



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE  
TELECOMUNICACIÓN

GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES  
Y MULTIMEDIA

TRABAJO FIN DE GRADO

**Analíticas automáticas en una plataforma web de  
robótica educativa**

Autora : Claudia Álvarez Bravo

Tutor : Dr. José María Cañas Plaza

Co-tutor : Dr. David Roldán Álvarez

Curso Académico 2021/2022

## **Agradecimientos**

Este trabajo va dedicado principalmente a toda mi familia, la cual me ha apoyado durante todo este proceso, en especial a mi hermano César, que siempre me ha animado a continuar con mis sueños y a sobrepasar mis límites.

También agradezco a mis tutores del TFG, José María Cañas y David Roldán Álvarez, por el esfuerzo y el seguimiento de mi TFG.

Y por último agradecer a todos los profesores y compañeros que he tenido durante la carrera los cuales me han enseñado lo maravilloso y apasionante que es la ingeniería.

## Resumen

Conocer las interacciones de los usuarios en una plataforma Web se ha convertido en un aspecto fundamental para poder analizar el comportamiento de éstos con idea de mejorar los servicios ofrecidos por la plataforma. La monitorización y posterior análisis de esta información permite lograr los objetivos deseados, mejorar la experiencia del usuario e incluso hacer una optimización de nuestra plataforma web. Para realizar estas tareas es necesario recoger el máximo de datos posibles para poder hacer un mejor análisis.

Este Trabajo de Fin de Grado (TFG) muestra el proceso seguido para la recogida de la información de los usuarios en la plataforma web educativa Unibotics (realizando un monitoreo automático) y cómo se han visualizado estos datos masivos para poder analizarlos. Unibotics está dirigido a estudiantes universitarios donde aprenden sobre robótica y a programar en Python a través de la realización de varios ejercicios. Estos datos recogidos permitirán a los administradores saber el comportamiento de los estudiantes y a la vez, los estudiantes podrán ver sus progresos en cada ejercicio.

Para la realización de esta tarea se han combinado diferentes tecnologías que han permitido tanto el almacenamiento de la información como su posterior visualización automática. En este TFG la base de datos utilizada ha sido Elasticsearch y se ha elegido Dash como entorno para la creación de las visualizaciones automáticas. Elasticsearch forma parte del ELK Stack, que tiene su propio visualizador, Kibana, pero se decidió utilizar Dash debido a su facilidad de uso, configuración y administración.

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Tecnologías web . . . . .	1
1.1.1. Tecnologías web del lado del cliente . . . . .	3
1.1.2. Tecnologías web del lado del servidor . . . . .	4
1.2. Plataforma web de programación . . . . .	5
1.3. Robótica y software de robot . . . . .	8
1.4. Plataformas de programación de robótica . . . . .	10
1.5. Unibotics . . . . .	12
1.6. Estructura del documento . . . . .	14
<b>2. Objetivos y Metodología del Trabajo</b>	<b>15</b>
2.1. Objetivos . . . . .	15
2.2. Metodología . . . . .	16
2.3. Plan de Trabajo . . . . .	17
2.4. Motivación . . . . .	17
<b>3. Herramientas utilizadas</b>	<b>19</b>
3.1. Tecnologías web . . . . .	19
3.1.1. HTML . . . . .	19
3.1.2. CSS . . . . .	22
3.1.3. JavaScript . . . . .	23
3.1.4. Django . . . . .	24
3.1.5. SQL . . . . .	26
3.2. Recogida y grabación de datos . . . . .	28

3.3. Visualización de estadísticas automática . . . . .	30
<b>4. Analíticas de Elasticsearch y Dash</b>	<b>31</b>
4.1. Recogida de sondas . . . . .	31
4.2. Visualización de la información . . . . .	36
4.3. Validación experimental . . . . .	39
<b>5. Conclusiones y trabajos futuros</b>	<b>52</b>
5.1. Conclusiones finales . . . . .	52
5.2. Competencias adquiridas . . . . .	53
5.3. Trabajos futuros . . . . .	53

# Índice de figuras

1.1. Primera página web . . . . .	2
1.2. Arquitectura Cliente-Servidor . . . . .	2
1.3. Plataforma Scratch . . . . .	5
1.4. Snap! . . . . .	6
1.5. Robocode . . . . .	7
1.6. CodeCombat . . . . .	7
1.7. Esquema ROS . . . . .	9
1.8. Curso en TheConstruct . . . . .	11
1.9. Curso en Riders.ai . . . . .	11
1.10. Infraestructura de Unibotics . . . . .	13
1.11. Unibotics en la actualidad . . . . .	14
3.1. Estructura HTML . . . . .	20
3.2. Estructura de un elemento HTML . . . . .	21
3.3. Sintaxis CSS . . . . .	22
3.4. Estructura de Django . . . . .	24
3.5. Esquema funcionamiento SQL . . . . .	26
3.6. Sentencia SQL . . . . .	27
3.7. Arquitectura Elasticsearch . . . . .	29
4.1. Elasticsearch dummy . . . . .	35
4.2. Menú de un administrador en Dash . . . . .	36
4.3. Filtros utilizados en Dash . . . . .	38
4.4. Gráfico sesiones totales por día . . . . .	39
4.5. Gráfico sesiones únicas por día . . . . .	40

4.6. Gráfico de tiempo en Unibotics . . . . .	41
4.7. Gráfico de tiempo en Unibotics de un usuario . . . . .	42
4.8. Histograma de las duraciones de sesiones . . . . .	42
4.9. Gráfico de navegadores . . . . .	43
4.10. Gráfico de sistemas operativos . . . . .	44
4.11. Mapa geográfico de sesiones . . . . .	45
4.12. Mapas de calor sesiones . . . . .	46
4.13. Gráficas relativas a los usuarios . . . . .	47
4.14. Gráfica registro de usuarios acumulado . . . . .	48
4.15. Gráfica de usuarios activos cada día . . . . .	48
4.16. Histograma del ejercicio follow_line de un grupo de usuarios . . . . .	49
4.17. Gráfica de puntuación de estilo . . . . .	50
4.18. Gráfica de puntuación de eficacia . . . . .	51

# Capítulo 1

## Introducción

La educación ha cambiado a lo largo de los años, actualmente es difícil no encontrar un aula donde las tecnologías web están muy presentes y más en el ambiente universitario. Para poder realizar una plataforma web competitiva en el mercado y además atrayente al usuario es imprescindible hacer una recogida de la interactividad que el usuario tiene con la plataforma web, guardar esos datos y después visualizarlos.

En este capítulo se va a hablar de los elementos clave en los que se basa este trabajo que son las tecnologías web, y la robótica educativa. También se introducirá en qué consiste un monitoreo y análisis de una página web.

### 1.1. Tecnologías web

Las tecnologías de internet se remontan a 1970 donde el departamento de defensa de Estados Unidos creó una red descentralizada la cual pudiera aguantar ataques nucleares. En 1990, Tim Berners-Lee trabajador del CERN<sup>1</sup> originó un protocolo de comunicación basado en hipertextos (HTTP) donde científicos compartían documentos, gracias a eso Berners-Lee junto a su equipo desarrolló el lenguaje de marcado HTML<sup>2</sup> y el sistema de direcciones de web URL<sup>3</sup>, ese mismo año creó la primera página web (Figura 1.1) dando inicio a la WWW<sup>4</sup>. Al cabo de

---

<sup>1</sup>Organización Europea para la Investigación Nuclear

<sup>2</sup>HyperText Markup Language

<sup>3</sup>Uniform Resource Locator

<sup>4</sup>World Wide Web



los años se fueron desarrollando los diferentes navegadores hasta los que se conocen hoy en día.

### World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#), [Policy](#), November's [W3 news](#), [Frequently Asked Questions](#).

[What's out there?](#)

Pointers to the world's online information, [subjects](#), [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#), [X11 Viola](#), [NeXTStep](#), [Servers](#), [Tools](#), [Mail robot](#), [Library](#))

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help?](#)

If you would like to support the web..

[Getting code](#)

Getting the code by [anonymous FTP](#), etc.

Figura 1.1: Primera página web

En la actualidad las principales ventajas de las tecnologías web son que se pueden utilizar en cualquier dispositivo, el usuario no necesita instalar nada más allá de un navegador web y su usabilidad es sencilla, pero dependen de Internet[1].

Las páginas Web modernas utilizan el modelo Cliente-Servidor, en el que el cliente hace peticiones al servidor esperando una respuesta de éste. Un mismo cliente puede estar conectado con varios servidores y a la vez interactúa con el usuario final normalmente a través de una interfaz gráfica. La parte del servidor se encarga de recibir las peticiones, analizarlas y procesarlas para enviar una respuesta. En las siguientes subsecciones, describimos algunas de las tecnologías más populares utilizadas en el desarrollo de páginas Web.

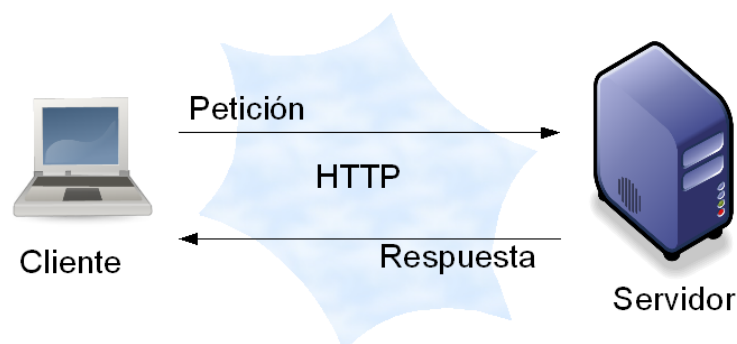


Figura 1.2: Arquitectura Cliente-Servidor

### 1.1.1. Tecnologías web del lado del cliente

Las tecnologías del lado del cliente suelen recibir el nombre de *frontend*, son las que interactúan con el usuario, las más importantes son:

- **HTML:** Lenguaje de marcado que se encarga de dar la estructura a la página web. HTML utiliza etiquetas para mostrar los diferentes contenidos. La última versión es HTML5 donde se introducen las etiquetas de audio y vídeo.
- **CSS**<sup>5</sup>: Lenguaje de estilo que se encarga del diseño gráfico, de la apariencia de las páginas web. La última versión es CSS3
- **JavaScript:** Lenguaje de programación interpretado que permite la ejecución de código orientado a eventos (por ejemplo, pulsar un botón). Puede actuar sobre el navegador a través de objetos integrados, es decir puede manipular el HTML.

---

<sup>5</sup>Cascading Style Sheets

### 1.1.2. Tecnologías web del lado del servidor

Las tecnologías web del lado del servidor, también llamadas *backend*, son las encargadas de procesar las peticiones del cliente y enviar una respuesta en forma de páginas HTML. Normalmente se utilizan *frameworks* que son herramientas que te permiten desarrollar una aplicación web de forma ágil y sencilla a partir de librerías o funciones ya creadas. Algunos de estos *frameworks* son:

- **Django:** Está escrito en código Python. Django hace uso de plantillas permitiendo hacer páginas web más rápidas y sencillas. Gracias a su ORM <sup>6</sup> no es necesario saberse el manejo de base de datos ya que este traduce directamente el código de Python al lenguaje estándar de las peticiones a bases de datos, SQL.
- **Node.js:** Permite ejecutar código JavaScript fuera del navegador y está construido con el motor de JavaScript V8 de Chrome. Node.js utiliza un modelo de entrada y salida no bloqueante, no espera a que los procesos finalicen.
- **Symfony:** Utiliza de lenguaje de programación PHP <sup>7</sup>. Está basado en el patrón Modelo Vista Controlador (MVC). Symfony es muy flexible permitiéndote instalar únicamente lo que se necesite en vez del *framework* completo.
- **Spring Boot:** Desarrollado para aplicaciones programadas en Java, surgió debido a que el Spring Framework era difícil de configurar.
- **Laravel:** Framework de PHP. En comparación con otros *frameworks* de PHP su curva de aprendizaje es más baja. Es flexible y adaptable, no solamente debido al MVC sino que también propone utilizar *routes with clousures*, a causa de lo cual reduce código.[2]

---

<sup>6</sup>Object Relational Mapping

<sup>7</sup>Hypertext Preprocessor

### 1.2. Plataforma web de programación

En la actualidad, gracias a Internet se tiene al alcance plataformas web donde se enseña a programar de una forma interactiva. Una de las ventajas de aprender programación en una plataforma web es que puedes hacer desde la comodidad de tu casa y una gran parte son gratuitas. Pueden ser utilizadas por usuarios de todos los niveles. A continuación se nombra algunos ejemplos de estas plataformas:

- **Scratch**<sup>8</sup>: Es un entorno de programación desarrollado por el MIT<sup>9</sup>. Se puede descargar en local o se puede utilizar la plataforma de forma *online*. Utiliza un lenguaje de programación visual llamado igual que la plataformas, Scratch, el cual está basado en bloques permitiendo crear programas arrastrando y soltando para agrupar dichos bloques como un puzzle. Cada bloque puede significar un evento, movimiento, un operador o una variable, entre otras. Con esta plataforma, por ejemplo, se pueden crear juegos, historias interactivas, música de manera sencilla y dinámica para el usuario. Las edades que tienen los usuarios que utilizan la plataforma esta entre los 8 y 16 años, aunque hay adultos que también la utilizan[3].

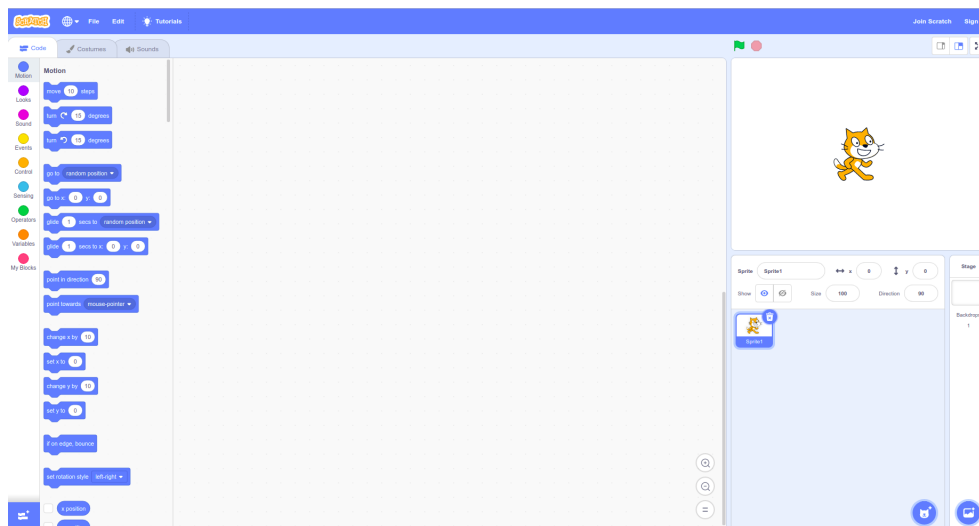


Figura 1.3: Plataforma Scratch

---

<sup>8</sup><https://scratch.mit.edu/>

<sup>9</sup>Massachusetts Institute of Technology

- **Snap!**<sup>10</sup>: Está basada en el código de programación de Scratch, creada en la Universidad de Berkeley. Esta aplicación la puedes utilizar tanto *online* como descargándola y pudiendo utilizarla sin necesidad de Internet. La principal diferencia con Scratch es que te permite crear tus propios bloques utilizando JavaScript y así poder formar tu librería propia, también se puede crear listas avanzadas donde se puede almacenar cualquier tipo de dato incluso otras listas. Esta aplicación web permite aprender a programar de una manera más visual y evitando los errores de sintaxis que se podrían cometer. Snap tiene una gran comunidad de usuarios que suben sus proyectos donde se puede aprender de ellos[4].

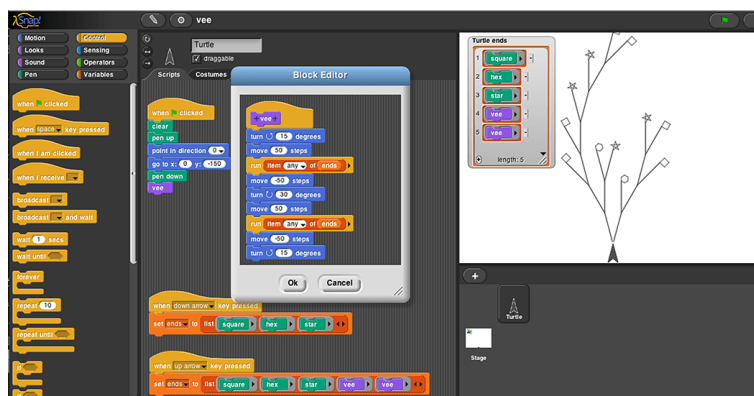


Figura 1.4: Snap!

- **Robocode**<sup>11</sup>: Es un videojuego multiplataforma que consiste en la creación de un tanque robot el cual tendrá que atacar y esquivar otros tanques para no ser destruido. No es necesario descargar nada aparte de lo que viene en Robocode. Está dirigido para aprender Java y .NET. Nos podemos encontrar ligas donde las batallas de los robots transcurren en tiempo real, éstas están divididas en diferentes categorías dependiendo del tamaño de código efectivo en bytes para que sean competiciones más justas. Se puede programar las tres partes del tanque que son: el cuerpo que se encarga de mover el tanque, el radar que detecta a los adversarios y el cañón que se utiliza para apuntar y disparar. La aplicación dispone de un editor de texto, un *debugger* y un compilador, todos los robots que se vayan desarrollando se podrán guardar para futuras batallas[4].

<sup>10</sup><https://snap.berkeley.edu/>

<sup>11</sup><https://robocode.sourceforge.io/>

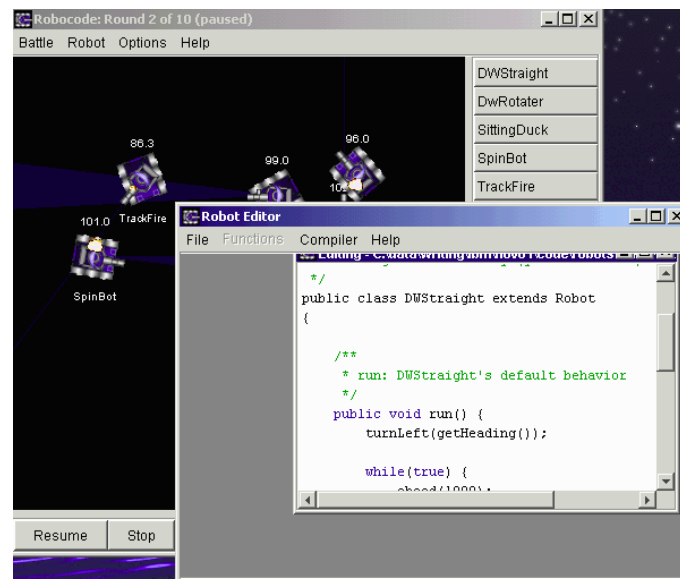


Figura 1.5: Robocode

- **CodeCombat**<sup>12</sup> Es una página web donde a través de un personaje se enseña a programar en diferentes lenguajes como Python, JavaScript, C++, entre otros. El objetivo del juego es ir superando los diferentes niveles, los cuales van aumentando de dificultad. Esta compuesto por 110 niveles gratuitos con opción de jugar más niveles si se paga, además pagando se pueden desbloquear diferentes héroes, aprender a programar juegos y páginas web[4].

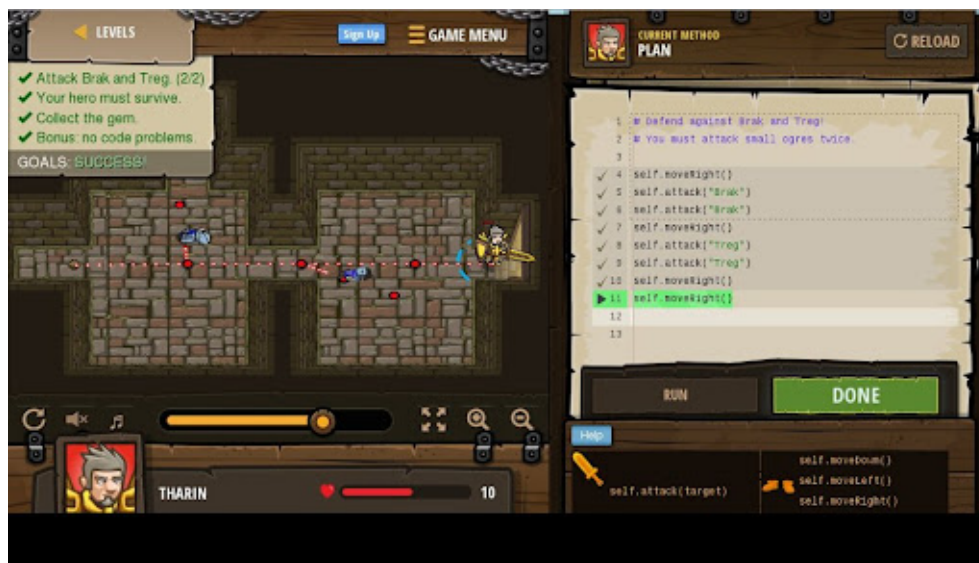


Figura 1.6: CodeCombat

<sup>12</sup><https://codecombat.com/>

## 1.3. Robótica y software de robot

La robótica es la ciencia o rama tecnológica encargada del estudio, diseño, creación y aplicación de los robots. En la actualidad existen muchas formas de estudiar robótica, puedes estudiarla tanto en universidades como en cursos. Existen diferentes herramientas para poder trabajar con robots, como los sistemas operativos robóticos. Encontramos varios ejemplos de sistemas operativos como RoboComp<sup>13</sup> o Player<sup>14</sup>, pero el más conocido y usado es ROS<sup>15</sup> (*Robot Operating System*).

ROS es un *framework* flexible para escribir *software* de robots. Las principales ventajas son que es de código abierto, contando con una comunidad global para mejorar el *software*. Además es una herramienta multiplataforma e independiente de la administración del *backend* y las interfaces de usuario[5].

El funcionamiento de ROS esta basado en nodos, éstos son procesos ejecutables que realizan una tarea simple. Las tareas de los nodos pueden ser controlar la velocidad, el motor de las rueda o la localización. Los nodos pueden enviar mensajes o recibir mensajes de otros nodos a través de los tópicos.

Los tópicos se encargan de transmitir el mensaje a todos los nodos que se han suscrito y reciben los mensajes que publican los nodos. Los nodos no saben que nodo a publicado ni que nodo se ha suscrito a un tópico. Los nodos también pueden ofrecer servicios que serán acciones que otro nodo puede pedir. El servidor que ofrece el servicio solo manda la respuesta cuando recibe una solicitud de otro nodo, haciendo la comunicación síncrona. En la figura 1.7 se muestra un esquema del funcionamiento de ROS[6].

Otro nodo importante es el nodo maestro que se encarga de registrar todas las suscripciones y publicaciones a los tópicos.

---

<sup>13</sup><https://robocomp.github.io/web/>

<sup>14</sup><http://playerstage.sourceforge.net/>

<sup>15</sup><https://www.ros.org/>



Figura 1.7: Esquema ROS

Para programar robots, otra herramienta potente son los simuladores de robots, que permiten emular a robots reales de forma virtual para poder trabajar con ellos. La simulación es importante para saber como se comportaría un robot en la realidad y saber si necesita cambios. Además una ventaja es que económicamente la construcción de un robot puede ser costoso, con un simulador se evitan estos gastos.

Existen varios simuladores como CoppeliaSim<sup>16</sup>, antiguamente llamado V-REP, cuenta con un entorno de desarrollo integrado y cada objeto o modelo se puede controlar vía ROS, un plugin, un script integrado, un cliente API remoto o una solución personalizada[7]. Webots<sup>17</sup> es otro simulador de código abierto y multiplataforma, ofrece bibliotecas con muestras de mundos, sensores o robots, entre otras características [8].

Entre todos los simuladores cabe destacar Gazebo<sup>18</sup>, el simulador utilizado en la plataforma Unibotics. Gazebo para su simulación dinámica utiliza cuatro motores diferentes: ODE<sup>19</sup> (simulación de cuerpos rígidos, que permite detectar y simular colisiones), Bullet (simulación de cuerpos rígidos y blandos, también detecta colisiones), DART<sup>20</sup> (simulación de la dinámica y cinemática de un robot) y Simbody (herramienta útil para simular articulaciones con restricciones y resuelve la segunda ley de Newton)[9].

---

<sup>16</sup><https://www.coppeliarobotics.com/>

<sup>17</sup><https://cyberbotics.com/>

<sup>18</sup><http://gazebo.org/>

<sup>19</sup>*open dynamics engine*

<sup>20</sup>*Dynamic Animation and Robotics Toolkit*



Gazebo representa gráficos 3D avanzados y realistas utilizando OGRE<sup>21</sup>, que facilita la renderización. Dispone de diferentes sensores con la opción de añadir ruido. Además se puede desarrollar complementos para los robots, los sensores o el ambiente. Gazebo ofrece varios robots o la posibilidad de crear uno propio[10].

## 1.4. Plataformas de programación de robótica

Una parte de la robótica es la programación de estos robots por lo que se han ido creando herramientas de aprendizaje sencillas y atrayentes para el usuario. En Internet se pueden encontrar diversas plataformas web las cuales te enseñan sobre programación y robótica, como es el caso de Unibotics. Entre ellas se encuentran:

- **TheConstruct**<sup>22</sup>: Plataforma web que enseña sobre robótica, ROS e inteligencia artificial. Tiene una versión gratuita en la que se ofrecen tres cursos: linux para robótica, python3 para robótica y C++ para robótica, si se quiere puedes acceder a todos los cursos con su versión de pago. Está desarrollado para que puedan utilizarlo tanto principiantes como profesionales. No requiere de la instalación de ROS. Además una de sus ventajas es que tiene una gran comunidad donde se puede establecer contactos y aprender nuevas formas de programar robots. TheConstruct cuenta con robots reales que se pueden alquilar por un determinado tiempo, dando la posibilidad de poder conectarse a ellos y programarlos desde cualquier lugar. Se encuentran cursos que una vez finalizados y presentando un proyecto, ofrecen certificados. Una vez se selecciona un curso, se tiene en la parte izquierda la teoría para realizar el curso. También tiene un interfaz de usuario dónde se escribe el código, un terminal para escribir comandos y una simulación del robot que se va a programar.

---

<sup>21</sup> *Object-Oriented Graphics Rendering Engine*

<sup>22</sup><https://www.theconstructsim.com/>

## 1.4. PLATAFORMAS DE PROGRAMACIÓN DE ROBÓTICA

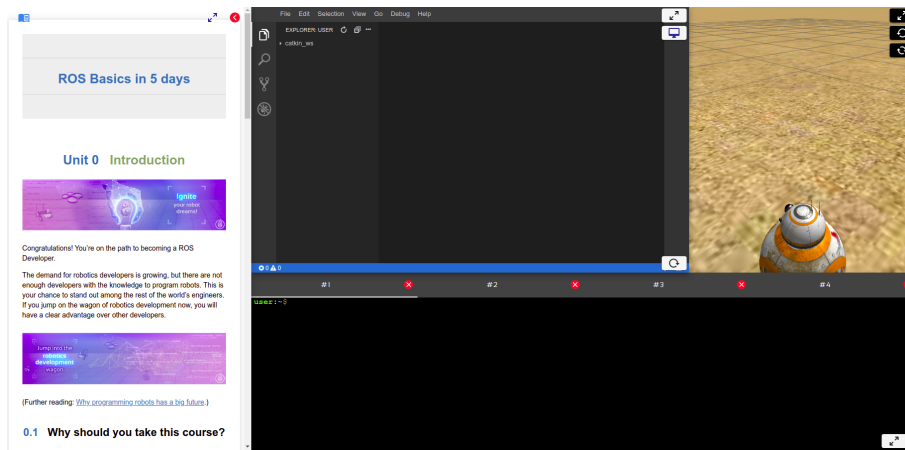


Figura 1.8: Curso en TheConstruct

- **Riders.ai**<sup>23</sup>: Plataforma robótica de simulación, educación y competencia basada en la nube desarrollada por Acrome Robotic Systems[11]. Es una plataforma de pago, solamente es gratis la primera lección del curso introducción a la robótica: parte 1. Consta de otros dos cursos, los cuales son la continuación del curso mencionado anteriormente, además una vez finalizados se obtiene un certificado. Los cursos te enseñan a programar en Python o C++ los robots. Las lecciones están formados por la teoría, el interfaz de usuario y el simulador Gazebo, todo ello en la misma pestaña como se muestra en la figura 1.9. Riders dispone de dos ligas para competir con otros programadores. Una liga consiste en drones voladores y otra sobre robots móviles.

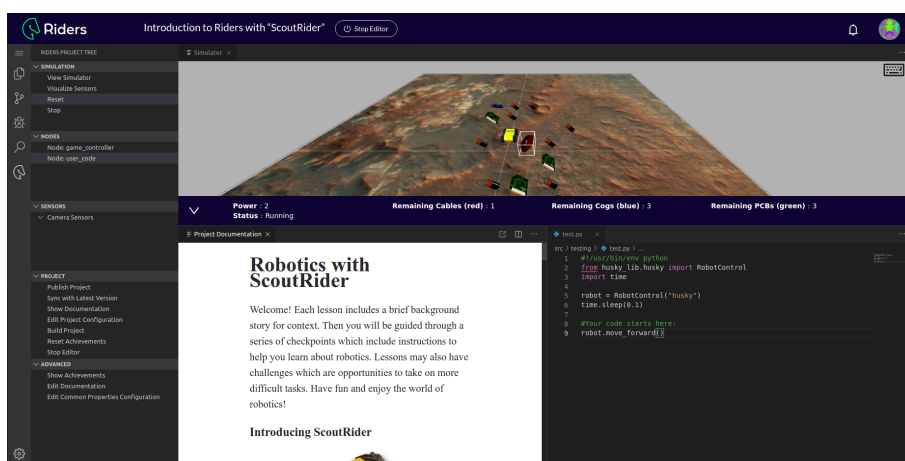


Figura 1.9: Curso en Riders.ai

<sup>23</sup><https://riders.ai/>

### 1.5. Unibotics

Unibotics<sup>24</sup> es una plataforma en línea de robótica educativa donde se desarrolla este Trabajo de Fin de Grado en la cual se enseña a programar de una manera llamativa y divertida para estudiantes universitarios. Unibotics proviene de otra plataforma que no está en línea, llamada Robotics Academy<sup>25</sup>. Ambas plataformas han sido creadas por la asociación de robótica e inteligencia artificial, JdeRobot.

La plataforma consta de varios ejercicios que se pueden dividir en ejercicios de **conducción autónoma** (*Follow Line, Obstacle avoidance, Global Navigation, Car Junction y Autoparking*), **robots de servicios** (*Vacuum Cleaner, Localized Vacuum Cleaner y Laser Mapping*), **drones** (*Drone Cat and Mouse y Rescue People*) y **visión artificial** (*3D Reconstruction, Color Filter, OpticalFlow Teleop y Montecarlo Visual Loc*).

Se puede programar los ejercicios directamente desde la web sin la necesidad de instalar la aplicación. En cada ejercicio, el usuario puede introducir código fuente, cargarlo en el cerebro de un robot simulado, ejecutarlo en simulación y visualizar el interfaz gráfico. Además se puede subir o guardar el código realizado y comprobar la eficacia y el estilo del código. Se utiliza el lenguaje Python para poder realizar los ejercicios. Unibotics utiliza una imagen de docker llamada RADI<sup>26</sup> (*Robots Academy Docker Image* donde están preinstaladas las dependencias y así no se tiene que descargar nada localmente. Está basada en ROS y Gazebo[12].

---

<sup>24</sup><https://unibotics.org/>

<sup>25</sup><http://jderobot.github.io/RoboticsAcademy/>

<sup>26</sup><https://hub.docker.com/r/jderobot/robotics-academy/>

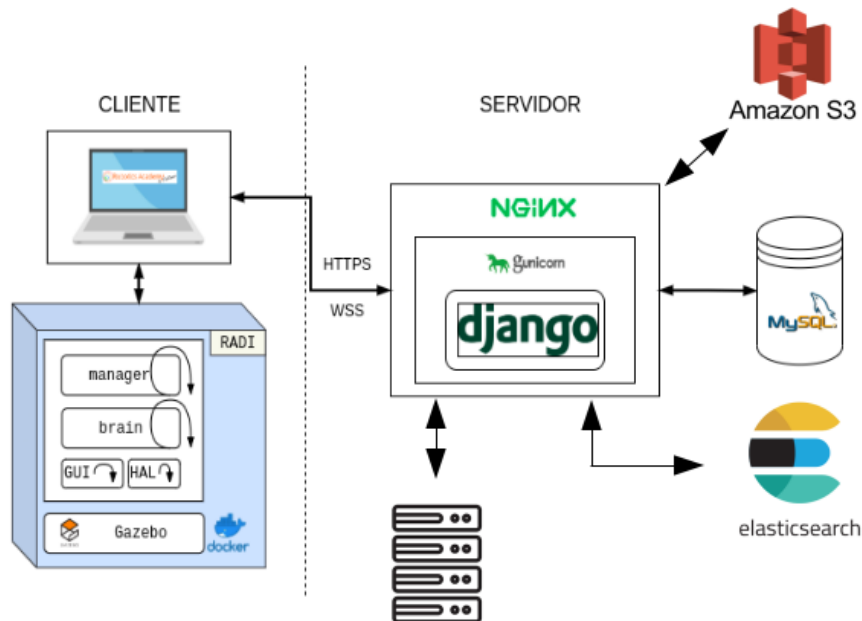


Figura 1.10: Infraestructura de Unibotics

En la figura 1.10 se muestra la infraestructura de la plataforma actualmente. En la parte del servidor se encuentra un *webserver*, el cual se conecta a base de datos como Elasticsearch y MySQL, también se conecta al almacenamiento de la nube de Amazon S3 y una granja de 80 puestos. En la parte del cliente, a parte de la plataforma web se conecta a través de tres websoc-  
ket al contenedor docker, RADI. Esto hace que el coste computacional del simulador sea menor.

En los últimos 5 años se ha recibido ayuda de *Google Summer of code*<sup>27</sup> donde se han ido añadiendo contenidos a la plataforma y también gracias a Trabajos de Fin de Grado.

En la actualidad esta plataforma es utilizada en las asignaturas de Robótica móvil y Robótica de servicios en el grado de Ingeniería de Robótica Software de la universidad Rey Juan Carlos y en el máster de visión artificial de la misma universidad.

---

<sup>27</sup><https://summerofcode.withgoogle.com/archive/>

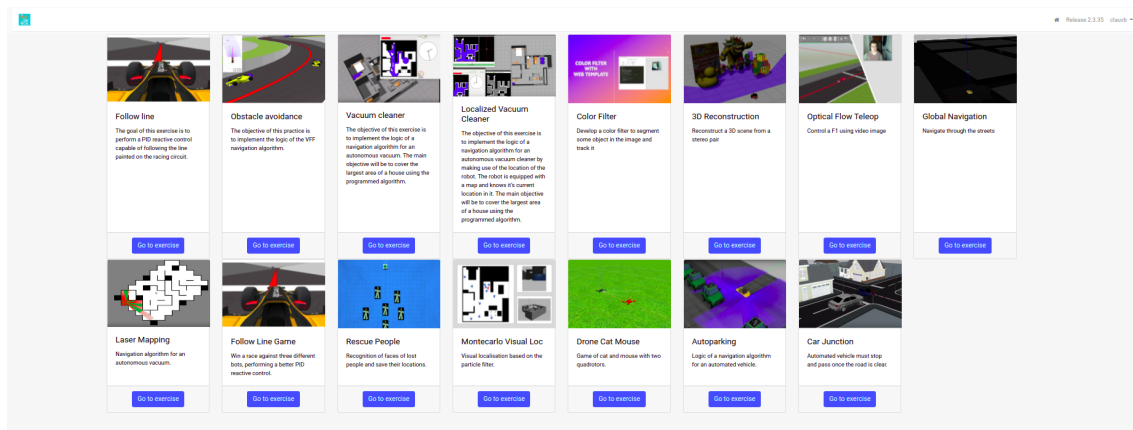


Figura 1.11: Unibotics en la actualidad

## 1.6. Estructura del documento

Este Trabajo de Fin de Grado consta de de los siguientes capítulos:

- *Capítulo 1 Introducción:* introducción a las tecnologías web, a páginas web educativas sobre robótica y Unibotics antes de la realización de este TFG.
- *Capítulo 2 Objetivos y Metodología:* Descripción de los diferentes objetivos a cumplir en el TFG y la metodología seguida para la realización de estos.
- *Capítulo 3 Herramientas utilizadas:* se describen las diferentes tecnologías web utilizadas en este trabajo y las herramientas usadas para la recogida y grabación de datos y la visualización de estadísticas automáticas.
- *Capítulo 4 Analíticas de Elasticsearch y Dash:* en este capítulo se expone el diseño y la implementación de la recogida de sondas y su posterior visualización en la plataforma de Unibotics.
- *Capítulo 5 Conclusiones:* se desarrollan las conclusiones de los resultados obtenidos en este TFG, además de las competencias que se ha adquirido y futuros trabajos que se podrían realizar a partir de este.

## Capítulo 2

# Objetivos y Metodología del Trabajo

En este capítulo de TFG se va a exponer los diferentes objetivos, la metodología seguida para la realización del proyecto, el plan de trabajo y la motivación para escoger el tema.

### 2.1. Objetivos

El objetivo principal que sigue este proyecto es la integración de una aplicación donde se monitorea y se analiza la plataforma web educativa Unibotics. En esta página web de análisis se desea que puedan acceder los administradores para poder comprobar y analizar como interactúan los usuarios con la página, además de poder ver su progreso en los diferentes ejercicios. También se quiere que los usuarios puedan ver un histórico de sus puntuaciones para que puedan evaluar su desempeño. Para cumplir el objetivo principal se ha marcado diferentes subobjetivos:

- Capturar las diferentes interacciones de los usuarios como puede ser el tiempo que un usuario esta en la plataforma Unibotics. Esos datos recogidos son llamados sondas. Estas sondas serán capturadas en el servidor de Django que tiene Unibotics.
- La grabación de las sondas en una base de datos.
- Creación de gráficas dinámicas a través de las sondas recogidas para poder realizar análisis.

## 2.2. Metodología

La metodología seguida en este proyecto ha comenzado con la planificación de reuniones semanales con los tutores. En estas reuniones semanales se hace un repaso de lo avanzado durante toda la semana y se fijan los nuevos objetivos para la semana siguiente. Estas reuniones han sido beneficiosas ya que permitía resolver las dudas más inmediatas y realizar un trabajo continuado. Gracias a la plataforma de Slack<sup>1</sup> se ha podido mantener la comunicación a lo largo de la semana y poder resolver dudas de una manera más dinámica. También en esta plataforma se encontraba un grupo con los administradores de Unibotics donde se soluciona los problemas relacionados con la plataforma.

Para poder hacer el seguimiento del trabajo se ha creado un blog<sup>2</sup>, donde periódicamente se han ido escribiendo los avances realizados.. El blog ha sido creado con *Github-Pages*<sup>3</sup>

Para comenzar a realizar el TFG se ha realizado un primer estudio del código de Unibotics para poder entender el funcionamiento y así añadir el código de mejora. Cuando se ha fijado un objetivo semanal en el que hubiera que añadir código nuevo, lo primero se creaba una incidencia o *issue* describiendo el problema por lo que se creaba una nueva rama o *branch* en la que se trabajaba en el código para no modificar el código original, una vez conseguido solucionar la incidencia se crea un parche o *pull request* con la solución donde los administradores de la plataforma lo revisan y si esta todo bien fusionan la rama creada con la original, lo que se llama hacer un *merge*.

Antes de la utilización de los componentes, como Elasticsearch o Dash, para llevar a cabo este TFG se ha realizado un estudio previo de estos, para poder utilizarlos de una mejor manera. A medida que ha ido avanzando el TFG se ha decidido que sondas convendrían mejor para el monitoreado de la plataforma.

---

<sup>1</sup><https://slack.com/>

<sup>2</sup><https://roboticslaburjc.github.io/2021-tfg-claudia-alvarez/>

<sup>3</sup><https://pages.github.com/>

### 2.3. Plan de Trabajo

El plan de trabajo consta de varias etapas:

- **Unibotics y repaso de tecnologías web:** Estudio del código de Unibotics y repaso de las tecnologías web utilizadas en la plataforma.
- **Estudio de los componentes y herramientas utilizadas:** Estudio de los diferentes componentes utilizados empezando por Django, después se estudió la base de datos Elasticsearch y el visualizador Dash y por último la herramienta *docker container*.
- **Primeros prototipos:** Se ha creado un primer prototipo de una aplicación de Django donde se recogía de manera simple diferentes sondas con Elasticsearch.
- **Recogida y captura de sondas:** Con Elasticsearch se recogen las diferentes sondas para el monitoreado de la plataforma, además se creo una base de datos de prueba para poder trabajar en local.
- **Visualización de las sondas:** Una vez definido que sondas se desean capturar, se crea la aplicación para poder visualizarlas y hacer el análisis.
- **Realización memoria:** Por último se ha escrito la memoria utilizando Latex<sup>4</sup>.

### 2.4. Motivación

En los últimos años el uso de plataformas web educativa ha ido incrementándose y más en los tiempos de pandemia donde se ha tenido que cambiar la educación presencial por una *online*. Por este modo cada vez más profesores buscan llamar la atención de sus alumnos a la hora de aprender y esto es lo que ocurre con la plataforma de Unibotics.

Par que la plataforma pudiera funcionar de la mejor manera y más atrayente a los alumnos es necesario hacer ese monitoreado y posterior análisis, donde los administradores podrán ver como los usuarios interactúan con la plataforma y tomar decisiones a partir de los resultados. Los

---

<sup>4</sup><https://www.latex-project.org/>



## 2.4. MOTIVACIÓN

---

administradores, por lo tanto, podrán ver por ejemplo cuanto tiempo los alumnos tardan en resolver un ejercicio y que nota obtienen de ello, pudiendo hacer correlaciones entre los datos obtenidos.

La monitorización y el análisis esta muy demandado en páginas web ya que esto te puede servir para formar una plataforma competitiva, como puede ser el caso de saber que sistema operativo utilizan los alumnos y añadir las mejoras para que se puedan realizar los ejercicios en esos sistemas operativos.

# Capítulo 3

## Herramientas utilizadas

En este capítulo se describen las tecnologías web que se han utilizado en este TFG. Para añadir la recogida de sondas a Unibotics se ha utilizado como tecnologías del lado del cliente HTML para dar estructura, CSS para el diseño y JavaScript para definir las acciones, las tecnologías del lado del servidor han sido el *framework* de Django y base de datos SQL<sup>1</sup>. Se ha integrado una base de datos llamada Elasticsearch para la recogida de la información y como visualización de estadísticas automáticas se ha utilizado Dash.

### 3.1. Tecnologías web

En esta sección se hablará de las diferentes tecnologías que forman Unibotics, la plataforma web educativa donde se realiza este trabajo y las cuales han sido usadas para llevar a cabo los objetivos de este.

#### 3.1.1. HTML

HTML son las siglas de *HyperText Markup Language* donde "HiperTexto" se refiere a un texto donde hay enlaces a otra página web o en la misma página permitiendo que los documentos estén interconectados entre sí, esto es una parte fundamental de la web. Con marcado hace referencia a que HTML define la estructura del documento por ejemplo que parte del documento va a ser un título y donde se va a encontrar y por último con lenguaje como ya hemos dicho

---

<sup>1</sup>Structured Query Language

anteriormente HTML es un lenguaje pero no de programación si no marcado.

La estructura de un documento HTML esta compuesto por la definición del tipo de documento con `<!DOCTYPE html>`, el elemento `<html >` y `</html >` para dar comienzo y final al documento HTML, el elemento `<head >` y `</head >` donde se introducen los metadatos como es el idioma del documento o el titulo que aparece en la pestaña de la página y el elemento `<body >` y `</body >` donde se escribe todo el contenido que se quiere mostrar a los usuarios.[13]

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <!-- cabecera del documento web -->
5 <title>Mi primer documento web</title>
6 <meta charset="utf-8"/>
7 <meta name="author" content="Francesc Ricart"/>
8 </head>
9 <body>
10
11 <!-- cuerpo del documento web -->
12 <p>El lenguaje HTML sirve para aportar contenido
13 y estructura a un documento web</p>
14 </body>
15 </html>
```

Figura 3.1: Estructura HTML

Un elemento de HTML esta compuesto por etiquetas que son palabras que marcan el inicio y final de una sección. La etiqueta de apertura esta formada por una palabra o letra rodeada por '`<`' y '`>`' dando comienzo al elemento y normalmente con una etiqueta de cierre al igual que la de la apertura pero rodeada por '`</`' y '`>`'. La etiqueta no distingue entre mayúsculas y minúsculas. Aunque haya una gran cantidad de etiquetas a veces se necesita información adicional para completar los elementos, esto se consigue gracias a los atributos.

El atributo se encuentra dentro de la etiqueta de apertura con un espacio en blanco del nombre de la etiqueta o de otro atributo, el atributo esta compuesto por un nombre seguido del signo igual ( = ) y el valor del atributo entre comillas.

Cada etiqueta tiene unos atributos asociados y estos a la vez unos valores predefinidos si se da un valor erróneo a un atributo al renderizar la página esta lo ignorará. Algunos atributos son obligatorios como en el caso de las imágenes, vídeos o enlaces como se muestra en la

figura3.2, la etiqueta para los enlaces es *a* y es obligatorio que le siga el atributo *href* para poder añadir la dirección a la que va a dirigir dicho enlace. Entre las etiquetas de apertura y cierre nos encontramos con el texto que será el contenido de la sección.[14]

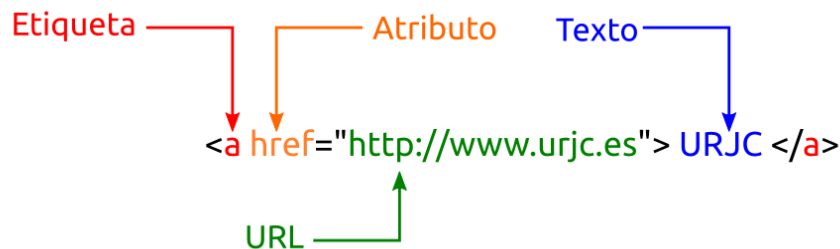


Figura 3.2: Estructura de un elemento HTML

Hay dos tipos de elementos, los elementos de bloques que son los elementos que ocupan toda una línea en el documento estos son los encabezados, listas o párrafos y los elementos en línea que solo ocupan el espacio de su contenido como los botones, enlaces o imágenes. Esto se puede cambiar gracias a atributos. Para estructurar el documento disponemos de dos etiquetas generales, `<div>` la cual crea una sección de tipo bloque y `<span>` para crear una sección en línea.[15]

La última versión es la HTML5 la cual es utilizada en este proyecto, como mejoras respecto a anteriores versiones está la introducción de las etiquetas de audio `<audio>` y vídeo `<video>`, anteriormente la web no estaba pensado para multimedia y había que meter parches como *Flash* de *Adobe*. Se introduce SVG<sup>2</sup> para hacer gráficos vectoriales y así no se podrán ver los píxeles de las imágenes, también como novedad se tiene Canvas la cual es una web de diseño gráfico y composición de imágenes y WebGL que es una especie de Canvas pero en 3D. En HTML5 si das permiso incorpora una API<sup>3</sup> de geolocalización donde se puede ver la ubicación y por último también incorpora lo denominado *Drag and Drop* que consiste en arrastrar y soltar para facilitar la interacción con el usuario.

---

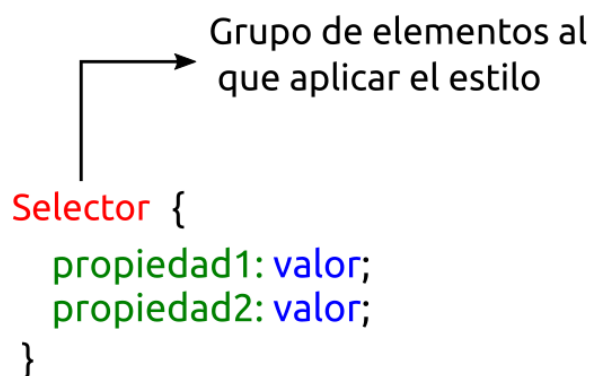
<sup>2</sup>Scalable Vector Graphics

<sup>3</sup>Application Programming Interfaces

#### 3.1.2. CSS

CSS es el lenguaje de estilo encarga del diseño y presentación de los documentos HTML. Para llamar la atención de los usuarios en páginas web es importante añadir estilo a los documentos por eso se utiliza CSS, el cual puede definirse como un atributo de HTML llamado *style*, otra forma de meter estilo es utilizando la etiqueta `<style>` en la cabeza del documento HTML pero la mejor opción para añadir estilo es separándolo de la estructura, es decir, del documento HTML creando una hoja de estilo *.css* debido a que es más sencillo realizar cambios y se podrá diversificar el trabajo en estructura y estilo siendo más productivo. Para vincular la hoja de estilo con el HTML se utiliza la etiqueta `<link>` en la cabecera.

La estructura de una regla CSS se divide en selectores que contienen pares de propiedad-valor como se muestra en la figura 3.3. En los selectores se pone el nombre de las etiquetas las cuales quieres cambiar su estilo como puede ser la etiqueta *body*, además estos selectores pueden ser el valor del atributo *id* el cual es un identificador único para un elemento de HTML en este caso se pone el símbolo `#` antes del valor de su *id*, a parte de identificar un elemento con un *id* se puede utilizar el atributo *class* que es un identificador para varios elementos, en este caso para añadirle estilo a los elementos pertenecientes a la misma clase al selector se le añade un punto delante.



```
Selector {  
    propiedad1: valor;  
    propiedad2: valor;  
}
```

Figura 3.3: Sintaxis CSS

Las propiedades indican cual es el estilo que se quiere cambiar en un elemento como puede ser el color o el tamaño. Cada propiedad tiene unos valores asociados que en algunas ocasiones se pueden escribir de diferente manera como en el caso de los colores, se puede escribir directamente como *red* o ponerlo en su valor hexadecimal o en valores RGB<sup>4</sup>.

A la hora de utilizar estilos se puede poner estilos contradictorios en este caso el último estilo definido será el que se acabe aplicando, esta sería la parte de cascada que indica las siglas CSS. Si se ha utilizado una hoja de estilo pero además se ha definido en una etiqueta HTML, el definido en la etiqueta será el utilizado al renderizar el documento. La herencia también es un concepto importante en CSS ya que si un elemento no tiene estilo pero esta contenido en otro elemento que si tiene, este heredará su estilo. Los identificadores únicos tendrán preferencia a añadir el estilo que los identificadores de clase, el nombre de la etiqueta como selector es el de menor preferencia entre los selectores.[16]

#### 3.1.3. JavaScript

JavaScript es un lenguaje de programación interpretado que permite la ejecución de código orientado a eventos. Pueden actuar sobre el navegador a través de objetos integrados como un botón. El DOM<sup>5</sup>, API que representa al documento y define la manera de interactuar con él, puede ser modificado dinámicamente gracias a JavaScript. Es importante la colocación del código JavaScript ya que se ejecuta ordenadamente de arriba a abajo.

JavaScript puede estar en el lado del cliente haciendo que su código se ejecute en el navegador donde podrá interactuar con el navegador, además podrá interactuar con el documento HTML o dibujar en la página. También existen *frameworks* para el desarrollo de la parte servidor de una plataforma Web basados en JavaScript, como Node.js<sup>6</sup>. [17]

Hay diferentes formas de agregar código JavaScript a un documento HTML, como ocurría con CSS la mejor opción es tener un fichero .js separado de la estructura (HTML) y el estilo

---

<sup>4</sup>Red, Green, Blue,

<sup>5</sup>Document Object Model

<sup>6</sup><https://nodejs.org/>

(CSS) para poder trabajar de una mejor manera, para añadirlo en la cabecera del HTML se le inserta la etiqueta `<script>` con el atributo `src` para indicar la ubicación del fichero. Para que no haya ningún problema conviene ejecutar el código cuando se haya cargado la página para ellos se utiliza el atributo `onload` que indica que se ha cargado la página y se llama a una función principal del fichero JavaScript. Otra forma de introducir JavaScript en el documento HTML es directamente en sus etiquetas por ejemplo cuando ocurre un evento o mediante la etiqueta `<script>` que ofrece HTML .[18]

En este TFG se ha utilizado JavaScript para detectar las diferentes interacciones de los usuarios en la página web a través de eventos. También se ha hecho uso de JQuery<sup>7</sup>, una librería de JavaScript que permite una manipulación más sencilla del DOM y de los eventos que se generan en éste.

#### 3.1.4. Django

Django es un *framework* de desarrollo web de código abierto escrito en Python. Django es un *framework* que actualmente dispone de mucha funcionalidad, es decir, viene con extras para ayudar al desarrollo de una web, además es versátil, escalable, rápida y segura. Una de sus principales ventajas es que levanta una página web de administración donde se pueden hacer acciones como retocar la base de datos. Django sigue el Modelo Vista Plantilla (MVT) como se muestra en la figura 3.4.

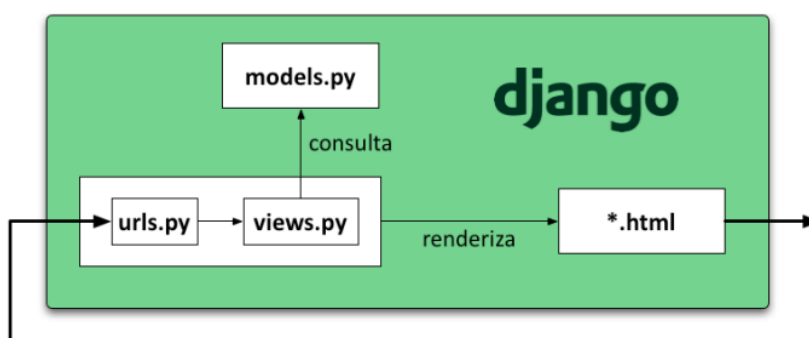


Figura 3.4: Estructura de Django

---

<sup>7</sup><https://jquery.com/>

Los modelos en Django son objetos en Python que son guardados en una base de datos. Los modelos son independientes al gestor de base de datos que se vaya a utilizar, se definen las estructuras de información de una manera genérica y además se puede añadir restricciones. Django puede traducir filtros de python en SQL gracias a su ORM, por lo tanto no es necesario saber lenguaje de datos. Los modelos estan formado por una lista de campos donde se define el dominio de cada campo, si no hay ningún campo que tenga una clave primaria, Django generará una columna llamada id que se incrementará automáticamente. Los modelos se escriben en el fichero *models.py*. [19]

Las vistas en Django esta formado por el fichero *urls.py* y *views.py*. En el fichero *urls.py* a partir de expresiones regulares se definen las urls que redirigen a funciones de *views.py*. La función *views.py* recibe un objeto *HTTP Request* y todos los parámetros de la URL capturados teniendo que devolver un objeto *HTTP Response*. Lo habitual en web dinámicas es hacer una consulta a la base de datos, generar un contexto que empotra en una plantilla y a través la cual se renderiza devolviendo un HTTP Response.

Las plantillas son páginas dinámicas, es decir, son protopáginas la cuales solo se pueden renderizar juntando el contexto (diccionario con los valores que dan contexto a una plantilla) pasado por las vistas y gracias al lenguaje de plantillas dando como resultado normalmente un documento HTML. Esto es útil al programar ya que permite hacer páginas dinámicas en pocas líneas

En resumen para formar un servidor en Django primero hay que diseñar el modelo de datos, luego se diseñan las urls que enrutaran a las vistas las cuales preparan un contexto que juntándolo con la plantilla se genera el HTTP Response.

La versión actual de Unibotics está basada en Django 2.2.



#### 3.1.5. SQL

SQL<sup>8</sup> es un lenguaje de consulta estructurado, se utiliza para definir, manipular y gestionar los datos almacenados en una base de datos relacional. SQL es un estándar reconocido en 1986 por ANSI<sup>9</sup> y en 1987 por ISO<sup>10</sup>

Se necesita un gestor de base de datos RDBMS<sup>11</sup> que se encargará de interactuar con la base de datos por ejemplo MySQL o Access SQL. Algunos de estos gestores trabajan en local y otros en un servidor remoto.[20]

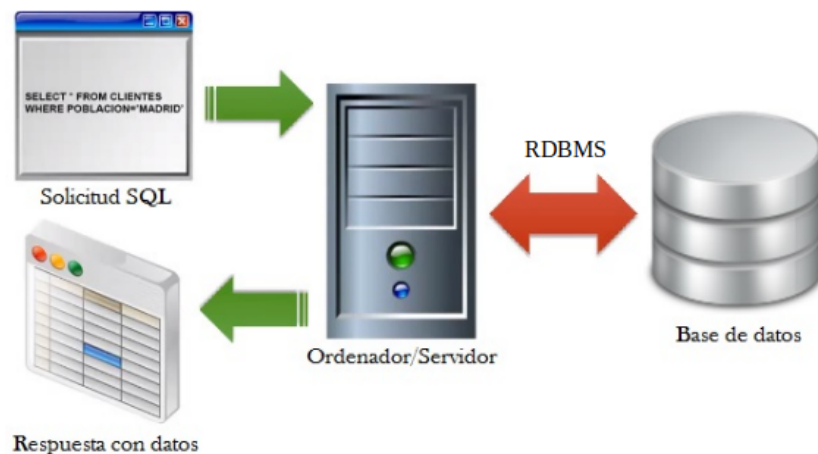


Figura 3.5: Esquema funcionamiento SQL

Una instrucción SQL esta formada por comandos, cláusulas, operadores y funciones . Los comandos de una sentencia SQL se dividen en cuatro tipos:

- **DDL (*Data Definition Language*)**: sirve para crear o modificar la estructura de una base de datos. Algunos de los comandos más importantes de este tipo son *CREATE* para crear nuevas tablas o base de datos, *ALTER* para modificar la tabla de la base de datos, *DROP* para eliminar tablas o índices
- **DML (*Data Manipulation Language*)**: sirve para hacer consultas de selección y de acción a la base de datos. Por ejemplo tenemos los comandos *SELECT* para extraer los datos,

---

<sup>8</sup>Structured Query Language

<sup>9</sup>American National Standards Institute

<sup>10</sup>International Organization for Standardization

<sup>11</sup>Relational Database Management System

*INSERT* para insertar nuevos datos, *UPDATE* para actualizar los datos y *DELETE* para eliminarlos.

- *DCL (Data Control Language)*: se utiliza para proporcionar seguridad a la base de datos. El comando para otorgar permiso es *GRANT* y para retirarlos es *REVOKE*
- *TCL (Transactional Control Language)*: su función es administrar los cambios en los datos. De este tipo de comandos se tiene *COMMIT* para guardar el trabajo realizado o *ROLLBACK* para deshacer las últimas modificaciones hechas después del último *COMMIT*

Las cláusulas son condiciones de modificación para poder definir los datos. Las cláusulas más importantes son *FROM* para especificar la tabla, *WHERE* para definir las condiciones de los registros que se desean, *GROUP BY* para hacer agrupaciones específicas de registros, *HAVING* para declarar la condición que debe cumplir cada grupo, *ORDER BY* para que los registros sigan un orden especificado.

Los operadores pueden ser lógicos, por ejemplo, *AND*, *OR* o *NOT*) o de comparación que serían las operaciones del estilo mayor que, menor que o igual que. Las funciones se utilizan con el comando *SELECT* para devolver un único valor de un grupo de registros como puede ser *AVG* que te devuelve la media o *COUNT* para devolver el número de registros. En la figura 3.6 se muestra un ejemplo de una sentencia SQL con con todas sus partes.[21]

```
SELECT Avg(Gastos) AS Promedio  
FROM Pedidos|  
WHERE Gastos > 100;
```

Figura 3.6: Sentencia SQL

En este TFG se ha utilizado SQL ya que Unibotics dispone de una base de datos MySQL donde se almacena la información estructural de la plataforma como los usuarios o ejercicios, a la cual se accede a través de Django.

## 3.2. Recogida y grabación de datos

Para la recogida y grabación de las sondas es necesario tener una base de datos, en este caso para este proyecto se ha decidido utilizar una base de datos no relacional (NoSQL<sup>12</sup>). Una base de datos no relacional se caracteriza por almacenar datos no estructurados o semi-estructurados, además sus datos no están organizados en tablas como era el caso de las bases de datos relacionales por ejemplo MySQL.

En este TFG se ha utilizado Elasticsearch<sup>13</sup>, el cual es un motor de búsqueda y análisis de código abierto, escrito en Java y basado en Lucene, una biblioteca de Java que proporciona funciones de indexación y búsqueda entre otras. Elasticsearch está orientado a documentos JSON<sup>14</sup> formado por un conjunto de pares clave-valor, donde la clave es una cadena de texto y el valor puede ser de diferentes tipos como un texto o una lista.[22]

Elasticsearch se caracteriza por ser una herramienta rápida permitiendo una búsqueda de texto completo bastante eficiente. Gracias al poco tiempo transcurrido entre la indexación (grabación de datos en Elasticsearch) y la posterior búsqueda hemos considerado que se trata de una herramienta adecuada para la grabación de las sondas y la posterior consulta de la información. Una de las ventajas de las bases de datos no relacionales es que están implementadas para permitir un escalado horizontal, es decir se puede dividir la base de datos en diferentes servidores, de una manera más sencilla que con las bases de datos relacionales. Los diferentes datos están agrupados en lo que se llama *shards* en los cuales se aplican técnicas de réplica para ser tolerante a los fallos, además si Elasticsearch tiene algún fallo es capaz de detectarlo y reorganizar la información. Existen diferentes módulos en diferentes lenguajes de programación que permiten una interacción sencilla con Elasticsearch.[23]

Elasticsearch esta formado por *clusters* que son un conjunto de nodos, en los cuales se almacena todos los datos, también puede estar formado por un único nodo. Al *cluster* se le asigna un nombre y un identificador único. Los nodos que se encuentran en un *cluster* son unos servi-

---

<sup>12</sup>*Not only SQL*

<sup>13</sup><https://www.elastic.co/>

<sup>14</sup>JavaScript Object Notation

### 3.2. RECOGIDA Y GRABACIÓN DE DATOS

---

dores únicos que almacenan documentos y ayuda en las capacidades de indexación del *cluster*, al nodo también se le asigna un nombre y un identificador. Dentro de los nodos se pueden encontrar uno o más índices que son una colección de documentos con características similares, tiene que ser nombrado en minúsculas y será este nombre al que se hará referencia para realizar las diferentes operaciones. Los documentos que encontramos en un índice esta formada por la información básica que es recogida.[24]

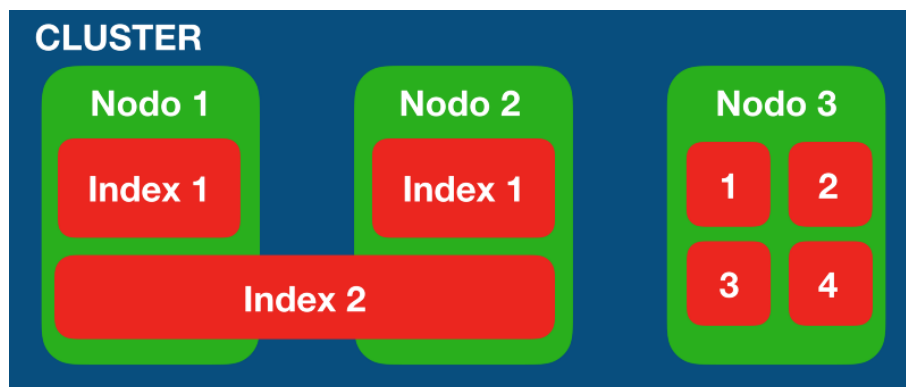


Figura 3.7: Arquitectura Elasticsearch

En el caso que se guarde una gran cantidad de documentos sobrepasando los límites de almacenamiento de un nodo, Elasticsearch divide cada índice en diferentes fragmentos (*shards*), la distribución de cada fragmento por los diferentes nodos se realiza de forma automática. Para evitar fallos en los nodos se hacen réplicas de los *shards* y se almacenan en un nodo diferente al *shard* primario, así en el caso de que se produzca un fallo en un nodo se podrá seguir trabajando.

En este TFG se ha utilizado la versión 7.12.0 de Elasticsearch.

## 3.3. Visualización de estadísticas automática

Una vez ya recogido las sondas y se han almacenado en Elasticsearch, es la hora de poder visualizarlos para poder hacer un análisis de los resultados. En este TFG se ha decidido utilizar el *framework* Dash<sup>15</sup> para poder realizar esta tarea. Dash es un *low-code framework*, es decir que permite la creación de aplicaciones de manera rápida y eficiente haciendo un menor uso de la programación manual, está escrito sobre Plotly.js y React.js. Dash está disponible en lenguajes de programación como Python, Julia, R o F. Dash es multiplataforma, por lo que puede ser utilizado desde cualquier dispositivo.[25]

En las aplicaciones de dash no es necesario escribir ningún documento HTML, JavaScript o CSS ya que Dash proporciona una abstracción pura en Python mediante el uso de la biblioteca de `dash.html_components`. Otra biblioteca de python que se utiliza en una aplicación Dash es `dash-core-components` el cual son una serie de componentes para una interfaz de usuario interactiva como un menú desplegable.

La estructura de los datos que se van a visualizar en Dash son *Dataframe* de la biblioteca de Pandas<sup>16</sup>. Un DataFrame es una tabla en la que los datos guardados en cada columna representan las diferentes variables.

Las aplicaciones de Dash están formadas por dos partes la primera es la llamada *layout* donde se describe cual va a ser el aspecto de la aplicación, esto son los menús y las gráficas que se visualizan, aquí es donde se utilizaran las bibliotecas mencionadas anteriormente. La segunda parte es la interactividad con los usuarios, para ello se utiliza `dash.dependencies` donde los componentes interactivos crean una entrada y a través de *callbacks* modifican las gráficas, de tal manera que se pueden integrar filtros para generar diferentes visualizaciones en base a las necesidades del usuario.

Para este TFG se ha utilizado Dash en la versión 1.17.0.

---

<sup>15</sup><https://dash.plotly.com/>

<sup>16</sup>Python Data Analysis Library

# Capítulo 4

## Analíticas de Elasticsearch y Dash

En este capítulo se explica la recogida de las sondas en Unibotics, su posterior guardado en la base de datos Elasticsearch, la visualización de dichos datos con el *framework* web Dash y la validación experimental del proceso.

### 4.1. Recogida de sondas

La primera parte de este proceso ha consistido en la recogida de diferentes sondas. Para realizar esta tarea, se ha decidido utilizar la herramienta de Elasticsearch por las ventajas que ofrece explicadas en el capítulo 3.

En este proyecto se ha utilizado docker, un contenedor a nivel de sistema operativo para Elasticsearch. Se ha utilizado Docker debido que permite que Elasticsearch funcione en cualquier sistema operativo, y replicar la instalación en cualquier máquina es directo. Primero hay que descargar la imagen de Elasticsearch con la versión deseada en este caso se ha utilizado la 7.12.0, el comando para descargar la imagen es:

```
docker pull docker.elastic.co/elasticsearch/elasticsearch:7.12.0
```

Para ejecutar el contenedor con la imagen descargada se utiliza el siguiente comando:

```
docker run --name=AcademyElastic -p 9200:9200 -p 9300:9300  
-e "discovery.type=single-node"  
docker.elastic.co/elasticsearch/elasticsearch:7.12.0
```

## 4.1. RECOGIDA DE SONDAS

---

Una vez ya puesto en marcha el contenedor de Elasticsearch ya tenemos nuestra base de datos donde se guardaran las sondas. Para hacer solicitudes se podrá hacer mediante *REST APIs*, un ejemplo sería introducir en el navegador la siguiente URL: `http://localhost:9200/`.

El siguiente paso es la integración de Elasticsearch en el servidor basado en Django. Para ello se hará uso de la librería `django-elasticsearch-dsl`. Además se ha modificado el archivo de configuración del proyecto, `settings.py`, añadiendo la librería a las aplicaciones instaladas y creando una nueva variable llamada `ELASTICSEARCH_DSL`, también en `settings.py`, ya que las variables declaradas en ese fichero se pueden utilizar en cualquier parte del servidor. En esa variable se indica el servidor de Elasticsearch con el que se tiene que conectar y sincronizar.

Una vez conectado Django con Elasticsearch se han definido y configurado los índices donde se guardaran las sondas, para ellos se ha creado un nuevo archivo llamado `probe.py`. Al definir un índice, se determinan los nombres de cada campo (información que se quiere almacenar) con el tipo de campo que es, por ejemplo si es un texto, un número, una fecha, entre otros. También se puede configurar el índice, por ejemplo poniendo el número de *shards* o el número de réplicas. Un ejemplo de la definición de un índice sería este:

```
1  from django_elasticsearch_dsl import Document, Date, Text, Double
2
3  class SessionDocument(Document):
4      username = Text()
5      start_date = Date()
6      end_date = Date()
7      duration = Double()
8      client_ip = Text()
9      browser = Text()
10     country = Text()
11     alpha_2 = Text()
12     alpha_3 = Text()
13     continent = Text()
14     class Index:
15         name = 'session_log'
16         settings = {
17             'number_of_shards': 1,
18             'number_of_replicas': 0
19         }
```

#### 4.1. RECOGIDA DE SONDAS

---

Para este proyecto se han definido cuatro índices diferentes:

- `session_log`: índice que recoge las sondas relativos a las sesiones. Consta de los campos de inicio y fin de sesión, duración de la sesión, la IP, el *browser* (aporta información sobre el sistema operativo, navegador y dispositivo utilizado), el continente y país del usuario, así como el nombre del usuario.
- `exercises_log`: índice que recoge las sondas relativos a los diferentes ejercicios. Esta compuesto por la fecha de entrada en un ejercicio, la fecha de salida del ejercicio, la duración total, el nombre del ejercicio y el usuario.
- `style_log`: índice que recoge los datos sobre la evaluación del estilo del código de los ejercicios. Este índice esta formado por el campo de la fecha en la que se realiza la evaluación, el nombre del ejercicio, la puntuación sobre 100 y el nombre del usuario.
- `efficacy_log`: índice que recoge los datos sobre la evaluación de la eficacia del código de los ejercicios. Los campos son iguales que en el índice de `style_log`.

Ya definidos los diferentes índices se importan al archivo `views.py` para poder crearlos. Las sesiones de los usuarios se guardan en el índice `session_log` y a través de las sondas creadas, por ello podemos recoger cuando el usuario *loguea* en la plataforma y cuando finaliza su actividad en ella. Se recoge la sonda de la siguiente manera:

```
probe_session = SessionDocument(username=username, start_date=datetime.now(),
                                end_date=datetime.now(), duration=0, client_ip=ip,
                                browser=request.META['HTTP_USER_AGENT'],
                                country=location_info["country_name"],
                                alpha_2=location_info["alpha_2"],
                                alpha_3=location_info["alpha_3"],
                                continent=location_info["continent"])

probe_session.save()
```



## 4.1. RECOGIDA DE SONDAS

---

Gracias al objeto HTTP Request que recibe el fichero `views.py` obtenemos la información del nombre del usuario, la IP y el *user-agent* el cual nos dirá el sistema operativo, el navegador o el dispositivo que utiliza el usuario. Para la localización se ha creado una función que a partir de la IP muestra la ubicación. Cuando el usuario *loguea*, los campos de fin de sesión y duración se inicializan con la fecha actual y 0 respectivamente, una vez que el usuario haga *logout* o finalice su sesión por inactividad estos campos se modificarán como se muestra a continuación:

```
s = Search(index="session_log").query('match', username=request.user.username) \
        .sort({"start_date": {'order': 'desc'}})[0]

for hit in s:
    end = datetime.now()
    Elasticsearch(settings.ELASTICSEARCH_DSL['default']['hosts']) \
        .update(index="session_log", id=hit.meta.id,
                body={"doc": {'end_date': end,
                              'duration': (end - datetime \
                                           .strptime(hit.start_date, '%Y%m%dT%H:%M:%S.%f')) \
                                           .total_seconds()}})
```

Las sondas relativas a los ejercicios se guardan cuando el usuario accede a un ejercicio y como ocurre con las sesiones, cuando el usuario abandone el ejercicio se modificarán los datos de duración y fin del ejercicio. Se ha tenido en cuenta que al recargar un ejercicio en el que el usuario ya se encontraba, no se cree una sonda nueva y se tenga en cuenta la primera sonda creada para dicho ejercicio. Las sondas no deseadas que se han creado se eliminan de la siguiente forma:

```
es = Search(index="exercises_log").query('match', duration=0) \
    .query('match', username=request.user.username) \
    .sort({"start_date": {'order': 'desc'}})

for hit in es:
    Elasticsearch(settings.ELASTICSEARCH_DSL['default']['hosts']) \
        .delete(index="exercises_log", id=hit.meta.id)
```

Cada ejercicio que se encuentra en Unibotics dispone de un botón de evaluación automática de estilo, donde se dan unas recomendaciones para mejorar el estilo del código del usuario con una puntuación. La puntuación es recogida en el índice de `style_log`. Para poder recoger la puntuación recibida en la evaluación de eficacia, se ha creado un nuevo botón en los ejercicios.

## 4.1. RECOGIDA DE SONDAS

---

Una vez pulsado el botón, empieza a ejecutar el código durante un tiempo determinado según el ejercicio. Pasado ese tiempo se obtiene la nota obtenida del ejercicio y se guarda en el índice de `efficacy_log`. Si el usuario pulsa dos veces al botón la sonda no se guardará.

En este proceso de recogida de sondas y su grabación ha sido muy útil la utilización de la API que proporciona Elasticsearch para poder depurar y comprobar los datos que se estaban almacenando. Utilizando por ejemplo la URL `http://127.0.0.1:9200/session_log/_search/?size=10000pretty` se comprueba las sondas de sesiones.

Para comprobar el funcionamiento de Elasticsearch se generaron datos de prueba para poder comenzar a trabajar antes de tener los datos reales los cuales llevan tiempo recoger. La base de datos Elasticsearch dummy se ha creado gracias a las librerías de python Faker<sup>1</sup> y Tornado<sup>2</sup> en ella se puede modificar las sondas como por ejemplo el número de sondas, de réplicas o de *shards*. Esto ayudará a futuros desarrolladores a utilizar la base de datos de Elasticsearch. En la figura 4.1 se muestra como ejemplo el resultado de la sonda de evaluación de estilo de la base de datos de prueba.

```
{
  "took" : 3,
  "timed out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10000,
      "relation" : "gte"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "style_log",
        "_type" : "doc",
        "_id" : "0Zp7EHoB05z3ipWglPPg",
        "_score" : 1.0,
        "_source" : {
          "username" : "Dora",
          "date" : "2021-06-07T09:32:32.00",
          "exercise" : "drone_cat_mouse",
          "score" : 6
        }
      },
      {
        "_index" : "style_log",
        "_type" : "doc",
        "_id" : "0pp7EHoB05z3ipWglPPg",
        "_score" : 1.0,
        "_source" : {
          "username" : "Regina",
          "date" : "2021-06-02T06:34:13.00",
          "exercise" : "obstacle_avoidance",
          "score" : 30
        }
      }
    ]
  }
}
```

Figura 4.1: Elasticsearch dummy

---

<sup>1</sup><https://faker.readthedocs.io/en/master/>

<sup>2</sup><https://www.tornadoweb.org/en/stable/>

### 4.2. Visualización de la información

Dash es un *framework* web que permite crear tableros web interactivos con visualizaciones dinámicas, para poder hacer análisis como se explica en el capítulo 3. En este Trabajo de fin de grado se ha decidió utilizar esta herramienta para la visualización de las sondas recogidas en Elasticsearch.

Sólo los usuarios registrados en Unibotics podrán acceder a las visualizaciones y, dependiendo del tipo de usuario, tendrán disponibles unas visualizaciones u otras. Para iniciar sesión, se ha hecho uso de los usuarios ya guardados en la base de datos MYSQL de la que depende Django en la que se encuentra la información sobre el tipo de usuario (staff, admin, user...). En el caso de los administradores podrán ver la información de todas las sondas, mientras que un usuario de Unibotics solo podrá acceder a las puntuaciones de estilo y de eficacia obtenidas en cada ejercicio. En la figura 4.2 muestra el menú disponible para los administradores.

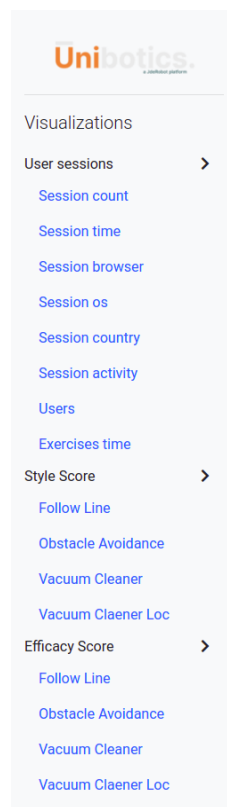


Figura 4.2: Menú de un administrador en Dash

## 4.2. VISUALIZACIÓN DE LA INFORMACIÓN

---

Gracias a las bibliotecas de Dash mencionadas en el capítulo 3 se ha dado estilo y formato a la aplicación, también se ha hecho uso de la *cookies* del navegador para comprobar si se está autorizado y si es un administrador de la plataforma.

Dash trabaja con *dataframes*, por lo que es necesario realizar una primera transformación de los datos de Elasticsearch a esta estructura, gracias a la biblioteca de Pandas. Un ejemplo de la realización de esta conversión es:

```
s = Search(using=es, index="session_log")
results = [hit.to_dict() for hit in s.scan()]
df = pd.DataFrame(results)
```

Con la biblioteca *dash-core-components* se han creado los filtros que algunas de las gráficas poseen. Estos filtros de forma dinámica cambian las visualizaciones en base a dicho filtrado. Uno de los filtros que más se utiliza es el filtro por fechas. Estableciendo una fecha de inicio o de fin o ambas es posible conocer la situación de Unibotics en un periodo de tiempo concreto. El código de filtrado ha sido:

```
if start_date is not None and end_date is not None:
    df=df.loc[(df['start_date'] > start_date) & (df['start_date'] <= end_date)]
elif start_date is not None:
    df=df.loc[(df['start_date'] > start_date)]
elif end_date is not None:
    df=df.loc[(df['start_date'] <= end_date)]
```

Otro filtro recurrente en la mayoría de las visualizaciones es el filtro por usuario. Este filtro se encuentra, además en las gráficas de las evaluaciones de estilo y eficacia del código pero solo para los administradores, así podrán ver las puntuaciones de los diferentes usuarios. Para este filtro se hace uso de las sondas de sesiones, recogiendo los nombres de los usuarios de forma única y añadiendo un 'Total' en los casos que se quiera visualizar las sondas de todos los usuarios. En resumen el filtro podrá filtrar por usuario único, un grupo de usuarios o por el total de los usuarios, `df = df[df.username.isin(user)]`. El código para conseguir los nombres de los usuarios y añadir la opción de 'Total', es el siguiente:

## 4.2. VISUALIZACIÓN DE LA INFORMACIÓN

---

```
s = Search(using=es, index="session_log")
results = [hit.to_dict() for hit in s.scan()]
df = pd.DataFrame(results)
df = df[df['username'].notna()]
users = df['username'].unique()
if not exercises:
    users = np.insert(users, 0, 'Total')
return users
```



The image shows a snippet of the Dash web interface. At the top, there is a date range filter with two input fields labeled 'Start Date' and 'End Date' separated by a right-pointing arrow. Below this, the text 'Users:' is followed by a dropdown menu. The dropdown menu is currently open, showing a single option labeled 'Total'.

Figura 4.3: Filtros utilizados en Dash

Dash utiliza el modulo Plotly para generar las visualizaciones. Plotly ofrece una gran variedad de gráficas que se pueden utilizar en Dash. Adicionalmente se puede combinar varias gráficas en los mismos ejes, haciendo más sencilla la correlación entre datos. El código, por ejemplo, para crear la gráfica de número de sesiones por día es:

```
fig = px.line(df, x='start_date', y='count')
fig.update_layout(xaxis_tick0=df['start_date'][0], xaxis_dtick=86400000 * 2)
fig.add_trace(go.Scatter(x=df["start_date"].tolist(), y=df["count"].tolist(),
                        mode="markers", textposition="top center",
                        name="Number of sessions",
                        text=df["count"].tolist()))
return fig
```

## 4.3. Validación experimental

En esta sección se detallan los resultados finales de las analíticas con datos reales de la plataforma de Unibotics, donde se muestran las diferentes gráficas creadas tanto para administradores como para los usuarios, así como una explicación de lo que representan.

La primera gráfica creada se muestra en la figura 4.4 , donde se representa en el eje x el tiempo y en el eje y el número de sesiones, dando como resultado una gráfica de línea con énfasis en los puntos para una mejor visualización. Esta gráfica muestra el número de sesiones por día. La gráfica muestra que días ha habido más *logings*, viéndose en este ejemplo como en los meses de verano dicho número se ha reducido.

### Total sessions count

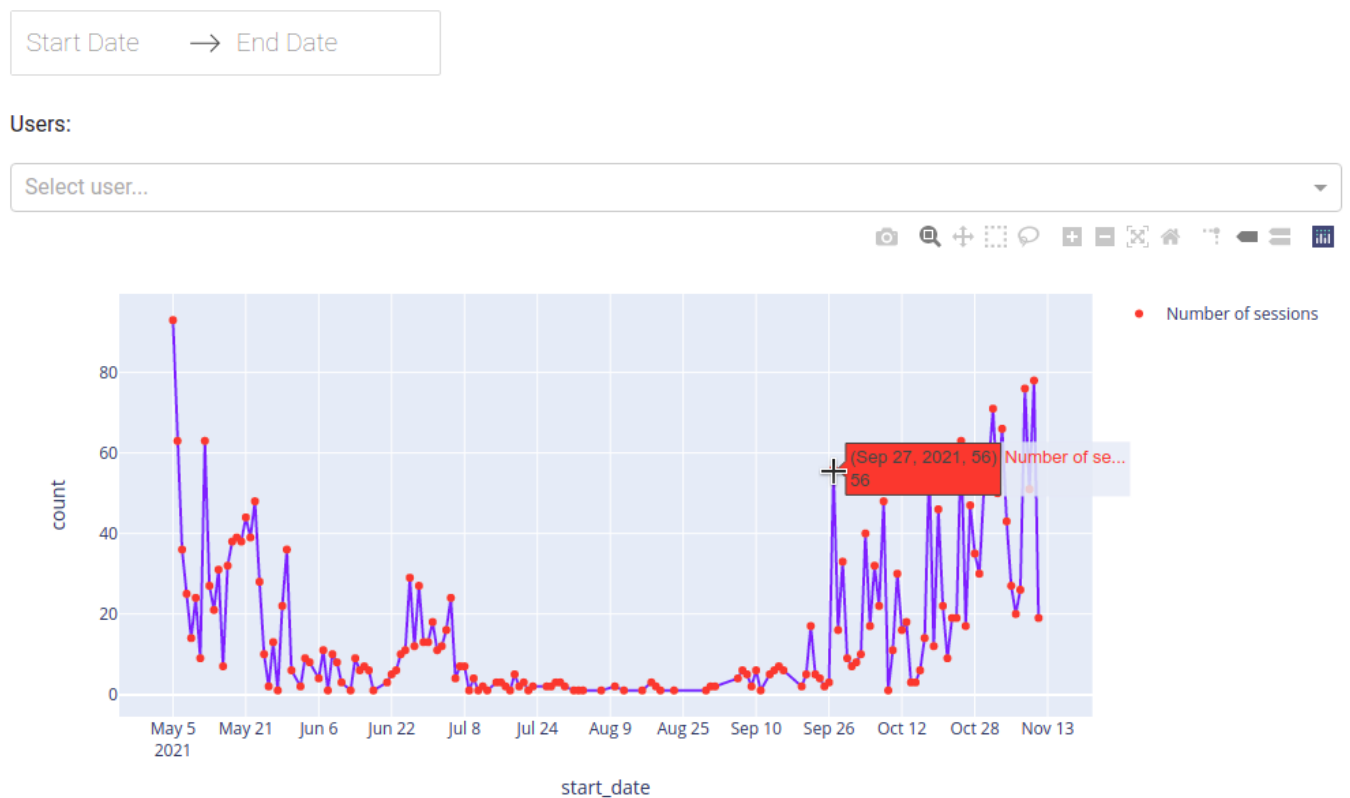


Figura 4.4: Gráfico sesiones totales por día

### 4.3. VALIDACIÓN EXPERIMENTAL

En todas las gráficas de Dash si se pone el cursor sobre uno de los puntos se puede ver la información con mayor detalle, como se muestra en la figura 4.4. Igualmente Dash añade varias opciones para interactuar con la gráfica, por ejemplo, se puede descargar la gráfica, hacer *zoom* o seleccionar una parte de ella.

Se ha creado otra gráfica lineal de sesiones por día, pero en este caso solo se tiene en cuenta el número de sesiones por usuario único. La gráfica esta creada de la misma manera que la gráfica del total de sesiones por día y con los mismos filtros (fechas y usuarios). En la figura 4.5 se muestra una parte ampliada de dicha gráfica.

#### Total sessions of unique users

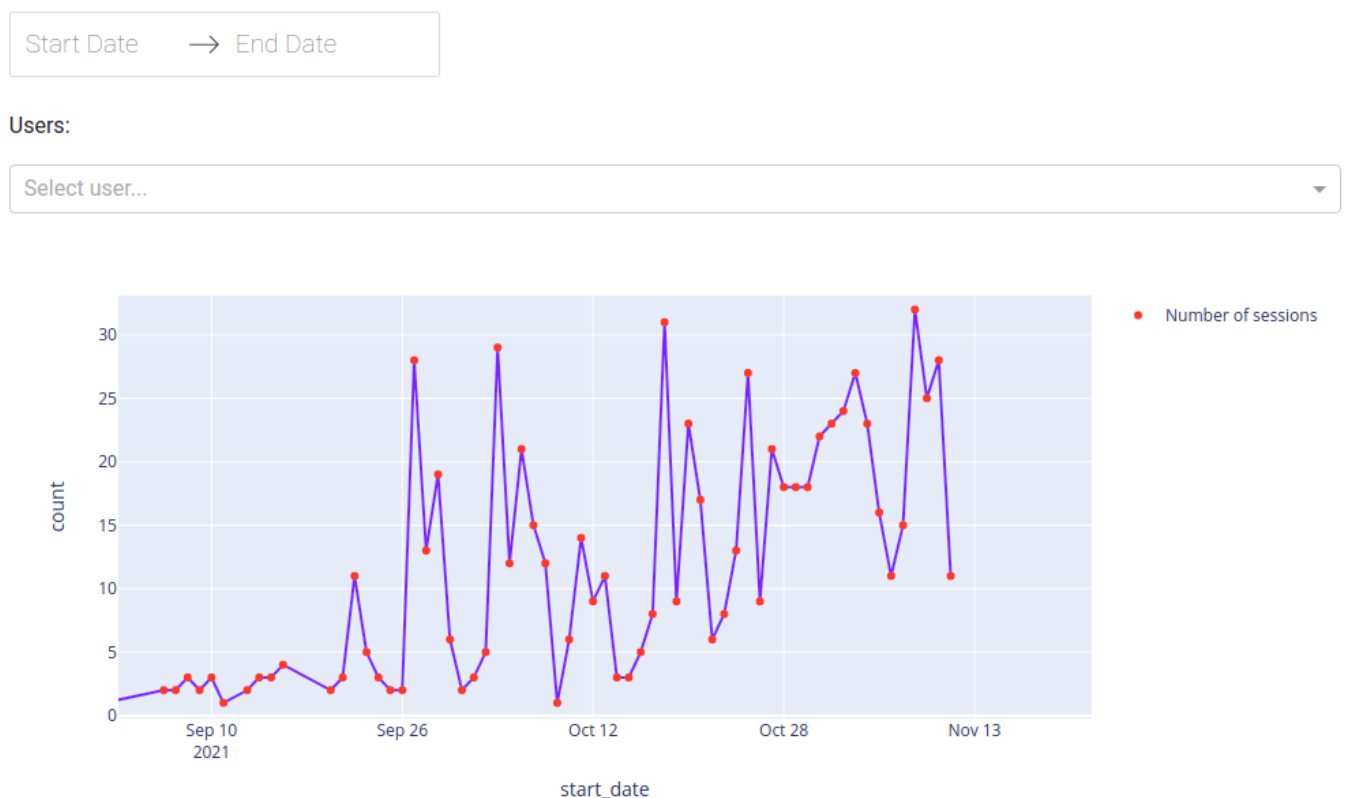


Figura 4.5: Gráfico sesiones únicas por día

### 4.3. VALIDACIÓN EXPERIMENTAL

La figura 4.6 muestra una gráfica de barras en la cual se representa el tiempo mediante el eje x y la duración total de la sesión de los usuarios en minutos en el eje y. Además, se ha incluido una media que representa la duración media de las sesiones. Aquí podemos comprobar que a veces la media coincide con la duración total debido a que solo ha tenido que haber una sesión en ese día.

#### Total Time

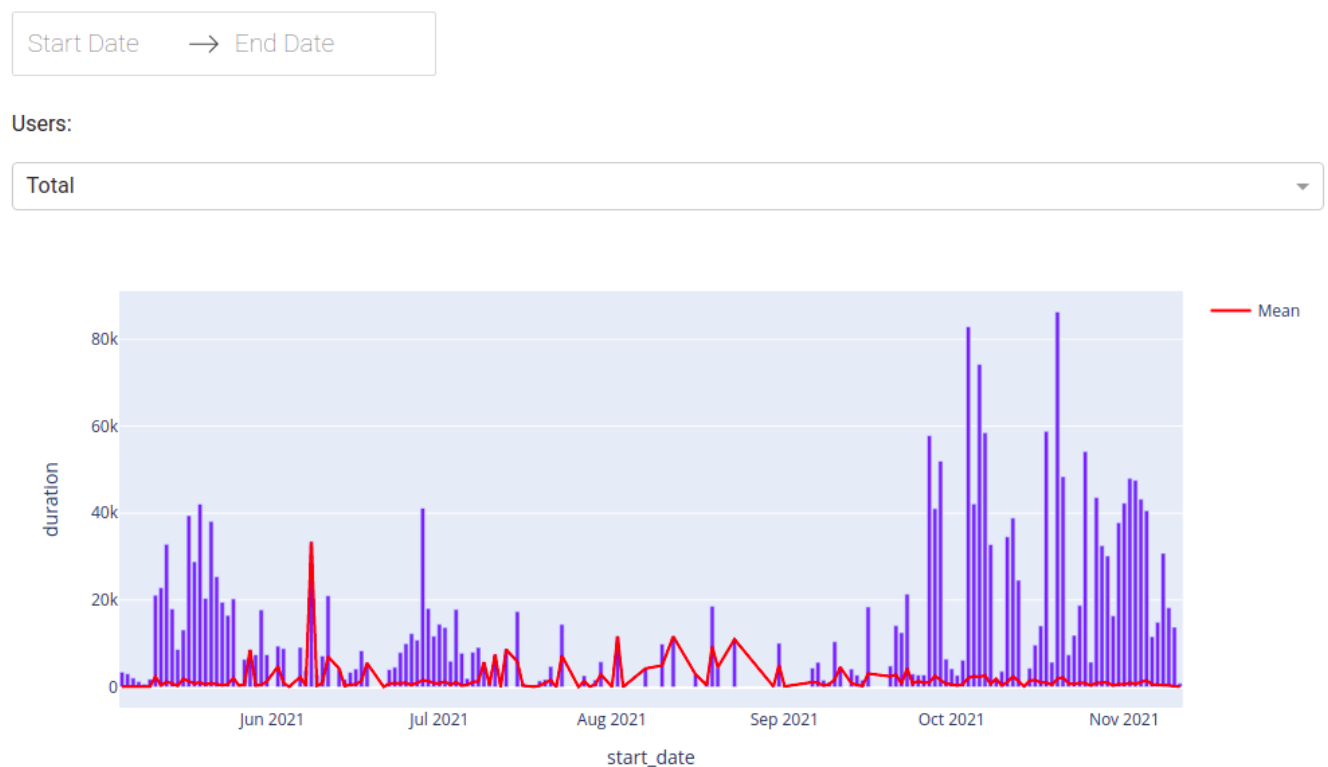


Figura 4.6: Gráfico de tiempo en Unibotics

Esta gráfica se puede filtrar tanto por fechas como por usuarios, pudiendo ver el tiempo dedicado de un usuario y la media total del tiempo que usa Unibotics dicho usuario, como se puede ver en la figura 4.7.



### 4.3. VALIDACIÓN EXPERIMENTAL

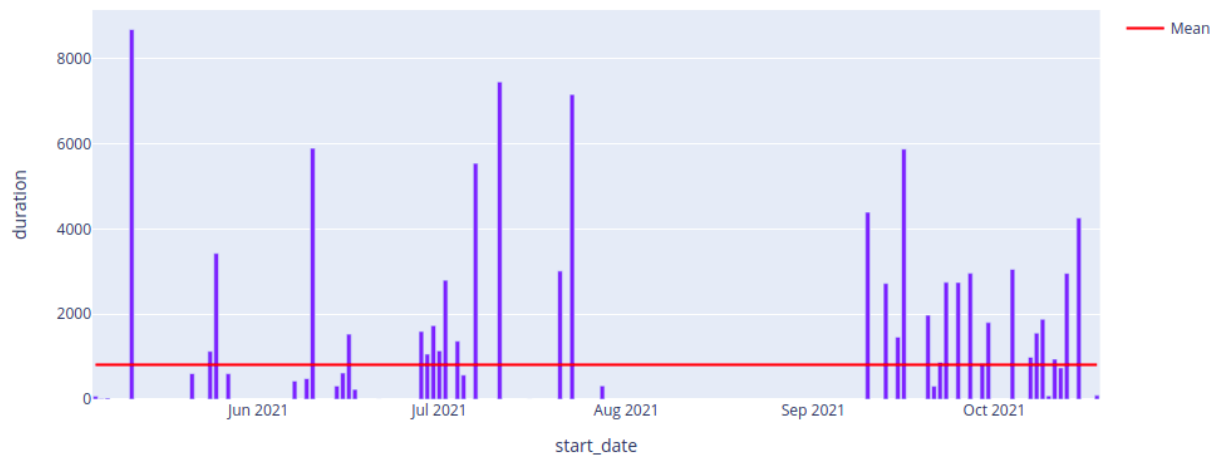


Figura 4.7: Gráfico de tiempo en Unibotics de un usuario

A fin de poder hacer un análisis de la media, la desviación típica o la moda se ha creado un histograma de las duraciones de las sesiones como se ve en la figura 4.7.

#### Time Histogram

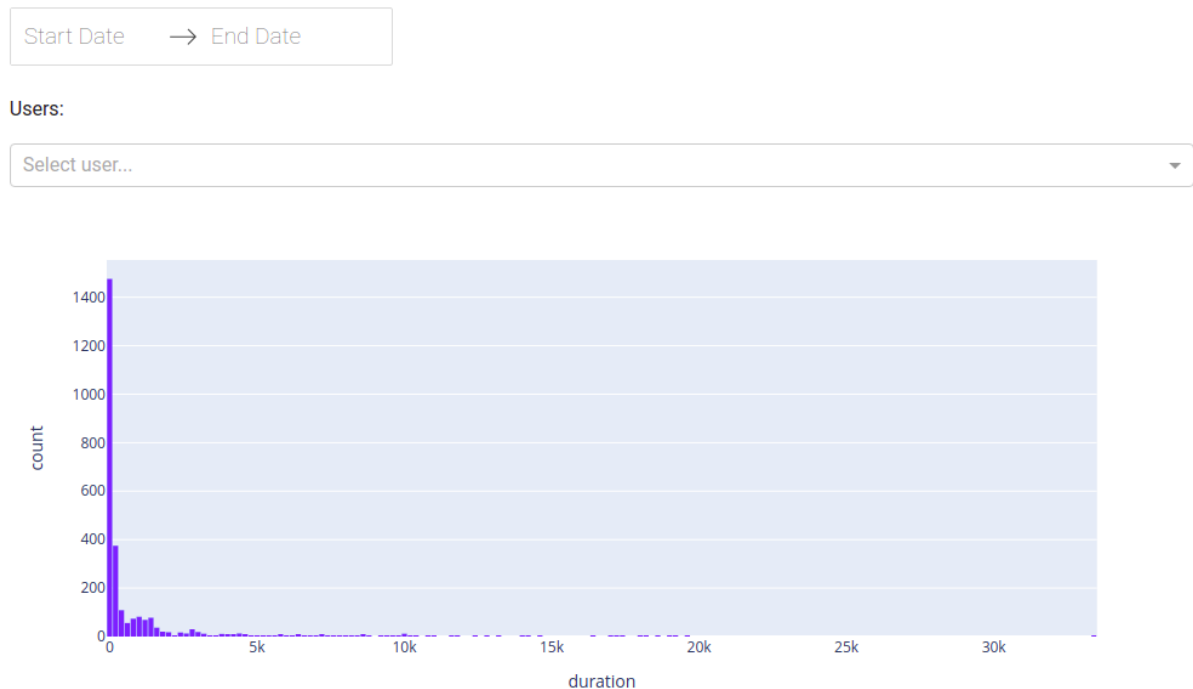


Figura 4.8: Histograma de las duraciones de sesiones

### 4.3. VALIDACIÓN EXPERIMENTAL

---

La figura 4.9 muestra una gráfica con los distintos navegadores que utilizan los usuarios para acceder a Unibotics, en formato porcentaje. En este caso se ve que una gran parte de inicio de sesiones es a través del navegador de Chrome.

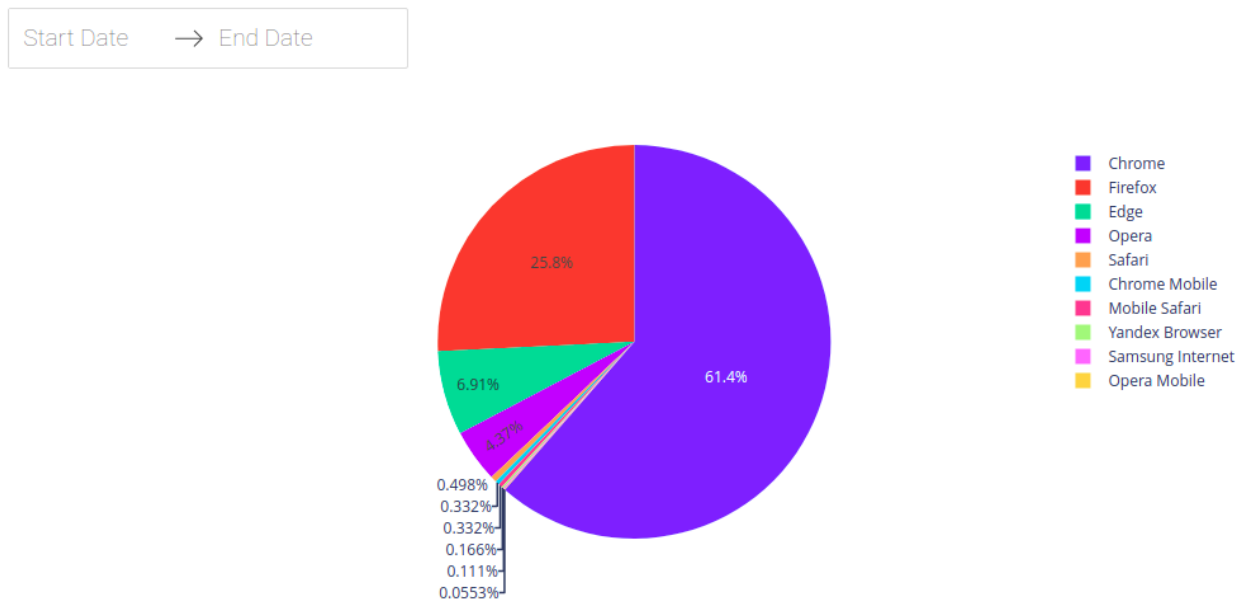


Figura 4.9: Gráfico de navegadores

### 4.3. VALIDACIÓN EXPERIMENTAL

---

Se ha creado una gráfica similar a la anterior para representar los Sistemas Operativos utilizados por los usuarios que acceden a Unibotics, como se puede ver en la figura 4.10. Ambas gráficas tienen filtro por fechas.

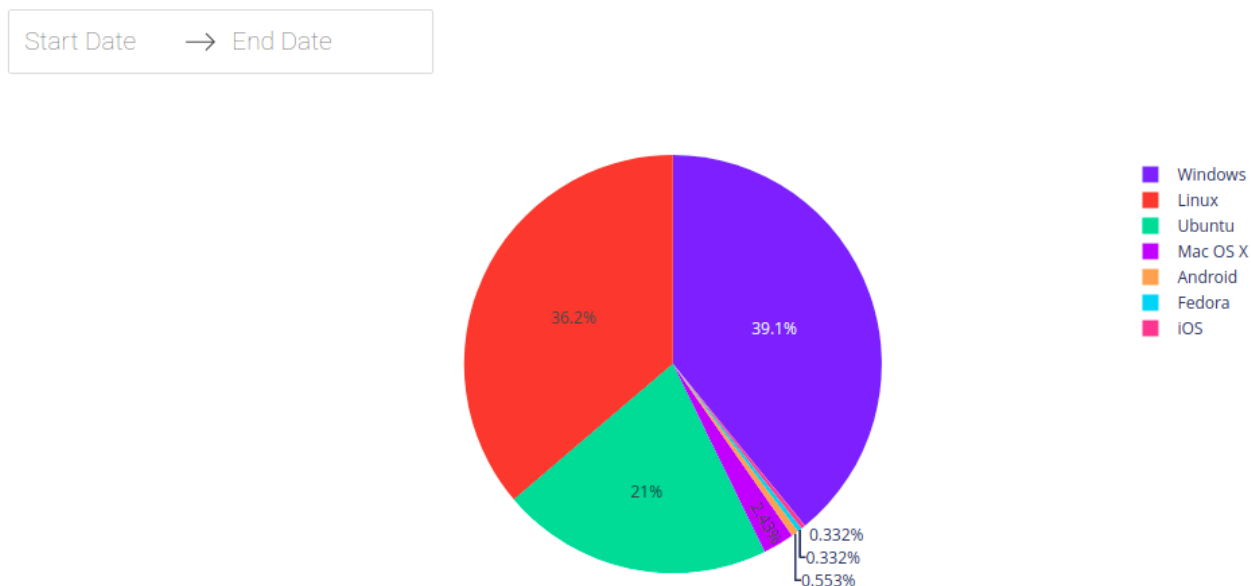


Figura 4.10: Gráfico de sistemas operativos

### 4.3. VALIDACIÓN EXPERIMENTAL

---

En la figura 4.11 se muestra un mapa geográfico con la localización de los usuarios que acceden a Unibotics. El tamaño de los puntos depende de la cantidad de sesiones, cuanto mayor sea el punto más sesiones hay. A la derecha se encuentra una leyenda con los países, se puede seleccionar uno o varios países en la leyenda para que solo se vean ellos en el mapa. Se puede filtrar por fechas.

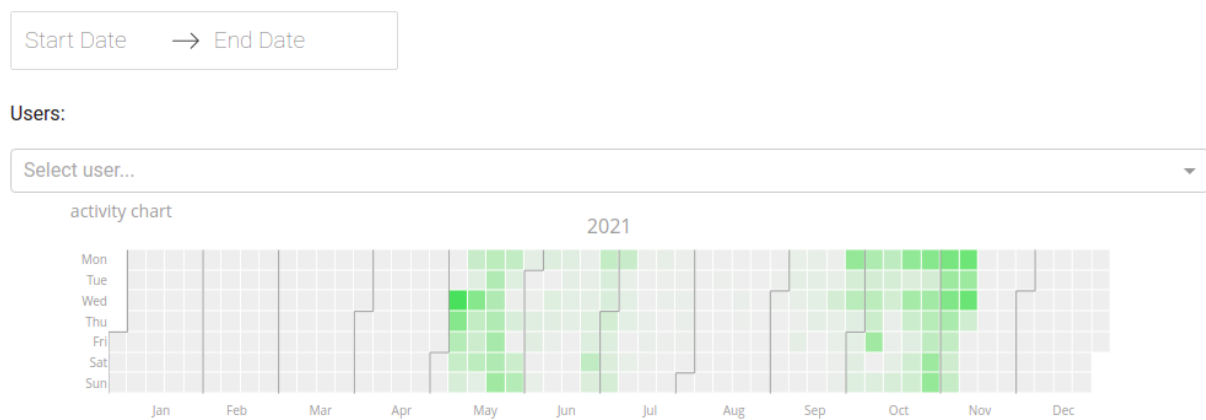


Figura 4.11: Mapa geográfico de sesiones

### 4.3. VALIDACIÓN EXPERIMENTAL

En la siguiente gráfica que se ve en la figura 4.12 representa un mapa de calor con la actividad de los usuarios. Esta dividido en cuadrados que representan cada día de un año, cuanto más intenso es el color verde más sesiones ha habido, ha medida que disminuyen las sesiones la intensidad también baja. A parte del filtrado por fechas. tiene un filtro por usuario para ver la actividad por usuarios únicos o grupo de ellos. Como ocurría con la gráfica lineal de sesiones, el primer mapa de color cuenta el total de las sesiones y el segundo mapa de color cuenta las sesiones únicas por usuario.

#### Total users activity



#### Total activity of unique users

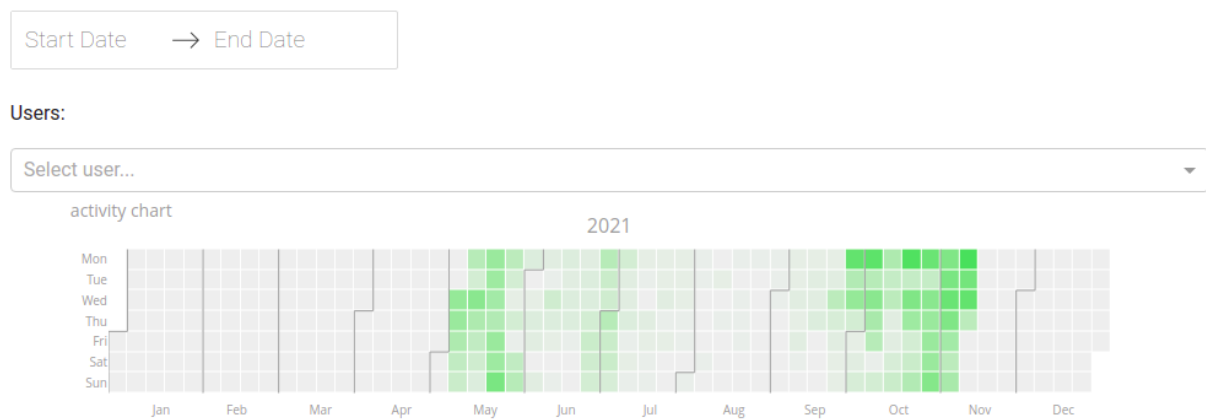


Figura 4.12: Mapas de calor sesiones

### 4.3. VALIDACIÓN EXPERIMENTAL

Los administradores tienen la opción de poder ver el número total de usuarios que son activos, el número de usuarios que han sido activos en los últimos 2 meses, en formato número. Además se muestra las gráficas de linea de registros por cada día, registros acumulados por días (figura 4.14) y usuarios activos en los últimos 2 meses.(figura 4.15) Cada día se comprueba cuantos usuarios han iniciado sesión desde 2 meses atrás hasta ese día concretamente. Cada una de estas gráficas contiene su propio filtrado por fechas.

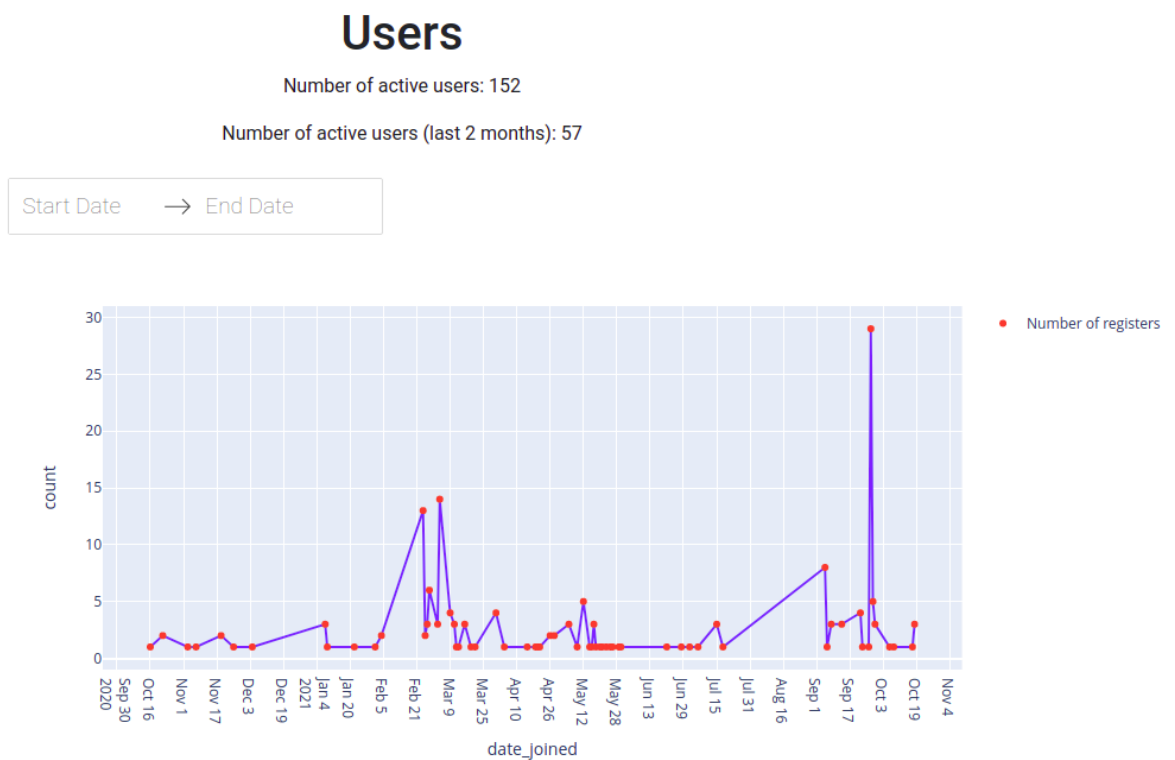


Figura 4.13: Gráficas relativas a los usuarios

### 4.3. VALIDACIÓN EXPERIMENTAL

#### Accumulated user registration

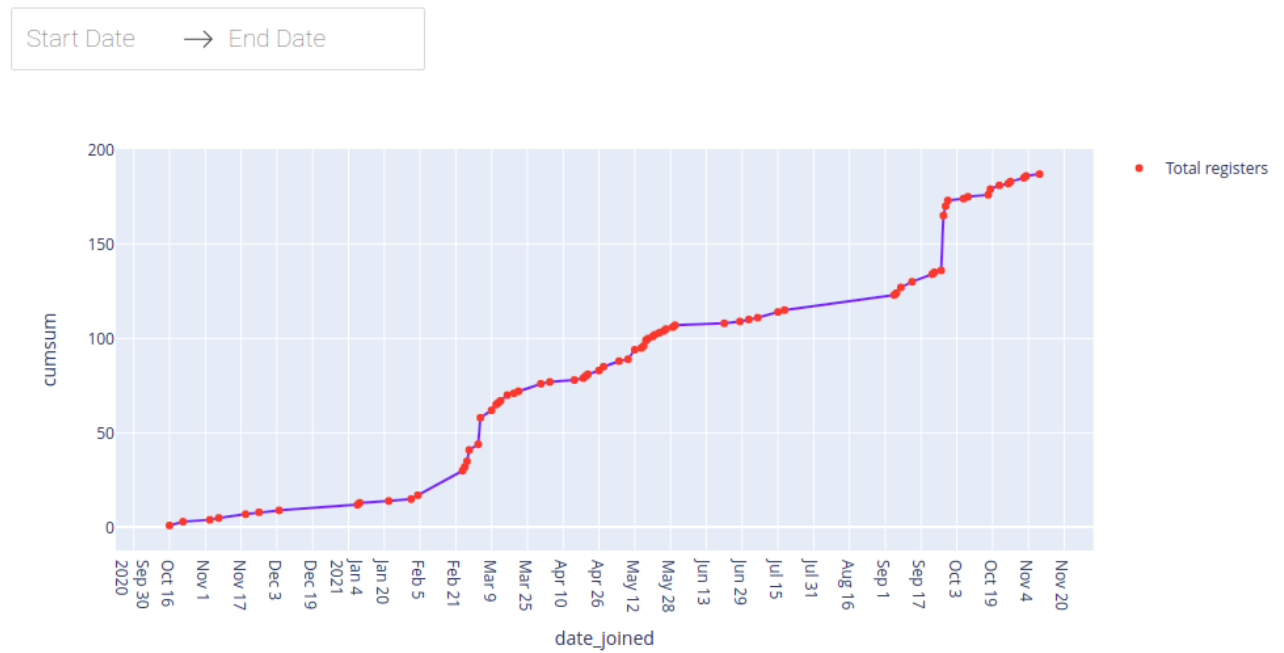


Figura 4.14: Gráfica registro de usuarios acumulado

#### Active users

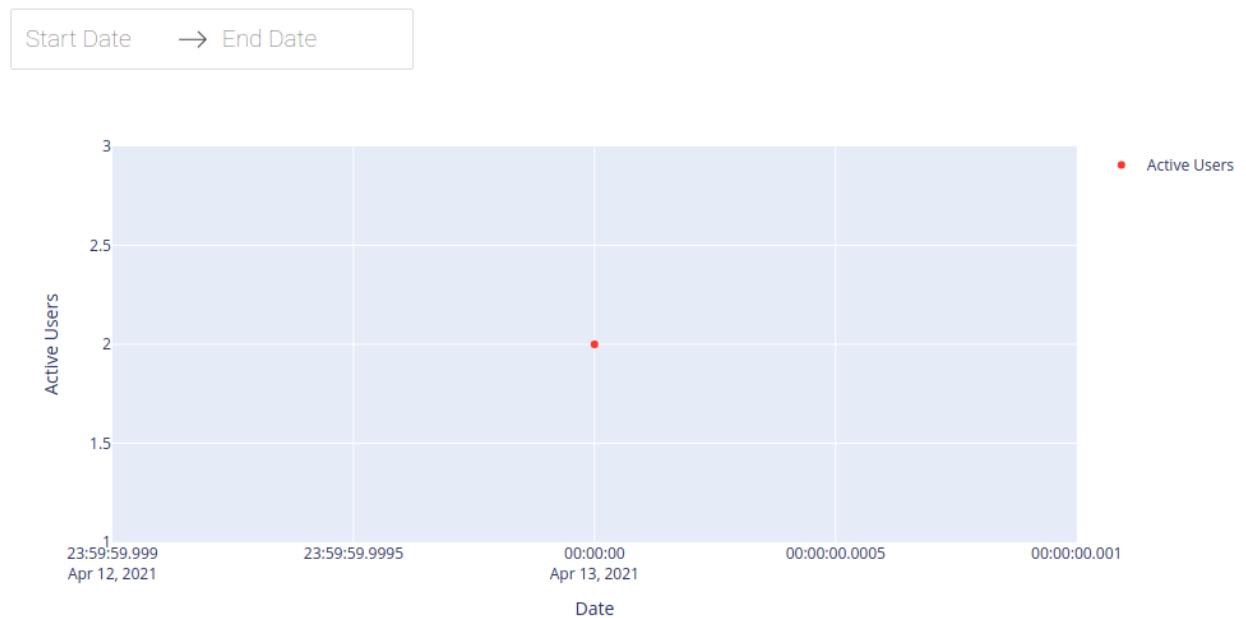


Figura 4.15: Gráfica de usuarios activos cada día

### 4.3. VALIDACIÓN EXPERIMENTAL

Con el propósito de conocer el tiempo que invierten los usuarios en cada ejercicio se ha elaborado un histograma de las duraciones de los ejercicios. Esta gráfica dispone de un filtro del ejercicio que se desea comprobar, con el filtro de usuario y el de fechas. Por ejemplo, la figura 4.16 es un histograma del ejercicio *follow\_line* de un grupo de usuarios y unas fechas concretas.

#### Time histogram in an exercise

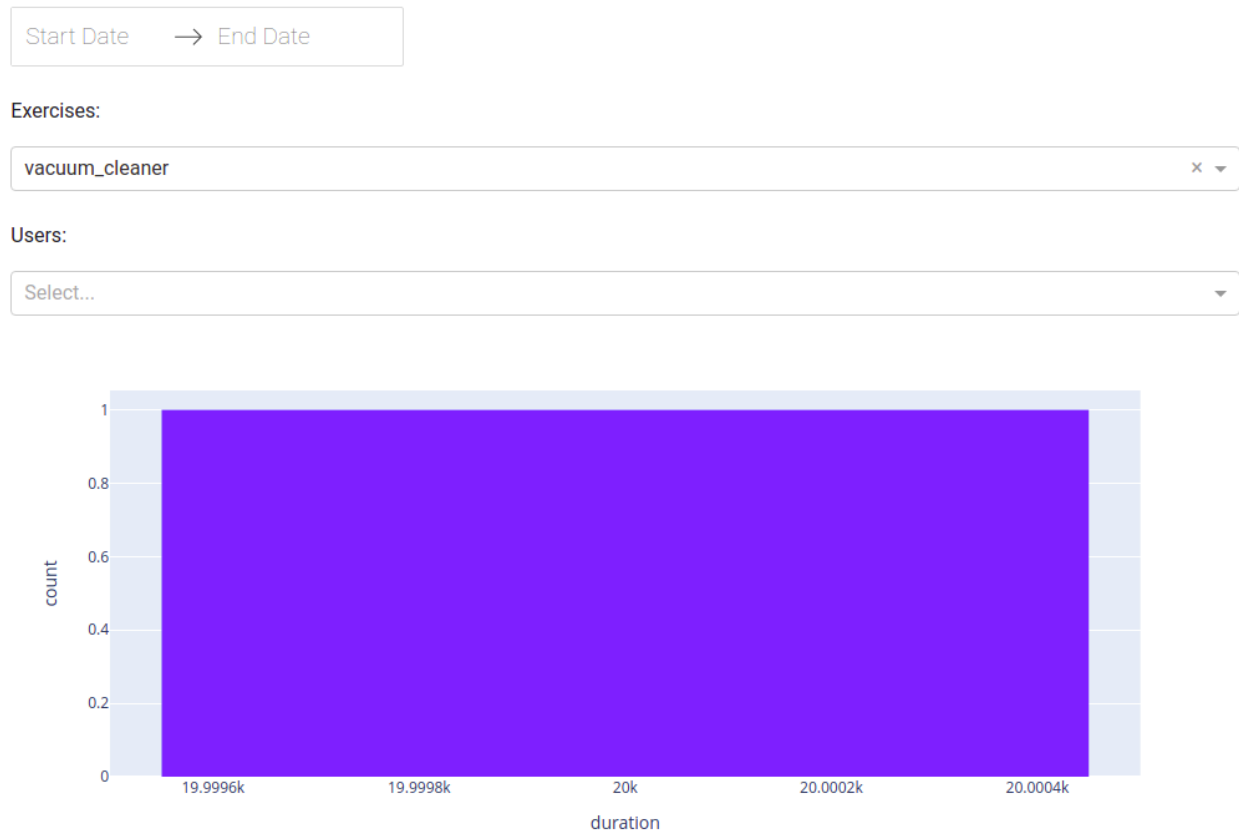


Figura 4.16: Histograma del ejercicio *follow\_line* de un grupo de usuarios



### 4.3. VALIDACIÓN EXPERIMENTAL

---

Las siguientes gráficas son las puntuaciones de estilo y eficacia que podrán ser vistas tanto por los usuarios (que pueden acceder a sus puntuaciones) como por los administradores (que pueden acceder a las puntuaciones de todos los usuarios). Actualmente las evaluaciones solo están disponibles en cuatro ejercicios representadas en una gráfica, en la que cada punto es una evaluación solicitada por el usuario. Las gráficas mostradas en las figuras 4.17 y 4.18 son un ejemplo de las notas de estilo y eficacia de un usuario de la base de datos de prueba en el ejercicio *follow\_line*. Las gráficas de los demás ejercicios son iguales, como las de las puntuaciones de eficacia.

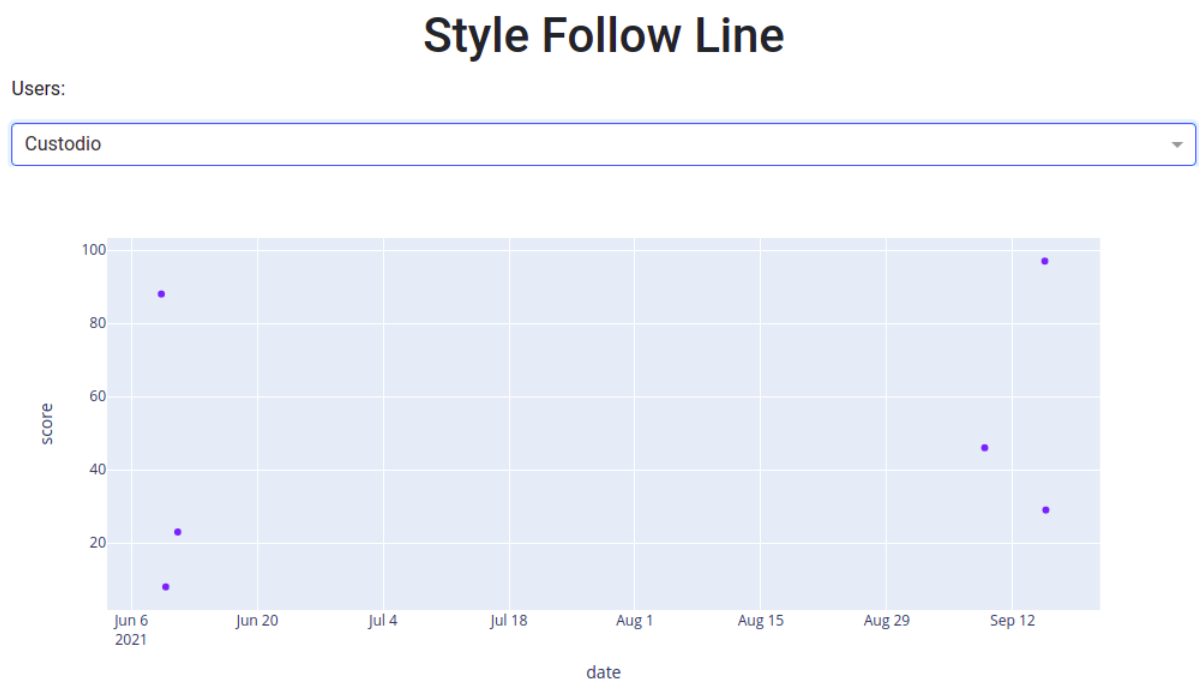


Figura 4.17: Gráfica de puntuación de estilo

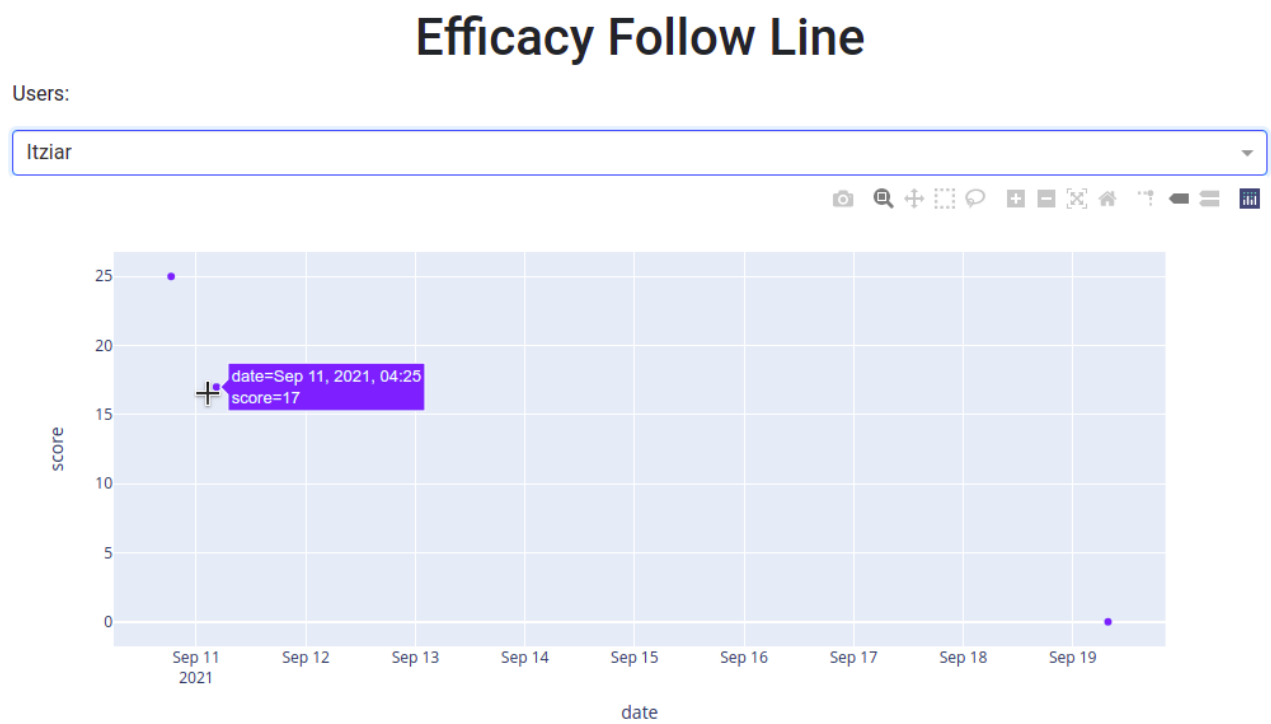


Figura 4.18: Gráfica de puntuación de eficacia

# Capítulo 5

## Conclusiones y trabajos futuros

En este último capítulo se manifiesta las conclusiones alcanzadas, así como las competencias adquiridas al realizar este TFG y futuros trabajos.

### 5.1. Conclusiones finales

En conclusión el objetivo principal de este TFG se ha cumplido con éxito, el cual consistía en integrar analíticas automáticas en Unibotics. Gracias a que los usuarios tienen que iniciar sesión cuando quieren ver las analíticas, se ha podido detectar si se trata de un administrador o no y así mostrar diferentes contenidos, esto era parte del objetivo principal que se menciona en la subsección 2.1. Analizando los subjetivos llegamos a las siguientes conclusiones:

- Se ha conseguido recoger sondas en Unibotics. Con la utilización de la herramienta de Elasticsearch se ha podido capturar y guardar las interacciones de los usuarios en la plataforma. Además se ha creado un nuevo botón en los ejercicios para poder recoger las sondas relativas a la puntuación de la eficacia del código. A parte de la sonda mencionada anteriormente, se han recogido las sondas relativas al inicio y fin de sesión, de los ejercicios y de la puntuación de estilo.
- Se ha podido integrar las gráficas dinámicas gracias al *framework* Dash, que ha permitido de una manera sencilla y potente la visualización de las sondas recogidas con Elasticsearch. En estas visualizaciones se han añadido filtros para un análisis más detallado.

### 5.2. Competencias adquiridas

Durante la realización del TFG he adquirido las siguientes competencias:

- Ampliado mis conocimientos sobre las tecnologías web, tanto el *framework* de Django como HTML, CSS y JavaScript.
- Conocer como funciona una base de datos en un proyecto real, en el caso de MySQL y saber desplegar e integrar una nueva base de datos, Elasticsearch. Importancia de la información para monitorizar un servicio web.
- Comprender las tecnologías de visualizaciones automáticas de la información gracias a Dash, la cual es una herramienta rápida y eficiente.
- Aprender a utilizar GitHub como repositorio donde desarrollar proyectos en equipo, haciendo uso de incidencias y parches.
- Trabajar en una plataforma que esta en continuo desarrollo, donde se han creado nuevas funcionalidades y trabajar en un equipo gracias a diferentes plataformas como puede ser Slack como método de comunicación.

### 5.3. Trabajos futuros

En esta sección se proponen futuras línea de trabajo para poder mejorar las analíticas automáticas:

- Enriquecer las sondas que se encuentran en Unibotics añadiendo nuevas o añadiendo nuevos campos a los índices creados para recabar más información sobre el uso de la plataforma. Actualmente las sondas de puntuación de estilo y de eficacia solo se encuentran en cuatro ejercicios, así que se podrían añadir a los nuevos ejercicios que sean creados añadiéndoles un evaluador de eficacia automático que algunos no poseen. Para recoger nuevas sondas será necesario la creación de nuevos índices en Elasticsearch y nuevas visualizaciones en la aplicación de Dash.

- Incorporar a Elasticsearch medidas de rendimiento en el servidor web. El servidor esta corriendo en *Amazon web services* en producción por lo que se pueden añadir métricas de memoria ocupada o CPU consumida entre otras, luego se podrá correlar esas métricas con las sondas actuales tales como el número de usuarios activos en la plataforma.
- En este TFG se ha visualizado las sondas con los datos directamente recogidos, el siguiente paso que se podría hacer es correlar las sondas recogidas, es decir, por ejemplo saber si el tiempo en el que realizan un ejercicio afecta la nota obtenida en el ejercicio. Esto nos permite hacer análisis estadísticos o investigaciones científicas como por ejemplo el efecto de la *gameficación*.

# Bibliografía

- [1] Juan González Gómez. *Introducción a las tecnologías web*. 2018. URL: <https://github.com/myTeachingURJC/2018-19-CSAAI/wiki/Introducci%C3%B3n-a-las-tecnolog%C3%ADas-web> (visitado 15-09-2021).
- [2] James Sanchez. *Laravel: ventajas del framework PHP de moda*. 2016. URL: <https://www.freelancer.es/community/articles/ventajas-del-framework-moda-laravel> (visitado 25-10-2021).
- [3] Carlos Travieso Merino. “Scratch”. En: *Crisalis* (2012). URL: <http://static.esla.com/img/cargadas/2267/Documentaci%C3%B3n%20Scratch.pdf> (visitado 24-11-2021).
- [4] Marcos Merino. “Seis aplicaciones gratuitas para aprender robótica y programación”. En: *Genbeta* (2020). URL: <https://www.genbeta.com/desarrollo/seis-aplicaciones-gratuitas-para-aprender-robotica-programacion> (visitado 15-09-2021).
- [5] *Why ROS?* 2021. URL: <https://www.ros.org/blog/why-ros/> (visitado 24-11-2021).
- [6] Raúl García Jerez. “Integración de Planificación Automática y ROS para el control autónomo de dos robots en el juego del Sokoban”. Tesis doct. Escuela Politécnica Superior. Universidad Carlos III de Madrid, 2014.
- [7] *CoppeliaSim*. URL: <https://www.coppeliarobotics.com/> (visitado 24-11-2021).
- [8] Marian Körber y col. *Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning*. 2021. arXiv: 2103.04616 [cs.RO].

- [9] Javier Velasco Seguido-Villegas. “Análisis y comparación de las principales plataformas de simulación robótica y su integración con ROS”. Sep. de 2019. URL: <http://oa.upm.es/56724/>.
- [10] *Gazebo*. URL: <http://gazebo-sim.org/> (visitado 24-11-2021).
- [11] *Riders.ai*. URL: <https://riders.ai/about/courses/intro-to-robotics#aboutus/> (visitado 24-11-2021).
- [12] JdeRobot organization. *Robotics Academy*. URL: <http://jderobot.github.io/RoboticsAcademy/> (visitado 27-09-2021).
- [13] *Conceptos básicos de HTML*. URL: [https://developer.mozilla.org/es/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics) (visitado 28-09-2021).
- [14] Uniwebsidad. *Etiquetas y atributos*. URL: <https://uniwebsidad.com/libros/xhtml/capitulo-2/etiquetas-y-atributos> (visitado 28-09-2021).
- [15] Juan González Gómez. *Sesión 2: HTML*. 2018. URL: <https://github.com/myTeachingURJC/2018-19-CSAAI/wiki/Sesi%C3%B3n-2:-HTML> (visitado 28-09-2021).
- [16] Juan González Gómez. *Sesión 3: CSS*. 2018. URL: <https://github.com/myTeachingURJC/2018-19-CSAAI/wiki/Sesi%C3%B3n-3:-CSS#reglas-de-aplicaci%C3%B3n-del-estilo> (visitado 28-09-2021).
- [17] Juan González Gómez. *Sesión 4: Introducción a Javascript*. 2018. URL: <https://github.com/myTeachingURJC/2018-19-CSAAI/wiki/Sesi%C3%B3n-4:-Introducci%C3%B3n-a-Javascript> (visitado 29-09-2021).
- [18] *JavaScript*. URL: <https://developer.mozilla.org/es/docs/Learn/JavaScript> (visitado 29-09-2021).
- [19] Django. *Modelos*. URL: <https://docs.djangoproject.com/en/2.2/topics/db/models/> (visitado 29-09-2021).
- [20] W3school. *Introducción a SQL*. URL: [https://www.w3schools.com/sql/sql\\_intro.asp](https://www.w3schools.com/sql/sql_intro.asp) (visitado 29-09-2021).

- [21] Micho García. *Conceptos básicos de SQL*. 2013. URL: [https://postgis.readthedocs.io/es/latest/conceptos-sql/conceptos\\_sql.html#](https://postgis.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html#) (visitado 29-09-2021).
- [22] Victor Cuervo. *¿Qué es Elasticsearch?* 2019. URL: <https://www.arquitectoit.com/elasticsearch/que-es-elasticsearch/> (visitado 04-10-2021).
- [23] *¿Qué es Elasticsearch?* URL: <https://www.elastic.co/es/what-is/elasticsearch> (visitado 04-10-2021).
- [24] Victor Cuervo. *Conceptos Básicos Elasticsearch*. 2019. URL: <https://www.arquitectoit.com/elasticsearch/conceptos-basicos-elasticsearch/> (visitado 04-10-2021).
- [25] *Introduction to Dash*. URL: <https://dash.plotly.com/introduction> (visitado 04-10-2021).