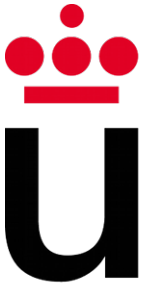

Universidad
Rey Juan Carlos

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Curso Académico 2024/2025

Trabajo Fin de Grado

# ANÁLISIS DE PATRONES DE ACTIVIDAD DE USUARIOS EN UNA PLATAFORMA WEB PARA DESARROLLO DE PROYECTOS DE ROBÓTICA

Autor : Alejandro Aguilera López
Tutor : José Felipe Ortega Soto

# Trabajo Fin de Grado

Título del Trabajo con Letras Capitales para Sustantivos y Adjetivos

**Autor/a :** Alejandro Aguilera López
**Tutor/a :** José Felipe Ortega Soto

La defensa del presente Proyecto Fin de Grado/Máster se realizó el día 3 de de 20XX, siendo calificada por el siguiente tribunal:

**Presidente:**

**Secretario:**

**Vocal:**

y habiendo obtenido la siguiente calificación:

**Calificación:**

Móstoles/Fuenlabrada, a de de 20XX

*Aquí normalmente*
*se inserta una dedicatoria corta*

# Agradecimientos

Lo primero, quiero dar las gracias a mis padres, por apoyarme incluso cuando no sabía ni qué quería estudiar, por tener paciencia conmigo y ayudarme en todo lo que fuese posible.

A mis amigos de siempre, gracias por estar ahí, por las risas de siempre y por esos planes de última hora que ayudaban a desconectar cuando la universidad se hacía cuesta arriba.

A mi amigo Daniel, que aunque ahora vive en Argentina, todo sigue como si siguiera aquí. Gracias por estar, por ayudarme y apoyarme, sin importar la distancia.

A mis compañeros de carrera, con los que he compartido gran parte de mi tiempo estos últimos cinco años. En especial a Miguel e Iván, que han pasado de ser compañeros a ser amigos de verdad.

Y por último, gracias a Felipe, mi tutor, por darme la oportunidad de hacer este trabajo, por guiarme durante todo el proceso, resolver mis dudas y estar siempre dispuesto a ayudar.

# Resumen

Este trabajo de fin de Grado tiene como objetivo principal entender mejor la interacción que se da entre los usuarios con la plataforma web Unibotics. Esta es una herramienta pensada para facilitar el aprendizaje de la robótica mediante simulaciones en entornos 3D. Gracias a esta plataforma, los estudiantes pueden crear y probar sus programas sin necesidad de configurar complejos entornos de programación, lo que hace el proceso más accesible y fomenta la experimentación.

El proyecto se ha centrado en desarrollar un sistema capaz de extraer, procesar y mostrar de forma visual los datos que genera la propia plataforma sobre la actividad de sus usuarios. Para ello, se han utilizado tecnologías como Python, Pandas, Dash, Flask y PostgreSQL, junto con contenedores Docker para facilitar su despliegue local, lo que ha hecho viable tanto analizar los datos como crear *dashboards* adaptados a diferentes patrones de uso.

El análisis se ha realizado a partir de la base de datos de Unibotics, lo que ha permitido obtener información relevante, como el tiempo que los estudiantes dedican a cada ejercicio, su ubicación geográfica o posibles diferencias de comportamiento según el género. Los *dashboards* están diseñados para que los docentes puedan, de forma sencilla, evaluar el progreso de sus alumnos y detectar las dificultades que puedan tener los estudiantes durante el aprendizaje.

Con este trabajo se ha querido dar un paso más para que Unibotics no sea solo una plataforma donde aprender robótica, sino también una herramienta que entienda mejor a cada estudiante. La idea es aprovechar los datos que se generan en el día a día para ofrecer una enseñanza más personalizada, eficiente y adaptada a las necesidades reales de quienes la utilizan.

# Summary

This final degree project aims to better understand how users interact with the Unibotics web platform. Unibotics is designed to make learning robotics easier through 3D simulations. With this platform, students can create and test their programs without setting up complex programming environments, making the learning process more accessible and encouraging experimentation.

The project focuses on building a system that can extract, process, and visually display the data generated by the platform about user activity. To do this, technologies like Python, Pandas, Dash, Flask, and PostgreSQL have been used, along with Docker containers to simplify local deployment. This setup made it possible to analyze the data and create *dashboards* adapted to different usage patterns.

The analysis was based on data from the Unibotics database, making it possible to gather useful insights such as the time students spend on each exercise, their geographic location, and behavior differences based on gender. The *dashboards* are designed to help teachers easily track student progress and identify common learning difficulties.

The goal of this project is to help Unibotics become more than just a platform for learning robotics. The idea is to use the data generated by students to offer a more personalized, efficient, and adaptive learning experience that better meets their real needs.

# Índice general

# Índice de figuras

# Índice de fragmentos de código

# Capítulo 1

# Introducción

Debido al avance de las tecnologías, la robótica se encuentra en constante evolución, lo que hace que cada vez la podamos ver en más sectores, desde el mundo laboral hasta en nuestro día a día. Sin embargo, el aprendizaje de la robótica es un terreno complejo, ya que para poder empezar en este campo se requiere la instalación y configuración de distintas herramientas y entornos de programación, los cuales pueden resultar bastante enrevesados, en especial a nuevos usuarios que quieren empezar en el mundo de la robótica.

Por ejemplo, la instalación de entornos como ROS2 y Gazebo [10] puede convertirse en un verdadero desafío para quienes empiezan. En ROS2 hay que asimilar de golpe conceptos como nodos y tópicos, lidiar con dependencias y versiones al instalar paquetes, y acostumbrarse a gestionar todo mediante línea de comandos. En Gazebo, por su parte, definir mundos de simulación implica escribir ficheros SDF o URDF desde cero, ajustar físicas y colisiones sin apenas ayuda gráfica y estar pendiente de la compatibilidad de los plugins. Estas tareas, lejos de facilitar el aprendizaje por prueba y error, pueden dispersar la atención del usuario y convertir sus primeros pasos en un laberinto de configuración y depuración.

Otro gran problema del aprendizaje de la robótica es el coste de un robot en el que probar nuestros códigos, ya que, a diferencia de otras materias, la robótica es un campo cuyo aprendizaje se basa en la prueba y el error.

Este trabajo de fin de grado gira en torno a Unibotics [12], una plataforma web que nace como solución a estos problemas. Unibotics permite a los usuarios acceder a ejercicios interactivos de robótica, en los cuales podrán programar robots y simularlos en escenarios 3D sin la necesidad de tener el robot de forma física y sin tener que instalar ningún tipo de programa.

Sin embargo, a pesar de sus ventajas, aún existen áreas de Unibotics que podrían me-

jorarse. Plataformas de enseñanza online como Moodle [2] llevan años aprovechando los datos de interacción de sus usuarios, como el acceso a recursos, la participación en foros o el tiempo dedicado a cada unidad, para generar informes que ayudan a los docentes a identificar dónde se quedan atascados los alumnos y adaptar el contenido en consecuencia.

En Unibotics sucede algo similar: todos los códigos de programación, tiempos de actividad y de resolución de ejercicios ya se recogen en PostgreSQL, pero por ahora esos registros solo se almacenan, sin extraerles ningún valor añadido. Si aplicáramos a Unibotics las mismas técnicas de analítica de comportamiento que utiliza Moodle, podríamos conocer en detalle cómo afrontan los retos los estudiantes, detectar automáticamente los puntos de bloqueo y ofrecer pistas o ejercicios de refuerzo personalizados. De este modo, la plataforma dejaría de ser solo un simulador interactivo y se convertiría en una herramienta inteligente capaz de guiar el aprendizaje de la robótica de forma dinámica y adaptada a cada usuario.

Dada esta situación, este proyecto quiere proporcionar una herramienta que permita extraer y procesar estos datos. Además, se diseñarán unos *dashboards* interactivos, donde se podrá ver de forma clara y detallada toda la información obtenida de la base de datos.

Con la implementación de estos *dashboards* , se busca que los profesores puedan ver de forma rápida y sencilla el trabajo realizado por sus alumnos, lo cual podrá facilitarles la enseñanza de la robótica a sus alumnos.

## 1.1 Objetivos

### 1.1.1 Objetivo general

Este Trabajo de Fin de Grado consiste en crear una herramienta que permita extraer, analizar y visualizar los registros que contienen información sobre la actividad y el comportamiento de los usuarios en la plataforma Unibotics.

### 1.1.2 Objetivos específicos

Dado el objetivo general, se presentan los siguientes objetivos específicos:

- **Extracción de datos:** Recopilar y estructurar la información relevante de la base de datos de Unibotics, asegurando su calidad y consistencia para su análisis posterior.

- **Análisis de estadísticas básicas del comportamiento de los usuarios:** Calcular métricas como el número de sesiones, el tiempo de uso por ejercicio y la frecuencia de acceso, con el fin de identificar patrones generales en la actividad de los usuarios.

- **Representación gráfica de la información:** Diseñar visualizaciones claras (gráficos, tablas y *dashboards*) que faciliten la interpretación rápida y precisa de los datos analizados sobre el comportamiento de los usuarios.

## 1.2 Planificación

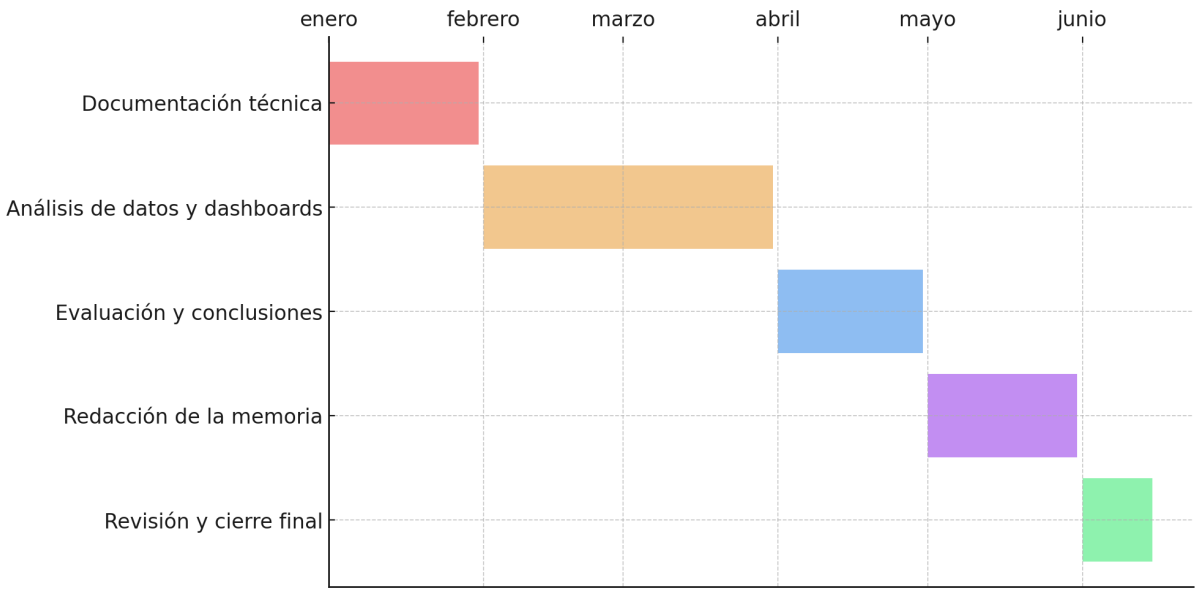


Figura 1.1: Planificación temporal del proyecto.

En la Figura 1.1 se presenta un diagrama de Gantt que ilustra la planificación temporal del proyecto. Este diagrama de Gantt se divide en cinco fases principales:

- **Documentación técnica**: En esta etapa se realiza el estudio del material necesario para abordar adecuadamente el proyecto.

- **Análisis de datos y *dashboards*** : Análisis de los datos existentes en la base de datos, selección de los más relevantes y construcción de *dashboards* que los representen visualmente.

- **Evaluación y conclusiones**: Incluye la revisión de los *dashboards* desarrollados y la elaboración de conclusiones basadas en los resultados obtenidos.

- **Redacción de la memoria**: Escritura formal de la memoria del proyecto, incluyendo su desarrollo y las conclusiones.
- **Revisión y cierre final**: Corrección de errores, ajustes menores y una revisión global del trabajo antes de su entrega final.

## 1.3 Estructura de la memoria

Por último, en esta sección se presenta a alto nivel la organización del resto del documento y los contenidos que se abordarán en cada capítulo.

- En el primer capítulo se hace una breve introducción al proyecto, se describen los objetivos del mismo y se refleja la planificación temporal.
- En el siguiente capítulo se describen las tecnologías utilizadas en el desarrollo de este trabajo de fin de grado (capítulo 2).
- En el capítulo 3 se describe la arquitectura de la plataforma Unibotics.
- En el capítulo 4 se presentan los *dashboards* y se realiza un análisis de la información que se extrae de ellos.
- Por último, se presentan las conclusiones del proyecto, así como los trabajos futuros que podrían derivarse de este (capítulo 5).

# Capítulo 2

# Tecnologías

En este capítulo se describen las tecnologías utilizadas para el desarrollo de este proyecto.

## 2.1 Python

Python es un lenguaje de programación de alto nivel, interpretado, dinámico y fuertemente tipado. Es un lenguaje multiparadigma, lo que significa que permite desarrollar software utilizando distintos enfoques, como la programación orientada a objetos, funcional e imperativa [1].

El principal motivo por el cual se eligió Python como lenguaje de programación para este trabajo es que la plataforma Unibotics está desarrollada en este lenguaje, por lo que utilizar otro resultaría ineficiente y supondría una complicación innecesaria. Además, Python destaca por su simplicidad y legibilidad, lo que facilita el desarrollo y mantenimiento del código; su amplia comunidad, que ofrece soporte constante; y la gran variedad de bibliotecas en las que se apoya este proyecto. A continuación, se enumeran y describen dichas bibliotecas.

### 2.1.1 Pandas

Pandas es una de las herramientas más poderosas para la manipulación y el análisis de datos en Python. Esta biblioteca fue diseñada para que la limpieza, transformación y análisis de datos sean rápidos y eficientes.

Una de las razones por las que se eligió Pandas para este proyecto es su capacidad para manipular datos de manera flexible y eficiente. Sus estructuras principales, *Series* y *DataFrame*, facilitan la organización y el acceso a la información de forma intuitiva. La estructura *Series* actúa como un array unidimensional con etiquetas asociadas, mientras que el *DataFrame* es una tabla bidimensional que permite realizar operaciones similares a las que se pueden llevar a cabo en bases de datos o herramientas como Excel.

En el contexto de este proyecto, Pandas resulta una opción adecuada, ya que la plataforma sobre la que se desarrolla está basada en Python y requiere una gestión eficiente de datos tabulares. Además, como se menciona en [5], Pandas no solo facilita la manipulación de datos, sino que también permite centrarse en la interpretación y visualización de la información, en lugar de invertir tiempo en tareas repetitivas de procesamiento.

### 2.1.2 Dash

Dash es una herramienta de Python que permite crear aplicaciones web interactivas de forma sencilla [11, 7].

En este proyecto, se utilizó Dash porque facilita la creación de visualizaciones interactivas y atractivas para representar los datos obtenidos de la plataforma Unibotics. Gracias a esta herramienta, es posible generar interfaces gráficas en las que los usuarios pueden interactuar con los datos de forma intuitiva, mediante filtros, menús desplegables u otros controles dinámicos. Esto resulta especialmente útil para representar información compleja de manera clara y accesible.

Además, Dash se integra fácilmente con otras bibliotecas del ecosistema Python, lo que permite construir visualizaciones dinámicas de diversos tipos, como gráficos de barras, líneas o incluso mapas interactivos.

### 2.1.3 Psycopg2

Psycopg2 es una librería de Python que permite establecer conexiones con bases de datos PostgreSQL. Es ampliamente utilizada por su eficiencia, estabilidad y facilidad de uso para la ejecución de consultas SQL desde Python [8].

En este proyecto, se utilizó Psycopg2 para realizar operaciones como la extracción, inserción y modificación de datos directamente desde scripts en Python. Esta biblioteca ofrece una interfaz sencilla para gestionar conexiones con la base de datos y ejecutar comandos SQL de forma segura, ayudando a prevenir vulnerabilidades comunes, como la inyección de SQL.

Además, su integración con bibliotecas como Pandas, como se explicó en la subsección 2.1.1, permite cargar los resultados de las consultas directamente en estructuras de datos tabulares, lo que facilita su análisis y posterior representación visual mediante Dash (véase subsección 2.1.2).

## 2.2 Flask

Flask es un framework web ligero y flexible para Python que permite desarrollar aplicaciones web de forma rápida y sencilla. Es de código abierto y cuenta con una amplia comunidad de usuarios y desarrolladores que contribuyen activamente a su evolución. Flask es especialmente popular por su simplicidad, lo que lo hace adecuado tanto para aplicaciones pequeñas como para proyectos de mayor envergadura [4].

En este proyecto, se utilizó Flask para crear una aplicación web que actúa como entorno de prueba, permitiendo integrar y visualizar los *dashboards* antes de su incorporación definitiva a la plataforma. Esta implementación facilitó la validación del funcionamiento del sistema en un entorno controlado.

## 2.3 Docker

Docker es una plataforma de virtualización ligera basada en contenedores que permite empaquetar una aplicación junto con todas sus dependencias en una única unidad portátil. Cada contenedor comparte el núcleo del sistema operativo, pero se ejecuta de forma aislada, lo que garantiza que el entorno interno sea siempre el mismo, independientemente de la máquina en la que se despliegue [6].

En este proyecto, se utilizó Docker principalmente para desplegar la base de datos PostgreSQL de Unibotics en un contenedor independiente. Esta estrategia permite iniciar, detener o reiniciar la base de datos mediante un único comando, sin necesidad de instalar PostgreSQL directamente en el sistema anfitrión ni preocuparse por posibles conflictos con otras versiones. Además, el uso de contenedores asegura que el entorno de ejecución sea idéntico tanto en entornos locales como en los de pruebas o producción, lo que facilita la reproducibilidad y la portabilidad del sistema.

## 2.4 Entorno de desarrollo: PyCharm

PyCharm es un Integrated Development Enviroment (Entorno de Desarrollo Integrado) (IDE) dedicado específicamente a la programación en Python, desarrollado por la compañía checa JetBrains.

Proporciona análisis de código, un depurador gráfico, una consola de Python integrada, control de versiones y soporte para desarrollo web con Django. Todas estas características lo convierten en un entorno completo e intuitivo, idóneo para el desarrollo de proyectos académicos como el presente. En la figura 2.1 se muestra la estructura del proyecto dentro de PyCharm, donde puede observarse la organización de la carpeta `app`.

Dentro de esta estructura, la carpeta `templates` contiene las diferentes páginas HTML del sistema. El archivo `index.html` representa la pantalla inicial, donde se visualizan varios gráficos interactivos a modo de resumen. Al hacer clic sobre cualquiera de estos gráficos, el usuario es redirigido a páginas más específicas de análisis, cada una representada por un archivo distinto ubicado en la carpeta `pages`, que contiene las diferentes *dashboards* del sistema.

El archivo `documentacion.html` corresponde a una sección dedicada a mostrar el documento PDF del Trabajo de Fin de Grado, permitiendo al usuario visualizar la memoria del proyecto directamente desde la interfaz web.

La carpeta `static` almacena los recursos estáticos del sitio web, como hojas de estilo CSS, imágenes y, en este caso, el archivo PDF con el TFG.

El archivo `app.py` constituye el núcleo de la aplicación desarrollada con Flask. Su función principal es gestionar las rutas de la aplicación, indicando al servidor qué recurso debe mostrarse al usuario según la dirección solicitada. Por ejemplo, cuando se accede a la página principal (`/`), `app.py` se encarga de renderizar el archivo `index.html`. Además, este archivo conecta cada sección visual del proyecto con su respectiva lógica de funcionamiento, como las funciones definidas en dashboard.py, responsables de generar los gráficos o los datos que se muestran.

Por último, el apartado `External Libraries` incluye las librerías y dependencias externas requeridas para el correcto funcionamiento del proyecto, como Flask, Pandas y otras utilizadas durante el desarrollo.

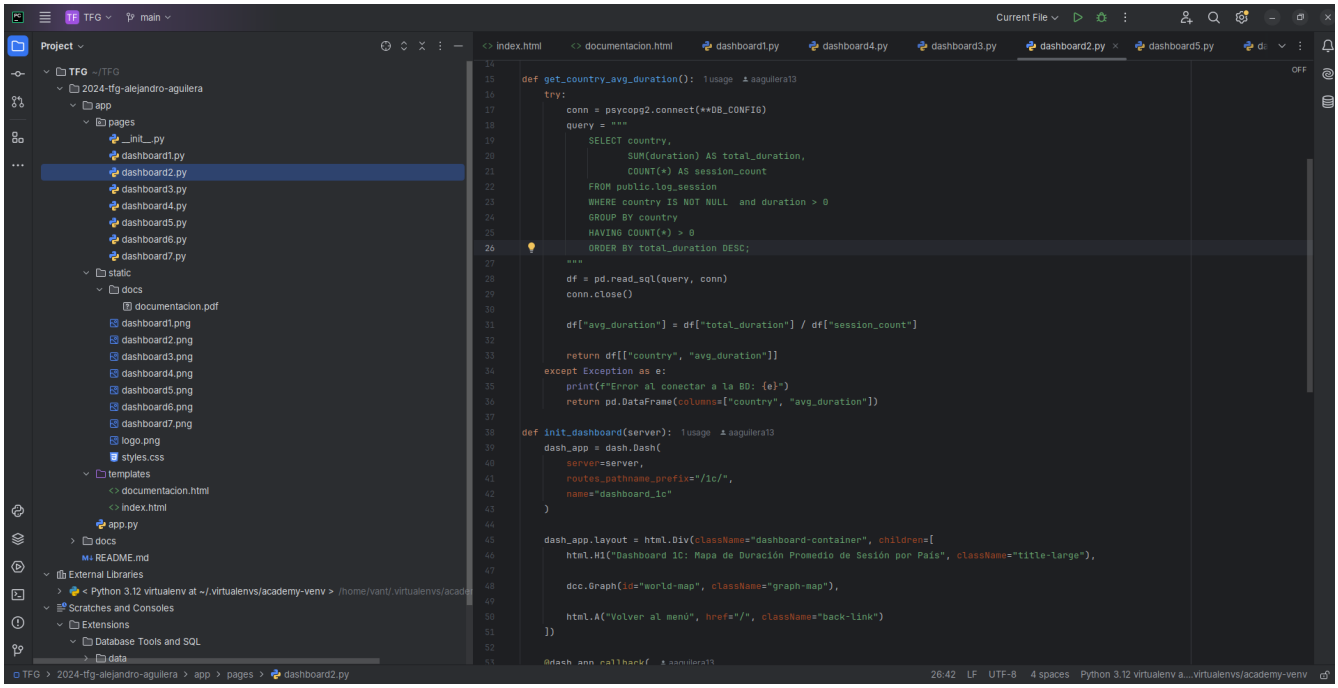


Figura 2.1: Estructura del proyecto en PyCharm

## 2.5 Entorno de desarrollo: Visual Studio Code

Visual Studio Code es un IDE ligero y altamente extensible, desarrollado por Microsoft y disponible de forma gratuita bajo licencia MIT. Gracias a su amplio ecosistema de extensiones, entre las que destaca la extensión oficial de Python, ofrece resaltado de sintaxis, completado inteligente (IntelliSense), análisis estático (linting), formateo automático y refactorización de código. Dispone, además, de un depurador integrado, una terminal embebida y soporte nativo para control de versiones con Git. Su interfaz modular, configuraciones basadas en JSON y catálogo de extensiones lo convierten en un entorno flexible y adaptable a proyectos académicos y profesionales en Python y otros lenguajes de programación.

En la figura 2.2 se muestra una captura de Visual Studio Code con el proyecto abierto. En ella puede observarse la estructura de carpetas y archivos, así como la disposición del código y el entorno de trabajo utilizado durante el desarrollo del proyecto.

## 2.6 Plataformas de ejecución

Este proyecto se ha probado en entornos locales, tanto en Windows como en Linux. En el sistema operativo Windows se utilizó Visual Studio Code (sección 2.5) como entorno de desarrollo, mientras que en Linux se empleó PyCharm (sección 2.4).

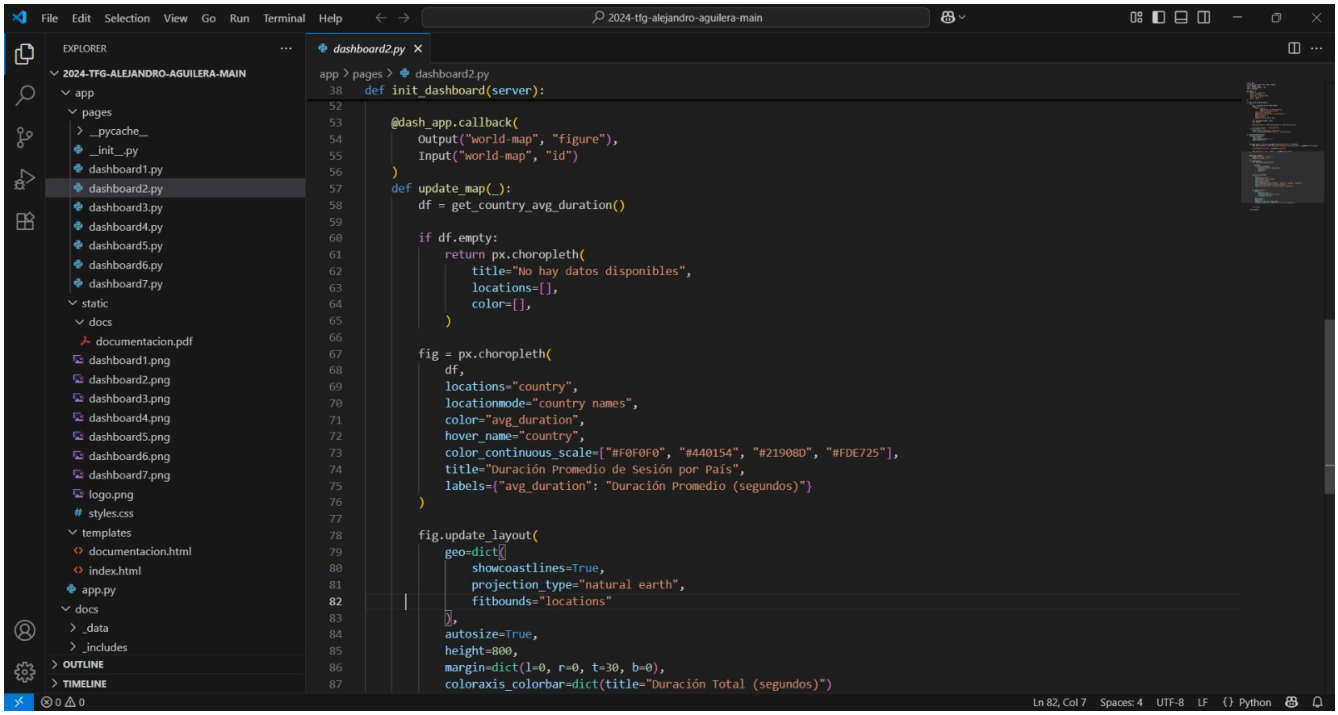


Figura 2.2: Estructura del proyecto en Visual Studio Code

Asimismo, los *dashboards* desarrollados han sido verificados en ambos sistemas operativos, garantizando su correcto funcionamiento y compatibilidad. De este modo, se asegura que tanto Unibotics en local como los *dashboards* implementados operan sin incidencias en las principales plataformas de ejecución.

## 2.7 Redacción de la memoria: LaTeX/Overleaf

LaTeX es un sistema de composición tipográfica de alta calidad que incluye características especialmente diseñadas para la producción de documentación técnica y científica. Estas características, entre las que se encuentran la posibilidad de incluir expresiones matemáticas, fragmentos de código, tablas y referencias, junto con el hecho de que se distribuya como software libre, han hecho que LaTeX se convierta en el estándar de facto para la redacción y publicación de artículos académicos, tesis y todo tipo de documentos científico-técnicos.

Por su parte, Overleaf es un editor LaTeX colaborativo basado en la nube. Lanzado originalmente en 2012, fue creado por dos matemáticos que se inspiraron en su propia experiencia en el ámbito académico para crear una solución satisfactoria para la escritura científica colaborativa.

Además de por su perfil colaborativo, Overleaf destaca porque, pese a que en LaTeX el

escritor utiliza texto plano en lugar de texto formateado (como ocurre en otros procesadores de texto como Microsoft Word, LibreOffice Writer y Apple Pages), éste puede visualizar en todo momento y paralelamente el texto formateado que resulta de la escritura del código fuente.

# Capítulo 3

# Arquitectura

En este capítulo se describe la arquitectura general de la plataforma Unibotics y sus componentes, así como la estructura de la base de datos PostgreSQL y los distintos tipos de datos que gestiona. A continuación, se expone cómo se ha realizado el despliegue local en el entorno `D1`, incluyendo la configuración de contenedores Docker y la puesta en marcha de los servicios asociados.

Seguidamente, se detalla el proceso de construcción de los *dashboards* de análisis, desde la extracción y tratamiento de los datos hasta la generación de gráficas interactivas. Por último, se explica el desarrollo de un script destinado a estimar el género de los usuarios a partir de sus nombres, así como su integración en la base de datos del sistema.

## 3.1 Arquitectura general

### 3.1.1 Entornos de despliegue: D1, D2 y D3

La plataforma Unibotics cuenta con tres entornos diferenciados de despliegue: **D1**, **D2** y **D3**, cada uno con un objetivo y una configuración específica que permiten cubrir las distintas fases del ciclo de desarrollo, pruebas y producción.

- **D1 (Despliegue local):** Es el entorno utilizado para el desarrollo local por parte de los programadores. Permite ejecutar una instancia completa de Unibotics en una máquina personal, incluyendo la base de datos, servidor web y frontend, así como contenedores de backend para la simulación robótica. Este entorno está pensado para el trabajo individual y pruebas en desarrollo, sin afectar al resto de usuarios ni a los datos de producción.

- **D2 (Entorno de test):** Es un entorno de pruebas intermedio desplegado en una máquina del laboratorio de la URJC. Su función principal es validar los cambios introducidos antes de pasar a producción. Permite probar nuevas funcionalidades de forma controlada, ya que está conectado a una versión persistente de la base de datos con datos *dummy*. En él se puede evaluar también el funcionamiento de granjas remotas y comunicaciones reales entre componentes.

- **D3 (Entorno de producción):** Es el entorno real en el que opera Unibotics para los usuarios finales. Está desplegado en un servidor de Amazon Web Services (AWS), con configuración de alta disponibilidad, certificados HTTPS y granjas de ejecución distribuidas en diferentes laboratorios de la universidad. Toda la actividad de los usuarios y ejercicios queda registrada en este entorno.

En este proyecto, se ha trabajado exclusivamente con el **entorno D1**, ya que permite una mayor autonomía para realizar pruebas, desarrollos y análisis sin comprometer los entornos compartidos de test o producción. Las tareas de despliegue local y configuración del entorno han seguido las directrices específicas del entorno D1, detalladas en la sección 3.2.

### 3.1.2 Arquitectura de Unibotics

Como se explicó anteriormente, Unibotics es una plataforma web diseñada con el fin de facilitar el aprendizaje práctico de la robótica, proporcionando a los estudiantes un entorno en el que pueden encontrar ejercicios y escenarios interactivos sin la necesidad de instalar o configurar entornos complejos de software.

Esta plataforma proporciona diferentes herramientas para trabajar con robots. La herramienta que gestiona el software robótico requerido es RADI (Robotics Academy Docker Image). Se trata de unos contenedores Docker especiales en los que se llevan a cabo la ejecución de ejercicios y las simulaciones robóticas.

Estos contenedores integran ROS2 y Gazebo, además de otras dependencias necesarias para poder ejecutar el código de los usuarios. ROS2 (Robot Operating System 2) ofrece un middleware estándar para la programación robótica, permitiendo comunicar diferentes nodos y controlar así el flujo de datos entre los distintos sensores. Gazebo aporta un entorno de simulación física realista que proporciona entornos 3D donde se puede simular el comportamiento de los robots con el código previamente programado por el usuario.

Para comunicar el contenedor RADI con el navegador, se utiliza otra herramienta llamada RAM (Robot Application Manager), que actúa como puente entre el navegador y el contenedor. De esta forma, cuando el usuario modifica o ejecuta código, RAM lo recibe, lo ejecuta en el contenedor y devuelve los resultados correspondientes. Estos resultados pue-

den ser desde imágenes del entorno y la posición del robot hasta cualquier dato relevante para el ejercicio.

La figura 3.1 ilustra esta arquitectura, mostrando la relación entre los distintos componentes de la plataforma, desde la interacción del usuario a través del navegador hasta la gestión de simulaciones y datos en los servidores.

Para almacenar toda esta cantidad de datos que ofrece cada simulación, además de otros datos de Unibotics como los usuarios, datos estadísticos de ejercicios y datos del desempeño del usuario, se emplea una base de datos relacional del tipo PostgreSQL, lo que facilita la gestión y el análisis futuro de la información, la cual se obtiene a través de una API para poder representarla en los *dashboards*.

Respecto a la parte del usuario, la interacción con esta plataforma se realiza a través de un navegador web, mediante el cual el usuario accede a una interfaz construida con React, donde se encuentran los ejercicios y escenarios interactivos.

Para implementar todas estas herramientas, Unibotics se apoya en Django, un *framework* de alto nivel orientado al desarrollo web en Python, especialmente a la creación de aplicaciones web complejas. Gracias a Django, es posible estructurar de forma clara la lógica interna de la plataforma, lo que permite relacionar usuarios con ejercicios y almacenar información sobre las sesiones de trabajo.

Además de Django, emplea Nginx y Gunicorn, herramientas que mejoran la eficiencia de la plataforma. Estas permiten gestionar un gran volumen de usuarios de manera simultánea, asegurando que el sistema opere sin interrupciones.

La base de datos de Unibotics está organizada para almacenar toda la información necesaria para el funcionamiento de la plataforma, desde los datos de los usuarios hasta los resultados de las simulaciones. Aunque toda esta información se encuentra en una única base de datos, está organizada en diferentes tablas que cubren varias áreas clave de la plataforma.

Por ejemplo, la información relacionada con los ejercicios y los universos de simulación está almacenada en tablas como `exercises` y `exercise_universes`, las cuales permiten asociar ejercicios a los distintos entornos de simulación disponibles. Esto asegura que un mismo ejercicio pueda ejecutarse en varios universos, proporcionando flexibilidad en las simulaciones. Actualmente, existen 15 tipos diferentes de ejercicios activos en la plataforma.

También está la información sobre los robots y los mundos de simulación, que se guarda en tablas como `robots` y `worlds`. Estas tablas contienen detalles sobre la configuración de los robots y los mundos de simulación, lo que permite a la plataforma gestionar los recursos necesarios para ejecutar los ejercicios en el entorno correcto.

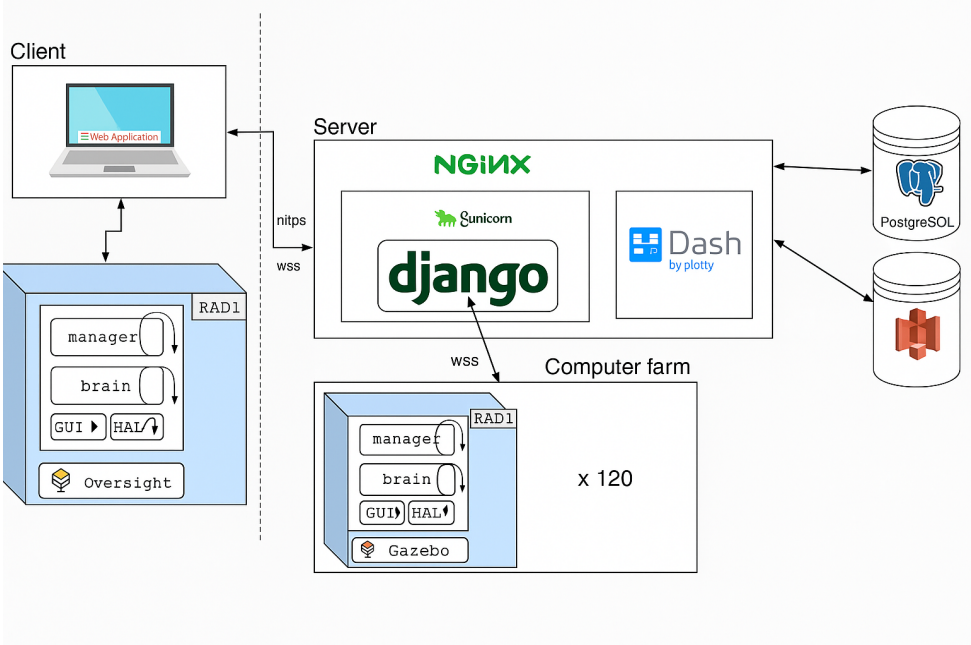


Figura 3.1: Estructura de Unibotics.

Por otro lado, las tablas de logs, como `log_session` y `log_exercises`, almacenan los registros más importantes sobre la actividad de los usuarios. Actualmente, existen 30.270 logs de sesión y 115.254 logs de ejercicios. Estos logs incluyen información detallada sobre la duración de las sesiones de los usuarios, los ejercicios realizados y las fechas en las que se llevaron a cabo. Además, los registros contienen información sobre la duración total en segundos, el navegador desde el cual los usuarios iniciaron sesión y el país desde el cual se conectaron. Los registros de sesión cubren un periodo que va desde 2021 hasta febrero de 2025, lo que proporciona un amplio conjunto de datos para el análisis.

Además, la base de datos gestiona también los datos clave sobre los 1.382 usuarios únicos, los permisos de acceso y las máquinas de granja (contenedores Docker) que ejecutan los ejercicios. Aquí se guardan los detalles de los usuarios y cómo se relacionan con los ejercicios a los que tienen acceso, así como el estado de las máquinas que se utilizan para ejecutar los ejercicios.

En la figura 3.2 se muestra el diagrama de entidad-relación (ER) que ilustra cómo todas estas tablas están conectadas entre sí y cómo se organiza la información en la base de datos. Este diagrama es esencial para comprender cómo fluye la información dentro del sistema y cómo se gestionan las relaciones entre los diferentes elementos.

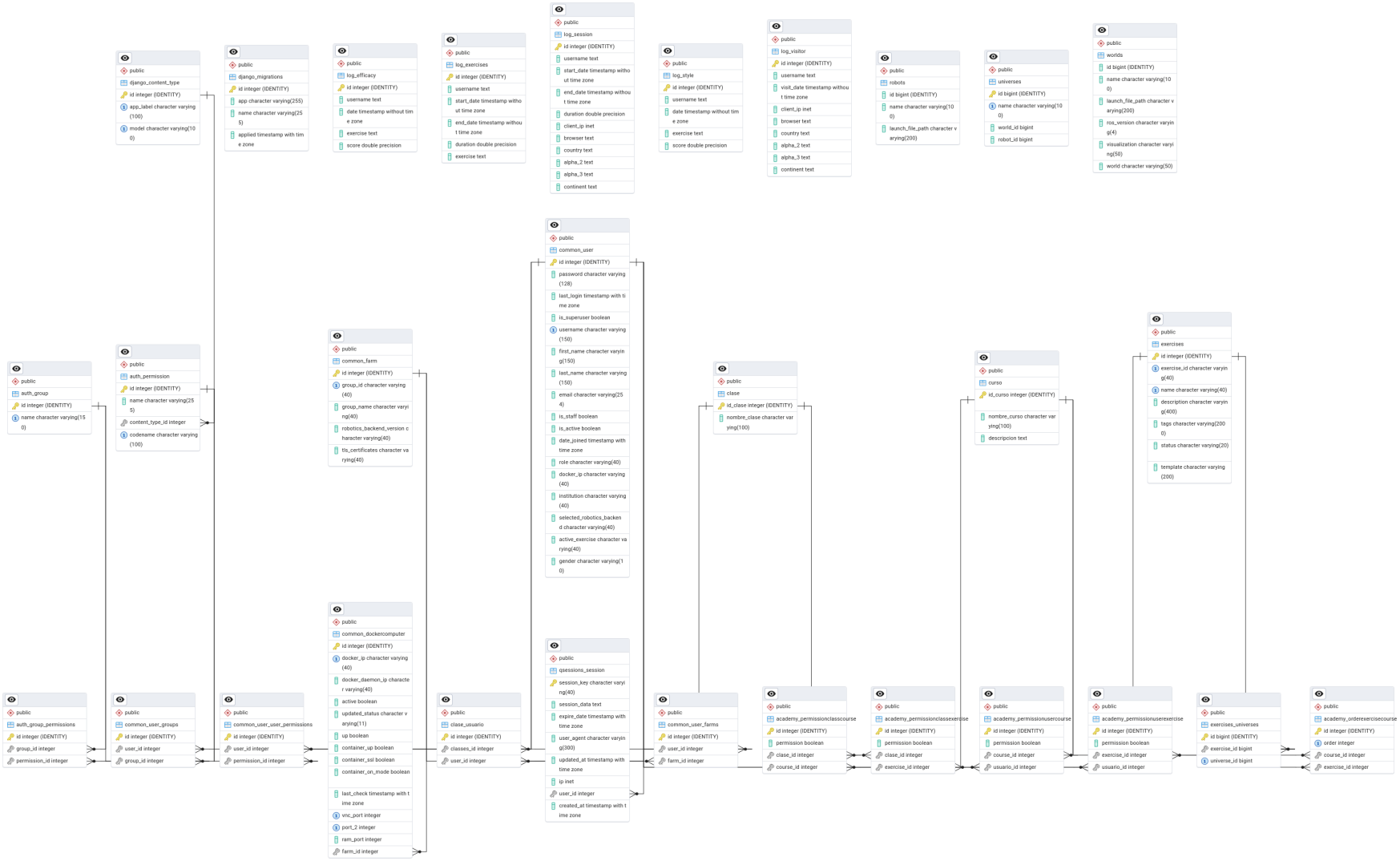


Figura 3.2: Diagrama entidad-relación.

Finalmente, en el apéndice A (sección A), se incluyen ejemplos de consultas SQL que permiten obtener información directamente de las tablas de logs y realizar análisis sobre la actividad de los usuarios y su interacción con los ejercicios.

## 3.2 Despliegue local en Linux

Para poder probar y depurar todos los componentes de Unibotics de forma sencilla, se instaló un entorno completo en una máquina Linux (**entorno D1**). En primer lugar, se creó un entorno virtual de Python 3.8 para aislar las dependencias del proyecto sin afectar al sistema. A continuación, se clonó el repositorio de `Unibotics-webserver` junto con sus submódulos y se ejecutó la instalación de los paquetes necesarios desde el fichero `utils/requirements.txt`.

Dado que Unibotics utiliza PostgreSQL como base de datos, se levantó un contenedor Docker con esa imagen. Esto permitió iniciar, detener o reiniciar la base de datos con total comodidad y sin necesidad de instalar nada en el sistema. Tras arrancar el contenedor, se aplicaron los *dumps* de Robotics Infrastructure y Robotics Academy para cargar los universos y ejercicios. Posteriormente, se ejecutaron las migraciones de Django para crear el resto de tablas y, finalmente, se importaron datos de prueba de cursos pasados para disponer de una muestra amplia sobre la que trabajar.

En la parte del *frontend*, se configuraron los enlaces simbólicos que Webpack necesita para localizar los componentes de `RoboticsAcademy`, se instalaron las dependencias de Node y se arrancó el modo desarrollo de Webpack. Con ello, cualquier cambio en el código React se recompila al vuelo y se refleja en el navegador.

Con este entorno es posible levantar la plataforma completa en local, probar cada cambio y comprobar al instante cómo funciona, sin riesgo de afectar al entorno de producción. Esto proporciona una zona de pruebas segura donde experimentar libremente y acelerar el ciclo de desarrollo y depuración.

## 3.3 Construcción de gráficas

En esta sección se explica cómo se ha llevado a cabo el proceso de creación de los distintos *dashboards* utilizados en este trabajo. El objetivo es mostrar paso a paso cómo se ha ido construyendo cada gráfica, desde las consultas a la base de datos hasta su visualización final en una aplicación web.

Aunque estos *dashboards* están pensados para integrarse en la plataforma Unibotics, por

el momento no se han podido incorporar en el entorno de preproducción debido a que es necesario resolver algunos errores que afectan a la integración de gráficas desarrolladas con Dash, tecnología utilizada por la plataforma. Por este motivo, durante el desarrollo se optó por utilizar Flask (sección 2.2) como entorno alternativo, lo que permitió probar y visualizar las gráficas sin depender del entorno principal.

Para desarrollar estos *dashboards*, el proceso de construcción sigue una estructura común. En primer lugar, se extraen los datos necesarios desde la base de datos PostgreSQL mediante consultas SQL adaptadas a cada caso. Estas consultas se lanzan desde Python utilizando la biblioteca `Psycopg2` (sección 2.1.3), lo que permite traer los datos directamente al entorno de trabajo.

A modo de ejemplo, en el código 3.1 se puede ver una consulta SQL que obtiene la duración total de sesiones por país.

```sql
SELECT country,
SUM(duration) AS total_duration, COUNT(*) AS session_count
FROM public.log_session
WHERE country IS NOT NULL AND duration > 0
GROUP BY country
HAVING COUNT(*) > 0
ORDER BY total_duration DESC;
```

Código 3.1: Consulta SQL para obtener duración total y número de sesiones por país.

Una vez obtenidos los datos, estos se procesan utilizando la biblioteca `Pandas` (sección 2.1.1). Esto permite dejar los datos preparados antes de mostrarlos en una gráfica, facilitando operaciones como agrupación, ordenación o filtrado, para que la visualización sea clara y organizada.

Como se muestra en el código 3.2, este procesamiento puede manejar los datos de diferentes maneras.

Una vez agrupados y ordenados los datos, la parte más relevante del proceso ha sido el diseño de las visualizaciones usando `Dash` (sección 2.1.2), que permitió construir *dashboards* interactivos adaptados a los distintos análisis planteados. A lo largo del desarrollo, se aprovecharon las múltiples opciones que ofrece la biblioteca Dash para mostrar los datos con distintos tipos de gráficos (mapas, líneas, barras, histogramas, *boxplot*), eligiendo en cada caso la representación más adecuada según la naturaleza de los datos [3].

Cada *dashboard* está compuesto por una serie de componentes visuales que se combinan para construir la interfaz interactiva. Estos componentes se definen utilizando elementos de tipo HTML (como títulos, contenedores o enlaces) y controles dinámicos proporcionados por Dash, como `dcc.Input` (para introducir texto) o `dcc.Dropdown` (para seleccionar

```
# Cálculo de duración promedio por país
df["avg_duration"] = df["total_duration"] / df["session_count"]

# Eliminación de valores nulos
df = df.dropna(subset=["avg_duration"])

# Conversión de nombre de países a formato categórico ordenado
df["country"] = pd.Categorical(df["country"], categories=df["country"].unique(),
↪  ordered=True)

# Orden descendente por duración promedio
df = df.sort_values("avg_duration", ascending=False)

# Formateo a dos decimales
df["avg_duration"] = df["avg_duration"].round(2)
```

Código 3.2: Ejemplo de procesamiento de datos con Pandas.

opciones). Estos elementos permiten al usuario interactuar con la aplicación. Por ejemplo, en el *dashboard* de la sección 4.1, el código 3.3 permite escribir un nombre de usuario para filtrar los datos.

```
dcc.Input(
    id="username-input",
    type="text",
    placeholder="Introduce el nombre de usuario...",
    debounce=True,
    className="input-box"
)
```

Código 3.3: Ejemplo de componente interactivo con dcc.Input.

Todos estos elementos se organizan dentro del *layout*, que es el bloque central donde se define la estructura del *dashboard*. Este *layout* actúa como un esquema visual en el que se indica qué elementos aparecen, en qué orden y cómo están distribuidos en la página.

Una vez definidos los elementos visuales, se programan los *callbacks*, que son funciones que reaccionan a los cambios en los controles de la interfaz. Por ejemplo, si el usuario introduce un nombre de usuario o selecciona un ejercicio, el *callback* recoge ese valor y genera automáticamente la gráfica correspondiente. Esto es lo que permite que los *dashboards* sean interactivos y se actualicen al instante sin necesidad de recargar la página.

En el código 3.4 se muestra un ejemplo de un *callback* sencillo que actualiza una gráfica a partir de un valor introducido.

Durante la creación de los *dashboards* también se tuvo en cuenta el diseño visual, ase-

```
@dash_app.callback(
    Output("exercise-duration-graph", "figure"),
    Input("username-input", "value")
)
def update_graph(username):
    return fig
```

Código 3.4: Ejemplo de callback que actualiza una gráfica.

gurando que las gráficas fueran claras y fáciles de leer. Para lograrlo, se ajustaron manualmente varios elementos, como los colores, el tamaño de los puntos o líneas, los títulos de los ejes, el orden de las categorías e incluso la altura total del gráfico. Estas configuraciones se realizan directamente en el código de Dash, utilizando funciones como `update_layout()` o parámetros de estilo específicos para cada tipo de gráfico.

En el código 3.5 se puede ver cómo se personaliza un gráfico ajustando la estética general.

```
fig.update_layout(
    title="Duración Total por Ejercicio",
    xaxis_title="Duración Total (segundos)",
    yaxis_title="Ejercicio",
    plot_bgcolor="white",
    height=800,
    margin=dict(l=0, r=0, t=30, b=0)
)
```

Código 3.5: Ejemplo de configuración visual con update_layout().

En algunas ocasiones, se incorporaron detalles adicionales como líneas auxiliares, leyendas o etiquetas para facilitar la interpretación de los datos. Cuando los valores variaban considerablemente entre los usuarios, se aplicó una escala logarítmica en el eje correspondiente para facilitar la visualización.

Como ejemplo, el código 3.6 muestra cómo se añaden leyendas y formas al gráfico.

Una vez desarrollados todos los *dashboards* de manera individual, se integraron en una aplicación web utilizando el microframework Flask (sección 2.2). Esta aplicación actúa como contenedor principal y permite al usuario navegar fácilmente entre los distintos *dashboards* desde una interfaz centralizada y visualmente clara. Además del diseño de cada gráfica, se implementó una hoja de estilos CSS que determina la apariencia general de la aplicación.

En la figura 3.3 se muestra la página principal de la aplicación, donde se presenta una

```
fig.add_trace(go.Scatter(
    x=df["total_duration"],
    y=df["exercise"],
    mode="markers",
    marker=dict(size=12, color="red"),
    name="Duración"
))
```

Código 3.6: Ejemplo de elementos adicionales como leyendas y líneas.

galería con miniaturas de cada uno de los *dashboards* disponibles. Al hacer clic sobre cualquiera de ellas, el usuario accede directamente al *dashboard* correspondiente, cargado de forma dinámica dentro del entorno Flask.

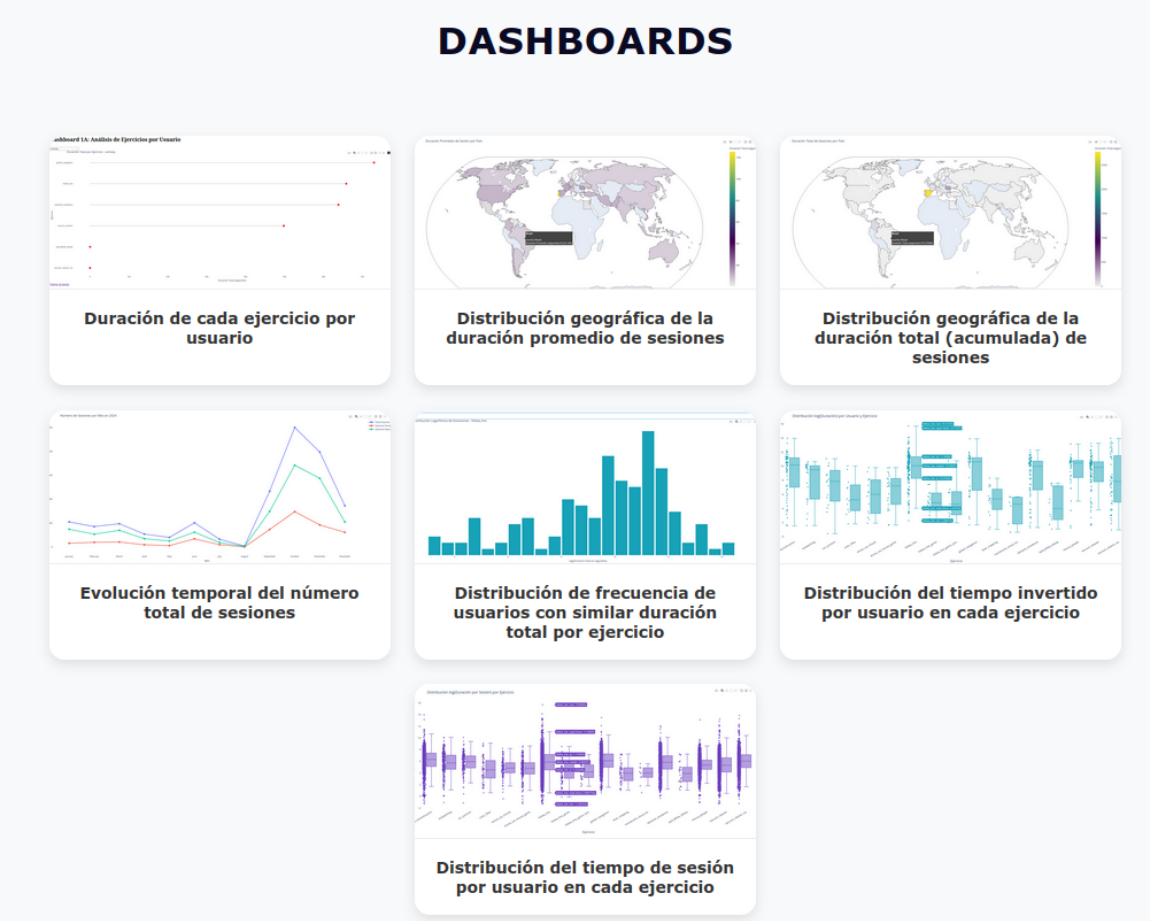


Figura 3.3: Menú principal de la aplicación.

La estructura principal de la aplicación está definida en el archivo `app.py`, donde se van incorporando todos los *dashboards* desarrollados. Cada uno está organizado por separado en su propio archivo dentro de la carpeta `pages`, lo que permite mantener el proyecto limpio y fácil de gestionar. Para añadirlos a la aplicación principal, simplemente se utiliza una

función llamada `init_dashboard`, que los integra automáticamente.

En el código 3.7 se puede ver un ejemplo de cómo está estructurado este archivo.

```
from flask import Flask, render_template
import pages.dashboard1 as page_1a
import pages.dashboard2 as page_1c
...

app = Flask(__name__)
app = page_1a.init_dashboard(app)
app = page_1c.init_dashboard(app)
...

@app.route("/")
def index():
    return render_template("index.html", active_page="dashboard")

@app.route("/documentacion")
def documentacion():
    return render_template("documentacion.html", active_page="documentacion")

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=5000)
```

Código 3.7: Código del contenido de app.py.

La página principal de la aplicación está definida en el archivo `index.html` y presenta una página donde se muestran todos los *dashboards* disponibles. Cada uno aparece como una imagen con su título, y al hacer clic sobre ella, el usuario accede directamente al *dashboard* correspondiente.

El código 3.8 es un ejemplo de un fragmento del código de `index.html`, donde se representa una imagen con su título y con su referencia a la página correspondiente.

```
<a href="/1a/" class="dashboard-card">
    <img src="{{ url_for('static', filename='../static/dashboard1.png') }}"
    ↪  alt="Dashboard 1A">
    <span>Dashboard 1A</span>
</a>
```

Código 3.8: Ejemplo de *dashboard* en index.html.

Por último, todo el diseño visual de la aplicación está definido en un archivo CSS propio ubicado en la carpeta `static`. Este archivo se encarga de dar estilo a la página de *dashboards*, controlando aspectos como los colores, tamaños, márgenes o fuentes.

## 3.4 Discriminación de género de usuarios

La base de datos actual de Unibotics dispone de multitud de variables, como se explicó en la sección 3.1.2; sin embargo, no existe un registro que clasifique el género de los usuarios. Este aspecto no fue considerado en el diseño inicial del sistema, pero se plantea su incorporación en futuras etapas para que los nuevos usuarios puedan proporcionar dicha información de manera voluntaria.

Esta modernización, además de ser útil para realizar un análisis más preciso del comportamiento, aporta una metodología más inclusiva que refuerza la imagen de Unibotics.

Gracias a estos ajustes, es posible identificar y corregir con mayor facilidad desigualdades entre hombres y mujeres, lo que permite analizar las diferencias y necesidades de cada grupo sin incurrir en sesgos ni estereotipos.

Por otro lado, contar con esta información ayudaría a evitar errores de interpretación como la "paradoja de Simpson" [13], un fenómeno estadístico que ocurre cuando los datos agrupados por una categoría, en este caso, el género, alteran la relación entre otras variables, lo que puede llevar a conclusiones erróneas si no se examina el contexto completo de los datos.

Para disponer de la variable género, se desarrolló un *script* que utiliza una aproximación automática para inferir el género de los usuarios a partir de su primer nombre, lo que permitió completar el análisis en un formato provisional. Este método no es perfecto y presenta limitaciones, ya que se basa en patrones de nombres que no siempre reflejan con precisión la identidad de género de los usuarios; sin embargo, resulta útil al proporcionar una muestra suficientemente amplia para trabajar.

El script 3.9 utiliza las siguientes bibliotecas:

- `psycopg2` (sección 2.1.3): para conectarse a la base de datos PostgreSQL y realizar consultas SQL.
- `pandas` (sección 2.1.1): para manejar y procesar los datos extraídos de la base de datos.
- `gender_guesser` [9]: permite estimar el género basado en el nombre de una persona, usando un detector preentrenado.

El script comienza conectándose a la base de datos PostgreSQL mediante la biblioteca `psycopg2`. Luego, crea una nueva columna llamada `gender` en la tabla `common_user`, aunque este paso se omite si la columna ya existe. A continuación, se extraen los usuarios

```python
import psycopg2
import pandas as pd
import gender_guesser.detector as gender

conn = psycopg2.connect(**DB_CONFIG)

with conn.cursor() as cur:
    cur.execute("""
        DO $$
        BEGIN
            IF NOT EXISTS (
                SELECT 1 FROM information_schema.columns
                WHERE table_name='common_user' AND column_name='gender'
            ) THEN
                ALTER TABLE common_user ADD COLUMN gender VARCHAR(10);
            END IF;
        END $$;
    """)
    conn.commit()

df = pd.read_sql("""
    SELECT id, first_name FROM common_user
    WHERE gender IS NULL OR gender = 'unknown';
""", conn, dtype={"first_name": str})

detector = gender.Detector()

def estimate_gender(name):
    if not name or not name.strip():
        return 'unknown'
    first = name.strip().split()[0].capitalize()  # Corrige mayúsculas
    g = detector.get_gender(first)
    if g in ['male', 'mostly_male']:
        return 'M'
    elif g in ['female', 'mostly_female']:
        return 'F'
    else:
        return 'unknown'

df['gender'] = df['first_name'].apply(estimate_gender)

cursor = conn.cursor()
for _, row in df.iterrows():
    cursor.execute(
        "UPDATE common_user SET gender = %s WHERE id = %s;",
        (row['gender'], row['id'])
    )
```

Código 3.9: Script para estimar el género de los usuarios.

cuyo género aún no está definido, es decir, aquellos con el valor `NULL` o `'unknown'` en dicho campo.

Utilizando la biblioteca `gender_guesser`, el script estima el género de los usuarios basándose en su primer nombre. Además, se corrigen posibles errores relacionados con mayúsculas o minúsculas, capitalizando adecuadamente el primer nombre para evitar fallos en la detección. Finalmente, se actualiza la base de datos con los géneros estimados para estos usuarios.

De este modo, la variable género queda disponible para futuros análisis, lo que permite incorporar esta dimensión en los estudios de comportamiento de los usuarios.

# Capítulo 4

# Experimentos y validación

En esta sección expondré los distintos *dashboards* que he desarrollado con el objetivo de poder visualizar y analizar la información que está almacenada en la base de datos de Unibotics.

Cada dashboard se centra en distintas características del comportamiento del usuario. Para su desarrollo he utilizado el framework Dash de Python junto con la biblioteca Plotly para la representación gráfica, y la información se extrae directamente desde la base de datos PostgreSQL mediante consultas SQL específicas.

Antes de entrar en detalle con cada uno de los dashboards, conviene dar una idea general sobre el volumen y el tipo de datos con los que se ha trabajado. La base de datos de Unibotics contiene una gran cantidad de información que cubre distintos aspectos del uso de la plataforma. En la Tabla 4.1 se resumen algunas de las cifras más relevantes, que permiten contextualizar el alcance del análisis realizado:

Tabla 4.1: Volumen total de datos analizados.

| **Elemento** | **Cantidad / Descripción** |
|---|---|
| Usuarios únicos registrados | 1.382 usuarios registrados en la plataforma. |
| Ejercicios disponibles | 15 ejercicios activos disponibles para los usuarios. |
| Registros de sesiones | 30.270 registros en la tabla `log_session`, que recogen información sobre la duración, navegador y país de origen. |
| Registros de ejercicios | 115.254 registros en la tabla `log_exercises`, que incluyen detalles de interacción de los usuarios con los ejercicios. |
| Periodo cubierto por los datos | Desde 2021 hasta febrero de 2025. |

Toda esta información está distribuida en distintas tablas dentro de la base de datos. La Tabla 4.2 recoge las tablas principales que he utilizado para construir los *dashboards*, junto

con una breve descripción de cada una de ellas[1].

Tabla 4.2: Tablas SQL utilizadas de la base de datos de Unibotics.

| Tabla | Descripción |
|---|---|
| `log_session` | Contiene información sobre las sesiones iniciadas por los usuarios. Entre los campos más relevantes se encuentran el nombre del usuario, la fecha y hora de inicio de la sesión, la duración total en segundos y el país desde el cual se conecta el usuario. |
| `log_exercises` | Contiene información sobre los accesos de los usuarios a los distintos ejercicios disponibles. Entre los campos más relevantes se encuentran el nombre del usuario, el nombre del ejercicio al que accede, la duración de la interacción, la fecha y hora de inicio, así como otros datos relacionados con la actividad. |
| `common_user` | Reúne información general sobre los usuarios registrados en la plataforma. Aunque cuenta con múltiples campos, el más relevante para este trabajo es el género del usuario. |
| `exercises` | Contiene un listado con los distintos ejercicios disponibles en la plataforma. |

En los siguientes apartados expondré de forma individual cada uno de los *dashboards* desarrollados, explicando su propósito, los datos que representa, el motivo de la visualización elegida y las principales conclusiones que se pueden extraer sobre el comportamiento de los usuarios.

## 4.1 Duración de cada ejercicio por usuario

Este *dashboard* permite analizar la duración total que un usuario ha dedicado a cada ejercicio de la plataforma.

Para obtener la información de duración por ejercicio, el *dashboard* se conecta directamente a la base de datos PostgreSQL de Unibotics mencionada anteriormente. Una vez establecida la conexión, la aplicación Dash ejecuta una consulta SQL parametrizada con el nombre de usuario introducido en el campo de entrada.

La consulta SQL 4.10 aprovecha la funcionalidad de agrupación y agregación de SQL: se filtran los registros por el usuario seleccionado y luego se agrupan por el identificador del ejercicio, calculando la suma de todos los tiempos asociados. La información se obtiene de la tabla `log_session`.

[1]La base de datos de Unibotics contiene más tablas, como se explicó en la sección 3.1.2, pero aquí se

```
SELECT exercise, SUM(duration) AS total_duration
FROM public.log_exercises
WHERE username = %s
GROUP BY exercise
HAVING SUM(duration) > 0
ORDER BY total_duration DESC;
```

Código 4.10: Consulta SQL para obtener la duración total por ejercicio de un usuario.

De este modo, la propia consulta realiza el cálculo de la duración total (en segundos) acumulada para cada ejercicio realizado por el usuario indicado. La cláusula `GROUP BY` agrupa las filas que tienen el mismo valor de ejercicio y aplica la función agregada `SUM()` para sumar los datos de cada grupo. En otras palabras, si el usuario ha realizado varias sesiones o intentos en un mismo ejercicio, todos esos tiempos se suman en un único resultado por ejercicio. Esta consulta retorna una tabla con dos columnas: el nombre de cada ejercicio y la suma de la duración total que el usuario ha invertido en él. Dichos resultados se cargan en un `DataFrame` de `pandas` para su manipulación en la aplicación y posteriormente se emplean para generar la visualización.

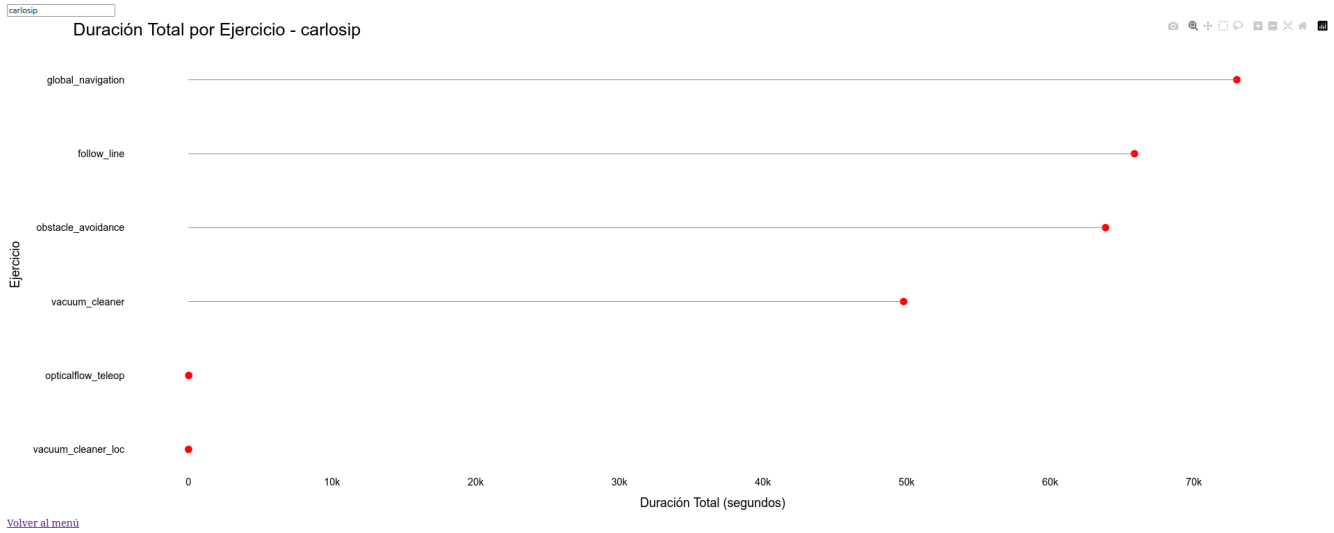


Figura 4.1: Análisis de la duración de cada uno de los ejercicios por usuario. En este ejemplo, la figura representa los datos registrados para el usuario con identificador carlosip.

Desde el punto de vista del usuario, el sistema muestra un *dashboard* y un cuadro de texto (figura 4.1) donde se puede introducir el nombre de usuario cuya duración total por ejercicio se desea consultar. Una vez introducido, se carga automáticamente la información correspondiente a ese usuario.

presentan solo las relevantes para este análisis.

El *dashboard* 4.1 presenta los datos mediante un gráfico horizontal de líneas con puntos, también conocido como gráfico tipo *Lollipop*. Este tipo de visualización es una variación del diagrama de barras, donde la barra tradicional se transforma en una línea delgada y el valor se resalta con un marcador circular al final.

Se ha elegido un gráfico del tipo *Lollipop* en el que cada ejercicio aparece en función del tiempo total que el usuario le ha dedicado, ya que de esta forma es muy fácil identificar de un vistazo en qué actividades ha pasado más tiempo. Al ordenar los ejercicios de mayor a menor duración, se prioriza la información relevante del comportamiento del usuario, evitando un orden alfabético que podría ocultar patrones útiles. Así, por ejemplo, puede observarse que el ejercicio `global_navigation` ocupa la mayor parte de la sesión, mientras que `opticalflow_teleop` apenas tiene unos minutos, sin necesidad de analizar cada número individualmente. Este enfoque facilita la comparación entre tareas con duraciones muy distintas, evitando confusiones o búsquedas innecesarias.

Gracias a ello, pueden extraerse varias conclusiones:

- **Identificación de ejercicios más y menos trabajados:** Las longitudes de las líneas revelan de un vistazo cuáles son los ejercicios en los que el usuario ha invertido más tiempo y cuáles menos.
- **Detección de patrones de esfuerzo o dificultad:** Una duración total muy elevada en un ejercicio puede indicar que el usuario ha tenido dificultades, ha necesitado múltiples intentos prolongados o que se trata de una tarea extensa. Por el contrario, duraciones muy bajas podrían sugerir que el ejercicio se completó rápidamente o incluso fue abandonado. En ambos casos, los valores extremos de la distribución temporal pueden señalar ejercicios que requieren atención específica al evaluar el progreso.
- **Distribución del tiempo de aprendizaje:** El conjunto de duraciones permite observar cómo el usuario ha distribuido su tiempo en la plataforma. Un perfil equilibrado mostraría tiempos similares entre ejercicios, mientras que un perfil más desequilibrado tendría unos pocos ejercicios concentrando la mayor parte del tiempo total.

## 4.2 Distribución geográfica de la duración promedio de sesiones

Este *dashboard* tiene como objetivo mostrar la duración promedio de las sesiones de los usuarios de Unibotics, agrupadas por país. A través de un mapa mundial interactivo, se representa gráficamente qué países presentan una mayor o menor media de tiempo por sesión, permitiendo identificar patrones geográficos en el uso de la plataforma.

La información utilizada en este *dashboard* se obtiene desde la tabla `log_session`. La consulta SQL 4.11 agrupa los datos por país, calcula la suma total de duración y el número total de sesiones por país, y posteriormente se calcula en Python la duración promedio dividiendo ambos valores.

```sql
SELECT country,
SUM(duration) AS total_duration, COUNT(*) AS session_count
FROM public.log_session
WHERE country IS NOT NULL AND duration > 0
GROUP BY country
HAVING COUNT(*) > 0
ORDER BY total_duration DESC;
```

Código 4.11: Consulta SQL para obtener duración total y número de sesiones por país.

A partir de la consulta 4.11 se genera un `DataFrame` de `pandas` con el país y su respectiva duración promedio de sesión, que luego se utiliza para crear el gráfico 4.2. Este cálculo permite representar una métrica más equilibrada que la duración total, y proporciona una mejor medida de cómo interactúan los usuarios con la plataforma en promedio.

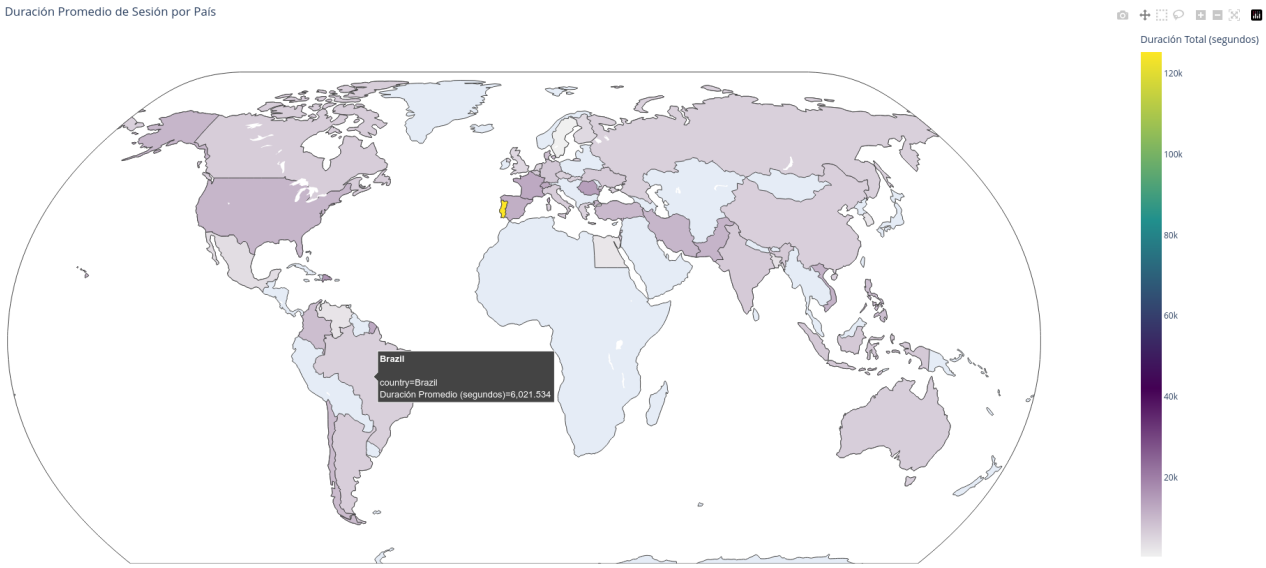


Figura 4.2: Mapa que representa la distribución geográfica (por país) de la duración promedio de las sesiones.

El *dashboard* 4.2 utiliza un mapa coroplético, generado mediante Plotly Express, en el que cada país se colorea en función de la duración promedio de sesión. Cuanto mayor es la duración, más intenso es el color asignado, siguiendo una escala cromática progresiva desde tonos claros hasta colores más saturados. En el ejemplo mostrado se emplea una paleta de colores continua personalizada para resaltar visualmente los valores extremos.

Se ha elegido un mapa coroplético porque este tipo de visualización permite mostrar de forma clara cómo se distribuye la duración promedio de las sesiones a lo largo del mundo. Al asignar un color a cada país según su valor de duración, se puede identificar al instante qué regiones concentran más actividad y cuáles menos, sin necesidad de consultar tablas o listas. El degradado de colores refuerza la percepción de diferencias entre países y facilita la identificación de patrones globales.

Del análisis del mapa 4.2, se pueden extraer las siguientes conclusiones:

- **Alta duración promedio en España y Portugal:** Esto indica que, en países como España y Portugal, las sesiones tienden a ser más largas, lo que refleja un uso más intensivo de la plataforma. No significa necesariamente que haya más sesiones en esos países, sino que, en promedio, los usuarios pasan más tiempo en cada sesión en comparación con otros lugares, donde el uso puede ser más breve o intermitente.
- **Bajas duraciones promedio en ciertos países:** Estos casos podrían indicar una adopción más reciente, menor familiaridad con la plataforma o simplemente menos disponibilidad de tiempo por parte de los usuarios.
- **Utilidad para la administración de Unibotics:** El *dashboard* permite identificar mercados con mayor impacto y uso sostenido, facilitando decisiones sobre soporte, localización de contenidos y planes de expansión. También señala regiones donde el uso es bajo, lo que puede inspirar estrategias específicas de difusión, colaboración educativa o mejoras en el acceso a la plataforma.

## 4.3 Distribución geográfica de la duración total (acumulada) de sesiones

Este *dashboard* tiene como objetivo mostrar la duración total acumulada de todas las sesiones de usuarios de Unibotics, agrupadas por país. A través de un mapa interactivo, permite observar en qué regiones geográficas se ha registrado un mayor tiempo de uso en conjunto, proporcionando una visión global de la distribución de la actividad en la plataforma.

La información utilizada en este *dashboard* se obtiene desde la tabla `log_session`. La consulta SQL 4.12 que se utiliza agrupa los datos por país y calcula la suma total de la duración de todas las sesiones correspondientes a cada país.

A diferencia de otros *dashboards* que analizan promedios, en este caso se calcula simplemente la acumulación del tiempo total de uso de la plataforma por país. Esto permite identificar los países donde el uso absoluto de Unibotics ha sido más intenso.

```sql
SELECT country, SUM(duration) as total_duration
FROM public.log_session
WHERE country IS NOT NULL
GROUP BY country
HAVING SUM(duration) > 0
ORDER BY total_duration DESC;
```

Código 4.12: Consulta SQL para obtener la duración total por país en sesiones.

Una vez obtenidos los resultados, se cargan en un `DataFrame` de `pandas` con el país y su respectiva duración total de sesiones, y se utilizan como base para la representación gráfica en el mapa.

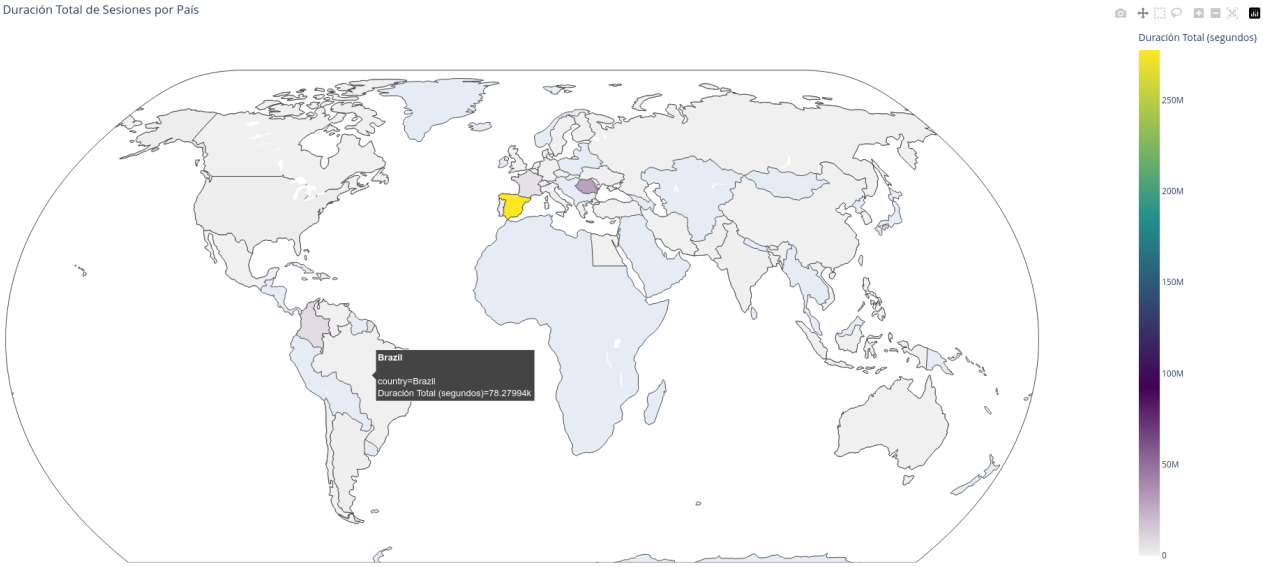


Figura 4.3: Mapa que representa la distribución geográfica (por país) de la duración total (acumulada) de todas las sesiones registradas en cada región.

El *dashboard* 4.3 utiliza un mapa coroplético (*choropleth map*), generado mediante Plotly Express, en el que cada país se colorea en función de la duración total de sesión. Cuanto mayor es la duración, más intenso es el color asignado, siguiendo una escala cromática progresiva desde tonos claros hasta colores más saturados. En el ejemplo mostrado, se emplea una paleta de colores continua personalizada para resaltar visualmente los valores extremos.

Se ha elegido un mapa coroplético porque, al igual que en el caso anterior 4.2, permite ver de un vistazo cómo se distribuye la duración total de sesiones por país: cada país se colorea según su valor, mostrando de forma inmediata qué regiones concentran más actividad y cuáles menos, sin necesidad de consultar tablas o listas.

Del análisis del mapa 4.3, se obtienen las siguientes conclusiones:

- **España destaca con el mayor volumen de tiempo total de sesiones:** Esto refleja su fuerte implantación y uso intensivo de la plataforma, probablemente por ser el país de origen o principal mercado de Unibotics. Otros países europeos y latinoamericanos presentan niveles más bajos, lo que puede interpretarse como una menor penetración o un uso más reciente de la plataforma en esas regiones.

- **Diferencias geográficas claras:** Algunos continentes como África o Asia presentan, en general, un uso más bajo, salvo excepciones, lo que podría señalar oportunidades de expansión o barreras de adopción actuales.

- **Utilidad para la gestión de Unibotics:** Este *dashboard* permite identificar en qué países la plataforma tiene mayor presencia, lo que facilita la toma de decisiones estratégicas relacionadas con campañas, traducción de contenidos o inversiones para mejorar el servicio. También ayuda a detectar regiones con potencial para nuevas colaboraciones educativas. Por ejemplo, se observa un alto número de sesiones en Rumanía, lo cual podría ser una oportunidad para contactar con el equipo de gestión de Unibotics en ese país, entender cómo están trabajando allí y comparar el rendimiento de los usuarios con el de España. Este análisis comparativo puede proporcionar información valiosa para implementar mejoras en ambas regiones.

## 4.4 Evolución temporal del número total de sesiones

Este *dashboard* muestra la evolución mensual del número de sesiones iniciadas en Unibotics a lo largo del año. Para ello, incluye un desplegable que actualiza el *dashboard* en función del año seleccionado, diferenciando también entre usuarios masculinos y femeninos. El objetivo es analizar cómo varía el uso de la plataforma mes a mes y si existen diferencias significativas por género.

Los datos provienen de las tablas `log_session` y `common_user`. La consulta SQL 4.13 realiza una agregación mensual, teniendo en cuenta: el total de sesiones iniciadas en cada mes, el número de sesiones realizadas por usuarias (género femenino) y el número de sesiones realizadas por usuarios (género masculino).

Una vez extraídos, los datos se procesan en `pandas` para asegurarse de que los meses estén correctamente ordenados de enero a diciembre, y se formatea el nombre de los meses.

```sql
SELECT
DATE_TRUNC('month', s.start_date) AS month,
COUNT(*) AS total_sessions,
COUNT(CASE WHEN u.gender = 'F' THEN 1 END) AS female_sessions,
COUNT(CASE WHEN u.gender = 'M' THEN 1 END) AS male_sessions
FROM public.log_session s
JOIN public.common_user u ON s.username = u.username
WHERE EXTRACT(YEAR FROM s.start_date) = %s
GROUP BY month
ORDER BY month;
```

Código 4.13: Consulta SQL para obtener sesiones mensuales totales y por género en 2024.


Figura 4.4: Evolución temporal del total de sesiones registradas en la plataforma cada mes. El ejemplo muestra la evolución del número total mensual de sesiones registradas en 2024.

El *dashboard* 4.4 utiliza un gráfico de líneas para representar la evolución de las sesiones durante el año seleccionado. Se incluyen tres líneas:

- Total de sesiones (en azul).
- Sesiones femeninas (en rojo).
- Sesiones masculinas (en verde).

Cada punto en el gráfico representa el número de sesiones registradas en ese mes para cada grupo.

Se ha elegido un gráfico de líneas con tres trazos (total de sesiones, sesiones femeninas y sesiones masculinas) porque permite visualizar mes a mes cómo evoluciona cada grupo al mismo tiempo. Al dibujar una línea distinta para mujeres y hombres, junto con la del total, es muy sencillo notar si, por ejemplo, en junio ambos géneros aumentan de forma paralela o si en octubre ellas repuntan mientras ellos se mantienen estables. Además, los colores ayudan a identificar rápidamente picos o caídas en cada categoría sin tener que consultar gráficos por separado, de modo que se aprecia de inmediato cómo se comporta la plataforma a lo largo de 2024.

Del análisis del gráfico 4.4 se pueden extraer las siguientes conclusiones:

- **Más actividad en otoño:** A partir de septiembre se nota un aumento claro en las sesiones, llegando al punto más alto en otoño. Esto es lógico, ya que coincide con el inicio del curso académico, cuando muchos usuarios retoman su actividad.
- **Bajada en verano:** En los meses de verano, especialmente agosto, la actividad cae notablemente. Es un periodo de vacaciones en muchos países, lo que reduce el uso de la plataforma. Este dato es importante al planificar campañas o recursos.
- **Inicio de año tranquilo:** De enero a junio el uso se mantiene relativamente estable, sin grandes picos ni caídas. Es un periodo de actividad moderada pero constante.
- **Más sesiones de chicos que de chicas:** Durante todo el año, se registran más sesiones masculinas que femeninas. Esto podría dar pie a reflexionar sobre cómo incentivar una mayor participación femenina en la plataforma.
- **Utilidad para la gestión de Unibotics:** Conocer estos patrones permite tomar decisiones más informadas sobre el calendario de actividades, campañas de comunicación o soporte técnico. También puede orientar estrategias específicas de inclusión y diversificación de la comunidad de usuarios.

## 4.5 Distribución de frecuencia de usuarios con similar duración total por ejercicio

Este *dashboard* analiza cómo se distribuyen los usuarios de Unibotics según el tiempo total que han dedicado a un ejercicio específico. Utiliza una escala logarítmica para poder representar mejor las diferencias en las duraciones, que pueden ser muy grandes entre unos usuarios y otros.

Los datos provienen de las tablas `log_exercises` y `exercises`. El usuario ve un desplegable que permite seleccionar un ejercicio concreto. Una vez seleccionado, se ejecuta la consulta SQL 4.14, que suma el tiempo total que cada usuario ha dedicado a ese ejercicio.

```sql
SELECT username, SUM(duration) as total_duration
FROM public.log_exercises
WHERE exercise = %s
GROUP BY username
HAVING SUM(duration) > 0
ORDER BY total_duration;
```

Código 4.14: Consulta SQL para obtener la duración total por usuario en un ejercicio específico.

Después de obtener los datos, aplico una transformación logarítmica al valor de duración total. Esto se hace para reducir la dispersión de los datos: como hay usuarios que pueden haber pasado desde unos pocos segundos hasta varias horas en un mismo ejercicio, el uso de una escala logarítmica permite ver toda la distribución de una forma más equilibrada.

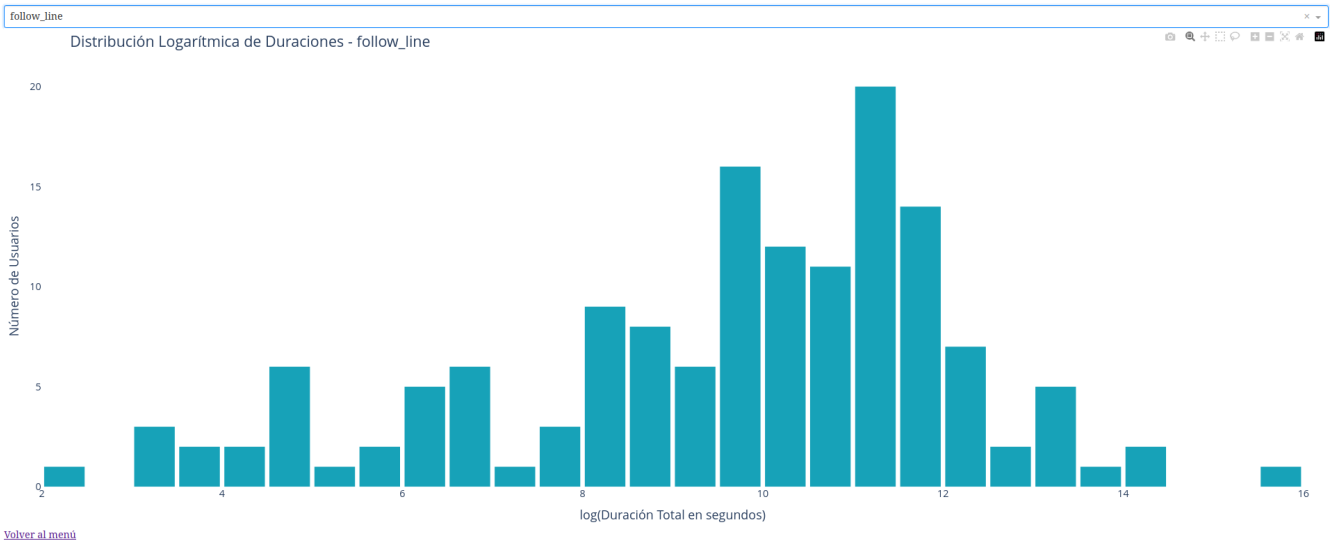


Figura 4.5: Histograma que representa la frecuencia (número de usuarios) de la duración total en segundos (escala log.) por ejercicio. La figura muestra los resultados obtenidos para el ejercicio follow_line.

El *dashboard* 4.5 muestra los datos en un histograma, donde el eje horizontal representa el logaritmo de la duración total (en segundos) y el eje vertical representa el número de usuarios que tienen una duración dentro de cada rango.

Cada barra del histograma indica cuántos usuarios cayeron en ese intervalo de duraciones. Como los valores varían mucho entre usuarios, apliqué la función logarítmica en el eje X para comprimir la escala. De ese modo, en lugar de tener muchas barras amontonadas en valores muy bajos y solo algunos casos aislados al final, se obtienen intervalos más

balanceados.

Elegí este tipo de gráfico porque quería representar visualmente cómo se distribuyen las duraciones entre todos los usuarios que realizaron un ejercicio concreto. Agrupar esos tiempos en intervalos hace muy fácil ver dónde se concentra la mayoría de los usuarios.

Al observar el histograma, se puede identificar de un vistazo si la mayoría se encuentra en un rango de tiempo intermedio, o si existen grupos con duraciones muy largas o muy cortas, apreciándose inmediatamente la forma de la distribución.

Este análisis ayuda a comprender con mayor profundidad cómo se comportan los usuarios al realizar un ejercicio determinado. A partir de esta información, destaco los siguientes aspectos:

- **Distribución de los tiempos:** Permite observar si la mayoría de los usuarios completan el ejercicio en tiempos similares o si existen grandes diferencias entre ellos.
- **Tiempos agrupados en valores bajos:** Pueden indicar que el ejercicio es sencillo o que muchos usuarios dedicaron poco tiempo, posiblemente porque lo abandonaron pronto.
- **Gran variedad de tiempos:** Sugiere que el ejercicio tiene un nivel de dificultad variable o que permite explorar distintas soluciones, lo que implica más tiempo dedicado por parte de los usuarios.
- **Utilidad para los profesores:** Este análisis facilita detectar si el ejercicio está bien equilibrado en dificultad o si requiere ajustes para mejorar la experiencia de aprendizaje.

## 4.6 Distribución del tiempo invertido por usuario en cada ejercicio

Este *dashboard* muestra cómo se distribuye el tiempo que cada usuario ha dedicado a distintos ejercicios de Unibotics. Utiliza diagramas de caja, también llamados *boxplots*, aplicando una escala logarítmica para representar mejor las diferencias entre los usuarios.

Los datos se obtienen de la tabla `log_exercises`, y mediante la consulta SQL 4.15 agrupo la información por nombre de ejercicio y por usuario, sumando la duración total que cada uno ha dedicado a cada ejercicio. Solo se consideran aquellos casos donde el tiempo total registrado es mayor que cero.

```sql
SELECT exercise, username, SUM(duration) as total_duration
FROM public.log_exercises
WHERE duration > 0
GROUP BY exercise, username
HAVING SUM(duration) > 0
ORDER BY exercise;
```

Código 4.15: Consulta SQL para obtener duración total por ejercicio y usuario.

Una vez extraídos los datos, calculo el logaritmo de la duración total para cada usuario. Esto se hace para normalizar los valores, ya que algunos usuarios pueden tener tiempos muy pequeños y otros muy grandes, y la escala logarítmica ayuda a visualizar mejor esa variabilidad.

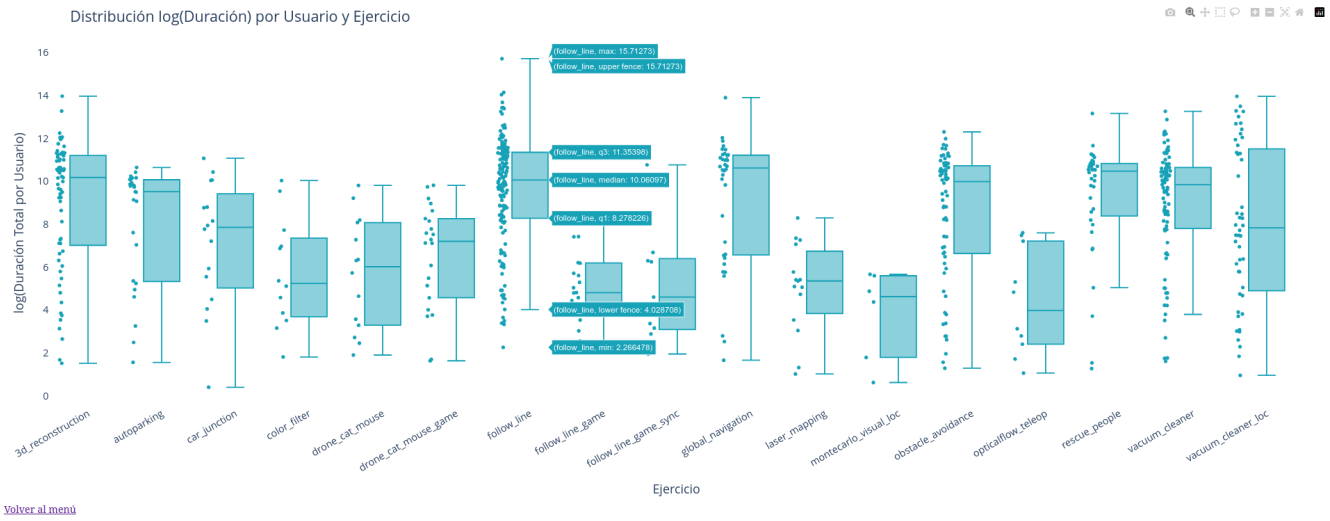


Figura 4.6: Boxplots que representan la distribución del tiempo invertido por cada usuario en la resolución de cada ejercicio. Los puntos individuales muestran los valores registrados para cada usuario individual en un ejercicio.

En este *dashboard* 4.6, cada ejercicio se representa en el eje horizontal, mientras que en el eje vertical se muestra el logaritmo del tiempo total invertido por los usuarios. Para cada ejercicio, el *boxplot* muestra:

- La mediana de las duraciones (línea central del rectángulo).
- El rango intercuartílico (el ancho de la caja), que representa a la mayoría de los usuarios.
- Los valores extremos o atípicos (puntos dispersos fuera de la caja).

Gracias a este tipo de gráfico, es posible visualizar de forma clara cómo varía la dedicación de los usuarios en cada ejercicio y detectar si hay muchos usuarios que dedican tiempos muy distintos o si la mayoría se concentra en valores similares. Además, se muestran todos los puntos individuales sobre los diagramas, lo que permite ver mejor la dispersión real de los datos.

Este *dashboard* me permite comparar rápidamente el comportamiento de los usuarios en cada ejercicio. A partir de los datos, destaco los siguientes puntos importantes:

- **Uniformidad en el tiempo dedicado:** Permite observar si la mayoría de los alumnos invierten un tiempo similar o si existen grandes diferencias entre ellos.
- **Ejercicios con tiempos elevados:** Identifica aquellos en los que algunos usuarios dedican mucho más tiempo, lo que podría indicar mayor dificultad o especial interés.
- **Casos extremos:** Detecta actividades con muchas diferencias marcadas en los tiempos, lo que puede señalar problemas en el planteamiento o una dificultad muy variable entre estudiantes.
- **Información para los profesores:** Ayuda a identificar ejercicios que podrían requerir cambios o mejoras para optimizar la experiencia de aprendizaje.
- **Orientación para diseñadores de contenidos:** Facilita evaluar si los ejercicios generan el trabajo esperado o si convendría añadir ayudas o dividir las actividades en partes más manejables.

## 4.7 Distribución del tiempo de sesión por usuario en cada ejercicio

El *dashboard* 4.7 es muy similar al 4.6, ya que también utiliza diagramas de caja (*boxplots*) para representar los datos, pero con una diferencia importante: en lugar de analizar el tiempo total que cada usuario dedica a un ejercicio, aquí analizo la duración de cada sesión individual.

Mientras que en el *dashboard* 4.6 sumaba todas las sesiones de un usuario para un ejercicio determinado, en este caso estudio cada sesión por separado, sin agruparlas. Además, también aplico una escala logarítmica para representar mejor las diferencias en los tiempos, ya que algunas sesiones pueden durar solo unos pocos segundos y otras mucho más.

Para obtener estos datos, utilizo la consulta SQL 4.16, que selecciona, para cada registro de sesión, el nombre del ejercicio y la duración concreta de esa sesión.

```sql
SELECT exercise, duration
FROM public.log_exercises
WHERE duration > 0
ORDER BY exercise;
```

Código 4.16: Consulta SQL para obtener duración y ejercicio ordenados por ejercicio.

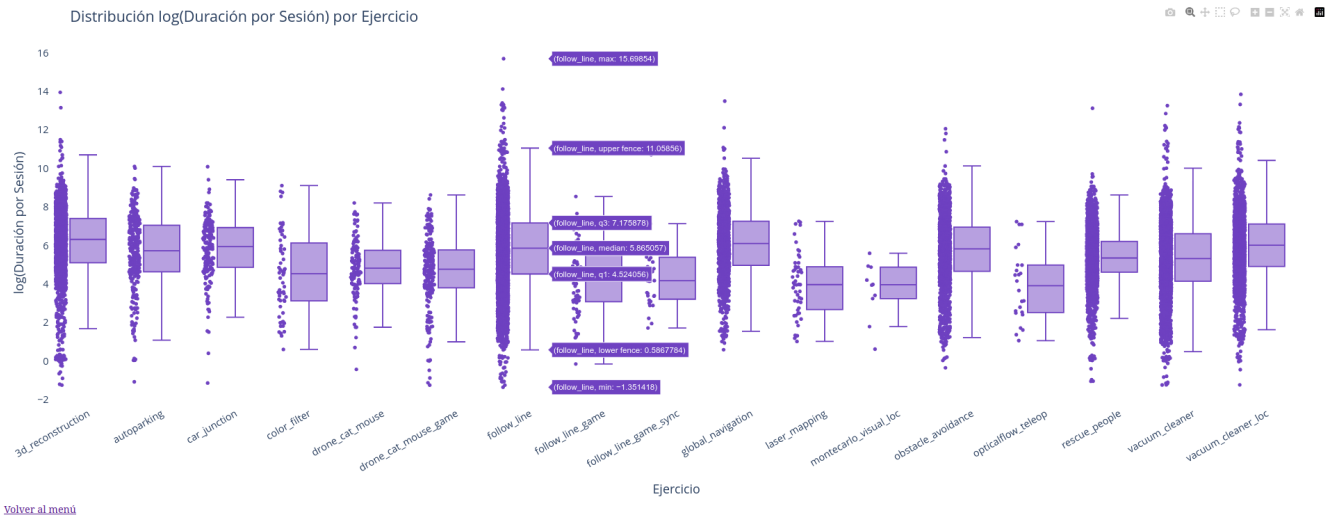


Figura 4.7: Boxplots que representan la distribución del tiempo de sesión por usuario en cada ejercicio. Los puntos individuales indican los valores registrados para cada usuario individual en un ejercicio.

El objetivo del *dashboard* 4.7 es ofrecer una visión más detallada sobre cómo varía el tiempo de trabajo en cada intento o sesión concreta, algo que no era visible en el *dashboard* 4.6, donde solo veía el esfuerzo acumulado de cada usuario. Aquí puedo detectar, por ejemplo, si un ejercicio tiene sesiones generalmente cortas o si hay mucha variación entre los intentos de distintos usuarios.

Esta representación es útil para entender mejor el comportamiento real dentro de cada ejercicio. Permite ver si los estudiantes tienden a resolver los ejercicios de forma rápida o si, por el contrario, dedican mucho tiempo en intentos sucesivos. Además, me ayuda a detectar ejercicios que generan más dispersión en el tiempo por sesión, lo que puede ser una pista sobre su dificultad o sobre la necesidad de mejorar su planteamiento o las instrucciones proporcionadas.

# Capítulo 5

# Conclusiones y trabajos futuros

## 5.1 Consecución de objetivos

A lo largo del desarrollo de este Trabajo Fin de Grado, he ido cumpliendo de forma progresiva y satisfactoria los objetivos que me planteé al inicio. El objetivo principal, que consistía en crear una herramienta capaz de extraer, analizar y visualizar los registros de actividad de los usuarios en la plataforma Unibotics, se ha alcanzado plenamente, consolidando así todo el trabajo realizado a lo largo del proyecto.

En cuanto a los objetivos específicos:

- **Extracción de datos:** He logrado una recopilación estructurada y precisa de la información almacenada en la base de datos de Unibotics. Para ello, utilicé consultas SQL optimizadas que me han permitido obtener los datos relevantes de las diferentes tablas, asegurando su calidad y consistencia para el posterior análisis. Gracias a este proceso, ahora es posible aprovechar de forma más clara y ordenada toda la información histórica que recoge la plataforma.

- **Análisis de estadísticas básicas:** He calculado métricas fundamentales como el número de sesiones por usuario, la duración de uso por ejercicio y la frecuencia de acceso. Estos datos me han permitido identificar patrones generales de comportamiento dentro de la plataforma, facilitando la comprensión de cómo los usuarios interactúan con ella. Además, este análisis ha aportado ideas sobre cómo mejorar la plataforma o cómo adaptar la experiencia de aprendizaje de forma más personalizada para cada usuario.

- **Representación gráfica de la información:** He creado varios *dashboards* interactivos utilizando Dash y Plotly, todos integrados en una aplicación web sencilla mediante

Flask. Estas visualizaciones permiten ver de forma clara y rápida cómo están usando la plataforma los usuarios, lo cual resulta útil tanto para profesores como para administradores de Unibotics o diseñadores de ejercicios que deseen orientar nuevos proyectos. Además, he prestado atención al diseño visual para asegurar que todo sea intuitivo y accesible, facilitando la navegación dentro de la aplicación.

Además, he implementado un sistema para estimar el género de los usuarios, lo cual ha permitido incluir análisis desagregados por sexo, aportando así una variable adicional al estudio del comportamiento y enriqueciendo las conclusiones obtenidas.

## 5.2 Aplicación de lo aprendido

En el Grado en Ingeniería en Tecnologías de Telecomunicación se abordan muchas ramas distintas: electrónica, señales, redes, antenas, programación, entre muchas otras. Sin embargo, para el desarrollo de este trabajo, las asignaturas que más me han servido han sido aquellas orientadas a la programación, al desarrollo de software y a la gestión de datos. Durante la creación de la herramienta he podido aplicar gran parte de lo aprendido a lo largo de estos años, y quiero destacar especialmente las siguientes asignaturas:

1. **Fundamentos de la Programación.** Fue mi primera toma de contacto con la programación y donde, poco a poco, le fui cogiendo el gusto. Sentó las bases que me han acompañado durante toda la carrera.

2. **Programación de Sistemas de Telecomunicación.** Supuso un paso más en dificultad, con desarrollos más complejos que me obligaron a pensar de forma más estructurada. Aquí empecé a enfrentarme a retos reales de programación.

3. **Desarrollo de Aplicaciones Telemáticas.** Aprendí muchos conceptos útiles relacionados con HTML, CSS y el desarrollo de interfaces. Fue clave para entender cómo se construyen las capas visuales y cómo interactúan con la lógica del servidor.

4. **Servicios y Aplicaciones Telemáticas.** Esta fue, sin duda, la asignatura más crucial para este TFG. Aprendí a trabajar con Python y Django, a estructurar un proyecto completo, a usar bases de datos, a hacer peticiones HTTP y, en general, a construir algo que se parece mucho a una aplicación real.

5. **Ingeniería de Sistemas de Información.** Aquí aprendí casi todo lo que sé sobre bases de datos, desde su diseño hasta cómo manejarlas correctamente. Ha sido fundamental para entender y extraer los datos de la base de datos de Unibotics.

6. **Ingeniería de Sistemas Telemáticos.** Además de reforzar conocimientos de programación orientada a objetos con Java, esta asignatura me enseñó algo muy importante:

cómo buscar documentación y soluciones por mi cuenta para resolver problemas de programación.

7. **Sistemas Operativos.** Me permitió profundizar aún más en programación y en el uso de herramientas del sistema operativo. Aprendí, por ejemplo, a usar Docker, lo cual ha sido crucial en este proyecto.

8. **Estadística y Control de Calidad.** Me ayudó a entender cómo interpretar correctamente los datos y cómo representarlos de forma clara y útil.

## 5.3 Lecciones aprendidas

Aunque durante el desarrollo del proyecto he podido aplicar muchos conocimientos adquiridos a lo largo de la carrera, también me he encontrado con situaciones nuevas que me han obligado a aprender por el camino. Gracias a ello, he adquirido nuevas habilidades y me he enfrentado a retos que antes no sabía cómo resolver. A continuación, resumo algunas de las cosas nuevas que he aprendido durante este trabajo:

1. He mejorado mis conocimientos sobre bases de datos.

2. He aprendido a utilizar la biblioteca Dash para la creación de visualizaciones interactivas.

3. He comprendido cómo representar datos de forma adecuada según el contexto.

4. He desarrollado habilidades en análisis gráfico para interpretar correctamente los datos y entender qué significan.

5. He aprendido a programar en LaTeX.

6. He mejorado en el seguimiento y planificación de proyectos.

7. He reforzado y ampliado mis habilidades programando en Python.

## 5.4 Trabajos futuros

Aunque este proyecto ha llegado a su fin en esta etapa, hay muchas líneas de desarrollo y mejora que pueden abordarse en el futuro para seguir ampliando las funcionalidades y el valor de la plataforma.

Actualmente, el trabajo ha concluido con la integración de varios *dashboards* mediante Flask, como solución temporal debido a un bug en la plataforma de Unibotics que impide incrustar correctamente las gráficas desarrolladas con Dash. Uno de los objetivos prioritarios a futuro será, una vez la administración de Unibotics resuelva ese error, integrar de forma nativa estos *dashboards* con Dash en la propia plataforma, lo que mejorará la integración visual y funcional del sistema.

Además, este trabajo se ha desarrollado con los datos disponibles en el momento, pero la plataforma está en constante crecimiento. A medida que se incremente la actividad de los usuarios, y con ella el volumen de información almacenada, será posible generar estadísticas mucho más completas y variadas. Contar con más datos permitirá hacer análisis más precisos, identificar tendencias y estudiar el comportamiento de los usuarios en mayor profundidad.

También es importante tener en cuenta que, con el tiempo, podrían añadirse nuevas variables a la base de datos, lo cual abriría la puerta a explorar nuevos indicadores y tipos de análisis. Por ejemplo, se podrían estudiar patrones de uso, evolución del rendimiento de los usuarios o realizar comparaciones entre distintos periodos o grupos.

# Siglas

**IDE** Integrated Development Enviroment (Entorno de Desarrollo Integrado). 8, 9

# Apéndice A

# Ejemplos base de datos Unibotics

En este apéndice se presentan ejemplos de las consultas SQL utilizadas para extraer datos de las tablas que conforman la base de datos de Unibotics. A continuación, se describe cómo se organizan los datos en las tablas de la base de datos y se presentan ejemplos visuales para ayudar a comprender la estructura de la información.

Una base de datos está organizada en tablas, donde cada fila representa un registro (por ejemplo, un ejercicio o un usuario), y cada columna representa un campo de ese registro (por ejemplo, el nombre de un usuario o la duración de un ejercicio). A continuación, se detallan algunas de las tablas más relevantes para este proyecto.

Tabla A.1: Muestra de registros de `Log_exercises`.

| id | username | start_date | end_date | duration | exercise |
|---|---|---|---|---|---|
| 298 | amgurjc | 2021-12-01 17:24:08.512187 | 2021-12-01 17:25:52.618119 | 104.105932 | obstacle_a |
| 299 | mariamh | 2021-12-01 17:23:49.164972 | 2021-12-01 17:25:59.414500 | 130.249528 | rescue_pe |
| 300 | mariamh | 2021-12-01 17:26:04.868734 | 2021-12-01 17:28:15.071450 | 130.202716 | rescue_pe |
| 301 | fgomezl | 2021-12-01 17:28:33.600277 | 2021-12-01 17:30:53.251092 | 139.650815 | obstacle_a |
| 302 | mariamh | 2021-12-01 17:28:23.518706 | 2021-12-01 17:31:45.124709 | 201.606003 | rescue_pe |
| 303 | amgurjc | 2021-12-01 17:28:07.886411 | 2021-12-01 17:32:38.985946 | 271.099535 | obstacle_a |
| 304 | amgurjc | 2021-12-01 17:32:39.026606 | 2021-12-01 17:39:19.107978 | 400.081374 | obstacle_a |
| 305 | amgurjc | 2021-12-01 17:39:19.174050 | 2021-12-01 17:41:18.162209 | 118.988159 | obstacle_a |
| 306 | carlosip | 2021-12-01 17:22:12.768520 | 2021-12-01 17:42:07.383919 | 1194.615399 | obstacle_a |
| 307 | chuismiguel | 2021-12-01 16:44:58.295889 | 2021-12-01 17:46:26.431287 | 3688.135398 | obstacle_a |
| 308 | mariamh | 2021-12-01 17:32:19.370519 | 2021-12-01 17:46:46.955035 | 867.584516 | rescue_pe |

La tabla A.1 muestra los registros de las sesiones de ejercicios realizadas por los usuarios. Cada fila representa una sesión, y los campos incluyen información sobre el usuario, el ejercicio realizado, la fecha de inicio y finalización, y la duración de la sesión.

Tabla A.2: Muestra de registros de `Log_session`.

| id | username | start_date | end_date | duration | client_i |
|---|---|---|---|---|---|
| 377 | superManuel | 2021-06-07 10:05:04.455056 | 2021-06-07 10:50:31.317011 | 2726.861955 | 83.32.26 |
| 378 | superManuel | 2021-06-04 14:18:34.770146 | 2021-06-04 18:34:38.774011 | 15244.003865 | 83.32.26 |
| 379 | davidrol | 2021-06-07 11:55:21.250197 | 2021-06-07 11:55:21.250203 | 0.000006 | 88.11.22 |
| 380 | Civantos | 2021-06-06 00:32:46.037435 | 2021-06-06 00:32:46.037439 | 0.000004 | 83.56.24 |
| 381 | civantos | 2021-06-09 14:29:13.635510 | 2021-06-09 14:29:13.635518 | 0.000008 | 176.87.4 |
| 382 | paulam09 | 2021-06-09 14:33:13.324971 | 2021-06-09 14:33:13.324975 | 0.000004 | 176.83.2 |
| 383 | davidrol | 2021-06-09 15:59:18.707238 | 2021-06-09 15:59:18.707242 | 0.000004 | 88.11.22 |
| 384 | davidrol | 2021-06-09 18:02:36.714896 | 2021-06-09 18:33:13.979659 | 1837.264763 | 88.11.22 |
| 385 | davidrol | 2021-06-10 12:08:24.930763 | 2021-06-10 13:36:08.206773 | 5263.276010 | 88.11.22 |
| 386 | davidrol | 2021-06-10 13:36:13.849895 | 2021-06-10 13:36:13.849901 | 0.000006 | 88.11.22 |

La tabla A.2 contiene información sobre los registros de visitas de los usuarios, incluyendo detalles como el usuario, la fecha de inicio y finalización de la visita, y el navegador y país desde donde se accedió.

Tabla A.3: Muestra de registros de `Exercises`.

| id | exercise_id | name | description |
|---|---|---|---|
| 1 | follow_line | Follow Line | The goal of this exercise is to perform a PID r |
| 2 | vacuum_cleaner | Vacuum Cleaner | Vacuum Cleaner exercise using React and RA |
| 3 | autoparking | Autoparking | Autoparking exercise testing |
| 4 | follow_person | Follow Person | Follow a person with kobuki robot |
| 5 | vacuum_cleaner_loc | Localized Vacuum Cleaner | Localized vacuum cleaner |
| 6 | global_navigation | Global Navigation | Global navigation exercise using REACT and |
| 7 | rescue_people | Rescue People | Rescue People exercise |
| 8 | obstacle_avoidance | Obstacle Avoidance | Obstacle Avoidance exercise using React and |
| 9 | 3d_reconstruction | 3D Reconstruction | 3D Reconstruction exercise using React and R |
| 10 | amazon_warehouse | Amazon Warehouse | Control an amazon-like robot to organize a w |
| 11 | montecarlo_laser_loc | Montecarlo Laser Loc | Calculate the position of the robot based on t |

La tabla A.3 almacena información sobre los diferentes ejercicios disponibles en la plataforma, como el nombre del ejercicio, su descripción, los tags asociados y su estado. Cada fila de esta tabla corresponde a un ejercicio específico.

La tabla A.4 contiene información sobre los usuarios registrados en la plataforma, incluyendo su nombre de usuario, nombre real, género, y la última fecha de inicio de sesión. También se almacena si el usuario tiene privilegios de superusuario.

A continuación, ya teniendo en contexto como se almacena la información en las tablas de la base de datos, se presentan ejemplos de consultas SQL que permiten extraer

Tabla A.4: Muestra de registros de `Common_user`.

| id | last_login | is_superuser | username | first_name | gender |
|---|---|---|---|---|---|
| 275 | 2024-04-02 09:56:01.352868+00 | false | i.linares | Ismael | M |
| 276 | 2023-04-12 15:09:51.994983+00 | false | jdpul_ | Jorge | M |
| 277 | 2022-04-19 10:26:08.507501+00 | false | enrique | Enrique | M |
| 278 | 2022-04-26 08:45:27.213432+00 | false | v.gilabert.2021 | VICENTE | M |
| 280 | 2022-04-19 15:29:00.179681+00 | false | hurta2 | Sergio | M |
| 281 | 2022-04-19 15:28:43.646237+00 | false | nacho9gs | Ignacio | M |
| 282 | 2022-04-19 23:38:44.874175+00 | false | alonsocn | Alonso | M |
| 283 | 2022-04-19 22:17:17.513394+00 | false | kcoutinho | Katherine | F |
| 284 | 2022-03-20 18:04:25.378575+00 | false | cv.lungu | Costin Valentin | M |
| 285 | NULL | false | PaulaST | Paula | F |
| 286 | 2022-12-12 01:08:10.920153+00 | false | PaulaS | Paula | F |

información relevante de las tablas mencionadas anteriormente. Estas consultas pueden ser utilizadas para analizar el comportamiento de los usuarios, las sesiones de ejercicio y el desempeño en los diferentes ejercicios.

## A.1 Ejemplo 1: Listado de usuarios que han realizado un ejercicio concreto

Si queremos saber todos los usuarios que han ejecutado un ejercicio concreto como puede ser `rescue_people`, basta con:

1. Seleccionar el campo `username` de la tabla `Log_exercises`.
2. Filtrar por `exercise = 'rescue_people'`.
3. Eliminar duplicados con `DISTINCT`.
4. Ordenar los resultados alfabéticamente.

```
SELECT DISTINCT username
FROM public.Log_exercises
WHERE exercise = 'rescue_people'
ORDER BY username;
```

Código A.17: Consulta SQL para listar usuarios que han ejecutado `rescue_people`.

Con la consulta A.17 obtenemos exactamente la lista de usuarios que han ejecutado el ejercicio `rescue_people`. A continuación se detalla su funcionamiento:

1. `SELECT DISTINCT username`: toma cada nombre de usuario sin repetirlo.
2. `FROM public.Log_exercises`: indica la tabla donde se buscan los registros de ejercicios.
3. `WHERE exercise = 'rescue_people'`: filtra sólo las filas que correspondan a ese ejercicio.
4. `ORDER BY username`: muestra los nombres de usuario ordenados de la A a la Z.

## A.2 Ejemplo 2: Duración media por ejercicio y usuario

Para calcular el tiempo promedio (en segundos) que cada usuario invierte en cada ejercicio:

1. Agrupamos los registros de la tabla `Log_exercises` por `username` y por `exercise`.
2. Aplicamos `AVG(duration)` para obtener la media de la duración.
3. Redondeamos el resultado a dos decimales con `ROUND(..., 2)`.

```
SELECT
username,
exercise,
ROUND(AVG(duration)::numeric, 2) AS avg_duration_seconds
FROM public.Log_exercises
GROUP BY username, exercise
ORDER BY username, exercise;
```

Código A.18: Consulta SQL para calcular la duración media de cada ejercicio por usuario.

Con la consulta A.18 conseguimos calcular, para cada usuario y cada ejercicio, el tiempo medio que han invertido en segundos. A continuación se explica su funcionamiento:

1. `AVG(duration)`: calcula la media de la columna `duration` (en segundos).
2. `ROUND(..., 2)`: redondea esa media a dos decimales para mayor claridad.

3. `GROUP BY username, exercise`: agrupa los registros por usuario y ejercicio, de modo que la función `AVG` opere en cada grupo.

4. `ORDER BY username, exercise`: ordena primero por usuario y, dentro de cada usuario, por ejercicio.

## A.3 Ejemplo 3: Número total de sesiones de ejercicio por usuario

Para obtener cuántas sesiones de ejercicio ha iniciado cada usuario (sin distinguir tipo de ejercicio):

1. Contamos todas las filas de `Log_exercises` para cada `username`.

2. Agrupamos por `username`.

3. Ordenamos de mayor a menor número de sesiones.

```
SELECT
username,
COUNT(*) AS total_sessions
FROM public.Log_exercises
GROUP BY username
ORDER BY total_sessions DESC, username;
```

Código A.19: Consulta SQL para contar el total de sesiones de ejercicio por usuario.

Con la consulta A.19 conseguimos obtener el número total de sesiones de ejercicio iniciadas por cada usuario. A continuación se explica su funcionamiento:

1. `COUNT(*)`: cuenta todas las filas (sesiones) de cada usuario.

2. `GROUP BY username`: agrupa los registros por nombre de usuario.

3. `ORDER BY total_sessions DESC, username`: muestra primero a quien más sesiones tiene; si hay empate, ordena por nombre.

## A.4 Ejemplo 4: Obtener la última fecha de sesión de cada usuario

Para saber cuándo fue la última vez que cada usuario accedió a la plataforma (tabla `Log_session`), podemos usar una consulta muy sencilla que agrupe por `username` y obtenga el valor máximo de `end_date`:

1. Seleccionamos `username` y la fecha máxima de `end_date` de `Log_session`.
2. Agrupamos por `username` para que la función `MAX` se aplique a cada usuario.
3. Ordenamos el resultado de más reciente a más antiguo.

```
SELECT
username,
MAX(end_date) AS last_end_date
FROM public.Log_session
GROUP BY username
ORDER BY last_end_date DESC;
```

Código A.20: Consulta SQL para obtener la última fecha de sesión de cada usuario.

Con la consulta A.20 conseguimos obtener, para cada usuario, la fecha más reciente en que finalizó una sesión. A continuación se explica su funcionamiento:

1. `SELECT username, MAX(end_date) AS last_end_date`: elige el campo `username` y calcula la fecha más reciente de `end_date` para cada uno.
2. `FROM public.Log_session`: indica que trabajamos sobre la tabla de registros de visitas.
3. `GROUP BY username`: agrupa todos los registros por usuario, de modo que `MAX(end_date)` devuelva la última fecha de cada grupo.
4. `ORDER BY last_end_date DESC`: ordena los usuarios de quien accedió más recientemente a quien lo hizo hace más tiempo.

# Referencias


[1] Paul J. Deitel y Harvey M. Deitel. *Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and The Cloud*. Pearson, 2019.

[2] Luan Einhardt, Tatiana Aires Tavares y Cristian Cechinel. «Moodle analytics dashboard: A learning analytics tool to visualize users interactions in moodle». En: *2016 XI Latin American Conference on Learning Objects and Technology (LACLO)*. IEEE. 2016, págs. 1-6.

[3] R Graph Gallery. *The R Graph Gallery – Inspiration and Help for R Charts*. https://r-graph-gallery.com/. Consultado el 2025-06-05. 2025.

[4] M Grinberg. «Flask Web Development: Developing Web Applications with Python. 2018». En: *URL http://flaskbook. com* ().

[5] Wes McKinney. *Python for data analysis: Data wrangling with pandas, numpy, and jupyter*. O'Reilly Media, Inc, 2022.

[6] Jeffrey Nickoloff y Stephen Kuenzli. *Docker in action*. Simon y Schuster, 2019.

[7] Inc. Plotly. *Dash by Plotly – Framework para crear aplicaciones web interactivas en Python (Documentación)*. https://dash.plotly.com/. Consultado el 2025-05-29. 2025.

[8] Equipo de desarrollo de Psycopg. *Psycopg – Adaptador de base de datos PostgreSQL para Python (Documentación)*. https://www.psycopg.org/docs/. Consultado el 2025-05-29. 2025.

[9] Pypi. *gender-guesser – Estimador de género basado en nombres*. https://pypi.org/project/gender-guesser/. Consultado el 2025-05-29. 2025.

[10] F.M. Rico. *A Concise Introduction to Robot Programming with ROS 2*. CRC Press, 2025. ISBN: 9781040363560. DOI: 10.1201/9781003516798.

[11] Adam Schroeder, Christian Mayer y Ann Marie Ward. *The book of Dash: build dashboards with Python and Plotly*. No Starch Press, 2022.

[12] Unibotics. *Unibotics – Plataforma educativa de robótica*. https://unibotics.org. Consultado el 2025-05-29. 2025.

[13] Wikipedia. *Paradoja de Simpson*. https://es.wikipedia.org/wiki/Paradoja_de_Simpson. Consultado el 2025-05-29. 2025.