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 — 3 de abril de 2023
Tutor: Alicia Olivares Gil

Índice general

Índice general I

Índice de figuras III

Índice de tablas IV

Apéndice A Plan de Proyecto Software 1
A.1. Introducción 1
A.2. Planificación temporal 1
A.3. Estudio de viabilidad 4

Apéndice B Especificación de Requisitos 5
B.1. Introducción 5
B.2. Objetivos generales 5
B.3. Catálogo de requisitos 5
B.4. Especificación de requisitos 5

Apéndice C Especificación de diseño 7
C.1. Introducción 7
C.2. Diseño de datos 7
C.3. Diseño procedimental 7
C.4. Diseño arquitectónico 7

Apéndice D Documentación técnica de programación 9
D.1. Introducción 9
D.2. Estructura de directorios 9
D.3. Manual del programador 11

D.4. Compilación, instalación y ejecución del proyecto 16
D.5. Pruebas del sistema 16

Apéndice E Documentación de usuario **17**
E.1. Introducción 17
E.2. Requisitos de usuarios 17
E.3. Instalación 17
E.4. Manual del usuario 17

Bibliografía **19**

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Este anexo detalla el procedimiento que se ha seguido para gestionar el progreso del proyecto y la realización de tareas.

A.2. Planificación temporal

La planificación del proyecto se llevará a cabo mediante una metodología ágil, en concreto Scrum, con el apoyo de la herramienta ZenHub para la gestión de tareas y sprints dentro de GitHub. En este caso cada sprint tiene una duración de 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.

Sprint 1 (6-2-2023 12-2-2023)

- 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)

- 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)

- 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)

- 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)

- Actualizar sección de herramientas de desarrollo.

Sprint 6 (13-3-2023 19-3-2023)

- Implementar obtimización 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)

- 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)

- 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.

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

B.2. Objetivos generales

B.3. Catálogo de requisitos

B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

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.

`/app/api/`

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

`/app/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).

`/app/web/`

Proyecto de NextJS (framework de JavaScript) que implementa el *frontend* de la aplicación.

`/docs/`

Proyecto en LaTeX que contiene este documento junto con la memoria.

`/docs/anexos/`

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

`/docs/common/`

Archivos `.tex` comunes entre la memoria y estos anexos.

`/docs/img/`

Diferentes imágenes utilizadas en la documentación.

`/docs/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.

`/paddel/src/paddel/`

Contiene los diferentes archivos de Python que componen la librería PADDEL.

`/paddel/src/paddel/hyper_parameters/`

Contiene los archivos de Python relacionados con la fase de optimización de hiperparámetros.

`/paddel/src/paddel/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:

  ```
  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 &&
  ```

[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 -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 NextJS, este framework está basado en React, una librería de JavaScript que permite la creación de componentes interactivos y reutilizables dentro de una web. Debido a esto, la documentación de NextJS 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.
- `next.config.js`: Define algunas configuraciones para NextJS.
- `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.

- `tsconfig.json`: Archivo que configura el compilador de TypeScript.
- `.eslint.json`: Archivo de configuración de ESLint.

Aunque los archivos anteriores son importantes, la mayoría de cambios y adiciones se realizan dentro de los directorios `/pages/`, que define la estructura de páginas de la web, y `/components/`, donde se encuentran los diferentes componentes reutilizables de la web.

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

En el directorio `/styles/` se alojan los archivos que definen los estilos (CSS) de la web, el contenido de esta carpeta 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

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

Bibliografía