misma la misma, dejó de ser utilizable en este proyecto.

Scrum tiene por objetivo llevar a cabo una progresión incremental, dividiendo el tiempo de trabajo en partes denominadas *sprint*. Antes de comenzar cada uno de estos *sprints* se realiza una reunión de planificación con los miembros del equipo en la que se asigna un determinado número de tareas que se intentará realizar durante dicho *sprint*. Las tareas realizadas durante el periodo de tiempo asignado al *sprint* se marcan como completadas, las que no puedan ser realizadas a tiempo pasarán al siguiente *sprint*.

En este caso concreto la duración de cada *sprint* se ha fijado a una semana.

Preparación

Antes de comenzar el sprint 1 se realizaron las siguientes tareas:

- Organizar el proyecto en diferentes carpetas con las partes que lo componen.
- Implementar la conectividad básica entre los contenedores de Docker que implementan la aplicación web.
- Implementar el parseado de los nombres de archivo de los vídeos.
- Implementar la extracción de puntos del esqueleto de la mano mediante Mediapipe Hands.
- Crear un paquete instalable o librería de Python con las utilidades que se utilizan en la fase de investigación para facilitar su uso posterior en la aplicación web.

Haber decidido en esta fase inicial la estructura que va a tener el proyecto y, de forma aproximada, las herramientas que van a ser utilizadas a ayudado en gran medida durante el desarrollo, debido a que las funcionalidades de todas las piezas que componen el proyecto están bien definidas y separadas.

Sprint 1 (6/2/2023 – 12/2/2023)

En este sprint se avanzó en la parte del preprocesado de los vídeos, extrayendo diferentes atributos a partir del esqueleto que obtenido a través de MediaPipe.

Además, se separó el proceso de construcción de los contenedores de Docker dependiento de si se lanzan en un entorno de desarrollo o en producción. Optimizando el despliegue en producción, mientras que, en desarrollo, se muestran mensajes adicionales y observa los archivos para reaccionar a cambios.

Tareas realizadas:

- Añadir extracción de frecuencia de toques a partir de la secuencia de poses de la mano extraida por Mediapipe.
- Añadir extracción de diferencia entre la frecuencia de toques del intervalo de tiempo inicial y final de la secuencia de poses.
- Establecer ángulo máximo para la detección de toques.
- Cambiar la configuración de los contenedores de Docker para usar multietapa, separando la configuración para los entornos de desarrollo y producción.

Sprint 2 (13/2/2023 – 19/2/2023)

Este sprint se centra en el preprocesado de los vídeos, añadiendo la extracción de varias características utilizadas en trabajos anteriores en este área de las series temporales con las poses de la mano.

Tareas realizadas:

- Añadir extracción de amplitud media.
- Añadir extracción de diferencia de amplitud media entre intervalo inicial y final del vídeo.
- Añadir extracción de variación de amplitud.
- Añadir extracción de velocidad del movimiento.
- Comenzar manual del programador.
- Reemplazar Nginx por Caddy como proxy inverso de la aplicación.
- Actualizar fichero README.md.

Sprint 3 (20/2/2023 – 26/2/2023)

En este sprint se añadió la extracción de caracerísticas mediante la librería TSFresh que extrae alrededor de 750 características adicionales a las que se extraían hasta este punto.

Tareas realizadas:

- Añadir extracción de características mediante TSFresh.
- Cambiar secuencia de instalación del contenedor de Docker para la API mejorando el cacheado de los pasos.

Sprint 4 (27/2/2023 – 5/3/2023)

En este sprint se cambió la extracción de características propias para aprovechar las metodologías ya implementadas por TSFresh, además de realizar algunos cambios sobre la memoria.

Tareas realizadas:

- Añadir herramientas de desarrollo a la memoria.

- Cambiar implementación de extracción de características de trabajos previos para que sea compatible con la librería TSFresh.

Sprint 5 (6/3/2023 – 12/3/2023)

Tareas realizadas:

- Actualizar sección de herramientas de desarrollo.

Sprint 6 (13/3/2023 – 19/3/2023)

En este sprint se comienzan a entrenar los primeros modelos de aprendizaje mediante una técnica que se conoce como *grid search*. Además se mejora la implementación de la extracción de las marcas temporales de las series temporales.

Tareas realizadas:

- Implementar optimización de hiperparámetros para varios modelos mediante *grid search*.
- Añadir información temporal a la extracción de poses de Mediapipe a partir de la tasa de fotogramas.

Sprint 7 (20/3/2023 – 26/3/2023)

En este sprint se cambian algunos detalles sobre la optimización de hiperparámetros, utilizando más modelos y cambiando los parámetros de la validación cruzada.

Tareas realizadas:

- Añadir Perceptrón multicapa, Adaboost y XGBoost a los modelos de la optimización de hiperparámetros.
- Cambiar validación cruzada para utilizar 5 repeticiones de 2 *folds*.
- Sustituir características de mano grabada y mano dominante por una única características, si está utilizando la mano dominante.

Sprint 8 (27/3/2023 – 2/4/2023)

En este sprint se crean visualizaciones de los resultados obtenidos durante la optimización de hiperparámetros, además, se añade un parámetro adiciona, el número de características a seleccionar.

Tareas realizadas:

- Arreglar Makefile para la compilación de la documentación.
- Añadir generación de gráficas con los resultados obtenidos de la optimización de hiperparámetros mediante *grid search*.
- Añadir selección del número de características a utilizar a la optimización de hiperparámetros.
- Refactorizar librería.
- Cambiar el framework de JavaScript de la web de SvelteKit a NextJS.
- Implementar una barra de navegación básica.

Sprint 9 (3/4/2023 – 9/4/2023)

En este sprint se realizaron varios cambios sobre los diferentes componentes del proyecto.

Tareas realizadas:

- Actualizar el manual del programador con los cambios realizados hasta este punto.
- Añadir requisitos de la apliación web.
- Añadir diagrama general de casos de uso.
- Cambiar la estructura de directorios del frontend de la web.
- Añadir y configurar contenedor de base de datos PostgreSQL al proyecto de Docker.
- Configurar la API para interactuar con la base de datos, creando, los modelos de SQLAlchemy con los que se va a trabajar.
- Añadir algunos endpoints para permitir una gestión muy básica de usuarios.

- Bloquear la versión de los contenedores de Docker para evitar problemas de dependencias en el futuro.
- Añadir persistencia de datos entre creaciones del contenedor de la base de datos.
- Bloquear versiones de las dependencias de Python. Debido a la versión 2 de Pandas, se rompe parte de la funcionalidad.
- Implementar una versión inicial del endpoint para obtener una predicción a partir de un vídeo.

Sprint 10 (10/4/2023 – 16/4/2023)

En este sprint se realizaron varias tareas sobre la aplicación web para permitir la fácil interacción con el formulario para obtener predicciones.

Tareas realizadas:

- Añadir y estilizar un formulario básico para interactuar desde el frontend con la API y obtener un predicción.
- Utilizar JavaScript para enviar los contenidos de este formulario de forma correcta al endpoint.
- Añadir barra de progreso de subida del vídeo una vez el formulario es enviado.
- Implementar el entrenamiento del modelo final con todos los datos y añadirlo a la API para poder realizar la predicción.
- Añadir funcionalidad de arrastrar y soltar al formulario para subir el vídeo.

Sprint 11 (17/4/2023 – 23/4/2023)

En este sprint se internacionalizó la aplicación web, entre otras cosas de menor importancia.

Tareas realizadas:

- Crear prototipos (mockups) para la página principal y para la gestión de modelos.

- Internacionalizar la aplicación mediante TypeSafe-i18n y añadir un selector de idioma.
- Añadir autenticación mediante tókenes JWT (JSON Web Token) a la API.

Sprint 12 (24/4/2023 – 30/4/2023)

En este sprint se solucionaron algunos bugs y mejoró la accesibilidad de la apliación web en gran medida gracias a un curso sobre accesibilidad que se realizó.

Tareas realizadas:

- Implementar la autenticación de usuario con la API desde la página web.
- Refactorizar y limpiar el código de la web.
- Solucionar bug de idioma indefinido.
- Pulir los estilos de algunos elementos de la web.
- Solucionar bug pantalla de administración se muestra brevemente antes de login.
- Mejorar accesibilidad del HTML.
- Añadir mensajes acompañando a la predicción dependiendo del resultado.

Sprint 13 (1/5/2023 – 7/5/2023)

En este sprint se implementó la administración de usuarios desde la página web.

- Pulir la pantalla de resultados de la web.
- Añadir ejecución de un script SQL por defecto en la creación de la base de datos.
- Añadir endpoints para crear y eliminar usuarios en la API.
- Añadir página de gestión de usuarios.

A.3. Estudio de viabilidad

En este apartado se analiza algunos aspectos que de deberían tener el cuenta si el producto creado en este proyecto fuese llevado a un entorno más real, en el que se tuviese que responder ante otras personas si hubiese algún problema.

Viabilidad económica

En este subapartado se va a realizar un pequeño estudio económico sobre los costes y las ganancias de este proyecto.

Costes

Existen varios costes, tanto fijos como variables, que tendría esta aplicación

Empleados Durante el transcurso del desarrollo del proyecto se han invertido aproximadamente 300 horas de trabajo durante un periodo de 5 meses.

Estableciendo el salario del alumno a 15€ por hora se obtiene el siguiente salario mensual bruto:

$$300 \text{ horas} * 15\frac{€}{hora} \ / \ 5 \text{ meses} = 900\frac{€}{mes}$$

Para obtener el gasto que supone este salario se deben añadir los impuestos que corren a cargo de la empresa, estos son los siguientes según la Seguridad Social:

- **Contingencias comunes**: Para tratar alteraciones de la salud no relacionadas con un accidente de trabajo o enfermedad profesional. Corresponde con el 23,60 % del salario.
- **Desempleo**: Correspondiente al 5,50 %.
- **FOGASA** (Fondo de Garantía Salarial): Para abonar a los trabajadores el importe de salarios pendientes por diversas razones relacionadas con la quiebra de una empresa. Corresponde al 0.20 %.
- **Formación profesional**: Utilizado para incentivar la formación de trabajadores. Corresponde al 0.60 %.

Según los impustos anteriores el importe a pagar por la empresa en materia de salario es:

$$\frac{900\frac{€}{mes}}{1-(0{,}236+0{,}055+0{,}002+0{,}006)} = 1283{,}88\frac{€}{mes}$$

Hardware

Equipo para el desarrollo Para el desarrollo de la aplicación se han utilizado dos equipos informáticos, uno de escritorio y otro portátil, con un precio de alrededor de 1100€ y 1200€ respectivamente.

Estableciendo un periodo de amortización de 4 años para ambos bienes se obtiene un coste anual de:

$$\frac{1100€ + 1200€}{4\ años} = 575\frac{€}{años}$$

Como el proyecto ha sido realizado durante 5 meses la amortización total en este periodo de tiempo es de 239,58€.

Alojamiento de la aplicación La aplicación debe ser alojada en una máquina con prestaciones suficientes como para cumplir con los requisitos funcionales y no funcionales establecidos.

Una opción sería alquinar espacio en un centro de datos de colocación en el que instalar un equipo informático propio, esto tiene ciertas ventajas al tener completo control sobre los componentes, pero al mismo tiempo supone un gasto muy grande, ya que se debe que tener en cuenta el gasto inicial de adquisición del hardware que podría rondar entre 500€ y 1000€, además del gasto recurrente de alquiler de la colocación que, para un *rack* de una unidad suele estar en torno a los 100€ mensuales.

Otra opción, y por la que se ha decidido al final, es utilizar alguno de los servicios de alojamiento virtualizados ofrecidos por varios proveedores como Hetzner o OVH, se han barajado ambas opciones y, aunque OVH dispone de una red con un servicio más rápido para españa, Hetzner ofrece una mejor relación rendimiento / precio, por lo que el servicio utilizado para alojar la aplicación es el servidor virtualizado CPX21 (3 procesadores virtuales, 4GB de RAM, 80GB de almacenamiento, IPv4 y tráfico de subida de 20TB) de Hetzner, con un precio de 9,13€ por mes.

La aplicación ha sido alojada durante los últimos 3 meses, por lo que el coste total de esto ha sido de:

$$9,13\frac{€}{mes} * 3\text{meses} = 27,39€$$

Todo lo anterior resulta en unos costes totales por hardware de 266,97€.

Software La mayor parte de programas utilizados durante el desarrollo son gratuitos, la única excepción es la versión profesional del IDE PyCharm, que aunque ha sido gratuito gracias a la licencia para estudiantes, en un entorno empresarial esto no sería posible, el precio mensual de este producto es de 30,13€. Por lo que el coste total durante estos 5 meses sería de:

$$5 \text{ meses} * 30,13\frac{€}{mes} = 150,65€$$

Costes totales La siguiente tabla muestra de forma resumida los costes en los que se ha incurrido durante el desarrollo del proyecto.

Concepto	Coste
Empleados	1283,88€
Hardware	266,97€
Software	150,65
Total	**1701,50€**

Tabla A.1: Costes

Ganancias

Una opción que se ha tenido en cuenta para la obtención de ingresos es un modelo de suscripción donde los clientes pagan mensualmente para tener acceso a la aplicación. Pero al final, debido al carácter de código abierto de la aplicación se ha optado por permitir el uso de la aplicación para cualquier persona de forma gratuita, debido a esto, esta empresa no obtendrá beneficios, esto se podría solventar en cierto modo abriendo un portal de donaciones para intentar cubrir al menos los costes de alojamiento.

Viabilidad legal

La siguiente tabla recopila las licencias de las diferentes librerías y utilidades de las que depende la aplicación junto con las licencias bajo las que han sido distribuidas.

Librería	Licencia
numpy	3-Clause BSD License
pandas	3-Clause BSD License
scipy	3-Clause BSD License
sklearn	3-Clause BSD License
imbalanced-learning	MIT
pydantic	MIT
Mediapipe	Apache License 2.0
OpenCV Python	MIT
tsfresh	MIT
matrixprofile	Apache License 2.0
xgboost	Apache License 2.0
skopt	3-Clause BSD License
fastapi	MIT
uvicorn	3-Clause BSD License
sqlalchemy	MIT
psycopg2	GNU Lesser General Public License
python-multipart	Apache License 2.0
python-jose	MIT
passlib	2-Clause BSD License
caddy	Apache License 2.0
nodejs	MIT
sveltekit	MIT
tailwindcss	MIT
axios	MIT
resolve-accept-language	MIT
typesafe-i18n	MIT
PostgreSQL	PostgreSQL Licence (Similar a BSD y MIT)

Tabla A.2: Licencias

Todas las librerías y herramientas utilizadas tienen licencias libres típicas en programas de código abierto que permiten su uso de forma gratuita.

Gracias a esto la aplicación puede ser distribuida bajo casi cualquier otra licencia, en este caso se ha optado por utilizar la licencia MIT, que otorga a cualquier persona que obetenga una copia del proyecto a usarlo, modificarlo y redistribuirlo libremente, al mismo tiempo que el dueño está protegido de la responsabilidad legal de cualquier problema que pueda dar el uso de dicho programa.

Apéndice B. Especificación de Requisitos

B.1. Introducción

Antes de comenzar a crear un programa es importante detallar las características que deberá tener (quienes son los usuarios, qué acciones pueden realizar, de qué manera deberá responder el programa, etc.). En este caso se ha seguido UML (Unified Modelling Language) como estándar para definir estas características.

B.2. Objetivos generales

Durante la fase de investigación del proyecto se ha creado una multitud de modelos de inteligencia artificial con diferentes grados de precisión para detectar la enfermedad de Parkinson en un individuo. Estos modelos son almacenados como archivos binarios y los conocimientos específicos de programación requeridos para su uso hacen imposible que la persona media o personal médico pueda obtener una predicción.

El objetivo principal de esta aplicación es implementar una interfaz intuitiva para interactuar con los modelos sin la necesidad de que el usuario final altere de ninguna forma su equipo informático.

Como objetivo secundario, la aplicación debe permitir a los usuarios registrados seleccionar el modelo utilizado mediante la misma interfaz usada para realizar las predicciones.

B.3. Catálogo de requisitos

Requisitos funcionales

RF1: El sistema debe permitir diferenciar entre dos roles: Usuario normal (que no ha iniciado sesión) y Administrador.

RF2: Cualquier usuario debe poder subir un archivo de vídeo junto con información adicional para ser procesados por un modelo de inteligencia artifical y obtener una predicción.

RF3: Los usuarios administradores deben poder iniciar sesión con su nombre de usuario y contraseña respectivos.

RF4: Los usuarios administradores deben poder finalizar su sesión.

RF5: Un inicio de sesión debe ser mantenido entre cambios de página y cierres del navegador durante 24 horas.

RF6: Un administrador debe poder gestionar una lista con los modelos disponibles.

RF7: Un administrador debe poder añadir modelos a la lista de modelos disponibles.

RF8: Un administrador debe poder eliminar modelos de la lista de modelos disponibles.

RF9: Un administrador debe poder seleccionar de la lista de modelos disponibles aquel que se va a utilizar para realizar predicciones.

RF10: Un administrador debe tener acceso a una lista con todos los usuarios registrados en la aplicación.

RF11: Un administrador debe poder añadir nuevos usuarios de tipo administrador a la aplicación, especificando su nombre de usuario y contraseña.

RF12: Un administrador debe poder eliminar usuarios de la aplicación.

Requisitos no funcionales

RNF1: El tiempo desde que un usuario termina de subir el vídeo y sus datos hasta que recibe el resultado debe ser inferior a un minuto.

RNF2: El sistema debe ser fácil de usar, con una interfaz intuitiva que el usuario entienda de forma inmediata.

RNF3: La página web deberá utilizar etiquetas y atributos HTML de tal forma que se facilite la navegación para personas con accesibilidad limitada.

RNF4: La información de los usuario se debe almacenar de forma segura, cifrando campos pertinentes como la contraseña.

B.4. Especificación de requisitos

Actores

Existen dos tipos de actores en esta aplicación:

- **Usuario Normal**: Es aquel usuario que no ha iniciado sesión y tiene un acceso limitado a las capacidades del sistema.
- **Administrador**: Es aquel usuario que sí ha iniciado sesión en la aplicación y tiene acceso a todas las funcionalidades.

Casos de uso

Un caso de uso es la descripción de una interacción que el sistema debe permitir realizar al usuario o usuarios pertinentes.

Figura B.1: Diagrama general de casos de uso

CU-1	**Obtener predicción**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF2
Descripción	Cualquier usuario debe poder obtener un predicción utilizado un modelo.
Precondición	Estar en la página respectiva de la web.
Acciones	1. El usuario selecciona el archivo de vídeo sobre el que desea recibir una predicción. 2. El usuario introduce la información adicional necesaria. 3. El usuario hace clic sobre el botón para subir la infomación. 4. Se muestra una pantalla que informa al usuario sobre el estado de su petición. 5. Se muestra la predicción obtenida.
Postcondición	
Excepciones	1 No se pueden extraer fotogramas con poses suficientes. 2 La información adicional introducida no es válida.
Importancia	Alta

Tabla B.1: CU-1 Obtener predicción.

CU-2	**Iniciar sesión**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF1, RF3
Descripción	Los usuarios administradores deben poder iniciar sesión con sus credenciales.
Precondición	Estar en la página de inicio de sesión.
Acciones	1. El usuario introduce su nombre de usuario. 2. El usuario introduce su contraseña. 3. El usuario hace clic en el botón de "iniciar sesión".
Postcondición	El usuario es redirigido a la página de administración pertinente.
Excepciones	3 Las credenciales introducidas no son válidas.
Importancia	Alta

Tabla B.2: CU-2 Iniciar sesión.

CU-2	**Finalizar sesión**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF1, RF4
Descripción	Los usuarios administradores deben poder finalizar su sesión.
Precondición	Tener una sesión activa.
Acciones	1. El usuario hace clic en el botón para cerrar su sesión. 2. La sesión se cierra y el usuario es redirigido a la página principal.
Postcondición	
Excepciones	
Importancia	Media

Tabla B.3: CU-2 Finalizar sesión.

CU-4.1	**Ver lista de modelos**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF1, RF6
Descripción	Los usuarios administradores deben poder ver la lista con los modelos disponibles.
Precondición	
Acciones	1. El usuario accede a la página de gestión de modelos. 2. Se muestra una lista con los modelos existentes junto con algunas acciones.
Postcondición	
Excepciones	1 Usuario no ha iniciado sesión.
Importancia	Alta

Tabla B.4: CU-4.1 Ver lista de modelos.

CU-4.2	**Seleccionar modelo**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF1, RF9
Descripción	Los usuarios administradores deben poder seleccionar el modelo a usar para realizar predicciones.
Precondición	El usuario debe estar en la lista de modelos.
Acciones	1. El usuario hace clic en el botón para seleccionar un modelo concreto de la lista de modelos. 2. Se informa al usuario de que el modelo ha sido seleccionado y es el que se va a utilizar para realizar predicciones.
Postcondición	La base de datos cambia para establecer el modelo respectivo como seleccionado.
Excepciones	1 Usuario no ha iniciado sesión.
Importancia	Alta

Tabla B.5: CU-4.2 Seleccionar modelo.

CU-4.3	**Añadir modelo**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF1, RF7
Descripción	Los usuarios administradores deben poder añadir modelos.
Precondición	El usuario debe estar en la lista de modelos.
Acciones	1. El usuario hace clic en el botón para añadir un modelo. 2. Se abre un formulario donde se pide el archivo del modelo junto con información adicional que describa al modelo. 3. El usuario rellena el formulario y sube el archivo. 4. Se muestra una página de carga mientras se procesa la información. 5. Se devuelve al usuario a la lista de modelos, donde el nuevo modelo debería estar presente.
Postcondición	
Excepciones	1 Usuario no ha iniciado sesión. 5 Modelo subido no válido.
Importancia	Alta

Tabla B.6: CU-4.3 Añadir modelo.

CU-4.4	**Eliminar modelo**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF1, RF8
Descripción	Los usuarios administradores deben poder eliminar modelos.
Precondición	El usuario debe estar en la lista de modelos.
Acciones	1. El usuario hace clic en el botón para eliminar un modelo en concreto. 2. Se abre una ventana modal que pregunta al usuario si está seguro. 3. Si está seguro, se refresca la página mostrando la información actualizada.
Postcondición	El modelo ha sido eliminado del sistema donde estaba almacenado y su entrada en la base de datos ha sido eliminada.
Excepciones	1 Usuario no ha iniciado sesión.
Importancia	Alta

Tabla B.7: CU-4.4 Eliminar modelo.

CU-5.1	**Ver lista de usuarios**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF1, RF6
Descripción	Los usuarios administradores deben poder ver la lista con los usuarios existentes.
Precondición	
Acciones	1. El usuario accede a la página de gestión de usuarios. 2. Se muestra una lista con los usuarios existentes junto con algunas acciones que se puede realizar sobre los mismos.
Postcondición	
Excepciones	1 Usuario no ha iniciado sesión.
Importancia	Alta

Tabla B.8: CU-5.1 Ver lista de usuarios.

CU-5.2	**Añadir usuario**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF1, RF7
Descripción	Los usuarios administradores deben poder añadir nuevos usuarios al sistema.
Precondición	El usuario debe estar en la página con la lista de usuarios.
Acciones	1. El usuario hace clic en el botón para añadir un usuario. 2. Se abre un formulario donde se pide el nombre de usuario junto con su contraseña. 3. El usuario rellena el formulario y lo envía. 4. Se devuelve al usuario a la lista de usuarios, donde el nuevo usuario debería estar presente.
Postcondición	
Excepciones	1 Usuario no ha iniciado sesión. 3 Nombre de usuario demasiado corto. 3 Contraseña demasiado corta.
Importancia	Alta

Tabla B.9: CU-5.2 Añadir usuario.

CU-5.3	**Eliminar usuario**
Versión	1.0
Autor	Catalin Andrei Cacuci
Requisitos asociados	RF1, RF8
Descripción	Los usuarios administradores deben poder eliminar usuarios.
Precondición	El usuario debe estar en la lista de usuarios.
Acciones	1. El usuario hace clic en el botón para eliminar un usuario concreto. 2. Se abre una ventana modal que pregunta al usuario si está seguro. 3. Si está seguro, se refresca la página mostrando la información actualizada.
Postcondición	El usuario ha sido eliminado de la base de datos.
Excepciones	1 Usuario no ha iniciado sesión.
Importancia	Alta

Tabla B.10: CU-5.3 Eliminar usuario.

Apéndice C. Especificación de diseño

C.1. Introducción

La especificación de diseño sirve como una guía para el proceso de diseño de la aplicación, de modo que todas las personas involucradas en el proyecto saben cómo debería ser el producto final y pueden comunicarse y colaborar de forma más fácil.

En este anexo se intenta definir de forma clara el modo en que se van a almacenar los datos, las distintas interacciones que deberán existir entre las partes de la aplicación y la forma en la que el usuario final deberá interactuar con ellas.

C.2. Diseño de datos

En este apartado se explica de forma detallada la forma en la que se almacenan los datos con los que trabaja la aplicación (información de usuario, información sobre los modelos y los archivos binarios de estos modelos).

En este caso se ha optado por utilizar el sistema gestor de bases de datos PostgreSQL, esta decisión es debido a que ya existía cierta familiaridad con el programa y a que es ampliamente soportado por SQLAlchemy, la librería ORM (Object-Relational Mapping) utilizada en la API.

El diseño de datos creado es muy simple debido a que la aplicación en sí es muy simple, su objetivo principal es permitir la fácil utilización de modelos previamente entrenados para obtener predicciones y selección de los mismos por parte de los administradores.

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Figura C.1: Mockup de la página principal

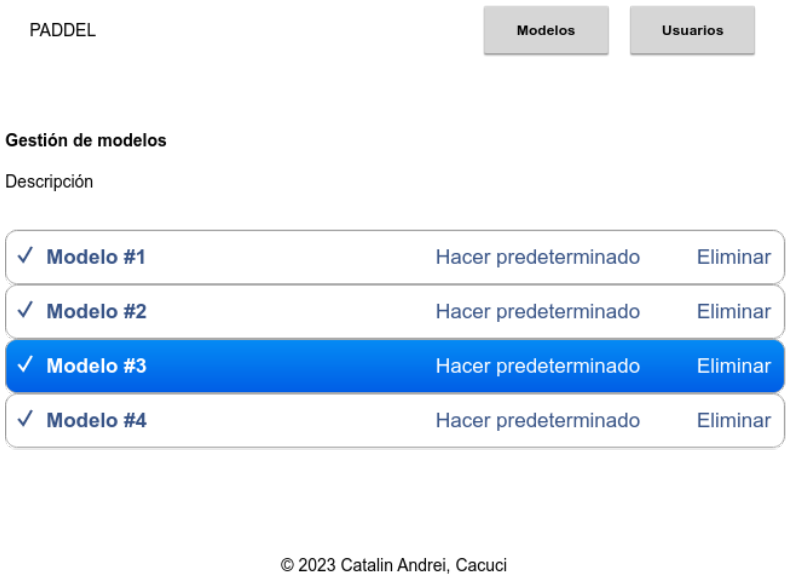

Figura C.2: Mockup de la página de gestión de modelos

Apéndice D. Documentación técnica de programación

D.1. Introducción

Esta sección contiene toda la información que una persona externa debería tener para poder trabajar con las diferentes partes de este projecto.

Existen varios archivos `Makefile` en diferentes lugares del proyecto que contienen diferentes comandos útiles que se repiten con frecuencia.

D.2. Estructura de directorios

En la raíz del proyecto se pueden encontrar los siguentes directorios:

`/app/`
: Proyecto en Docker que implementa los contenedores que conforman la aplicación web. Este directorio contiene varios subdirectorios con el código fuente y configuración de las diferentes partes de la aplicación.

  `api/`
  : Este directorio contiene la implementación de la API de la aplicación, se ha realizado en Python mediante la librería FastAPI.

  `proxy/`
  : Configuración para el contenedor de Caddy que implementa un proxy inverso que redirige las peticiones que llegan desde el exterior al contenedor apropiado (API o web).

  `web/`
  : Proyecto de SvelteKit (framework de JavaScript) que implementa el *frontend* de la aplicación.

`/docs/`
: Proyecto en LaTeX que contiene este documento junto con la memoria.

  `anexos/`
  : Contiene diferentes ficheros `.tex` que conforman estos anexos.

  `common/`
  : Archivos `.tex` comunes entre la memoria y estos anexos.

  `img/`
  : Diferentes imágenes utilizadas en la documentación.

  `memoria/`
  : Ficheros `.tex` que contienen los diferentes apartados de la memoria.

`/notebooks/`
: Notebooks de Jupyter que se han utilizado para realizar pruebas, entrenar modelos, optimizar hiperparámetros y generar gráficas.

`/paddel/`
: Librería PADDEL, es un proyecto de Python instalable mediante `pip` que contiene el código utilizado para realizar la fase de investigación (procesamiento de los vídeos, extracción de características, etc.) del que va a depender la aplicación web.

`src/paddel/`
: Contiene los diferentes archivos de Python que componen la librería PADDEL.

`hyper_parameters/`
: Contiene los archivos de Python relacionados con la fase de optimización de hiperparámetros.

`preprocessing/`
: Contiene los archivos de Python relacionados con el preprocesado y transformación de los vídeos para reducirlos a un conjunto de características.

D.3. Manual del programador

En esta sección se detalla todo lo que debería saber una persona que para realizar cambios sobre las diferentes partes que componen este proyecto.

Aplicación web

Para poder utilizar esta aplicación Docker debe ser instalado con antelación. La documentación de Docker detalla el proceso de instalación sobre diferentes plataformas. Una vez Docker está instalado el proceso siguiente debería ser agnóstico al sistema operativo utilizado.

La aplicación se compone por un conjunto de contenedores de Docker que interactúan entre ellos. En la raíz de la aplicación (`/app/`) se encuentran varios archivos *docker-compose* de tipo YAML:

- `docker-compose.yml`: Contiene la configuración básica común tanto para el entorno de desarrollo como para el de producción. Contiene información como dependencias entre contenedores, variables de entorno

que se pasa a cada contenedor y puertos en los que se va a servir la aplicación.

- `docker-compose.dev.yml`: Fichero con la configuración de los contenedores específica al entorno de desarrollo. Simplemente establece un mapeo entre los directorios del equipo anfitrión y los de los contenedores para que los cambios realizados desde el anfitrión se vean reflejados dentro de los contenedores y éstos se actualicen de forma acorde.

- `docker-compose.prod.yml`: Fichero con la configuración de los contenedores específica al entorno de producción. Simplemente establece el reinicio automático de los contenedores, para que en caso de fallo o reinicio del sistema, los contenedores se lancen junto al servicio de Docker.

Gracias al uso de Docker Compose se crea una red interna de Docker que conecta los diferentes contenedores entre sí y estableciendo *hostnames* simples que pueden utilizar para conectarse unos con otros. Por ejemplo, el proxy inverso puede realizar una petición HTTP a la API en la ruta `http://api`.

La configuración de los distintos contenedores se realiza mediante variables de entorno. Con el fin de establecer estas variables se ha creado el fichero `sample.env` que contiene unos valores básicos para estas variables que deberán ser cambiadas para adaptarse al entorno en el que se van a ejecutar los contenedores. Una vez realizados los cambios este fichero debe ser guardado con el nombre `.env` en el mismo directorio donde se encuentra `sample.env`. Este fichero `.env` es detectado de forma automática por Docker cuando se lanzan los contenedores.

Para las operaciones de arranque y parada de los contenedores se utiliza el comando `docker compose` junto con los parámetros adecuados para la operación que se desea realizar. Como estos comando no son triviales y se utilizan de forma frecuente se encuentran guardados en un fichero `Makefile` . Son posibles los siguientes comandos[1]:

- `make down`: Equivalente al comando:

[1]Para utilizar un archivo `Makefile` se necesita la utilidad `make`, en caso de que no se disponga de la misma se pueden copiar los comandos del archivo `Makefile` y utilizar manualmente.

```
docker compose down --remove-orphans --rmi all --volumes
--timeout 0
```

Elimina los contenedores y todo lo relacionado con los mismos sin esperar. No se elimina el caché que guarda Docker, por lo que si se vuelve a lanzar los contenedores no es necesario volver a descargar e instalar todo.

- `make dev`: Equivalente al comando:

```
make down &&
docker compose -f docker-compose.yml
-f docker-compose.dev.yml up -d
```

Para y elimina los contenedores si estos estaban funcionando y los lanza en modo desarrollo.

- `make prod`: Equivalente al comando:

```
make down &&
docker compose -f docker-compose.yml
-f docker-compose.prod.yml up -d
```

Para y elimina los contenedores si estos estaban funcionando y los lanza en modo producción.

Para el desarrollo de la aplicación, como es de esperar, se utiliza el comando `make dev` para arrancar los contenedores (habiendo antes instalado Docker y creado el archivo `.env`).

En modo desarrollo se pueden editar los ficheros directamente desde el sistema anfitrión y los cambios se van a ver reflejados sobre los contenedores, esto puede ser útil para realizar cambios pequeños en los que no se necesiten las ayudas que puede dar una IDE.

Para utilizar un entorno integrado para realizar el desarrollo se deben utilizar herramientas que pueden conectarse a contenedores de Docker, como, por ejemplo, Visual Studio Code con la extensión Dev Containers o IDEs de JetBrains, como IntelliJ o Pycharm, con el plugin de Docker.

En los siguientes apartados se detalla el proceso de desarrollo sobre los diferentes contenedores.

Proxy

El contenedor *proxy* contiene una instancia de Caddy funcionando como proxy inverso. Caddy es un servidor web similar a Nginx, pero con algunas características adicionales, como la gestión automática de certificados SSL.

Los archivos que utiliza este contenedor se encuentran en la carpeta `/app/proxy`:

- `Dockerfile`: Contiene la imágen y el proceso a seguir que Docker debe realizar para construir el contenedor.
- `Caddyfile`: Contiene la configuración de redirecciones de Caddy que determina el contenedor de destino para las peticiones que recibe.

Este contenedor es el único que tiene acceso al exterior, por lo que todas las peticiones que se hagan a la aplicación van a pasar por él.

API

El contenedor *api* se trata de una implementación basada en la librería FastAPI. Se puede obtener una visión general de las peticiones posibles en la ruta `/api` de la localización en la que se encuentra el servicio.

El fichero `Dockerfile` define como se prepara el contenedor, los pasos generales que sigue son:

1. Copiar (o montar) el código fuente de la API y la librería PADDEL desde el anfitrión.
2. Instalar los requisitos (`requirements.txt`), que incluyen PADDEL y las dependencias específicas de la API.
3. Se lanza el servicio de FastAPI con los parámetro adecuados dependiendo de si se lanza en modo desarrollo o producción.

Web

El contenedor *web* es un proyecto basado en NodeJS que utiliza el framework de JavaScript SvelteKit, este framework está basado en Svelte, una librería de JavaScript que permite la creación de componentes interactivos y reutilizables dentro de una web. Debido a esto, la documentación de SvelteKit es un recurso de gran valor para el desarrollo de este proyecto.

En la raíz del proyecto existen varios archivos de interés:

- `Dockerfile`: Define cómo se crea el contendor de Docker.
- `svelte.config.js`: Define algunas configuraciones para SvelteKit.
- `package.json`: Especifica detalles sobre el proyecto de NodeJS como nombre del proyecto, autor, dependencias y algunos comandos de utilidad.
- `package.lock.json`: Este archivo define las versiones específicas que se han utilizado de cada dependencia junto con sumas de comprobación para asegurar que se puede reproducir el entorno en el que se va a ejecutar la aplicación de forma exacta.
- `postcss.config.js`, `tailwind.config.js`: Archivos utilizados para la configuración de Tailwind CSS.
- `vite.config.ts`: Archivo que configura Vite, que es la herramienta utilizada para lanzar el servidor local de la página web en el entorno de desarrollo.
- `.prettierrc`: Archivo de configuración de Prettier, que es el formateador de código utilizado para esta parte de la aplicación.

Aunque los archivos anteriores son importantes, la mayoría de cambios y adiciones se realizan dentro de los directorios `/src/routes`, que define la estructura de páginas de la web, y `/src/lib`, donde se encuentran los diferentes componentes reutilizables de la web, además de algunos archivos de TypeScript (`.ts`) con funciones de utilidad y el archivo `api.ts` que implementa una interfaz que refleja las llamadas que se pueden hacer a la API implementada con FastAPI.

Además cabe destacar el directorio `/src/i18n` que contiene los idiomas soportados por la aplicación en diferentes carpetas, `en-GB` y `es-ES`, con las traducciones. La implementación realizada permite añadir nuevos idiomas simplemente creando una nueva carpeta con el código de idioma y país respectivos, por ejemplo, para añadir el idioma portugues se añadiría la carpeta `pt-PT`, se copiarían los archivos de la carpeta `es-ES` y editarían para añadir las traducciones necesarias. La página con el nuevo idioma debería aparecer en el menú de selección de idioma y ser accesible en la ruta `/pt-PT/` de la web.

Los archivos estáticos que se utilizan, imágenes, vídeos, iconos, se almacenan en el directorio `/static/`.

En el archivo `/src/app.css` se definen definen los estilos (CSS) globales de la web, el contenido de este archivo es bastante escaso debido al extenso uso de las clases que provee la librería Tailwind CSS.

Documentación

La documentación se trata de un proyecto de LaTeX compuesto por varios archivos `.tex` que se compilan a archivos `.pdf`.

Para poder compilar estos archivos se necesita tener instalada una distribución de LaTeX, en general la más recomendada es TeX Live, disponible en varias plataformas.

Una vez instalado LaTeX se debería tener acceso al comando `latexmk`, que se utiliza dentro del archivo `Makefile` tanto para compilar los archivos como para limpiar los archivos sobrantes. Están definidos los siguientes comandos:

- `make memoria`: Compila la memoria. Equivalente a ejecutar:

  ```
  latexmk -cd -pdf memoria.tex
  ```

- `make anexos`: Compila los anexos. Equivalente a ejecutar:

  ```
  latexmk -cd -pdf anexos.tex
  ```

- `make all`: Compila la memoria y los anexos. Equivalente a ejecutar los comandos anteriores.

- `make clean`: Elimina archivos auxiliares generados durante la compilación. Equivalente a ejecutar:

  ```
  latexmk -cd -pdf -bibtex-cond1 -c memoria.tex
  latexmk -cd -pdf -bibtex-cond1 -c anexos.tex
  ```

Para editar los archivos de este proyecto existen algunas alternativas, como el entorno integrado creado específicamente para LaTeX Texmaker o Visual Studio Code con la extensión Latex Workshop.

Librería PADDEL

La libraría PADDEL es una librería de Python que se puede instalar mediante el comando `pip` pasándole la ruta donde se encuentra el archivo `pyproject.toml`, por ejemplo:

```
pip install ./paddel
```

La librería contiene diferentes módulos que se utilizan para el preprocesado de los vídeos del paciente y el entrenamiento de modelos.

Configuración

Para configurar algunos parámetros del funcionamiento se ha optado por utilizar variables de entorno, gracias a la librería Pydantic las siguientes variables de entorno son leídas e interpretadas cuando el módulo sea importado por primera vez:

- `PADDEL_MIN_DETECTION_SECONDS`: Tiempo mínimo de detección en segundos para que un vídeo no se descarte debido a no tener suficientes fotogramas seguidos con poses detectadas. Por defecto es 15 segundos.
- `PADDEL_ROLLING_STD_SECONDS`: Ventana utilizada para el cálculo de la desviación estándar móvil en la extracción de algunas características, esto se utiliza para obtener un umbral y, así, saber cuando considerar o no máximos locales dentro de una secuencia de valores. Por defecto es 3 segundos.
- `PADDEL_SLOT_SIZE_SECONDS`: Para las características que se calculan como una diferencia entre la parte inicial y final de una secuencia de poses (vídeo), es el tamaño de cada una de estas partes. No puede ser mayor que `PADDEL_MIN_DETECTION_SECONDS` y establecerlo a más de $\frac{1}{2}$ de su valor puede ser problemático. Por defecto es 7 segundos.
- `PADDEL_MAX_PROCESSES`: Número máximo de procesos a utilizar en operaciones paralelizadas. Por defecto es el número de núcleos del procesador de la máquina donde se ejecute la librería.

Tipos

Esta librería utiliza extensamente el tipado estático que ofrece Python por defecto, aunque este no es impuesto en tiempo de ejecución, resulta muy útil al ser utilizado con un entorno de desarrollo integrado (IDE) ya que se pueden obtener ayudas y avisos que no se podría de otro modo.

El archivo `types.py` contine las definiciones de algunos tipos complejos que se utilizan a través de la librería debido a que son específicos a este problema concreto. Algunos de estos tipos incluyen:

- `Image`: Definido como un array de Numpy de cualquier dimensión que tenga como tipo de dato números enteros sin signo de 8 bits, lo que se corresponde con imágenes en RGB, RGBA, BGR, etc.
- `Video`: Definido como un objeto iterable de objetos de tipo `Image`, lo que es lógico, ya que un vídeo es una secuencia de imágenes.
- `Point`: Definido como una tupla con componenetes x, y y z representa un punto en un espacio de tres dimensiones, es utilizado para definir las *landmarks* que componen una pose de la mano.

Convenciones del código

En la librería se han seguido las convenciones de formato definidas por Black, que es una libería de formateado de código para Python, escrita en Python.

Se ha seleccionado Black debido a que incluye mucha opiniones predefinidas y limita en gran medida las opciones que puede controlar el usuario, esto puede parecer una desventaja, pero en este caso simplemente se deseaba una utilidad que diese un aspecto homogéneo al código con el menor esfuerzo posible por parte del programador. Además, gracias a la limitada personalización que ofrece Black, se cumple con el principio de Convención sobre Configuración [1].

Además de Black, se utilizan las herramientas autoflake, isort y mypy para eliminar sentencias `import` y variables no utilizadas, reordenar las sentencias `import` y comprobar el tipado estático utilizado respectivamente.

Como antes, el uso de estas herramientas se simplifica gracias al uso de la utilidad `make`. En este caso el archivo `Makefile` aporta los siguientes comandos:

- `make lint`:
- `make format`

Notebooks de Jupyter

Para ejecutar los notebooks de Jupyter del proyecto se debe instalar la librería Paddel, además de Jupyter en el sistema anfitrión, para esto se puede utilizar el archivo `requirements.txt` presente en la raíz del proyecto:

```
pip install -r requirements.txt
```

Tras el paso anterior se puede lanzar el kernel de IPython (backend que utiliza Jupyter) mediante en comando:

```
jupyter notebook
```

Este comando inicia un servidor web al que se puede acceder de forma local mediante cualquiera de los enlaces que se muentran por consola (acceder de forma remota resulta algo más complicado dependiendo de casos), en esta página web se muestra un explorador de archivos mediante el que se puede navegar a la localización donde se encuentran los notebooks de Jupyter (`/notebooks/`) y abrir el notebook que se desee utilizar.

Existen los siguientes notebooks:

- `00-train.ipynb`: Utilizado para entrenar una multitud de modelos, buscar los parámetros óptimos de los mismos mediante la técnica conocida como *grid search* y exportar los resultados a un fichero `.csv`.

- `01-plots.ipynb`: Utiliza los resultados obtenidos en la ejecución del notebook anterior para crear diferentes gráficas que pueden revelar información adicional sobre el comportamiento de los algoritmos utilizados con diferentes combinaciones de parámetros y conjuntos de datos de entrada.

- `02-model.ipynb`: Utilizado para entrenar un modelo final que vaya a ser utilizado en la aplicación web para realizar las predicciones. En este caso en entrenamiento se realiza con el conjunto de datos completo. Los parametros utilizados para el modelo serán los que se consideren más adecuados en base al los resultados observados en la ejecución de los notebooks anteriores.

D.4. Compilación, instalación y ejecución del proyecto

La mayor parte de este proyecto ha sido realizada Python, un lenguaje interpretado (no compilado), por lo que estas partes no necesitan ser compiladas.

API

Para lanzar la API se necesita un entorno con la versión 3.9 de Python disponible, dependiendo del sistema operativo puede ser necesario instalar la librería FFmpeg para poder realizar el procesado de vídeo al realizar las predicciones.

A continuación se deberán instalar las dependencias recogidas en el fichero `requirements.txt` mediante:

```
pip install -r requirements.txt
```

Además, la API necesita tener acceso a una base de datos PostgreSQL para almacenar de forma persistenete la información con la que trabaja. Se puede comunicar la información de acceso a dicha base de datos mediante las siguientes variables de entorno:

- `DB_USER`: Usuario a utilizar para acceder a la base de datos.
- `DB_PASSWORD`: Constraseña para el usuario definido por `DB_USER`.
- `DB_NAME`: Nombre de la base de datos a utilizar.

Para controlar la sesión de los usuarios se utilizan cadenas JWT (JSON Web Token), las cuales deben ser firmadas mediante una clave secreta que también debe ser establecida mediante una variable de entorno denominada `SECRET_KEY`. Esta variable puede ser cualquier cadena de texto pero es recomendable que sea generada mediante una herramienta de criptografía diseñada para este propósito, como el comando `openssl` presente en la mayoría de distribuciones de Linux:

```
openssl rand -hex 32
```

Con el que se obtiene una salida similar a la siguiente:

```
5e95e9e53d95d2892a63682a118a834429f43f3fd3eb951eda7811d9a1ad3f64
```

Una vez las dependencias están instaladas y las variables de entorno establecidas se puede lanzar el servidor de la API de forma local mediante:

`uvicorn app.main:app`

Gracias a utilizar FastAPI se crea automáticamente la ruta `/docs` que presenta una interfaz que muestra las diferentes acciones que permite hacer la API además de permitir ejecuar estas acciones para comprobar su correcto funcionamiento de forma bastante conveniente.

Web

Para utilizar la aplicación web se necesita un entorno que disponga de NodeJS, en concreto la versión 18 es la ideal, al ser la utilizada durante el desarrollo del proyecto, además, si nos se ha instalado con NodeJS, se debe instalar NPM también, que es el gestor de paquetes de Node.

Para instalar las dependencias del proyecto se utiliza:

`npm install` o `npm i`

Los comandos anteriores leen los archivos `package.json` y `package-lock.json` e instalan los paquetes declarados en estos.

La aplicación web ha sido realizada mediante el framework SvelteKit, y utiliza archivos `.ts` y `.svelte` que no son interpretables por un navegador, por lo que deben ser convertidos a únicamente ficheros `.html`, `.css` y `.js`. Para esto se realiza una compilación mediante el comando `npm run build` que realiza la conversión anterior además de realizar ciertas optimizaciones como eliminar código no utilizado para reducir el tamaño final de los archivos. Una vez realizada la compilación, los archivos finales se almacenan en el directorio `/build/`.

Esta aplicación web sirve como *frontend* para la API, por lo que necesita saber cómo comunicarse con la ella, para esto se utiliza la variable de entorno `PUBLIC_API_LOCATION` que podría tomar el valor `http://localhost:44444` por ejemplo.

Ejecución mediante Docker

Este proyecto está compuesto por varias piezas que interactúan entre sí, instalar todos los programas y dependencias, además de desplegar una base de datos, puede resultar una tarea ardua y tediosa, asimismo, dependiendo

del sistema sobre el que se esté intentando realizar el despliegue puede introducir inconsistencias en los procesos definidos anteriormente.

Debido a lo anterior se ha optado por utilizar la conocida herramienta Docker, gracias a esto se pueden definir diferentes contenedores con entornos controlados sobre los que se pueden realizar los pasos de instalación de forma determinista, (obteniendo siempre los mismos resultados).

El único requisito para realizar un despliegue mediante Docker es, valga la redundancia, tener Docker instalado en el sistema.

Para establecer las diferentes variables de entorno mencionadas anteriormente se utiliza un fichero de variables de entorno denominado `.env`, un ejemplo con los posibles valores de estas variables se encuentra en el fichero `sample.env`.

Para realizar el despliegue se utiliza el comando `make prod` desde la ruta `/app/`, donde sen encuentran los archivos `docker-compose.yml`. Si no se dispone de la utilidad `make` se puede utilizar:

```
docker compose -f docker-compose.yml -f docker-compose.prod.yml
up -d
```

Este comando iniciará una base de datos PostgreSQL, lanzará el servicio de la API, compilará e iniciará el servicio de la web y lanzará un proxy inverso para permitir el acceso a la API y a la web mediante distintos subdominios, por ejemplo, `https://localhost` para acceder a la web y `https://api.localhost` para acceder a la API (estas rutas son definidas en el fichero `.env`).

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

Bibliografía

[1] Wikipedia. Convención sobre configuración — wikipedia, la enciclopedia libre, 2020. [Internet; descargado 17-abril-2020].