Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Automatic generation of multiple-choice questions using knowledge graphs

Bachelor's Degree in Software Engineering Final Project

June, 2020

Director: Jose Emilio Labra Gayo
Author: Emilio Cortina Labra

Table of contents

Table of figures

Index of tables

1. Introduction

1.1 Motivation

Web Semantics Oviedo (WESO) is a research group at the University of Oviedo whose main research activity involves semantic Web. Their projects empower the use of semantic technologies and web standards.

The software company Zapiens offers a mobile application that allows other companies to train their employees with quick multiple-choice questions.
This learning method has proven to be effective for employers to teach their teams and improve their knowledge about new products and services.

However, it has a drawback. Questions, along with their answers and distractors, must be curated and manually introduced into the system by a responsible from the company. This is what lead Zapiens to look for a software that automates this process.

The project developed consists on a REST API that shall provide basic functionality to generate multiple choice questions, using Wikidata knowledge graph as the source of information.

1.1.1 Linked Data

Linked Data is structured data which is interlinked with other data.

Linked Data can be expressed as semantic triples, or simply triple. As its name indicates, a triple is a set of three entities that codifies a statement about semantic data in the form of subject–predicate–object expressions (e.g. "Bob is 35", or "Bob knows John").

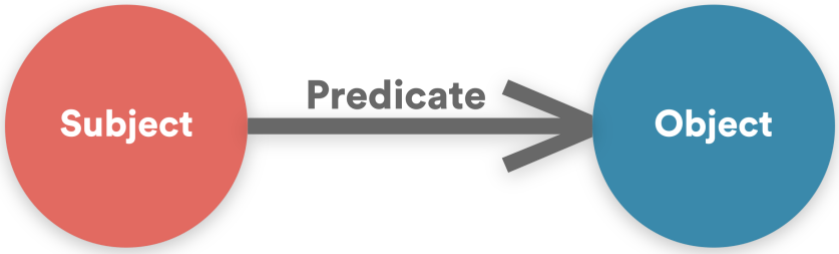

Figure 1.1: Example of a semantic statement

These links result in knowledge graphs of entities and the relationships between them.

Part of the vision of linked data is for the Internet to become a global database.
This is especially useful for our purpose, as it allows us to retrieve data dynamically from any domain.

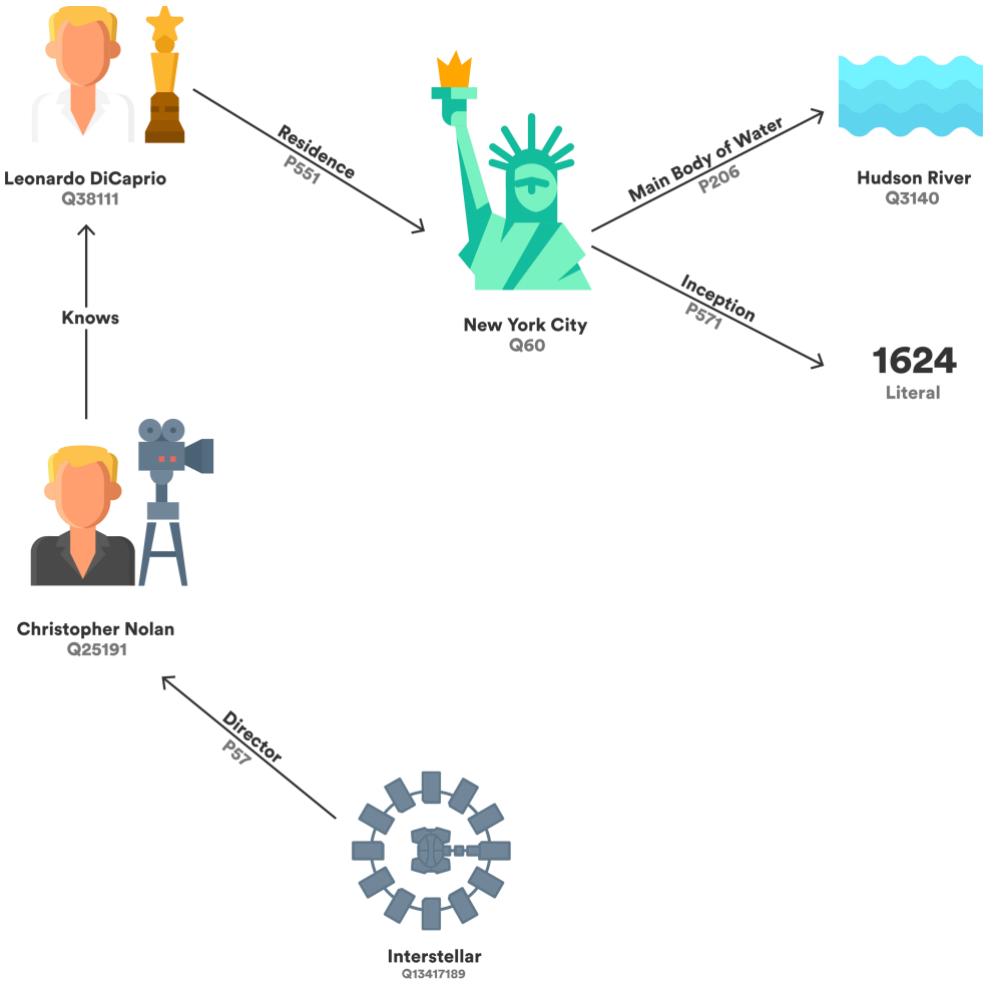

Figure 1.2: Knowledge graph with several entities and the relationships between them

1.1.2 Wikidata

Wikidata is a knowledge base similar to Wikipedia, but it structures its content as Linked Data.

It uses a few more elements than the ones explained in Linked Data, below you can find an example of the most relevant terms used in Wikidata.

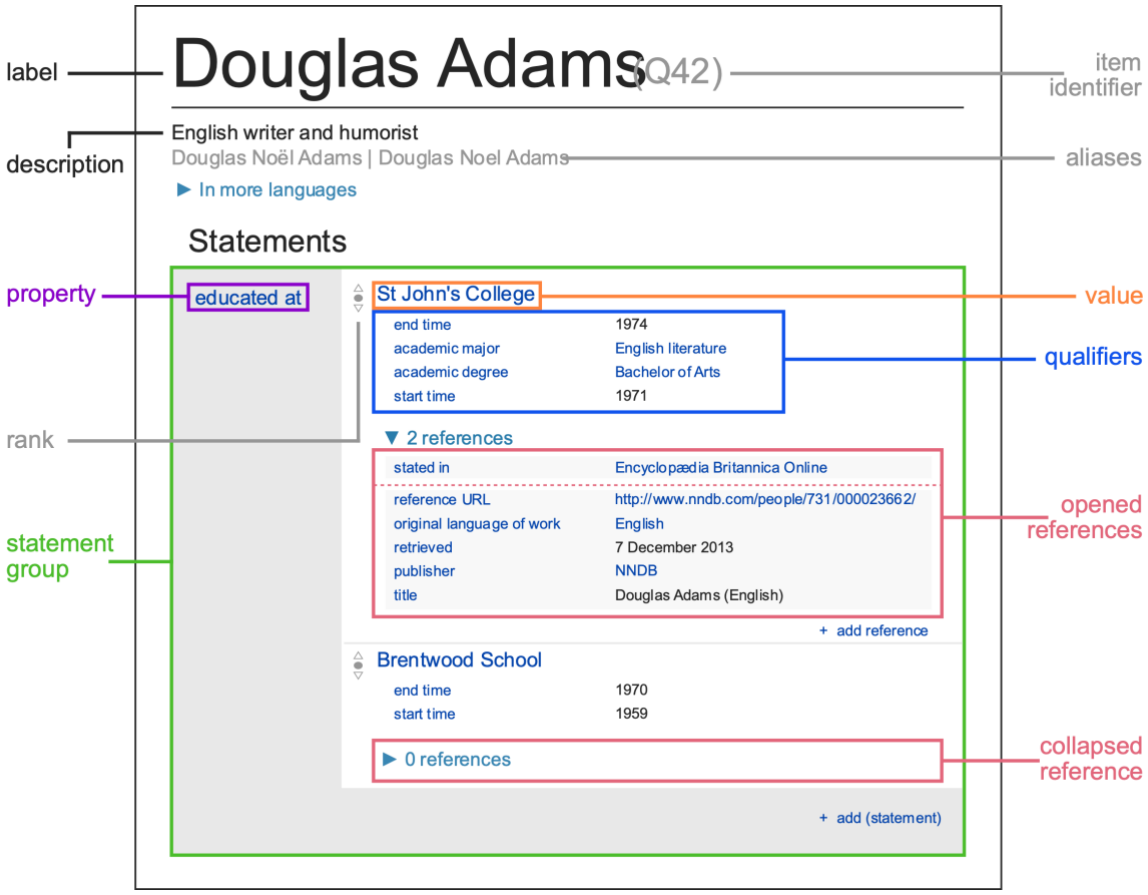

Figure 1.3: Data model in Wikidata

Wikidata exposes a public API that we will use to query against its data set using the semantic query language SPARQL.

1.2 Context and scope

It is worth start by defining the context and scope of the project and this document.

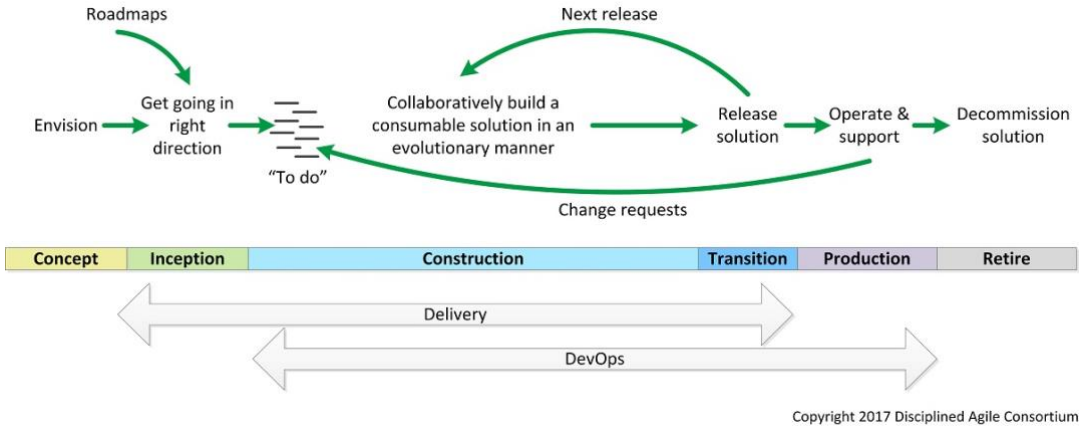

Figure 1.4: High-level view of the system lifecycle

As explained in Motivation, this project was developed for the company Zapiens as part of my internship in WESO.
This factor determined the scope of the project to cover the delivery of a working software that satisfies their needs and requirements.

In Figure 1.4, the different phases of a system's lifecycle can be seen.
In the following sections we will focus on the 'Delivery' phases that this project covers.

1.3 Software Development Lifecycles

In this section, we will discuss how the methodology used to develop the project was selected. This is relevant not only for the system itself, but it has also determined the structure of this document.

For the selection of the software development lifecycle, two large groups were taken into consideration.

1.3.1 Traditional methodologies

On the one hand, traditional or *heavy* methodologies focus on well-defined processes and extensive documentation.

Phases in traditional software development lifecycles are sequential, meaning that each phase must finish before the next starts.

- **Advantages:** traditional methodologies provide control over the process, allowing to plan and forecast the cost, effort and quality of the products that will be obtained.
  They also provide reproducible outcomes (i.e. the final product is closer to the planned product), this allows organizations to define standard processes proven to work and reuse them in future projects.
- **Disadvantages:** traditional methodologies are rigid and do not adapt well to changes; they are not designed to. Requirements must be clear in the beginning, or else the project planning would be at risk of becoming obsolete, and therefore, useless.

Some traditional methodologies: Waterfall model, Spiral model, Phased Release model…

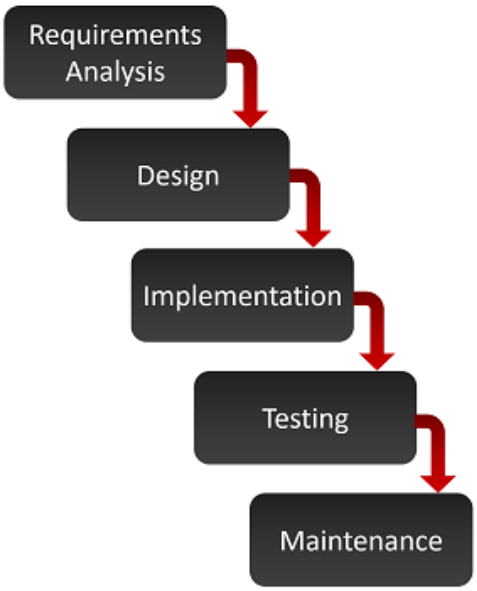

Figure 1.5: Phases on a traditional lifecycle

1.3.2 Agile methodologies

On the other hand, agile methodologies focus on continuous deliveries of working software, as well as on people and their interactions.

Some of their advantages and disadvantages are:

- **Advantages:** They are usually iterative, which allows them to scope the risk to each iteration rather than to the final product.
  They are highly adaptable to rapid changes in the work plan when an unexpected requirement arises or any variation is needed.
  They allow continuous validation of the system, ensuring that it meets the client's requirements.
- **Disadvantages:** They are non-deterministic, meaning that they are not intended to provide reproducible or planned products.
  They require highly skilled people that are trained in the use of such methodologies.

Some agile methodologies: Scrum, Kanban, Extreme Programming (XP), Feature Driven Development (FDD)…

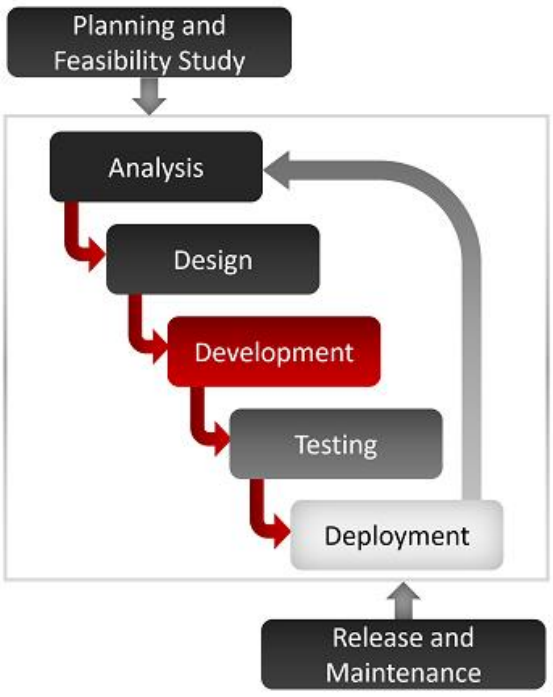

Figure 1.6: Phases on an agile lifecycle

1.3.3 Traditional vs Agile

When making a choice between traditional and agile methodologies, it is also worth noting how they compare in real-life projects.

The CHAOS Report by the Standish Group studies IT project success rates and project management best practices.
They classify projects as 'Successful', 'Challenged' or 'Failed' based on six attributes: OnTime, OnBudget, OnTarget, OnGoal, Value, and Satisfaction.

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

Figure 1.7: Chaos resolution by Agile vs Waterfall (Standish Group, 2015)

As it can be seen in Figure 1.7, the results for all projects show that agile projects have almost four times the success rate as waterfall projects, and waterfall projects have three times the failure rate as agile projects. (Standish Group, 2015)

When considering this project, the following aspects where decisive for making the decision:

- The initial requirements were not well-defined at the beginning.
- The client company expected a continuous validation of a working product, to evaluate the progress and decide how to evolve the project 'on the go', as it adapts better to their internal way of working.
- Changes on the requirements were expected to appear as the project evolves.

With all these constraints, using a traditional methodology was hardly an option, so the team decided to develop the project using an agile methodology that will be described in a following section.

1.4 Disciplined Agile

There are plenty of agile practices, techniques, and strategies available.
One of the greatest challenges is knowing what to choose and how to combine them.
The Project Management Institute (PMI) owns a tool kit called Disciplined Agile that fits together the advantages and characteristics of several agile and non-agile frameworks and methodologies.

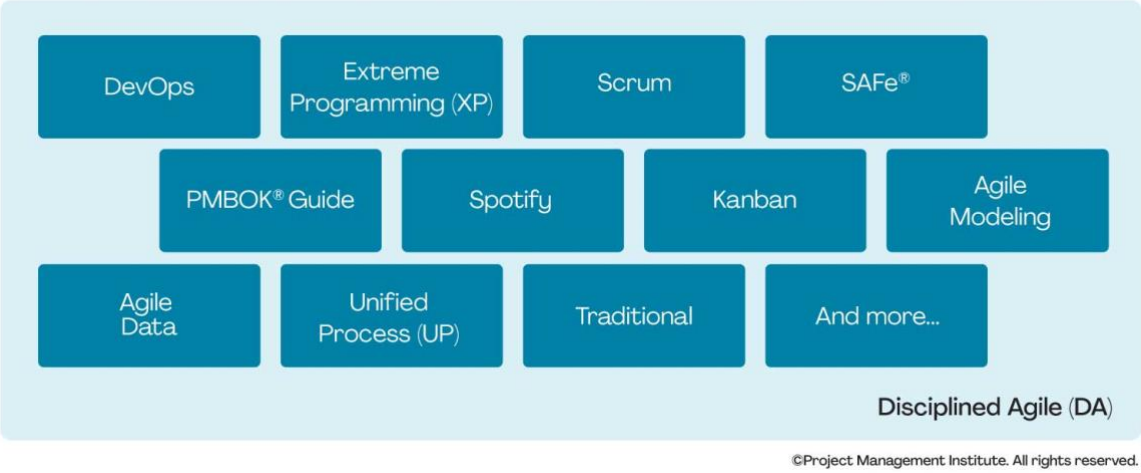

Figure 1.8: Sources of the DA tool kit

As it can be seen in Figure 1.8, DA's sources like the PMBOK Guide, a well-known set of standard terminology and guidelines for project management also developed by PMI helped shape this new tool kit.
Even processes from traditional methodologies are included in PMI's Disciplined Agile.

This diversity of sources makes Disciplined Agile a more robust alternative for developing projects than any other of the agile methodologies by itself and is also the reason that it was selected to develop this project.

1.4.1 Disciplined Agile Delivery

Disciplined Agile Delivery (DAD) is the subset of Disciplined Agile that provides the guidelines for a solution delivery of a software system, and, as stated in Context and scope, this is the scope of the project.

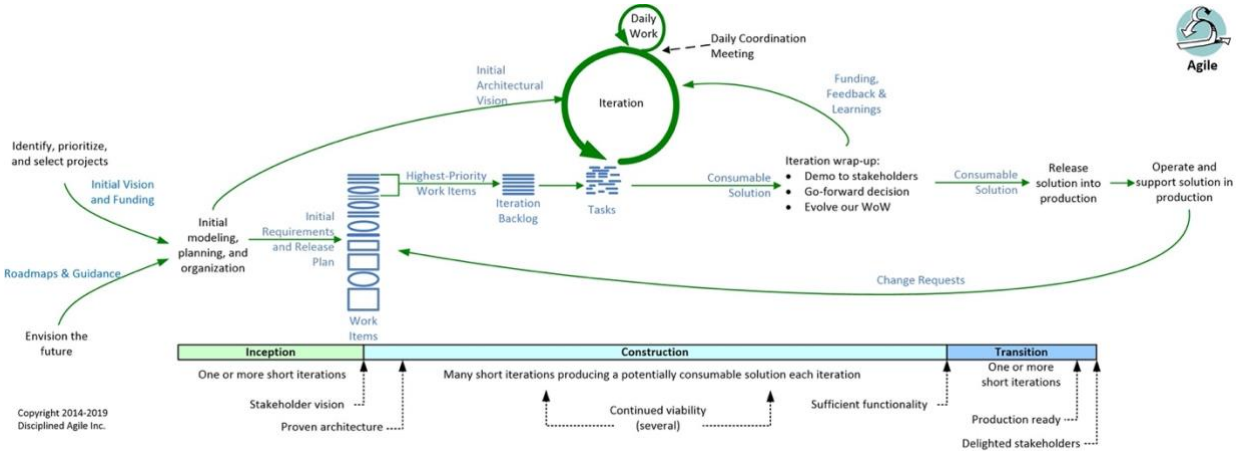

Figure 1.9: DAD's Agile (Scrum based) lifecycle

DAD supports several delivery lifecycles. In Figure 1.9 the different processes that make up DAD's Agile lifecycle (the one we will use) can be seen.

DAD's Agile is a Scrum based lifecycle, this means that even although it uses non-Scrum terminology, we can find the same elements and processes as in Scrum.
However, as the team has experience working with Scrum and its terminology, we will use Scrum terms rather than the generic ones shown in Figure 1.9.

In the following subsections we will define important concepts that will be referenced in the rest of the document.

1.4.1.1 Sprints

Depicted in Figure 1.9 as iterations.
As stated previously, this lifecycle is Scrum based, meaning that the construction phase will be split in cycles of work of *four weeks* each.

The duration of these iterations was agreed by both the development team and the product owner according to our schedules and availability.

1.4.1.2 Scrum roles

In Scrum, there are three roles: Product Owner, Team, and Scrum Master. Together these are known as the Scrum Team.

However, in this project some additional roles appear. The following are the roles of each of the people involved in the project and their descriptions.

Product Owner

The Product Owner for this project was *Iván Arrizabalaga* from *Zapiens.*
As a Product Owner he is responsible for maximizing return on investment (ROI) by identifying product features (i.e. he is responsible for the value of the work).

Some other of his responsibilities are to regularly interact with the Development Team, prioritize by working with all of the stakeholders and review the results each sprint. (Deemer, et al., 2012)

Development Team

The Development Team, or Team is composed by *Emilio Cortina Labra*.

The Team should be "cross-functional" (i.e. it includes all the expertise necessary to deliver the potentially shippable product each Sprint) and "self-organizing".

"The Team in Scrum for a software product might include people with skills in analysis, development, testing, interface design, database design, architecture, documentation, and so on. The Team develops the product and provides ideas to the Product Owner about how to make the product great." (Deemer, et al., 2012)

In this special case, where the team is composed by only one person, all those skills will have to be delivered by that person.

Scrum Master

"The **ScrumMaster** coaches and guides the Product Owner, Team and the rest of the organization in the skillful use of Scrum. The ScrumMaster is a *coach* and *teacher*.
There should be a dedicated full-time Scrum Master, although a smaller Team might have a team member play this role." (Deemer, et al., 2012)

Due to the lack of a dedicated person to play this role, the Development Team had to self-educate in the skillful use of Scrum.

External consultants

Additionally, some external stakeholders did make contributions to the project.
These people attended the Sprint Review meetings to give feedback from their different points of view:

- *Daniel Fernández:* Web semantics consultant from *WESO*.
- *Eduardo Di Santi:* Artificial Intelligence consultant from *Zapiens*.

1.4.1.3 Product backlog

The Product Backlog is a collection of **work items**.
These **work items** include customer features, engineering improvement goals (e.g. refactoring), research work, and known defects.

"Product Backlog **items** are articulated in any way that is clear and sustainable. Contrary to popular misunderstanding, the Product Backlog does *not* contain "user stories"; it simply contains *items*. Those items can be expressed as user stories, use cases, or any other requirements approach that the group finds useful. But whatever the approach, most items should focus on delivering value to customers." (Deemer, et al., 2012)

A good Product Backlog should be ***DEEP****:*

Detailed appropriately

The top priority items are more detailed than the lower priority items, since the former will be worked on sooner than the latter.

Estimated

The items in the Product Backlog need to have estimates, and will be constantly considered for re-estimation as everyone learns and new information arises

These estimates may not only include effort estimates for each item but can also include technical risk estimates.

Story Points will be the unit of measurement for expressing the overall size of the items in the Product Backlog.

"The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story. Rather, a story point estimate is an amalgamation of the amount of effort involved in developing the feature, the complexity of developing it, the risk inherent in it, and so on." (Cohn, 2005)

The estimations will be made by the Team using analogies between items in the Product Backlog and assigning a range of fixed values from the Fibonacci sequence:

1, 2, 3, 5, 8, 13, 21.

Story Points are a linear metric, that is, an item estimated with 2 story points is expected to take twice the effort as one estimated with 1 story point.

Emergent

The Product Backlog is regularly refined. Each Sprint, items may be added, removed, modified, split, and changed in priority. Thus, the Product Backlog is continuously updated by the

Product Owner to reflect changes in the needs of the customer, new ideas or insights, moves by the competition, technical hurdles that appear, and so forth.

Prioritized

The Product Owner is in charge of prioritize the items in the Product Backlog.

They can base this prioritization on revenue gained, costs reduced, business risks, importance to various stakeholders, and more. Usually, a good criterion is to prioritize items that deliver the highest business value for the lowest cost.

Another motivation to increase the priority of an item is to *tackle high risks early, before the risks attack you*.

1.4.1.4 Sprint Planning Meeting

The Sprint Planning Meeting is a meeting to prepare for the upcoming sprint divided in two parts.

Part one

This part focuses on understanding *what* the Product Owner wants, and that is called the Sprint Goal.
The Sprint Goal is a summary of the Sprint objectives, that should share a cohesive theme.

The Product Owner and the Team review and select the high-priority items to be implemented in the next Sprint, reordering the Product Backlog.

Part two

From the reordered Product Backlog, and with the Sprint Goal defined, this part of the meeting focuses on *how* to implement the items that the Team decides to take on.
It is responsibility of the Team to decide the size of the increment that they are going to implement.

As Scrum does not prescribe how to proceed with the analysis of the Sprint Planning Part Two, in this case we will use UML diagrams to analyze and design the increment that will be implemented.

The results of this meeting will be the updated system diagrams along with the subset of the Product Backlog that will be implemented during the Sprint, also called, **Sprint Backlog.**

				New Estimates of Effort Remaining at end of Day...					
Product Backlog Item	**Sprint Task**	**Volunteer**	**Initial Estimate of Effort**	**1**	**2**	**3**	**4**	**5**	**6**
As a buyer, I want to place a book in a shopping cart	modify database	Sanjay	5	4	3	0	0	0	
	create webpage (UI)	Jing	3	3	3	2	0	0	
	create webpage (Javascript logic)	Tracy & Sam	2	2	2	2	1	0	
	write automated acceptance tests	Sarah	5	5	5	5	5	0	
	update buyer help webpage	Sanjay & Jing	3	3	3	3	3	0	
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5	5	5	5	5	5	
	complete machine order for pRank		3	3	8	8	8	8	
	change DCP and reader to use pRank http API		5	5	5	5	5	5	
	. . .	. . .	. . .						
		Total	50	49	48	44	43	34	

Figure 1.10: Example of Sprint Backlog with daily updates on work remaining

1.4.1.5 Backlog Grooming

During the Sprint, the Team and Product Owner will conduct activities to continually improve the quality of the items in the Product Backlog, using the knowledge obtained during the process.
This turns Sprint Planning into a relatively simple process, since the Product Owner and the Team start the meeting with a well-analyze Product Backlog.

These activities include splitting big items into smaller ones, analyze items, re-estimate and re-prioritize for future Sprints.

Note that these activities are *not* for the items selected for the current Sprint, since the current increment should not be modified during the Sprint.

1.4.1.6 Tracking progress during the sprint

Since the Team is self-managing, they carry out everyday activities for monitoring their progress such as updating the estimates for each of the sprint tasks, as seen in Figure 1.10: Example of Sprint Backlog with daily updates on work remaining.

A common technique for representing these daily updates and the effort remaining to complete the current Sprint is to plot the **Sprint Burndown Chart.**
Figure 1.11 shows an example of a Sprint Burndown Chart, representing for each day a new

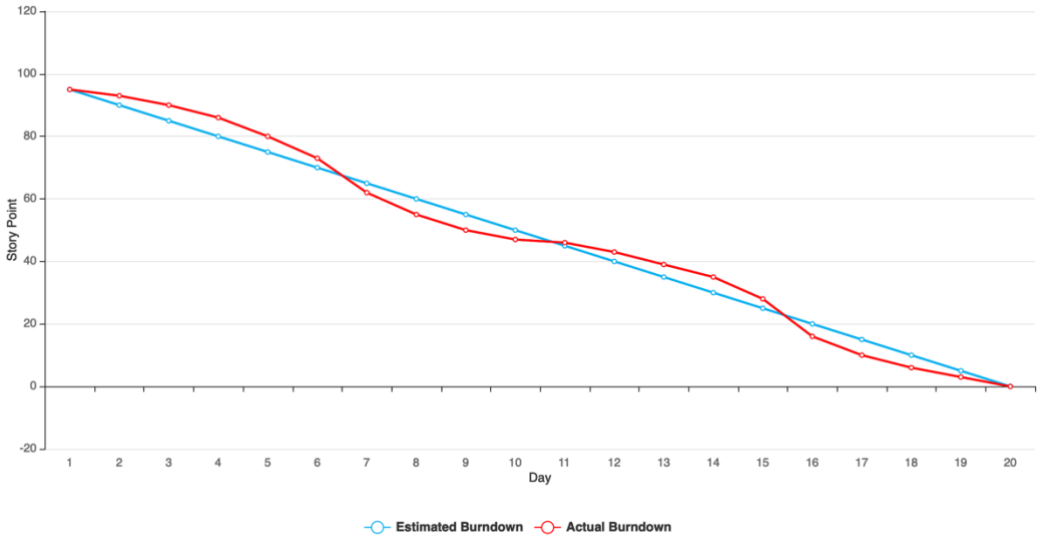

Figure 1.11: Example of a Sprint Burndown Chart

estimate of how much work remains until the Team is finished.

Ideally the burndown line the burndown line is always tracking downwards towards completion near the end of the Sprint, but if it wasn't, the Team would need to adjust and reduce the scope and size of the increments so that the expectations are met.

1.4.1.7 Sprint Review

The Sprint Review is an activity conducted after the Sprint ends, where Product Owner, Team members and other stakeholders review the ***product*** obtained in the Sprint.

It is a time for the Product Owner to learn what is going on with the product and with the Team; and for the Team to learn what is going with the Product Owner and the market.

The review includes using the actual live software that the Team developed during the Sprint.

1.4.1.8 Sprint Retrospective

Whereas the Sprint Review is an activity that involves inspect and adapt regarding the ***product,*** the Sprint Retrospective involves inspect and adapt regarding the process and environment.

The Team discusses what's working and what's not working, and plans changes on the process to evolve their Way of Working (WoW).

2 Inception

2.1 Initial Product Backlog Creation

The initial requirements for the system were:
"Build a REST API using Python that allows to generate multiple-choice questions from a label expressed in natural language. This API must use Wikidata as the source of knowledge".

In order to begin with the process of building the system, and to further develop these initial requirements, an Initial Product Backlog Creation workshop was done.

In this Workshop, an interview with the Product Owner and a brainstorming session with the external consultants were conducted. The result of these sessions was a deeper understanding of the needs of the Product Owner and the scope of the system, which was represented in the initial Product Backlog created and validated by the Product Owner together with the Team.

Table 1: Initial Product Backlog

ID	Work item	Description	Estimate
1	As a user, I want to be able to make HTTP requests to the REST API.	Build the initial infrastructure for a REST API using Python 3.7 as the programming language.	8
2	As a user, I want to be able to obtain Wikidata entities from a label.	Given a label entered as input to the API in natural language, retrieve the associated entities from Wikidata.	13
3	As a user, I want to be able to generate multiple-choice questions from a Wikidata entity.	Given a Wikidata entity entered as input to the API referenced by its identifier, generate and return multiple-choice questions related to the entity.	21

2.2 Analysis of previous work

Before starting the development, a research on automatic generation of questions was conducted. This research includes the analysis of several scientific papers that tackled the same or similar problems, together with the advantages, disadvantages and other insights on each of the approaches.

2.2.1 Automatic generation of multiple-choice questions from slide content using linked data (Faizan & Lohmann, 2018)

Aims at composing multiple-choice questions from content inside slides documents.
They do this by identifying entities from the text, and possible statements where these entities appear, which they later use for the generation of the questions.

It defines three different types of questions:

- ***Gap-fill:*** they are generated from a statement where a given entity appears. Then, the entity is substituted by a blank space and other similar entities are given as distractors.
- ***Choose-the-type:*** they ask about the type of an entity. Distractors are also similar to the type of the entity.
- ***Jeopardy-style:*** questions that include some kind of hints about the answer in the statement.

It makes use of DBPedia as the source of information, and adopts an approach based in templates to convert the semantic information into questions expressed in natural language.

Its strengths and weaknesses when compared to our goals are:

- ***Advantages*:** It allows to choose between different *difficulty levels* that it supports by selecting distractors that are semantically similar to the correct answer.
- ***Disadvantages*:** it requires to give the system as input a document with all the information used to generate the questions.
  This results in a rather *ad hoc* process instead of an automated one, which is our aim, since requiring users or administrators to input the domain information is already the current behavior of Zapiens' system.

2.2.2 Automatic generation of quizzes from DBPedia according to educational standards (Rodríguez Rocha & Faron Zucker, 2018)

This article studies how to apply automatic generation of quizzes into schools.

Its strengths and weaknesses when compared to our goals are:

- ***Advantages*:** It uses five different strategies for generating the questions, resulting in five different types of questions available.
- ***Disadvantages*:** It is focused on schools, so the ontologies used (Eduprogression and Les Incollables Knowledge Base) are very specific of this domain.

2.2.3 Knowledge questions from knowledge graphs (Seyler, et al., 2016)

This approach makes use of DBPedia as the source of information.

It adopts a model based in templates to convert the semantic statements into natural language sentences, it also allows to evaluate the difficulty of the questions to adapt to the knowledge of the user.

2.2.4 Semantic multiple-choice question generation and concept-based assessment (Jelenkovic & Tosic, 2013)

This system stands out from the others for the technique used to vary the difficulty of the questions. For instance, in order to make a question easier, it can choose entities that do not appear in a similar relationship as the correct answer (e.g. select a movie as distractor of a question whose correct answer is a city).

Its strengths and weaknesses when compared to our goals are:

- ***Advantages*:** It allows to change the difficulty of the questions.
- ***Disadvantages:***
  - It requires to set up several parameters.
  - It is not possible to choose to generate questions about a specific entity, only about a certain topic or domain (e.g. it would not possible to generate questions about ‘Madrid’, only about ‘Geography’ or ‘Cities’).
  - Generated questions are hardly comprehensible in natural languages (e.g. ‘Which one of the following response pairs relate in the same way as x and y in the relationship p?’).

2.3 Analysis of alternatives

2.3.1 Python Framework

For the selection of a framework for building the API, and being restricted to the initial requirement given by the Product Owner that the system **must** be built using **Python 3.7** (see Initial Product Backlog Creation) as the programming language, Flask and Django were considered as the two main options due to their popularity.

We will briefly discuss how they compare against each other.

2.3.1.1 Django

Django is a free and open-source full-stack web framework for Python.

It follows the *batteries-included* approach, that is, it supports many standard functionalities to build websites out-of-the-box.

In terms of stability, Django generally has longer, more rigid release cycles.

Django is suited for more significant projects that need much **functionality**. For more straightforward projects, the features might be an overdose.

2.3.1.2 Flask

Flask, on the other hand, is a lightweight and extensible framework, also categorized as a micro-framework.

Flask is simple and unopinionated; it doesn't decide what your application should look like – developers do.

Flask generally takes longer to set up since you have to add different extensions based on your needs (i.e., ORM, permissions, authentication, etc.). This initial cost results in more **flexibility** down the road for applications that don't fit the standard Django model

2.3.2 IDE

An Integrated Development Environment (**IDE**) is a software application that provides the essential functionalities for the construction of a software system.

It typically includes a source code editor, build automation tools, a debugger and a version control system.

It is an important element for the construction process of the system, so it is worth taking into account a couple of alternatives to choose from. In our case, we will consider **Pycharm** and **Visual Studio Code** as they are the most popular options for **Python** development.

2.3.2.1 Pycharm

Pycharm is an IDE specifically created for Python; therefore, all its features are targeted toward streamlining the process of coding with Python.

Among other things, Pycharm includes a built-in debugger and support for multiple Python frameworks out-of-the-box. This helps make the process of developing, running and debugging the system a much faster and seamless task.

2.3.2.2 Visual Studio Code

Visual Studio Code is technically a code editor, and not a proper IDE.

However, it is highly extensible, providing a great number of extensions available on its marketplace to customize the environment so that it suits best the type of development.

This means that the developers have to make an effort of analyzing the features and extensions that they will need and add them manually to the application.

2.3.3 Communication with Wikidata

Wikidata provides several tools for developers to obtain information from their knowledge graph.

2.3.3.1 Wikidata Python library

Wikidata provides a package with a few APIs to use Wikidata for Python.

This package allows to get entities by their identifier, and retrieve their name, description, image and some extra information.

The library, however, lacks some essential functionality like searching entities given a label or making SPARQL queries.

2.3.3.2 Wikidata API endpoints

Wikidata provides several ways to access its data via their URL endpoints.

Some of these APIs include searching entities and querying information from their SPARQL endpoints.

2.3.4 Generation process

For generating the questions about an entity, several methods and techniques were analyzed in Analysis of previous work. From this analysis, the Team came up with two alternatives to evaluate together with the Product Owner to see which one fits best the use cases of the system.

2.3.4.1 Template based

This approach consists on defining templates of questions that will be generated for the entity.

These templates of questions consist on a set of predicates (Wikidata properties) grouped according to different categories or themes.
Each predicate is associated to the natural language representation of its question.

From these properties, a question is generated by simply querying for the value of that property for the given entity (i.e. the correct answer), and also the value of that property for other entities other than the given entity (i.e. the distractors).

- **Advantages:** allows to curate the natural language representation of the statement of each question making it grammatically correct and easily understandable.
- **Disadvantages:** it requires to manually implement each template, defining all the included predicates, and for each predicate writing the natural language representation of the statement of the question.
  It is not guaranteed that any entity will have a value for every property defined in a template.
  It is not flexible for adding support for new languages.

This approach is the one we selected, as it provides users with questions expressed in natural language in a more intuitive way and more similar to the ones that they are used to.
The technique is further explained in Generation of the questions.

2.3.4.2 Dynamic predicates

Consists on evaluating the predicates or Wikidata properties that the entity ‘has’ and using these predicates for making the questions about the property.

- **Advantages:** it does not require to choose a template of questions by the user.
  It is highly extensible to add support for new languages, since every Wikidata property is localized.
- **Disadvantages:** questions would not be expressed in a natural way. They would have to be of the form: ‘Which is the value of the property *capital* for the entity *Spain*?’; rather than: ‘Which is the capital of Spain?’.

2.4 Tech stack

2.4.1 Programming language and framework

As part of the initial requirements, it was agreed that the programming language used to construct the system would be ***Python*** 3.7.

Previously, in Analysis of alternatives, the main characteristics of two of the most popular Python web frameworks were analyzed side by side. ***Flask*** was finally chosen due to its flexibility and low amount of boilerplate code and unnecessary functionality that didn't fit the scope of this system.

2.4.2 Testing environment

Pytest is the testing framework that was used to implement the scripted unit tests of the system. ***Flask*** includes full support and plenty of documentation to set up ***Pytest***, and those were the main reasons why it was chosen.

2.4.3 Development environment

The Integrated Development Environment used to develop, debug and test the system will be ***JetBrain's PyCharm.***

As seen previously in Analysis of alternatives, PyCharm offers full support of Flask and other useful functionality for developing Python projects out-of-the-box, as opposed to Visual Studio Code that requires a heavy process to set up the development environment.

2.4.4 Communication environment

The Team and Product Owner used ***Basecamp***, a web-based project management tool to do a continuous tracking of the progress of the system.

For the several meetings that were held between the Team, Product Owner and other stakeholders, ***Google Hangouts*** was used.

2.4.5 Other technologies

As explained in previous sections, Linked Data was the central technology of the project, with Wikidata being the source of the semantic information that was used. They were also part of the initial requirements of the project (see Initial Product Backlog Creation).

In order to query this information, the semantic query language SPARQL was used. This language was made a standard by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is recognized as one of the key technologies of the semantic web.

3 Construction

3.1 First Sprint

3.1.1 Sprint Planning

3.1.1.1 Part one

This first part of the Sprint Planning meeting consisted on the Team, Product Owner and the external stakeholders choosing what is going to be done.

Since the Product Backlog was previously defined in Initial Product Backlog Creation, this meeting was a fairly simple activity were the Product Owner defined priorities for the items in the Product Backlog.

The Sprint Goal was also defined: “Create the initial REST API architecture and return Wikidata entities giving the label as input of an HTTP GET request”.

Reordered Product Backlog

Table 2: Reordered Product Backlog (1st Sprint)

ID	Work item	Description	Estimate	Priority
1	As a user, I want to be able to make HTTP requests to the REST API.	Build the initial infrastructure for a REST API using Python 3.7 as the programming language.	8	1
2	As a user, I want to be able to obtain Wikidata entities from a label.	Given a label entered as input to the API in natural language, retrieve the associated entities from Wikidata.	13	2
3	As a user, I want to be able to generate multiple-choice questions from a Wikidata entity.	Given a Wikidata entity entered as input to the API referenced by its identifier, generate and return multiple-choice questions related to the entity.	21	3

3.1.1.2 Part two

In this second part of the Sprint Planning, the Team and the Product Owner analyzed the functionality that is going to be implemented by creating the Sprint Backlog and modelling the system with UML diagrams.

Use case diagram

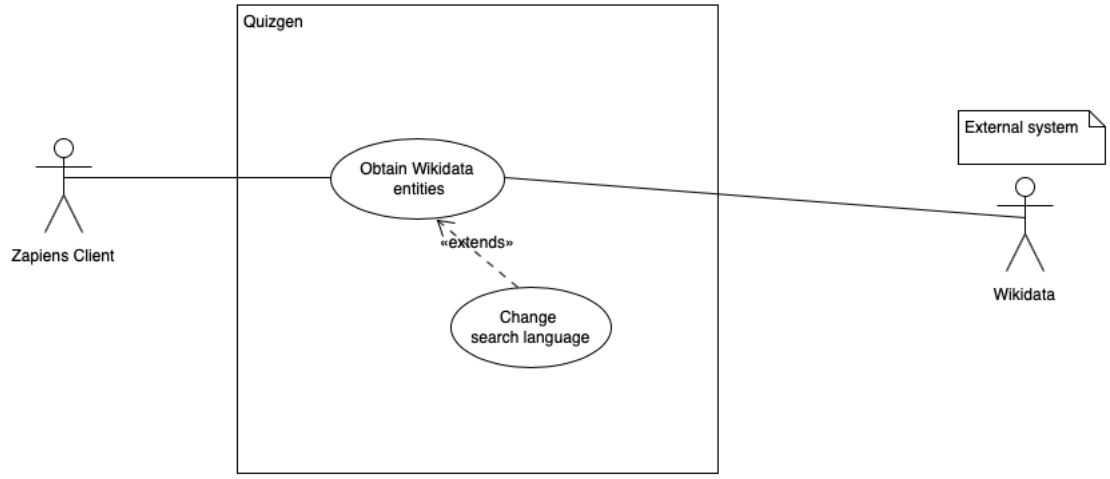

Figure 3.1: Use Case diagram of the system (1st Sprint)

Component diagram

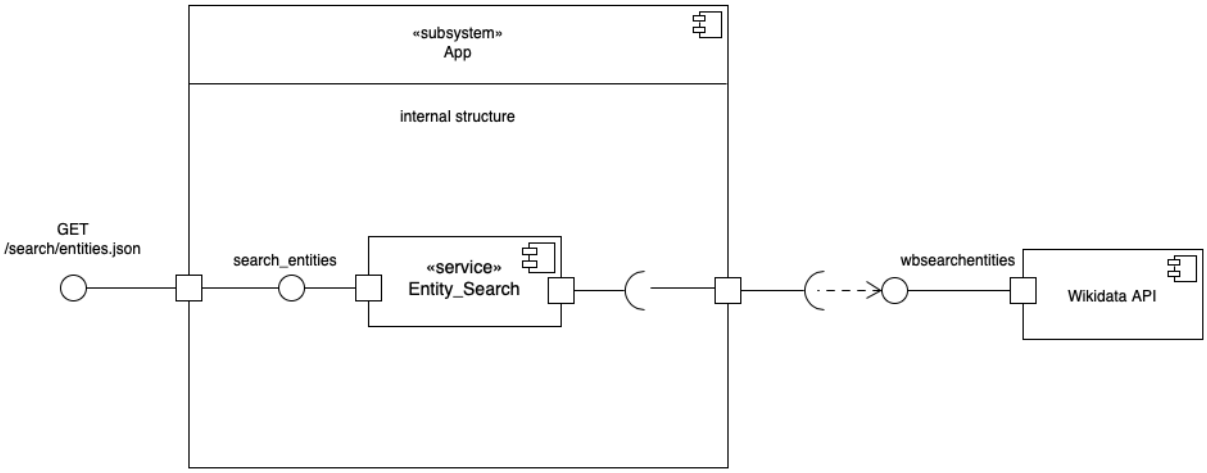

Figure 3.2: Component diagram of the system (1st Sprint)

Sequence diagrams

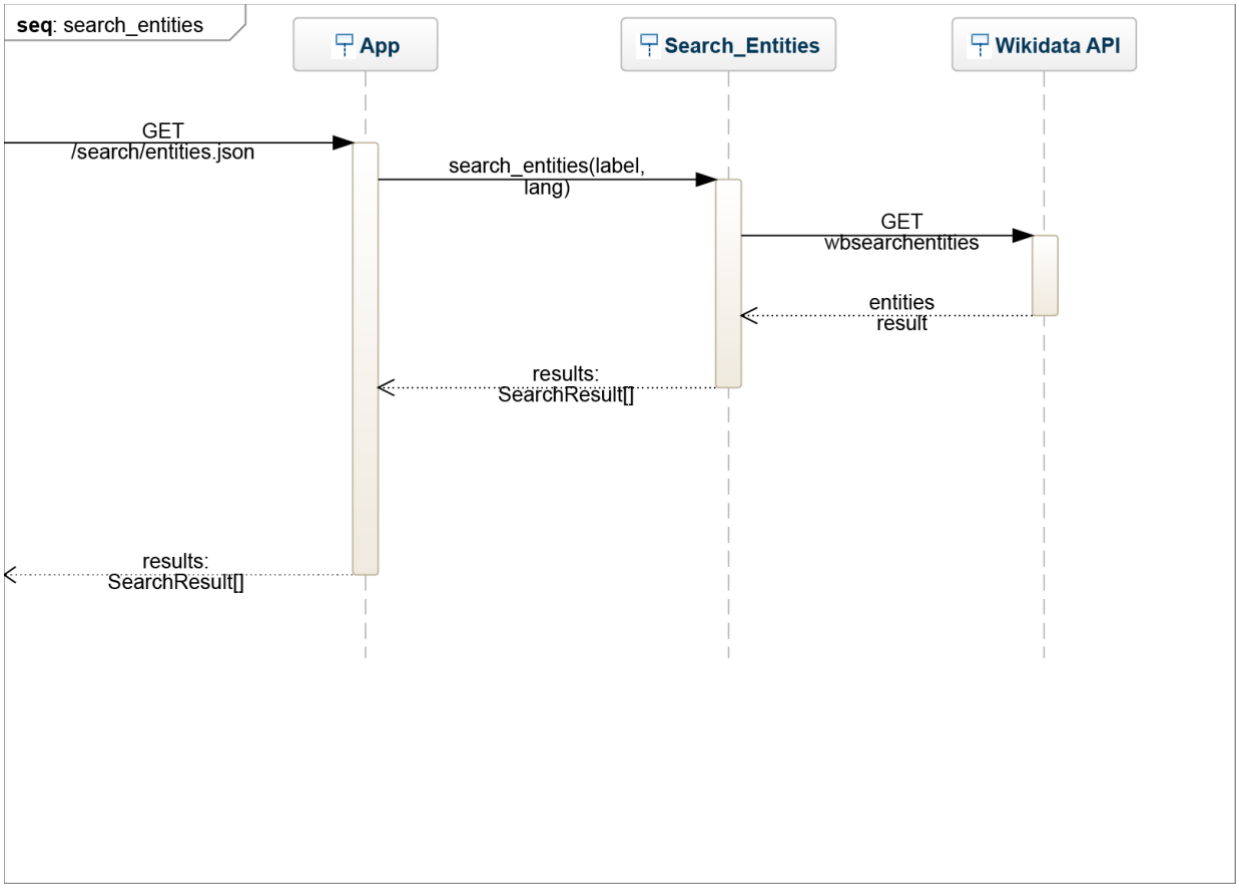

Figure 3.3: Sequence diagram of the functionality for searching entities (1st Sprint)

Class diagrams

In this section we will add the different data types that appear in the other diagrams.

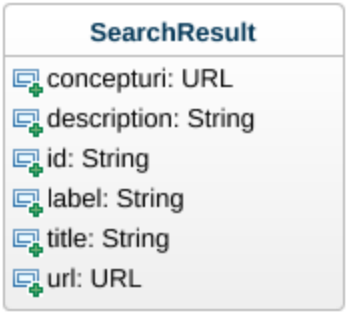

Figure 3.4: Class diagram of the data types

3.1.2 Sprint Backlog

The Sprint Backlog obtained in the second part of the Sprint Planning is as follows.

Table 3: Sprint Backlog (1st Sprint)

ID	Product Backlog item	Sprint Task	Initial Estimate of Effort
1	As a user, I want to be able to make HTTP requests to the REST API.	Set up development environment	3
		Build initial project structure	3
		Set up the initial endpoints of the API.	2
2	As a user, I want to be able to obtain Wikidata entities from a label.	Define the new endpoint(s) for searching entities.	1
		Define the inputs and data needed for searching entities	2
		Connect with Wikidata API for searching entities	5
		Process Wikidata responses and return the obtained entities	5

3.1.3 Testing

In this section we will explain the different tests that were carried out for the system.

The tests designed are a combination of scripted and exploratory testing.
Scripted tests focus on checking the correct behavior of the API, whereas exploratory tests focus on analyzing the outputs to check their meaning.

3.1.3.1 Design

The approach for designing the test coverage items will be the minimized approach: create the minimum number of cases to cover valid equivalence classes and one test case to cover each of the invalid classes (to avoid masking of bugs).

Search entities: Test coverage items

Table 4: Search entities test coverage items (1st Sprint)

ID	label	lang	Expected output	Actual output
1	‘Madrid’	‘es’	Obtain entities related to Madrid in Spanish.	Obtain entities related to Madrid in Spanish.
2	‘European Union’	‘en’	Obtain entities related to the	Obtain entities related to the

			European Union in English.	European Union in English.
3	‘Moscú’	‘it’	Obtain entities related to Moscow in Italian.	Obtain entities related to Moscow in Italian.
4	‘aewqkeqw’	‘fr’	Invalid or obtain random entities.	Error showing no entities obtained.
5	‘’	‘pt’	Invalid or obtain random entities.	Error showing no entities obtained.
6	‘Madrid’	‘’	Obtain entities related to Madrid in English.	Error
7	‘Madrid’	‘portugués’	Obtain entities related to Madrid in English.	Error

3.1.3.2 Bug reports

In this section we will discuss the errors identified from the tests carried out on the system.

These errors were fixed during the development to behave as expected.

Table 5: Bug reports (1st Sprint)

Function	Error title	Error description
Search entities	Empty language	When an id for the language is not entered, it causes an error instead of setting English as default.
Search entities	Wrong language identifier	When an id for the language is not entered correctly, it causes an error instead of setting English as default.

3.1.4 Tracking progress

The following table represents the daily updates of work remaining (measured in story points) during the sprint.

Table 6: Daily updates of work remaining (1st Sprint)

Sprint task	Estimate at day										
	0	2	4	6	8	10	12	14	16	18	20
Set up development environment	3	3	2	1	0	0	0	0	0	0	0
Build initial project structure	3	3	3	3	3	1	0	0	0	0	0
Set up the initial endpoints of the API.	2	2	2	2	2	2	2	1	0	0	0
Define the new endpoint(s) for searching entities.	1	1	1	1	1	1	1	1	0	0	0
Define the inputs and data needed for searching entities	2	2	2	2	2	2	2	2	2	0	0
Connect with Wikidata API for searching entities	5	5	5	5	5	5	5	5	5	2	0
Process Wikidata responses and return the obtained entities	5	5	5	5	5	5	5	5	5	5	0
Total	21	21	20	19	18	16	15	12	10	5	0

The following graph represents the values from the previous table as the **Burndown Chart**.

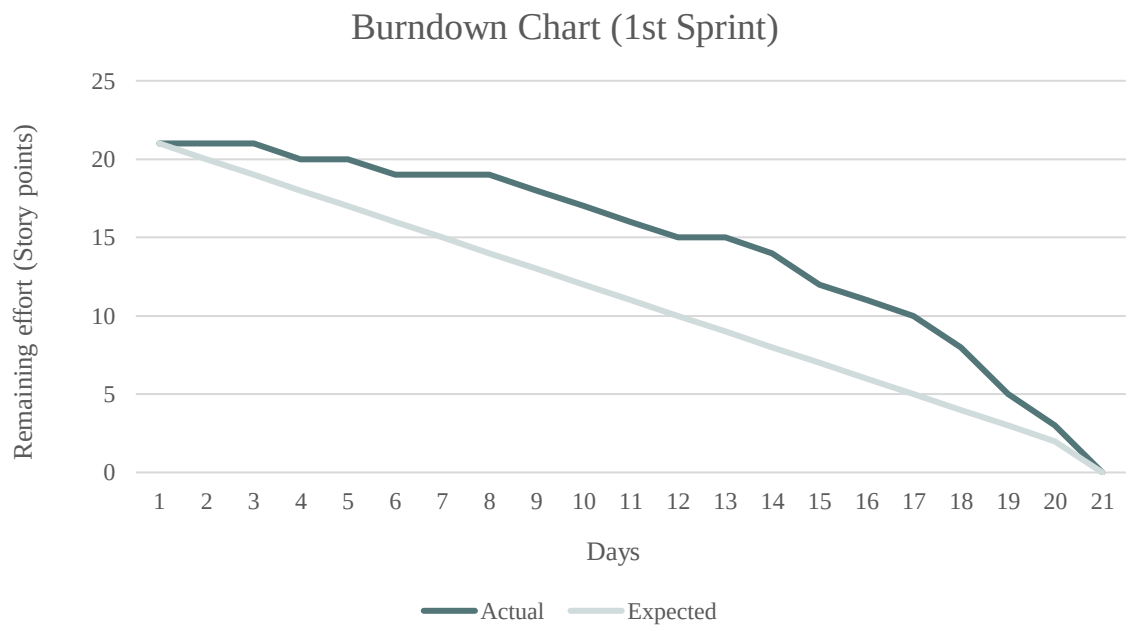

Figure 3.5: Burndown chart (1st Sprint)

3.1.5 Backlog Grooming

During this Sprint, the remaining items in the Product Backlog were refined.

The analysis and research carried out on Wikidata API in order to implement the functionality of this Sprint facilitated also the planning of the questions generation process that will be tackled in the following Sprints.

Additional requirements given by the Product Owner were also included for future Sprints.

Table 7: Backlog Grooming (1st Sprint)

ID	Work item	Description	Estimate
3	As a user, I want to be able to generate multiple-choice questions from a Wikidata entity.	Given a Wikidata entity entered as input to the API referenced by its identifier, generate and return multiple-choice questions related to the entity.	21
4	As a user, I want to be able to choose the language of the generated questions.	Given a label or identifier representing the language, change the language of the questions that the system generates.	8
5	As a user, I want to be able to choose the topic of the questions.	Given a label or identifier representing the topic, change the topic of the questions that the system generates (i.e. questions about 'Geography', 'Business'…)	13

3.1.6 Sprint Review

This Sprint involved mainly the creation of the initial infrastructure and architecture for the system: setting up the environment and dependencies, create the folder structure and define the initial endpoints.

However, functionality regarding transforming concepts expressed in natural language to Wikidata entities was also implemented.

This Sprint Review focused on validating with the Product Owner and the other stakeholders all the functionality that was implemented into the delivered system. This validation was performed as a *demo* using **Postman** (an application for making HTTP requests to our REST API).

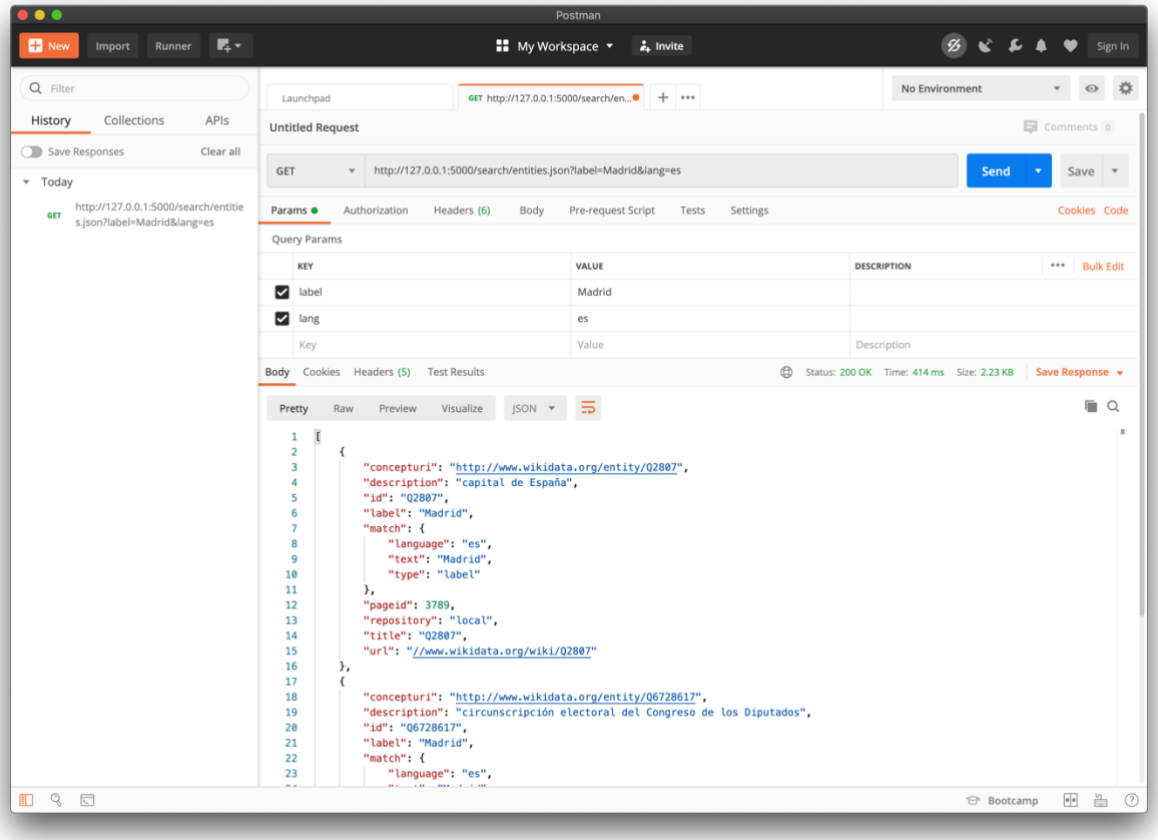

Figure 3.6: Example of HTTP request for validating the search functionality

3.1.7 Sprint Retrospective

During this Sprint Retrospective, the Team and Product Owner analyzed the information obtained during the daily updates of work remaining and the Burndown chart to reflect on the progress of the process.

As seen in Figure 3.5: Burndown chart (1st Sprint), overall the development started with a slower pace than expected but still ended as planned.

This means that the initial tasks were underestimated, setting up the environment and creating the initial infrastructure of the API took more effort than expected and slowed the progress of the Team at the beginning.

However, once these tasks were completed, the knowledge obtained by the Team caused the pace of the development to speed up again and the Team managed to complete the Sprint in time.

The conclusions are that, even although the estimates of the Product Backlog items were accurate, the estimates of the individual Sprint tasks should have taken into account the complexity inherent to the beginning of the development process, that make the Team progress at a much slower pace.

3.2 Second Sprint

3.2.1 Sprint Planning

3.2.1.1 Part one

This first part of the Sprint Planning meeting consisted on the Team, Product Owner and the external stakeholders defining the priorities of the remaining items in the Product Backlog and setting the goal of the Sprint.

The Sprint Goal was defined as: “Build the basic functionality to generate multiple-choice questions from a Wikidata entity”.

Reordered Product Backlog

Table 8: Reordered Product Backlog (2nd Sprint)

ID	Work item	Description	Estimate	Priority
3	As a user, I want to be able to generate multiple-choice questions from a Wikidata entity.	Given a Wikidata entity entered as input to the API referenced by its identifier, generate and return multiple-choice questions related to the entity.	21	1
5	As a user, I want to be able to choose the topic of the questions.	Given a label or identifier representing the topic, change the topic of the questions that the system generates (i.e. questions about ‘Geography’, ‘Business’…)	8	2
4	As a user, I want to be able to choose the language of the generated questions.	Given a label or identifier representing the language, change the language of the questions that the system generates.	8	3

3.2.1.2 Part two

In this second part of the Sprint Planning, the Team and the Product Owner analyzed the functionality that is going to be implemented by creating the Sprint Backlog and modelling the system with UML diagrams.

Use case diagrams

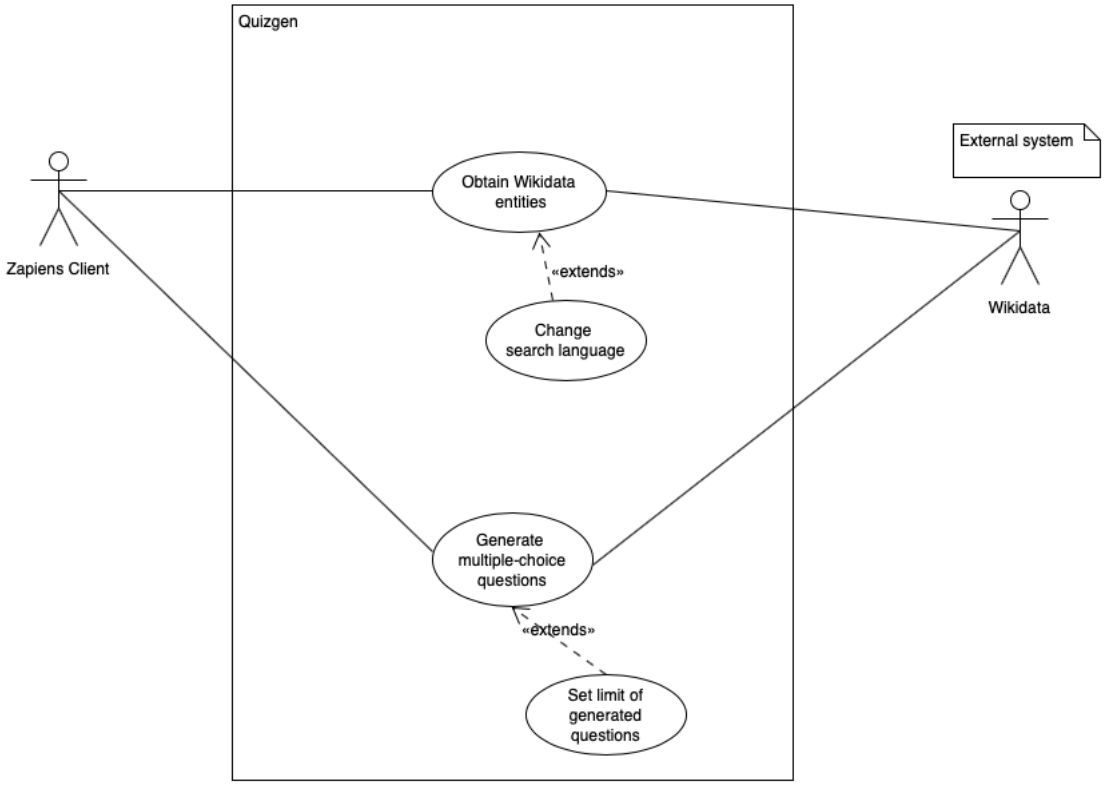

Figure 3.7: Use case diagram of the system (2nd Sprint)

Components diagrams

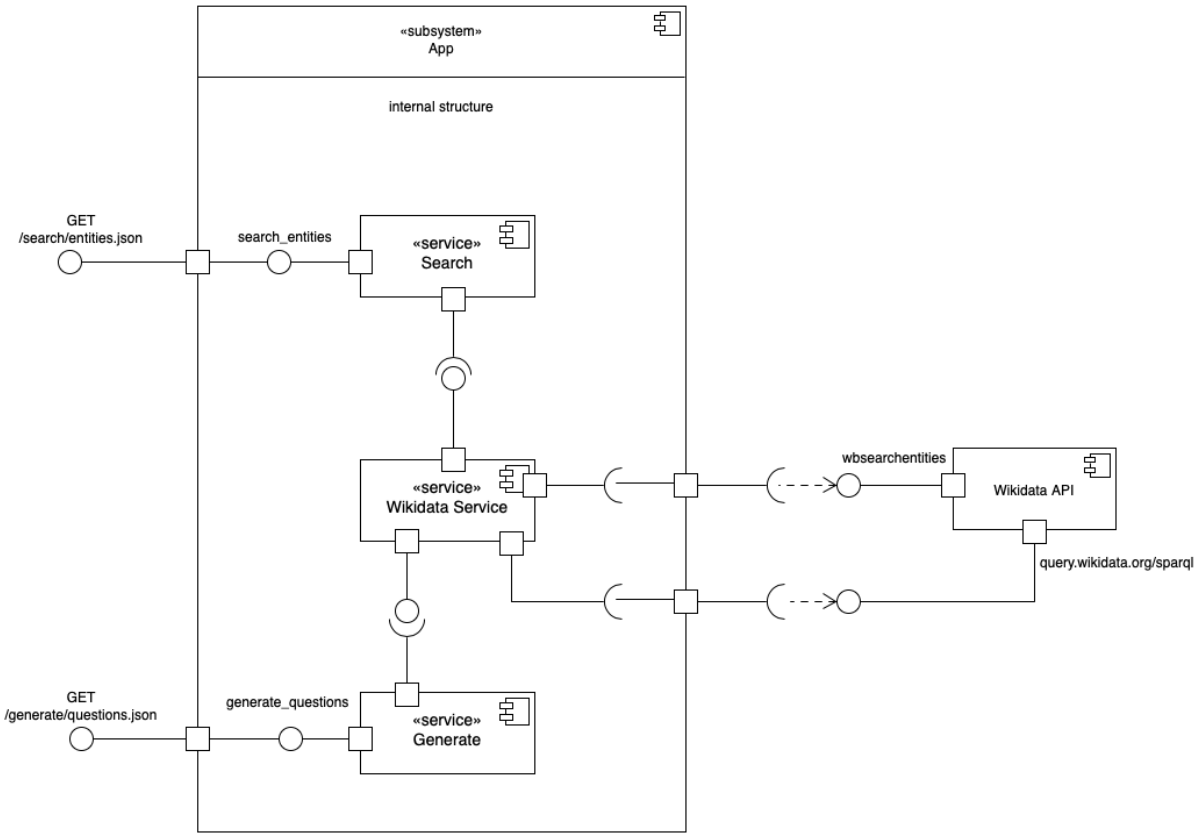

Figure 3.8: Component diagram of the system (2nd Sprint)

Sequence diagrams

The sequence diagram representing functionality for searching entities was updated due to refactoring with respect to the previous Sprint.

A new diagram modelling the process of generating the questions was added.

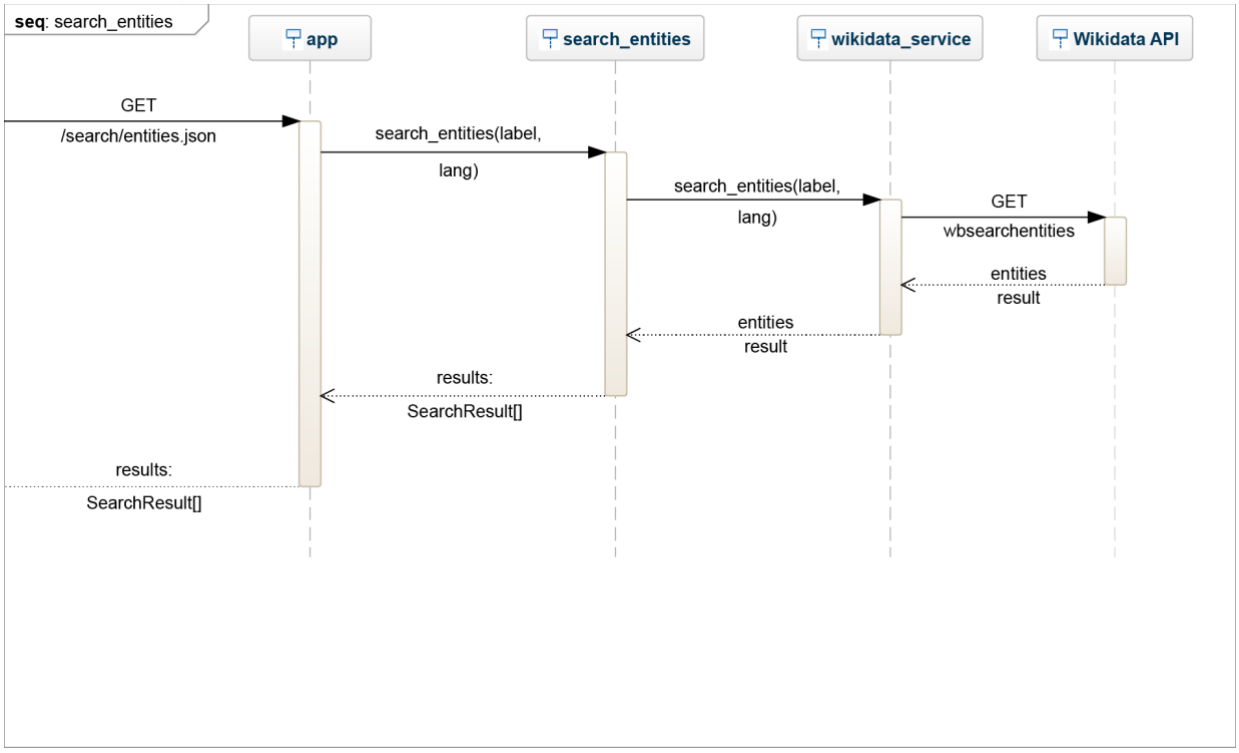

Figure 3.9: Sequence diagram of the functionality for searching entities (2nd Sprint)

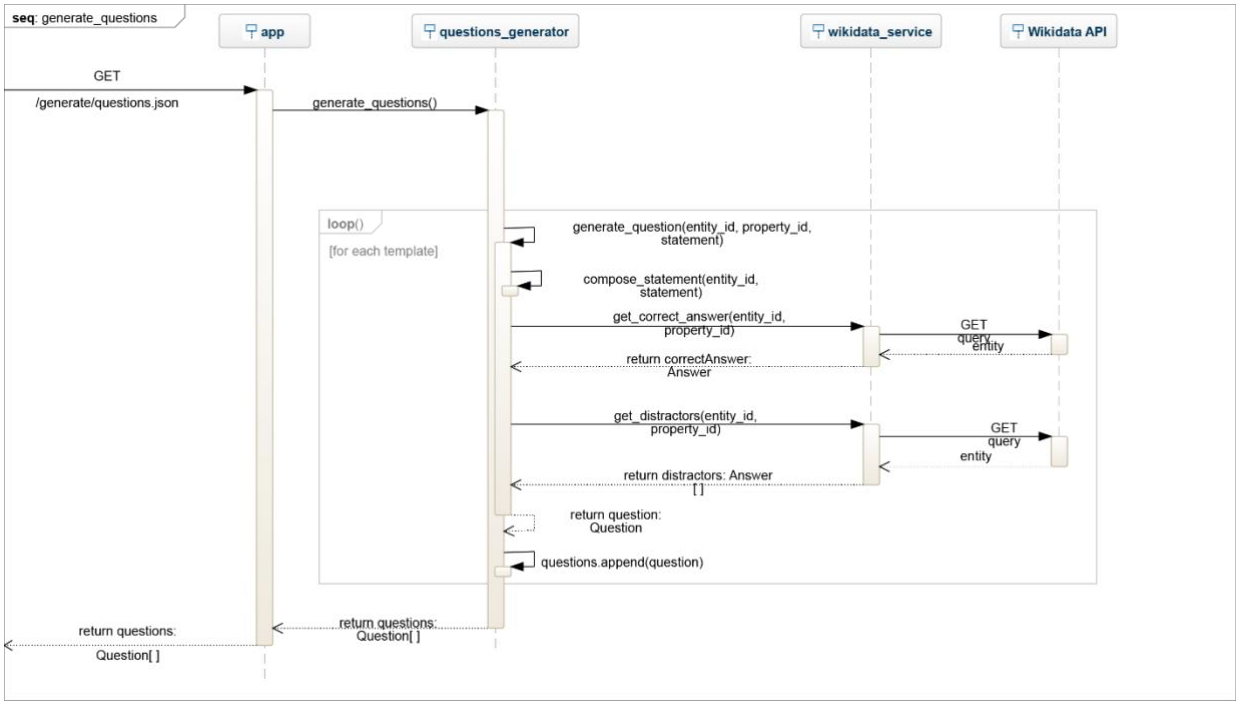

Figure 3.10: Sequence diagram of the functionality for generating questions (2nd Sprint)

Class diagrams

In this section, the different data types that appear in the other diagrams are shown.

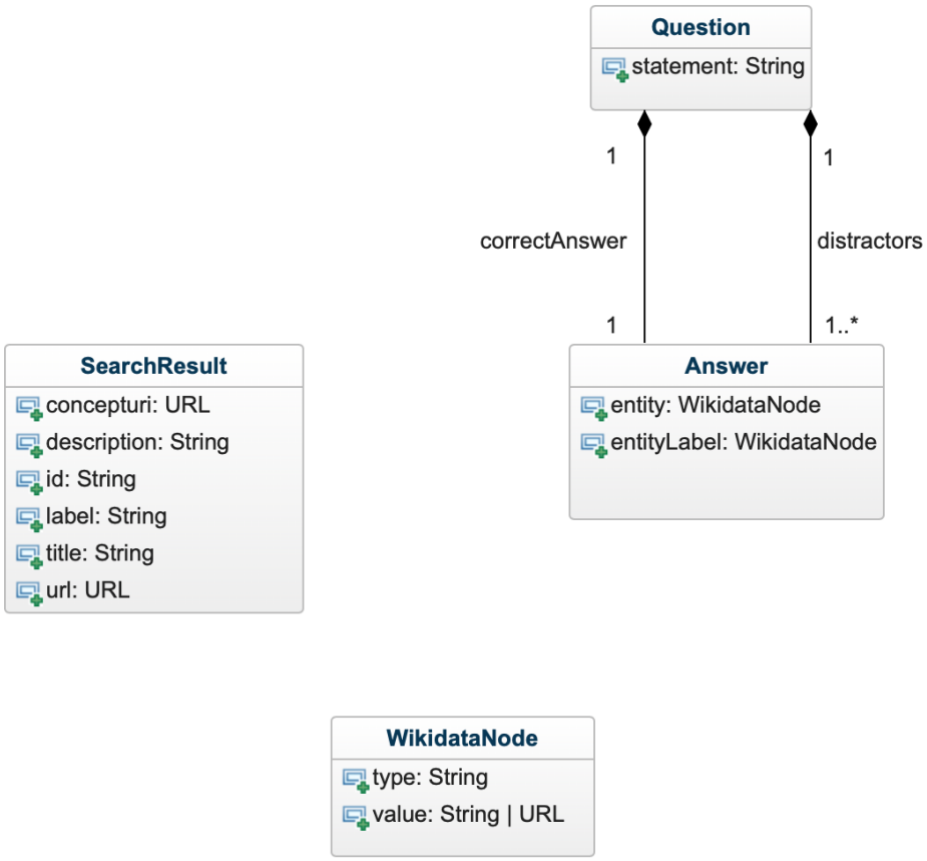

Figure 3.11: Class diagram of the data types (2nd Sprint)

3.2.2 Sprint backlog

The Sprint Backlog obtained in the second part of the Sprint Planning meeting is as follows.

Table 9: Sprint Backlog (2nd Sprint)

ID	Product Backlog item	Sprint Task	Initial Estimate of Effort
3	As a user, I want to be able to generate multiple-choice questions from a Wikidata entity.	Create the new endpoint for the generation functionality	3
		Create the SPARQL query for obtaining the correct answer.	5
		Create the SPARQL query for obtaining the distractors.	8
		Create the template of predicates for generating the questions.	5

3.2.3 Testing

In this section we will explain the different tests that were carried out for the system.

The tests designed are a combination of scripted and exploratory testing.
Scripted tests focus on checking the correct behavior of the API, whereas exploratory tests focus on analyzing the outputs to check their meaning.

3.2.3.1 Design

The approach for designing the test coverage items will be the minimized approach: create the minimum number of cases to cover valid equivalence classes and one test case to cover each of the invalid classes (to avoid masking of bugs).

Generate questions: Test coverage items

Table 10: Generate questions test coverage items (2nd Sprint)

ID	entity	limit	Expected output	Actual output
1	'Q312'	'2'	Generate at most 2 questions about Apple, Inc.	Generate at most 2 questions about Apple, Inc.
2	''	'2'	No questions generated.	No questions generated.
3	'Hola'	'2'	No questions generated.	No questions generated.
4	'Q312'	'Hola'	Generate questions about Apple, Inc without a limit.	Generate questions about Apple, Inc without a limit.
5	'Q312'	'0'	No questions generated.	Generated 1 question about Apple Inc.
6	'Q312'	'-1'	Generate questions about Apple, Inc without a limit.	Generate questions about Apple, Inc without a limit.

3.2.3.2 Bug reports

In this section we will discuss the errors identified from the tests carried out on the system.

These errors were fixed during the development to behave as expected.

Table 11: Bug reports (2nd Sprint)

Function	Error title	Error description
Generate questions	Time between requests	When making requests to Wikidata API the system must wait 1 second between them.
Generate questions	Correct answers can appear as distractors.	Generating geography questions, Belgium appeared as a distractor of the question: Which country borders with France?
Generate questions	When limit is 0 a question is generated.	When input 0 as limit of questions, 1 question is generated. Condition is not check before generating questions.

3.2.4 Tracking progress

The following table represents the daily updates of work remaining (measured in story points) during the sprint.

Table 12: Daily updates of work remaining (2nd Sprint)

Sprint task	Estimate at day										
	0	2	4	6	8	10	12	14	16	18	20
Create the new endpoint for the generation functionality	3	2	2	1	0	0	0	0	0	0	0
Create the SPARQL query for obtaining the correct answer.	5	5	5	5	5	5	2	0	0	0	0
Create the SPARQL query for obtaining the distractors.	8	8	8	8	8	5	5	5	3	1	0
Create the template of predicates for generating the questions.	5	5	5	5	5	8	8	8	8	8	8
Total	21	20	20	19	18	17	15	13	12	9	8

The following graph represents the values from the previous table as the **Burndown Chart**.

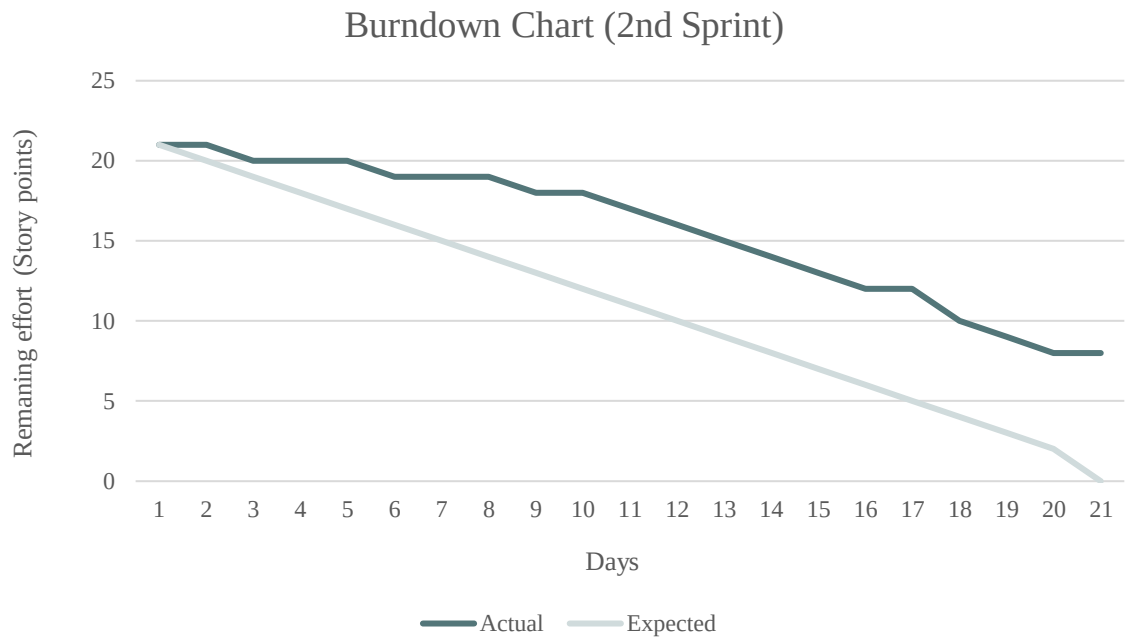

Figure 3.12: Burndown chart (2nd Sprint)

3.2.5 Backlog Grooming

During this Sprint, the remaining items in the Product Backlog were refined.

The knowledge obtained during the process of implementing the functionality of the Sprint lead the Team to re-estimate some of the items in the Product Backlog.

The functionality related to change the topic of the questions was more complex than initially expected, since the Product Owner decided to support several templates.

Additional requirements given by the Product Owner were also included for future Sprints.

Table 13: Backlog Grooming (2nd Sprint)

ID	Work item	Description	Estimate
5	As a user, I want to be able to choose the topic of the questions.	Given a label or identifier representing the topic, change the topic of the questions that the system generates (i.e. questions about 'Geography', 'Business'…)	13
4	As a user, I want to be able to choose the language of the generated questions.	Given a label or identifier representing the language, change the language of the questions that the system generates.	3
6	As a user, I want to be able to obtain metadata of the entities.	Within the generated questions, include metadata for the entities representing the correct answer and distractors.	5

3.2.6 Generation of the questions

Our goal is to compose questions about an entity. We can form a question about an entity by asking for the value of one of its properties.

For instance, in the previous example shown in Figure 1.3: Data model in Wikidata, if we were given the identifier 'Q42' representing the entity 'Douglas Adams' as the input for the program, a possible output of a question could be: 'Where was Douglas Adams educated?', and a correct answer would be 'St. John's College'.

This approach comes with a few problems:

- There are thousands of properties and not all of them are equally interesting.
- Transforming a Wikidata property into a question expressed in natural language is not trivial.

3.2.6.1 Templates

A workaround to the problems stated above can be defining *templates*.

Wikidata structures properties by categories. Adopting a similar approach, we have defined *templates*, that is, groups of properties that are related and are relevant for making questions about them.

A *template* also links each property with its representation as a question in natural language.

Using the same example as before, a possible *template* of questions applicable for 'Douglas Adams' could be *'People'*. This template would contain properties that are more likely to appear when the entity is a person. A very simple version of *'People'* could be:

```
{
    "P69" : {
        "en": "Where was :entity educated?",
        "es": "¿Dónde fue educado :entity?"
    },
    "P27" : {
        "en": "Which is the country of citizenship of :entity?",
        "es": "¿De qué país es :entity ciudadano?"
    }
}
```

This template would define enough information to generate up to two questions about an entity, in English or Spanish. The keys "P69"and "P27" correspond directly to the Wikidata identifier of those properties.

As it can be seen, using *templates* we have managed to solve both problems. Only relevant properties to the domain chosen by the user will be used for generating the questions, and the application provides well-formed questions expressed in natural language.

3.2.6.2 Generation process

User input

The user must give as input the identifier that represents the entity to generate questions about, as well as the identifier of the template that they want to use.

The process starts then by retrieving the questions defined inside the chosen template and processing each of them.

Remember that each question is associated with a property of Wikidata.

Completion of the statement

First, the question defined in the *template* as:

```
"Where was :entity educated?"
```

Is transformed into:

```
"Where was Douglas Adams educated?"
```

Correct answer

For each question defined in the template, we first obtain the correct answer. This is done by querying the value of the property for the given entity.

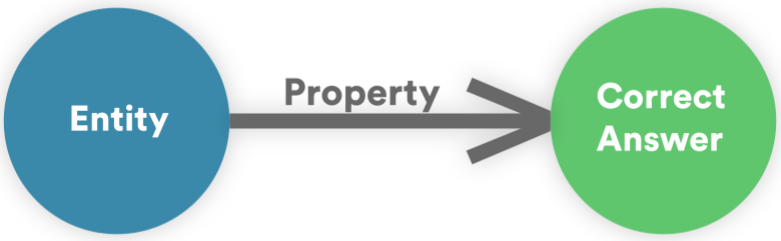

Figure 3.13: Correct answer semantics

Distractors

For obtaining the distractors we follow a similar approach. We want to find entities that appear in triples as values of the property that represents the question.

Figure 3.14: Distractor semantics

Note that in Figure 3.14, the node represented as '?' can be any entity except the entity that we are generating questions about, since that would mean that the value of the property is a correct answer instead of a distractor.

3.2.7 Sprint Review

This Sprint focused on implementing the functionality for generating the questions, which involved implementing the process described previously in Generation of the questions.

The Team encountered several problems during this Sprint, regarding the complexity of the generation process. Due to the great amount of properties in Wikidata, defining templates requires more effort than expected, this resulted on only one template composed of five properties being implemented for testing the generation of questions in this Sprint.

However, at the end of the Sprint, the system was able to generate multiple choice questions using this simple template, given the identifier representing a Wikidata entity.

Some of the feedback received from the Product Owner and the other stakeholders is that the system should also retrieve additional information and metadata to give context about the entities. Also, it became evident that doing an analysis of the different categories that the properties are classified into in Wikidata was necessary in order to implement meaningful templates.

All of this feedback was incorporated into the Product Backlog during the Sprint, as represented in Backlog Grooming.

3.2.8 Sprint Retrospective

During this Sprint Retrospective, the Team and Product Owner analyzed the information obtained during the daily updates of work remaining and the Burndown chart to reflect on the progress of the process.

As seen in Figure 3.12: Burndown chart (2nd Sprint), during this Sprint the Team underperformed the expectations, and couldn't finish the functionality in the Sprint Backlog in time.

However, the Team was able to cut down the functionality regarding the templates due to its complexity and effort required, and implemented a fully-working first version of the generation

of questions, with a simple template containing only five properties so that it would still be possible to test the functionality and deliver a working product.

The unfinished work items were re-incorporated into the Product Backlog to be included in a future Sprint.

The conclusions are that, the Team underestimated the effort required for the work items of this Sprint, which made them unable to complete everything in time.
However, the Team was still able to stick to the **Sprint Goal** previously agreed in Sprint Planning, and delivered a working product that received good feedback from the Product Owner and stakeholders.

3.3 Third Sprint

3.3.1 Sprint Planning

3.3.1.1 Part one

This first part of the Sprint Planning meeting consisted on the Team, Product Owner and the external stakeholders defining the priorities of the remaining items in the Product Backlog and setting the goal of the Sprint.

The Sprint Goal was defined as: “Refine the existing functionality for generating questions, adding metadata to the entities and the ability of changing the language”.

Reordered Product Backlog

Table 14: Reordered Product Backlog (3rd Sprint)

ID	Work item	Description	Estimate	Priority
5	As a user, I want to be able to choose the topic of the questions.	Given a label or identifier representing the topic, change the topic of the questions that the system generates (i.e. questions about ‘Geography’, ‘Business’…)	13	1
4	As a user, I want to be able to choose the language of the generated questions.	Given a label or identifier representing the language, change the language of the questions that the system generates.	3	2
6	As a user, I want to be able to obtain metadata of the entities.	Within the generated questions, include metadata for the entities representing the correct answer and distractors. Metadata includes description and representative image.	5	3

3.3.1.2 Part two

In this second part of the Sprint Planning, the Team and the Product Owner analyzed the functionality that is going to be implemented by creating the Sprint Backlog and modelling the system with UML diagrams.

Use case diagrams

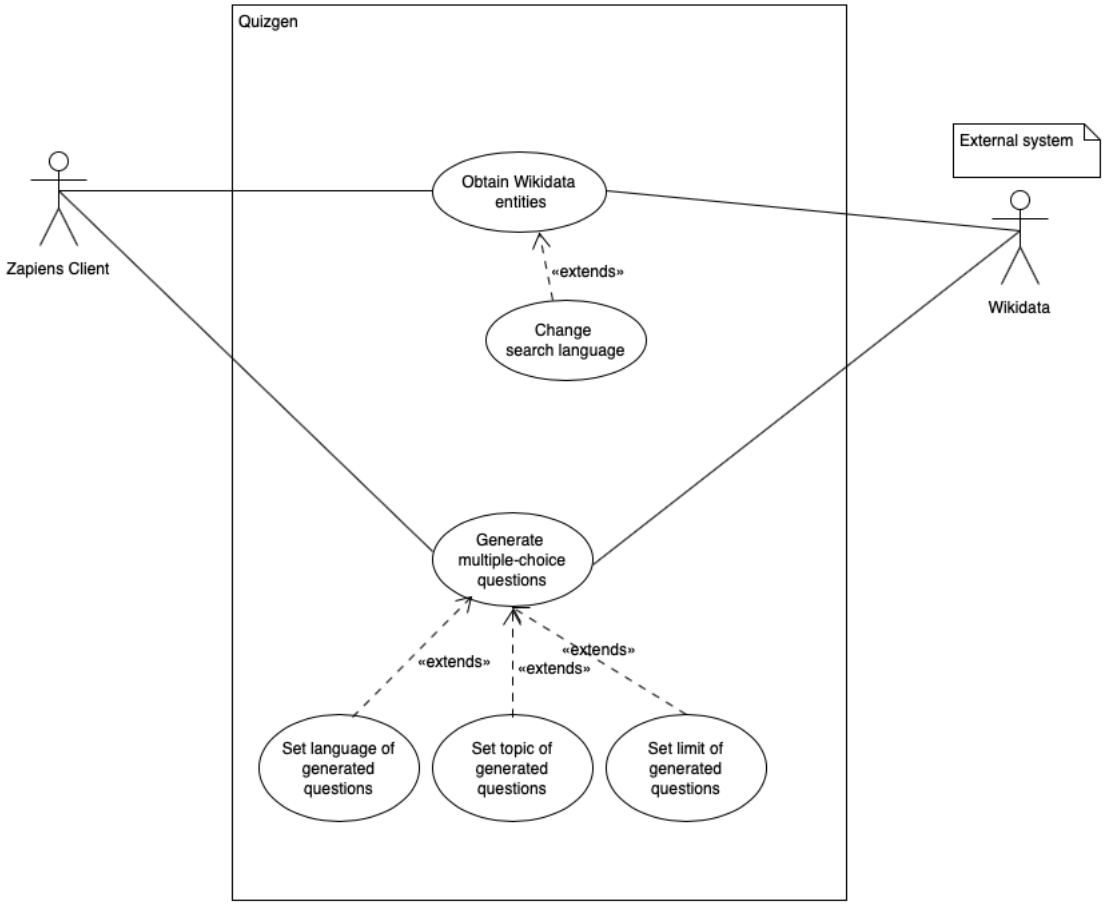

Figure 3.15: Use case diagram of the system (3rd Sprint)

Components diagrams

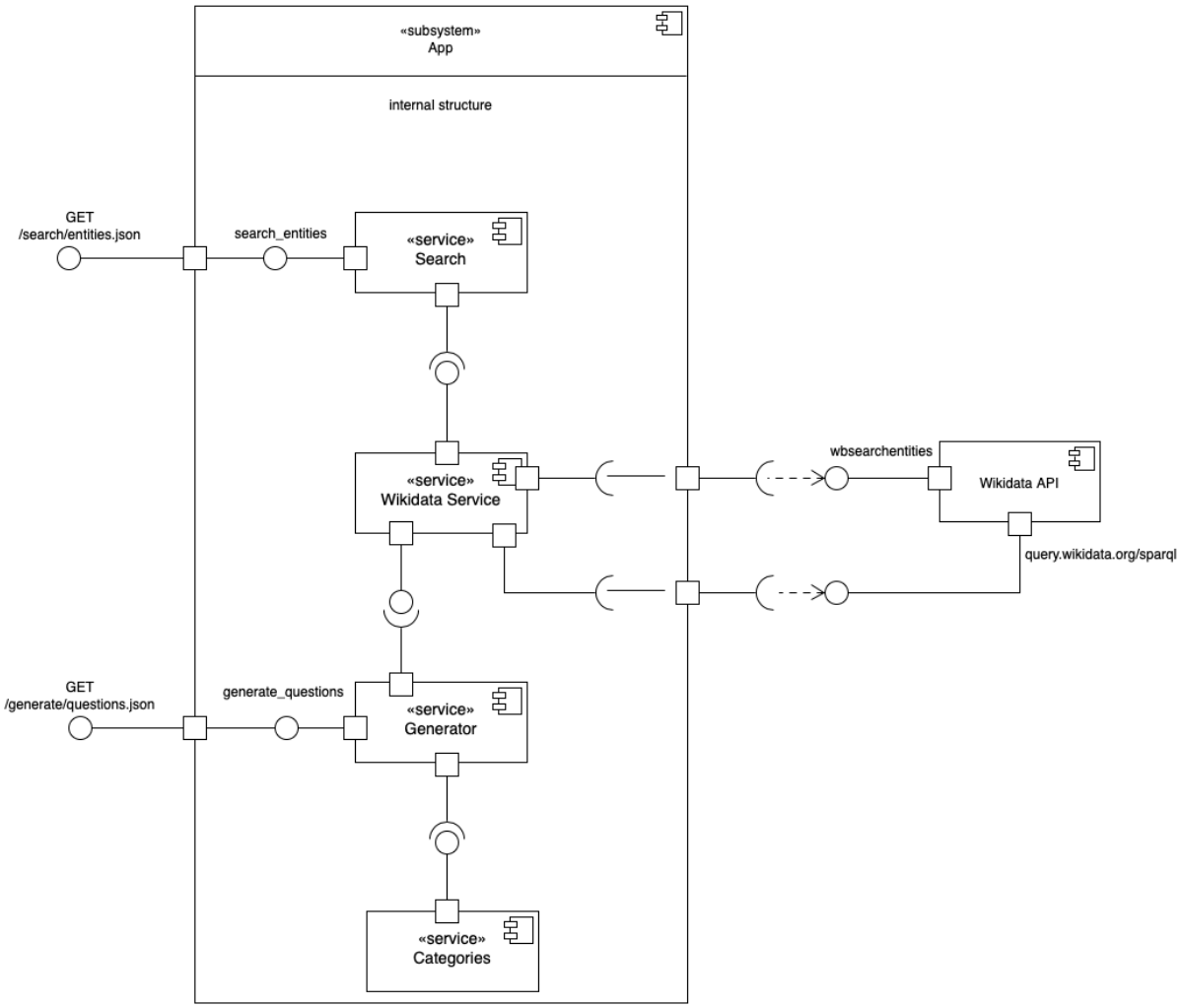

Figure 3.16: Component diagram of the system (3rd Sprint)

Sequence diagrams

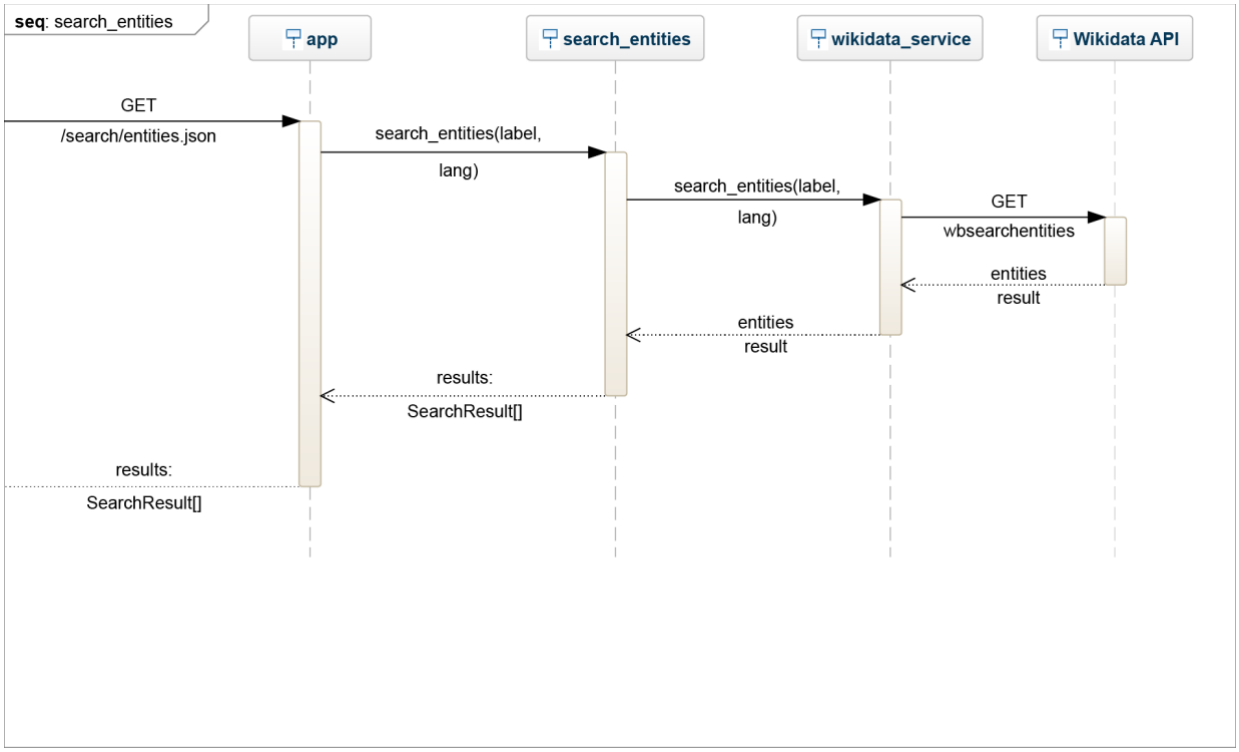

Figure 3.17: Sequence diagram of the functionality for searching entities (3rd Sprint)

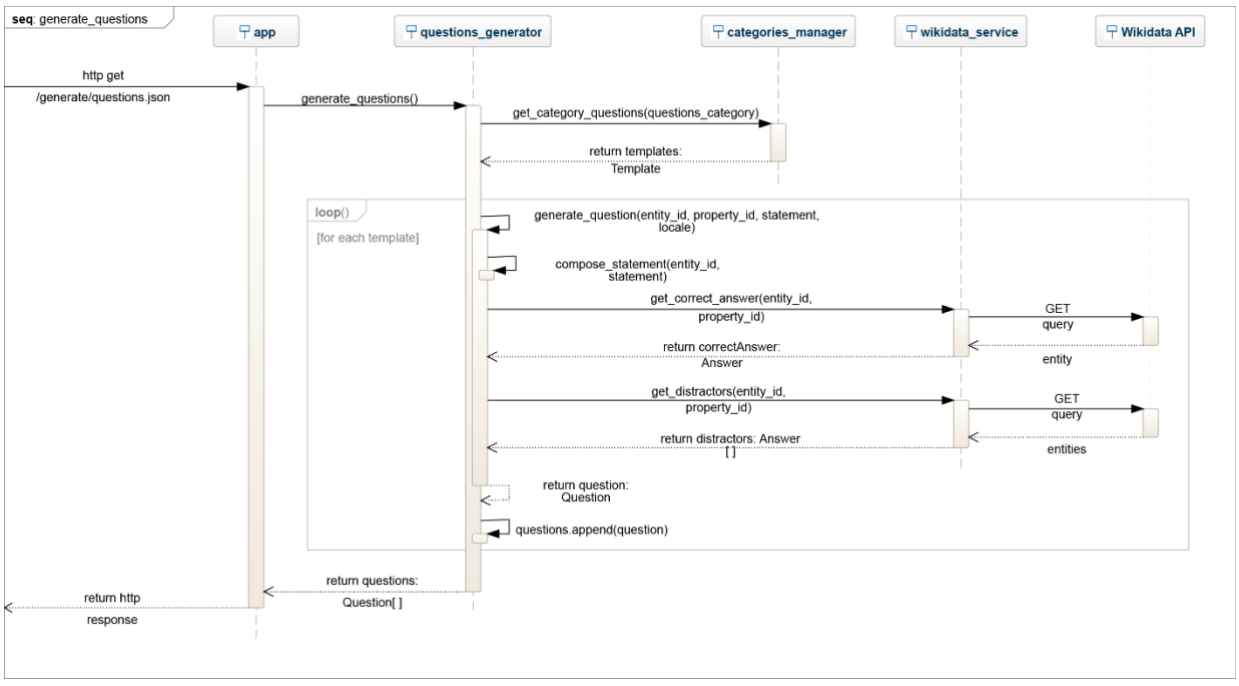

Figure 3.18: Sequence diagram of the functionality for generating questions (3rd Sprint)

Class diagrams

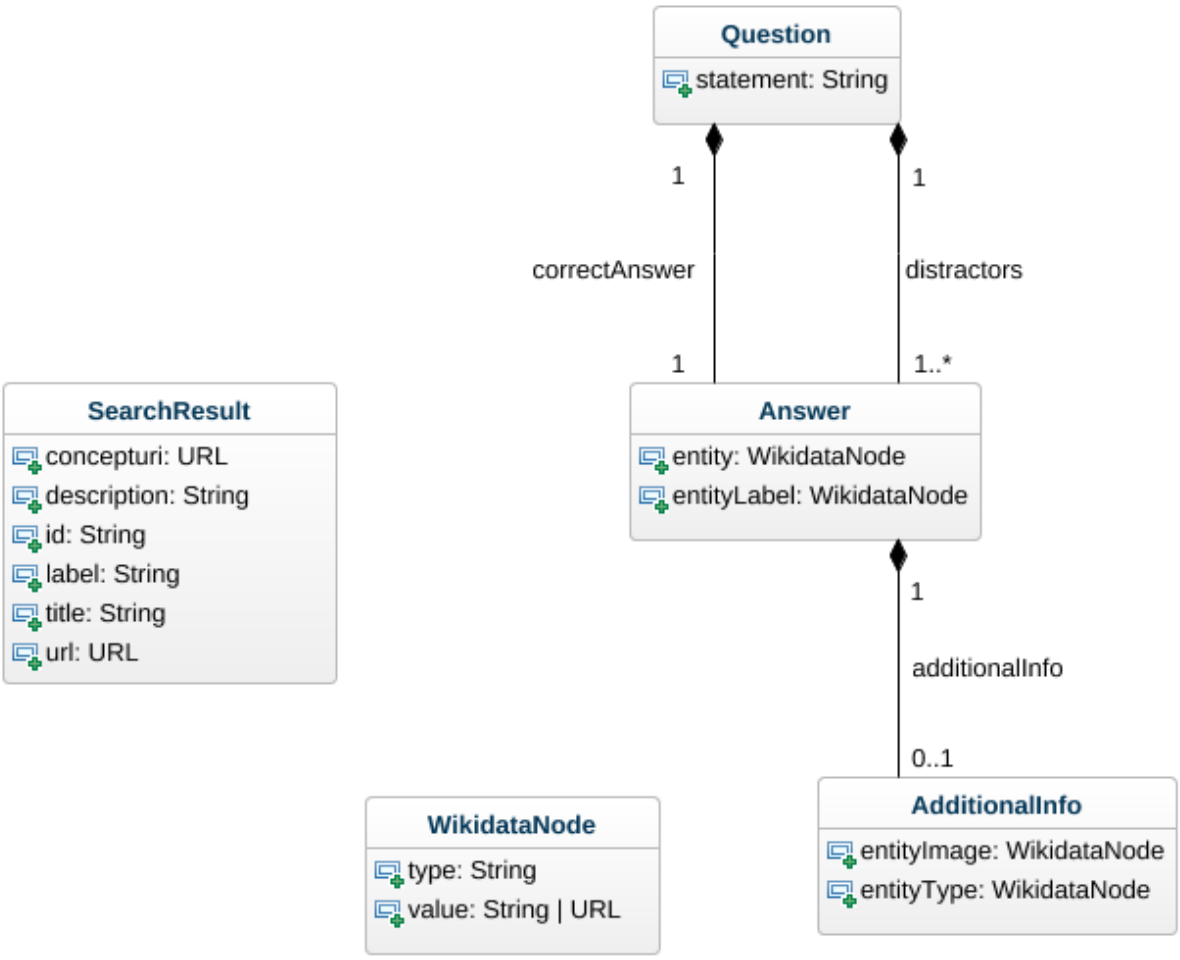

Figure 3.19: Class diagram of the data types of the system (3rd Sprint)

3.3.2 Sprint backlog

The Sprint Backlog obtained in the second part of the Sprint Planning meeting is as follows.

Table 15: Sprint Backlog (3rd Sprint)

ID	Product Backlog item	Sprint Task	Initial Estimate of Effort
5	As a user, I want to be able to choose the topic of the questions.	Modify generation endpoint to accept new parameter for choosing the template.	2
		Analyze different categories of Wikidata properties to create the different templates	3

		Create a template for questions about ‘Business’ and another about ‘Geography’	3
		Adapt the system to manage several templates.	5
4	As a user, I want to be able to choose the language of the generated questions.	Modify generation endpoint to accept new parameter for choosing the template.	1
		Adapt the system to manage several languages.	2
6	As a user, I want to be able to obtain metadata of the entities.	Modify the queries to include the description of the entities.	2
		Modify the queries to include the representative image of the entities.	3

3.3.3 Testing

In this section we will explain the different tests that were carried out for the system.

The tests designed are a combination of scripted and exploratory testing.
Scripted tests focus on checking the correct behavior of the API, whereas exploratory tests focus on analyzing the outputs to check their meaning.

3.3.3.1 Design

The approach for designing the test coverage items will be the minimized approach: create the minimum number of cases to cover valid equivalence classes and one test case to cover each of the invalid classes (to avoid masking of bugs).

Generate questions: Test coverage items

Table 16: Generate questions test coverage items (3rd Sprint)

ID	entity	template	limit	lang	Expected output	Actual output
1	'Q312'	'Q18608993'	'2'	'es'	Generate at most 2 questions about Apple, Inc from the Business template in Spanish.	Generate at most 2 questions about Apple, Inc from the Business template in Spanish.
2	'Q312'	'Q18608993'	'0'	'en'	No questions generated.	No questions generated.
3	''	'Q18608993'	'2'	'fr'	No questions generated.	No questions generated.
4	'Hola'	'Q18608993'	'2'	'it'	No questions generated.	No questions generated.
5	'Q312'	'Hola'	'2'	'pt'	Show error invalid template.	Internal server error
6	'Q312'	''	'2'	'es'	Show error invalid template.	Internal server error

7	‘Q312’	‘Q18608993’	‘Hola’	‘es’	Generate questions about Apple, Inc from the Business template without a limit.	Generate questions about Apple, Inc from the Business template without a limit.
8	‘Q312’	‘Q18608993’	‘-1’	‘es’	Generate questions about Apple, Inc from the Business template without a limit.	Generate questions about Apple, Inc from the Business template without a limit.
9	‘Q312’	‘Q18608993’	‘2’	‘’	Generate at most 2 questions about Apple, Inc from the Business template in English.	Generate at most 2 questions about Apple, Inc from the Business template in English.
10	‘Q312’	‘Q18608993’	‘2’	‘ddaskq’	Generate at most 2 questions about Apple, Inc from the Business template in English.	Generate at most 2 questions about Apple, Inc from the Business template in English.

3.3.3.2 Bug reports

In this section we will discuss the errors identified from the tests carried out on the system.

These errors were fixed during the development to behave as expected.

Table 17: Bug reports (3rd Sprint)

Function	Error title	Error description
Generate questions	Invalid template id makes the system crash	When entering an invalid or empty identifier as the template, the system crash showing 'Internal server error'.

3.3.4 Tracking progress

Table 18: Daily updates of work remaining (3rd Sprint)

Sprint task	Estimate at day										
	0	2	4	6	8	10	12	14	16	18	20
Modify generation endpoint to accept new parameter for choosing the template.	2	1	0	0	0	0	0	0	0	0	0
Analyze different categories of Wikidata properties to create the different templates	3	3	3	3	2	1	0	0	0	0	0
Create a template for questions about 'Business' and another about 'Geography'	3	3	3	3	3	3	2	0	0	0	0
Adapt the system to manage several templates.	5	5	5	5	5	5	5	2	0	0	0
Modify generation endpoint to accept new parameter for choosing the template.	1	1	1	1	1	1	1	1	0	0	0
Adapt the system to manage several languages.	2	2	2	2	2	2	2	2	0	0	0
Modify the queries to include the description of the entities.	2	2	2	2	2	2	2	2	0	0	0
Modify the queries to include the representative image of the entities.	3	3	3	3	3	3	3	3	3	0	0
Total	21	20	19	19	18	17	15	10	3	0	0

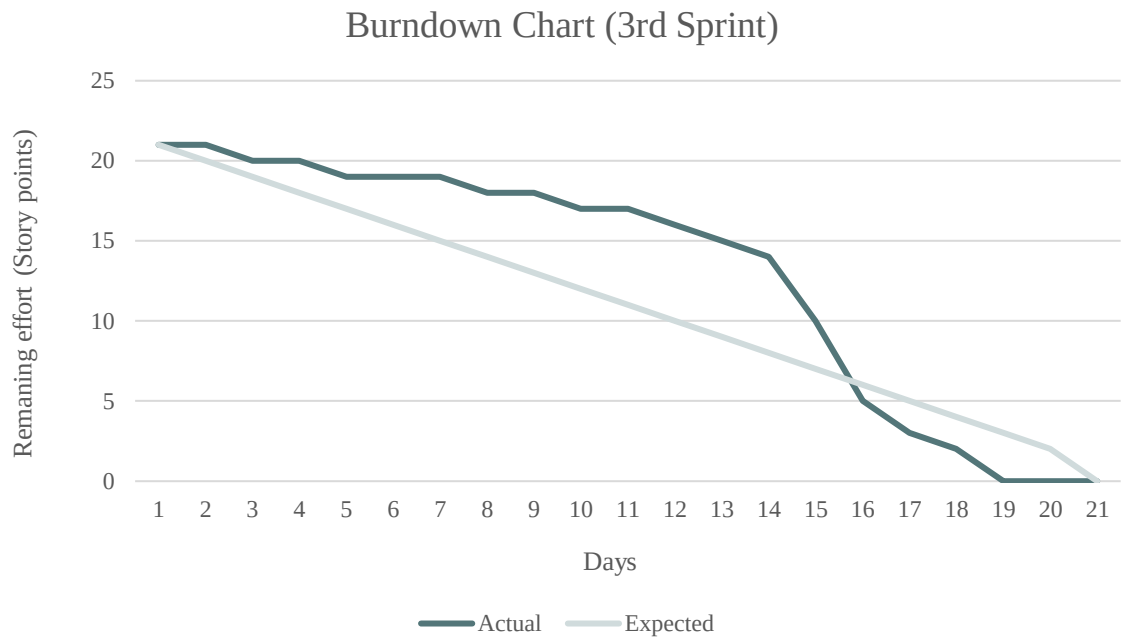

Figure 3.20: Burndown chart (3rd Sprint)

3.3.5 Sprint Review

This Sprint focused on improving the functionality for generating the questions.

The result of this Sprint was a system that allows users to generate multiple-choice questions from a concept or entity represented by a string expressed in natural language.

The system first allows to obtain a list of Wikidata entities related to the label entered by the user.

From this list of entities, the user can check the description and title of each, and even check its Wikidata page (URL) to obtain more information and verify which is the one they are looking for.

Once the users identify the Wikidata entity that they were looking for, they can choose to generate questions about this entity related to a specific topic (e.g. geography, business) and in a specific language.

Finally, they obtain a JSON file containing all the questions generated for that entity.

The feedback obtained by the Product Owner and the other stakeholders of the product obtained in this Sprint was positive, and, the Team and Product Owner agreed to proceed with the Transition phase.

3.3.6 Sprint Retrospective

During this Sprint Retrospective, the Team and Product Owner analyzed the information obtained during the daily updates of work remaining and the Burndown chart to reflect on the progress of the process.

The Team had to tackle already known problems, so the experience obtained in previous Sprints helped smooth the development process, and even although the progress at the beginning was worse than expected, they managed to finish implementing and validating all the functionality ahead of schedule.

As seen in Figure 3.20: Burndown chart (3rd Sprint), at the beginning of this Sprint the Team was underperforming compared to the expectations. This was due to an increase in the need of collaboration between the Team and the Product Owner to validate the functionality and create the templates of questions.

However, during the second half of the Sprint, the tasks required less participation of the Product Owner and the Team managed to speed up again the process.

The conclusions are that, the Team did a good job of anticipating the risks of this Sprint by previously doing an analysis on Wikidata categories for the properties, which allowed them to foresee the problems that may occur due to the great number of Wikidata properties.

Also, the Team was able to complete the **Sprint Goal** previously agreed with the Product Owner and delivered a working product that received good feedback from the Product Owner and stakeholders.

4 Transition

After the third Sprint, the Product Owner considered that the system was ready to go into the Transition phase, creating the different manuals for the correct use of the system.

4.1 Installation guide

4.1.1 POSIX platform (Mac, Linux and more)

From the command line interface, execute the following commands.

4.1.1.1 Setting up the environment

Head into the project root folder and create a Python virtual environment.
Then, activate that environment.

```
cd quizgen
python3 -m venv .venv
source .venv/bin/activate
```

4.1.1.2 Installing dependencies

Install the dependencies required to run the system that are specified inside the requirements.txt file.

```
pip install -r requirements.txt
```

4.1.1.3 Running

Export the environment variable used to specify how to load the application and run it.

```
export FLASK_APP=app
flask run
```

The command line interface should show several messages, prompting that it is running and the URL for making the http requests.

4.1.2 Windows platform

Using PowerShell in Windows, execute the following commands.

4.1.2.1 Setting up the environment

Head into the project root folder and create a Python virtual environment.
Then, activate that environment.

```
cd quizgen
python -m venv .\.venv
.\.venv\Scripts\activate.bat
```

4.1.2.2 Installing dependencies

Install the dependencies required to run the system that are specified inside the requirements.txt file.

```
pip install -r requirements.txt
```

4.1.2.3 Running

Export the environment variable used to specify how to load the application and run it.

```
set FLASK_APP=app
flask run
```

4.2 User manual

4.2.1 API endpoints

Once the system is running, it is possible to execute the functionality by making HTTP requests to the URL exposed.

4.2.1.1 Search by label

- `/search/entities.json`

Takes a string representing an entity and returns a json file with the associated entities.

Parameters

Table 19: Parameters for the search endpoint

Name	Required	Description	Default value
label	Required	String used to search the entities.	
lang	Optional	Language used to search the entities and obtain the information. Supported: English ('en'), Spanish ('es'), French ('fr'), Italian ('it') and Portuguese ('pt').	en

Example

Search of entities given the label 'Madrid' in Spanish:
`/search/entities.json?label=Madrid&lang=es`

4.2.1.2 Generate questions

- `/search/entities.json`

Takes several parameters and returns a json file containing the generated questions for the given entity and category.

Parameters

Table 20: Parameters for the generate questions endpoint

Name	Required	Description	Default value
entity	Required	String used to search the entities.	
category	Required	String identifier associated to the template used to generate the questions. See Available templates.	
limit	Optional	Number, limit of questions to generate.	None, every available question in the template will be generated.
lang	Optional	Language used to search the entities and obtain the information. Supported: English (‘en’), Spanish (‘es’), French (‘fr’), Italian (‘it’) and Portuguese (‘pt’).	en

Example

Generate only one question in Spanish about the entity Q312 (Apple Inc.), using the template : ***`/generate/questions.json?entity=Q312&category=Q18608993&limit=1&lang=es`***

4.2.2 Available templates

4.2.2.1 Q18608993 (Companies)

Table 21: Companies template

Propiedad	Nombre	Statement (Spanish)	Statement (English)
P571	Inception	¿Cúando se fundó %s?	**When was %s founded?**
P159	Headquarters location	¿Dónde están las oficinas de %s?	**Where are %s headquarters located?**
P17	Country	¿En qué país se encuentra %s?	**In which country is %s based?**
P154	Logo	¿Cuál es el logo de %s?	**Which is the logo of %s?**
P1454	Legal form	What is %s legal form?	**Legal form**
P452	Industry	¿Cuál es la industria de %s?	**Which is the industry of %s?**
P127	Owned by	¿Quién es el propietario de %s?	**Who owns %s?**
P414	Stock Exchange	¿En qué mercado cotiza %s?	**In which stock exchange is %s traded?**
P740	Location of formation	¿Dónde se originó %s?	**Where was %s created?**
P856	Official website	Which is %s official website?	**¿Cuál es el sitio web oficial de %s?**
P1448	Official name	¿Cuál es el nombre oficial de %s?	**What's %s official name?**
P1128	Number of employees	¿Cuánta gente trabaja en %s?	**How many people work at %s?**
P2403	Total assets	What's the value of assets held by %s?	**¿Cuál es el valor de los activos de %s?**
P2138	Total liabilities	¿Cuál es el valor de los pasivos de %s?	**What's the value of liabilities held by %s?**
P2137	Total equity	¿Cuál es el valor del capital propio de %s?	**What's the value of equity of %s?**
P2226	Market capitalization	¿Cuál es el valor de la capitalización bursátil de %s?	**What's the total value of issued shares by %s?**
P1830	Owner of	¿Cuál de las siguientes entidades es propiedad de %s?	**Which entity is owned by %s?**
P169	CEO	¿Quién es el CEO de %s?	**Who is the CEO of %s?**
P1789	COO	¿Quién es el Chief Operating Officer (COO) de %s?	**Who is the Chief Operating Officer (COO) of %s?**
P112	Founded by	¿Quién fundó %s?	**Who is the founder of %s?**
P1037	Director/Manager	¿Quién es el director de %s?	**Who is the director/manager of %s?**

P3320	**Board Member**	**¿Quién es un miembro del consejo de %s?**	**Who is a member of the board of %s?**

4.2.2.2 Q52511956 (Geography)

Table 22: Geography template

Property	Name	Statement (Spanish)	Statement (English)
P30	Continent	¿En qué continente se encuentra %s?	**In which continent is %s located?**
P36	Capital	¿Cuál es la capital de %s?	**Which is the capital of %s?**
P47	Border	%s hace frontera con…	**%s shares border with…**
P1549	Demonym	¿Cómo se denominan los habitantes de %s?	**How are the people from %s called?**
P17	Country	¿En qué país se encuentra %s?	**In which country is %s located?**
P6	**Head of government**	**Who is the head of government of %s?**	**¿Quién es el jefe de gobierno de %s?**

5 Future improvements

Even although the obtained system meets the needs of the Product Owner and its clients, some improvements were also identified for possible future releases.

On the following subsections we will briefly discuss these ways of improving the system and how they could be implemented.

5.1 Query performance

Query performance, especially for distractors is the bottleneck of the system.

Part of this problem lays on Wikidata's required waiting time between requests, as well as on the optimization of the queries used to retrieve the correct answers and the distractors.

5.2 Selection of properties

Currently the selection of properties for the generated questions must be done by the user, by selecting the template of questions that fits best the domain of the entity they chose.

This also causes the problem that the entity may not have a value defined for some properties of the selected template, resulting in some questions not being generated.

As explained in Generation process, an alternative could be to dynamically choose the Wikidata properties that appear for the given entity.

5.3 Automatic natural language generation

The current approach using templates requires each property to be associated to its representation as a question in natural language.

This is not flexible for adding new templates, extending the existing ones or adding support for new languages, since all of those actions would require to manually implement the new properties.

A study on how to convert semantic predicates into questions expressed in natural language should be conducted, to evaluate whether or not it would be possible to automate this process.

5.4 Progressive difficulty

The process for obtaining the distractors (see Generation process) selects entities that already have a value for the given property.

This ensures that the distractors are similar to the correct answer. However, this simple approach could be extended, analyzing the semantic distance between entities to vary the difficulty of the questions.

6 Conclusions

This document has defined the process taken to obtain the final system delivered to the software company Zapiens.

It is structured to show how the work was developed and tracked, from the inception of the system, its construction and its later transition, until it was delivered.

The final system obtained is a fully-working REST API that provides the functionality necessary to generate multiple-choice questions using knowledge graphs.

Some problems were encountered during the development especially during the inception and the initial phases of the construction, as the technologies used were completely new for the development team.

However, the final system meets the needs of the Product Owner and was accepted by the company, which after the product was delivered proceeded to integrate it into their system.

7 Bibliography

Visual Paradigm, n.d. *Classical Project Management vs Agile Project Management.* [Online]
Available at: https://www.visual-paradigm.com/scrum/classical-vs-agile-project-management/
[Accessed 21 May 2020].

Schwaber, K. & Sutherland, J., 2017. *The Scrum Guide.* [Online]
Available at: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100
[Accessed 21 May 2020].

S. G., 2015. *Chaos Report 2015.* [Online]
Available at: https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf
[Accessed 21 May 2020].

Ambler, S. W. & Lines, M., 2016. *The Disciplined Agile Process Decision Framework.* [Online]
Available at:
https://www.disciplinedagileconsortium.org/resources/Documents/TheDAFramework.pdf
[Accessed 21 May 2020].

Deemer, P., Benefield, G., Larman, C. & Vodde, B., 2012. *The Scrum Primer.* 2 ed. s.l.:s.n.

Cohn, M., 2005. *Agile Estimating and Planning.* s.l.:s.n.

Faizan, A. & Lohmann, S., 2018. *Automatic Generation of Multiple Choice Questions from Slide Content using Linked Data.* [Online]
Available at: https://dl.acm.org/doi/10.1145/3227609.3227656
[Accessed October 2019].

Rodríguez Rocha, O. & Faron Zucker, C., 2018. *Automatic Generation of Quizzes from DBpedia According to Educational Standards.* [Online]
Available at: https://hal.inria.fr/hal-01758737/document
[Accessed October 2019].

Seyler, D., Yahya, M. & Berberich, K., 2016. *Knowledge Questions from Knowledge Graphs.* [Online]
Available at: https://hal.inria.fr/hal-01758737/document
[Accessed October 2019].

Jelenkovic, F. & Tosic, M., 2013. *Semantic Multiple-Choice Question Generation and Concept-Based Assessment.* [Online]
Available at: https://www.researchgate.net/publication/290168699_Semantic_Multiple-Choice_Question_Generation_and_Concept-Based_Assessment

Wikidata, n.d. *Wikidata Query Service/User Manual.* [Online]
Available at:
https://www.mediawiki.org/wiki/Wikidata_Query_Service/User_Manual#SPARQL_endpoint

Wikidata, n.d. *Wikidata:Data access.* [Online]
Available at: https://www.wikidata.org/wiki/Wikidata:Data_access/es

Wikidata, n.d. *Wikidata - PyPI.* [Online]
Available at: https://pypi.org/project/Wikidata/