Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Ruiz Ferrer, Sergio; Garcia Font, Victor, dir. TwitchAdc: Sistema de soporte a la toma de decisiones sobre creación de contenido en Twitch. 2023. (Enginyeria Informàtica)

This version is available at https://ddd.uab.cat/record/280683

under the terms of the BY-NC-ND license

TwitchAdc: Sistema de soporte a la toma de decisiones sobre creación de contenido en Twitch

Sergio Ruiz Ferrer

Resumen– Twitch.tv es una página de retransmisiones en vivo que ha incrementado mucho su popularidad en los últimos años. Sin embargo, a medida que ha aumentado su cantidad de espectadores, estos no se han distribuido equitativamente entre los diferentes creadores, sino que se han concentrado en unos pocos, derivando en dificultades para los nuevos creadores que llegan a la plataforma. Páginas de exposición de estadísticas como Twitchtracker o Stream Charts contribuyen a la problemática a través de opciones de pago y de una exposición masiva de datos sin una interpretación asociada difíciles de utilizar para la mayoría, ambos factores que generan que no todos los creadores de contenido se puedan beneficiar de estas plataformas. Este artículo presenta el proceso de desarrollo de un sistema de soporte a la toma de decisiones sin opciones de pago y con datos fácilmente interpretables, que fomenta la igualdad de oportunidades entre los creadores.

Palabras clave– Twitch, retransmisión en vivo, análisis de datos, toma de decisiones, ausencia de pagos, ayuda a nuevos creadores.

Abstract– Twitch.tv is a live streaming platform that has greatly increased its popularity in recent years. However, as the number of viewers has grown, they have not been distributed evenly among different creators, but rather concentrated on a few, resulting in difficulties for new creators entering the platform. Statistics showcase platforms like Twitchtracker or Stream Charts contribute to the issue through paid options and massive data exposure without associated interpretation. These factors prevent all content creators from benefiting from these platforms. This article presents the development process of a decision support system without payment options and with easily interpretable data that promotes equal opportunities among the different content creators.

Keywords– Twitch, live streaming, data analysis, decision-making, absence of payments, support for new creators.

1 Introducción

TWITCH.TV [1] es una plataforma de transmisiones en vivo que fue creada en 2011 y que en los últimos años ha incrementado de forma exponencial su número de espectadores. Dicho incremento ha generado inevitablemente un crecimiento en el número de creadores de contenido, que compiten por la atención del público en un esfuerzo por convertir la creación de contenido digital en su principal ocupación y fuente de ingresos. No obstante, gran parte de ellos no consigue generar resultados satisfactorios y obtener una cantidad suficiente de espectadores a pesar de sus esfuerzos. Esto se debe a que la mayoría del público de la plataforma se concentra en tan solo unos pocos creadores de contenido.

Hay varios factores desequilibrantes que pueden dar explicación a la problemática anterior, pero el más relevante es el que comprende a plataformas como twitchtracker.com [2], Stream Charts [3] y sus derivadas; las plataformas de proporción de estadísticas.

¿Por qué contribuyen al problema estas plataformas? La razón es simple: no proporcionan la misma utilidad a todos los creadores debido a la necesidad de pagos por parte del usuario y a la dificultad de extraer conocimiento útil de los datos que se exponen a causa de la enorme cantidad de ellos y a la ausencia de explicaciones asociadas a los mismos.

El objetivo de este proyecto es crear una plataforma similar que le ofrezca a todos los creadores de contenido de la plataforma por igual datos útiles para tomar decisiones

• Email de contacto: sergio.ruizf@autonoma.cat
• Mención realizada: Tecnologías de la Información
• Trabajo autorizado por: Victor Garcia Font (Departamento de Ingeniería de la Información y de las Comunicaciones Area de Ciencias de la Computación e Inteligencia Artificial)
• Curs 2022/23

sobre su creación de contenido, sin necesidad de pagos.

El resto de este informe se estructura de la siguiente manera. El capítulo 2 describe el estado del arte y las plataformas ya existentes. El capítulo 3 describe los objetivos del proyecto. El capítulo 4 describe la metodología y la planificación. A continuación, las diferentes fases del proyecto (Análisis, Diseño, Implementación y Testeo) son descritas en los capítulos 5, 6, 7 y 8 respectivamente. Habiendo explicado el desarrollo del proyecto, el capítulo 9 introduce los resultados y discute el cumplimiento de los objetivos. Para terminar, los capítulos 10 y 11 exponen las conclusiones y los futuros pasos para el proyecto. Por último, los agradecimientos cierran este documento.

2 Estado del arte

Como se ha mencionado en el apartado anterior, ya existen plataformas similares a la que será desarrollada en el marco de este proyecto. Los dos principales exponentes de este tipo de plataformas son TwitchTracker y Stream Charts.

Por un lado, TwitchTracker ofrece rankings en los que apreciar quiénes son los creadores líderes de la plataforma en base a diferentes criterios (número de espectadores habitual, número de suscriptores, horas retransmitidas al mes, etc). Además de esto, permite acceder a una página para cada creador de contenido de la plataforma, que incluye una serie de datos de interés sobre este creador y un histórico que permite intuir si su situación mejora o empeora y a qué ritmo. También permite hacer una división por categorías, y saber datos relevantes sobre cada categoría o juego que se puede retransmitir en Twitch, entre otras cosas. Por desgracia, la enorme cantidad de datos que ofrece puede ser abrumadora, especialmente para para cualquiera que no esté acostumbrado a la exposición a tales volúmenes de datos. Se podría decir que Twitchtracker ofrece datos en bruto que pueden ser utilizados de forma correcta por unos pocos con la capacidad para ello, pero no por el público generalizado.

Por otro lado, Stream Charts ofrece datos muy similares sólo que expuestos en distintos formatos. Su situación es prácticamente la misma que la de su competidora: no es fácil de usar de forma efectiva para el público general. Esto se debe a que presenta una gran cantidad de datos sin explicaciones asociadas, lo cual dificulta a los usuarios encontrar la información que buscan, y también utilizarla una vez la encuentran. Además, Stream Charts ofrece una versión premium de pago que genera todavía más inaccesibilidad a los creadores de contenido que empiezan en la plataforma y carecen de un gran número de espectadores, en este caso una inaccesibilidad por el lado económico.

En resumen, las plataformas existentes no tienen un propósito concreto, sino que presentan ingentes cantidades de datos sin explicaciones (o con escasas explicaciones) asociadas, lo cual dificulta al usuario encontrar la información que necesita en cada momento, y entenderla una vez la encuentra. Además, existen necesidades de pago que imposibilitan el uso completo de algunas de estas plataformas a todos los creadores.

Por lo tanto, este proyecto será desarrollado de forma que presente concretamente los datos que los usuarios necesiten en cada momento. En vez de proveer una gran cantidad de datos y dejar al usuario buscar lo que necesita, se le pedirá que estableza diversos criterios respecto a cómo quiere crear contenido (por ejemplo, en qué categoría, en qué idioma, o en qué horario quiere hacerlo) y se le proveerá la cantidad exacta de datos necesaria para facilitar su toma de decisiones, evitando de esta forma un exceso de datos que dificulten su comprensión. Los datos provistos como respuesta a las peticiones del usuario estarán asociados con el número de espectadores en relación con los criterios escogidos (media de espectadores por categoría, horas con más espectadores para un idioma, etc), de manera que el usuario pueda maximizar su número de espectadores desviándose lo mínimo posible de lo que realmente quiere hacer. Todo esto sin necesidad de que el usuario realice ningún tipo de pago para obtener ventajas de la plataforma, fomentando así una igualdad de posibilidades en cuanto a acceso a la información para todos los creadores de contenido de Twitch.

3 Objetivos

El objetivo de este proyecto es desarrollar una plataforma de soporte a la toma de decisiones sobre creación de contenido en Twitch que pueda satisfacer las necesidades de información de todos los usuarios. La máxima prioridad es la diferenciación respecto a plataformas ya existentes gracias a la explicación y presentación sencilla de datos útiles y a la ausencia de opciones de pago. Esto es así porque, de no cumplir con estas características, el proyecto no resolverá la problemática presentada en los puntos anteriores, y será, por lo tanto, una versión inferior a las plataformas que ya se dedican a esto. Para poder garantizar que eso no suceda, un elemento clave es la automatización, que evitará la necesidad de contratar a gente para que realice tareas de análisis manuales (cosa que dificultaría la ausencia de cobros a los usuarios), y favorecerá la escalabilidad del proyecto.

Con esto establecido, se apuntan a continuación los objetivos principales del proyecto junto a otros subobjetivos.

- Que el proyecto se pueda mantener en funcionamiento por sí mismo, sin necesidad de cobrar a los usuarios (**Prioridad máxima**).
- Proveer a los usuarios de informaciones útiles y competitivas a partir del análisis de datos accedidos de forma gratuita (**Prioridad máxima**).
- Presentar sólo los datos que el usuario necesita junto a una explicación de éstos para asegurar que los usuarios encuentran la información que necesitan y la interpretan correctamente. (**Prioridad máxima**).
- Que cada una de las fases del proyecto (extracción, inserción, análisis y representación de los datos) se realice de forma automatizada, sin la intervención manual de ninguna persona.(**Prioridad máxima**).
- Crear un módulo de extracción de datos de la API de Twitch e inserción a una base de datos (**Prioridad alta**).
- Crear una base de datos capaz que permita almacenar los datos extraídos de Twitch en el formato utilizado por el módulo de extracción. (**Prioridad alta**).

- Crear una página web que permita a sus usuarios realizar consultas respecto a qué, cómo y cuándo retransmitir en la plataforma (**Prioridad alta**).
- Presentar todos y cada uno de los datos extraídos de forma gráfica en la página web para así facilitar su interpretación y entendimiento (**Prioridad alta**).
- Se ha de poder ejecutar cada funcionalidad de la página web con un tiempo de carga inferior a 2 segundos. (**Prioridad alta**).
- La plataforma web ha de estar protegida ante los ataques básicos: SQL injection, cross-site scripting y cross-site request forgery (**Prioridad media**).
- La página web ha de constar de una versión en inglés y otra en español. (**Prioridad baja**).
- Obtener feedback de creadores de contenido de la plataforma Twitch sobre qué datos les podrían ser útiles a través de encuestas (**Prioridad baja**).

4 METODOLOGÍA

Para el proyecto se hará uso de una implementación específica de un tipo de metodología iterativa incremental. Se apuntan a continuación las características principales de dicha metodología:

- El proyecto a nivel de tiempo se divide por semanas; cada semana tendrá asignadas hasta 2 tareas.
- En el caso de haber 2 tareas asignadas a la misma semana, una será principal (P) y la otra secundaria (S).
- La división de tiempo entre tarea principal y secundaria será del 70 – 30 aproximadamente.
- El proyecto a nivel de trabajo se divide en actividades; cada actividad corresponde a un bloque importante para la conclusión satisfactoria del proyecto.
- Las actividades son las siguientes: la extracción de los datos de la API de Twitch, el almacenamiento de éstos en una base de datos, la limpieza (en caso de ser necesaria) y análisis de los datos almacenados, la representación gráfica de los datos y/o análisis y la creación de una plataforma web de soporte a la toma de decisiones donde exponer dichos datos y gráficos.
- Cada actividad del proyecto se dividirá en 3 tareas: su toma de decisiones o planteamiento (que incluyen análisis y diseño), su desarrollo y su testeo o revisión.
- Cada tarea de testeo incluirá también una revisión de la integración de la última actividad desarrollada con todas las anteriores.
- La última semana se reserva para un control de calidad final de todo el proyecto.

Esta metodología no pretende ser mejor a nivel general que otras metodologías ya establecidas y ampliamente reconocidas en el sector. No obstante, se ha considerado que teniendo en cuenta el alcance de este proyecto y buscando optimizar el uso del tiempo, es una metodología especialmente útil, ya que permite siempre tener otra tarea disponible si una se complica, y además habilita la posibilidad de escoger cada vez entre una tarea más técnica (de programación) y una menos técnica (de análisis, diseño o testeo), favoreciendo un rendimiento máximo.

Se puede ver la planificación inicial del proyecto en un diagrama de Gantt en el anexo A.1.

5 ANÁLISIS

En este apartado se describen los requisitos obtenidos a lo largo de las distintas fases de análisis del proyecto, derivados del análisis de la API de Twitch, de la opinión de diversos creadores de contenido interesados en el proyecto y de las necesidades derivadas de la propia naturaleza de los datos que se pretenden extraer.

5.1. Requisitos funcionales

- Ha de existir un módulo de gestión de datos que permita la comunicación con la API de Twitch a través de un token.
- El código de gestión de datos ha de poder almacenar los datos que reciba de la API de Twitch en un formato fácil de entender y manipular.
- El código de gestión de datos ha de poder extraer datos en tiempo real.
- El código de gestión de datos ha de obtener, como mínimo, datos sobre los streams, las categorías, los idiomas y los streamers de la API de Twitch.
- El código de gestión de datos ha de poder insertar los datos extraídos de la API de Twitch en una base de datos relacional.
- El código ha de controlar cadenas de texto que podrían incluir caracteres especiales que rompieran las inserciones a la base de datos.
- El código de gestión de datos ha de poder ser ejecutado de forma periódica a través de un scheduler.
- La base de datos ha de poder almacenar los datos en el formato devuelto por el módulo de gestión de datos.
- La base de datos ha de controlar de forma correcta la presencia de valores duplicados y nulos.
- La base de datos ha de incluir normas que mantengan la coherencia de los datos.
- La web ha de poder incluir representaciones gráficas de los datos.
- La web ha de permitir enviar formularios que no estén totalmente completados.
- La web ha de poder modificar los contenidos que muestra en función de lo que solicite el usuario.
- La web ha de tener secciones bien diferenciadas.

5.2. Requisitos no funcionales

- El módulo de gestión de datos ha de obtener los datos de la API de Twitch con el mínimo número de peticiones posible.
- La organización del módulo de gestión de datos ha de facilitar su entendimiento y uso, su testeo y la creación de nuevas funciones de extracción y subida de datos.
- La web ha de ser simple y fácil de navegar.
- La web ha de tener un sistema de colores que resalte los datos más relevantes para el usuario.
- La web ha de tener un bajo tiempo de respuesta y de recarga.

6 Diseño

En las secciones asociadas a este apartado se describen las tareas realizadas a lo largo de las diferentes fases de diseño del proyecto. Esto incluye el diseño de la arquitectura global del sistema, de la extracción y subida de los datos, de la base de datos y del backend y el frontend de la página web, en ese orden.

6.1. Arquitectura global

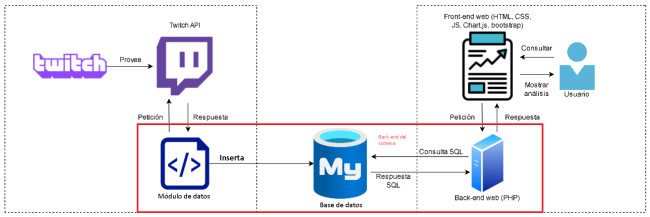

Fig. 1: Arquitectura global del proyecto

En la imagen anterior se puede ver un diagrama que representa la arquitectura del proyecto a nivel general, especificando cómo interactúan los diferentes elementos de ésta, cosa que describo con más detalle a continuación:

El código asociado a la extracción y subida de datos constará de dos tipos de funciones diferentes: funciones de extracción, que se comunicarán con la API de Twitch para hacerle peticiones y obtener datos, y funciones de subida, que se comunicarán con la base de datos del proyecto e insertarán los datos extraídos, que serán almacenados de forma indefinida para mantener un histórico de éstos. Esto aparece en un recuadro en el diagrama para indicar que es una parte del proyecto asociada más con la API de Twitch que con el usuario, para el cual es totalmente invisible.

Una vez haya datos almacenados, la base de datos se comunicará con el back-end de la página web, para proporcionarle los datos que el usuario necesite. El back-end y el front-end de la página web se comunicarán de forma bidireccional, ya que por un lado el usuario introducirá diferentes criterios que condicionarán los datos que aparecerán en la web, y por otro, el back-end, después de haber interactuado con la base de datos, le devolverá al usuario los datos que le interesen al usuario en función de dichos criterios, junto a una representación gráfica de éstos y una explicación. Esto aparece en un recuadro en el diagrama para indicar que es una parte del proyecto asociada más con el usuario que con la API de Twitch.

Ambas partes del sistema se relacionan a través de la base de datos, donde la parte asociada con la API de Twitch deposita los datos que luego recoge y utiliza la parte asociada con el usuario.

6.2. Extracción e inserción de datos

Para la extracción e inserción de los datos se ha generado un diseño que, sin dejar de lado la necesidad de que las funciones se realicen de forma automática, esencial en este proyecto, trata de optimizar dos aspectos: la facilidad para la ampliación de estas funciones cuando sea decidida la obtención de nuevos datos, y la capacidad de detección de errores.

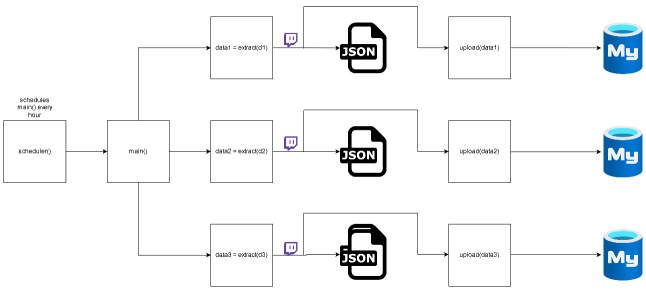

Fig. 2: Esquema de la extracción y subida de datos

La imagen anterior muestra el diseño de la extracción e inserción de los datos. Se puede observar que el esquema consta de una función principal que se ejecuta cada hora a través de un planificador, una función de extracción de datos que se ejecuta varias veces con distintos parámetros, y una función de inserción que recibe como parámetro los datos extraídos en las llamadas a la función de extracción y los inserta a la base de datos. Esta estructura permite una fácil ampliación de las tareas de extracción e inserción sin necesidad de conocer cómo funciona lo que se había implementado previamente, ya que independientemente de lo ya implementado, extraer e insertar nuevos datos requiere únicamente de añadir un nuevo caso a la función de extracción y un nuevo caso a la función de inserción.

Los archivos JSON que aparecen en el centro del diagrama tienen función doble: sirven como copia de seguridad de los datos extraídos en la última iteración y permiten la visualización rápida de los datos en el propio JSON o a través de plataformas que permiten visualizarlos de una forma tabular más clara, como Excel. El hecho de poder visualizar los datos así permite la detección rápida de ciertos errores que podrían darse en los datos, como la aparición de campos en blanco o de carácteres especiales no válidos.

6.3. Base de datos

El proyecto constará de una base de datos de tipo relacional, debido a su sencillez y seguridad, y a que encaja con el tipo de datos que usa este proyecto.

Fig. 3: Esquema de la base de datos

El diagrama anterior muestra el diseño de la base de datos, con las distintas tablas, atributos, claves primarias (marcadas en negrita) y relaciones.

En cuanto a decisiones de diseño que no aparecen en el diagrama, se ha decidido que lo más lógico es introducir primero las categorías, los idiomas y los streamers, y sólo después, los streams. Además, para garantizar la integridad de los datos, serán establecidas cláusulas a la tabla streams para controlar qué sucede cuando una entrada de una tabla es modificada o eliminada. Dichas cláusulas son:

3 clausulas ON DELETE RESTRICT, una por cada una de las otras tablas (languages, categories y streamers), que evitan que se puedan borrar un idioma, categoría o streamer mientras sigan existiendo en la tabla streams entradas que los incluyan.

3 cláusulas ON UPDATE CASCADE que aseguran que, en caso de que un streamer_id, un category_id o un language_code sean modificados en una de sus respectivas tablas, las columnas correspondientes en la tabla streams se actualicen automáticamente para corresponder a los cambios realizados.

6.4. Página web

El diseño de la página web es el más extenso de los diseños realizados, por lo que aparece dividido en casos de uso, backend y frontend para facilitar su comprensión.

6.4.1. Diagrama de casos de uso

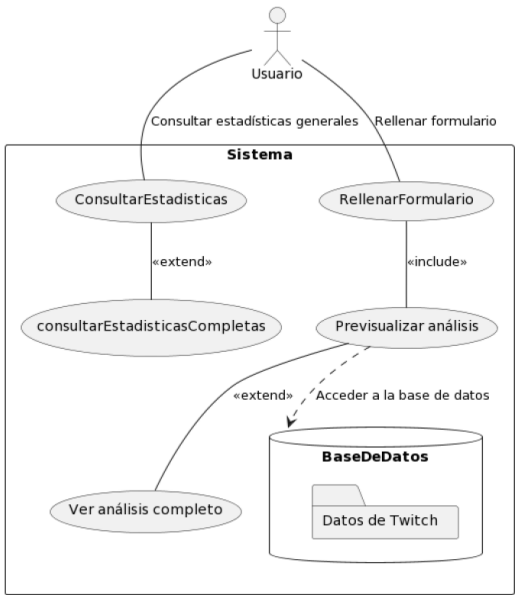

Fig. 4: Diagrama de casos de uso de la página web

En el diagrama de casos de uso de la figura anterior aparecen representadas las distintas acciones que puede llevar a cabo el usuario en la página web.

Por un lado, el usuario puede acceder a la página inicial, desde donde puede consultar estadísticas genéricas sobre Twitch y rellenar un formulario respecto a cómo le gustaría crear contenido en la plataforma (en qué idioma, de qué categoría, a qué hora...).

Si quiere seguir consultando más estadísticas genéricas, puede pulsar un botón y dirigirse a una página que contiene todas las estadísticas generales.

En caso de querer recomendaciones para su caso específico, puede rellenar el formulario. Esto le cargará una previsualización de los datos correspondientes a la respuesta a su consulta y, en caso de querer ver el análisis completo de la situación que ha introducido en el formulario, puede pulsar un botón que le llevará a otra página con la respuesta completa a su consulta.

Las especificaciones de los casos de usos presentes en la figura 4 pueden ser consultados en el anexo A.2.

6.4.2. Backend

Los archivos del backend se han estructurado siguiendo el modelo-vista-controlador (MVC). Es decir, cuando se quiere implementar una funcionalidad, se llama a un controlador, que a su vez llama a diversos modelos para que realicen la funcionalidad y finalmente llama a una vista para que exponga el trabajo realizado por los modelos.

En el caso de esta web, los modelos se encargan de la conexión a la base de datos y de realizar todo tipo de consultas para obtener valores mínimos, máximos, medias, etc de las diferentes tablas, y las vistas exponen los gráficos realizados con los datos obtenidos a partir de las consultas.

La principal razón para seguir este patrón es mantener un código ordenado y fácil de actualizar. En el momento de añadir nuevos datos y nuevas representaciones gráficas a la web, únicamente será requerido añadir una función al modelo que contiene las consultas a la base de datos y una llamada a la nueva función desde el controlador correspondiente, para lo cual no harán falta conocimientos amplios de cómo se ha implementado la web.

6.4.3. Frontend

El diseño de la página web va a consistir principalmente en 3 páginas.

La primera es la página principal. En esta página aparecerá un formulario con una serie de selectores. A la derecha habrá un apartado de la página donde aparecerá la previsualización parcial de la respuesta de la página a lo que rellene el usuario en el formulario, y en la parte de arriba una serie de estadísticas genéricas sobre la plataforma Twitch. En el lado izquierdo de la página aparecerá un índice con las diferentes páginas.

A la segunda página, la página de estadísticas generales, se podrá acceder o bien a través de un índice presente en el lado izquierdo o bien a través de un botón presente en la esquina superior derecha de la página. En esta segunda página se incluirán una serie de gráficas en tiempo real sobre todo tipo de datos sobre Twitch que puedan resultar relevantes a los creadores de contenido de la plataforma, como por ejemplo qué categorias están de moda o a qué horas hay más espectadores.

A la última página se podrá acceder al pulsar un botón en el apartado de previsualización de la página principal,

una vez el formulario ya haya sido enviado por el usuario. Aquí se incluirán gráficas y conjuntos de datos específicos para la consulta que haya realizado el usuario a través del formulario, junto a explicaciones y recomendaciones sobre las decisiones que puede tomar el usuario en la plataforma para tener mayores oportunidades de éxito desviándose lo mínimo posible de sus preferencias.

Cada una de las páginas mencionadas incluirá un header y un footer con el nombre de la página.

7 IMPLEMENTACIÓN

En este apartado se describen las tareas asociadas a la fase de implementación del proyecto. Es decir, se introducen las implementaciones específicas derivadas del análisis y diseño previos, incluyendo las tecnologías utilizadas y los flujos de datos necesarios, acompañadas de distintos diagramas y representaciones que puedan facilitar la comprensión, y de justificaciones que puedan justificar el uso de algunas opciones respecto a otras. Además, también se incluyen las pequeñas variaciones respecto a lo decidido en las fases de diseño.

7.1. Código de extracción e inserción de datos

El código de extracción e inserción de datos ha sido desarrollado con el lenguaje de programación Python, utilizando el IDE Pycharm. La principal razón ha sido la presencia de multitud de reconocidas librerías con paquetes para el análisis o la visualización de datos en Python, y el soporte para lo relacionado con el sector de data science en Pycharm, que si bien ninguna de estas características han sido utilizadas hasta ahora, pueden resultar beneficiosas para el desarrollo del proyecto a largo plazo.

Esta parte del proyecto consiste en 2 funciones principales que siguen una estructura switch-case y otras 2 para llamar a estas 2 primeras y hacer que se ejecuten de forma periódica.

La primera de las funciones es download_data(option). Consta de un switch-case que dependiendo de la opción que haya recibido la función, descarga unos datos u otros de la API de Twitch y los almacena en un archivo .JSON y en una variable de tipo diccionario. Las opciones que puede recibir esta función son languages, categories y streams.

Hay que tener en cuenta que lo que la API de Twitch devuelve es, en la mayoría de casos, un array con las retransmisiones que hay en directo en el momento de la consulta a la API con datos sobre cada una de las transmisiones (título, número de espectadores, idioma, categoría, +18/-18, etc), por lo que download_data también ha de manipular el formato de las respuestas de la API para que sean más fáciles de tratar una vez se encuentren en la base de datos y se quieran analizar y representar gráficamente.

La segunda de las funciones es insert_data_to_database_table(data,table). Ésta recibe el diccionario devuelto por la función download_data y la tabla a la cual se quiere realizar la inserción como parámetros, y en función de estos datos hace la inserción a la base de datos, limpiando éstos previamente en los casos necesarios. La función process_data() se encarga simplemente de llamar a estas funciones para descargar los datos de Twitch y luego subirlos a la base de datos. La adjunto a continuación:

```
def process_data():
    categories_table = download_data("categories")
    languages_table = download_data("languages")
    streams_table = download_data("streams")


    insert_data_to_database_table("categories", categories_table)
    insert_data_to_database_table("languages", languages_table)
    insert_data_to_database_table("streams", streams_table)
```

Fig. 5: Función process_data

Por último, la función main() se encarga de que cada hora en punto la función process_data() sea ejecutada, para así actualizar de forma regular la base de datos.

De la forma que se ha estructurado el código se facilita lo máximo posible que cualquier persona que quiera modificar el código para descargar e insertar nuevos datos pueda hacerlo, ya que sólo sería necesario crear dos nuevas llamadas a funciones dentro del código de process_data, un case en download_data y otro case en insert_data_to_database_table, y no sería necesario entender detalles de la implementación de las descargas e inserciones de los otros datos.

7.2. Base de datos

Se ha utilizado XAMPP [5] para crear una base de datos MySQL [6] llamada twitch. Se ha decidido hacer uso de MySQL por ser una opción gratuita, lo cual es necesario para que este proyecto no requiera de cobrar a los usuarios, y por su capacidad de integración con múltiples lenguajes de programación y frameworks.

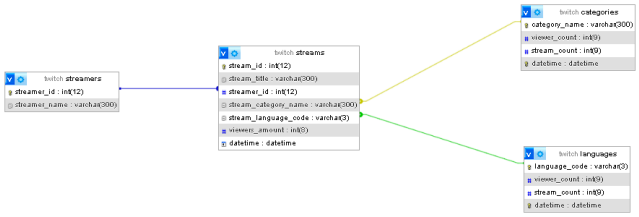

Fig. 6: Esquema de la base de datos

En la figura anterior se puede apreciar que es una base de datos sencilla cuya tabla más importante es la tabla streams, la cual hace posibles la mayoría de las consultas utilizadas para este TFG. En la implementación de la base de datos se ha realizado un cambio respecto al diseño que se había planificado, y es que categories y languages constan de los atributos viewer_count, stream_count y datetime repetidos. Esto sucedió porque a la hora de implementar las consultas a la base de datos, el datetime se convirtió en un punto muy relevante para la mayoría de consultas. Se implementaron inicialmente otras alternativas para gestionarlo, como el uso de tablas aisladas para los Viewers y para la gestión del tiempo, y finalmente se decidió esta alternativa para no complicar en exceso la implementación y minimizar el número de errores y problemas asociados a la base de datos.

7.3. Página web

En esta sección se explica la implementación de la página web en un orden que facilita su entendimiento pese a la ausencia de imágenes (ya que estas serán utilizadas en la sección de resultados).

7.3.1. Tecnologías

La página web se ha desarrollado utilizando HTML, CSS, JavaScript, la librería Chart.js [7] y el framework Bootstrap [8] para el frontend, y PHP para el backend. Por un lado, las razones para utilizar Bootstrap han sido la facilidad de hacer los elementos responsivos y poder utilizar estilos CSS predefinidos, además del hecho de que consta de una documentación completa que puede agilizar la resolución de errores. Por otro lado, la principal razón para escoger PHP fue su facilidad de integración con el resto del proyecto. De hecho, inicialmente se trató de realizar una implementación del backend con Flask [9], pero la configuración para permitir su uso en XAMPP generó demasiados errores y acabó siendo descartada.

7.3.2. Concepto base

La página web consta de una página principal donde hay un formulario y una preview de la respuesta que devuelve el servidor a los datos introducidos al formulario, otra página con estadísticas genéricas sobre Twitch y una última página que contiene la respuesta completa que ha devuelto al servidor, a la cual se accede desde una referencia en la preview. Lo más relevante para entender el funcionamiento de la página es el formulario, y qué sucede desde que el usuario lo rellena hasta que ve la respuesta completa, es decir, el flujo principal de datos. Y para entender este flujo, es conveniente tener una representación visual del sistema de directorios utilizada en la página web.

7.3.3. Sistema de directorios

El sistema de directorios sigue el patrón modelo-vista-controlador, tal y como se establece en la fase de diseño y como se refleja en la siguiente figura:

```
Web/
├──index.php
├──generic_charts.php
├──chart.html
├──Controladores
│  ├──calcula_estadisticas.php
│  ├──load_selector.php
│  ├──genera_graficos_gen.php
├──Modelos
│  ├── db_connect.php
│  └── form_answer_sql_queries.php
│  └── get_selectors_data.php
│  └── generic_graphs.php
├──Vistas
│  ├──selector_view.php
│  ├──generic_graphs_view.php
├── js
│  ├──charts.js
│  ├──data.js
│  ├──form_validation.js
│  ├──generic_graphs.js
```

Fig. 7: Sistema de directorios de la web

No aparecen mis propios archivos de css en el sistema de directorios debido a que el estilo se aplica haciendo uso de las clases de Bootstrap incluidas en los tags del código HTML.

7.3.4. Flujo principal de datos

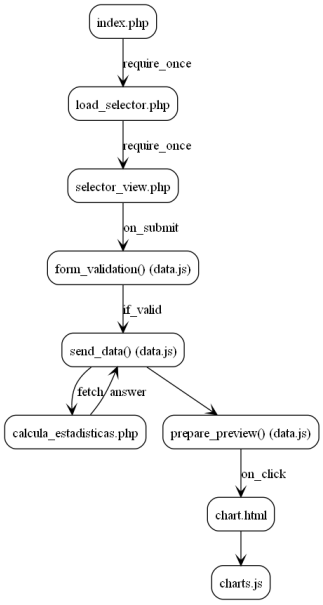

Fig. 8: Flujo de datos principal de la página web

La figura anterior describe de forma resumida el flujo de datos principal de la página web. En el resto del apartado, se describe de manera detallada.

El flujo principal de datos, el cual responde de forma dinámica a las peticiones del usuario, consiste en lo siguiente:

Desde index.php se incluye el controlador load_selector.php, que a su vez incluye el modelo db_connect.php para conectarse a la base de datos y el modelo get_selectors_data, que obtiene de la base de datos los datos para rellenar los selectores de la página inicial. Finalmente, load_selector.php incluye la vista selector_view.php que muestra el formulario con los datos de los selectores cargados.

Este formulario tiene un evento onsubmit=send_data() que, al pulsar el botón Calculate Statistics que aparece en la imagen 10, envía las respuestas de los usuarios al fichero data.js, a la funcion send_data(). En esta función se hace un fetch al controlador calcula_estadisticas.php, que consta de una estructura switch-case donde se decidirá cómo actuar en función de lo que ha enviado (y lo que no ha enviado) el usuario a través del formulario.

Calcula_estadisticas.php obtiene los datos necesarios para la consulta que ha realizado el usuario y éstos son devueltos a la función send_data() de data.js a través del fetch. Una vez send_data() recibe los datos, los almacena en el session storage de la web.

Por último, send_data() llama a otra función presente en data.js llamada prepare_preview(), la cual utiliza los datos del session storage para generar la representación gráfica de los datos en el apartado preview de la página principal de la web.

Si el usuario decide pulsar en la referencia que hay en el apartado preview, ésta le lleva a la página ”Your analysis”, es decir, al fichero chart.html, el cual tiene asociado el archivo charts.js. Este archivo ejecuta una función render_graphics() en su evento on_load, la cual genera de forma dinámica los elementos y gráficos que ha de representar en

función de los datos almacenados en el session storage, y de esta forma el usuario obtiene el análisis de datos completo para los parámetros que ha especificado en el selector de la página principal.

Es relevante mencionar la función form_validation(), que se asegura de que al menos uno de los selectores haya sido rellenado para poder pulsar el botón que envía el formulario.

El resto de archivos que todavía no han sido mencionados son generic_charts.php, genera_graficos_gen.php, generic_graphs.php y generic_graphs_view.php. Todos están relacionados con la página que contiene los gráficos genéricos, y son utilizados para generar los gráficos de esa página en el momento en que el usuario accede a ella.

8 TESTEO

Se han utilizado distintos métodos para testear el correcto funcionamiento de cada parte del proyecto.

Para la descarga de datos de la API de Twitch, ha sido utilizada la propia documentación que provee la API junto a los códigos de error que ésta devolvía para ir perfeccionando el formato de las peticiones hasta evitar completamente los errores. Además, para asegurar que devolvía datos correctos y evitar problemas al querer introducirlos a la base de datos, se han utilizado los archivos JSON que genera la función de extracción de datos para comprobar visualmente la presencia de campos vacíos o erróneos, para así ir perfeccionando el control de errores.

Para la base de datos, se han introducido casos límite desde el código de inserción de datos para ver qué ocurría, y así poder corregir las clausulas de la base de datos y las comprobaciones previas a las inserciones de datos. De esta forma, por ejemplo, se reconoció que no se estaban formateando las fechas de manera correcta en un inicio.

En cuanto a la web, el frontend se ha debuggeado a través de la inclusión de trazas en lugares específicos del código para controlar los valores que iban tomando ciertas variables, y el backend se ha debuggeado haciendo uso del plugin Xdebug helper [10] de Google Chrome y la extensión PHP Debug para Visual Studio Code.

9 RESULTADOS

El resultado de este proyecto ha sido un sistema de soporte a la toma de decisiones con tres partes que se relacionan entre sí para extraer datos en bruto de Twitch, procesarlos y exponer informaciones útiles a los usuarios de la plataforma que quieran hacer uso de ellas.

En cuanto a la extracción e inserción de datos, tenemos un código en Python fácilmente ampliable que se puede mantener en ejecución de forma indefinida extrayendo e insertando datos.

En cuanto a la base de datos, tenemos una base de datos ya puesta en funcionamiento que se adapta a las necesidades del proyecto y que consta de distintas clausulas que garantizan la integridad de los datos.

En cuanto a la página web, se ha desarrollado una web con tres páginas, con la funcionalidad principal repartida entre dos de ellas.

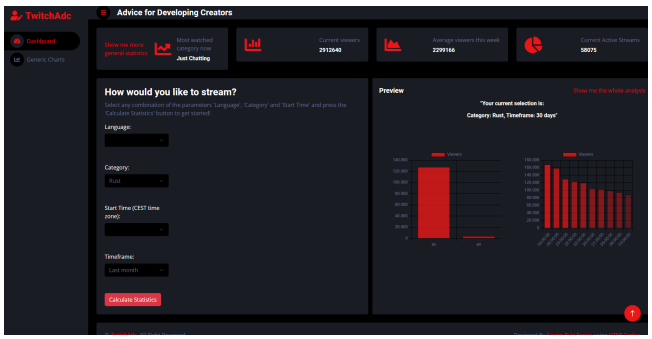

Fig. 9: Pagina inicial de la web

En la figura 9 se puede ver la página principal de la web.

Arriba a la izquierda se ve TwitchAdc, que es el nombre de la página. En la parte de arriba aparecen algunos datos genéricos que pudieran interesar a usuarios de la página, y en el lado izquierdo se puede ver en qué página estamos (Dashboard, es decir, la página inicial) y la otra página a la cual podemos navegar (Generic Charts).

No obstante, lo más relevante de esta primera página es el formulario del lado izquierdo, ya que será la principal funcionalidad de la página web. Este formulario consta de diversos selectores en los cuales el usuario puede escoger entre diversas opciones ya cargadas desde la base de datos. El usuario puede decidir qué opciones contestar y qué opciones dejar en blanco, y en función de lo que escoja, se harán diferentes consultas a la base de datos de forma dinámica y se crearán distintos gráficos útiles para el usuario que se utilizarán para hacerle recomendaciones concretas sobre qué stremear, entre otras cosas.

A la derecha del formulario se ve un apartado de “Preview”, que es donde se cargarán los gráficos obtenidos como respuesta a los criterios introducidos por el usuario en el formulario. En caso de pulsar en “Show me the whole analysis”, el usuario será redirigido a la página “Your analysis”.

Fig. 10: Página de análisis completo

En la figura 10 se muestra la página ”Your analysis”, la cual permanece oculta hasta que se accede por primera vez de la manera ya descrita. En esta página, que completa la funcionalidad principal de la web, el usuario verá los diferentes gráficos que aparecían en la preview junto a textos explicativos sobre cada uno de los gráficos generados, además de una cabecera que le recuerda los valores que había seleccionado en el formulario.

Fig. 11: Página de estadísticas generales

Por último, en la figura 11 aparece la página de estadísticas generales, que cumple la función de presentar algunos datos útiles y fáciles de entender a aquellos usuarios que no tengan claro ni el idioma en el que quieren crear contenido, ni la categoría, ni la hora.

Todo lo descrito en este apartado hasta ahora se realiza de forma automatizada y parametrizada, y dado que se han utilizado únicamente datos gratuitos de la API de Twitch y ni siquiera se hace uso de un servidor por ahora (todas las partes del proyecto están ubicadas en mi ordenador), no ha supuesto ningún tipo de gasto económico.

Se podría discutir que dado que mi ordenador actúa como servidor, sí que hay un gasto, y que por lo tanto este proyecto no tendría costes únicamente en el caso de que regalara mis recursos. No obstante, debido a la minimización de costes que se logra hacer a través de la automatización, el uso exclusivo de datos accesibles de forma gratuita y el uso de software también gratuito, no sería difícil cubrir los costes que se pudieran tener a nivel de recursos con métodos como el uso de publicidad en la web, evitando el cobro directo a los usuarios.

Aclarado el estado final del proyecto, y para concluir con los resultados, habría que hacer un repaso de los objetivos y comprobar el grado de cumplimiento de cada uno de ellos.

En cuanto a los objetivos de prioridad máxima, asociados con que la plataforma se diferencia de opciones similares ya existentes, se han cumplido todos en su totalidad. Se han conseguido informaciones útiles de forma gratuita a través de la API de Twitch, y se ha conseguido automatizar absolutamente cada parte del proceso por el que pasan los datos, desde la extracción a la representación gráfica, pasando por la inserción y el análisis. Esto implica que el proyecto puede funcionar, ser útil y mantenerse en el tiempo sin la necesidad de cobrar a los usuarios ni de la intervención de ninguna persona. Como consecuencia, la plataforma logra diferenciarse de otras opciones ya establecidas como Twitchtracker o Stream Charts y ser más equitativa y justa para con los creadores de contenido de Twitch.

Refiriéndonos a los objetivos de prioridad alta, había 2 tipos de objetivos. Por un lado, aquellos que hacían referencia al hecho de que las distintas partes del proyecto funcionaran bien como conjunto (ya que, al fin y al cabo, de nada servía que funcionaran las distintas partes de manera independiente si no podían integrarse). Éstos, dado que el proyecto funciona, han sido completados.

Por otro lado, había dos casos concretos de objetivos de prioridad alta que quiero mencionar. El primero de ellos era que el tiempo de carga de cualquier parte de la página web fuera inferior a 2 segundos. Este objetivo se estableció para que la página web pudiera servirle a los usuarios, dado que si el tiempo de carga hubiera sido excesivo, los usuarios no harían uso de la página web por falta de comodidad. Este objetivo se ha cumplido también, pero ha obligado a que algunas consultas complejas no se pudieran realizar, o como mínimo representar gráficamente, por provocar el incumplimiento de este objetivo. Así que, pese a haber sido cumplido, ha hecho necesario sacrificar algo de complejidad en los análisis. El segundo era conseguir representar gráficamente todos y cada uno de los datos obtenidos de Twitch, y esto no se ha completado en su totalidad, ya que se han desarrollado funciones de extracción de datos que no han acabado ni en la base de datos ni en la página web debido a que generaban demasiada complejidad, la cual me resultaba imposible de gestionar con la cantidad de tiempo disponible para el proyecto. Estos dos objetivos se traducen en futuros pasos para el proyecto que serán mencionados en su correspondiente apartado.

Los objetivos de prioridad media, referentes a temas de seguridad e integridad de la base de datos y de la página web, han sido completados. La base de datos consta, como se ha mencionado en este artículo, de clausulas restrict y cascade que evitan fallos en la integridad y errores derivados de estos fallos. La página web, por su lado, está protegida contra los ataques principales a páginas web como SQL Injection, para lo cual se utiliza la función bind_param() en todas las consultas SQL que se realizan a la base de datos, o cross-site scripting, para lo cual se utilizan funciones como htmlspecialchars() para escapar caracteres especiales y evitar que se interpreten como código HTML.

Por último, los objetivos de prioridad baja se han cumplido parcialmente. Por un lado, en cuanto a la traducción de la página, pese a que la página sirve para el idioma español en cuanto a que puedes seleccionar el idioma español en el selector de idioma y obtener datos al respecto en las consultas, no hay una versión desarrollada por mí en la que el texto de la página esté en español, eso sólo se puede obtener a través de la traducción que proporcionan navegadores como Google Chrome. Por otro lado, pese a que sí se ha obtenido feedback de creadores de contenido de la plataforma a través de conversaciones con ellos, no se les ha realizado encuestas ni se ha recogido el feedback recibido de forma númerica o quantitativa ya que no se podía obtener un número relevante de respuestas y, además, las que se podían obtener eran de un grupo muy concreto de creadores de contenido, así que la información recogida habría estado sesgada.

10 Conclusiones

Pese a la presencia de casos de acceso limitado o restringido a parte de los datos de Twitch (como la edad o género de los espectadores de una categoría), la realización de este proyecto me ha demostrado que, para cualquier persona con un mínimo de conocimientos de informática (incluso para un estudiante), es definitivamente posible extraer datos relevantes y útiles de la API de Twitch y convertirlos en informaciones útiles a disposición de los creadores de contenido de la plataforma, favoreciendo un modelo de toma de decisiones basado en los datos (data-driven). Tampoco es difícil garantizar ciertos principios mínimos de seguridad al proyecto siempre y cuando se conozcan estos principios.

Automatizar todo este proceso haciendo uso de una base de datos y códigos parametrizados y generalizados que adaptan su forma de actuar en función de los datos que se están manejando también es viable, pese a ser considerablemente más difícil que realizar algunas tareas de forma manual. No obstante, me he dado cuenta de que el conocimiento sobre visualización de datos puede ser limitante a la hora de automatizar una plataforma como la desarrollada en este proyecto, pese a no ser un factor que necesariamente impida un resultado satisfactorio.

Al fin y al cabo, el proyecto ha tenido éxito. TwitchAdc es una plataforma que ofrece informaciones útiles haciendo uso de datos de la API de Twitch sin necesidad de cobrar a los usuarios ni de realizar ninguna parte del proceso de forma manual, permitiendo además una ampliación del proyecto relativamente fácil. Y ante todo, es una plataforma que se diferencia de la competencia, haciendo que su creación no pierda el significado. Sin embargo, lo cierto es que la plataforma podría haber sido mucho mejor si no hubiera sido por problemas relacionados con la visualización.

La cantidad de problemas derivados de automatizar que datos variables que se reciben de forma dinámica se representen en una página web ha sido mucho más alta de lo esperado, ya que junta muchos factores a la vez, como el layout de la página, el número de ejes a representar, el cómo interactuan los elementos gráficos en una web, etc. Esto ha generado limitaciones en factores como el tipo de datos que he podido representar o el número máximo de gráficos que he podido devolver como respuesta a una consulta (que me habría gustado que fuera superior en algunos casos). Y lo que no lo ha limitado funcionalmente, lo ha limitado a nivel de tiempo, dado que ha habido detalles que no han podido ser implementados por falta de tiempo, precisamente por el tiempo invertido en conseguir el correcto funcionamiento de las representaciones gráficas.

Por lo tanto, pese a que considero que el proyecto ha sido un éxito y ha cumplido con la mayoría de los objetivos propuestos, y en especial con los más esenciales, lo cierto es que todavía quedan cosas por hacer.

11 Futuros pasos

TwitchAdc es un proyecto que va a seguir en desarrollo en un futuro, y ya ha sido planeada la inclusión de distintas funcionalidades. Por un lado, sería importante incluir todos los idiomas que provee la API de Twitch en el selector de idiomas, en vez de sólo el idioma español e inglés. Además, se podrían añadir nuevos selectores con opciones como por ejemplo una diferenciación entre categorías exclusivas para mayores de edad y el resto.

Por otro lado, se tendría que conseguir representar todos los datos que se extraen de la API de Twitch, y que cada una de las funciones de extracción se vea representada en la página web. Para esto será necesario deshacerse de la limitación en el número de gráficos que se pueden representar de forma correcta como respuesta a una consulta.

Sería también una buena idea plantear una reestructuración de la base de datos para lograr que consultas más complejas se puedan ejecutar en un tiempo lo suficientemente bajo como para no ser un problema para el tiempo de carga de la web.

Otro punto a tener en cuenta en un futuro sería crear dependencias entre los selectores de la página web. Es decir, implementar los selectores de forma en que si, por ejemplo, un usuario escoge la categoría 'Just Chatting', y no hay datos para esa categoría a las 22h, la opción 22h deja de aparecer en el selector de hora. De esta forma, en caso de que se presentaran combinaciones de selecciones para los cuales no hubiera datos por alguna razón, no se podría llegar a generar ningún problema.

Por último, si la página tiene éxito y empiezan a llegar una cantidad muy elevada de peticiones en relación con TwitchAdc, será prácticamente obligatorio migrar el proyecto a un servidor.

Agradecimientos

Me gustaría expresar mi más sincera gratitud a cuatro personas en concreto. En primer lugar, a mi padre, por adentrarse en Twitch e inspirarme a empezar este proyecto, además de aconsejarme y permitirme contactar con otros creadores de la plataforma. En segundo lugar, a mi amigo David, que me convenció de que la idea era lo suficientemente amplia como para desarrollar un trabajo de final de grado al respecto. En tercer lugar, a mi amigo Dani, que todas y cada una de las veces que estaba horas y horas para no conseguir nada y llegaba a mi límite, sin falta, me decía que 'todo saldrá bien' y, aunque yo le respondía 'no estoy tan seguro', en el fondo me motivaba a seguir. Y por último, al profesor Victor Garcia Font, por guiarme a lo largo del desarrollo del proyecto y ser siempre paciente y comprensivo conmigo, resolviéndome una y otra vez las mismas dudas que no me entraban en la cabeza. Sin alguien como él, este proyecto no habría sido posible.

Referencias

[1] Twitch. Available at: https://www.twitch.tv/

[2] Twitch channels, games and Global Statistics · Twitchtracker. Available at: https://twitchtracker.com/

[3] Streams charts Pro Data Access · STREAMS CHARTS. Available at: https://streamscharts.com/pricing

[4] Twitch API. Available at: https://dev.twitch.tv/docs/api/

[5] XAMPP. Available at: https://www.apachefriends.org/es/index.html

[6] MySQL. Available at: https://www.mysql.com/

[7] Chart.js. Available at: https://www.chartjs.org/

[8] Bootstrap. Available at: https://getbootstrap.com/

[9] Flask. Available at: https://flask.palletsprojects.com/en/2.3.x/

[10] Xdebug helper. Available at: https://chrome.google.com/webstore/detail/xdebug-helper/

ANEXOS

A.1. Diagrama de Gantt

En este anexo se adjunta el diagrama de Gantt con la planificación temporal del proyecto que fue decidida previamente a su desarrollo.

El diagrama de Gantt se divide en imágenes para que se pueda ver de forma correcta.

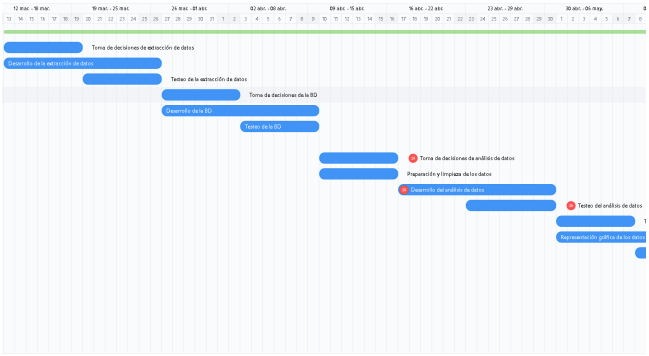

Fig. 12: Diagrama de Gantt: parte 1

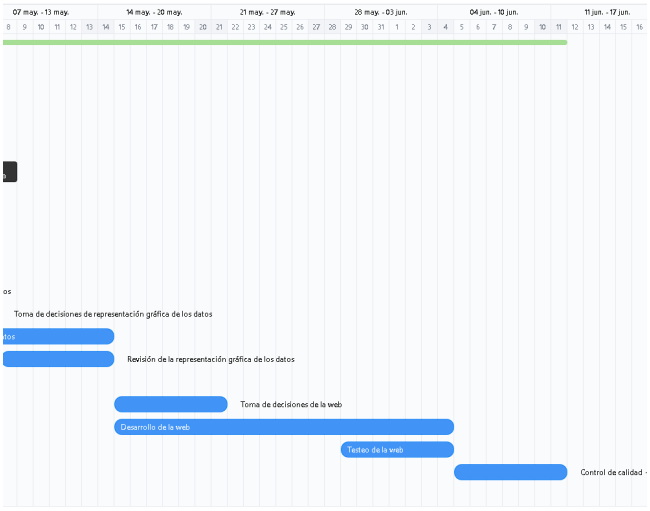

Fig. 13: Diagrama de Gantt: parte 2

A.2. Especificaciones de los casos de uso de la página web

En este anexo se adjuntan las especificaciones técnicas de los casos de uso presentados en la figura 4.

Propiedad	***Descripción***
Id. Caso de uso	*001*
Nombre o título	*ConsultarEstadísticas*
Precondiciones	*Haber accedido a la url de la página principal.*
Postcondiciones	*No hay postcondiciones relevantes.*
Descripción	*El usuario puede ver una previsualización que incluye algunas de las estadísticas generales en la parte alta de la página principal.*

Fig. 14: Especificación del caso de uso 1

Propiedad	***Descripción***
Id. Caso de uso	*002*
Nombre o título	*ConsultarEstadísticasCompletas*
Precondiciones	*Haber accedido a la url de la página principal.*
Postcondiciones	*Se carga la url de la página donde están recopiladas las estadísticas generales completas.*
Descripción	*El usuario puede ver una página donde están recopiladas las estadísticas generales.*
Escenario principal	*El usuario pulsa en el botón "Ver todas las estadísticas generales", arriba a la izquierda de la página principal.*
Escenario alternativo	*El usuario pulsa en el botón "Estadísticas generales" en el índice que aparece en el lado izquierdo de la página.*

Fig. 15: Especificación del caso de uso 2

Propiedad	***Descripción***
Id. Caso de uso	*003*
Nombre o título	*RellenarFormulario*
Precondiciones	*Haber accedido a la url de la página principal.*
Postcondiciones	*El botón "Calcular Estadísticas" perteneciente al formulario ha sido pulsado*
Descripción	*El usuario puede rellenar un formulario para recibir estadísticas, análisis y explicaciones que le puedan ayudar para su caso concreto.*
Escenario principal	1- *El usuario rellena los campos del formulario que cree convenientes.* 2- El usuario pulsa el botón "Calcular Estadísticas" perteneciente al formulario.

Fig. 16: Especificación del caso de uso 3

Propiedad	***Descripción***
Id. Caso de uso	*004*
Nombre o título	*PrevisualizarAnálisis*
Precondiciones	*Haber accedido a la url de la página principal.* Haber pulsado el botón "Calcular Estadísticas". Que la base de datos esté cargada.
Postcondiciones	*En la página principal, ahora también aparece la previsualización de la respuesta al formulario enviado por el usuario.*
Descripción	*El usuario puede ver una previsualización de parte de la respuesta que ha recibido por rellenar el formulario a la izquierda de éste.*

Fig. 17: Especificación del caso de uso 4

Propiedad	***Descripción***
Id. Caso de uso	*005*
Nombre o título	*VerAnálisisCompleto*
Precondiciones	*El usuario debe haber pulsado el botón "Calcular Estadísticas".* Debe haberse consultado la base de datos. Debe haberse cargado una previsualización del análisis en la página principal.
Postcondiciones	*El usuario ha sido redirigido a una nueva url en la cual puede ver una serie de gráficas, datos y explicaciones útiles en base a las opciones rellenadas en el formulario.*
Escenario principal	*El usuario pulsa el botón "Enséñame el análisis completo" que aparece arriba a la derecha del apartado de previsualización en la página principal.*

Fig. 18: Especificación del caso de uso 5