TREBALL FINAL DE GRAU

Desarrollo de una aplicación web de consulta y recopilación de preferencias

Guillem Poy Trujillo
Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

CURS 2021-22

TecnoCampus
Escola Superior Politècnica

Centre adscrit a la

upf. Universitat Pompeu Fabra *Barcelona*

Agradecimientos

Si bien me han acompañado muchas personas durante el desarrollo de este Trabajo de Fin de Grado, la mención de honor se la merece el Dr. Léonard Janer, por su incalculable dedicación y atención en cada paso de este proyecto.

¡Muchas gracias!

Abstract

Development of a web page to communicate and visualize the preferences of users regarding Wikidata entries about which they are surveyed. The information is stored in a database and the results can be consulted by the users themselves, who can apply different filters that take into account the properties of the entries.

Resum

Desenvolupament d'una pàgina web per comunicar i visualitzar les preferències dels usuaris referents a entrades de Wikidata sobre les que són enquestats. La informació es desa en una base de dades i els resultats es poden consultar pels mateixos usuaris, els quals poden aplicar diferents filtres que tinguin en compte les propietats de les entrades.

Resumen

Desarrollo de una página web para comunicar y visualizar las preferencias de los usuarios referentes a entradas de Wikidata sobre las que son encuestados. La información se guarda en una base de datos y los resultados se pueden consultar por los usuarios mismos, los cuales pueden aplicar distintos filtros que tengan en cuenta las propiedades de las entradas.

Índice

Índice de figuras

1. Introducción

La visualización del vídeo de Tom Scott “1,204,986 Votes Decided: What Is The Best Thing?”[1] ha motivado la idea de crear una página web dinámicamente actualizada y colaborativa, que permita consultar y aportar información referente a las preferencias de los usuarios sobre gran cantidad de elementos listados en la información pública de Wikidata.

Para poder lograr alcanzar ese objetivo se han estudiado las herramientas facilitadas por Wikidata con la intención de averiguar qué métodos existen para obtener datos y cuál es el más adecuado para el proyecto. También se han investigado métodos recomendados para la correcta creación de cuestionarios además de su adecuada valoración.

Utilizando una metodología de reuniones y objetivos bisemanales se ha desarrollado la aplicación web. Este desarrollo ha empezado con la confección de una aplicación básica usada como prueba para asegurar que las tecnologías se puedan usar entre sí.

Tras el desarrollo de la aplicación base se ha empezado el de la aplicación que potencialmente permitiese alcanzar los objetivos del proyecto, empezando por la visualización de entradas de Wikidata, seguido de la posibilidad de votar entre dos de ellas, el registro de los datos obtenidos, visualización de los resultados y modificaciones recurrentes para conseguir un resultado que se pueda desplegar a gran escala.

Se han encontrado inconvenientes durante el proceso como la imposibilidad de usar AWS Amplify junto Flutter web, la baja consistencia que tiene la información en Wikidata, limitaciones de las herramientas usadas para obtener datos…

Sin embargo, se ha conseguido realizar una aplicación completa y funcional que se ha podido desplegar además de ser usada por una cantidad limitada de usuarios. La aplicación ha demostrado que es posible usar los datos de Wikidata como base de información para la creación de software pero también la gran importancia que tiene generar un elevado tráfico en aplicaciones que dependen de datos generados por sus usuarios para obtener valor.

[1] El video puede ser encontrado a través de esta URL: https://www.youtube.com/watch?v=ALy6e7GbDRQ

2. Objetivos y Alcance

2.1 Desarrollo de una aplicación web de recogida y consulta de opinión de preferencia sobre elementos de Wikidata

Desarrollar una aplicación web que permita consultar y recoger datos referentes a las opiniones de los usuarios sobre sus preferencias en cuanto a elementos extraídos de Wikidata [1].

La intención es que, gracias al acceso abierto que Wikidata ofrece de sus datos, estos se puedan extraer para utilizarlos en la aplicación web desarrollada.

Esta aplicación tiene la finalidad de descubrir cuál de los elementos extraídos de Wikidata es el preferido por los usuarios además de permitir la visualización de los resultados obtenidos.

2.2 Objetivos secundarios

2.2.1 Desarrollo de un "back end" y/o base de datos que utilicen la infraestructura de AWS como host

Desarrollar la aplicación web para que utilice y aproveche los servicios provistos por Amazon Web Services (también conocidos como AWS) [2].

Estos servicios pueden proveer de host para la base de datos y el "back end" en caso de que sean necesarios, y gozan de gran popularidad y una amplia cantidad de herramientas.

2.2.2 Implementación de un diseño que facilite la recogida de nuevos datos

Implementar un diseño que, de algún modo, consiga que los usuarios aporten nuevos datos a parte de consultarlos para evitar que la aplicación web solo se utilice para consultar resultados y que haya una escasez en la recogida de datos.

Se desconoce qué cantidad de usuarios va a utilizar la aplicación web, pero se teme que estos no sean suficientes como para conseguir que los resultados sean mínimamente valiosos. Así pues, con la intención de minimizar este problema, se busca realizar un diseño que fomente la provisión de nuevos datos por parte de los usuarios.

3. Marco teórico y análisis de referentes

3.1 Referentes

El referente principal de este proyecto es el vídeo de Tom Scott "1,204,986 Votes Decided: What Is The Best Thing?" [3].

En el vídeo, Tom Scott se propone responder a una pregunta: "¿Cuál es la mejor cosa?". Para hacerlo, se descarga todas las entradas de Wikidata, elimina todas aquellas que no tengan relevancia para la mayoría de personas (descartando todas aquellas entradas que no estén traducidas a más de 50 idiomas), además de las entradas sobre grupos de personas o individuos, localizaciones, religiones, ficciones, zonas horarias, crímenes, banderas, himnos, mitología, … hasta reducir la lista a pocos más de 7.000 elementos. Con ellos genera una encuesta online en la que más de 1.2 millones de votaciones se produjeron para conocer cuál de todas las cosas en la lista era considerada "la mejor".

La página original donde se produjeron las votaciones se puede consultar en la "Way Back Machine" [4].

Han existido otros referentes como "Most Awesomest Thing Ever" [5] que compartía unos objetivos muy similares a los de este proyecto, pero que debido a la retirada de Flash ya no es funcional.

"Which Is The Best Thing" era otra web que ya solo se puede consultar en la "Way Back Machine" [6]. Esta página permitía también votar entre dos elementos para terminar decidiendo cuál era la mejor "cosa"[2]. Esta aplicación añadía la posibilidad de que los usuarios enviasen "cosas" que faltaban en la lista y que creían que debían estar incluidas.

Otra página que sirve también como referente es "Higher Lower Game" [7], la cual permite a los usuarios jugar a un juego en el que tienen que adivinar cuál de los dos elementos mostrados tuvo más búsquedas mensuales en Google durante 2017. Sigue una idea similar a la del proyecto y añade un factor de gamificación que genera *engagement*.

3.2 Wikidata

Según la propia Wikidata[1]:

[2] Se utiliza el término "cosa" para definir cualquier término, información, persona, objeto, suceso, …

"Wikidata es una base de conocimiento libre y abierta que puede ser leída y editada tanto por humanos como por máquinas.

Wikidata actúa como almacenamiento central para los datos estructurados de sus proyectos hermanos de Wikimedia, incluyendo Wikipedia, Wikivoyage, Wiktionary, Wikisource y otros.

Wikidata también proporciona soporte a muchos otros sitios y servicios más allá de los proyectos de Wikimedia."

3.2.1 Estructura de los datos

Wikidata almacena y proporciona datos estructurados [8]. Estos son datos que han sido organizados de una manera definida, a menudo con la intención de codificar el significado y preservar las relaciones entre los diferentes puntos de datos.

Wikidata se organiza principalmente a través de elementos y sus datos siguen la siguiente estructura [9]:

- Item (entrada, ítem o elemento): cualquier representación de conocimiento. Está atado a un identificador único que siempre empieza por la letra "Q" y es seguida de un número.
    - Label (etiqueta): identificador de un elemento entendible por humanos. Se pueden repetir, pues las descripciones lo desambiguan si es necesario.
    - Description (descripción): Especificación corta utilizada para desambiguar las etiquetas proporcionando más detalles sobre un elemento.
    - Alias (alias): Un nombre alternativo para un elemento. Aunque un artículo sólo puede tener una etiqueta y una descripción por idioma, puede tener varios alias.
    - Statements (declaraciones): Cada elemento puede tener varios. Permiten la definición de distintos conceptos (elementos, palabras, números, imágenes, …)
        - Property (propiedad): Están definidas por un identificador único formado por una "P" seguida de números. Cada propiedad acepta solo unos ciertos tipos de valores, aunque puede tener varios valores a la vez.

- Value (valor): Determinan una propiedad. Pueden referenciar elementos, cantidades, desconocimiento o "sin valor"[10].
- Reference (referencia): Describe el origen de la información descrita en una declaración para respaldar tal afirmación. Una referencia está compuesta por una propiedad (definiendo el tipo de referencia) y un valor.

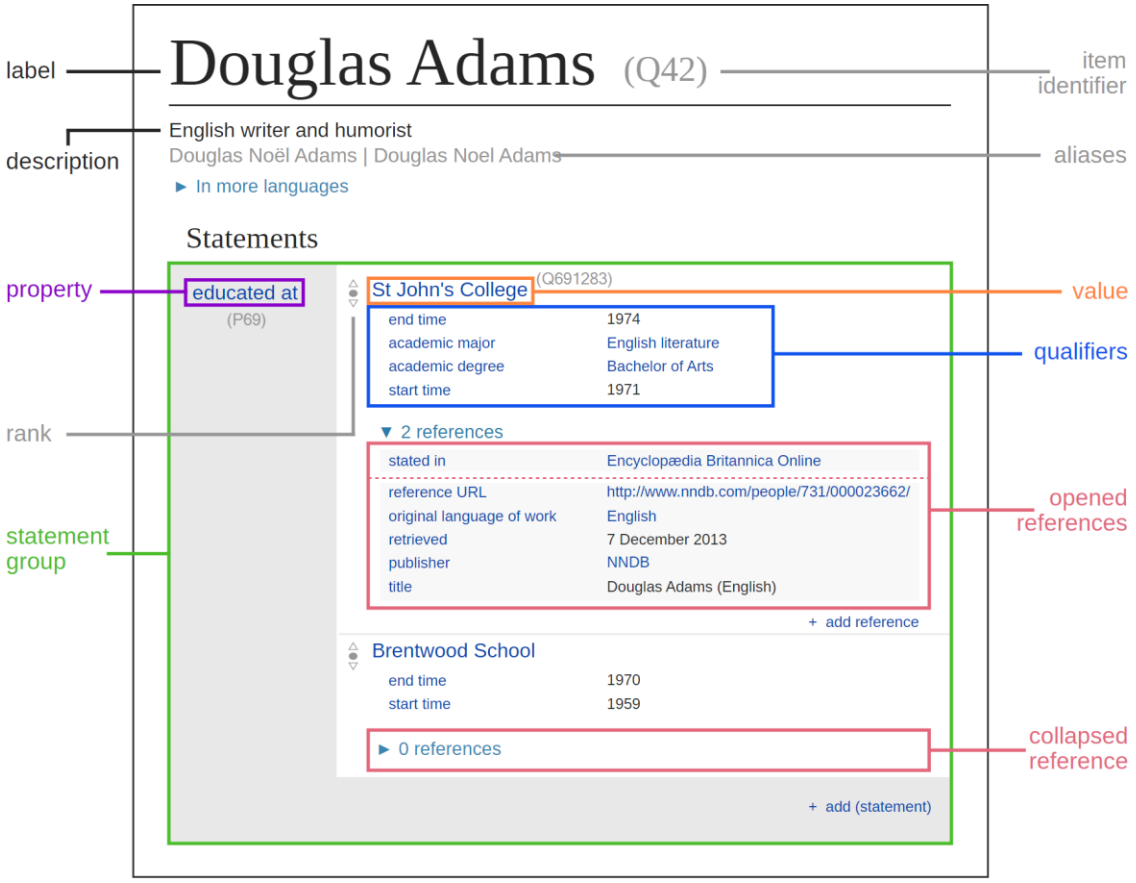

Figura 1: Una imagen con etiquetas mostrando la organización de la información en Wikidata con un ejemplo [11].

3.2.2 Obtención de los datos

Wikidata tiene extensa información y algunas guías que ayudan a entender cómo se pueden obtener y usar los datos [12]:

El contenido de Wikidata está disponible bajo una licencia libre, se exporta utilizando formatos estándar, y puede interconectarse con otros conjuntos de datos abiertos en la web de datos enlazados.

La obtención de los datos de Wikidata se puede realizar de distintas maneras:

- Descargando todos los datos de forma completa: Permite el acceso relativamente rápido, en una máquina local, pero tiene como mayores inconvenientes el hecho de que todos esos datos no son automáticamente actualizados y que ocupan mucho espacio. Sin embargo, esto permite un análisis y filtraje más ágil y controlado [13]. Los contenidos se extraen en formato JSON y contienen todas las entradas de Wikidata sin un orden específico [14].
- Suscribiéndose al flujo de cambios recientes, recibiendo notificaciones de actualizaciones cada vez que se produzcan ediciones/actualizaciones en páginas. Permitiría sólo tener acceso a información que no es interesante para el desarrollo de este trabajo [15].
- Obteniendo datos de elementos de manera individual
    - Linked Data interface: Permite el acceso a un solo elemento a través de una URI única y persistente obtenida a través de su ID que se puede usar como URL [16]. En el encabezado de la petición o modificando la URL se puede especificar el formato en el que se quiere el resultado (HTML, JSON, RDF, …), revisión, … Permite también obtener elementos aleatorios si se establece "Random" en vez de un ID en la petición, aunque no existe un modo de filtrar las entradas devueltas.

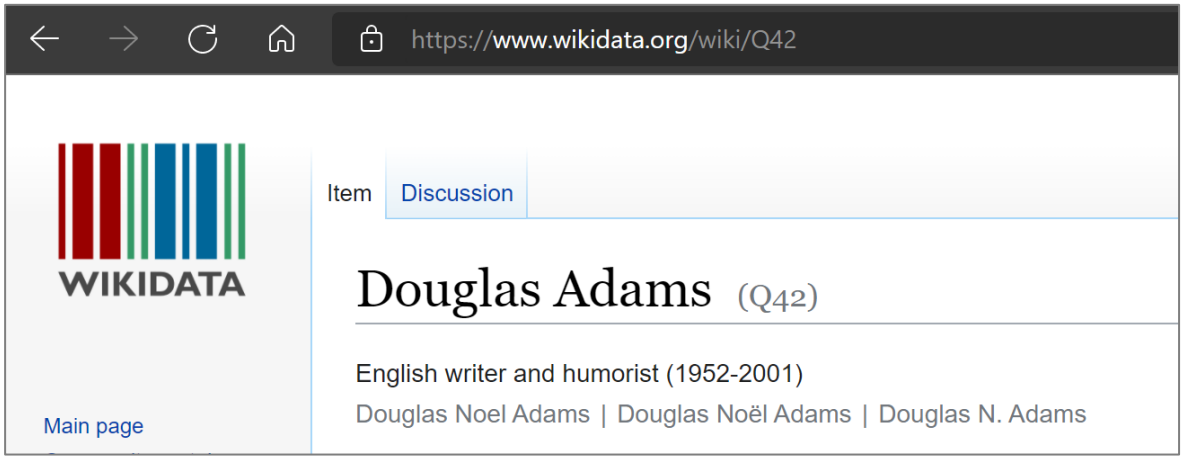

Figura 2: Ejemplo de URI (elemento "Douglas Adams", Q42)

- MediaWiki API: Es muy similar al "Linked Data Interface" pero no realiza las peticiones directamente a través de URI/URL, si no usando una API con acciones y parámetros predefinidos. Entre sus capacidades existe la edición

y obtención de datos, … [17]. Sin embargo, permite un nivel de filtro bastante bajo para peticiones de datos aleatorios, limitándolo a los namespaces[3] genéricos de Wikipedia (los cuales no distinguen entre elementos de ninguna manera, solo entre tipos de datos).

- SPARQL endpoints: Permite al usuario realizar consultas y ediciones de los datos con opciones de filtraje muy amplias. Esto se hace utilizando el "endpoint" SPARQL de Wikidata: "Wikidata Query Service". Este se puede usar como una web interactiva o enviando peticiones GET/POST programáticamente [18]. El lenguaje SPARQL permite interactuar con datos almacenados en el formato RDF [19], lo cual resulta ideal para Wikidata.
- Bots: Son herramientas pensadas para realizar modificaciones automáticas de los datos sin necesidad de intervención humana. No es interesante para este proyecto pues el concepto de Bot de Wikidata no gira alrededor de la obtención de información, si no de su alteración [20].

Todos los métodos para trabajar con los datos de Wikidata que permiten exportarlos lo hacen utilizando el formato **RDF** (entre otros).

El formato RDF, aparte de posibilitar el hecho de trabajar con los datos programáticamente, se centra en mostrar relaciones entre entidades y atributos ("sujeto" como identificador de la entidad, "predicado" como nombre del atributo y "objeto" como valor del atributo).

A diferencia de otros lenguajes, RDF no tiene el concepto de "null", pues solo recoge aquellas relaciones (predicados) que existen entre un sujeto y un objeto [19] y por ende, permite que se repitan predicados para un mismo sujeto.

Además, si se siguen los estándares FOAF [21] para nombrar los atributos, debería ser posible unir información procedente de distintos orígenes sin gran dificultad.

[3] Las páginas de una wiki de MediaWiki se agrupan en colecciones llamadas "namespaces" que diferencian el propósito de las páginas a un alto nivel. Algunos ejemplos (en inglés) son: Media, User, File, Help, …

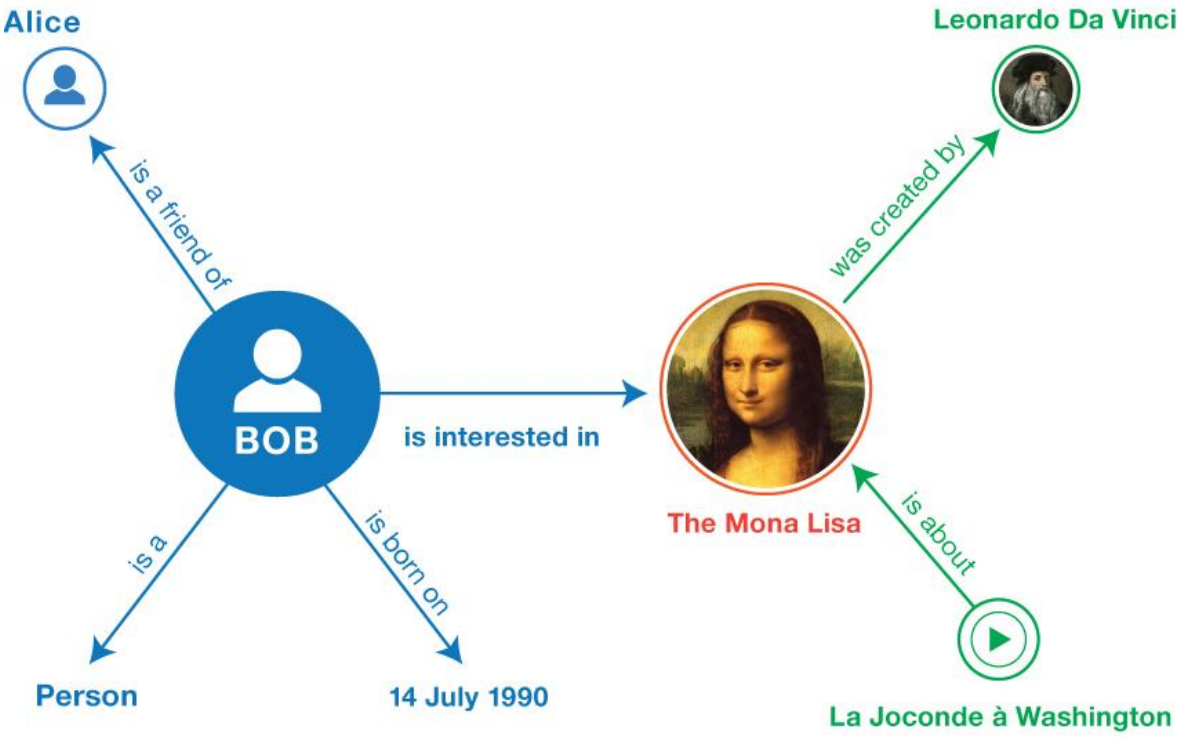

Figura 3: Esquema informal de un ejemplo de relaciones entre entidades y atributos siguiendo el estilo RDF. [22]

3.3 Estadística

Para poder alcanzar los objetivos del proyecto es esencial tener un buen entendimiento sobre las mejores prácticas relacionadas con el diseño de cuestionarios y sus análisis estadísticos.

3.3.1 Diseño de cuestionarios

Un cuestionario es el elemento base para conocer la opinión de los usuarios, pero ¿cómo se diseña un buen cuestionario?

Según P. Samuels [23], un cuestionario debe centrarse en opiniones o hechos y ser entregado a distintas personas en el mismo formulario.

Además, para maximizar el número de respuestas, hay que intentar relacionarse con el encuestado como persona, no como un usuario. Además, el formulario debe tener un título corto pero significativo y se tiene que hacer conocer su propósito.

Existen técnicas que se pueden utilizar para hacer que el cuestionario sea atractivo como prestar atención al apartado visual, mantenerlo tan corto como sea posible, conseguir que

para el encuestado sea lo más sencillo posible completarlo, ser consistente, ser claro y conciso…

En cuanto a cómo hay que realizar las preguntas de un cuestionario, se recomiendan los siguientes principios de diseño:

- La redacción de las preguntas debe ser clara y sin ambigüedades: hay que evitar especialmente dobles negaciones.
- Las preguntas no deben ser demasiado difíciles de responder.
- Las respuestas deben agotar todas las opciones posibles o tener una opción residual como "Otros".
- Las respuestas deben ser mutuamente excluyentes si sólo se puede elegir una opción.

También hay que intentar evitar integrar diseños con preguntas que puedan conducir a:

- Selección de la respuesta incorrecta involuntariamente.
- Preguntas engañosas.
- Escala sesgada.
- Las preguntas en las que su respuesta pueda perjudicar el prestigio puede que no se respondan con la verdad.
- Preguntas demasiado complicadas.
- Petición de clasificación de demasiados ítems.

Se recalca también que solo se debe preguntar aquello que se necesita y que para poder comparar grupos es útil hacer preguntas demográficas como la edad, género, nivel de educación, región, …

En cuanto a las preguntas relacionadas con clasificar elementos se menciona que de normal se dan entre 5 y 7 opciones (ejemplo: muy de acuerdo, de acuerdo, neutral, en desacuerdo, muy en desacuerdo). Menos opciones pueden reducir la calidad de los resultados y más pueden ser difíciles de procesar. En el mismo documento se argumenta que proporcionar un número par de respuestas fuerza al usuario a realizar una elección. Sin embargo, la mayoría de usuarios terminan realizando una elección positiva cuando habrían elegido al intermedia.

Para terminar, hay que tener en cuenta que el orden de las preguntas es importante, empezando por preguntas sencillas, seguidas de las más importantes y las más complejas y abiertas al final mientras que se recomienda también evitar preguntas muy similares a no ser que se estén construyendo escalas.

3.3.2 Tipo de encuesta

Un ensayo de Bernoulli es un método que permite clasificar un elemento entre dos resultados: éxito y fracaso [24]. Para que éste se realice correctamente, la probabilidad de éxito en cada iteración debe ser siempre la misma y para conseguir eso se puede utilizar una selección aleatoria de opciones.

Con este procedimiento se puede conocer la preferencia de los encuestados sobre un elemento en comparación a otros de un mismo grupo.

3.3.3 Análisis de resultados

Para analizar los resultados de manera correcta se puede seguir el algoritmo de Evan Miller [25], que expone como realizar una correcta ordenación de los resultados obtenidos en un estudio estadístico de valoración.

Este procedimiento tiene como objetivo reducir los resultados positivos o negativos que sufran de limitada cantidad de valoraciones.

Tal y como expone Miller, uno de los métodos más usados para realizar una ordenación según la valoración acostumbra a ser la diferencia entre las opiniones positivas y las negativas:

> *"Supongamos que un artículo tiene 600 valoraciones positivas y 400 negativas: eso da un resultado 60% positivo. Supongamos que el artículo dos tiene 5.500 valoraciones positivas y 4.500 negativas: 55% positivo. Este algoritmo sitúa el ítem dos (puntuación = 1.000, pero sólo 55% positivo) por encima del ítem uno (puntuación = 200, y 60% positivo)."*

Otros métodos utilizados en páginas como Amazon calculan la valoración de los productos usando la media de las valoraciones y eso puede dar lugar también a errores. Miller manifiesta:

> *"La valoración media funciona bien si siempre se tiene un montón de valoraciones, pero suponga que el artículo 1 tiene 2 valoraciones positivas y 0 negativas. Suponga que el artículo 2 tiene 100 valoraciones positivas y 1 negativa. Este algoritmo sitúa el artículo 2 (con muchas*

valoraciones positivas) por debajo del artículo 1 (con muy pocas valoraciones positivas).”

Miller considera que estos métodos no son los más óptimos y que lo que se debería hacer es usar como puntuación el límite inferior del intervalo de confianza de la puntuación de Wilson para un parámetro Bernoulli:

“Tenemos que equilibrar la proporción de valoraciones positivas con la incertidumbre de un pequeño número de observaciones. Afortunadamente, las matemáticas necesarias para esto fueron elaboradas en 1927 por Edwin B. Wilson”

3.4 Infraestructura

La infraestructura más común para desarrollar una aplicación web consta de un “host” que permite el acceso a los servicios a través de internet. En este se acostumbra a instalar un servidor web en el que se despliega la aplicación que usan los usuarios (“frontend”).

Además, una base de datos para almacenar y consultar información es normalmente necesaria especialmente si se quiere añadir persistencia a la web app.

Para que el “front end” se pueda comunicar con la base de datos, normalmente, se utiliza un “back end”, el cual permite exactamente esto, aumenta la seguridad y permite una mejor gestión de las conexiones a la base de datos.

3.4.1 Host

Se desconocen los requisitos exactos de la máquina que serán necesarios para el correcto funcionamiento de la aplicación, así que se tendrá que tener en cuenta qué opciones de host permiten ampliar los recursos de la máquina bajo demanda. El único requisito que es conocido desde el inicio del desarrollo del proyecto es que la máquina debe ser accesible en todo momento.

Uno de los factores a tener en cuenta es que puede existir una máquina de desarrollo y una de producción, por lo que el host puede cambiar a lo largo del proyecto.

Existen distintas opciones para obtener y usar un host. Algunas de las más populares son las siguientes:

- Personal en el hogar: Permite un control prácticamente absoluto de la máquina y la eliminación del coste de alquiler de ésta, aunque aumentaría los costes indirectos. Sin embargo, requiere de una máquina propia, una conexión a internet y electricidad estable. Hay que tener en cuenta que es un sistema muy viable, sencillo y rápido para realizar pruebas, especialmente en las primeras fases del desarrollo.
- Amazon Web Service [2]: Se trata de uno de los hosts más populares hoy en día y ofrece una gran cantidad de servicios, documentación y estabilidad de la infraestructura. Algunos de sus puntos débiles más importantes son los costes y su interfaz poco intuitiva. Sin embargo, los costes son escalables en función de las necesidades y existen también opciones gratuitas para ciertos servicios.
- Heroku (free) [26]: Facilita un hosting gratuito que proporciona hasta 1000 horas de servicio al mes de sus contenedores llamados “dynos”. Esto permite mantener un dyno conectado permanentemente. Sin embargo, estos se apagan si no reciben tráfico en 30 minutos y, al encenderse, el usuario puede sufrir un tiempo de espera de más de 1 minuto.
- Toolforge [27]: Es un servicio de hosting enfocado a prestar servicio a desarrolladores que trabajen en servicios que proporcionen valor a Wikimedia. Sin embargo, las tecnologías que soporta son más limitadas que en la mayoría de los competidores.
- Back4App [28]: Servicio de back end centrado en la creación de aplicaciones modernas rápidamente con poco código y sin la necesidad de tener que gestionar infraestructura. Dispone de SDKs dedicados a muchas plataformas, entre ellas Flutter además de facilitar la escalabilidad de características con consultas a tiempo real, funciones en la nube, …

Los distintos elementos que habrá que integrar en el host elegido son:

- Web: La parte “visible” de la aplicación. Se debe instalar y desplegar un servidor web en el host para poder acceder a la página a través de cualquier navegador. En este servidor se debe poder proporcionar un “front end” que permita que la app integre todas las funcionalidades deseadas y que funcione correctamente en cualquier dispositivo convencional (smartphone, ordenador, tablet, …).
    - Para el servidor existen muchas opciones siendo Apache [29] y NGINX [30] dos de las más populares.

    - En cuanto al "front end", se puede elegir una variedad muy grande de tecnologías bastante distintas. Desde desarrollarlo con HTML "puro" a usar frameworks como Flutter[4] [31] o Ionic [32].

- Back end: Aunque no es completamente necesario y se puede desarrollar una aplicación web sin este, un "back end" que proporcione escalabilidad, limite y controle las conexiones a la base de datos puede ser deseable en ciertas situaciones, especialmente si la aplicación tiene un número elevado de usuarios.
    - Existen muchas tecnologías que se pueden usar para desarrollar un "back end". Una de las más populares es Django [33], aunque si se quiere limitar la cantidad de dificultades del proyecto, desarrollarlo en Asp.net [34] lo haría seguramente más rápido, pues el lenguaje que utiliza (C#) es conocido.

- Base de datos: Con la finalidad de poder almacenar la información recogida sobre las opiniones de los usuarios, es completamente necesario desarrollar y desplegar una base de datos. Estas pueden ser de distintos tipos según el tipo de datos [35] que se quieran almacenar en ellas:
    - Relacional: se basan en la organización de la información en trozos pequeños, que se relacionan entre ellos mediante la relación de identificadores. Algunos ejemplos de bases de datos relacionales son MySQL, MariaDB, PostgreSQL, Oracle, … [36]
    - No Relacional: los datos no tienen un identificador que sirva de relación entre unos y otros. La información se organiza normalmente mediante documentos y es útil cuando no existe un esquema exacto de lo que se va a almacenar. Algunos ejemplos de bases de datos no relacionales son MongoDB, Cloud Firestore, Cassandra, Amazon DynamoDB, … [37]

  Sin embargo, debido a la simpleza de los datos que se quieren almacenar con la aplicación a desarrollar en este proyecto, prácticamente cualquier tipo de base de datos sería adecuada.

[4] Flutter es una tecnología ya conocida por el autor. Este hecho se tiene en cuenta durante la planificación original del proyecto.

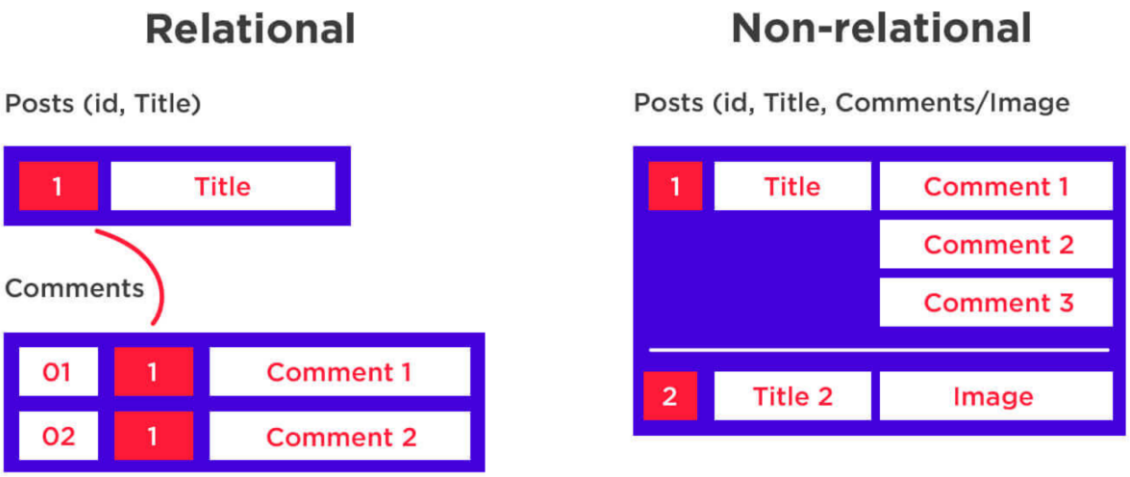

Figura 4: Comparación visual entre la estructura de una base de datos relacional (izquierda) y una no relacional (derecha). [38]

4. Metodología

El Trabajo de Final de Grado es un proyecto extenso, por tanto, necesita una planificación y organización adecuada para conseguir los resultados esperados.

En primer lugar, se ha establecido con el tutor del proyecto realizar reuniones bisemanales. De cada reunión, se extraen tareas alineadas con los objetivos del TFG. Las tareas se deben completar (o haber realizado un notorio progreso) antes de la siguiente reunión.

El tutor y el alumno comparten una carpeta en *Google Drive* donde se almacena toda la información referente al proyecto.

También se utiliza *Google Calendar* como recordatorio y gestor de las reuniones.

Para gestionar las tareas se ha utilizado un documento *Google Sheets* en el que se apunta información referente a cada tarea, la fecha de inicio, la fecha de finalización esperado y, el tiempo dedicado a ella, además de su estado.

Además, el documento permite ver métricas como las horas trabajadas cada semana, el retraso que hay en la entrega de cada tarea, los tipos de tareas que se realizan, … Todo esto con el objetivo de detectar problemas relacionados con la metodología durante el desarrollo del TFG.

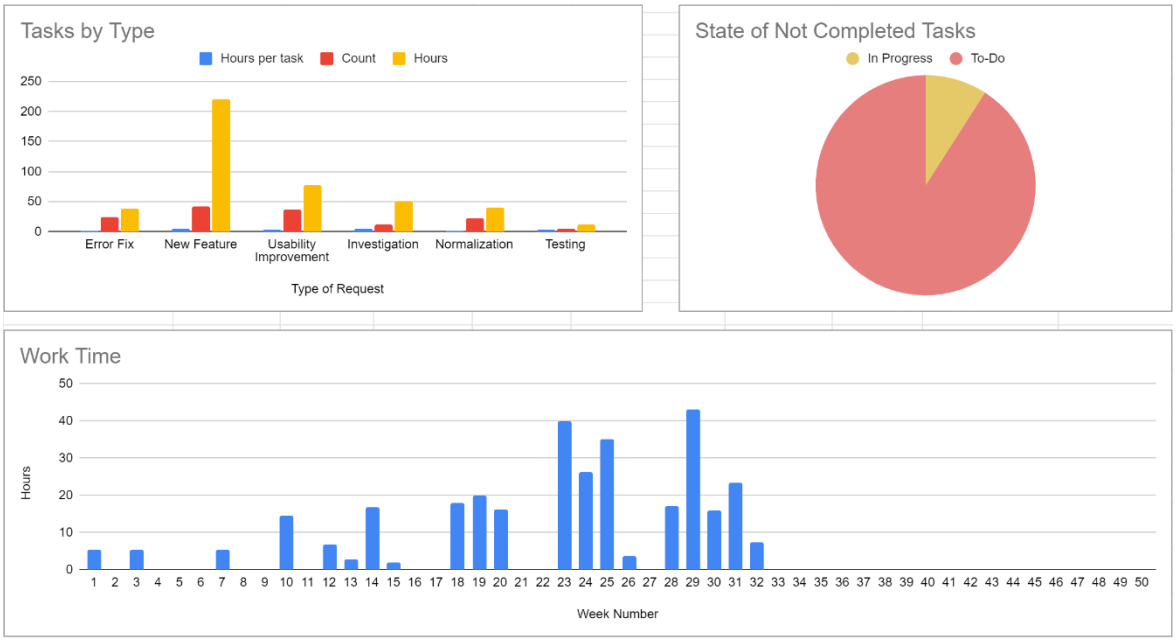

Figura 5: Distintas métricas reportando información sobre las tareas referentes al TFG que se habían realizado y las que no en el momento de la captura.

5. Definición de requisitos funcionales y tecnológicos

El proyecto requiere del desarrollo de una aplicación web que cumpla con los objetivos anteriormente descritos.

Así pues, los siguientes subapartados describen qué requisitos funcionales y tecnológicos se derivan de esos objetivos:

5.1 Conectividad con Wikidata

La aplicación debe obtener todos los elementos sobre los que realizar los cuestionarios de los datos de Wikidata y tener así una cantidad de elementos mayor, más diversa y completa que si se hiciera a mano. Además, este requisito permite la adición y actualización automática de información que puede aparecer en la aplicación.

5.1.1 Filtraje de los datos de Wikidata

Las entradas obtenidas de Wikidata deben filtrarse con el objetivo de utilizar solo aquellas que:

- Sean populares, conocidas y mínimamente relevantes por la mayoría de usuarios.
- No sean ofensivas.

Solo los elementos que pasen el filtro[5] deben ser mostrados en la aplicación.

5.2 Uso de los servicios de AWS

La aplicación debe hacer uso de Amazon Web Services como host para cumplir con todos los objetivos del TFG.

Este host, como se ha descrito anteriormente en el apartado *3.4.1*, debe proveer de “back end”, base de datos y “front end”.

5.3 Sistema de votación

El usuario, al acceder a la aplicación web, debe poder votar entre dos o más elementos para elegir cuál de ellos es su preferido.

El sistema de votación debe incluir elementos gráficos que intenten conseguir que completar encuestas no sea aburrido y aumenten el “*engagement*”.

[5] Se espera que más filtros sean añadidos tras analizar los elementos candidatos.

5.3.1 Registro de los votos en la base de datos

Todas las votaciones realizadas por los usuarios deben registrarse en una base de datos automáticamente.

5.4 Consulta de los resultados

Los usuarios deben poder consultar los resultados a través de la aplicación web si así lo desean.

Estos resultados deben mostrar, ordenadamente, los elementos preferidos entre todos los usuarios.

El ranquin se debe confeccionar utilizando el método de Evan Miller descrito en el apartado *3.3.3 Análisis de resultados*.

5.4.1 Categorización de los resultados

Los usuarios deben poder filtrar por categorías los resultados de las votaciones para poder obtener información más relevante para ellos.

Algunos ejemplos de categorías que se podrían utilizar podrían ser:

- Películas
- Libros
- Personas
- Comida
- Localizaciones
- Banda musical
- Etc.

5.5 Interacción que facilite la recogida de nuevos datos

El diseño de la aplicación debe incluir algún sistema para incrementar la recogida de nuevos datos y aumentar el *engagement*[6] de los usuarios.

Este diseño, que se encuentra aún por determinar, puede ser de distintos tipos como, por ejemplo:

[6] Se entiende como “engagement” la interacción constante, confianza e, incluso, empatización con los valores de la empresa, producto, … por parte de los usuarios [39].

- Limitar el número de consultas que se pueden realizar de los resultados sin responder a cuestionarios. Haciendo que para ver resultados tengas que haber respondido cuestionarios previamente.
- Mostrar siempre en pantalla un cuestionario. Es decir, que, aunque se estén consultando resultados, el usuario pueda responder nuevos cuestionarios constantemente.
- Seguir las prácticas mencionadas en el apartado 3.3.1 sobre cómo realizar un diseño de un cuestionario.

6. Desarrollo

6.1 Desarrollo del punto de partida

Con la intención de crear un punto de partida sobre el que aplicar modificaciones hasta conseguir los objetivos del trabajo se ha seguido el tutorial "Crear una aplicación Flutter" [40] de la plataforma AWS. Este tiene como finalidad guiar al usuario para crear una aplicación con Flutter sencilla con la que se aprende a "administrar el back end en la nube sin servidor con la CLI de AWS Amplify" entre otros.

Para seguir el tutorial no hace falta ningún conocimiento previo de AWS ni de cualquiera de sus servicios. Sin embargo, sí son necesarios algunos conceptos intermedios relacionados con Flutter y Dart, ambos conocidos por el autor, por lo que no ha sido necesario estudiarlos.

Al terminar el tutorial, existe una aplicación básica que sirve como base para crear la aplicación del que este proyecto requiere.

Sin embargo, la aplicación no funciona debidamente cuando es usada a través de un navegador web. Eso se debe a la falta de compatibilidad entre Aplify (back end) y Flutter web (front end) explicada en esta "issue" de GitHub: https://github.com/aws-amplify/amplify-flutter/issues/234

Así pues, se ha decidido optar por otra tecnología/plataforma de back end que sí proporcione un SDK compatible con Flutter web y permita el acceso a una base de datos: Back4App.

Sin embargo, esta decisión va en contra de los objetivos y requisitos definidos anteriormente, pero se ha optado por ella debido a que el back end se ha considerado la tecnología más fácil de migrar. Las desventajas son prácticamente inexistentes y, si en vez de cambiar de back end se migrase de front end, se tendrían que estudiar dos tecnologías en vez de una y, seguramente, la planificación general del proyecto se tendría que ver alterada, posiblemente afectando otros objetivos.

Back4App es un servicio de back end que permite la rápida creación (y con muy poco código) de bases de datos accesibles con una latencia muy baja e incluso a tiempo real. Dispone de planes gratuitos para ser usados durante el desarrollo y uso ligero de la base de datos y, a partir de su despliegue, se pueden usar planes que tienen un coste de a partir de 25€/mes (o 180€/año).

Al haber dejado de usar AWS, la aplicación fruto del tutorial que se había seguido ha sido descartada y se ha desarrollado una aplicación desde cero con una arquitectura que permite la integración con el SDK de Parse para Flutter que Back4App usa.

La aplicación desarrollada finalmente como punto de partida es capaz de establecer una conexión con el back end y permite la visualización de elementos simples en pantalla.

6.2 Muestra de entradas de Wikidata como opciones para votar

Se ha seleccionado aleatoriamente un conjunto de identificadores de entradas de Wikidata. Esto se debe a que la obtención de entradas de manera dinámica aún no ha sido implementada, así que para realizar una primera versión de la muestra de entradas de Wikidata en la aplicación solo es necesario elegir identificadores de una lista que puede ser finita y que contenga elementos seleccionados manualmente.

Con los identificadores de cada entrada se puede realizar peticiones HTTP a Wikidata con la finalidad de obtener como respuesta un JSON que contenga toda la información relevante de la entrada en cuestión. La URL de la petición realizada tiene la siguiente forma: “https://www.wikidata.org/wiki/Special:EntityData/WIKIDATA_ID?format=json” donde “WIKIDATA_ID” representa el identificador de la entrada.

Con el JSON de la respuesta se pueden crear objetos Dart que permiten su fácil uso e integración en la app.

El primer uso que se le ha dado a la información de cada entrada es la correcta visualización de la información más relevante para el usuario encuestado: imagen de la entrada, nombre y descripción corta, tal y como se ve en la Figura 6.

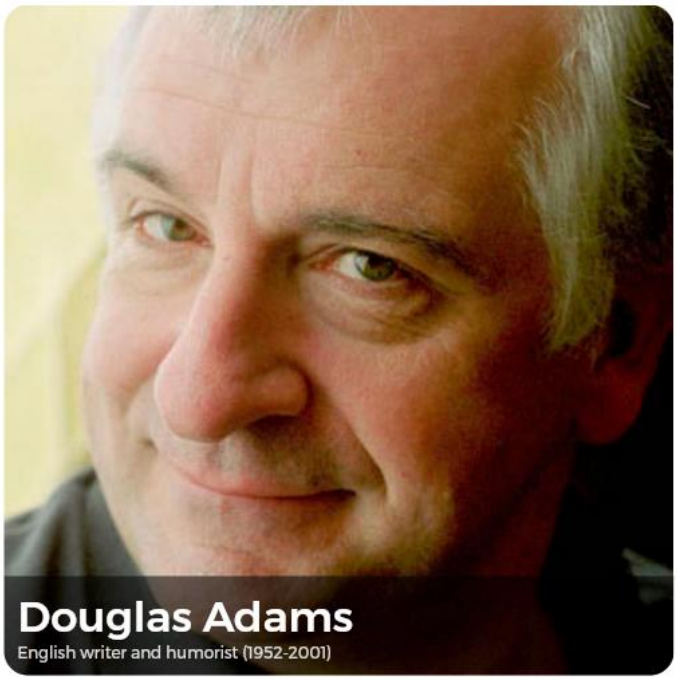

Figura 6: Tarjeta de una entrada desarrollada para la aplicación web para poder visualizar la imagen, nombre y descripción corta de una entrada de Wikidata.

6.3 Votaciones y registro de los votos

Para permitir al usuario votar entre dos entradas de Wikidata se ha optado por mostrar una opción al lado de otra (Figura 7). Además, para realizar la opción de "votar" se ha añadido un botón dedicado a ello que, posteriormente ha sido eliminado para disminuir la densidad de elementos visuales, especialmente en dispositivos móviles. En su lugar la entrada elegida puede ser votada haciendo clic en cualquier lugar de la tarjeta, sin hacer falta ningún botón.

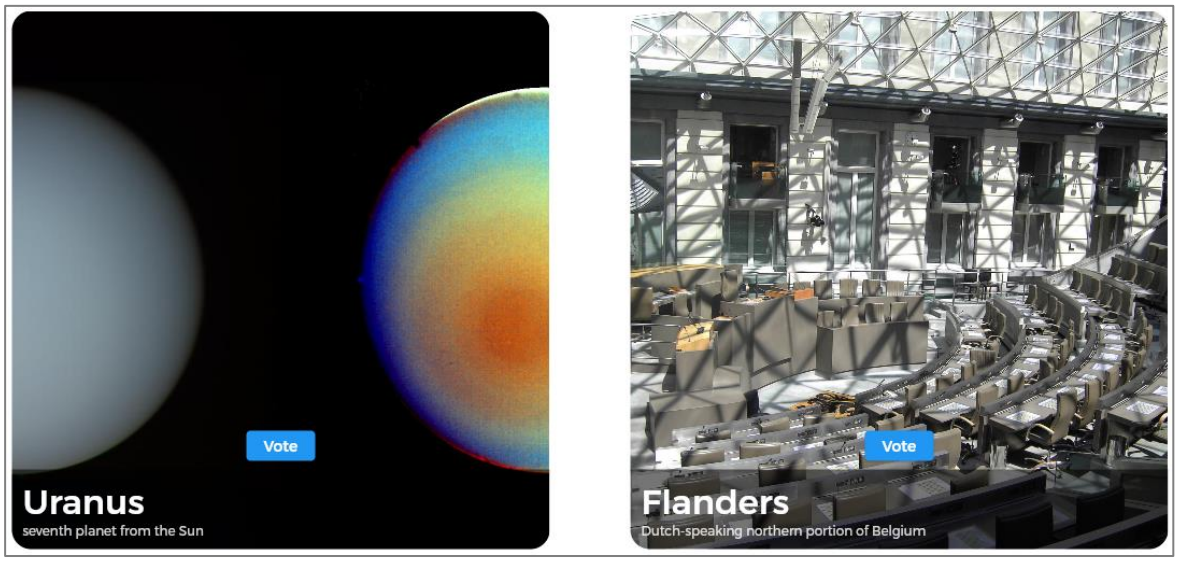

Figura 7: Visualización de la disposición de las tarjetas de las entradas sobre las que el usuario puede realizar sus votos de preferencia. Estas tarjetas incluyen el botón para realizar la votación en sí.

Al elegir una de las dos opciones, se ha programado que ambas se sustituyan por dos opciones nuevas y distintas de la lista que se ha creado previamente.

Además, tras realizar un voto, la información referente a este (entrada ganadora, entrada perdedora y fecha) se guarda en la base de datos.

winner String	loser String	createdAt Date
Q54	Q42	17 Apr 2022 at 07:…
Q23	Q100	8 Apr 2022 at 15:0…
Q534	Q76	8 Apr 2022 at 15:0…
Q234	Q54	8 Apr 2022 at 15:0…

Figura 8: Visualización de información guardada referente a las votaciones hechas en la aplicación. Cada fila representa un voto diferente y en las columnas se puede observar qué entrada ha ganado, cuál ha perdido y cuando se ha producido el voto.

A parte de almacenar la información referente al voto también se almacena información referente a la entrada en sí. Esta información contiene el número total de victorias y derrotas.

entry String*	victories	defeats
Q234	2	1
Q534	2	1
Q42	0	1

Figura 9: Información almacenada en la base de datos referente a las entradas sobre las que se han producido votos. Cada fila representa una entrada diferente y en las columnas se pueden observar su identificador, el número total de victorias y de derrotas.

6.4 Obtención candidatos entre todas las entradas de Wikidata

Inicialmente se ha decidido obtener entradas aleatorias de entre todas las que se encontrasen disponibles en Wikidata. Para ello se ha optado por realizar una petición SPARQL a Wikidata con la ayuda de su servicio "Wikidata Query".

Esta petición ha sido desarrollada con la finalidad de cumplir los siguientes objetivos:

- Obtener entradas populares o conocidas por la gran mayoría de encuestados (usando la cantidad de "sitelinks[7]" como medida de popularidad[8]).
- Que las entradas obtenidas sean aleatoriamente elegidas entre todas las posibles en Wikidata.
- Que no sean seleccionadas entradas de temas indeseados o no entendibles por los usuarios.

[7] Los "sitelinks" (también conocidos como "interwiki links" o "interlanguage links") son enlaces que van desde elementos individuales en Wikidata a páginas en otros sitios de Wikimedia como Wikipedia, Wikisource y Wikivoyage [41].

[8] Este sistema imita al algoritmo "PageRank" originariamente utilizado por Google para medir la importancia de páginas web [42].

```
SELECT ?item ?itemLabel ?itemDescription ?sitelinks
WITH
{
  SELECT ?item ?sitelinks
  WHERE
  {
    #Ensure popularity. Set the minimum number of sitelinks
    ?item wikibase:sitelinks ?sitelinks.
    hint:Prior hint:rangeSafe true.
    FILTER (?sitelinks > 20 )

    #To get random results
    BIND(SHA512(CONCAT(STR(RAND()), STR(?item))) AS ?random)
  }
  ORDER BY ?random
  LIMIT 1000
} AS %subquery1
WITH
{
  SELECT ?item ?sitelinks
  WHERE
  {
    INCLUDE %subquery1

    #Filters to remove undesired entries (templates, categories, ...)
    FILTER NOT EXISTS {?item wdt:P31 wd:Q11266439}
    FILTER NOT EXISTS {?item wdt:P31 wd:Q97950663}
    # FILTER NOT EXISTS {?item wdt:add_more wd:WorkInProgress}
  }
  LIMIT 100
} AS %subquery2
WHERE
{
  INCLUDE %subquery2
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en" . }
}
```

Figura 10: Petición hecha a “Wikidata Query Service” para obtener hasta 100 entradas populares aleatorias y filtradas para ser usadas en las votaciones de la app [43].

Debido a la naturaleza del tamaño de la petición se han tenido que realizar varias optimizaciones para evitar errores de tipo “time-out” esperando la respuesta del servicio.

La petición primero busca 1000 elementos que tengan más de 20 sitelinks y los ordena de manera aleatoria. A continuación, se aplican filtros para eliminar todas aquellas peticiones indeseadas dejando un máximo de 100 viables. A estas 100 se les traduce el identificador

por la etiqueta (nombre) para poder así entender los resultados de un modo más sencillo durante el desarrollo[9].

El resultado llega a la aplicación en formato XML y de este se puede extraer la información necesaria para incluir las entradas en las opciones sobre las que el usuario encuestado puede votar.

6.4.1 Filtraje de las entradas obtenidas

El problema más recurrente con este sistema es la aparición de entradas irrelevantes, aunque consideradas como populares. Estas son entradas dedicadas a la organización interna de Wikidata que se usan muy recurrentemente. Es por esto que se aplican filtros que excluyen de los resultados entradas que sean una instancia (P31) de entradas que estén fuera del árbol principal de conocimientos de Wikimedia [44] o que sean elementos internos tal y como se ve en la Figura 11.

```
FILTER NOT EXISTS {?item wdt:P31/wdt:P279* wd:Q17379835.} #Wikimedia page outside the main knowledge tree
FILTER NOT EXISTS {?item wdt:P31/wdt:P279* wd:Q17442446.} #Wikimedia internal item
```

Figura 11: Filtros aplicados a la petición para evitar la inclusión de elementos no relevantes y relacionados con Wikimedia

6.5 Votaciones divididas por temas

Se ha optado por modificar qué opciones aparecen como candidatas para ser votadas. El cambio implica que se deja de preguntar la preferencia entre entradas sin relación y se pasa a preguntarla entre entradas directamente relacionadas (agrupadas por tema).

Para seleccionar un tema el usuario debe acceder a la página de selección de temas (Figura 12). En ella aparece una lista de temas que se obtiene de forma dinámica. La lista es fruto de una petición SPARQL que busca todas aquellas entradas (consideradas “temas”) en Wikidata. Es decir, que se encuentren como valor (en la propiedad “instance of, P31”) en otras entradas (candidatas a aparecer en las votaciones) que tengan como mínimo 60 sitelinks. La lista se ordena de modo descendiente según la cantidad de entradas (candidatas a voto) que cumplen este requisito por cada tema.

[9] La inclusión de la etiqueta (nombre) de las entradas seleccionadas se puede eliminar y disminuiría el tiempo de ejecución de la petición. Pero durante el desarrollo puede ser de gran ayuda poder leer el nombre de las entradas devueltas por el servicio.

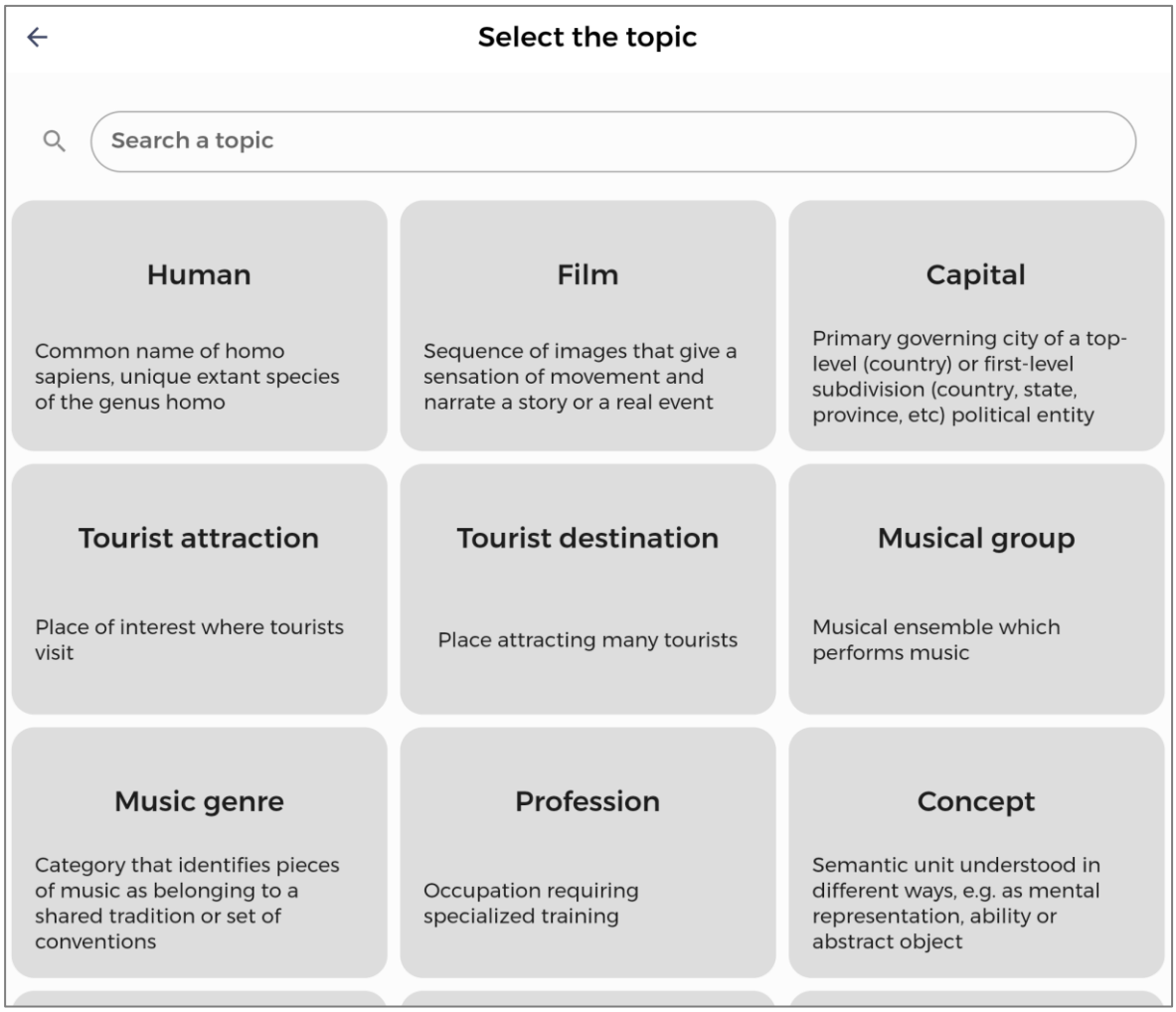

Figura 12: Captura de pantalla de la página de selección de temas.

Con la petición que busca los temas más populares, se ha conseguido que gracias a información dinámica y actualizada las entradas candidatas a ser votadas formen parte de temas relevantes.

Pero para ello, la petición usada para obtener las entradas candidatas de entre todas las disponibles en Wikidata descrita en el apartado 6.4, ha tenido que ser alterada. La nueva versión provee de todas las entradas consideradas relevantes que sean instancia del tema seleccionado con una sola petición, en vez de una lista de candidatos teóricamente aleatorios. La aleatorización se realiza ahora en la lógica del front end en vez de en la petición en sí.

Todos estos cambios se deben al descubrimiento de tres problemas principales que pueden ser resueltos con esta alteración:

6.5.1 Baja aleatoriedad entre las entradas a votar:

A pesar de existir miles de entradas suficientemente populares sobre las que realizar votos, en las pruebas realizadas durante el desarrollo, se ha detectado un nivel de aleatoriedad sospechosamente bajo debido a que un conjunto de ellas se repite con frecuencia[10].

Esto se puede deber a que el sistema utilizado para conseguir resultados aleatorios [45] no funciona correctamente pues no es algo nativamente soportado por SPARQL.

Dividiendo las votaciones por temas se pueden hacer peticiones a Wikidata que devuelvan todas las entradas, filtradas, de un mismo tema. Una petición de este tipo sí que es viable, pero una petición para obtener todas las entradas (de todos los temas, como la anterior) da un error del tipo "timeout" pues la cantidad de datos a recoger y transmitir es demasiado grande.

Con todas las entradas de un mismo tema accesibles en el front end, se puede programar fácilmente un sistema[11] que asegure la aleatoriedad de las entradas mostradas.

6.5.2 Falta de vinculación entre las entradas a votar

Entre los usuarios que han participado en pruebas durante el desarrollo se ha mostrado preocupación por el hecho de que las entradas sobre las hay que elegir son difíciles de comparar. Por ejemplo: "¿Qué te gusta más? ¿La ciencia o el sufragio universal?".

Así pues, agrupando las entradas por tema aparecen comparativas más sencillas de valorar. Por ejemplo: "¿Qué te gusta más? ¿El color naranja o el color verde?".

De este modo el usuario parece más satisfecho y realiza más votos en el mismo periodo de tiempo, pues no le cuesta tanto decidirse por una de las dos opciones.

6.5.3 Baja relevancia de entradas a votar

Al usar el mismo número de sitelinks para todas las entradas, resulta que muchas entradas conocidas no son mostradas nunca, mientras que otras que sí se muestran no son realmente conocidas (debido a tener más sitelinks que los necesarios).

Esto se puede observar comparando las entradas referentes a humanos con las entradas que hacen referencia a las películas. De humanos con más de 80 sitelinks existen más de 2.200,

[10] Aproximadamente, por cada cien votos realizados, varias entradas se muestran en más de una ocasión.

[11] El sistema desarrollado usa una lista que contiene todas las entradas que se pueden mostrar y de la que se van extrayendo de manera aleatoria.

mientras que de películas hay 45 (y es posible que la mayoría de usuarios pueda reconocer más de 45 películas y menos de 2.200 humanos).

Así pues, existen temas que requieren de una cantidad de sitelinks distintos a otros para tener una muestra representativa de entradas sobre las que votar.

Para solventar este problema, aparte de dividir las votaciones por tema, se almacena en la base de datos la cantidad de sitelinks mínimos que las entradas de cada tema deben tener para aparecer en las votaciones.

Cada vez que se selecciona un tema por un usuario se realiza una petición a Wikidata para recibir las entradas con más sitelinks de los establecidos en la base de datos (siendo 40 para aquellos temas que no se eligen por primera vez). Si se devuelven menos de 15 entradas, el número de sitelinks mínimo decrece y se vuelve a realizar la petición hasta que se devuelvan 15 o más entradas o el número de sitelinks sea inferior a 0[12].

A parte del balanceo del número mínimo de sitelinks según el número de entradas, existe un balanceo fruto de la interacción de los usuarios: si los usuarios votan entradas como desconocidas, el número de sitelinks mínimo por el tema aumenta 0.25, pero si votan una entrada se supone que la reconocen y el número de sitelinks mínimo del tema disminuye en 0.1.

De este modo se pretende encontrar un equilibrio automático aprovechando la interacción de los usuarios, siendo ellos mismos los que determinen cuánto de popular debería ser una entrada en cada tema de manera inconsciente.

6.6 Almacenamiento de datos referentes al usuario

Con la finalidad de permitir la realización de análisis estadísticos y posibilitar una visualización de los resultados con más riqueza de filtrados, se ha optado por almacenar información del usuario con cada respuesta.

La información almacenada de forma automática comprende el número de veces que se ha accedido a la página web y el número de votaciones que ha hecho tanto en total como en la sesión actual.

[12] Si se da el caso de que el número de sitelinks de un tema sea inferior a 0, significa que el tema en cuestión no tiene suficientes entradas como para poder realizar un ranquin, así que aparece un mensaje en pantalla anunciando al usuario de que el tema seleccionado no tiene suficientes entradas.

Además, se ha creado una sección en la que los usuarios pueden voluntariamente proporcionar información adicional como su país, género y edad. Esta información también se relaciona a cada uno de los votos registrados.

Figura 13: Configurador de datos personales a disposición del usuario encuestado por si quiere compartir información adicional sobre su persona acerca de su país, género y edad.

Sin embargo, se desconoce la fiabilidad que pueden tener estos últimos datos ni qué cantidad de usuarios optará a proporcionarlos.

6.7 Almacenamiento de las propiedades de las entradas

Con la finalidad de poder conseguir realizar un filtrado atractivo y funcional de los resultados de las votaciones, se ha detectado la necesidad de almacenar los valores de las propiedades de cada entrada.

Así que se ha decidido que, tras realizar una votación y que se mande la notificación al servidor para almacenar este hecho en la base de datos, el servidor, una vez ha almacenado los datos referentes a la votación, responde a la aplicación informándole de la última vez que se han actualizado las propiedades de cada una de las dos opciones que han participado en la votación (tanto la ganadora como la perdedora).

Si no se han actualizado nunca las propiedades de una entrada en la base de datos (es decir, es la primera vez que se realiza una votación que involucra esa entrada) o si hace más de una semana que no se actualizan, la aplicación manda una petición de actualización de las propiedades (enviando toda la información sobre estas al servidor) y el servidor actualiza los valores que tiene guardados. Si es necesario, el servidor crea "columnas[13]" nuevas dinámicamente si se encuentra con propiedades que nunca han sido contenidas en ninguna de las entradas que se ha almacenado anteriormente.

6.7.1 Filtraje y selección de las propiedades de interés

Existen casi 10.000 propiedades diferentes en Wikidata [46]. Muchas de ellas se pueden considerar irrelevantes o inútiles para el uso que se les quiere dar en este proyecto (un ejemplo es la P1580, "University of Barcelona authority ID", que no solo está obsoleta si no

[13] Una "columna" hace referencia a un atributo de una clase almacenada en la base de datos.

que está relacionada con un tipo de información que un humano sería incapaz de comprender sin conocimientos muy específicos).

La existencia de miles de propiedades irrelevantes genera un problema. Con el sistema programado de almacenaje de las propiedades, se almacenan todas las propiedades de una entrada, sean o no de interés. Esto puede limitar la capacidad del servidor (pues una sola entrada puede tener miles de propiedades) y dificultar el filtraje de los resultados.

Para analizar las propiedades se ha creado una página de uso interno dedicada a la valoración de las propiedades de manera ágil. Esta página muestra todas las propiedades que han aparecido en cualquier entrada votada para que, al menos durante el desarrollo, se puedan ir fácilmente descartando o seleccionando para guardar su información en la base de datos.

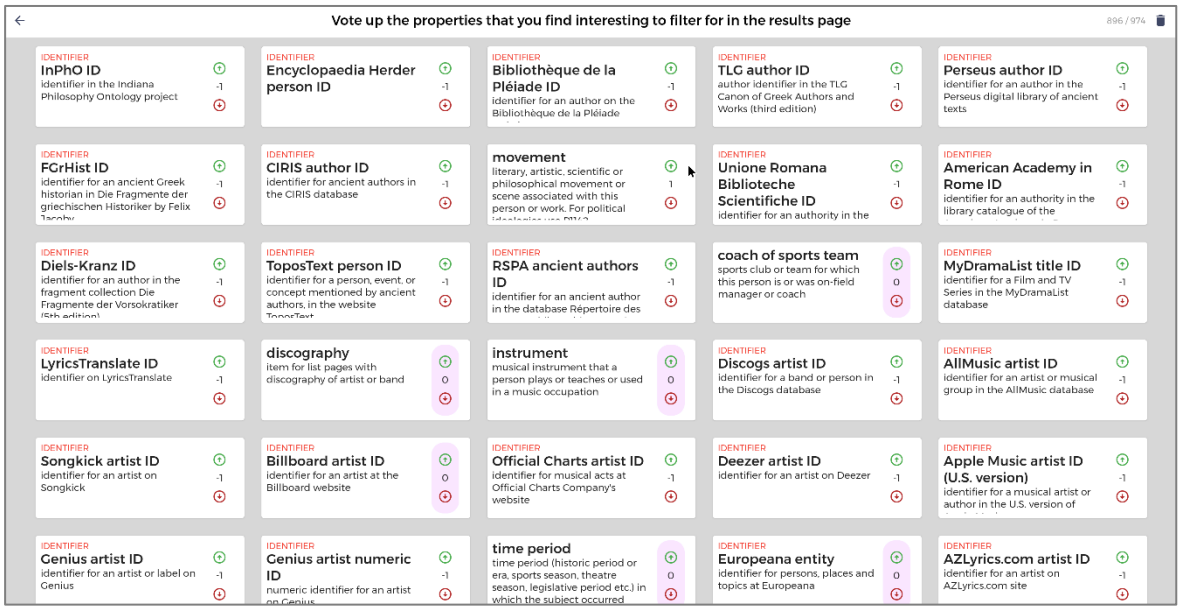

Figura 14: Página de visualización de las propiedades registradas en la base de datos. Permite su positiva o negativa valoración para ser posiblemente incluidas en los filtros de los resultados.

Se ha planteado la posibilidad de permitir a los mismos usuarios comunicar su interés por cada propiedad, pero se ha descartado por los siguientes motivos:

- Se quiere maximizar el número de votaciones entre entradas e integrar otro tipo de encuesta en la aplicación podría reducir el número de respuestas totales de la encuesta principal.

- La gran mayoría de propiedades no son de interés (de las primeras 257 valoradas, solo 26, un 10%, se han considerado de interés[14]). Hasta el punto donde votar si lo son o no puede resultar confuso y repetitivo.

6.8 Consulta de los resultados de las encuestas

Los resultados de las votaciones se pueden visualizar en una página distinta a la de las votaciones.

La página de los resultados muestra una lista ordenada por puntuación de las entradas sobre las que los encuestados han votado (Figura 15).

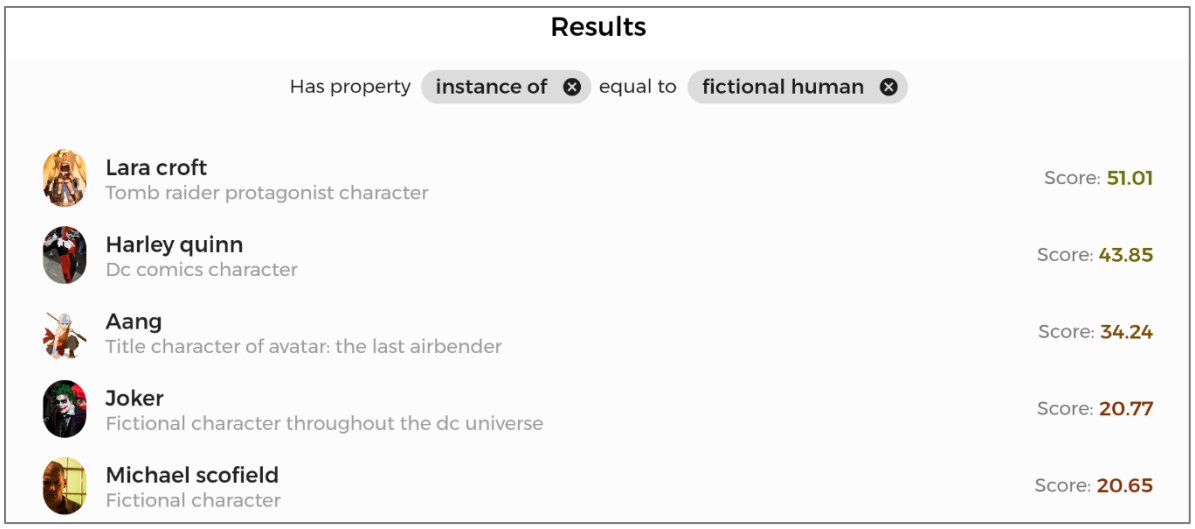

Figura 15: Visualización ordenada de las entradas que son una instancia de un personaje humano ficticio ordenadas de más a menos puntuación en la página de resultados.

La puntuación se calcula usando el algoritmo de Evan Miller [25], descrito en la sección 3.3.3 Análisis de resultados, que expone como realizar una correcta ordenación de los resultados obtenidos en un estudio estadístico de valoración.

6.8.1 Filtraje en la visualización de los resultados

Gracias al almacenamiento de las propiedades y sus valores, en la página de resultados se puede filtrar qué entradas aparecen listadas.

La configuración de los filtros se puede observar en la zona superior de la Figura 15. Éstos tienen dos partes con efectos diferentes:

[14] Al finalizar el proyecto se han valorado un total de 2123 propiedades, siendo 1863 descartadas y 260 seleccionadas, un 12% del total.

- **Propiedad (*has property*):** Si se escoge una propiedad, en los resultados se muestran solamente aquellas entradas que contengan la propiedad seleccionada con cualquier valor. Ejemplo: "director" muestra todas aquellas entradas que tengan un director.
- **Valor (*equal to*):** Si se selecciona una opción, se determina un valor exacto de la propiedad seleccionada que deben tener las entradas a mostrar. No se puede seleccionar ningún valor si no está seleccionada una propiedad. Ejemplo: "Tim Burton" (habiendo seleccionado "director") muestra todas aquellas entradas en las que Tim Burton esté identificado como el director.

Para facilitar la visualización de los resultados relevantes y demostrar el funcionamiento de los filtros al usuario, al acceder a la página de resultados, se aplican automáticamente los filtros necesarios para mostrar sólo las entradas relacionadas con el tema sobre el que se están realizando las votaciones. Sin embargo, estos filtros se pueden cambiar a discreción del usuario.

6.9 Diseños para incrementar el número de votos

Con el objetivo de aumentar el número de votos que se realizan se ha decidido que la página de inicio de la aplicación sea la página en la que se realizan las votaciones y que esta sea la que da acceso a la selección de los temas y a la página de resultados.

Adicionalmente, se ha creado un sistema para poder compartir el enlace a la aplicación web con un tema preseleccionado (Figura 16). De este modo, las comunidades y grupos tienen la posibilidad de compartirse fácilmente temas que resulten interesantes sin tener que buscar el tema en la página.

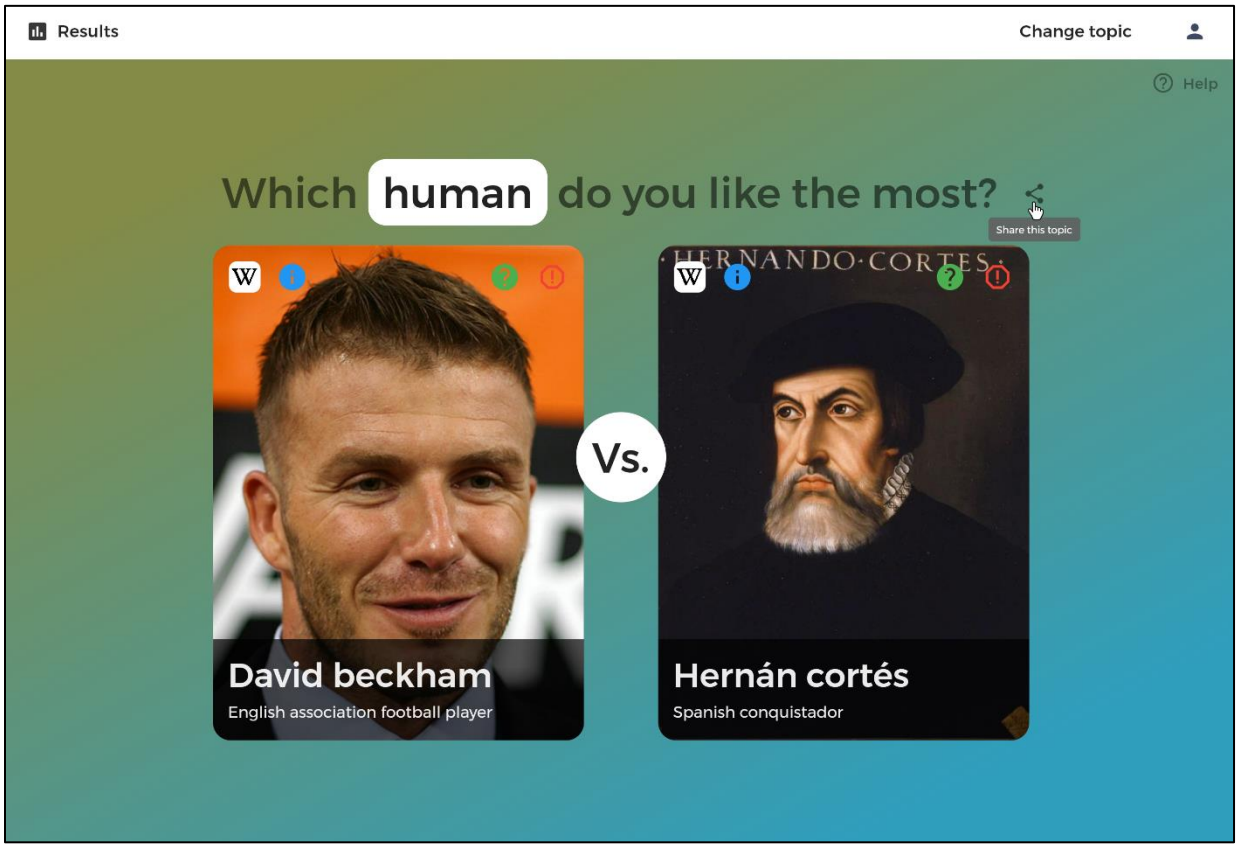

Figura 16: Captura de la página de visualizaciones con el ratón encima del botón para copiar el enlace al tema seleccionado (“humano” en el caso de la captura).

Finalmente, se ha decidido mostrar la puntuación global de cada entrada tras realizar un voto (Figura 17), de este modo se pretende satisfacer la curiosidad de los usuarios sin que tengan que ir a consultar la página de los resultados. Es decir, que pueden consultar parcialmente los resultados mientras contestan encuestas.

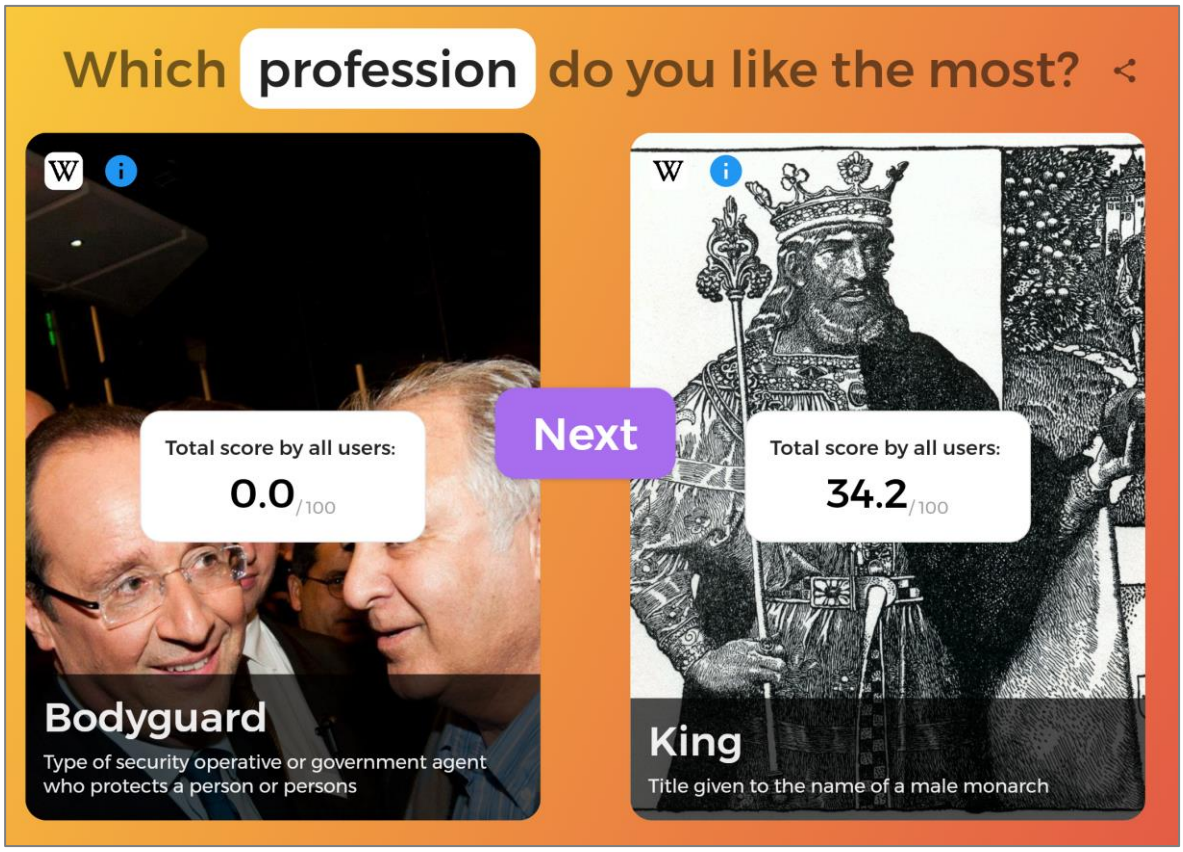

Figura 17: Captura de pantalla de la aplicación tras realizar un voto, en este caso sobre la opción de la derecha.

Las modificaciones realizadas hasta el momento se alinean con el diseño de cuestionario propuesto por P. Samuels [23] (preguntas claras, concisas y con un título significativo, se facilitan siempre respuestas excluyentes, no aparecen opciones difíciles de comparar gracias a las encuestas por temas, …).

6.10 Optimización de las peticiones a Wikidata

Durante las pruebas realizadas a lo largo del desarrollo se han detectado errores relacionados con las respuestas de Wikidata a las peticiones de JSONs sobre la información de las propiedades de las entradas. Los errores se han manifestado especialmente en la página de resultados, donde se realizan un gran número de peticiones simultáneas, siendo el número de peticiones la fuente de los errores expuestos.

Para solucionar el problema, se ha diseñado una petición utilizando el servicio de SPARQL que devuelve la información mínima necesaria de las entradas deseadas. La información devuelta consta solo de la etiqueta (nombre), descripción e imagen. Es decir, que no se puede utilizar para recabar información sobre las propiedades de las entradas, así que sólo se puede

usar cuando ya se conocen exactamente las entradas de las que se requiere la información básica y no se necesiten los datos referentes a las propiedades.

Sin embargo, las peticiones usando el servicio SPARQL tienen un límite de caracteres y, por la naturaleza de la petición, cada entrada sobre la cual se quiere obtener la información requiere de la adición de ciertos caracteres. Así pues, para conseguir obtener la información de tantas entradas como se quiera, es necesario encadenar peticiones sin que ninguna supere el límite de caracteres permitido.

6.11 Estructura de la base de datos

El servicio de back end (Back4App) utiliza el framework Parse y tiene una base de datos MongoDB.

Esta se organiza por clases y para esta aplicación se han creado las siguientes:

- **Entry**: Contiene información relacionada con una entrada de Wikidata sobre la que se han realizado votos. Sus columnas/atributos principales son:
    - **Entry**: Identificador de la entrada.
    - **Victories**: Cantidad de veces que ha sido elegida frente a otra entrada.
    - **Defeats**: Cantidad de veces que no ha sido elegida frente a otra entrada.
    - **PropertiesLastUpdate**: Fecha de la última vez que se han actualizado los valores de sus propiedades.
    - **Sitelinks**: Número de sitelinks que tiene la entrada.
    - **Múltiples "PX"** (siendo X el número identificador de una propiedad): Valor de cada propiedad. Null si la entrada no la contiene.

- **InstanceOf**: Comprende información de los temas sobre los que se han realizado votaciones. Sus columnas/atributos principales son:
    - **InstanceOfId**: Identificador del valor de la propiedad "instance of" (P31) que las entradas del tema deben tener. Identifica al tema.
    - **MinimumSiteLinks**: Número mínimo de sitelinks que las entradas de este tema deben tener.

- **Property**: Contiene información relativa a la importancia de cada una de las propiedades comprendidas en cualquier entrada sobre la que se han realizado votos. Sus columnas/atributos principales son:
    - **Property**: Identificador de la propiedad.
    - **Score**: Puntuación dada a la propiedad para determinar si sus valores se van a guardar o no (en la clase "Entry").

- **Report**: Incluye toda la información referente a los deseos de un usuario de intercambiar una entrada sobre la que se proponía votar (ya sea por desconocimiento o por inadecuada). Sus columnas/atributos principales son:
    - **Entry**: Identificador de la entrada intercambiada/reportada.
    - **UnknownReports**: Número de veces que se ha reportado como desconocida.
    - **InnappropiateReports**: Cantidad de veces que se ha reportado como inapropiada.
    - **Sitelinks**: Número de sitelinks de la entrada reportada.

- **Response**: Contiene la información relativa a cada uno de los votos realizadas. Sus columnas/atributos principales son:
    - **Winner**: Identificador de la entrada seleccionada por el usuario.
    - **Loser**: Identificador de la entrada no seleccionada por el usuario.
    - **Country**: País del usuario indicado por el mismo.
    - **Age**: Edad del usuario indicado por el mismo.
    - **Gender**: Género del usuario indicado por el mismo.
    - **UserAccessTime**: Número de veces que el usuario ha accedido a la página.
    - **UserVotingTime**: Número votos totales que el usuario ha realizado.
    - **UserSessionVotingTime**: Número de votos que el usuario ha realizado durante esta sesión.

6.12 Otras dificultades encontradas durante el desarrollo

A lo largo del desarrollo han ido apareciendo contratiempos que requieren de distintas soluciones. Estos son algunos de ellos:

6.12.1 Información inconsistente

Cada entrada de Wikidata es distinta. No existe un idioma común para todas, no es necesario que tengan propiedades clave como la etiqueta (o nombre), la descripción, o imagen…

Así que hay que programar la aplicación para que sea capaz de adaptarse a esas situaciones. Se puede diseñar la aplicación para que se realicen ajustes visuales para adaptarse a la falta de información o, por ejemplo, que las imágenes, en caso de que la entrada no tenga tal propiedad, se busquen en la Wikipedia.

6.12.2 Distintos formatos de archivos

Cada imagen obtenida puede tener un formato distinto (es común encontrarse imágenes .png, .jpg, .svg, .tif, …), así que la aplicación debe soportar tantos formatos como sea posible.

Desafortunadamente, Flutter no soporta de manera nativa muchos formatos, así que el uso de plugins para poder mostrar imágenes con más formatos es necesario.

Sin embargo, esto no es suficiente, pues las imágenes de tipo .tif requieren de una decodificación y recodificación en formato .jpg para poderse visualizar usando Flutter. Sin embargo, este proceso requiere de una gran cantidad de recursos que puede congelar la interfaz durante varios segundos a causa de las limitaciones de Flutter relacionadas con el multithreading. Para solucionar este inconveniente se ha decidido intentar evitar las imágenes .tif y, si no se encuentra ninguna alternativa, evitar su visualización en pantalla.

6.12.3 Limitación de peticiones hacia el servicio de back end

Debido a que Back4App contabiliza en gran medida el uso de los recursos con la cantidad de peticiones realizadas, es necesario intentar limitarlas al máximo.

Es por este motivo que se ha aprovechado la existencia de “Cloud Code” para reducir el número de peticiones y, a la vez, aumentar la velocidad de algunos procesos.

Con Cloud Code se puede programar lógica en el servidor en vez de en el front end. De este modo, si alguna parte del código requiere de lógica entre peticiones o de guardar información en distintas clases de la base de datos, éstas se pueden realizar en el servidor y no hace falta hacer nuevas peticiones para cada consulta a la base de datos.

Un ejemplo lo encontramos con el guardado de votos. En el proceso se guarda una nueva fila/entrada en la clase de “Response”, pero también se actualiza o se crea una en la clase “Entry”. Si la lógica se encontrase en el front end no habría otra opción que la de realizar

dos peticiones distintas a la base de datos. Sin embargo, al estar la lógica en el servidor, se puede realizar una sola petición al Cloud Code para que guarde la información que se le hace llegar relativa a un voto en tantas clases como sea necesario.

6.12.4 Tiempos de carga demasiado largos

Se ha detectado la existencia de fugas de memoria relacionadas con peticiones realizadas a SPARQL. El problema se manifiesta cuando el usuario sale de una página en la que se estaba esperando una respuesta a una petición a Wikidata. Para solucionarlo se puede cancelar la petición cuando el usuario cierre la página.

Relacionado con los tiempos de carga, también se ha detectado que algunas peticiones, como la que busca temas relevantes en Wikidata puede llegar a tardar más de un minuto. Así pues, para evitar la frustración del usuario, se ha revisado la lista de los temas más populares y se han seleccionado temas que se han considerado llamativos. De este modo, en la página de selección de temas, mientras se espera respuesta a la petición que devuelve la lista de temas de Wikidata, se pueden mostrar inmediatamente alrededor de 100 temas entre los que el usuario puede elegir sin tener que esperar.

6.13 Despliegue

Para realizar el despliegue de la aplicación se ha optado para aprovechar la posibilidad que brinda Back4App de alojar la página web además de proveer de un subdominio para acceder a ella: https://wikipoll.b4a.app/

Para poder desplegar la aplicación en Back4App primero hay que realizar una “build” del código Flutter para que este se adapte al código HTML/JavaScript necesario para que funcione en un navegador web accediendo desde una URL. Una vez creada la “build” se pueden subir los archivos al servidor utilizando su herramienta CLI [47].

6.13.1 Pruebas de uso

Las pruebas se realizan mayoritariamente con videollamadas, pero también en persona.

Los usuarios acceden a la página web e interactúan con ella durante un periodo de tiempo de al menos 5 minutos.

Durante el periodo de prueba se presta atención a sus reacciones, errores (tanto de la aplicación como cometidos por el usuario), dificultades, opinión, …

La información recabada se estudia y se intenta llegar a una conclusión. Si la conclusión lo requiere, se establecen tareas con el objetivo de mejorar la experiencia de uso de la página web.

Algunos ejemplos de cambios realizados frutos de la observación de las pruebas son:

- *Feedback* visual al votar para dejar claro que el voto ha sido registrado.
- Muestra de la puntuación de cada opción tras realizar el voto para satisfacer la curiosidad del usuario.
- Inclusión de campos de búsqueda en el selector de temas y filtros.
- Inclusión de una sección de "ayuda" que contenga información relevante.
- Adición de textos en botones que inicialmente sólo contaban con iconos (como el de acceso a la página de resultados) debido a la confusión del significado de los iconos.
- Cambio del formato de la encuesta de entre entradas no relacionadas a encuestas entre entradas relacionadas por tema debido por la existencia de dificultades para compararlas.
- Muestra de temas preseleccionados que no requieran de una carga lenta para poderse elegir para evitar que sea tedioso cambiar de tema.

Tras la realización de los cambios mencionados, los usuarios, mientras interactúan con la página web, describen la experiencia como fluida y valoran positivamente el ciclo de votaciones, mostrándose cómodos mientras completan múltiples.

7. Conclusiones

El proyecto se ha completado satisfactoriamente y su desarrollo ha permitido comprender ampliamente las herramientas que Wikimedia Fundation provee con Wikidata, APIs y más, además de demostrar su posible utilización para facilitar el desarrollo de aplicaciones que necesiten gran cantidad de datos.

Estas herramientas son capaces de proporcionar el acceso a la información necesaria para muchas aplicaciones. Sin embargo, cuentan con ciertas limitaciones normalmente adheridas a la naturaleza de cada proyecto que requieren de la búsqueda de soluciones alternativas. Un ejemplo descubierto durante el desarrollo de este proyecto sería la imposibilidad de obtener entradas completamente aleatorias pero filtradas directamente de alguna fuente de Wikidata.

La metodología utilizada para gestionar las tareas y las reuniones prácticamente semanales, ha facilitado un seguimiento y planificación precisos que ha permitido conseguir un producto final acabado a pesar de haberse encontrado con la imposibilidad de usar AWS junto Flutter web. La decisión de cambiar de servicio de back end a Back4App no ha dificultado que se cumplieran los demás objetivos y ha permitido una amplia exploración y dedicación de recursos a trabajar con las herramientas de Wikidata.

Flutter y Back4App han demostrado ser una buena combinación de front end y back end gracias a la fácil comunicación que permiten entre ellos. Además, facilitan una gran fluidez (tanto visual como en las comunicaciones con el servidor) que ha sido apreciada por múltiples usuarios. Este hecho, junto a muchos otros, pero especialmente el recibimiento de *feedback* de los usuarios, ha ayudado al cumplimiento del objetivo de crear un diseño atractivo y que facilite la recogida de nuevos votos.

Sin embargo, la falta de grandes cantidades de usuarios hace que un producto como el desarrollado haga perder el interés a potenciales nuevos usuarios, pues una página centrada en estadísticas puede perder mucho valor si las muestras con las que se trabajan no son ni representativas ni suficientemente grandes. A pesar de esto, al almacenar todos los datos y haber creado un sistema acumulativo en el que cada una de las votaciones (contenido creado por el usuario) no pierde valor con el tiempo, la página puede potencialmente aumentar su valor a lo largo de los meses si se consigue que haya usuarios accediendo esporádicamente.

Siendo la “inmortalidad” de los datos, una idea posiblemente a contemplar y perseguir si se quiere crear un software que requiera de datos creados por los usuarios para funcionar.

7.1 Reflexión personal

Este proyecto ha permitido explorar y entender las posibilidades de las herramientas de Wikidata para poderlas tener en cuenta en futuros proyectos además de posibilitar la puesta en práctica de muchos conocimientos de la carrera como la gestión de base de datos, APIs, arquitecturas, …

También ha conseguido demostrar que hasta los servicios más genéricamente recomendados (como AWS por su amplia variedad de herramientas y gran popularidad) no son siempre la mejor opción y es necesario estudiar todas las herramientas principales para asegurarse de que pueden trabajar conjuntamente sin problemas importantes y pueden cubrir todos los requisitos funcionales.

Gracias a intentar confeccionar una aplicación sencilla como base del proyecto, se ha conseguido manifestar rápidamente la imposibilidad de trabajar con Flutter Web junto a AWS Amplify. Si esta imposibilidad no se hubiera manifestado hasta un punto bastante más avanzado del desarrollo del proyecto, podría haber causado grandes disrupciones afectando notoriamente el resultado final.

Adicionalmente, el proyecto también ha evidenciado que, por mucho que haya un software completo, sin usuarios, el producto puede ser muy difícil de mejorar por la falta de *feedback* o hasta inútil si depende del contenido o datos generados por los usuarios para tener valor. Así pues, una mayor atención al márquetin relacionado con el producto se considera ahora esencial para conseguir un grueso de usuarios significativo y aumentar así el valor del producto.

8. Posibles ampliaciones

Algunas ampliaciones de funcionalidad que se podrían implementar en la aplicación web son:

- Obtención de datos de otras fuentes de información (DBpedia, tendencias en redes sociales, vídeos de YouTube, APIs de buscadores, mapas digitales, …).
- Mejora de la aplicación de los filtros en la página de resultados para que estos permitan valores relacionados con rangos. Sería útil para visualizar, por ejemplo, ránquines de entradas con un coste superior a cierta cantidad o batallas históricas a menos de una determinada distancia de una ciudad.
- Capacidad de aplicar múltiples filtros para poder hacer que las entradas del ranquin requieran todos los filtros (o sólo uno de ellos), permitiendo, por ejemplo, consultar un ranquin de personas casadas.
- Conseguir que los usuarios que quieran puedan votar sin tener ningún tema seleccionado, mostrando entradas de cualquier tema.
- Adición de mecánicas jugables para aumentar la satisfacción y el *engagement* del usuario.
- Permitir el acceso rápido a servicios que permitan consultar información de la entrada. Por ejemplo, si una entrada dispone de unas coordenadas, poder abrir con un clic un servicio que permita visualizar la localización en un mapa interactivo.

Además, aparte de ampliaciones funcionales, tal y como se ha mencionado en las conclusiones, realizar una ampliación del tiempo dedicado a la difusión y márquetin de la aplicación podría ser altamente beneficioso para la misma.

9. Bibliografía

[1] «Wikidata». https://www.wikidata.org/wiki/Wikidata:Main_Page (accedido 17 de diciembre de 2021).

[2] «Cloud Services - Amazon Web Services (AWS)», *Amazon Web Services, Inc.* https://aws.amazon.com/ (accedido 8 de febrero de 2022).

[3] Tom Scott, *1,204,986 Votes Decided: What Is The Best Thing?*, (7 de septiembre de 2020). Accedido: 3 de febrero de 2022. [En línea Video]. Disponible en: https://www.youtube.com/watch?v=ALy6e7GbDRQ

[4] T. Scott, «What Is The Best Thing?», 21 de agosto de 2020. https://web.archive.org/web/20200821153707/http:/best.tomscott.com/ (accedido 3 de febrero de 2022).

[5] «Most Awesomest Thing Ever». http://mostawesomestthingever.com/ (accedido 3 de febrero de 2022).

[6] J. Rickard y T. Davies, «Which Is The Best Thing?» https://web.archive.org/web/20210309053712/https:/whichisthebestthing.com/ (accedido 3 de febrero de 2022).

[7] «Which Gets Googled More? Play The Higher Lower Game», *The Higher Lower Game*. http://www.higherlowergame.com (accedido 3 de febrero de 2022).

[8] «Help:About data - Wikidata». https://www.wikidata.org/wiki/Help:About_data#Understanding_Wikidata (accedido 17 de diciembre de 2021).

[9] «Wikidata:Introduction». https://www.wikidata.org/wiki/Wikidata:Introduction (accedido 17 de diciembre de 2021).

[10] «Help:Statements - Wikidata». https://www.wikidata.org/wiki/Help:Statements#Values (accedido 17 de diciembre de 2021).

[11] «Wikidata:Introduction - Wikidata». https://www.wikidata.org/wiki/Wikidata:Introduction#/media/File:Datamodel_in_Wikidata.svg (accedido 8 de febrero de 2022).

[12] «Wikidata:Data access». https://www.wikidata.org/wiki/Wikidata:Data_access#How_can_I_get_data_out_of_Wikidata? (accedido 17 de diciembre de 2021).

[13] «What is a Database Dump? - Definition from Techopedia», *Techopedia.com*. http://www.techopedia.com/definition/23340/database-dump (accedido 21 de enero de 2022).

[14] «JSON Dump Files — qwikidata 0.4.0 documentation». https://qwikidata.readthedocs.io/en/stable/json_dump.html (accedido 21 de enero de 2022).

[15] «API:Recent changes stream - MediaWiki». https://www.mediawiki.org/wiki/API:Recent_changes_stream (accedido 21 de enero de 2022).

[16] «Linked Data Interface — qwikidata 0.4.0 documentation». https://qwikidata.readthedocs.io/en/stable/linked_data_interface.html (accedido 21 de enero de 2022).

[17] «API:Main page - MediaWiki». https://www.mediawiki.org/wiki/API:Main_page (accedido 21 de enero de 2022).

[18] «SPARQL End Point — qwikidata 0.4.0 documentation». https://qwikidata.readthedocs.io/en/stable/sparql.html (accedido 21 de enero de 2022).

[19] «SPARQL vs SQL», *Cambridge Semantics*. https://cambridgesemantics.com/blog/semantic-university/learn-sparql/sparql-vs-sql/ (accedido 21 de enero de 2022).

[20] «Wikidata:Bots». https://www.wikidata.org/wiki/Wikidata:Bots (accedido 21 de enero de 2022).

[21] «FOAF Vocabulary Specification». http://xmlns.com/foaf/spec/ (accedido 21 de enero de 2022).

[22] «RDF 1.1 Primer». https://www.w3.org/TR/rdf11-primer/ (accedido 8 de febrero de 2022).

[23] P. Samuels, «Statistical Methods 2. Questionnaire Design». Accedido: 21 de enero de 2022. [En línea]. Disponible en: https://www.statstutor.ac.uk/resources/uploaded/2questionnairedesign2.pdf

[24] «Bernoulli trials - Encyclopedia of Mathematics». https://encyclopediaofmath.org/index.php?title=Bernoulli_trials (accedido 11 de enero de 2022).

[25] E. Miller, «How Not To Sort By Average Rating». https://www.evanmiller.org/how-not-to-sort-by-average-rating.html (accedido 11 de enero de 2022).

[26] «Build Apps for Free on Heroku | Heroku». https://www.heroku.com/free (accedido 8 de febrero de 2022).

[27] «Help: Toolforge/Web - Wikitech». https://wikitech.wikimedia.org/wiki/Help:Toolforge/Web (accedido 21 de enero de 2022).

[28] B. Inc, «Accelerate Backend Development | Back4App», *Back4App™*. https://www.back4app.com/features (accedido 3 de junio de 2022).

[29] «Welcome to The Apache Software Foundation!» https://www.apache.org/ (accedido 8 de febrero de 2022).

[30] «Advanced Load Balancer, Web Server, & Reverse Proxy», *NGINX*. https://www.nginx.com/ (accedido 8 de febrero de 2022).

[31] «Flutter - Build apps for any screen». //flutter.dev/ (accedido 8 de febrero de 2022).

[32] Ionic, «Cross-Platform Mobile App Development», *Ionic Framework*. https://ionicframework.com/ (accedido 8 de febrero de 2022).

[33] «The web framework for perfectionists with deadlines | Django». https://www.djangoproject.com/ (accedido 8 de febrero de 2022).

[34] «ASP.NET | Open-source web framework for .NET», *Microsoft*. https://dotnet.microsoft.com/en-us/apps/aspnet (accedido 8 de febrero de 2022).

[35] A. Lafuente, «Bases de datos relacionales vs. no relacionales: ¿qué es mejor?», *Aukera*, 1 de agosto de 2018. https://aukera.es/blog/bases-de-datos-relacionales-vs-no-relacionales/ (accedido 11 de enero de 2022).

[36] «Las mejores bases de datos en la nube». https://www.tecnologias-informacion.com/basedatosnube.html (accedido 8 de febrero de 2022).

[37] L. Arora, «NoSQL Databases Every Data Scientist Should Know About in 2020!», *Analytics Vidhya*, 21 de septiembre de 2020. https://www.analyticsvidhya.com/blog/2020/09/different-nosql-databases-every-data-scientist-must-know/ (accedido 8 de febrero de 2022).

[38] R. Nabimanya, «Relational vs Non-Relational Databases». https://morioh.com/p/0f0f033ab8ae (accedido 8 de febrero de 2022).

[39] É. Mafra, «¿Qué es engagement? El término de marketing que debes conocer bien», *Rock Content - ES*, 1 de julio de 2020. https://rockcontent.com/es/blog/que-es-engagement/ (accedido 8 de febrero de 2022).

[40] «Crear una aplicación Flutter en AWS», *Amazon Web Services, Inc.* https://aws.amazon.com/es/getting-started/hands-on/build-flutter-app-amplify/ (accedido 18 de febrero de 2022).

[41] «Help:Sitelinks - Wikidata». https://www.wikidata.org/wiki/Help:Sitelinks (accedido 3 de junio de 2022).

[42] «Facts about Google and Competition», 4 de noviembre de 2011. https://web.archive.org/web/20111104131332/https://www.google.com/competition/howgooglesearchworks.html (accedido 3 de junio de 2022).

[43] «Wikidata:Request a query/Archive/2022/04 - Wikidata». https://www.wikidata.org/wiki/Wikidata:Request_a_query/Archive/2022/04#Query_a_random_set_of_popular_entries (accedido 3 de junio de 2022).

[44] «Wikidata:Request a query/Archive/2022/05 - Wikidata». https://www.wikidata.org/wiki/Wikidata:Request_a_query/Archive/2022/05#Filter_out_all_%22Wiki%22_entries (accedido 3 de junio de 2022).

[45] A. Larsson, «Getting Random Results in SPARQL», 17 de septiembre de 2020. https://byabbe.se/2020/09/17/getting-random-results-in-sparql (accedido 3 de junio de 2022).

[46] «Wikidata:List of properties - Wikidata». https://www.wikidata.org/wiki/Wikidata:List_of_properties (accedido 17 de abril de 2022).

[47] B. Inc, «Command Line Interface Setup», *Back4App™*. https://www.back4app.com/docs/local-development/parse-cli (accedido 4 de junio de 2022).