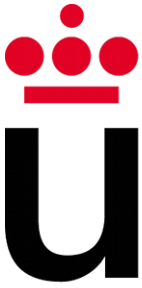

Universidad
Rey Juan Carlos

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/a : Alejandro Aguilera López
Tutor/a : 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

Aquí vienen los agradecimientos...

Hay más espacio para explayarse y explicar a quién agradeces su apoyo o ayuda para haber acabado el proyecto: familia, pareja, amigos, compañeros de clase...

También hay quien, en algunos casos, hasta agradecer a su tutor o tutores del proyecto la ayuda prestada...

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

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 quiere empezar en el mundo de la robótica.

Por ejemplo, la instalación de entornos como ROS 2 y Gazebo [10] pueden convertirse en un verdadero desafío para quienes empiezan. En ROS 2 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 cual ir probando nuestros códigos, ya que a diferencia de otras materias la robótica es un campo que su aprendizaje es a base de prueba y error.

Este trabajo de fin de grado gira entorno 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 poder 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 acceso a recursos, participación en foros, 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 sólo 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 API que permita extraer y procesar estos datos. Además, se diseñarán unas dashboards interactivas, donde se podrán ver de forma clara y detallada toda la información obtenida a través de la API.

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

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.

1.1.2 Objetivos específicos

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

- **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 de comportamiento de usuarios.** Calcular métricas como número de sesiones, tiempo de uso por ejercicio y 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 permitan interpretar de forma rápida y precisa las conclusiones extraídas sobre el comportamiento de los usuarios.

1.2 Planificacion

crear gant

1.3 Estructura de la memoria

Por último, en esta sección se introduce a alto nivel la organización del resto del documento y qué contenidos se van a encontrar 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 TFG (capítulo 2).
- En el capítulo 3 se describe la arquitectura de la plataforma de Unibotics.
- En el capítulo 4 se presentan las dashboard y se realiza un análisis de la información que se extrae de ellas.
- Por último, se presentan las conclusiones del proyecto así como los trabajos futuros que podrían derivarse de éste (capítulo 5).

Capítulo 2

Tecnologías

En este capitulo describiré las tecnologías utilizadas para este proyecto.

2.1 Python

Sobra el 1.3 Python es un lenguaje de programación de alto nivel, interpretado, dinámico y fuertemente tipado. Python es un lenguaje multiparadigma, lo que quiere decir que permite desarrollar software utilizando distintos enfoques como la programación orientada a objetos, la programación funcional y la programación imperativa [1].

El principal motivo por el cual he elegido Python como lenguaje de programación para este trabajo es que la plataforma Unibotics está desarrollada en este lenguaje, por lo que usar otro sería ineficiente y una complicación innecesaria. Además, Python destaca por su simplicidad y legibilidad, lo que facilita el desarrollo y mantenimiento del código, su gran comunidad que ofrece soporte constante y su gran variedad de bibliotecas en las cuales se apoya este proyecto. A continuación se enumeran y se detallan el funcionamiento de cada una:

2.1.1 Pandas

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

Una de las razones por las que he elegido Pandas en 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 manera intuitiva. La Series actúa como un array unidimensional con etiquetas asociadas, mientras que el DataFrame es una tabla bidimensional que permite realizar operaciones similares a las de bases de datos o herramientas como Excel.

En el contexto de este proyecto, Pandas es la mejor opción porque la plataforma en la que se desarrollará ya 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 nos permite enfocarnos en la interpretación y visualización de la información, en lugar de perder 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, lo he utilizado porque facilita la creación de gráficos atractivos e interactivos para visualizar los datos de la plataforma Unibotics. Con Dash, puedo generar visualizaciones que permiten a los usuarios interactuar con los datos de manera intuitiva, como hacer filtros o explorar diferentes métricas. Esto es ideal para representar información compleja de forma clara y visualmente atractiva. Además, Dash se integra muy bien con otras bibliotecas de Python, lo que hace que sea fácil crear gráficos de todo tipo, desde barras hasta mapas interactivos.

2.1.3 Psycopg2

Psycopg2 es una librería de Python que permite conectar aplicaciones Python con bases de datos PostgreSQL. Es ampliamente utilizada debido a su eficiencia y facilidad de uso para interactuar con bases de datos [8].

En este proyecto, he utilizado Psycopg2 para ejecutar consultas SQL directamente desde Python, lo que permite extraer, insertar y modificar datos de manera eficiente. Esta librería ofrece una interfaz sencilla y rápida para gestionar la conexión con la base de datos y ejecutar comandos SQL de forma segura, evitando vulnerabilidades como la inyección de SQL. Además, su integración con otras bibliotecas de Python, como Pandas 2.1.1, facilita la manipulación y análisis de los datos obtenidos desde la base de datos, lo que hace que sea más fácil la integración con Dash 2.1.2.

2.2 Flask

Flask es un framework web ligero y flexible para Python que permite desarrollar aplicaciones web de manera rápida y sencilla. Es de código abierto y tiene una amplia comunidad de usuarios y desarrolladores que contribuyen constantemente a su evolución. Flask es popular por su simplicidad, lo que lo hace adecuado tanto para aplicaciones pequeñas como grandes[4].

En este proyecto, utilicé Flask para crear una aplicación simple que sirviera como entorno de prueba e integrara los dashboards, con el fin de realizar las pruebas necesarias antes de la integración final.

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 donde se despliegue [6].

En este proyecto hemos utilizado Docker principalmente para levantar la base de datos PostgreSQL de Unibotics en un contenedor independiente. De esta manera, pude iniciar, detener o reiniciar la base de datos con un solo comando, sin necesidad de instalar PostgreSQL en el sistema anfitrión ni preocuparme por posibles conflictos con otras versiones de la base de datos. Además, al usar Docker, tengo la garantía de que el entorno de ejecución es idéntico en mi máquina local y en cualquier otro entorno de pruebas o producción donde se despliegue la plataforma.

2.4 Entorno de desarrollo: PyCharm

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

Proporciona análisis de código, un depurador gráfico, una consola de Python integrada, control de versiones y, además, soporta desarrollo web con Django. Todas estas características lo convierten en un entorno completo e intuitivo, idóneo para el desarrollo de proyectos académicos como el que nos ocupa. En la figura 2.1 se muestra la estructura de

nuestro proyecto dentro de PyCharm, donde puede apreciarse cómo está organizada la carpeta app cambiar mas adelante con nombres definitivos y explicar que es cada cosa

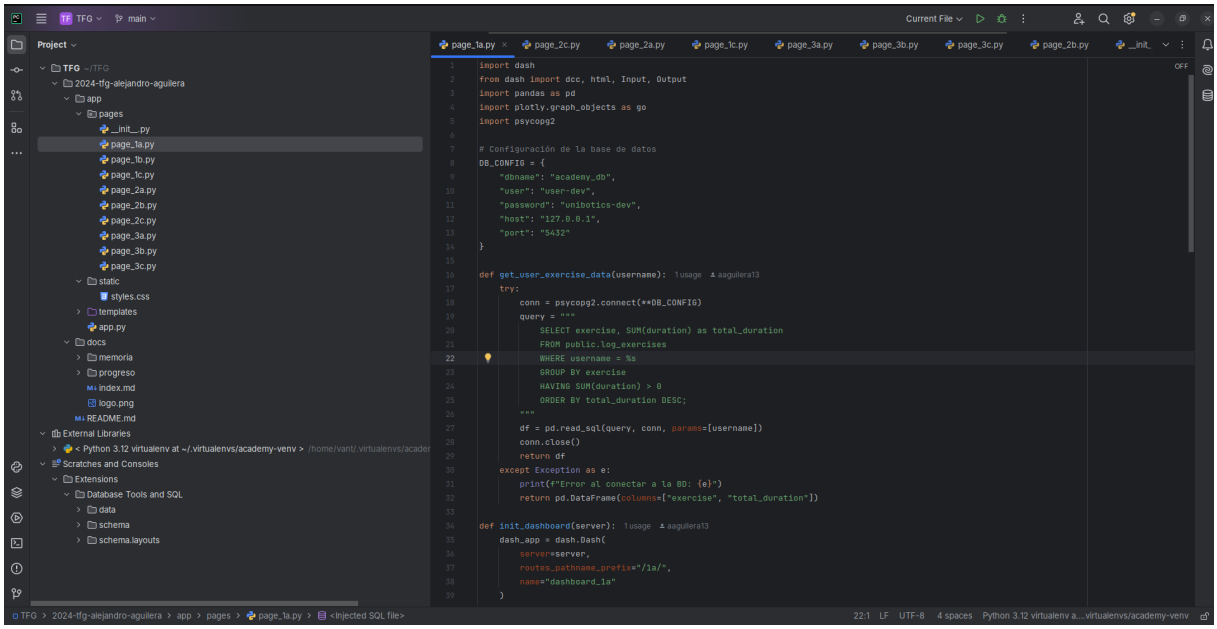

Figura 2.1: Estructura del proyecto en PyCharm

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.

2.6 Plataformas de ejecución

Este proyecto se ha probado en entornos locales tanto bajo Windows como bajo Linux. En Windows se utilizó 2.5 como entorno de desarrollo, mientras que en Linux se empleó 2.4.

Asimismo, las dashboards desarrolladas han sido verificadas en ambos sistemas operativos, comprobando su correcto funcionamiento y compatibilidad. De este modo, se garantiza que tanto Unibotics en local como las dashboard implementadas 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 Unibotics y sus componentes, así como la estructura de la base de datos PostgreSQL y los diferentes datos que gestiona. A continuación, se expone como se ha realizado el despliegue local en el entorno "D1", incluyendo la configuración de contenedores Docker y la puesta en marcha de los servicios. Seguidamente, se muestra cómo se construyeron los dashboards de análisis paso a paso, desde la extracción y tratamiento de datos hasta la generación de gráficas interactivas. Por último, se explica el desarrollo de un script para estimar el género de los usuarios a partir de sus nombres y su integración en la base de datos.

3.1 Arquitectura general

Como ya dije antes, 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 pueden 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, de la cual obtendremos los datos a través de nuestra API para así poder representarlos 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.

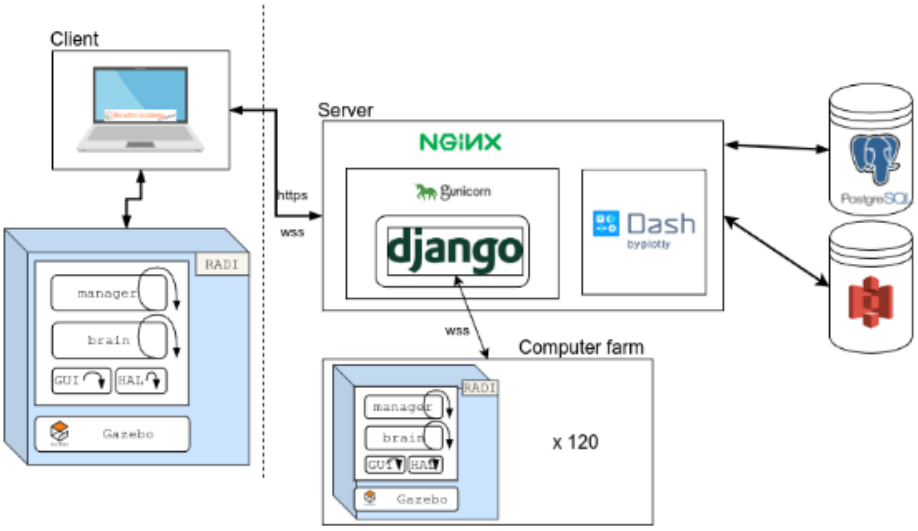

Figura 3.1: Estructura de Unibotics.

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.

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 realizaron. 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.

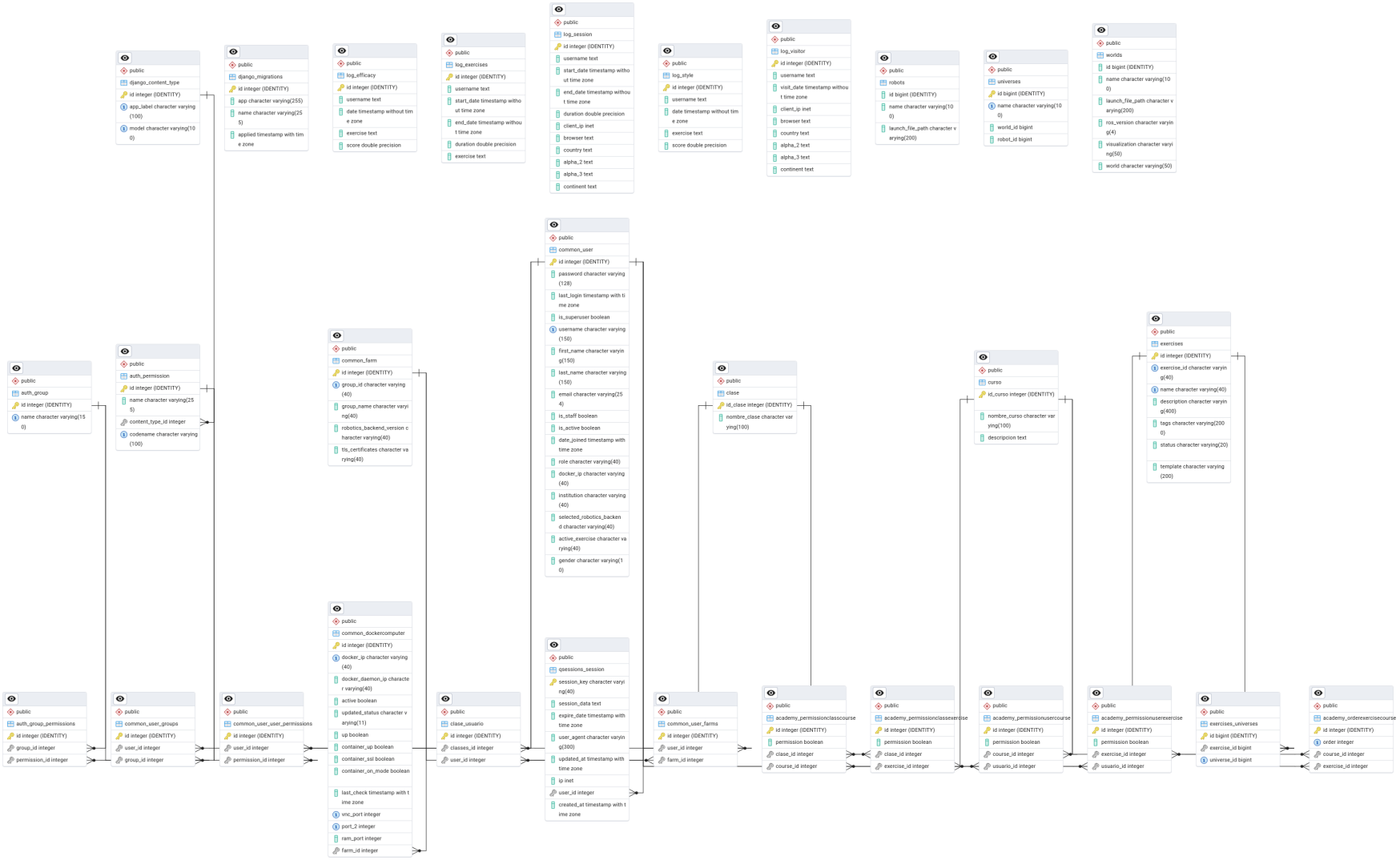

Figura 3.2: Diagrama entidad-relación.

Finalmente, en el apéndice A A, encontrarás ejemplos de consultas SQL que te permitirán 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, instalé un entorno completo en mi máquina Linux (despliegue "D1"). En primer lugar, creé un entorno virtual de Python 3.8 para aislar las dependencias del proyecto sin afectar al sistema. A continuación, cloné el repositorio de `Unibotics-webserver` junto con sus submódulos y ejecuté la instalación de los paquetes necesarios desde el fichero `utils/requirements.txt`.

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

En la parte de frontend, configuré los enlaces simbólicos que Webpack necesita para localizar los componentes de RoboticsAcademy, instalé las dependencias de Node y arrancé 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 ya puedo levantar la plataforma completa en local, probar cada cambio y comprobar al instante cómo funciona sin miedo a romper nada en producción. Esto me 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 se tienen que resolver algunos bugs que afectan a la integración de gráficas desarrolladas

con Dash que utiliza la plataforma. Por este motivo, durante el desarrollo se optó por utilizar Flask 2.2 como entorno alternativo, lo que ha permitido probar y visualizar las gráficas sin depender del entorno principal.

Para desarrollar estos dashboard, 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` 2.1.3, lo que permite traer los datos directamente al entorno de trabajo.

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

```
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() > 0ORDER BY total_duration DESC;
```

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

Una vez obtenemos los datos, estos se procesan utilizando la biblioteca `Pandas` 2.1.1. Esto permite dejar los datos bien preparados antes de mostrarlos en una gráfica, permitiendo agruparlos, ordenarlos o filtrarlos fácilmente para que la visualización sea clara y ordenada.

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` 2.1.2, que ha permitido construir dashboards interactivos adaptados a los distintos análisis planteados. A lo largo del desarrollo, se ha aprovechado 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 opciones). Estos elementos permiten al usuario interactuar con la aplicación, por ejemplo, en el dashboard 4.1, el código 3.3 permite escribir un nombre de usuario para filtrar los datos.

```
# 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.

```
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 permitiendo que se actualicen al instante sin necesidad de recargar la página.

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 ha tenido en cuenta el diseño visual, asegurando que las gráficas sean claras y fáciles de leer. Para lograr esto, se han ajus-

```
@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.

tado 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.

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 han incorporado 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 para poder visualizar mejor las diferencias.

Como ejemplo, la Listing 3.6 muestra cómo se añaden leyendas y formas al gráfico.

```
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.

Todo ello se integra en un app creada por Flask de la que hablaré más cuando tenga le

versión final.

Además del diseño de cada gráfica, se ha implementado una hoja de estilos CSS que determina la apariencia general de la aplicación.

3.4 Discriminación de genero de usuarios

La base de datos actual de Unibotics dispone de multitud de variables como vimos en 3.1, sin embargo, no hay un registro que clasifique el género de los usuarios. Este aspecto no se consideró en el diseño inicial del programa, pero la idea es implementarlo en futuras etapas para que los nuevos usuarios sean capaces de proporcionar dicha información de manera voluntaria.

Esta modernización, además de ser necesaria para poder realizar un mejor análisis del comportamiento de los datos, consigue aportar una metodología más inclusiva que refuerza la imagen de Unibotics.

Gracias a estos ajustes, podemos identificar y cambiar con mayor facilidad las desigualdades que se dan entre hombres y mujeres, llegando así a analizar las diferencias y necesidades de cada grupo sin caer en sesgos ni estereotipos.

Por otro lado, contar con más 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 equivocadas si no se examina el contexto completo de los datos.

Para disponer de la variable género, he desarrollado 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 tiene limitaciones, ya que está basado en patrones de nombres que no reflejan con precisión la identidad de género de todos los usuarios, pero sí nos resulta útil, pues nos proporciona una muestra lo suficientemente grande para trabajar.

El script 3.7 utiliza las siguientes bibliotecas:

- `Psycopg2 2.1.3` : Para conectarse a la base de datos PostgreSQL y realizar consultas SQL.
- `Pandas 2.1.1` : Para manejar y procesar los datos extraídos de la base de datos.

```
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 3.7: Script para estimar el género de los usuarios.

- `Gender_guesser[9]` : Esta biblioteca nos permite estimar el género basado en el nombre de un persona, usando un detector preentrenado.

El script 3.7 comienza conectándose a la base de datos PostgreSQL utilizando la biblioteca Psycopg2. Luego, crea una nueva columna llamada gender en la tabla common_user, pero solo la primera vez que se ejecuta, ya que en ejecuciones posteriores, este paso es omitido si la columna ya existe. A continuación, se extraen los usuarios cuyos géneros aún no están definidos, es decir, aquellos que tienen el valor NULL o unknown en el campo de género. Utilizando la biblioteca gender_guesser, el script estima el género de los usuarios basándose en su primer nombre. Además, se asegura de corregir posibles errores relacionados con las mayúsculas o minúsculas en los nombres, capitalizando correctamente la primera letra del primer nombre para evitar errores de detección. Finalmente, actualiza la base de datos con los géneros estimados para estos usuarios. De este modo, el dato de género ahora está disponible para futuros análisis, permitiendo incorporar esta variable 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 Plotly para la visualización gráfica y la información se extrae directamente desde la base de datos PostgreSQL mediante consultas SQL específicas.

En los siguientes apartados expondré de forma individual cada uno de los dashboard desarrollados, explicando su propósito, los datos que representa, el porqué ese tipo de visualización y las diferentes conclusiones que pueden obtenerse sobre el comportamiento de los usuarios.

La Tabla 4.1 presenta las tablas SQL de las que se extrae la información utilizada en los dashboards [1].

[1]La base de datos de Unibotics contiene mas tablas como vimos en 3.1, pero estas son las que uso para realizar los dashboards.

Tabla 4.1: 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

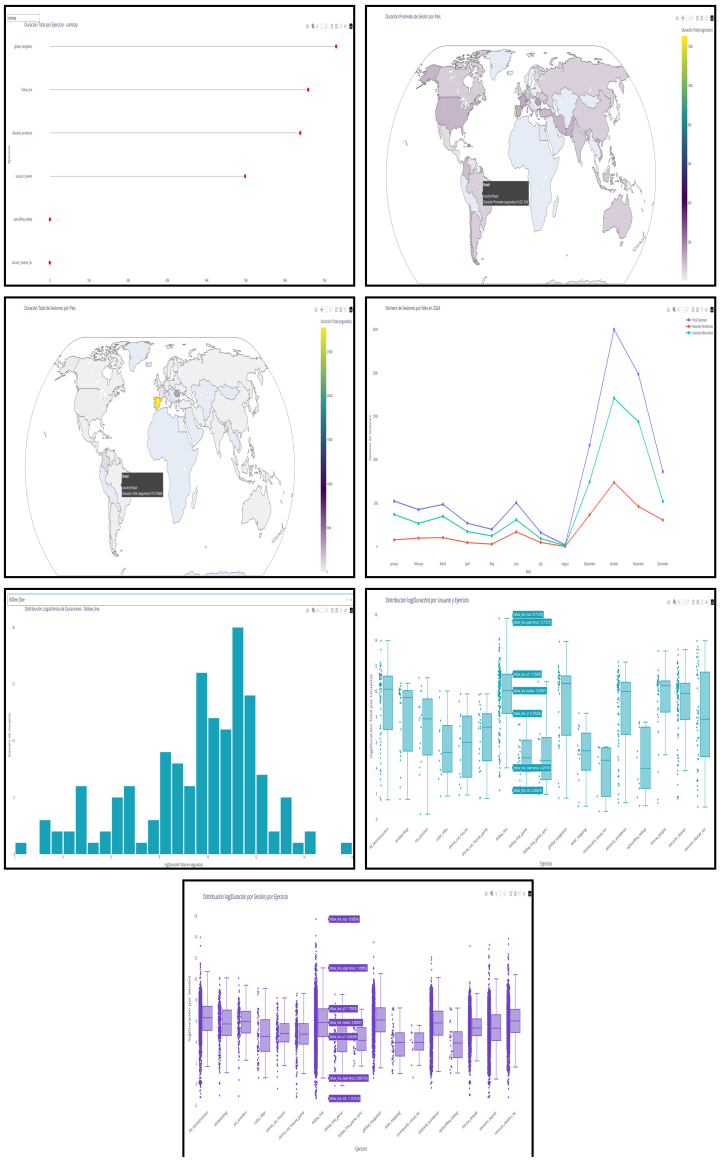

Figura 4.1: Resumen visual de los dashboards

4.1 dashboard 1A

El dashboard 1A nos permite analizar la duración total que un usuario ha dedicado a cada ejercicio de la plataforma.

Para poder 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 input.

La consulta SQL 4.8 aprovecha la funcionalidad de agrupación y agregación de SQL: se filtran los registros por el usuario seleccionado y luego se agrupan por el identificador de ejercicio, calculando la suma de todos los tiempos asociados,la información se obtiene de la tabla log_session.

```sql
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 4.8: 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.

Desde el punto de vista del usuario, el usuario ve un dashboard y un cuadro de texto 4.2 donde poder escribir el nombre del usuario que deseas visualizar su duración total por ejercicio, una vez escrito un nombre de usuario carga el dashboard con la información de ese usuario.

El dashboard 4.2 presenta los datos mediante un gráfico horizontal de líneas con puntos, también conocido como gráfico tipo Lollipop . Este tipo de visualización es una variación de un diagrama de barras, la barra tradicional se transforma en una línea delgada y el valor se resalta con un marcador circular al final.

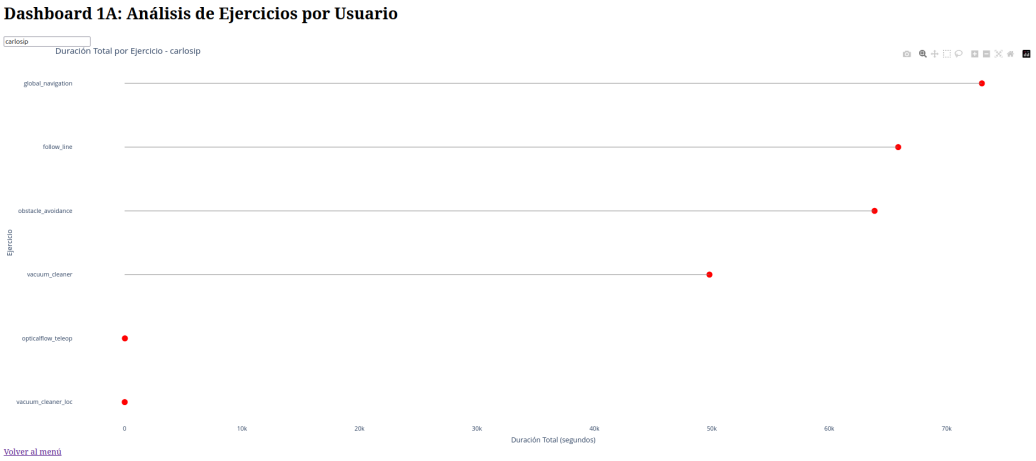

Figura 4.2: Dash 1a.

Elegí un gráfico del tipo Lollipop donde cada ejercicio aparece en función del tiempo total que el usuario le dedica, porque de esa manera es muy fácil ver de un vistazo en cuáles actividades pasa más tiempo. Al ordenar los ejercicios de mayor a menor duración, no estamos siguiendo el orden alfabético ni el listado original, sino que dejamos claro cuáles son realmente prioritarios para cada usuario. Esto hace que, nada más mirar el gráfico, se aprecie enseguida que, por ejemplo, el ejercicio "global_navigation" ocupa la mayor parte de la sesión, mientras que el ejercicio "opticalflow_teleop" apenas tiene unos minutos, sin necesidad de fijarse en cada número. De este modo, cuando hay varias tareas con duraciones muy distintas, facilita compararlas sin confusiones ni búsquedas innecesarias.

Gracias a ello podemos obtener 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 cierto ejercicio podría indicar que el usuario tuvo dificultades significativas con ese ejercicio o que requirió múltiples intentos prolongados para completarlo, también podría significar que el ejercicio era muy extenso o el usuario dedicó tiempo extra explorando más allá de lo mínimo requerido. Por el contrario, ejercicios con duraciones muy bajas pueden sugerir que el usuario los completó rápidamente o incluso que los abandonó pronto. En ambos casos, los extremos en la distribución de tiempos señalan ejercicios que merecen una atención particular al evaluar el progreso del usuario.

- **Distribución del tiempo de aprendizaje:** El conjunto de todas las duraciones permite ver cómo el usuario ha distribuido su tiempo de aprendizaje en la plataforma. Un alumno con un perfil equilibrado mostraría barras de duraciones relativamente simi-

lares entre ejercicios, mientras que un perfil más desequilibrado tendría unos pocos ejercicios dominando la mayor parte del tiempo total.

4.2 dashboard 1C

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 4.9 utilizada 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 4.9: Consulta SQL para obtener duración total y número de sesiones por país.

A partir de la consulta 4.9 se genera un DataFrame de Pandas con el país y su respectiva duración promedio de sesión, que luego se utiliza para crear el gráfico 4.3. 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.

El dashboard 4.3 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 4.3, se emplea una paleta de colores continua (aclarar que color dejo) personalizada para resaltar visualmente los valores extremos.

Elegí un mapa coroplético porque este tipo de visualización muestra de forma muy clara cómo se distribuye la duración total de 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 leer tablas o listas. El degradado

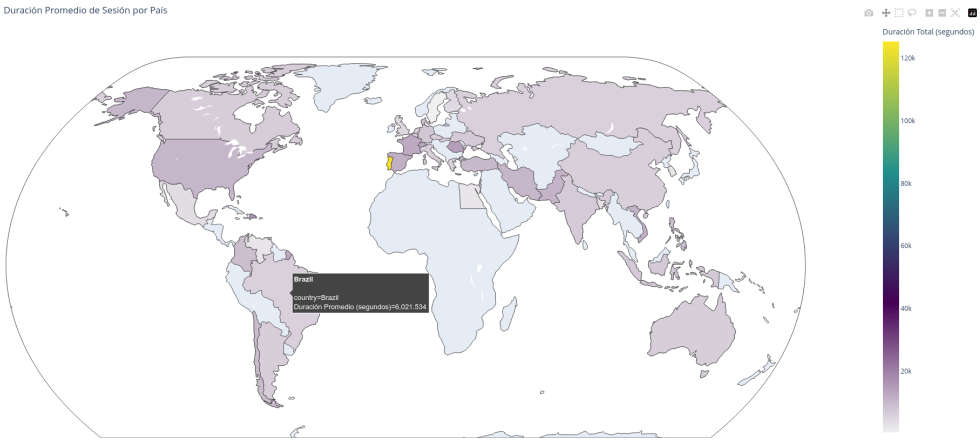

Figura 4.3: Dash 1c.

de colores refuerza la percepción de diferencias entre países y también facilita ver patrones globales.

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

- **Alta duración promedio en España:** Esto indica un uso intensivo y sesiones posiblemente más prolongadas en comparación con otros países, donde el uso puede ser más breve o esporádico.

- **Bajas duraciones promedio en ciertos países:** 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, al tiempo que señala aquellas 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 dashboard 2A

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.

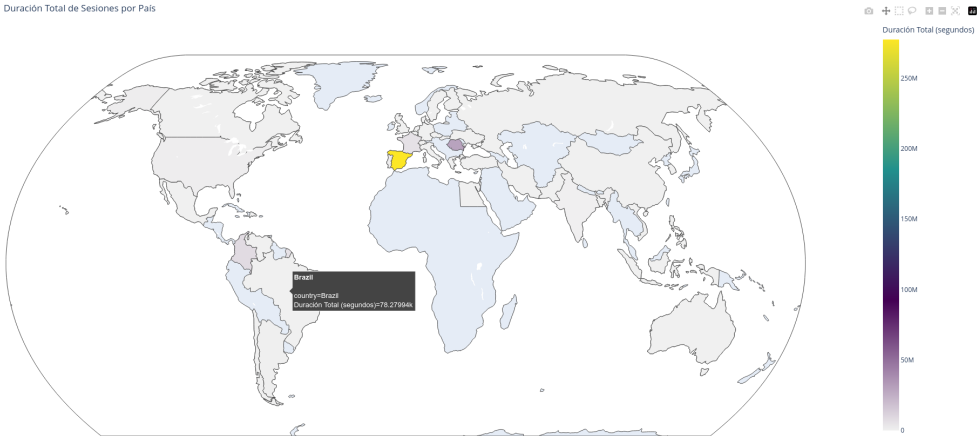

Figura 4.4: Dash 2a.

La información utilizada en este dashboard se obtiene desde la tabla log_session, la consulta SQL 4.10 que se utiliza agrupa los datos por país y calcula la suma total de la duración de todas las sesiones correspondientes a cada país.

```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 4.10: Consulta SQL para obtener la duración total por país en sesiones.

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.

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

El dashboard 4.4 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 4.4 , se emplea una paleta de colores continua (aclarar que color dejo) personalizada para resaltar visualmente los valores extremos.

Elegí 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 tener que depender de leer tablas o listas.

Del análisis del mapa 4.4, 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 ayuda a visualizar en qué países la plataforma tiene mayor presencia, facilitando decisiones sobre campañas, traducción de contenidos o inversiones para mejorar el servicio. Además, señala qué zonas podrían beneficiarse más de nuevas colaboraciones educativas.

4.4 dashboard 2C

Este dashboard muestra la evolución mensual del número de sesiones iniciadas en Unibotics a lo largo del año, para ello tiene 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 4.11 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.

El dashboard 4.5 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).

```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 4.11: Consulta SQL para obtener sesiones mensuales totales y por género en 2024.

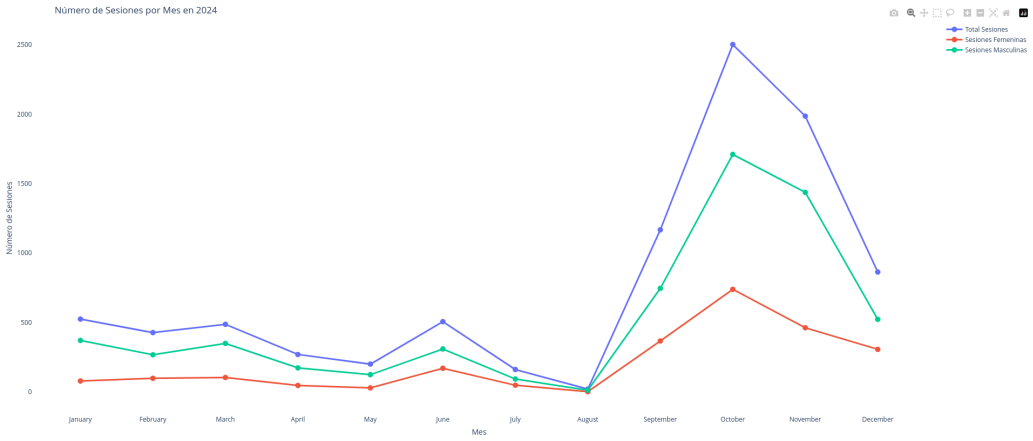

Figura 4.5: Dash 2c.

- 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.

Escogí un gráfico de líneas con tres trazos (total de sesiones, sesiones femeninas y sesiones masculinas) porque así se puede ver mes a mes cómo crece o baja 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 suben 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 mirar varios gráficos por separado, de modo que se aprecia de inmediato cómo se comporta la plataforma a lo largo de 2024.

Explicar conclusiones porque de aqui mas alla de que en verano nadie hace nada no se que mas decir.

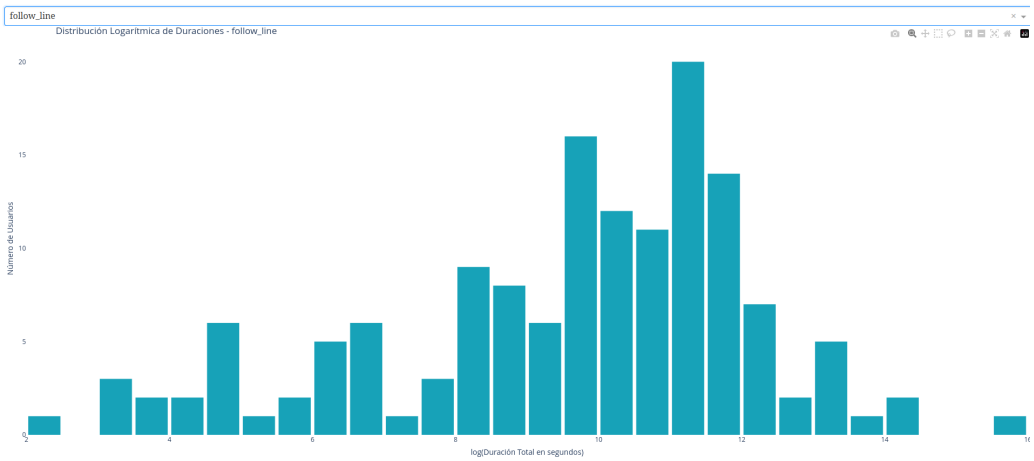

Figura 4.6: Dash 3a.

4.5 dashboard 3A

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 de ejercicios que permite seleccionar un ejercicio concreto mediante un desplegable. Una vez seleccionado, se ejecuta la consulta SQL 4.12 que suma el tiempo total que cada usuario ha dedicado a ese ejercicio.

```
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 4.12: Consulta SQL para obtener la duración total por usuario en un ejercicio específico.

Después de obtener los datos, se aplica 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.

El dashboard 4.6 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 muestra cuántos usuarios cayeron en ese rango de duraciones. Como las duraciones se extienden mucho, apliqué la función logaritmo en el eje X para comprimir la escala, de ese modo, en lugar de tener barras juntas en valores muy bajos y un algunos casos aislados en el extremo derecho, se obtienen intervalos más balanceados

Elegí que este gráfico fuera un histograma porque quería mostrar cómo se distribuyen las duraciones de todos los usuarios en un ejercicio concreto, y al agrupar esos tiempos en intervalos resulta muy fácil ver dónde se concentra la mayoría de la gente.

Al observar el histograma, podemos identificar de un vistazo si la mayoría de los usuarios se sitúa en un tiempo intermedio o, por el contrario, si hay grupos que destacan con duraciones muy largas o muy cortas, apreciándose de forma inmediata la forma de la distribución.

Este análisis del histograma ayuda a comprender con mayor profundidad cómo se comportan los usuarios al realizar un ejercicio determinado. A partir de esta información, se pueden destacar 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:** Indican que el ejercicio puede ser 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 los usuarios.
- **Utilidad para los profesores:** Facilita detectar si el ejercicio está bien equilibrado en dificultad o si requiere ajustes para mejorar la experiencia de aprendizaje.

4.6 dashboard 3B

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

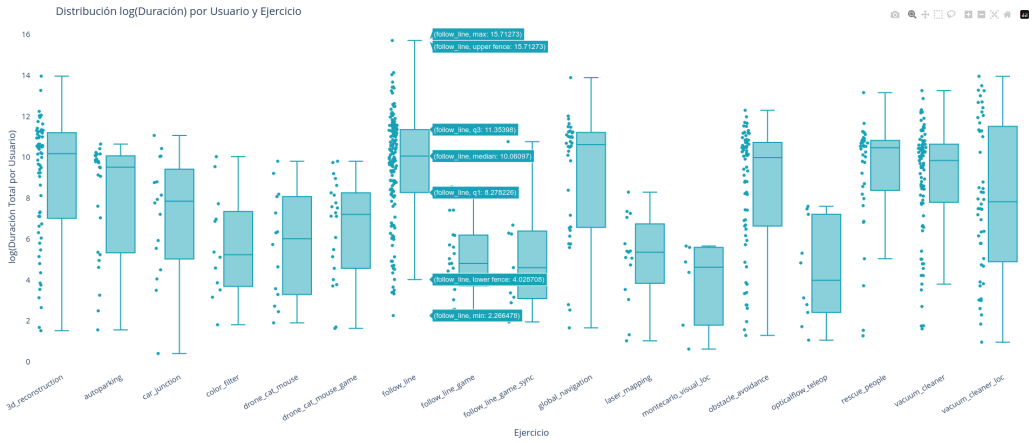

Figura 4.7: Dash 3b.

Los datos se obtienen de la tabla log_exercises, con la consulta 4.13 se agrupa 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.

```
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 4.13: Consulta SQL para obtener duración total por ejercicio y usuario.

Una vez extraídos los datos, se calcula 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.

En este dashboard 4.7 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 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 permite comparar rápidamente el comportamiento de los usuarios en cada ejercicio. A partir de los datos, se pueden destacar 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 dashboard 3C

El dashboard 4.8 es muy similar al 4.7, 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 se analiza la duración de cada sesión individual.

Mientras que en 4.7 sumábamos todas las sesiones de un usuario para un ejercicio determinado, en este dashboard se estudia cada sesión por separado, sin agruparlas. Además, también se aplica 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 se utiliza la consulta SQL 4.14 que selecciona, para cada registro de sesión, el nombre del ejercicio y la duración de esa sesión concreta.

El objetivo del dashboard 4.8 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

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

Código 4.14: Consulta SQL para obtener duración y ejercicio ordenados por ejercicio.

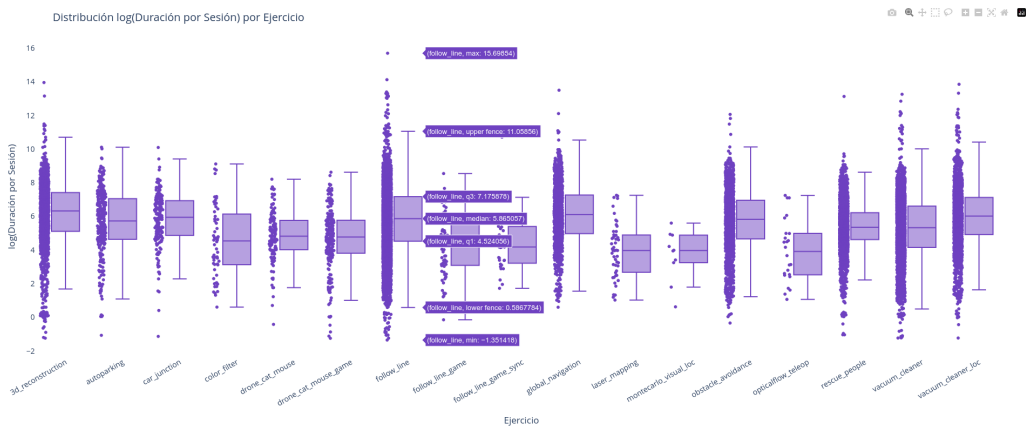

Figura 4.8: Dash 3c.

4.7, donde sólo veíamos el esfuerzo acumulado de cada usuario. Aquí podemos detectar, por ejemplo, si un ejercicio tiene sesiones generalmente cortas o si hay mucha variación entre los intentos de los 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, puede ayudar 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 dadas.

Capítulo 5

Conclusiones y trabajos futuros

5.1 Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

Y si has llegado hasta aquí, siempre es bueno pasarle el corrector ortográfico, que las erratas quedan fatal en la memoria final. Para eso, en Linux tenemos aspell, que se ejecuta de la siguiente manera desde la línea de *shell*:

```
aspell --lang=es_ES -c memoria.tex
```

5.2 Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a
2. b

5.3 Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. Aquí viene uno.
2. Aquí viene otro.

5.4 Trabajos futuros

Ningún proyecto ni software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

Siglas

IDE Integrated Development Enviroment (Entorno de Desarrollo Integrado). 7, 8

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.15: Consulta SQL para listar usuarios que han ejecutado `rescue_people`.

Con la consulta A.15 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.16: Consulta SQL para calcular la duración media de cada ejercicio por usuario.

Con la consulta A.16 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.17: Consulta SQL para contar el total de sesiones de ejercicio por usuario.

Con la consulta A.17 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.18: Consulta SQL para obtener la última fecha de sesión de cada usuario.

Con la consulta A.18 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.