



Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**



Grado en Matemáticas e Informática

Trabajo Fin de Grado

Interfaz de Usuario para Acceso a Sistema de Pregunta-Respuesta

Autor: José Manuel Vega Gradi

Tutores: Óscar Corcho García, Carlos Badenes-Olmedo

Madrid, Junio 2022

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Matemáticas e Informática

Título: Interfaz de Usuario para Acceso a Sistema de Pregunta-Respuesta

Junio 2022

Autor: José Manuel Vega Gradi

Tutor: Óscar Corcho García, Carlos Badenes-Olmedo
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

La búsqueda de respuestas o *Question Answering* es una de las principales vertientes del Procesamiento de Lenguaje Natural, la cual se ha beneficiado enormemente de los recientes avances de la Inteligencia Artificial y la Web Semántica. Una de sus principales variantes, el *Knowledge Graph Question Answering* (KGQA), tiene como objetivo responder preguntas en lenguaje natural a partir de consultas a bases de conocimiento estructuradas. La accesibilidad a los sistemas de pregunta-respuesta en el área del KGQA, así como las maneras de evaluarlos objetivamente, están todavía en desarrollo, dificultando el crecimiento de este campo.

El objetivo de este proyecto será realizar un servicio que resuelva las cuestiones anteriormente planteadas. Para ello, hemos diseñado una aplicación en el lenguaje de programación Python que integra bajo un mismo servicio una interfaz web, una base de datos no relacional (MongoDB) para la gestión de conjuntos de datos de KGQA y un sistema de pregunta-respuesta con acceso API REST (mediante peticiones HTTP).

Como resultado de este trabajo se ha desarrollado un servicio desplegable con Docker que facilita el uso de sistemas de pregunta-respuesta de KGQA por medio de una interfaz visual. Este permite al usuario realizar preguntas libres, introduciéndolas por teclado, o seleccionar la pregunta de entre uno de los conjuntos de datos de preguntas y respuestas subido previamente a la interfaz web. Tras recibir la respuesta del sistema, el usuario puede validar si las respuestas obtenidas son correctas, permitiendo así evaluar el sistema y generar nuevos conjuntos de datos anotados automáticamente. Por otro lado, también será posible obtener una serie de métricas sobre la fluidez de escritura y los tipos de preguntas de los conjuntos de datos subidos.

Abstract

Recent advancements in Artificial Intelligence and Semantic Web have greatly benefited Question Answering, one of the major tasks of Natural Language Processing. Knowledge Graph Question Answering (KGQA) is one of the main subareas of this field, which aims to answer natural language questions by querying structured knowledge bases. Question-answering systems accessibility and objective evaluation in the KGQA area are still under development, hindering the growth of this field.

Our objective is to develop an application that solves the issues raised above. To do so, we have designed a Python service that integrates under a single system a web interface, a non-relational database (MongoDB) for KGQA dataset management and a question-answering system with REST API access (via HTTP requests).

As a result of this project, we have developed a Docker deployable service that facilitates the use of KGQA question-answering systems through a visual interface. This service allows the user to ask free-form questions by typing them, or to select the questions from one of the datasets uploaded onto the interface. After receiving the system's answers, the user can validate if the answers obtained are correct, thus allowing to evaluate the system and generate new annotated datasets automatically. Furthermore, it is also possible to obtain a series of metrics on written fluency and question types by answer on the uploaded datasets.

Tabla de contenidos

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	5
1.3. Estructura de la Memoria	6
2. Diseño del Sistema	7
2.1. Base de Datos	7
2.2. Servicio Web	10
2.3. Sistema de Pregunta-Respuesta	11
2.4. Estructura del Repositorio	13
3. Desarrollo del Sistema	15
3.1. Archivo principal	15
3.2. Módulo de Pregunta-Respuesta	16
3.3. Módulo de Gestión de Conjuntos de Datos	22
3.4. Módulo de Generación de Informes	23
3.5. Herramientas auxiliares	28
3.5.1. Multipage	28
3.5.2. Gestor de MongoDB	29
3.5.3. Gestor de Google Sheets	32
3.5.4. Preprocesamiento de Conjuntos de Datos	33
3.5.5. Clasificador de preguntas	34
3.6. Despliegue	38
3.6.1. Prerrequisitos	38
3.6.2. Registro de una Cuenta de Servicio de Google Cloud	38
3.6.3. Creación de la imagen del Servicio Web	42
3.6.4. Definición de las variables de entorno	43
3.6.5. Puesta en marcha del contenedor	44
4. Guía de Uso	45
4.1. Hacer una pregunta y validar el Sistema de Pregunta-Respuesta	45
4.2. Subir un conjunto de datos	48
4.3. Generar un informe sobre un conjunto de datos	48
5. Resultados y Conclusiones	53

5.1. Resultados	53
5.2. Líneas Futuras de Trabajo	53
5.3. Conclusiones	54
6. Análisis de impacto	55
Bibliografía	57

Capítulo 1

Introducción

1.1. Motivación

En los últimos años el campo de la Inteligencia Artificial (IA) está experimentando un desarrollo exponencial, por medio de avances como la creación de modelos más grandes, conjuntos de datos mayores y de mejor calidad o nuevas técnicas como la atención y los transformadores. Tareas de campos como el Procesamiento de Lenguaje Natural (PLN) o el Reconocimiento y Generación de Imágenes que hace pocos años parecían extremadamente arduas hoy están a nuestro alcance. Gracias a modelos como GPT-3 ¹ o DALL-E 2 ², cuyas capacidades han conseguido incluso llamar la atención de la esfera popular, podemos esperar una mayor adopción de la IA en los años venideros; lo cual probablemente incentive un desarrollo aún más rápido. También hay múltiples iniciativas para hacer llegar la IA a un público mayor, siendo la más destacada la página web Hugging Face ³ cuya empresa recaudó 100 millones de dólares en su última financiación de mayo de 2022 para alcanzar el estatus de unicornio (empresa emergente sin presencia en Bolsa con un valor que supera los 1.000 millones). [1] Hugging Face es un repositorio especializado en PLN que provee de forma abierta modelos de IA pre-entrenados para diversas tareas, con el objetivo de “Avanzar y Democratizar la Inteligencia Artificial”

Dentro del PLN uno de los campos que más se ha beneficiado de los avances de la IA es la Búsqueda de Respuestas o *Question Answering* (QA), una disciplina que busca construir sistemas que respondan automáticamente a respuestas formuladas en lenguaje natural. Para este ámbito son cruciales los Modelos de Pregunta-Respuesta, modelos de aprendizaje automático o aprendizaje profundo que responden preguntas a partir de un texto que sirve como contexto o sin contexto alguno (pregunta-respuesta de dominio abierto). [2] HuggingFace es una de las páginas que más ha ayudado a visibilizar el campo de QA, alojando entornos de trabajo como HayStack ⁴ o la librería

¹OpenAI API

²DALL-E 2

³Hugging Face - The AI community building the future

⁴Haystack

de Python Transformers ⁵ que abren el uso de los Modelos de Pregunta-Respuesta al público general, permitiendo construir sistemas de pregunta-respuesta sobre textos de manera sencilla. Precisamente en Hugging Face se pone de manifiesto el pujante desarrollo experimentado por estos modelos en los últimos años: A mayo de 2022 podemos encontrar al menos 6 modelos con más de 100.000 descargas totales, con el más empleado (roberta-base for QA) ⁶ superando el millón de descargas. [3]

Una de las variantes del QA que más tracción está generando en los últimos tiempos es el *Knowledge Graph Question Answering* (KGQA), debido a la relevancia que ha cobrado la Web Semántica en la última década. La semántica es la parte de la lingüística que estudia la forma de las estructuras léxicas y los procesos mentales a través de los cuales los seres humanos damos sentido a las expresiones lingüísticas. La Web Semántica o Web 3.0 tiene como objetivo crear un tipo de web que, haciendo uso de la semántica, facilite la búsqueda de información haciéndola más intuitiva para las personas. Así, podemos definir la Web Semántica como el conjunto de actividades desarrolladas en el seno del entorno de la *World Wide Web* (WWW) para crear tecnologías de publicación de datos que sean más fácilmente legibles para las aplicaciones informáticas. [4] Esta contiene una vasta cantidad de información organizada en bases de conocimiento: Bases de datos para la gestión del conocimiento que proveen medios para la recolección, organización y recuperación de conocimiento.

Las bases de conocimiento de la Web Semántica se presentan de múltiples formas, de manera que se puede distinguir entre bases de conocimiento estructuradas y no estructuradas. Entre las bases de conocimiento estructuradas, una de las formas más habituales son los grafos de conocimiento. Los grafos de conocimiento estructuran los datos en forma de grafo para enlazar entidades del mundo real entre sí mediante propiedades semánticas que definen relaciones y clases. [5] Estos empezaron a tomar popularidad a partir de la introducción del Grafo de Conocimiento de Google, el cual se introdujo en 2012 para mejorar los resultados obtenidos por su motor de búsqueda [6] y fue imitado posteriormente por otras compañías como Yahoo o Bing. También hay grandes iniciativas como DBPedia, ⁷ un repositorio de datos extraído de Wikipedia que estructura relaciones entre ellos o WikiData, ⁸ una base de conocimiento colaborativa, libre y abierta que almacena información estructurada; haciendo uso ambas del marco *Resource Description Framework* (RDF) de Web3.

En este contexto surge el campo del KGQA, cuyo objetivo consiste en responder de manera automática a preguntas en lenguaje natural a partir de información extraída de grafos de conocimiento como los ya mencionados DBPedia y Wikidata. Para esto, los sistemas de pregunta-respuesta combinan técnicas del Procesamiento de Lenguaje natural, la Búsqueda y Recuperación de Información, el Aprendizaje Automático y la Web Semántica. [7] [8]

Ya hemos comentado cómo el enorme desarrollo experimentado por los Modelos de

⁵Transformers - HuggingFace

⁶deepset/roberta-base-squad2 - Hugging Face

⁷DBPedia en español

⁸Wikidata

Introducción

Lenguaje y la web semántica en el pasado reciente ha hecho mucho más accesible la realización de sistemas de pregunta-respuesta. No obstante, es igual de importante para el futuro crecimiento de este campo la accesibilidad a dichos sistemas desde el punto de vista del usuario. Tecnologías como Flask ⁹ o FastAPI ¹⁰ han hecho que sea más fácil definir API REST en Python de manera rápida y eficiente, motivo por el cual muchos sistemas se han creado siguiendo un formato API RESTful. Sin embargo, esto dificulta el acceso a dichos sistemas de pregunta-respuesta para la mayor parte de los usuarios. Al margen del proceso de despliegue del sistema, la tarea de realizar peticiones HTTP y de recibir la respuesta se escapa de las capacidades de muchos de estos usuarios. Para hacer estos sistemas de QA accesibles al mayor número posible de usuarios, sería necesario que contasen con una interfaz gráfica que facilite la realización de preguntas en lenguaje natural.

Haciendo una revisión sobre el campo de KGQA también se puede observar una gran dificultad para evaluar los sistemas de pregunta-respuesta de manera objetiva. En primer lugar, la cantidad de conjuntos de datos para esta tarea es limitada. Los tres principales *datasets* existentes son VQuAnDa ¹¹, VANiLLA ¹² y LC-QuAD 2.0 ¹³; los cuales presentan preguntas junto a su respuesta y la SPARQL *query* necesaria para obtenerla consultando Wikidata o DBPedia. Para aquellos sistemas que no construyen la consulta SPARQL a partir de la pregunta en lenguaje natural, la manera de evaluar la respuesta será comparar la respuesta en lenguaje natural obtenida por el sistema con la dada por el conjunto de datos. Para ello, habitualmente se emplean medidas que miden la calidad de sistemas de traducción automática como estas:

- **Exact Match (EM):** Métrica que evalúa las respuestas según si son estrictamente iguales a la respuesta modelo, en cuyo caso $EM = 1$. Esta métrica es de "todo o nada": bastará que un único carácter de la respuesta difiera del modelo para que sea 0. [12]
- **Bilingual Evaluation Understudy (BLEU) :** Una de las tareas más comunes del PLN es la tokenización, que consiste en separar un trozo de texto en unidades más pequeñas o tokens. La métrica BLEU, cuyo rango va de 0 a 1, cuenta los n-gramas comunes entre el texto de referencia y el texto dado, tomando como 1-gramas cada token. En nuestro caso, cada palabra será un token. [13]
- **Metric for Evaluation of Translation with Explicit ORdering (METEOR) :** Los problemas de clasificación binarios consisten en clasificar elementos de un conjunto en dos grupos en base a una regla de clasificación, por ejemplo como positivos o negativos. En estos, la precisión o *precision* intenta identificar cuántas predicciones de positivo fueron correctas, mientras que la exhaustividad o *recall* nos da la proporción de positivos correctos (Figura 1.1).

$$Precision = \frac{Verdaderos \ Positivos}{Verdaderos \ Positivos + Falsos \ Positivos}$$

⁹Flask

¹⁰FastAPI

¹¹GitHub - VQuAnDa - Verbalization Question Answering Dataset [9]

¹²GitHub - VANiLLa : Verbalized Answers in Natural Language at Large scale [10]

¹³GitHub - LC-QuAD2.0 [11]

$$Exhaustividad = \frac{Verdaderos \ Positivos}{Verdaderos \ Positivos + Falsos \ Negativos}$$

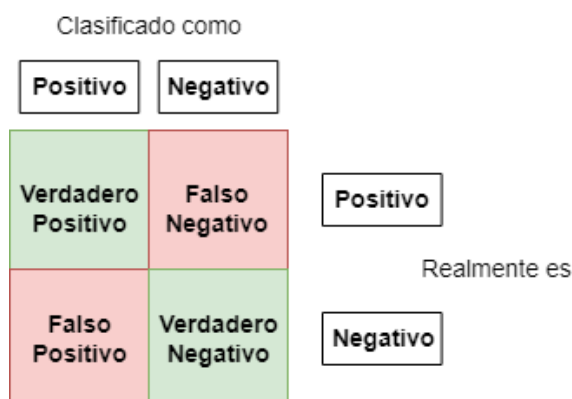


Figura 1.1: Diagrama explicativo sobre Positivos y Negativos

METEOR es una media basada en la media armónica de la precisión y la exhaustividad (cuyo rango es de 0 a 100), donde se tomará como positivo la coincidencia por unigramas entre el texto y el texto de referencia. Esta métrica incluye funciones como las comparativas por *stemming* (reducción de las palabras a su raíz) o por sinonimia para acercarse más al juicio humano. [14]

- **Distancia de Levenshtein:** Esta distancia entre dos cadenas de caracteres se corresponde con el número mínimo de operaciones necesarias para transformar una cadena en otra. [15]
- **Similitud Coseno:** Los *embeddings* son un conjunto de técnicas de PLN en donde las palabras o frases del lenguaje natural son representadas como vectores. En estas representaciones se plasman las relaciones sintácticas y semánticas con otros términos. La similitud coseno es una medida de la similitud entre dos vectores, y será más cercana a 1 cuan más similares sean ambos vectores.

En base a estas métricas, haremos ahora una comparativa en dos tablas. Supongamos que queremos evaluar las respuestas de varios sistemas de pregunta-respuesta a la pregunta “ ¿Dónde nació Fernando Alonso? ”, tomando como respuesta modelo “Oviedo” y “Oviedo, Asturias, España” respectivamente en las Tablas 1.1 y 1.2:

Cuadro 1.1: Medidas de distintas respuestas con respuesta modelo “Oviedo”

Respuesta	EM	BLEU	METEOR	Dist. Lev.	Sim. Cos.
Oviedo	1	1	100	0	1
Oviedo, Asturias, Spain	0	0.2028	0.0	17	0.7226
Asturias	0	0.0	0.0	8	0.77975
Spain	0	0.0	0.0	6	0.7202
London	0	0.0	0.0	5	0.5436
Cat	0	0.0	0.0	6	0.3396

Introducción

Cuadro 1.2: Medidas con respuesta modelo “Oviedo, Asturias, Spain”

Respuesta	EM	BLEU	METEOR	Dist. Lev	Sim. Cos.
Oviedo	0	0.0588	0.0	17	0.7226
Oviedo, Asturias, Spain	1	1	100	0	1
Asturias	0	0.1533	0.0	15	0.8199
Spain	0	0.0273	0.0	18	0.8006
London	0	0.0	0.0	20	0.4184
Cat	0	0.0	0.0	22	0.225

Podemos ver cómo en función de las métricas empleadas y de la respuesta modelo usada, los resultados diferirán profundamente e incluso respuestas que no son incorrectas necesariamente (como que Fernando Alonso nació en Asturias o España) son penalizadas como una respuesta inadecuada. Así pues, la única manera viable de señalar si una respuesta dada es correcta o no sería mediante la revisión de un humano, pero esto es también inviable actualmente dado lo inaccesibles que resultan al público general los sistemas de pregunta-respuesta.

En definitiva, recapitulando todo lo anteriormente expuesto:

- La búsqueda de respuestas o QA es una tarea central del PLN que se ha beneficiado enormemente de los recientes avances de la IA.
- El campo del KGQA, dentro del QA, tiene como objetivo responder preguntas en lenguaje natural a partir de consultas a bases de conocimiento estructuradas (grafos de conocimiento).
- La accesibilidad a los sistemas de pregunta-respuesta en el área del KGQA, tanto de manera técnica como de cara al usuario, es una tarea crucial para el desarrollo de este campo; y la cual se sigue desarrollando en la actualidad.
- La evaluación de los sistemas de pregunta-respuesta es una tarea complicada. Esto es debido a la falta de métricas cercanas al juicio humano, lo cual dificulta enormemente la validación y mejora estos sistemas.

1.2. Objetivos

En este proyecto, buscamos aportar una solución al problema de accesibilidad a los sistemas de pregunta-respuesta que hemos comentado en el anterior capítulo. Nuestro objetivo será la creación una interfaz web en Python que facilite el uso de los sistemas de pregunta-respuesta. También se buscará satisfacer las necesidades del campo del KGQA mencionadas en el capítulo anterior. A alto nivel, construiremos un sistema que ofrecerá las siguientes funcionalidades:

1. Acceso a sistemas de pregunta-respuesta en formato API Rest con una interfaz web visual.

2. Consulta de grafos de conocimiento por medio de preguntas en lenguaje natural.
3. Validación de sistemas de QA ya existentes mediante la valoración del usuario.
4. Creación de nuevos conjuntos de datos de KGQA.
5. Evaluación de conjuntos de datos ya existentes.

1.3. Estructura de la Memoria

Los capítulos que restan tienen el siguiente contenido:

- **Capítulo 2: Diseño del Sistema.** Explicación de la estructura del sistema.
- **Capítulo 3: Desarrollo del Sistema.** Documentación del proceso de desarrollo del proyecto. Esto incluye tanto la realización del servicio como su despliegue e integración continua.
- **Capítulo 4: Uso de la Interfaz Web.** Demostración de uso del sistema.
- **Capítulo 5: Resultados y Conclusiones. Líneas Futuras de Trabajo.** Reflexión basada en los resultados obtenidos del TFG y posibles ampliaciones del proyecto.
- **Capítulo 6: Análisis de impacto.** Alcance de los resultados derivados de la realización del TFG, en múltiples ámbitos.
- **Bibliografía**

Capítulo 2

Diseño del Sistema

Para este proyecto queremos crear una interfaz que permita acceder a sistemas de pregunta-respuesta en formato API Rest y que ofrezca funcionalidades como la evaluación de conjuntos de datos y sistemas de QA ya existentes o la creación de nuevos conjuntos de datos. Así pues, el diseño de la arquitectura del sistema ha sido pensado con la intención de integrar tanto la gestión de conjuntos de datos como la conexión a un sistema de pregunta-respuesta de manera sencilla en el propio sistema. De esta manera, el sistema está compuesto por una interfaz web, una base de datos y un sistema de pregunta-respuesta con acceso API REST tal y como figura en la Ilustración 2.1. La base de datos será accedida por el servicio web tanto para introducir nuevos conjuntos de datos como para obtener preguntas de dichos conjuntos.

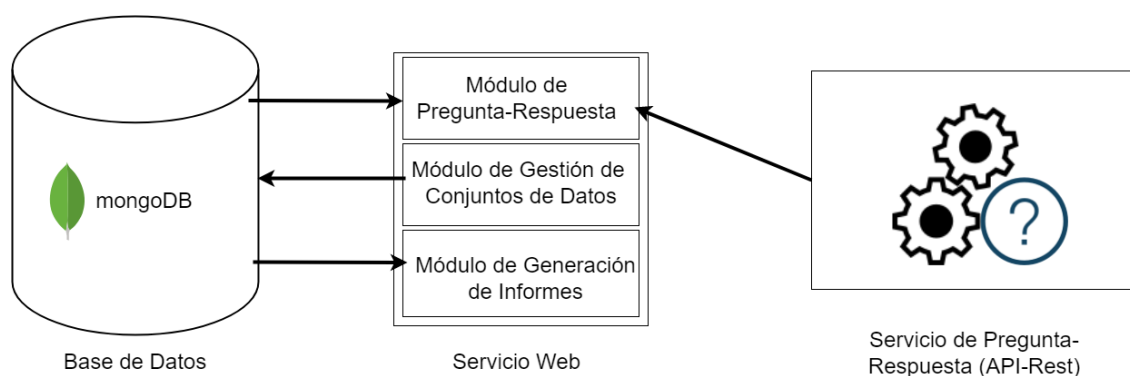


Figura 2.1: Diagrama de la arquitectura del Sistema.

2.1. Base de Datos

La base de datos del servicio almacenará los conjuntos de datos de preguntas y respuestas subidos por los usuarios. Las decisiones del sistema relacionadas con ella se han hecho teniendo en cuenta los conjuntos de datos pertenecientes al estado del arte, como los ya mencionados VQuAnDA, LC-QuAD o VANiLLA. Estos habitualmente se presentan como documentos en formato JSON, distinguiéndose dos tipos. Tendremos

por un lado archivos en formato JSON Array (archivos JSON por defecto en formato de lista) y por otro en formato JSON Line (donde cada línea del conjunto de datos será un objeto JSON). A continuación tenemos un ejemplo de varias entradas de dichos *datasets*:

Listing 2.1: Extracto del conjunto de datos VQuAnDA (formato JSON Array).

```

1  [
2    {
3      "uid": "3986",
4      "question": "Count the number of people became famous for when Andrew Jackson
                    was a commander ?",
5      "verbalized_answer": "There are [8] people known for works commanded by Andrew
                    Jackson.",
6      "query": "SELECT DISTINCT COUNT(?uri) WHERE { ?x <http://dbpedia.org/ontology/
                    commander> <http://dbpedia.org/resource/Andrew_Jackson> . ?uri <http://
                    dbpedia.org/ontology/knownFor> ?x . }"
7    },
8    {
9      "uid": "2262",
10     "question": "Which location city of Denver Broncos is the palce of birth of
                    Steven Clark Cunningham ?",
11     "verbalized_answer": "The location city of Denver Broncos and place of birth of
                    Steven Clark Cunningham is [Colorado, Denver].",
12     "query": "SELECT DISTINCT ?uri WHERE { <http://dbpedia.org/resource/
                    Denver_Broncos> <http://dbpedia.org/ontology/locationCity> ?uri. <http://
                    dbpedia.org/resource/Steven_Clark_Cunningham> <http://dbpedia.org/ontology/
                    birthPlace> ?uri}"
13   },
14   ...
15 ]

```

Listing 2.2: Extracto del conjunto de datos LC-QuAD 2.0 (formato JSON Array).

```

1  [
2    {
3      "NNQT_question": "What is the {country} for {head of state} of {Mahmoud Abbas}"
4      ,
5      "uid": 20258,
6      "subgraph": "simple question left",
7      "template_index": 604,
8      "question": "Who is the {country} for {head of state} of {Mahmoud Abbas}",
9      "sparql_wikidata": " select distinct ?subj where { ?subj wdt:P35 wd:Q127998 . ?
                    sbj wdt:P31 wd:Q6256 } ",
10     "sparql_dbpedia18": "select distinct ?subj where { ?statement <http://www.w3.
                    org/1999/02/22-rdf-syntax-ns#subject> ?subj . ?statement <http://www.w3.org
                    /1999/02/22-rdf-syntax-ns#predicate> <http://www.wikidata.org/entity/P35> .
                    ?statement <http://www.w3.org/1999/02/22-rdf-syntax-ns#object> <http://
                    wikidata.dbpedia.org/resource/Q127998> . ?subj <http://www.wikidata.org/
                    entity/P31> <http://wikidata.dbpedia.org/resource/Q6256> } ",
11     "template": "<?S P 0 ; ?S InstanceOf Type>",
12     "answer": [],
13     "template_id": 2,

```



```

14     "paraphrased_question": "What country is Mahmoud Abbas the head of state of?"
15   },
16   {
17     "NNQT_question": "What is {population} of {Somalia} that is {point in time} is
18       {2009-0-0} ?",
19     "uid": 7141,
20     "subgraph": "statement_property",
21     "template_index": 3586,
22     "question": "What was the population of Somalia in 2009-0-0?",
23     "sparql_wikidata": "SELECT ?obj WHERE { wd:Q1045 p:P1082 ?s . ?s ps:P1082 ?obj
24       . ?s pq:P585 ?x filter(contains(YEAR(?x),'2009')) }",
25     "sparql_dbpedia18": "select distinct ?obj where {\n?statement <http://www.w3.
26       org/1999/02/22-rdf-syntax-ns#subject> <http://wikidata.dbpedia.org/resource
27       /Q1045> .\n?statement <http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate
28       > <http://www.wikidata.org/entity/P1082> .\n?statement <http://www.w3.org/1
29       999/02/22-rdf-syntax-ns#object> ?obj .\n?statement <http://www.wikidata.org
30       /entity/P585> <2009-0-0>\n} \n",
31     "template": "(E pred F) prop ?value",
32     "answer": [],
33     "template_id": "statement_property_2",
34     "paraphrased_question": "As of 2009, how many people lived in Somalia?"
35   },
36   ...
37 ]

```

Listing 2.3: Extracto del conjunto de datos VANiLLA (formato JSON Line).

```

1 {"answer":"male","answer_sentence":"sterjo is a male.,"question":"Which sex does
2   Claude Nicolas Emmery possess ?","question_entity_label":"Claude Nicolas Emmery","
3   question_id":58504,"question_relation":"P21"}
4 {"answer":"Lake Ann","answer_sentence":"lake ann is a place in benzie county,
5   california.,"question":"whats the name of a lake in benzie county, michigan","
6   question_entity_label":"Benzie County","question_id":87393,"question_relation":"R13
7   1"}
8 {"answer":"Spain","answer_sentence":"his country of citizenship is spain.,"question":
9   "Which administrative territory is the country of citizenship of Jon Hern\u00e9ndez
10  ?","question_entity_label":"Jon Hern\u00e9ndez","question_id":36854,"
11  question_relation":"P27"}
12 ...
13 ]

```

Una base de datos relacional es un tipo de base de datos que sigue el modelo relacional. Es decir, está compuesta por varias tablas relacionadas entre sí, tablas que a su vez son conjuntos de atributos (columnas) y registros (filas). En las bases de datos relacionales se utiliza el lenguaje de consulta estructurado o SQL (*Structured Query Language*) para administrar y recuperar datos. Por otro lado, una base de datos no relacional, también llamada *NoSQL*, almacena información en un formato específico que se ajusta a unas necesidades determinadas. Por ejemplo, MongoDB¹ es una base de datos no relacional y de código abierto en el que los datos se presentan en forma de documentos en formato BSON, un formato de almacenamiento de datos ligero cu-

¹MongoDB

yo nombre proviene de JSON Binario. Un objeto BSON consiste en una lista ordenada de elementos donde cada elemento está constituido por un campo nombre (que será de tipo cadena de caracteres), un tipo y un valor; al igual que los objetos JSON.

Se ha elegido MongoDB como base de datos del sistema frente a otras opciones, como podría ser una base de datos relacional implementada en MySQL o PostgreSQL al estimarse que las características de MongoDB se ajustaban mejor a las necesidades del proyecto. Se consideró que una base de datos no relacional otorga mayor flexibilidad, lo cual es de vital importancia al recibir conjuntos de datos que provienen de distintas fuentes. Otro motivo para la elección es que, como acabamos de exponer, los conjuntos de datos pertenecientes al estado del arte se presentan como documentos en formato JSON, y los archivos BSON de MongoDB están diseñados como una representación de JSON optimizada para el almacenamiento y la extracción de datos. [16]

MongoDB está organizado en colecciones, conjuntos de registros análogos a las tablas de una base de datos [17]. En la base de datos cada colección se corresponderá con un conjunto de datos, y cada registro tendrá al menos los siguientes campos:

- **_id:** Campo indexado automáticamente y de tipo ObjectID por defecto (cadena de caracteres en hexadecimal) que sirve como clave principal de los elementos de una colección.
- **question:** Pregunta.
- **answer:** Respuesta a la pregunta.

2.2. Servicio Web

Este Sistema se ha desarrollado en el lenguaje de programación Python. Para el apartado visual de la interfaz web se ha escogido Streamlit,² un marco de trabajo en código abierto utilizado habitualmente en los campos del Aprendizaje Automático, el Procesamiento de Lenguaje Natural y la Ciencia de Datos. Esta librería de Python permite construir aplicaciones web en pocas líneas de código, ahorrándole así al desarrollador tareas tales como programar un *back-end*, definir enrutamientos o utilizar otros lenguajes como Angular, HTML o CSS. Esta decisión se tomó dado que Python es uno de los lenguajes más comunes para el desarrollo de Sistemas de Pregunta y Respuesta. Además, usar Streamlit permitiría aliviar la carga de trabajo que constituye el desarrollo del apartado visual de la página web, pudiendo así centrarnos en las funcionalidades del Sistema.

El servicio permite al usuario realizar preguntas libres, introduciéndolas por teclado, o seleccionar la pregunta de entre uno de los conjuntos de datos de preguntas y respuestas subido previamente a la interfaz. Tras recibir la respuesta del Sistema de Pregunta-Respuesta, el usuario puede validar si las respuestas obtenidas son correctas, permitiendo así evaluar el Sistema y generar nuevos conjuntos de datos anotados

²Streamlit [18]

automáticamente. El usuario puede también obtener una serie de métricas sobre los conjuntos de datos subidos y generar un reporte en PDF. Para la implementación de estas funcionalidades se ha dividido la aplicación en tres módulos accesibles desde la barra de navegación lateral de la página:

- **Módulo de Pregunta-Respuesta:** Desde esta página se pueden realizar preguntas libres o de un conjunto de datos. Al usuario se le ofrecerán las respuestas obtenidas por el Sistema junto a los textos a partir de los cuales se ha creado la respuesta, y este podrá marcarlas como correctas o incorrectas.
- **Módulo de Gestión de Conjuntos de Datos:** Aquí el usuario podrá subir sus conjuntos de datos en formato JSON o CSV.
- **Módulo de Generación de Informes:** En esta página el usuario podrá seleccionar uno de los conjuntos de datos ya subidos y generar un reporte que ofrecerá métricas sobre los tipos de pregunta del conjunto y la fluidez de la escritura de estas. Esta página es exportable a PDF.

2.3. Sistema de Pregunta-Respuesta

El sistema admite el despliegue de un servicio de Pregunta-Respuesta. Este servicio ha de funcionar como una API REST a la cual se le realizarán peticiones *GET HTTP* con la pregunta del usuario, y este devolverá una respuesta en formato JSON. La dirección de dicha API se pasará al Servicio como variable de entorno, así como un booleano que indicará el formato de la respuesta JSON, la cual puede ser de dos tipos:

- De una única respuesta por Objeto JSON.

Listing 2.4: Ejemplo de JSON con una sola respuesta.

```
1 {
2   "answer": "Oviedo, Asturias, Spain",
3   "source": dbpedia,
4   "confidence": 0.801,
5   "evidence": {
6     "end": 149,
7     "summary": " The car number of Fernando Alonso is 14. The Last win of
      Fernando Alonso is 2013. The birth place of Fernando Alonso is Oviedo,
      Asturias, Spain. The name of Fernando Alonso is Fernando Alonso. The
      First win of Fernando Alonso is 2003. The last season of Fernando Alonso
      is 2018. The birth name of Fernando Alonso is Fernando Alonso D\u00edez
      . The caption of Fernando Alonso is Alonso in 2016. The First race of
      Fernando Alonso is 2001. The image size of Fernando Alonso is 240. The
      last win of Fernando Alonso is 2013 Spanish Grand Prix. The nationality
      of Fernando Alonso is Spanish. The title of Fernando Alonso is Fernando
      Alonso achievements, Fernando Alonso teams and series. The first race
      of Fernando Alonso is 2001 Australian Grand Prix. The 2021 Team of
      Fernando Alonso is Alpine F1, Renault in Formula One. The source of
      Fernando Alonso is Alonso's race engineer at Ferrari, Andrea Stella, on
      Alonso's ability and similarities to Michael Schumacher. The first wi of
      Fernando Alonso is 2003 Hungarian Grand Prix. .
8   },
```

2.3. Sistema de Pregunta-Respuesta

```
9     "start": 126
10   },
11   "question": "where was Fernando Alonso born?"
12 }
```

- De múltiples respuestas por cada Objeto JSON.

Listing 2.5: Ejemplo de JSON con múltiples respuestas.

```
1 {
2   "question": "where was Fernando Alonso born?"
3   "answers": [
4     {
5       "answer": "Oviedo",
6       "source": wikidata,
7       "confidence": 0.9754,
8       "evidence": {
9         "end": 523,
10        "start": 517,
11        "summary": " The Encyclopedia Universalis ID of Fernando Alonso is fernando
-alonso. The BAnQ author ID of Fernando Alonso is ncf10786137. The
Treccani ID of Fernando Alonso is fernando-alonso. The sibling of
Fernando Alonso is Lorena Alonso. The Quora topic ID of Fernando
Alonso is Fernando-Alonso. The AS. com athlete ID of Fernando Alonso
is fernando_alonso/24337. The image of Fernando Alonso is http://
commons.wikimedia.org/wiki/Special:FilePath/Alonso%202016.jpg. The
place of birth of Fernando Alonso is Oviedo. The sex or gender of
Fernando Alonso is male. The father of Fernando Alonso is Jose Luis
Alonso. The spouse of Fernando Alonso is Raquel del Rosario. The
country of citizenship of Fernando Alonso is Spain. The instance of of
Fernando Alonso is human. The position held of Fernando Alonso is
UNICEF Goodwill Ambassador. The member of sports team of Fernando
Alonso is Minardi, Scuderia Ferrari, McLaren, Renault F1 Team, Alpine
F1 Team. The native language of Fernando Alonso is Spanish"
12     }
13   },
14   {
15     "answer": "Oviedo, Asturias, Spain",
16     "source": dbpedia,
17     "confidence": 0.801,
18     "evidence": {
19       "end": 149,
20       "summary": " The car number of Fernando Alonso is 14. The Last win of
Fernando Alonso is 2013. The birth place of Fernando Alonso is Oviedo,
Asturias, Spain. The name of Fernando Alonso is Fernando Alonso.
The First win of Fernando Alonso is 2003. The last season of
Fernando Alonso is 2018. The birth name of Fernando Alonso is
Fernando Alonso D\u00edez. The caption of Fernando Alonso is Alonso
in 2016. The First race of Fernando Alonso is 2001. The image size
of Fernando Alonso is 240. The last win of Fernando Alonso is 2013
Spanish Grand Prix. The nationality of Fernando Alonso is Spanish.
The title of Fernando Alonso is Fernando Alonso achievements,
Fernando Alonso teams and series. The first race of Fernando Alonso
is 2001 Australian Grand Prix. The 2021 Team of Fernando Alonso is
Alpine F1, Renault in Formula One. The source of Fernando Alonso is
```

```
Alonso's race engineer at Ferrari, Andrea Stella, on Alonso's ability
and similarities to Michael Schumacher. The first win of Fernando
Alonso is 2003 Hungarian Grand Prix.
21     ",
22     "start": 126
23     }
24 },
25 ...
26 ]
27 }
```

2.4. Estructura del Repositorio

El control de versiones del proyecto se ha realizado por medio de GitHub³. El código del Sistema se encuentra en el siguiente repositorio:

<https://github.com/Drugs4CoVid/QA-Webapp>

El proyecto sigue la siguiente estructura:

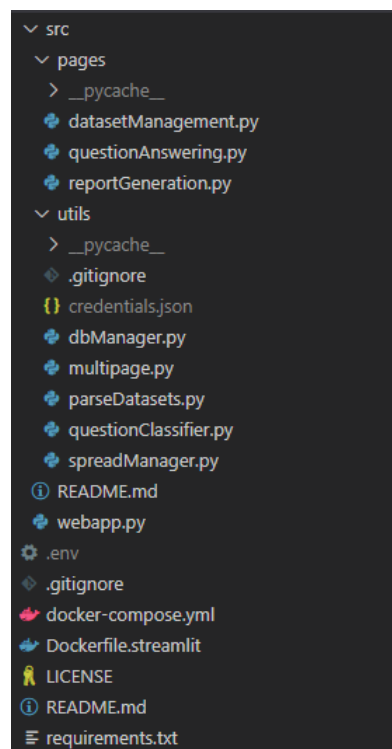


Figura 2.2: Estructura del Repositorio

Como podemos ver en la Figura 2.4 el proyecto sigue la siguiente estructura:

- **Carpeta docs:** Carpeta que contiene las imágenes del archivo *README* en la

³GitHub

subcarpeta *imgs* y la memoria de este Trabajo de Fin de Grado como documentación para el usuario.

- **Carpeta src:** Carpeta que contiene el código de la interfaz web. En esta carpeta tendremos el archivo principal (*webapp.py*) junto al código de los 3 módulos de nuestros sistemas. Además, tenemos una subcarpeta *utils*: donde se encuentra el código de las funciones auxiliares.
- **.gitignore:** Archivo de texto que le dice a Git qué archivos o carpetas ignorar en un proyecto, es decir, qué archivos no debe subir al repositorio.
- **docker-compose.yml:** Archivo para definir aplicaciones con múltiples contenedores en Docker.
- **Dockerfile.streamlit:** Archivo que define la imagen de Docker de la interfaz.
- **LICENSE:** Archivo que contiene el tipo de licencia del repositorio.
- **README.md:** Archivo en lenguaje Markdown con información sobre el repositorio y una guía de uso.
- **requirements.txt:** Archivo de texto con las librerías de Python requeridas.

Capítulo 3

Desarrollo del Sistema

La aplicación de Streamlit que monta el servicio web está contenida en el archivo *webapp.py*. Para correrlo, usaremos el siguiente comando:

```
$ streamlit run webapp.py
```

3.1. Archivo principal

Primero, importamos la librería Streamlit, los gestores de múltiples páginas (*MultiPage*) y MongoDB, y los scripts correspondientes a las páginas de la interfaz (que se encuentran en la carpeta *page*). Inicializamos la clase *MultiPage* y la base de datos.

Listing 3.1: Inicialización de librerías y objetos del archivo principal *webapp.py*

```
1 import streamlit as st
2 from utils.multipage import MultiPage
3 from utils import dbManager
4 from pages import questionAnswering, datasetManagement, reportGeneration
5
6 #Initialize Multipage and Database.
7 app = MultiPage()
8 db = dbManager.DbManager("mongodb:27017")
```



Figura 3.1: Título y cuerpo de la página web. Común a los tres módulos.

Tras esto, definimos los atributos de la página web con el método *set_page_config*. Establecemos el título y el icono que se mostrarán en la pestaña, que la disposición sea

3.2. Módulo de Pregunta-Respuesta

centrada y que la barra lateral de navegación esté desplegada por defecto. Escribimos el título y el cuerpo (en formato Markdown) de la página web en la interfaz.

Listing 3.2: Código que escribe el cuerpo y el título comunes a toda página.

```
1 #Set Page attributes
2 st.set_page_config(
3     page_title = "QA UI",
4     page_icon = ":book:",
5     layout = "centered",
6     initial_sidebar_state = "auto",
7 )
8
9 #Set page title and body
10 st.title("Web Interface for Question-Answering and Dataset Validation")
11
12 st.markdown("""
13     Streamlit Web Interface. \n
14     It allows users to make questions onto this Service, giving input on its performance, and
15     Upload their own Question-Answering Datasets.
16 """, unsafe_allow_html=True)
```

Finalmente, añadimos las aplicaciones de los 3 módulos de nuestra página web a MultiPage, y corremos la página seleccionada con la conexión al MongoDB como argumento. Por defecto, la aplicación comenzará con el Módulo de Pregunta-Respuesta.

Listing 3.3: Código que agrega los 3 módulos a la interfaz y la pone en ejecución.

```
1 #Add pages to the MultiPage.
2 app.addPage("Question-Answering",questionAnswering.app)
3 app.addPage("Upload Dataset",datasetManagement.app)
4 app.addPage("Report Generation",reportGeneration.app)
5
6 #Run the MultiPage (executes the selected page).
7 app.run(db)
```

3.2. Módulo de Pregunta-Respuesta

Esta será la página inicial de nuestra aplicación, desde la cual el usuario podrá realizar preguntas, ya sean libres (introduciéndolas por teclado) o seleccionadas de entre uno de los conjuntos de datos de preguntas y respuestas previamente subidos; y valorar la respuesta obtenida. Con esto, buscamos hacer accesibles los Sistemas de Pregunta-Respuesta en formato API REST a cualquier usuario.

Listing 3.4: Imports y código para la lectura de las variables de entorno.

```
1 import os
2 import pytz
3 import requests
4 import operator
5 import streamlit as st
6 from utils import spreadManager
7 from datetime import datetime
8 from annotated_text import annotated_text
9
```


Desarrollo del Sistema

```
10 #Read environment variables and setup spreadsheet timezone
11 EQA_SERVICE_DIRECTION = os.getenv("EQA_SERVICE_DIRECTION")
12 EQA_SERVICE_ROUTINGS = os.getenv("EQA_SERVICE_ROUTINGS","").split(",")
13
14 WORKSHEET = os.getenv("WORKSHEET")
15 WORKSHEET_ID = os.getenv("WORKSHEET_ID")
16 SPREADSHEET = os.getenv("SPREADSHEET")
17 SPREAD_TIMEZONE = pytz.timezone("Europe/Madrid")
18
19 DEFAULT_NUMBER_OF_ANSWERS = int(os.getenv("DEFAULT_NUMBER_OF_ANSWERS"))
20 MULTIPLE_ANSWERS_JSON = bool(os.getenv("MULTIPLE_ANSWERS"))
```

Importamos Streamlit, la librería *requests* para hacer solicitudes GET HTTP, el gestor de Google Sheets para introducir la valoración del usuario sobre las preguntas en la hoja de cálculo, los paquetes *datetime* y *pytz* para poder escribir la fecha actual en un huso horario determinado y la extensión de Streamlit *annotated_text* para anotar el contexto de la respuesta. Después, leemos las variables de entorno y guardamos cada una de ellas en su propia variable.

Este Módulo usa tres funciones auxiliares para obtener la respuesta a la pregunta del usuario y mostrarla en la página con dicha respuesta resaltada en el texto del que ha sido extraída: *queryJSON*, *getAnswers* y *annotateContext*.

Listing 3.5: Función que realiza una petición HTTP al servicio de pregunta-respuesta.

```
1 def queryJSON(queryURL, question):
2     """
3     Auxiliary function to query the QA service
4     """
5     files = {
6         'question': (None, question),
7     }
8     response = requests.get(queryURL, files = files)
9     if response:
10         return response.json()
```

queryJSON realiza la pregunta al Servidor de Pregunta-Respuesta, y devuelve su respuesta en formato JSON si la respuesta del servidor no es None.

Listing 3.6: Función que obtiene todas las respuestas posibles a la pregunta dada.

```
1 @st.cache(show_spinner=False, allow_output_mutation=True)
2 def getAnswers(question):
3     """
4     Auxiliary function that queries the QA service and returns the answers
5     """
6     answerList = []
7     #We iterate over the routings defined in EQA_SERVICE_ROUTINGS if it is not empty
8     if EQA_SERVICE_ROUTINGS:
9         for routing in EQA_SERVICE_ROUTINGS:
10             queryURL = EQA_SERVICE_DIRECTION + routing
11             answer = queryJSON(queryURL,question)
12             #If the answer is not None, we add it to the answerList
13             if answer:
14                 #If there are multiple answers in the returned JSON, we iterate over them
15                 if MULTIPLE_ANSWERS_JSON:
16                     for uniqueAnswer in answer["answers"]:
```

3.2. Módulo de Pregunta-Respuesta

```
17         answerList.append(uniqueAnswer)
18     else:
19         answerList.append(answer)
20 else:
21     queryURL = EQA_SERVICE_DIRECTION
22     answer = queryJSON(queryURL,question)
23     if answer:
24         if MULTIPLE_ANSWERS_JSON:
25             for uniqueAnswer in answer["answers"]:
26                 answerList.append(uniqueAnswer)
27         else:
28             answerList.append(answer)
29     return answerList
```

getAnswers consulta el Sistema de Pregunta-Respuesta usando queryJSON, y crea una lista de JSONs con todas las respuestas que devuelve el Sistema. Distinguimos si hay varias rutas en el Sistema, y si en cada JSON hay una o varias respuestas tal y como vimos en los formatos de respuesta aceptados por nuestro. Esta función se ha definido con la anotación st.cache para optimizar el rendimiento de la interfaz, haciendo que se guarde en caché el último par de pregunta-respuesta.

Listing 3.7: Método que anota la respuesta en su contexto.

```
1 def annotateContext(answer, context, answerStart, answerEnd):
2     '''
3     Auxiliary function that annotates the context of the answer
4     '''
5     #Extract answer from context. Initialize tag as "ANSWER" and colour as green
6     answerInText = context[answerStart:answerEnd]
7     color = "#adff2f"
8     tag = "ANSWER"
9     #If the answer is different from the one in the context, we annotate it in blue with "EVIDENCE"
10    " tag
11    if answer != answerInText:
12        #Cambiamos la etiqueta a "EVIDENCE" y el color a azul
13        tag = "EVIDENCE"
14        color = "#8ef"
15    #Annotate answer in context text
16    annotated_text(context[:answerStart],(answerInText,tag,color),context[answerEnd:],)
```

La función auxiliar annotateContext es la encargada de señalar la respuesta en el contexto de la pregunta. Dadas la respuesta, su contexto y la posición del inicio y el final de la respuesta, marca la respuesta como ANSWER o EVIDENCE en función de si la respuesta dada por el sistema coincide con la parte exacta del texto donde se debería encontrar o no.

Question: What games were produced by Riot Games?

... The location city of Riot Games is Los Angeles. The colwidth of Riot Games is 40. The industry of Riot Games is Video game industry. The type of Riot Games is Subsidiary. The num employees of Riot Games is 2500. The products of Riot Games is , League of Legends, League of Legends: Wild Rift, Legends of Runeterra, Teamfight Tactics, Valorant . The product of Riot Games is League of Legends, League of Legends: Wild Rift, Legends of Runeterra, Teamfight Tactics, Valorant. The founded of Riot Games is in Santa Monica, California, US. The founders of Riot Games is , Brandon Beck, Marc Merrill. The image caption of Riot Games is Riot Games' West Los Angeles headquarters. The name of Riot Games is Riot Games, Inc.. The hq location country of Riot Games is US. The key person of Riot Games is Chairperson, Chief executive officer, Chief financial officer. The logo caption of Riot Games is Logo used since 2019. The num locations of Riot Games is 24. The number of employees of Riot Games is 2500. The number of locations of Riot Games is 24. The hq location city of Riot Games is Los Angeles. The parent of Riot Games is Tencent...

Answer: League of Legends, League of Legends: Wild Rift, Legends of Runeterra, Teamfight Tactics, Valorant

Figura 3.2: Respuesta a pregunta anotada en la interfaz.

Desarrollo del Sistema

Question: Was Valorant created by Riot Games?

... The location city of Riot Games is Los Angeles. The colwidth of Riot Games is 40. The industry of Riot Games is Video game industry. The type of Riot Games is Subsidiary. The num employees of Riot Games is 2500. The products of Riot Games is , League of Legends, League of Legends: Wild Rift, Legends of Runeterra, Teamfight Tactics, Valorant.

The product of Riot Games is League of Legends, League of Legends: Wild Rift, Legends of Runeterra, Teamfight Tactics, Valorant. **EVIDENCE** The founded of Riot Games is in Santa Monica, California, US. The founders of Riot Games is , Brandon Beck, Marc Merrill. The image caption of Riot Games is Riot Games' West Los Angeles headquarters. The name of Riot Games is Riot Games, Inc. . The hq location country of Riot Games is US. The key person of Riot Games is Chairperson, Chief executive officer, Chief financial officer. The logo caption of Riot Games is Logo used since 2019. The num locations of Riot Games is 24. The number of employees of Riot Games is 2500. The number of locations of Riot Games is 24. The hq location city of Riot Games is Los Angeles. The parent of Riot Games is Tencent...

Answer: yes

Figura 3.3: Respuesta a pregunta booleana en la interfaz.

Como se puede ver en la Figura 3.3 hay Sistemas de Pregunta y Respuesta que a partir de una parte concreta del texto construyen una nueva respuesta. Esto suele darse en preguntas de tipo booleano o numéricas. Es en estos casos, dado que lo que se encuentra en el texto no es la respuesta como tal sino evidencia para contruirla, que marcamos en el texto con la etiqueta EVIDENCE y el color azul (#8ef) en lugar de verde (#adff2f, Figura 3.2)

Listing 3.8: Inicialización de conexiones. Código que genera el título y descripción de la pestaña; y la barra de búsqueda de preguntas.

```
1 def app(db):
2     #Create worksheet connection
3     spread = spreadManager.SpreadManager(WORKSHEET, WORKSHEET_ID, SPREADSHEET)
4
5     #Question-Answering module subtitle and description
6     st.subheader('Question-Answering')
7
8     st.markdown("""
9     Write any question below or use a random one from a pre-loaded datasets!
10    """, unsafe_allow_html=True)
11
12    #Search bar
13    question = st.text_input("")
```

Con respecto al código de la página web como tal, comenzamos creando la conexión a la Hoja de Cálculo, y mostrando el subtítulo del Módulo de Pregunta-Respuesta, el cuerpo de la página y el campo de texto (buscador) en el que escribiremos la pregunta.

Listing 3.9: Código que crea el selector y botón para hacer una pregunta de un conjunto de datos y genera las opciones de la barra lateral.

```
1     #Dataset Selector for random questions.
2     selectorList = ["All"]
3     selectorList.extend(db.getCollections())
4     dataset = st.selectbox("Select a DataSet", selectorList)
5
6     #Button to get a random question
7     randomQuestion = st.button("Random Question")
8
9     #Sidebar title and slider
10    st.sidebar.subheader('Options')
11    answerNumber = st.sidebar.slider('How many relevant answers do you want?', 1, 10,
12    DEFAULT_NUMBER_OF_ANSWERS)
13
14    modelAnswer = None
15
16    #Obtain a random question and answer it if the random question button is pressed
```

3.2. Módulo de Pregunta-Respuesta

```
16     if randomQuestion:
17         randomDict = db.getRandomDocument(1,dataset)[0]
18         question = randomDict["question"]
19         modelAnswer = randomDict["answer"]
```

Luego, escribimos en la interfaz el selector para seleccionar el conjunto de datos del que obtendremos las preguntas aleatorias y un botón para pulsar cuando queramos obtener una. Las preguntas aleatorias obtenidas se guardarán en la variable `question`, al igual que aquellas provenientes del buscador, para así simplificar el código. Por último, creamos un deslizador para la barra de navegación lateral, con el cual seleccionaremos cuántas preguntas queremos que nos muestre la página.

Listing 3.10: Código que se ejecuta al recibir una pregunta.

```
1     #If the question is not empty, we query the QA service
2     if question:
3         st.write("**Question: **", question)
4         #If there is a model answer, we show it
5         if modelAnswer:
6             st.write("**Expected Answer: **", modelAnswer)
7             st.write("\n")
8             modelAnswer = None
9         #Spinner to show that the app is looking for answers
10        with st.spinner(text=':hourglass: Looking for answers...'):
11            counter = 0
12            highestScoreAnswer = {}
13            results = getAnswers(question)
14            #Sort the answers by score
15            results.sort(key = operator.itemgetter('confidence'), reverse = True)
16            for idx,response in enumerate(results):
17                if counter >= answerNumber:
18                    break
19                counter += 1
20                answer = response['answer']
21                if answer and answer != "-":
22                    context = "... " + response["evidence"]["summary"] + "... "
23                    confidence = response["confidence"]
24                    annotateContext(response, answer, context, response["evidence"]["start"],
25                                response["evidence"]["end"])
26                    st.write("**Answer: **", answer)
27                    source = response["source"]
28                    st.write('**Relevance:** ', confidence , '**Source:** ', source)
29                    st.write('**Relevance:** ')
30                    #Save the answer with the highest score in a dictionary
31                    if idx == 0:
32                        highestScoreAnswer = {
33                            "answer": answer,
34                            "confidence": confidence
35                        }
```

Si se hace una pregunta (la cadena de caracteres que contiene la pregunta no está vacía), se escribe la pregunta en la página y un *spinner loader* con un texto que le indicará al usuario que la respuesta está cargando (el Sistema de Pregunta-Respuesta está buscándola). Una vez encontradas todas las respuestas, se ordenan por su puntuación o confianza y se mostrarán en pantalla tantas respuestas como el usuario haya indicado que quiera que se muestren.

Desarrollo del Sistema

Listing 3.11: Código ejecutado después de obtener todas las respuestas. Crea botones para valorar las respuestas y guarda estas valoraciones.

```
1      #Once answers are found, we display buttons to rate them
2      st.write("Please rate if our answer has been helpful to you so we can further improve our
          system!")
3      col1, col2 = st.columns([1,1])
4      with col1:
5          isRight = st.button(":thumbs_up:")
6      with col2:
7          isWrong = st.button(":thumbs_down:")
8
9      #If the correct/incorrect button is pressed, we save the answer in the spreadsheet
10     if isRight or isWrong:
11         spread.insertRow([[question, highestScoreAnswer["answer"], str(highestScoreAnswer["
            confidence"]), isRight, str(datetime.now(tz="Europe/Madrid"))]])
12         #Reset buttons value and show a receipt message to the user
13         isRight = False
14         isWrong = False
15         st.success(":sparkles: Thanks for your input!")
```

Una vez mostradas en la página web todas las respuestas, aparecen dos botones bajo estas que sirven para que el usuario pueda marcar si alguna de ellas es correcta o no. Al pulsar un botón, el *input* del usuario es registrado en la Hoja de Cálculo de Google Sheets definida en las variables de entorno. En ella se escribirán el nombre de la pregunta, la respuesta con mayor puntuación junto a esta puntuación, si la respuesta es correcta y la fecha en que se ha hecho la pregunta como se puede ver en 3.4.

	A	B	C	D	E	F
1	Question	Answer	Confidence	Correct	Date	
2	Where was Fernando Alonso born?	Oviedo	0.9754	TRUE	2022-03-22 16:26:51.847035+01:00	
3	Was Valorant created by Riot Game: yes		0.1266	TRUE	2022-04-10 18:23:57.381504+02:00	
4	What games were produced by Riot League of Legends, League of Legends: W		0.5483	TRUE	2022-04-10 11:37:04.969497+02:00	
5	Who is the leader of Allgemeine SS	Heinrich Himmler	0.8236	TRUE	2022-05-03 12:04:30.565131+02:00	
6	To which family does korean fox belk vulpes		0.2013	FALSE	2022-05-03 12:07:19.685028+02:00	
7						

Figura 3.4: Hoja de Cálculo con valoraciones de los usuarios.

Esta funcionalidad nos permite validar el Sistema por medio de los propios usuarios que lo utilizan, sorteando los inconvenientes que presentan habitualmente los conjuntos de datos y métricas mencionados en la Introducción. Además, también nos crea un conjunto de datos anotado que podría ser usado posteriormente para evaluar otros Sistemas de Pregunta-Respuesta.

Listing 3.12: Código que hace aparecer una casilla de selección en la barra lateral

```
1      #Sidebar Checkbox. If checked, we show the QA Service JSON response
2      if question and st.sidebar.checkbox('Show JSON Response', key = 0):
3          st.subheader('API JSON Response')
4          st.write(results)
```

Tras mostrarse las respuestas, también aparece en la barra lateral una casilla de selección que hará aparecer el JSON obtenido como respuesta en la página, como se puede ver en Figura 3.5. También se ve el deslizante para el número de respuestas.

3.3. Módulo de Gestión de Conjuntos de Datos

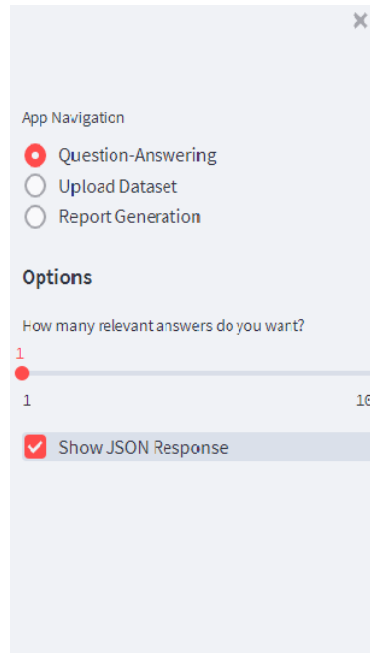


Figura 3.5: Barra de Navegación del Módulo de Question-Answering después de ofrecer la respuesta al usuario.

3.3. Módulo de Gestión de Conjuntos de Datos

Esta sección de la herramienta es la encargada de subir los conjuntos de datos a la base de datos.

Listing 3.13: Código ejecutado por el módulo de Gestión de Conjuntos de Datos

```
1
2 import streamlit as st
3 from utils import processDatasets
4
5 def app(db):
6
7     #Question-Answering module subtitle and how to use it
8     st.subheader('Dataset Management')
9
10    st.markdown("""
11    You may upload your dataset below. For it to be processed and uploaded to our database, please
12    follow these guidelines:
13    - 1. Upload your dataset either on .CSV or .JSON format.
14    - 2. JSONs may be on JSON lines or JSON array format.
15    - 3. Answers should be on the "answer" column/key, and Questions on the "question" column/key.
16    - 4. If your Answer is verbalized, you shall name its key/column "verbalized_answer", and
17    format it with the answer between brackets, i.e. "Fernando Alonso was born in [Oviedo]."
18    """, unsafe_allow_html=True)
19
20    #Buffer to store the uploaded file (dataset)
21    inputBuffer = st.file_uploader("Upload an Image", type=["csv","json"])
22
23    #If a file is uploaded, we process it and upload it to the database
24    if inputBuffer:
25        try:
```

```
24     filename = inputBuffer.name
25     #Split name into file name and extension
26     splitFilename = filename.split(".")
27     datasetDict = processDatasets.formatDatasets(inputBuffer, isCsv=(splitFilename[1] == "
        csv"))
28     datasetName = splitFilename[0].lower()
29     #If the dataset is correctly processed, we try to upload it to MongoDB
30     if datasetDict:
31         db.importDataset(datasetDict, datasetName)
32         #If the dataset is successfully uploaded, we show a success message
33         if datasetName in db.getCollections():
34             st.success(":sparkles: Your dataset has been registered on our database!")
35             st.write("A dataset with name ", datasetName, "and length ", len(datasetDict),
                " questions has been registered on MongoDB")
36         else:
37             st.error("We could not upload your dataset on our database. Please contact the
                administrator.")
38     else:
39         st.error("Your dataset could not be processed correctly. Please revise the format
                or contact the administrator")
40 except Exception as e:
41     st.exception(e)
```

Empezamos el código escribiendo el subtítulo del Módulo y una breve guía sobre los conjuntos de datos aceptados. Estos conjuntos deben estar en formato csv o json, y en caso de ser jsons se permiten dos opciones:

- JSON Array: Archivo JSON por defecto, en formato de lista (Código 2.1 y 2.2).
- JSON Line: Cada línea del conjunto de datos debe ser un objeto JSON (2.3).
- CSV: También se permiten archivos CSV, con cualquier separador.

Como podemos ver en los ejemplos, se exige que las preguntas estén bajo la clave o la columna *question*, y las respuestas o bien en *verbalized_answer* con la respuesta entre corchetes en caso de darse la respuesta larga o bien en *answer*.

El usuario sube el conjunto de datos a la interfaz usando el siguiente componente, y se extrae la extensión del archivo para pasarla como argumento a la función auxiliar que procesa los *datasets*.

Una vez procesado el archivo, se intenta subir a Mongo en una nueva colección con el mismo nombre que el archivo. Si se ha podido subir correctamente el conjunto de datos, se muestra un mensaje de éxito, y en caso contrario saltará un mensaje de error en función de si el proceso ha fallado durante el procesamiento del conjunto de datos o la subida a la base de datos:

3.4. Módulo de Generación de Informes

Como hemos comentado previamente, al probar los Sistemas de Pregunta-Respuesta usando preguntas de un conjunto de datos, cuán bien escritas están estas preguntas y la distribución de los tipos de respuesta son métricas cruciales a tener en cuenta para poner en contexto el desempeño del Sistema. Con este módulo, pretendemos

3.4. Módulo de Generación de Informes

evaluar también los conjuntos de datos subidos por los usuarios, ofreciendo un reporte exportable en PDF sobre un conjunto de datos previamente subido. Crearemos un *DataFrame* de Pandas, una estructura de datos en dos dimensiones, mutable y que permite datos de distintos tipos en la cual guardaremos las preguntas junto a su respuesta, y les asociaremos un tipo (*Boolean*, *Number*, *String* o *Date*) y una puntuación de fluidez o *fluency score*. Por medio de este Dataframe se le presentarán estadísticas y gráficas sobre las preguntas del *dataset* al usuario.

Las preguntas se clasificarán en base al tipo de su respuesta. Distinguiremos entre preguntas de Verdadero o Falso (booleanas), numéricas, con una cadena de caracteres como respuesta o con una fecha, y esta clasificación se hará usando un *fine-tuning* del Modelo de Lenguaje BERT. Por otro lado, la fluidez con la que está escrita la pregunta es calculada usando el Modelo de Lenguaje de ZAMIA para la lengua inglesa. Entraremos más en detalle sobre la implementación y la interpretación de estas métricas en el Capítulo 3.5.5

Listing 3.14: Código con las librerías importadas para la generación de reportes e inicialización del PDF y el clasificador de preguntas

```
1 import os
2 import base64
3 import pandas as pd
4 import streamlit as st
5 import plotly.graph_objs as go
6 from fpdf import FPDF, HTMLMixin
7 from utils import questionClassifier
8 from tempfile import NamedTemporaryFile
9
10 resourcesDir = "/utils/resources_dir"
11 classifier = questionClassifier.QuestionClassifier(resourcesDir)
12
13 class MyFPDF(FPDF, HTMLMixin):
14     pass
```

Realizamos los *imports* necesarios. Creamos una clase MyFPDF vacía para poder usar la librería fpdf con la que guardaremos el informe en un archivo PDF, e inicializamos el directorio donde estarán los Modelos de Lenguaje que emplearemos como “src/re-

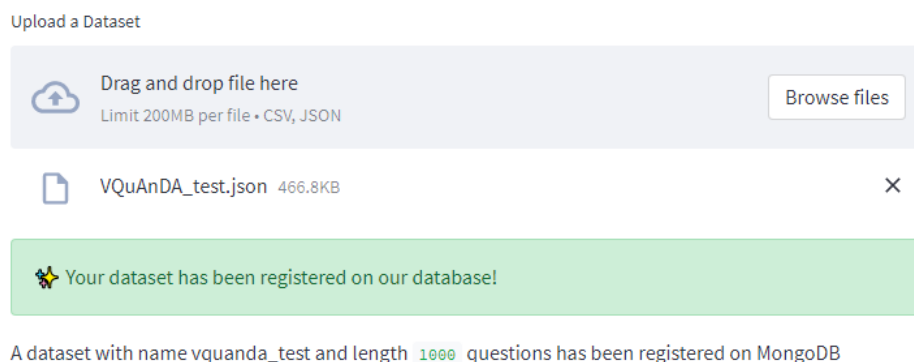


Figura 3.6: Mensaje de éxito al subir un conjunto de datos correctamente.

sources_dir" y la clase auxiliar del clasificador de preguntas.

Listing 3.15: Métodos auxiliares que escriben el reporte en el PDF y en la interfaz

```
1 def writeSection(pdf, title, body, newPage = False):
2     """
3     Auxiliary Function to write a section in the PDF and in the interface
4     """
5     #If newPage is True, add a new page onto the PDF
6     if newPage:
7         pdf.add_page()
8
9     #Write the title of the new section
10    pdf.set_font('Times', 'B', 16)
11    pdf.cell(40, 10, title)
12    pdf.ln(10)
13    st.subheader(title)
14
15    #Write the body of the section
16    pdf.set_font('Times', '', 12)
17    for i in body:
18        st.write(i)
19        pdf.cell(40, 10, i)
20        pdf.ln(10)
21    #Add extra space
22    pdf.ln(5)
23
24 def writeTable(pdf, df, newPage = False, title = ""):
25     """
26     Auxiliary function to write a table in the PDF and in the interface
27     """
28     if newPage:
29         pdf.add_page()
30
31     #Write the title of the new section if it exists
32     if title:
33         st.subheader(title)
34         pdf.set_font('Times', 'B', 16)
35         pdf.cell(40, 10, title)
36         pdf.ln(10)
37
38     #Write the table on the UI
39     st.write(df)
40
41     #The table written on the PDF report will be a preview (first 5 rows)
42     df = df.head(5)
43     #Write the table headers
44     pdf.set_font('Times', '', 10)
45     pdf.cell(100, 10, "Question", 1, 0, 'C')
46     pdf.cell(45, 10, "Answer", 1, 0, 'C')
47     pdf.cell(22, 10, "fluencyScore", 1, 0, 'C')
48     pdf.cell(22, 10, "answerType", 1, 0, 'C')
49     pdf.ln(10)
50     #Write the table rows
51     for i in range(0, len(df)):
52         pdf.cell(100, 10, '%s' % (df["question"].iloc[i][:50] + "...") if (len(df["question"].iloc[i]) > 50) else df["question"].iloc[i], 1, 0, 'C')
53         pdf.cell(45, 10, '%s' % (df["answer"].iloc[i][:20] + "...") if (len(df["answer"].iloc[i]) > 20) else df["answer"].iloc[i], 1, 0, 'C')
54         pdf.cell(22, 10, '%s' % "{:.2f}".format(df["fluencyScore"].iloc[i]), 1, 0, 'C')
55         pdf.cell(22, 10, '%s' % df["answerType"].iloc[i], 1, 0, 'C')
56         pdf.ln(10)
57     pdf.ln(5)
58
```

3.4. Módulo de Generación de Informes

```
59 def writeFigure(pdf, fig, newPage = False, title = ""):
60     """
61     Auxiliary function to write a figure in the PDF and in the interface
62     """
63     if newPage:
64         pdf.add_page()
65
66     if title:
67         pdf.set_font('Times', 'B', 16)
68         pdf.cell(40, 10, title)
69         pdf.ln(10)
70     #Write the figure on the UI
71     st.plotly_chart(fig)
72     #Write the figure on the PDF report. Save it onto a temporary file in png format
73     with NamedTemporaryFile(delete=False, suffix=".png") as tmpfile:
74         fig.write_image(tmpfile.name)
75         #Read the temporary file and insert it into the PDF
76         pdf.image(tmpfile.name, w = 115, h = 115)
77     pdf.ln(5)
78
79 def generatePDF(pdf, fileName, export = False):
80     """
81     Auxiliary function to generate the PDF report
82     """
83     #Create download link and write it on the UI
84     downloadLink = f'<a href="data:application/octet-stream;base64,{(base64.b64encode((pdf.output(
85         dest = "S")))).decode()}" download="{fileName}.pdf">Download file</a>'
86     st.markdown(downloadLink, unsafe_allow_html=True)
87     #If export is True, save the report as a local file
88     if export:
89         pdf.output(fileName, "F")
```

Tenemos los siguientes métodos para generar el reporte como PDF:

- **writeSection:** Funcion auxiliar que escribe una seccion del reporte en la interfaz y en el archivo pdf. Puede agregar una nueva pagina al documento. Los argumentos pasados son el archivo pdf sobre el que escribiremos, un título para la sección, el texto a escribir como una lista de cadenas de caracteres y un booleano *newPage*, *False* por defecto, que indica si esta sección debe crearse en una nueva página
- **writeTable:** Funcion auxiliar que escribe una tabla (pandas dataframe) en en la interfaz y en el archivo pdf. Dado un pdf, la tabla, el booleano *newPage* y el título de la sección (cadena vacía por defecto), escribe el dataframe en la interfaz y sus primeras 5 filas en el pdf, con el texto de cada celda limitado a 50 caracteres.
- **writeFigure:** Funcion auxiliar que escribe una figura en la interfaz y en el archivo pdf. Dado un pdf, la figura, el booleano *newPage* y el título de la sección, añade la figura a la página web y al PDF.
- **generatePDF:** Funcion auxiliar que genera el reporte en PDF como enlace o archivo. Crea el enlace de descarga del PDF y lo muestra en la web, y en caso de ser llamado con *export = True* guarda también el reporte PDF en local.

Listing 3.16: Método que genera la tabla con las métricas para cada pregunta

```
1 def generateDataframe(db, datasetName):
```

Desarrollo del Sistema

```
2 """
3 Auxiliary function to generate a dataframe from the database
4 """
5 #Get the dataset from the database and convert it into a dataframe
6 df = pd.DataFrame(db.getAllDocuments(datasetName))
7 #Keep only "question" and "answer" columns
8 df = df[["question", "answer"]]
9
10 #Create new columns answerType and fluencyScore using the classifier methods
11 df["answerType"] = df["question"].apply(classifier.getAnswerCategory)
12 df["fluencyScore"] = df["question"].apply(classifier.getFluencyScore)
13 return df
```

El Dataframe con las preguntas y sus características se construye usando la función auxiliar `generateDataframe`. Usando el método “`apply`” de Pandas, que aplica una función dada a cada elemento a lo largo de una fila o columna (en este caso la columna “`question`”), aplicamos las funciones del clasificador de preguntas que nos dan la categoría y la fluidez de la pregunta a cada pregunta del conjunto de datos. Con esto, creamos dos nuevas columnas “`answerType`” y “`fluencyScore`”.

Listing 3.17: Código ejecutado por la pestaña de Generación de Reporte

```
1 def app(db):
2
3     st.markdown("""
4     Select a dataset to generate a quality report on it.
5     """, unsafe_allow_html=True)
6
7     selectorList = []
8     selectorList.extend(db.getCollections())
9     dataset = st.selectbox("Select a DataSet", selectorList)
10    run = st.button("Run")
11
12    if dataset and run:
13
14        with st.spinner(text=":hourglass: Generating report. This may take some minutes..."):
15
16            df = generateDataframe(db, dataset)
17            pdf = MyFPDF()
18
19            count = df["answerType"].value_counts()
20            labels = count.index
21            values = count.values
22            layout = go.Layout(height = 600,
23                               width = 600,
24                               autosize = False,
25                               title = "Questions by answer type, " + dataset + " dataset")
26            traces = go.Pie(labels = labels, values = values)
27            fig = go.Figure(data = traces, layout = layout)
28
29            st.header("Quality Report")
30            writeTable(pdf, df, title = "Dataset Preview", newPage = True)
31            body = ["The number of distinct questions in the dataset is " + str(len(df)),
32                  "The average fluency score is " + "{:.2f}".format(df.fluencyScore.mean()) + " and its
33                    median is " + "{:.2f}".format(df.fluencyScore.median()),
34                  "Maximum and minimum fluency scores are " + "{:.2f}".format(df.fluencyScore.max()) + "
35                    and " + "{:.2f}".format(df.fluencyScore.min()) + " respectively. Standard
36                    deviation is " + "{:.2f}".format(df.fluencyScore.std())]
37            writeSection(pdf, "Fluency Score", body)
38
39            answerTypes = df.answerType.unique()
```

```
37     body = ["There are " + str(len(answerTypes)) + " different types of answer in this  
        dataset, which are: " + ", ".join(answerTypes)]  
38     writeSection(pdf, "Answer Types", body)  
39     writeFigure(pdf, fig)  
40     generatePDF(pdf, dataset + "_report")
```

Con respecto al código que ejecuta la página en sí, tenemos un texto y un título, junto a un selector que permite elegir el conjunto de datos y un botón para correr el código que genera el reporte. Una vez elegido un conjunto de datos y pulsado el botón, aparecerá un mensaje de carga hasta que el reporte haya terminado de generarse. Creamos el DataFrame, el PDF vacío y un gráfico circular o de tarta para representar la distribución de los tipos de pregunta. Se ha elegido este tipo de gráfica debido al bajo número de clases (4). Con respecto a la librería para construir esta figura, se ha escogido Plotly ¹ debido a su facilidad y a la multitud de funcionalidades que ofrece tales como la posibilidad de hacer zoom o de ver información sobre cada sector al poner el cursor encima.

3.5. Herramientas auxiliares

En la carpeta “src/utills” del repositorio se encuentran los scripts auxiliares llamados por los módulos principales de la página web.

3.5.1. Multipage

Streamlit no tiene ninguna opción integrada que permita navegar entre múltiples páginas dentro de una misma aplicación. Por ello, para separar las 3 páginas que constituyen nuestro servicio, se ha implementado una clase *MultiPage* en *multipage.py* que permite viajar entre estas páginas por medio de una barra de navegación lateral.

En primer lugar tenemos el constructor de la clase, que inicializa un diccionario *pages* vacío como único atributo. Los diccionarios son estructuras de datos en Python que almacenan datos como pares de claves de la forma Clave - Valor, permitiéndonos así guardar una serie de mapeos entre dos conjuntos de elementos. En nuestro caso estos dos elementos serán el título de la página y la función que se ha de ejecutar para correr esta página.

Listing 3.18: Constructor de la clase *MultiPage*

```
1 import streamlit as st  
2  
3 class MultiPage:  
4     """  
5     Framework that combines multiple Streamlit applications in one single website.  
6     """  
7     def __init__(self) -> None:  
8         """  
9         Class constructor  
10        """  
11        self.pages = {}
```

¹Plotly - Data Visualization for ML and data science models

Tenemos también dos métodos:

- **addPage:** Recibe el título de la página y su función, y los añade a pages con el título como clave y la función como valor.
- **run:** Método que recibe la conexión a la base de datos como argumento y ejecuta el código de la aplicación. Dibuja el selector para navegar entre las páginas en la barra lateral, inicializando la variable page como la página escogida, y después ejecuta la función asociada a page.

Listing 3.19: Método que agrega una nueva pestaña a la página web y método que cambia de pestaña (ejecutando su código)

```
1  def addPage(self, title, func) -> None:
2      """
3      Method that adds a new page to the MultiPage (adds new entry to the pages dictionary)
4      """
5      self.pages.update({title: func})
6
7  def run(self, db):
8      """
9      Method that runs the MultiPage (executes the selected page).
10     """
11     #Selector
12     page = st.sidebar.radio(
13         "App Navigation",
14         self.pages.keys()
15     )
16
17     #Run app of the selected page
18     self.pages[page](db)
```

3.5.2. Gestor de MongoDB

La clase dbManager.py maneja la conexión a la base de datos y las tareas necesarias para la gestión de los conjuntos de datos subidos, en dbManager.py. Para ello, utiliza la librería PyMongo, controlador oficial de MongoDB para aplicaciones síncronas en Python.² Para esta clase, además de PyMongo, importaremos las librerías certifi (que nos permitirá crear un certificado SSL en caso de que nos conectemos a una base de datos en la nube) y random.

Esta clase está preparada tanto para tener una base de datos en local como para conectarse a la base de datos en la nube oficial de MongoDB en la nube, MongoDB Atlas. En el constructor los campos de contraseña, nombre de usuario y nombre del *cluster* de la base de datos se inicializarán como la cadena vacía ya que únicamente son necesarios para conectarse a Atlas. La dirección de la base de datos es el único atributo obligatorio al ser común para ambas opciones. Al crear la conexión a la base de datos, distinguimos entre base de datos local y en la nube comprobando si clusterName, userName y userPassword son la cadena vacía.

²PyMongo - MongoDB Drivers

Listing 3.20: Constructor de la clase *dbManager*.

```

1 import random
2 import certifi
3 from pymongo import MongoClient
4
5 class DbManager:
6
7     def __init__(self, serverDir, clusterName = "", userName = "", userPassword = ""):
8         """
9         Class Constructor
10        """
11        self.serverDir = serverDir
12        self.clusterName = clusterName
13        self.userName = userName
14        self.userPassword = userPassword
15
16        if clusterName and userName and userPassword:
17            self.database = MongoClient("mongodb+srv://" + userName + ":" + userPassword + "@" +
18                                       clusterName + "?retryWrites=true&w=majority", tlsCAFile = certifi.where())
19        else:
20            self.database = MongoClient(serverDir).database

```

La clase tiene los siguientes métodos:

- **getCollections:** Método que devuelve la lista de colecciones en nuestra base de datos
- **importDataset:** Método que recibe un conjunto de datos y lo inserta a la base de datos. Distingue si la longitud del conjunto de datos es superior a 1 o no.
- **getRandomDocument:** Método que dado un argumento numérico n, devuelve n preguntas-respuestas aleatorias de la base de datos.
- **getAllDocuments:** Método que dado el nombre de un conjunto de datos de nuestra colección, devuelve todas las preguntas y respuestas de este.
- **getDocumentCount:** Método que dado el nombre de un conjunto de datos de nuestra colección, devuelve el número de preguntas y respuestas que contiene.
- **dropCollection:** Método que dado el nombre de un conjunto de datos de nuestra colección, elimina dicho *dataset*
- **clearDatabase:** Método que elimina todo conjunto de datos de nuestra base de datos.
- **getStatus:** Método que devuelve el estado de la base de datos.

Listing 3.21: Métodos de la clase *dbManager*

```

1 def getCollections(self):
2     """
3     Method that returns the name of the collections in the database
4     """
5     return self.database.list_collection_names()
6
7 def importDataset(self, dataset, datasetName):
8     """
9     Method that imports a dataset into the database

```

```
10         """
11         newCol = self.database[datasetName]
12         if len(dataset) > 1:
13             newCol.insert_many(dataset)
14         else:
15             newCol.insert_one(dataset)
16
17     def getRandomDocument(self, number, dataset):
18         """
19         Method that returns number random documents from a dataset
20         """
21         if dataset == "All":
22             dataset = random.choice(self.getCollections())
23
24         documentList = []
25         randomCursor = self.database[dataset].aggregate([{"$sample": {"size": number}}])
26
27         for document in randomCursor:
28             documentList.append(document)
29         return documentList
30
31     def getAllDocuments(self, dataset):
32         """
33         Method that returns all documents from a dataset
34         """
35         collection = self.database[dataset]
36
37         documentList = []
38         for document in collection.find():
39             documentList.append(document)
40
41         return documentList
42
43     def getDocumentCount(database, dataset):
44         """
45         Method that returns the number of documents in a dataset
46         """
47         return database[dataset].count_documents({})
48
49     def dropCollection(self, collectionName):
50         """
51         Method that drops a collection
52         """
53         if collectionName in self.getCollections():
54             col = self.database[collectionName]
55             col.drop()
56
57     def clearDatabase(self):
58         """
59         Method that clears the database
60         """
61         for collectionName in self.getCollections():
62             col = self.database[collectionName]
63             col.drop()
64
65     def getStatus(self):
66         """
67         Method that returns the status of the database
68         """
69         return self.database("serverStatus")
```

3.5.3. Gestor de Google Sheets

La clase `spreadManager.py` maneja la conexión al Libro de Cálculo en el que guardaremos las validaciones introducidas por los usuarios para generar conjuntos de datos anotados.

Importamos la librería `os`, que proporciona funciones para interactuar con el Sistema Operativo, y las librerías `googleapiclient` y `oauth2client` para realizar la conexión a Google Sheets. A continuación, definimos el directorio donde se encuentra el archivo JSON con los credenciales de la cuenta que usaremos para modificar el Libro de Cálculo.

Listing 3.22: Librerías importadas e inicialización del directorio con las credenciales de la Cuenta de Servicio.

```
1 import os
2 from googleapiclient.discovery import build
3 from oauth2client.service_account import ServiceAccountCredentials
4
5 #Path to the credentials file
6 credentialsPath = '/utils/credentials.json'
```

En el constructor inicializamos los siguientes atributos de clase: `Scope`, que definirá las APIs de Google que usaremos para acceder la Hoja de Cálculo (en nuestro caso las de Google Drive y Sheets), nombre e id del Libro de Cálculo a modificar y el nombre de la Hoja en concreto donde guardaremos los *inputs* de los usuarios. Definim

Listing 3.23: Constructor de la clase *SpreadManager*

```
1 class SpreadManager:
2
3     def __init__(self, spreadsheet, spreadsheetId, validationSheet):
4         """
5         Class constructor
6         """
7         self.scope = ['https://spreadsheets.google.com/feeds', 'https://www.googleapis.com/auth/drive']
8         self.spreadsheet = spreadsheet
9         self.spreadsheetId = spreadsheetId
10        self.validationSheet = validationSheet
11
12        if os.path.exists(credentialsPath):
13            self.creds = ServiceAccountCredentials.from_json_keyfile_name(credentialsPath, self.scope)
14            self.service = build("sheets", "v4", credentials=self.creds, cache_discovery=False)
15            self.connector = self.service.spreadsheets()
16        else:
17            print("DB ERROR > Missing Credentials for accessing")
18            exit()
```

La clase tiene un único método `insertRows` que inserta una nueva fila en la Hoja. Este añade la fila dada en la primera que tenga 5 columnas libres. Los campos introducidos son la pregunta, la respuesta obtenida, si esta ha sido calificada como correcta por el usuario (booleano `True/False`), la confianza que el Sistema de Pregunta-Respuesta le da a la respuesta y la fecha y hora en que se ha hecho la pregunta.

Listing 3.24: Método que inserta una nueva fila en la hoja de cálculo

```
1 def insertRow(self, row):
2     """
3     Method that inserts a row into the spreadsheet
4     """
5     values = (
6         self.connector.values().append(
7             spreadsheetId=self.spreadsheetId,
8             range=f"{self.validationSheet}!A:E",
9             body=dict(values=row),
10            valueInputOption="USER_ENTERED",
11        ).execute()
12    )
```

3.5.4. Preprocesamiento de Conjuntos de Datos

En el archivo *processDatasets.py* se encuentran las funciones auxiliares necesarias para procesar y filtrar los conjuntos de datos subidos por el usuario y enviarlos a nuestra base de datos.

Listing 3.25: Código de la clase *processDatasets*.

```
1 import re
2 import os
3 import json
4 import pandas as pd
5
6 # Dataset fields we want to keep
7 keysToKeep = ["question", "answer"]
8
9 def jsonToDict(file):
10     """
11     Auxiliary function that converts a given a JSON file to dictionary list.
12     """
13     return json.loads((file.read()).decode('utf-8'))
14
15 def jsonLineToDict(file):
16     """
17     Auxiliary function that converts a given a JSON-Line file (JSON Objects in each line) to
18     dictionary list.
19     """
20     return json.loads(json.dumps([json.loads(jsonLine) for jsonLine in (file.read()).decode('utf-8')
21     ].splitlines()))
22
23 def csvToDict(file):
24     """
25     Auxiliary function that converts a given a CSV file to dictionary list.
26     """
27     df = pd.read_csv(file, sep=None, engine="python")
28     df = df.fillna("")
29     return df.to_dict('records')
30
31 def formatDataset(file, isCsv = False, toDf = False):
32     """
33     Auxiliary function that formats a dataset and returns it as CSV or Pandas Dataframe
34     """
35     #Convert csv or JSON to dictionary list
36     if isCsv:
37         dictList = csvToDict(file)
38     else:
```

```

37     try:
38         dictList = jsonToDict(file)
39     except:
40         dictList = jsonLineToDict(file)
41
42     #Retrieve question and answer for each dictionary
43     for i in dictList:
44         #If answer is verbalized (between brackets), extract it with a regular expression
45         if "verbalized_answer" in i.keys():
46             answer = re.search(r"\[([^\]]+)\]", i["verbalized_answer"])
47             if answer:
48                 i["verbalized_answer"] = answer.group(1)
49                 i["answer"] = i.pop("verbalized_answer")
50             keysToDelete = set(i.keys()).difference(keysToKeep)
51             for k in keysToDelete:
52                 del i[k]
53
54     #Delete repeated question and answers
55     res = list({frozenset(item.items()) : item for item in dictList}.values())
56     if toDf:
57         return pd.DataFrame(res)
58     return res

```

Tenemos las siguientes funciones:

- **jsonToDict:** Función auxiliar que dado un archivo json, lo abre y lo convierte a lista de diccionarios.
- **jsonLineToDict:** Función auxiliar que dado un archivo json con JSONObjects en cada línea, lo convierte a lista de diccionarios.
- **csvToDict:** Función auxiliar que dado un archivo CSV, lo abre y lo convierte a lista de diccionarios.
- **formatDataset:** Función auxiliar que limpia un conjunto de datos y lo devuelve en formato CSV o Dataframe de Pandas. Convertimos el csv o el JSON en una lista de diccionarios usando las funciones anteriores. Luego, revisamos si la respuesta está verbalizada o no, en cuyo caso la extraemos por medio de una expresión regular que encuentra texto entre corchetes. Finalmente, eliminamos las entradas repetidas convirtiendo la lista de diccionarios en un *set*.

3.5.5. Clasificador de preguntas

La clase *questionClassifier* es la encargada de la clasificación de preguntas en base a su respuesta y la obtención de la fluidez de las preguntas. Tanto las funciones relativas a esta primera tarea (*getAnswerCategory*, *classifyAnswerCategory* y *classifyLiterals*) como el constructor de la clase son una ligera modificación del script *BERTENClassifier.py* del Sistema de Pregunta Respuesta MuHeQa. [19]

Para esta tarea utilizamos el modelo de lenguaje BERT³, al cual se le ha realizado un *fine-tuning*. Esta tarea consiste en emplear un modelo de redes neuronales ya previamente entrenado (en este caso BERT, el modelo de Google especializado en

³BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Desarrollo del Sistema

procesamiento de lenguaje natural en inglés) para, conservando todos sus parámetros iniciales salvo la capa de salida u *output layer*, entrenarlo en una nueva tarea relacionada con la inicial.

Listing 3.26: Constructor de la clase *QuestionClassifier*.

```
1 """
2 Based on @cbadenes' BertENClassifier.py
3 """
4 import csv
5 import json
6 import kenlm
7 import torch
8 import numpy as np
9 from transformers import BertTokenizer, BertForSequenceClassification
10
11 class QuestionClassifier:
12     """
13     Class Constructor: Initializes local files in resources_dir
14     """
15     def __init__(self, resources_dir):
16
17         category_model_dir = resources_dir+'/BERT Fine-Tuning category'
18         literal_model_dir = resources_dir+'/BERT Fine-Tuning literal'
19         resource_model_dir = resources_dir+'/BERT Fine-Tuning resource'
20         mapping_csv = resources_dir+'/mapping.csv'
21         hierarchy_json = resources_dir+'/dbpedia_hierarchy.json'
22
23         self.id_to_label = {}
24         self.label_to_id = {}
25         with open(mapping_csv) as csvfile:
26             reader = csv.reader(csvfile, delimiter=',')
27             for row in reader:
28                 self.id_to_label[row[1]] = row[0]
29                 self.label_to_id[row[0]] = row[1]
30
31         self.category_tokenizer = BertTokenizer.from_pretrained(category_model_dir)
32         self.category_model = BertForSequenceClassification.from_pretrained(category_model_dir,
33                                     num_labels=3)
34
35         self.literal_tokenizer = BertTokenizer.from_pretrained(literal_model_dir)
36         self.literal_model = BertForSequenceClassification.from_pretrained(literal_model_dir,
37                                     num_labels=3)
38
39         self.resource_tokenizer = BertTokenizer.from_pretrained(resource_model_dir)
40         self.resource_model = BertForSequenceClassification.from_pretrained(resource_model_dir,
41                                     num_labels=len(self.id_to_label))
42
43         self.hierarchy = {}
44         with open(hierarchy_json) as json_file:
45             self.hierarchy = json.load(json_file)
46
47         self.fluencyScoreModel = kenlm.Model(resources_dir+"/ZAMIA_Fluency_Score/en_large_model.
48             binary")
```

En el constructor se inicializan los modelos de lenguaje a utilizar, y los csv y json necesarios para su carga.

Listing 3.27: Métodos para la clasificación de preguntas en base a tipo de respuesta

```
1 def getAnswerCategory(self, question):
```

```

2      """
3      Method that gets the answer category of a question.
4      """
5      res = self.classifyAnswerCategory(question)
6      #If the question is a literal, it returns the literal category.
7      if res == 'Literal':
8          return self.classifyLiterals(question)
9      return res
10
11 def classifyAnswerCategory(self,q):
12     """
13     Auxiliary method that gets the answer category of a question.
14     """
15     input_ids = torch.tensor(self.category_tokenizer.encode(q, add_special_tokens=True)).
16         unsqueeze(0) # Batch size 1
17     labels = torch.tensor([1]).unsqueeze(0) # Batch size 1
18
19     with torch.no_grad():
20         outputs = self.category_model(input_ids, labels=labels)
21     logits = outputs[1]
22     result = np.argmax(logits.detach().numpy(),axis=1)[0]
23     if result == 0:
24         categoryLabel = 'Boolean'
25     elif result == 1:
26         categoryLabel = 'Literal'
27     else:
28         categoryLabel = 'String'
29     return categoryLabel
30
31 def classifyLiterals(self,q):
32     """
33     Method that gets the literal category of a question.
34     """
35     input_ids = torch.tensor(self.literal_tokenizer.encode(q, add_special_tokens=True)).
36         unsqueeze(0) # Batch size 1
37     labels = torch.tensor([1]).unsqueeze(0) # Batch size 1
38
39     with torch.no_grad():
40         outputs = self.literal_model(input_ids, labels=labels)
41     logits = outputs[1]
42     result = np.argmax(logits.detach().numpy(),axis=1)[0]
43     if result == 0:
44         categoryLabel = 'Date'
45     elif result == 1:
46         categoryLabel = 'Number'
47     else:
48         categoryLabel = 'String'
49     return categoryLabel

```

Estas funciones, usando PyTorch y Numpy clasifican las preguntas en base a la categoría más probable de su respuesta. Tenemos las siguientes categorías:

- **Boolean:** Preguntas de Verdadero o Falso.
- **String:** Preguntas cuya respuesta es una cadena de caracteres.
- **Date:** Preguntas cuya respuesta es una fecha.
- **Number :** Preguntas cuya respuesta es un número.

Listing 3.28: Método que calcula la fluidez de una pregunta.

Desarrollo del Sistema

```
1 def getFluencyScore(self,question):
2     """
3     Method that gets the fluency score of a question.
4     """
5     return self.fluencyScoreModel.score(question, bos = True, eos = True)
```

Se ha construido un método que calcula la fluidez de la pregunta. Esto se hace usando KenLM ⁴, una librería de C en Python que permite consultar modelos de lenguaje en formato arpa o binario (arpa optimizado). Estos formatos describen probabilidades para textos, de manera que dada cualquier secuencia de N o menos palabras te devuelven la probabilidad de que esta secuencia aparezca en una muestra lo suficientemente grande del lenguaje. [20] Con esta librería, podemos calcular la fluidez haciendo uso del Modelo de Lenguaje de ZAMIA para la lengua inglesa. La puntuación obtenida se corresponde con la probabilidad en log10 de generar la frase con dicho Modelo de Lenguaje, de modo que una puntuación de -20, por ejemplo, significa que el Modelo tiene una probabilidad de 10^{-20} de dar lugar a la oración. Por lo que cuanto menos negativa sea la puntuación de una pregunta, más cercana a 1 será esta probabilidad, y mejor escrita estará la pregunta generalmente.

Cuadro 3.1: Ejemplos de preguntas junto a su puntuación de fluidez.

Pregunta	Fluency Score	Longitud
Who owns GetTV?	-22.4042	3
GetTV is owned by which organisation?	-28.2313	6
What is the capital of India?	-26.6326	6
What is the capital of the Republic of India?	-38.1202	9
Does Messi play for Barcelona?	30.7145	5
Is Lionel Messi a part of F.C. Barcelona's squad?	-51.9804	9

En la Tabla 3.1 podemos ver ejemplos de las mismas preguntas formuladas de distinta manera (una manera corta y otra larga), junto a su longitud y la puntuación de fluidez que le da nuestro sistema. Podemos observar como pese a que ambas instancias de pregunta están bien escritas, nuestro modelo penaliza la longitud incluso aunque esta no sea excesiva.

De este modo, debido al Modelo de Lenguaje empleado (que como hemos visto penaliza las frases en base a su longitud) y a la naturaleza de la puntuación de fluidez (que define una probabilidad y no está estrictamente ligada a la fluidez), es muy complicado señalar un valor de fluidez a partir del cual se pueda decir que una pregunta está escrita de manera fluida. No obstante, sí que se puede señalar que cuanto menos negativo sea el valor de la *fluency score* para una pregunta, mejor escrita estará generalmente.

⁴KenLM - Language Model Toolkit

3.6. Despliegue

Para el despliegue de la herramienta se ha utilizado Docker ⁵, un proyecto de código abierto que automatiza el despliegue de aplicaciones en contenedores de software. En este apartado describiremos el proceso seguido para el despliegue de la aplicación, incluyendo una guía de los pasos previos.

3.6.1. Prerrequisitos

Los siguientes requisitos son necesarios para desplegar la aplicación:

- Tener instalado Docker ⁶ y Docker Compose ⁷
- Crear un Libro de Cálculo en Google Sheets para guardar las validaciones de las respuestas por parte de los usuarios.
- Tener una Cuenta de Servicio (*Service Account*) de Google Cloud, con permisos de editor en este Libro de Cálculo.
- Generar un archivo “credentials.json” con las claves necesarias para realizar la conexión al Libro por medio de las APIs de Google, y guardarlo en “src/utils”.
- Desplegar el Sistema de Pregunta-Respuesta. Si este se despliega junto a la Interfaz, es necesario asegurarse de que la Interfaz no comience a correr antes de que el Sistema esté levantado por completo.
- Definir las variables de entorno necesarias en el archivo “.env”

3.6.2. Registro de una Cuenta de Servicio de Google Cloud

Nota: Las imágenes e instrucciones de esta sección pueden estar desactualizadas, por lo que animamos al lector a revisar la documentación de Google Cloud ⁸ en caso de encontrar algún problema durante la creación de la cuenta.

Una cuenta de servicio es un tipo especial de cuenta que usa una aplicación o carga de trabajo de procesamiento en lugar de una persona. Las aplicaciones usan cuentas de servicio para realizar llamadas a APIs. Estas difieren de las cuentas de usuario en varios aspectos, tales como que están asociadas a pares de claves RSA públicas o privadas que se usan para la autenticación en lugar de contraseñas. [21] En nuestra aplicación usaremos una cuenta de servicio para modificar el Libro de Cálculo en el que guardaremos las valoraciones del usuario.

Primero, iniciamos sesión en Google Cloud Platform y navegamos a la pestaña de Proyectos. Desde ahí, damos click en “CREAR PROYECTO” y creamos un nuevo proyecto con un nombre y bajo una organización a nuestra elección.

⁵Home - Docker

⁶Install Docker Engine

⁷Install Docker Compose

⁸Cuentas de Servicio - Documentación de IAM

Desarrollo del Sistema

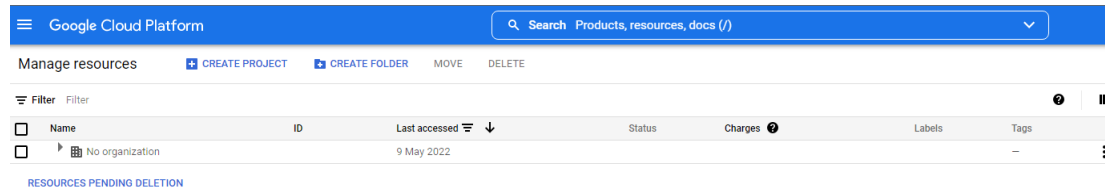


Figura 3.7: Pantalla de Proyectos en Google Cloud Computing

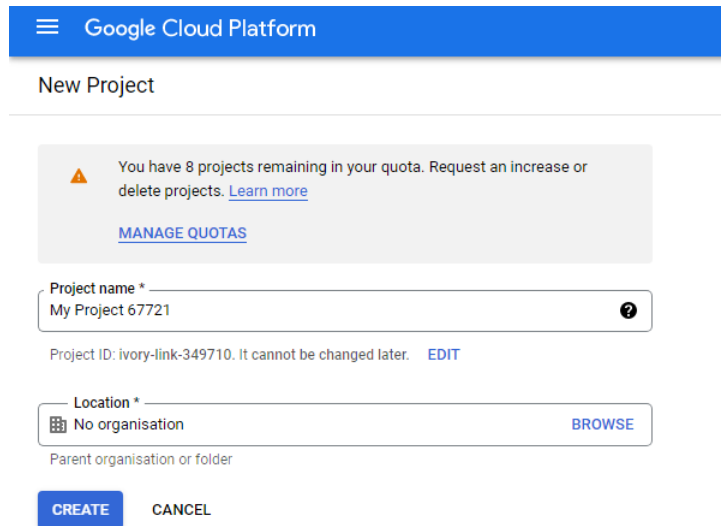


Figura 3.8: Crear un nuevo proyecto

A continuación, será necesario Habilitar la API de Gestión de Identidades y Acceso (IAM) para nuestro nuevo proyecto. Desde esta página, confirmamos el proyecto para el cual habilitaremos la api, pulsamos SIGUIENTE y habilitamos la API con el botón HABILITAR.

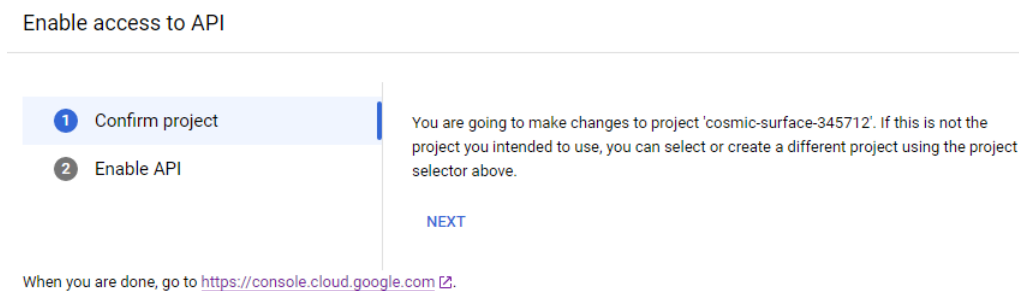


Figura 3.9: Habilitar IAM API - Confirmar proyecto

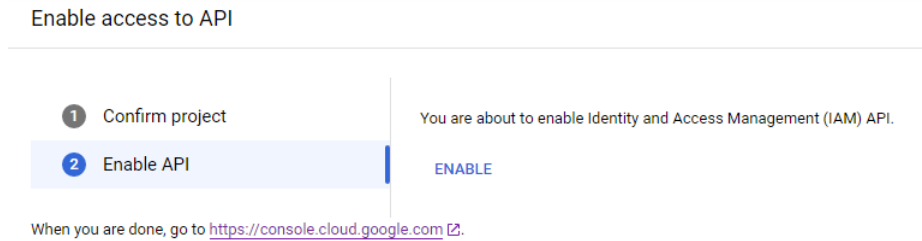


Figura 3.10: Habilitar IAM API - Habilitar

Una vez creado el proyecto y habilitada la API de IAM, también es posible crear la cuenta de servicio con el siguiente script de Python: [22]

Listing 3.29: Código para la creación de una cuenta de servicio desde Python

```
1 import os
2 from google.oauth2 import service_account
3 import googleapiclient.discovery
4
5 def create_service_account(project_id, name, display_name):
6     """Creates a service account."""
7     credentials = service_account.Credentials.from_service_account_file(
8         filename=os.environ['GOOGLE_APPLICATION_CREDENTIALS'],
9         scopes=['https://www.googleapis.com/auth/cloud-platform'])
10    service = googleapiclient.discovery.build(
11        'iam', 'v1', credentials=credentials)
12    my_service_account = service.projects().serviceAccounts().create(
13        name='projects/' + project_id,
14        body={
15            'accountId': name,
16            'serviceAccount': { 'displayName': display_name}
17        }).execute()
18    return my_service_account
```

Para permitir a la cuenta acceder y modificar Libros de Cálculo en Google Sheets necesitaremos habilitar las APIs de Google Drive y Google Sheets en el proyecto. Desde el Panel de Control de APIs de Google Cloud, navegamos con la barra lateral a APIs y Servicios Permitidos y clickamos en + Activar APIs y Servicios.

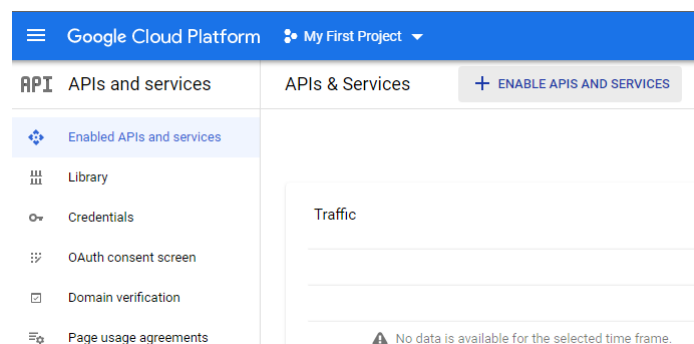


Figura 3.11: Habilitar APIs y Servicios en Google Cloud

Desarrollo del Sistema

Desde la barra de búsqueda en la nueva pantalla que se nos abrirá, buscamos Google Drive API y Google Sheets API y las habilitamos.

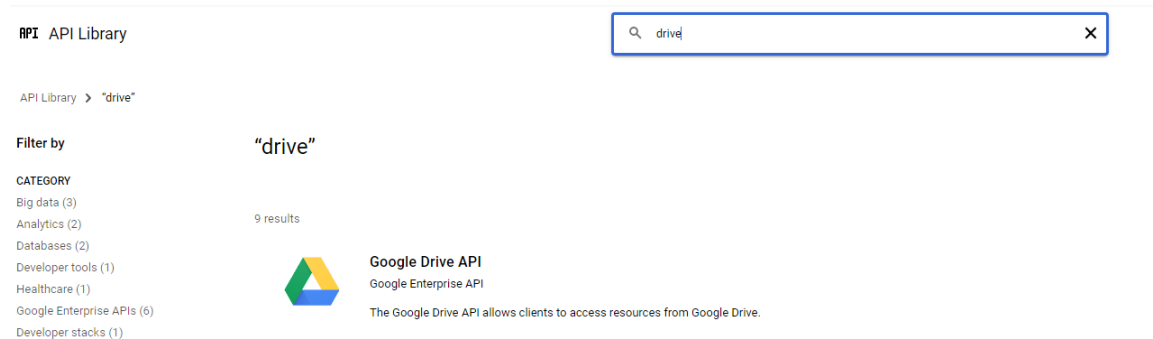


Figura 3.12: Habilitar API Google Drive

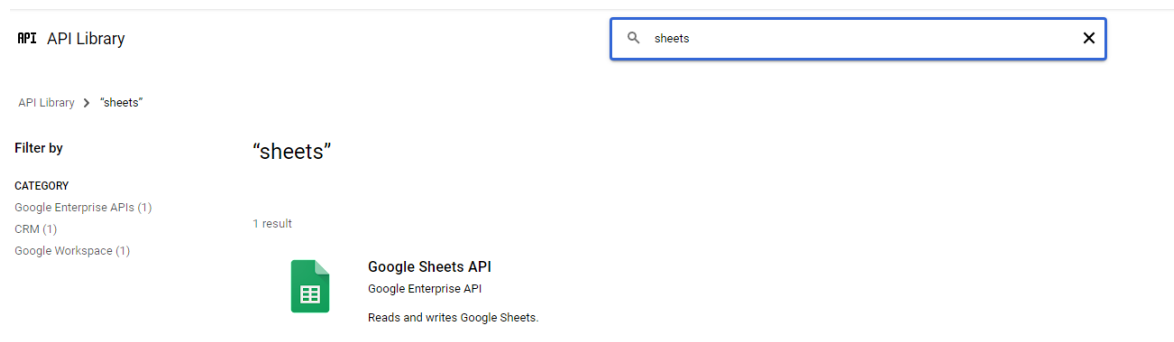


Figura 3.13: Habilitar API Google Sheets

Tras crear la cuenta de servicio y activar las APIs, necesitaremos generar el archivo JSON con sus claves. Para ello, volvemos al Panel de Control de APIs de Google Cloud, navegamos a Credenciales y en el apartado de Cuentas de Servicio pulsamos sobre la cuenta recién creada.

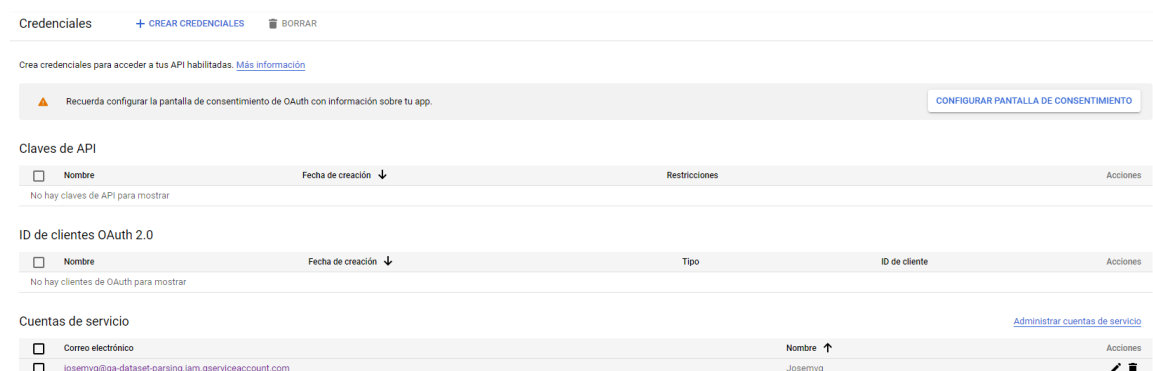


Figura 3.14: Pestaña de Credenciales en Google Cloud

Allí, nos movemos a la pestaña de Claves y creamos una nueva clave pulsando sobre

Agregar Clave. Seleccionamos tipo de clave JSON y pulsamos CREAR.

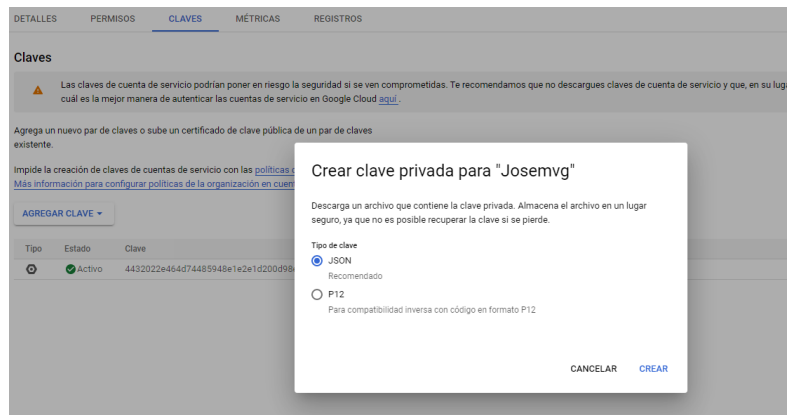


Figura 3.15: Crear Clave privada para la Cuenta de Servicio

Con esto, se nos descargará un JSON con el siguiente formato:

Listing 3.30: JSON de ejemplo con claves para una Cuenta de Servicio de Google

```
1 {
2   "type": "service_account",
3   "project_id": " ",
4   "private_key_id": " ",
5   "private_key": "-----BEGIN PRIVATE KEY-----\n KEY =\n-----END PRIVATE KEY-----\n",
6   "client_email": " ",
7   "client_id": " ",
8   "auth_uri": "https://accounts.google.com/o/oauth2/auth",
9   "token_uri": "https://oauth2.googleapis.com/token",
10  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
11  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/ "
12 }
```

Guardamos el JSON en la carpeta `src/utils` como `credentials.json`. Para terminar, le damos permisos de editor a esta cuenta de servicio en el Libro de Cálculo en el que guardaremos las valoraciones de los usuarios.

3.6.3. Creación de la imagen del Servicio Web

Los comandos necesarios para construir la imagen en Docker de la aplicación web de Streamlit se encuentra en el fichero "Dockerfile.Streamlit":

Listing 3.31: Dockerfile para la creación de la imagen de nuestro servicio web

```
1 FROM python:3.9
2
3 #Create resources_dir
4 RUN apt-get update && apt-get install -y \
5     curl \
6     unzip \
7     tzdata
```

Desarrollo del Sistema

```
8 RUN curl https://delicias.dia.fi.upm.es/nextcloud/index.php/s/Jp5FeoBn57c8k4M/download --output /
  src/Utils/resources_dir.zip --create-dirs
9 RUN unzip /src/Utils/resources_dir.zip -d /src/Utils/
10 RUN curl https://delicias.dia.fi.upm.es/nextcloud/index.php/s/JCarxYnnReHPwBP/download --output /
  src/Utils/resources_dir/ZAMIA_Fluency_Score/en_large_model.binary --create-dirs
11
12 #Install requirements and run the webapp
13 COPY requirements.txt requirements.txt
14 RUN pip install -r requirements.txt
15
16 COPY ./src /src
17 WORKDIR /src
18 EXPOSE 8501
19 ENTRYPOINT ["streamlit", "run", "webapp.py"]
```

Partimos de la imagen `python:3.9`⁹, la versión de Python más estable en la cual se han probado las librerías utilizadas en la Interfaz. Creamos el directorio `resources_dir`, descargando y extrayendo los modelos de lenguaje de clasificación de preguntas y el modelo de ZAMIA para el cálculo de la *fluency score* e instalamos las librerías requeridas, que se encuentran en “`requirements.txt`”. Finalmente, copiamos la carpeta “`/src`”, que contiene el código de la aplicación, y la corremos en el puerto 8501.

Desde una terminal abierta en la carpeta del repositorio, usaremos el comando `docker build` para construir la imagen a partir del archivo “`Dockerfile.Streamlit`” y el directorio actual como contexto:

```
$ docker build --pull --rm -f "Dockerfile.streamlit" -t qa-webapp/streamlit-ui:latest "."
```

Con esto, crearemos la imagen del Servicio Web como “`qa-webapp/streamlit-ui:latest`”.

3.6.4. Definición de las variables de entorno

Las variables de entorno se definen en el archivo `.env` del directorio principal del repositorio, y son las siguientes:

- **EQA_SERVICE_URL:** Dirección URL del Sistema de Pregunta-Respuesta al que se harán las preguntas.
- **EQA_SERVICE_ROUTINGS:** Hay Sistemas de Pregunta y respuesta que tienen varias rutas, lo cual puede ser útil para ofrecer al usuario respuestas provenientes de distintas bases de conocimiento. En caso de que las haya, se definirán por medio de esta variable de entorno, siguiendo la estructura “`routing1,routing2,...`”. En cualquier otro caso, esta variable se puede dejar vacía.
- **WORKSHEET:** Nombre del libro de cálculo.
- **WORKSHEET_ID:** Id del libro de cálculo, se encuentra en la URL del libro después de “`gid=`”
- **SPREADSHEET:** Hoja concreta del libro que vamos a utilizar.

⁹Python - Official Image (Docker)

- **DEFAULT_NUMBER_OF_ANSWERS:** Número de 1 a 10 que define el número de respuestas que mostrará la interfaz por defecto
- **MULTIPLE_ANSWERS_JSON:** Booleano, True o False, que nos define la estructura del JSON que devolverá el Sistema de Pregunta-Respuesta. Si es True, entonces contendrá múltiples respuestas por cada objeto JSON.

3.6.5. Puesta en marcha del contenedor

El despliegue de la Interfaz junto a una instancia de MongoDB se define en el archivo “docker-compose.yml”:

Listing 3.32: Archivo para el despliegue de la interfaz de Streamlit junto a Mongo.

```
1 version: "3.3"
2 services:
3
4   web:
5     image: "qa-webapp/streamlit-ui"
6     ports:
7       - "8501:8501"
8     restart: on-failure
9     depends_on:
10      - mongodb
11     links:
12      - "mongodb:db"
13     env_file:
14      - .env
15
16   mongodb:
17     image: "mongo"
18     ports:
19       - "27017:27017"
20     restart: on-failure
21     volumes:
22      - ./src/utils:/db
```

Este archivo construye una aplicación multi-contenedor, donde cada contenedor definirá un servicio. Tendremos el Servicio Web en el puerto 8501, el cual leerá las variables de entorno del archivo “.env” y estará conectado a la Base de Datos mediante un enlace gracias al cual le será visible con el nombre “mongo”. Por otro lado tendremos la Base de Datos en el puerto 27017, la cual se construirá a partir de la imagen oficial de MongoDB ¹⁰ como un volumen, un mecanismo que permite a los datos guardados en un cierto directorio persistir en sucesivas ejecuciones [23]. Cargaremos los archivos de la base de datos en el directorio “/src/utils/db”.

Inicializaremos la aplicación del siguiente modo:

```
$ docker-compose -f “docker-compose.yml” up
```

¹⁰MongoDB - Official Image (Docker)

Capítulo 4

Guía de Uso

En este Capítulo se describirán los pasos a seguir para usar las distintas funcionalidades de la Interfaz por medio de ejemplos.

4.1. Hacer una pregunta y validar el Sistema de Pregunta-Respuesta

Por defecto el usuario comenzará en el Módulo de Pregunta-Respuesta, desde el cual podrá hacer preguntas al Sistema. No obstante, en caso de estar en otra página, será posible navegar hasta este módulo haciendo uso de la barra de navegación lateral y seleccionando *Question-Answering*, botón que aparece rodeado en la Figura 4.1:

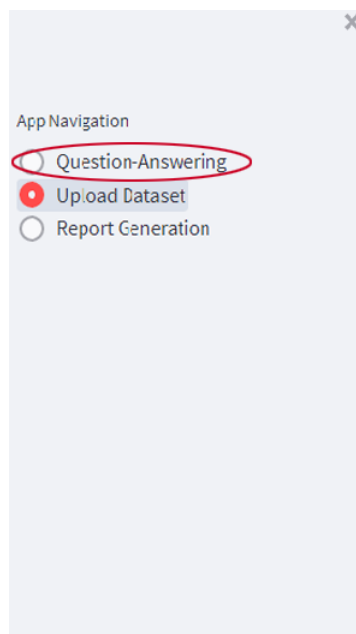


Figura 4.1: Barra de navegación lateral con *Question-Answering* rodeado

4.1. Hacer una pregunta y validar el Sistema de Pregunta-Respuesta

Una vez en la página de Pregunta-Respuestas el usuario podrá hacer una pregunta en lenguaje natural, introducida por teclado, o seleccionar la pregunta de entre uno de los conjuntos de datos de preguntas y respuestas subido previamente a la interfaz web. Para hacer una pregunta libre bastará con escribirla en la barra de búsqueda y pulsar Intro. Por contra, para formular una pregunta de un conjunto de datos habrá que seleccionar un *dataset*, en cuyo caso obtendremos una pregunta aleatoria de este conjunto de datos, o dejar el selector con la opción *All* para recibir una pregunta de cualquier conjunto de datos; y luego pulsar el botón *Make a Random Question* (Figura 4.2):



Figura 4.2: Barra de búsqueda y selector para realizar una pregunta

Al hacer una pregunta saldrá un mensaje de carga junto a la pregunta realizada. En caso de que la pregunta provenga de un conjunto de datos también se mostrará como *Expected Answer* la respuesta esperada, es decir, la respuesta que aparece en el conjunto de datos (Figura 4.3).

Question: Where is the tombstone of the parliament members who served with James Roberts as the Vice President, US?

Expected Answer: Christchurch



  Looking for answers...

Figura 4.3: Mensaje de carga al realizar una pregunta

Después de obtener todas las respuestas del Sistema de Pregunta-Respuesta estas se mostrarán ordenadas por confianza (que mide la seguridad que el sistema le da a cada respuesta de ser correcta) y desde el deslizador en la barra lateral el usuario podrá seleccionar cuántas respuestas quiere ver. Esto se podrá hacer tanto antes como después de que se presenten las respuestas.

Debajo tendremos los dos botones para evaluar las respuestas, un botón con un emoticono de un pulgar hacia arriba para indicar que son correctas y otro con un pulgar hacia abajo para indicar que son incorrectas, como podemos observar en Figura 4.4

Guía de Uso

que muestra múltiples respuestas en nuestra interfaz junto a estos dos botones.

Question: Where was Fernando Alonso born?

... The Racing-Reference driver ID of Fernando Alonso is Fernando_Alonso. The RERO ID of Fernando Alonso is 02-A028187874. The Encyclopædia Universalis ID of Fernando Alonso is fernando-alonso. The Treccani ID of Fernando Alonso is fernando-alonso. The sibling of Fernando Alonso is Lorena Alonso. The Quora topic ID of Fernando Alonso is Fernando-Alonso. The AS. com athlete ID of Fernando Alonso is fernando_alonso/24337. The image of Fernando Alonso is http://commons.wikimedia.org/wiki/Special:FilePath/Alonso%202016.jpg. The place of birth of Fernando Alonso is Oviedo ANSWER. The sex or gender of Fernando Alonso is male. The father of Fernando Alonso is José Luis Alonso. The spouse of Fernando Alonso is Raquel del Rosario. The country of citizenship of Fernando Alonso is Spain. The instance of of Fernando Alonso is human. The position held of Fernando Alonso is UNICEF Goodwill Ambassador. The member of sports team of Fernando Alonso is Minardi, Scuderia Ferrari, McLaren, Renault F1 Team, Alpine F1 Team...

Answer: Oviedo

Relevance: 0.9756 **Source:** wikidata

... The nationality of Fernando Alonso is Spanish. The image size of Fernando Alonso is 240. The last position of Fernando Alonso is 11. The birth place of Fernando Alonso is Oviedo, Asturias, Spain ANSWER. The quote of Fernando Alonso is The essence of his qualities is that he is very complete. You struggle to find a weak point, basically, in terms of high-level skills. The technical preparation in terms of driving. The ability to cope with a variety of situations. Intelligence - just [the] capacity to understand the situation while he is in or out of the car. Commitment. . . it is so difficult for Fernando to accept he is slower than someone else. It is very essential to his nature[,] which potentially might have created problems when he was not mature enough to manage this fundamental aspect of his identity. . . I have seen this with Michael [Schumacher]. . . This aspect of Fernando is certainly not less than Michael, but it expresses itself in different ways...

Answer: Oviedo, Asturias, Spain

Relevance: 0.8034 **Source:** dbpedia

...The copyright holder for this preprint this version posted January 13, 2022. ; https://doi.org/10.1101/2022.01.10. As such, whenever possible, disaggregating by place of birth within specific Latine heritage groups (e.g., US-born Mexican, foreign-born Mexican) may be advisable, and when not possible to divide by heritage groups, expanding nativity to three categories for Latines (i.e., US-born [50 states/DC], territory-born [e.g., Puerto Rico], and foreignborn) may be another option for consideration. Admittedly, these categories are also limited in their portrayal of the multidimensional cultures they label. Nonetheless, they represent a move toward data disaggregation to more effectively promote population health. where DE FB,g,i is the demographic estimate of native-born young children for group g (defined by race/ethnic group) and single year of age i, and rate g is the chosen annual emigration rate. 1. Parent carers were asked about their child's birth order: 'what is your child's birth order?' and selected from one of the following options: first-born, second-born, third-born, fourthborn, fifth-born, or sixth-born ANSWER or later. The responses were recoded to create a binary variable indicating whether the young person was first born (0 = no, 1 = yes). The median size of foreign-born workers' households was four persons, and that of U...

Answer: first-born, second-born, third-born, fourthborn, fifth-born, or sixth-born

Relevance: 0.6096 **Source:** cord19

Please rate if our answer has been helpful to you so we can further improve our system!

Figura 4.4: Tres respuestas a la pregunta *Where was Fernando Alonso born?* y boto- nes para validación de respuestas abajo

API JSON Response

```
{
  "0": {
    "answer": "Oviedo"
    "confidence": 0.9756
    "evidence": {
      "end": 590
      "start": 584
      "summary":
        " The Racing-Reference driver ID of Fernando Alonso is Fernando_Alonso. The RERO ID of Fernando Alonso is 02-A028187874. The Encyclopædia Universalis ID of Fernando Alonso is fernando-alonso. The Treccani ID of Fernando Alonso is fernando-alonso. The sibling of Fernando Alonso is Lorena Alonso. The Quora topic ID of Fernando Alonso is Fernando-Alonso. The AS. com athlete ID of Fernando Alonso is fernando_alonso/24337. The image of Fernando Alonso is http://commons.wikimedia.org/wiki/Special:FilePath/Alonso%202016.jpg. The place of birth of Fernando Alonso is Oviedo. The sex or gender of Fernando Alonso is male. The father of Fernando Alonso is José Luis Alonso. The spouse of Fernando Alonso is Raquel del Rosario. The country of citizenship of Fernando Alonso is Spain. The instance of of Fernando Alonso is human. The position held of Fernando Alonso is UNICEF Goodwill Ambassador. The member of sports team of Fernando Alonso is Minardi, Scuderia Ferrari, McLaren, Renault F1 Team, Alpine F1 Team"
    }
    "question": "where was Fernando Alonso born?"
    "source": "wikidata"
  },
  "1": {
    "answer": "Oviedo, Asturias, Spain"
    "confidence": 0.8034
    "evidence": {
      "end": 201
      "start": 178
      "summary":
        " The nationality of Fernando Alonso is Spanish. The image size of Fernando Alonso is 240. The last position of Fernando Alonso is 11. The birth place of Fernando Alonso is Oviedo, Asturias, Spain. The quote of Fernando Alonso is The essence of his qualities is that he is very complete. You struggle to find a weak point, basically, in terms of high-level skills. The technical preparation in terms of driving. The ability to cope with a variety of situations. Intelligence - just [the] capacity to understand the situation while he is in or out of the car. Commitment. . . it is so difficult for Fernando to accept he is slower than someone else. It is very essential to his nature[,] which potentially might have created problems when he was not mature enough to manage this fundamental aspect of his identity. . . I have seen this with Michael [Schumacher]. . . This aspect of Fernando is certainly not less than Michael, but it expresses itself in different ways"
    }
  }
}
```

Figura 4.5: Respuesta JSON para la pregunta *Where was Fernando Alonso born?*

Después de presentar al usuario las respuestas en la interfaz, aparecerá en la barra de navegación una casilla que nos dará la opción de mostrar la respuesta JSON construida por el Sistema de Pregunta-Respuesta (ver Figura 3.5). Al marcar esta casilla la interfaz imprimirá el JSON bajo el texto como se muestra en la Figura 4.5

4.2. Subir un conjunto de datos


Para subir conjuntos de datos, nos moveremos al Módulo de Gestión de Conjuntos de Datos por medio de la barra de navegación lateral, pulsando *Upload Dataset*. Al llegar a esta pantalla, se nos presentarán las instrucciones sobre los formatos permitidos para estos conjuntos de datos y un buffer con el cual subir los archivos.

Dataset Management

You may upload your dataset below. For it to be processed and uploaded to our database, please follow these guidelines:

- 1. Upload your dataset either on .CSV or .JSON format.
- 2. JSONs may be on JSON lines or JSON array format.
- 3. Answers should be on the "answer" column/key, and Questions on the "question" column/key.
- 4. If your Answer is verbalized, you shall name its key/column "verbalized_answer", and format it with the answer between brackets, i.e. "Fernando Alonso was born in [Oviedo]."

Upload a Dataset

 Drag and drop file here
Limit 200MB per file • CSV, JSON

Browse files

Figura 4.6: Texto con instrucciones y buffer para subir conjuntos de datos

Para introducir un *dataset* en la base de datos haremos click en *Browse Files*, lo cual abrirá una ventana del explorador de archivos que le permitirá al usuario subir archivos en formato JSON o CSV que tenga en su propio ordenador.

Tras subir el archivo, este se procesará y se intentará subir al MongoDB como una nueva colección de archivos. El usuario podrá ver un mensaje de éxito en caso de que el conjunto de datos se suba correctamente (Figura 3.6) o un mensaje de excepción que especificará si se ha producido un error durante el procesado del conjunto de datos (lo cual significará que el conjunto de datos no está en un formato adecuado) o si se ha producido un error al introducir el conjunto de datos en la base de datos.

4.3. Generar un informe sobre un conjunto de datos

De forma análoga a las otras páginas, nos movemos hasta este Módulo pulsando *Report Generation* en la barra de navegación lateral. En esta página se nos presentará un selector para escoger un conjunto de datos sobre el cual realizar un reporte de calidad, como podemos observar en la Figura 4.7. Sin embargo, si la base de datos se encuentra vacía el selector se bloqueará y se imprimirá un mensaje que indica que no hay conjuntos de datos disponibles, instando al usuario a subir uno desde el Módulo de Gestión de Conjuntos de Datos (Figura 4.8).

Para generar un informe el usuario deberá seleccionar un *dataset* y pulsar *Run*.

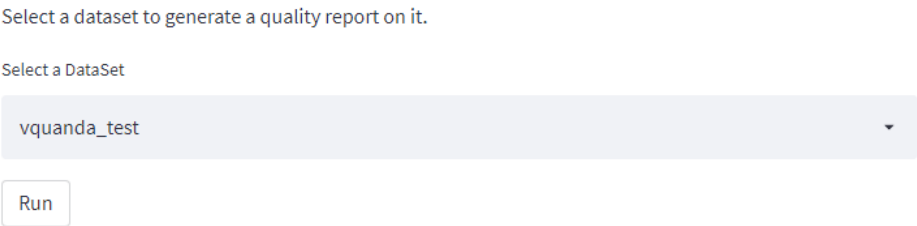


Figura 4.7: Selector en Módulo de Generación de Reportes

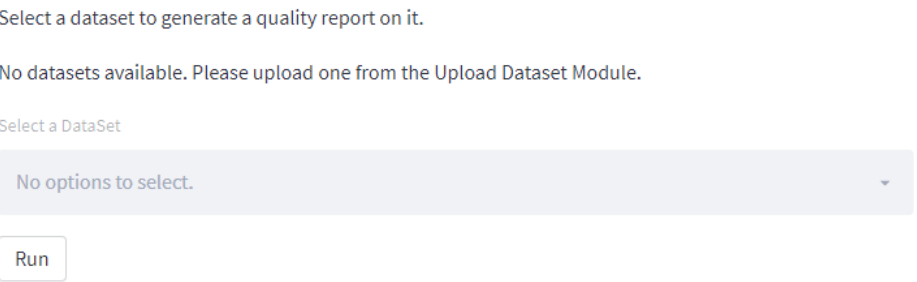


Figura 4.8: Selector sin conjuntos de datos en Módulo de Generación de Reportes

Tras pulsar *Run*, saldrá un *spinner loader* con un mensaje de carga animado análogo al mostrado en el Módulo de Pregunta-Respuesta (Figura 4.3) para indicar el usuario que el contenido está cargando. Dado que para generar el informe se hacen uso de varios Modelos de Lenguaje, los cuales son llamados para cada una de las preguntas del conjunto de datos, esta tarea puede tardar varios minutos.

Lo primero que se le presenta al usuario al terminar la generación del informe es el *dataframe* que contiene las preguntas, respuestas, tipo de la pregunta y fluidez:

Quality Report

Dataset Preview

	question	answer	answerType	fluencyScore
0	Count the number of peop...	8	Number	-59.7859
1	Which location city of Den...	Colorado, Denver	String	-83.5485
2	Which people are known f...	The people known for Dra...	String	-46.2257
3	Which office holder's gove...	William Henry Harrison	String	-83.1327
4	List all the faiths that Britis...	Anglican, Anglicanism, Cat...	String	-50.0965
5	In which cities can the bev...	Jasper, Indiana, Indiana	String	-51.2676
6	Trainees at the national fil...	Poland, Szczecin, Order of ...	String	-47.2164
7	Give me the count of all pe...	65	Number	-47.1849
8	What is the region of Kim S...	North America, Canada	String	-40.2483
9	I list the total number of ex	173	Number	-84.8727

Figura 4.9: Ejemplo de *dataframe* presentado al usuario en un informe.

4.3. Generar un informe sobre un conjunto de datos

En la Figura 4.9 tenemos un ejemplo de una de estas tablas. Streamlit, al ser estar pensado para aplicaciones de Ciencia de Datos, ofrece diversas opciones para *dataframes* en su interfaz. El usuario podrá visualizar todas las filas de la tabla haciendo *scroll* vertical mediante la barra rodeada en azul en esta figura, o abrir la tabla a pantalla completa pulsando en el icono señalado en rojo. Otra funcionalidad de Streamlit es la posibilidad de cambiar el orden en el que se muestran las filas de la tabla dando click en una columna, lo cual hará que las filas se ordenen en base a esta. De esta forma podemos ordenar las filas en base a diversos criterios, como de mayor a menor fluidez (Figura 4.10), por tipo de respuesta o por orden alfabético de la pregunta o la respuesta.

	question	answer	answerType	fluencyScore ▾
379	GetTV is owned by which ...	Sony	String	-28.2313
308	What organisations purp...	Greenpeace, Internationa...	String	-28.9854
429	Which rivers start in Wyo...	The rivers whose source r...	String	-29.1258
387	Which person's successor...	Hammurabi	String	-29.2124
419	Is morrissey in the smiths?	Yes	Boolean	-29.2953
162	Semani languages are sp...	Iran	String	-29.7865
972	Who owns the airport in P...	Cyprus	String	-29.8476
64	Who are the key people o...	The key people of FWD.us...	String	-29.9382
112	What is the parent compa...	Facebook	String	-29.9885
697	Who designed the bridge ...	Sri Lanka, British Ceylon	String	-30.0610
191	Who gives the license of ...	The licensees of WXXV-DT...	String	-30.0762
776	What municipalities are a...	Geneva, Vandœuvres, Col...	String	-30.2673
835	What are some orthonych...	Logrunners, Chowchilla, ...	String	-30.2863
810	Which rivers flow into the ...	Hudson River	String	-30.3012
60	How many organizations ...	18	Number	-30.3523
193	Where did john o conner ...	Villanova University, St. C...	String	-30.4435
706	Where did ed podolak go ...	Atlantic Ocean	String	-30.6945
200	Was Castillo discovered b...	Yes	Boolean	-30.8482
670	Was james watt a mechan...	Yes	Boolean	-31.1239
494	How many fictional chara...	102	Number	-31.2258
342	Katharevousa writers hav...	Nobel Prize	String	-31.2877
608	Where are renaissance ce...	400, Detroit, Michigan	String	-31.5745
786	What sports are played in ...	Association football, Hoc...	String	-31.6942
623	Name the origin of Henry ...	Belfast, Northern Ireland	String	-31.7963
92	What is the political party...	The political parties of th...	String	-32.0265
170	Who manufactures the S...	Yamaha Motor Company, ...	String	-32.4500
10	Who all were venerated in...	75	String	-32.4542
767	Who is married to Tony R...	Jessica Simpson	String	-32.5834
343	List the school of Bobby S...	The schools of Bobby Ski...	String	-32.6076
762	Which countries led by Eli...	answer	String	-32.6090

Figura 4.10: *Dataframe* en pantalla completa. Filas ordenadas por *fluency score*.

Al usuario también se le mostrarán una serie de estadísticas sobre la fluidez con la que están escritas las preguntas del conjunto de datos. En la página se podrá visualizar la fluidez media, así como la mediana, el máximo, el mínimo y la desviación estándar de

esta métrica (Figura 4.11).

Fluency Score

The number of distinct questions in the dataset is 1000

The average fluency score is -53.19 and its median is -50.88

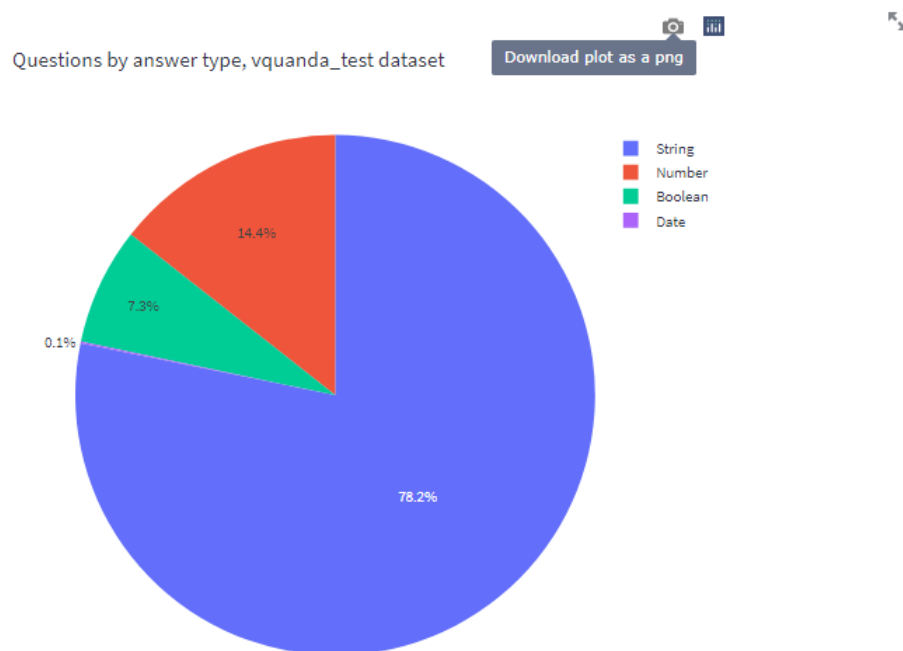
Maximum and minimum fluency scores are -22.90 and -112.22 respectively. Standard deviation is 14.36

Figura 4.11: Estadísticas sobre la fluidez de las preguntas mostradas en la interfaz.

Por último, el reporte mostrará los tipos de pregunta según respuesta que se encuentran en el conjunto de datos y los mostrará en un gráfico circular. Bajo él tendremos un enlace para descargar el reporte en formato PDF.

Answer Types

There are 4 different types of answer in this dataset, which are: Number, String, Boolean, Date



[Download file](#)

Figura 4.12: Apartado de tipo de preguntas y enlace para descarga del reporte.

Tal y como explicamos en el Capítulo 3.4, gracias a la librería Plotly en conjunción con Streamlit podemos realizar múltiples operaciones sobre la gráfica. En primer lugar, como se ve en la Figura 4.12, podemos descargar la gráfica como png pulsando sobre

4.3. Generar un informe sobre un conjunto de datos

el icono de la cámara fotográfica de la esquina superior derecha de la gráfica, tras lo cual nos saldrán una ventana emergente de carga y posteriormente otra que nos notificará cuando se haya terminado de guardar la imagen (Figura 4.3).

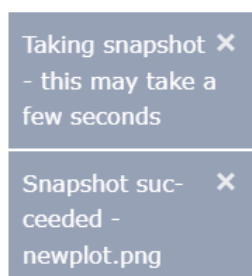


Figura 4.13: Apartado de tipo de preguntas y enlace para descarga del reporte.

A la derecha de la cámara hay un icono que nos redirigirá a la página oficial de Plotly, y aún más a la derecha tenemos un icono con el que podremos abrir la gráfica en pantalla completa de manera análoga a como se hace con los *dataframes* de Pandas.

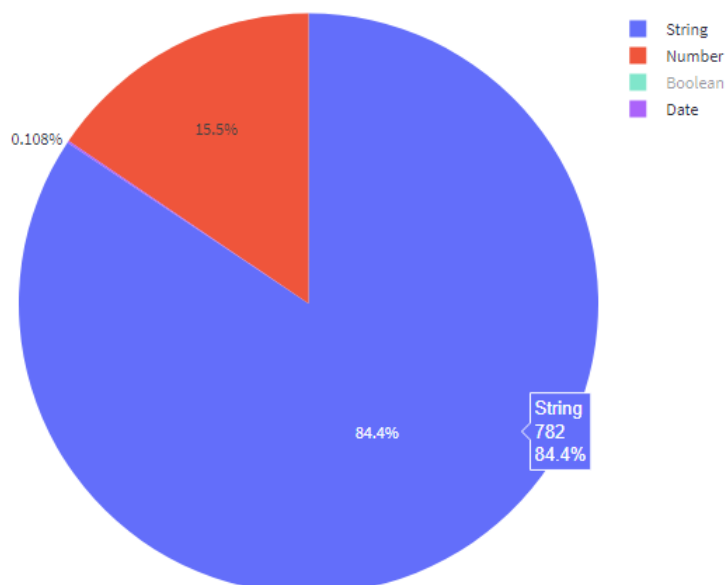


Figura 4.14: Gráfica sin preguntas booleanas.

La principal característica de Plotly es la interactividad de sus gráficas, frente a otras librerías de visualización de datos como Matplotlib o Seaborn cuyas gráficas son estáticas. Como se puede apreciar en 4.3, Plotly muestra datos sobre un sector en concreto al pasar el ratón por encima de este (número de preguntas y porcentaje sobre el total) y además permite seleccionar y deseleccionar los tipos de pregunta que figurarán en la gráfica pulsando sobre ellos en la leyenda de la esquina superior derecha. Por ejemplo, en esta gráfica se ha prescindido de las preguntas de tipo booleano.

Capítulo 5

Resultados y Conclusiones

5.1. Resultados

En este proyecto hemos creado una interfaz web que permite realizar preguntas en lenguaje natural y obtener respuestas también en lenguaje natural. Esto se hace intercambiando mensajes en formato JSON (via peticiones HTTP) con sistemas de pregunta-respuesta con acceso API REST. Además, mediante la definicion de variables de entorno de Docker durante el despliegue, se puede ajustar el servicio a cualquier sistema de pregunta-respuesta. Esto nos permite consultar tanto fuentes de datos estructuradas como no estructuradas de manera visual y sencilla, aumentando sustancialmente la accesibilidad de los sistemas de QA en el proceso.

Una vez recibidas las respuestas del Sistema, el usuario puede validar si son correctas, y este *input* del usuario se registrará, generando automáticamente nuevos conjuntos de datos anotados en formato CSV. También podremos obtener métricas sobre la fluidez de escritura y los tipos de preguntas de los conjuntos de datos subidos a la interfaz. De este modo, evaluamos tanto conjuntos de datos como sistemas de pregunta-respuesta ya existentes, y generamos conjuntos de datos nuevos.

5.2. Líneas Futuras de Trabajo

Debido a la importancia que se le ha otorgado al desarrollo de las funcionalidades de la página frente a la estética de esta, su apartado visual puede resultar algo básico. Si bien la librería Streamlit permite utilizar CSS y definir fondos de página y plantillas propias, [24] creemos que la solución más apropiada (en caso de haber contado con un tiempo mayor para el desarrollo del proyecto) hubiese sido utilizar otras tecnologías tales como Flask en combinación con Bootstrap ¹ para la presentacion visual. Otra solución sería el marco de trabajo PyScript ², que permite ejecutar scripts de Python en HTML. No obstante, este se encuentra en sus versiones preliminares ya

¹Bootstrap - The most pupular HTML, CSS and JS library

²Pyscript

que fue presentado por Anaconda en la Pycon 2022 (mayo de 2022), por lo que podría presentar incompatibilidades con nuestra interfaz. [25]

5.3. Conclusiones

Se ha conseguido desarrollar un servicio que cumple con los objetivos expuestos en el Capítulo 1.2. Para ello se han explorado tecnologías que no son habitualmente tratadas durante los estudios de Grado, como las bases de datos no relacionales o el despliegue de aplicaciones. También se han estudiado campos como la Inteligencia Artificial o el Procesamiento de Lenguaje Natural, revisando modelos de lenguaje y sistemas de pregunta-respuesta con diversas arquitecturas.

Desde un punto de vista personal, este proyecto ha resultado profundamente enriquecedor. El Practicum y posterior Trabajo de Fin de Grado me han permitido conocer la metodología de trabajo de uno de los grupos de investigación de la UPM, así como profundizar en varios de los temas tratados en este como IA, PLN o Web Semántica. Esto se ha hecho por medio no solo de este proyecto, sino también por la interacción con investigadores y la asistencia a eventos formativos de este grupo de investigación. Además, considero que este trabajo se ha beneficiado ampliamente de haberse realizado como continuación de unas prácticas relacionadas con el tema del *Question Answering*, lo cual le ha permitido alcanzar una mayor madurez conceptual.

Capítulo 6

Análisis de impacto

Este trabajo está enmarcado dentro del proyecto DRUGS4COVID ¹, un proyecto europeo que utiliza servicios basados en Inteligencia Artificial para proveer conocimiento sobre tratamientos para el COVID-19. Concretamente, nuestra interfaz fue ideada para el sistema de pregunta-respuesta MuHeQA ².

MuHeQA consulta fuentes de datos estructuradas y no estructuradas, entre las que se encuentra CORD-19 ³. Este es un conjunto de datos abierto que contiene recursos sobre artículos científicos acerca del coronavirus. Nuestro trabajo facilitará el acceso a MuHeQA, que previamente solo era accesible mediante peticiones HTTP, por medio de una página web. Así, se habilitará la consulta de información sobre el coronavirus a cualquier usuario independientemente de su formación en informática.

Este proyecto será enviado como Demo-Paper para la International Semantic Web Conference 2022 (ISWC 22) ⁴, el principal foro internacional para las comunidades de Web Semántica, Datos Enlazados y Grafos de Conocimiento. Además, como parte de nuestro compromiso con la ciencia abierta, el código del servicio está publicado en abierto bajo el repositorio de GitHub <https://github.com/Drugs4CoVid/QA-Webapp>. Por lo tanto este trabajo beneficia también a la comunidad científica, ya que puede proporcionar acceso no solo a MuHeQA sino a cualquier otro sistema de QA en caso de configurar el servicio para ello.

En términos de Objetivos de Desarrollo Sostenible, este trabajo se alinea con los Objetivos 3 (Garantizar una vida sana y promover el bienestar para todos en todas las edades) y 4 (Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos). Más concretamente, los objetivos abordados son los siguientes:

- **Meta 3.b:** Apoyar las actividades de investigación y desarrollo de vacunas y medicamentos para las enfermedades transmisibles y no transmisibles que afectan

¹Drugs4Covid [26]

²MuHeQA: Question Answering over Multiple and Heterogeneous Knowledge Bases [27]

³CORD-19: The Covid-19 Open Research Dataset [28]

⁴ISWC 2022

primordialmente a los países en desarrollo y facilitar el acceso a medicamentos y vacunas esenciales asequibles de conformidad con la Declaración de Doha relativa al Acuerdo sobre los ADPIC y la Salud Pública, en la que se afirma el derecho de los países en desarrollo a utilizar al máximo las disposiciones del Acuerdo sobre los Aspectos de los Derechos de Propiedad Intelectual Relacionados con el Comercio en lo relativo a la flexibilidad para proteger la salud pública y, en particular, proporcionar acceso a los medicamentos para todos. [29]

- **Meta 3.c:** Aumentar sustancialmente la financiación de la salud y la contratación, el desarrollo, la capacitación y la retención del personal sanitario en los países en desarrollo, especialmente en los países menos adelantados y los pequeños Estados insulares en desarrollo. [29]
- **Meta 4.3:** De aquí a 2030, asegurar el acceso igualitario de todos los hombres y las mujeres a una formación técnica, profesional y superior de calidad, incluida la enseñanza universitaria. [30]
- **Meta 4.4:** De aquí a 2030, aumentar considerablemente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento. [30]

Bibliografía

- [1] K. Cai. (2022) The \$2 Billion Emoji: Hugging Face Wants To Be Launchpad For A Machine Learning Revolution - Forbes. [Online]. Available: <https://www.forbes.com/sites/kenrickcai/2022/05/09/the-2-billion-emoji-hugging-face-wants-to-be-launchpad-for-a-machine-learning-revolution/?sh=70ae25fdf732>
- [2] S. Rohit, Kumar. (2021) How to Train A Question-Answering Machine Learning Model (BERT). [Online]. Available: <https://blog.paperspace.com/how-to-train-question-answering-machine-learning-models/>
- [3] H. Face. (2022) Models - Hugging Face. [Online]. Available: https://huggingface.co/models?pipeline_tag=question-answering
- [4] Webtematica. (2022) Web semántica, qué es y cómo mejora tu web. [Online]. Available: <https://webtematica.com/que-es-la-web-semantica>
- [5] E. Lisa and W. Wolfram. (2016) Towards a Definition of Knowledge Graphs. [Online]. Available: <http://ceur-ws.org/Vol-1695/paper4.pdf>
- [6] A. Singhal. (2012) Introducing the Knowledge Graph: things, not strings. [Online]. Available: <https://blog.google/products/search/introducing-knowledge-graph-things-not/>
- [7] P. Sukannya, D. Saswati, B. G P Shrivatsa, K. Dinesh, and G. Dinesh. (2020) Entity and Relation Linking is All You Need for KGQA. [Online]. Available: <https://research.ibm.com/publications/entity-and-relation-linking-is-all-you-need-for-kgqa>
- [8] D. Dennis, L. Vanessa, S. Kamal, and P. Maret. (2017) Core techniques of question answering systems over Knowledge Bases: a survey (Knowledge and Information Systems 2017). [Online]. Available: <https://link.springer.com/article/10.1007/s10115-017-1100-y>
- [9] E. Kacupaj, H. Zafar, J. Lehmann, and M. Maleshkova, "VQuAnDa: Verbalization QUEStion ANSwering DATaset," in *The Semantic Web*. Springer International Publishing, 2020, pp. 531–547.

-
- [10] B. Debanjali, D. Mohnish, A. H. Md Rashad, and L. Jens. (2021) VANiLLa : Verbalized Answers in Natural Language at Large Scale. [Online]. Available: <https://arxiv.org/abs/2105.11407>
- [11] D. Mohnish, B. Debayan, A. Abdelrahman, and L. Jens, "LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia," in *The Semantic Web*. Springer International Publishing, 2019, pp. 220–230.
- [12] N. for Question Answering | CFF. (2020) Evaluating QA: Metrics, Predictions, and the Null Response. [Online]. Available: https://qa.fastforwardlabs.com/no%20answer/null%20threshold/bert/distilbert/exact%20match/f1/robust%20predictions/2020/06/09/Evaluating_BERT_on_SQuAD.html#Exact-Match
- [13] P. Kishore, R. Salim, W. Todd, and Z. Wei-Jing. (2002) BLEU: a Method for Automatic Evaluation of Machine Translation. [Online]. Available: <https://aclanthology.org/P02-1040.pdf>
- [14] A. Lavie and A. Agarwal, "METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments," in *Proceedings of the Second Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 07 2007, pp. 228–231.
- [15] N. N. I. of Standard and Technologies. (2019) Levenshtein Distance. [Online]. Available: <https://xlinux.nist.gov/dads/HTML/Levenshtein.html>
- [16] MongoDB. (2022) JSON and BSON | MongoDB. [Online]. Available: <https://www.mongodb.com/json-and-bson>
- [17] —. (2022) Databases & Collections - MongoDB Manual. [Online]. Available: <https://www.mongodb.com/docs/manual/core/databases-and-collections/>
- [18] Streamlit. (2022) Streamlit The Fastest way to build and share data apps. [Online]. Available: <https://streamlit.io/>
- [19] Carlos Badenes-Olmedo. (2022) GitHub - MuHeQa/BertEnClassifier.py. [Online]. Available: <https://github.com/librairy/MuHeQA/blob/main/application/response/BertENClassifier.py>
- [20] CMUSphinx. (2022) CMUSphinx - ARPA Language models. [Online]. Available: <https://cmusphinx.github.io/wiki/arpaformat/>
- [21] Google Cloud. (2022) Cuentas de Servicio | Documentación de IAM. [Online]. Available: <https://cloud.google.com/iam/docs/service-accounts>
- [22] —. (2022) Crea y administra Cuentas de Servicio | Documentación de IAM. [Online]. Available: <https://cloud.google.com/iam/docs/creating-managing-service-accounts#iam-service-accounts-create-python>
- [23] Docker. (2022) Use volumes | Docker Documentation. [Online]. Available: <https://docs.docker.com/storage/volumes/>

- [24] Tijana Nikolic. (2022) Sogeti creates an educational Streamlit app for data preprocessing. [Online]. Available: <https://blog.streamlit.io/sogeti-creates-an-educational-streamlit-app-for-data-preprocessing/>
- [25] Peter Wang. (2022) PyScript - Programming for Everyone: 2022 PyCon Keynote. [Online]. Available: <https://anaconda.cloud/pyscript-pycon2022-peter-wang-keynote>
- [26] Ontology Engineering Group. (2019) DRUGS4COVID. [Online]. Available: <https://drugs4covid.oeg.fi.upm.es/lang/ES>
- [27] Carlos Badenes-Olmedo. (2021) MuHeQA - Question Answering over Multiple and Heterogeneous Knowledge Bases. [Online]. Available: <https://github.com/librairy/MuHeQA>
- [28] Lu Wang, Lucy and Kyle, Lo and Yoganand, Chandrasekhar and Russell, Reas et al. (2021) CORD-19: The Covid-19 Open Research Dataset. [Online]. Available: <https://arxiv.org/abs/2004.10706v2>
- [29] Organizacion de las Naciones Unidas. (2022) Salud - Desarrollo Sostenible. [Online]. Available: <https://www.un.org/sustainabledevelopment/es/health/>
- [30] Educación - Desarrollo Sostenible. (2022) CORD-19: The Covid-19 Open Research Dataset. [Online]. Available: <https://www.un.org/sustainabledevelopment/es/education/>