NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

JOSÉ DIOGO MOTA JUNCEIRA MENDES DE OLIVEIRA

Bachelor in Computer Science and Engineering

AN INTERACTIVE PLATFORM FOR REPRESENTING, INTERLINKING AND ANALYSING CONTENT ON GLASS ART ON THE WEB

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
March, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

AN INTERACTIVE PLATFORM FOR REPRESENTING, INTERLINKING AND ANALYSING CONTENT ON GLASS ART ON THE WEB

JOSÉ DIOGO MOTA JUNCEIRA MENDES DE OLIVEIRA

Bachelor in Computer Science and Engineering

Adviser: Matthias Knorr
Assistant Professor, NOVA University Lisbon

Co-adviser: Armanda Rodrigues
Associate Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
March, 2023

An interactive Platform for Representing, Interlinking and Analysing Content on Glass Art on the Web

Acknowledgements

I have received immense support throughout the development of this thesis, and I would like to thank the people who helped and made me achieve this goal. Firstly, I would like to thank Professor Matthias Knorr for his constant availability to share his knowledge and aid in the development of this thesis. I would also like to thank Professor Armanda Rodrigues for her help and suggestions that greatly improved the quality of this project. I would of course like to thank my friends and family for their everlasting support and motivation, along with some of their help when testing the end result of this thesis.

Finally, I want to thank VICARTE, as this thesis wouldn't have been possible without their knowledge and help.

Abstract

Portugal has a rich heritage of stained glass. However, the inventories of historical collections in stained glass are commonly stored in catalogs/printed books which makes maintenance difficult and provides a considerable obstacle for integrating information from different collections (within and across different countries). Making use of Semantic Web technology, such as Linked Open Data, in this thesis, we describe how we created a platform that allows one to store and interact with a comprehensive database of collections of stained glass (including images, iconographic and technical information, as well as results of previous studies on their history, techniques of production, and conservation condition), and to enable the visualization and querying of such data on stained glass collections in the world, interlinkable with relevant information from other linked open data repositories. For this purpose, several types of interfaces for different objectives and devices will be discussed, including an integrated interactive map of the relevant regions, informing the user on the location of the stained glass collections, as well as specific interactive interfaces for visualizing and studying specific relevant pieces. All of the information stored in the repository is processed and stored using Semantic Web and Linked Data principles, where users are able to access it via an online Wiki, where information can be edited by specialists or viewed by any user. As previously mentioned, we created an online platform that was tested both locally and in a public environment that lets specialist users create and edit stained glass-related pages in an intuitive way and also lets all users view each page's content. After this, we tested the resulting platform with two distinct groups of users, one being regular users and specialists users, and we received their feedback regarding several aspects of the platform, such as the level of difficulty when creating new stained glass pages and how well structured each page is within the platform. This dissertation was finished with the collaboration of NOVA's School of Science and Technology investigation unit VICARTE and with Mosteiro de Santa Maria da Vitória (Batalha).

Keywords: Linked Open Data, Semantic Web, Glass Art, Interlinking, Database, RDF, SPARQL, Repositories, Standardization, Semantic Wiki

Resumo

Portugal tem uma herança rica em vitrais. No entanto, os inventários de coleções históricas de vitrais são normalmente guardadas em catálogos ou em livros, o que torna a sua manutenção mais árdua e levanta alguns obstáculos na integração de informação de diferentes coleções (dentro e ao redor de países diferentes). Utilizando tecnologias de Semantic Web, como Linked Open Data, neste projeto criámos uma plataforma que permite o armazenamento e interação com uma base de dados compreensiva relacionada com as coleções de vitrais nacionais (incluindo imagens, informação iconográfica e técnica, assim como resultados de estudos anteriores na sua história, técnicas de produção, e conservação da sua condição), e para habilitar a visualização e a inquirição de tais dados da arte em vidro em Portugal, interligada com informação relevante de outros repositórios de linked open data. Para este propósito, serão considerados diversos tipos de interfaces com funcionalidades e ferramentas, incluindo um mapa interativo integrado que mostra regiões relevantes, informando deste modo o utilizador em relação à localização das coleções de arte de vidro, assim como interfaces interativas mais específicas que ajudam a visualizar e a estudar peças específicas. Toda a informação guardada nos repoitórios vai ser processada e armazenada com a utilização de Semantic Web e princípios de Linked Data, onde os utilizadores vão poder aceder à mesma através de uma Wiki online, onde a informação pode ser editada por especialistas e vista por qualquer utilizador. Criámos então uma plataforma que permite aos especialistas registados criar e editar páginas de vitrais de maneira intuitiva e permite a todos os seus utilizadores visualizar os conteúdos de cada página. Após isto, testámos a plataform com dois grupos de utilizadores, sendo eles os especialistas e os não-especialistas, e recebemos feedback relativamente a diferentes aspetos da platforma, como o nível de dificuldade de criar páginas na plataforma e a qualidade da estrutura da plataforma. Esta dissertação foi feita em colaboração com a unidade de investigação da FCT-NOVA VICARTE - Vidro e cerâmica para as artes - e com o Mosteiro de Santa Maria da Vitória (Batalha).

Palavras-chave: Linked Open Data, Semantic Web, Glass Art, Interlinking, Database, RDF, SPARQL

Contents

[1] https://corpusvitrearum.de/
[2] https://www.cvma.ac.uk/jsp/index.jsp
[3] https://vitrosearch.ch/en

List of Figures

List of Tables

List of Listings

1

Introduction

1.1 Motivation

Glass art has been used as a form of expression for many centuries, dating back to the times of the Roman Empire in the first century BC. Some of the earliest and most practical examples of glass art include different vessels such as goblets and pitchers. Artists would decorate these vessels by painting or etching on their surfaces. Techniques such as millefiori, a form of decorated glass art, date back to this period. Glassblowers and their techniques spread throughout the empire from Italy to Egypt in many different forms. Moving through history from the Middle Ages to the Renaissance, glass art was produced throughout Europe.

Glassblowing techniques continued to evolve with heavy influence from the Franks and Byzantine glassworkers. Byzantine glass artists would decorate their pieces with Christian and Jewish symbols in Jerusalem in the 6th and 7th centuries. This time period also birthed another important form of glass art, stained glass. Churches and monasteries employed many glass artists to decorate their buildings with stained glass windows during this time. This practice would dominate cathedrals and other religious buildings for hundreds of years to come. At the same time period in the Middle East, Syria dominated the glass industry. The major centers of glass production during the Islamic Period include Raqqa, Aleppo, and Damascus. Glass art from this period is transparent colorless glass and gilded glass [15].

It is clear to see that there is a great deal of history and complexity behind the craftsmanship of glass art pieces. Both modern and antique glass art has caught the eye of many, creating many passionate people on the subject. In the case of Portugal and many countries in Europe which have a rich religious background (namely Christian) are bursting with pieces of glass art. Since all information regarding stained glass in Portugal is in a physical format and can be mostly only found in books, the necessity of storing this information in an online repository arose. In addition to this, there already have been created platforms with this intent that use Linked Open Data. There are benefits of the implementation of Linked Open Data, such as allowing for the interlinking of data between different data sources in an automatic way, such that each endpoint can retrieve

data from the other. For this reason, it can also be very beneficial to implement this in an online platform for Portuguese stained glass-related data. What is going to be different from our platform to the already created ones is how data is created in our platform, as we intend to let specialist users create stained glass pages without having to know anything about Semantic Web or Linked Open Data. With this, there are still important challenges in achieving a concise solution, having to opt through some difficult choices, in terms of tools and syntax to be used, depending on the wanted final result of the platform that is planned to be built, as it will be clear to see later on.

1.2 Context

A major factor behind the beginning of this idea lies in the collaboration that is being done between NOVA School of Science and Technology research unit Faculty of Fine Arts of the Universidade de Lisboa. This partnership originated the research unit VICARTE [1] which stands for "Vidro e cerâmica para as artes", which translates to "Glass and ceramic for the arts". This research unit *connects the present and the past, by developing new materials for glass and ceramics contemporary art, by studying the traditional and historical production practices and the exploration of different aesthetical concepts in art.* The members of VICARTE are also extremely important for the testing of this platform, as they are the ones to take the role of specialists, which means they are responsible to add stained glass entries to the platform.
The Corpus Vitrearum's presence in Portugal can be mainly observed in the Monastery of Santa Maria da Vitória and, due to the information of the glass art collection found in it being limited to written documents, there is a growing interest in representing these collections of stained glass in an online platform, as it would be helpful in the areas of museology, providing a more accessible and informative way of presenting the information in said documents.
Starting with this collection, it is aimed to then move towards other collections in Portugal and those spread throughout Europe. Ideally, the resulting platform will interconnect repositories by their similarities, as a way of unifying and standardizing sources of information regarding glass art. This is done and can occur due to the existence of common terms within repositories, which will then be useful to extract similar information from two or more different repositories. For example, there is one repository with glass art from Berlin, another one the contains glass art data from Lisbon, it is possible to query glass art made before XIV century in both of these repositories, as the creation date would be a common property in both repository's glass art data tables. With the help of users that are verified as a specialist in the platform, it will be possible to greatly expand the number of collections covered by it.

[1] RESEARCH UNIT VICARTE – GLASS AND CERAMICS FOR THE ARTS - https://vicarte.org/

1.3 Goals and Objectives

In this thesis, the main focus lies in storing information about glass art that can be encountered in Portugal in a comprehensive database, providing a platform that lets certified researchers and experts about the subject add additional information or entries on their own describing pieces in detail, along with editing already existing entries. Providing these details improves the accessibility for users of this platform when they are trying to find the piece they are looking for, along with all its details. In the case of its geolocation, it will be important to provide the user an intuitive and informative interface, which lets them explore and observe where a certain glass collection is located, or simply freely look for existing glass art collections in a certain area in the globe.
This way, avid fans of the subject will have significant control in managing the displayed data regarding each piece of glass art presented on this platform. In case of users that simply want to know more about glass art, they will have tools that will let them search for glass art based on several topics. By applying filters, it will be possible to focus each search based on each user's specific criteria through the implemented Semantic technologies that will be explained in the next chapter.
Along with this, another important goal of this project lies in accessing and representing data regarding glass art from records that lie within other data sources across the web. There have already been created stained glass repositories on the web which use Linked Open Data and can be accessed via SPARQL queries, as will be explained in section 2.2.2.2. One important online library that stores digital cultural heritage content in Europe, which also includes stained glass content, is Europeana. By creating a connection between our platform and other endpoints that store stained glass-related information, it is, in theory, possible for us to access the other endpoint's data and add it to the repository or simply display it in our platform.

To briefly summarize the main objectives and goals of this project, it is important to:

1. Create one or more repositories that will store data regarding glass art in a standardized and machine-readable format, which will later be accessed and presented in an online, publicly accessible platform;

2. Give the ability to add or alter the information presented in the application for users which are verified as experts in glass art.

3. By following step 1, respect and implement the principles behind Semantic Web and Linked Open Data;

4. Create the user-centered side of the application through the usage of preexisting frameworks;

5. Inside the created Wiki mentioned in step 4, develop a map interface which lets users see where each glass art collection is located in, initially, the Portuguese territory;

6. Enable the management of the data inside the repository created in step 1 through the usage of data management applications (where some are extensions of the frameworks referred in step 4);

7. Enable access to Glass art repositories from additional sources (Such as Europeana), making it possible to access data from other domains other then our own.

2

State of the Art

2.1 Fundamental Concepts

To fully understand what has to be achieved through this project, there are some core concepts that must be understood. Before getting into them, it is important to also understand what needs to be done in order to store the data and make it accessible to the users, both in terms of viewing the information and editing it. In this chapter's section, we will focus on explaining these terms, on how to implement them and on how applying them will help in this problem. The main two are related with one another, the Semantic Web and Linked Data. Although the proposed solution will be implemented with Linked Open Data, this concept will be fairly easy to grasp once Linked Data is explained. The concepts and technologies that will be presented have been increasingly adopted by companies, governments and users, as they have caught the eye of many enterprises. Knowing this, it will be clear to see the potential positive effect of implementing these technologies on future web applications, such as improving their internal data curation processes and maintain better links between, for instance, digitized objects and their descriptions. It can improve data publishing processes within organizations even where data is not entirely open.

2.1.1 Semantic Web

A key aspect of the traditional web is that its contents are distributed throughout itself, where each excerpt of data has its own web server and its own party. There are some basic technologies that define Semantic Web, as stated in [1]:

1. Using labeled graphs as the data model for objects and their relations, with objects as nodes in the graph, and the edges in the graph depicting the relations between these objects. RDF is used as the formalism to represent such graphs.

2. Using URIs [1] to identify the individual data-items and their relations that appear in the datasets. Again, this is reflected in the design of RDF.

[1] A Uniform Resource Identifier is a sequence of characters that defines a logical or physical resource

3. Using ontologies [2] as the data model to formally represent the intended semantics of the data. Formalisms such as RDF Schema [3] and The Web Ontology Language (OWL) [4] are used for this purpose, again using URIs to represent the types and their properties.

The main objective behind the idea of implementing a Semantic Web architecture consists in making data on the web accessible and machine-readable, creating a standardized format which let's anyone use and refer to data that belongs to other entities. In order to fully implement a Semantic Web architecture according to [1], the following must occur:

1. An agreement must be made on the standard syntax that will be used to represent both data and metadata.

2. There must be a sufficient agreement on the metadata vocabularies, as a way to share intended semantics of the data.

3. It is crucial to publish large volumes of data in the formats of step 1, using the vocabularies of step 2.

2.1.2 Metadata

The Semantic Web technologies that were previously mentioned are used to formally represent metadata. Metadata, on its own, summarily is data with information about data. Implementing metadata is a way enabling the end-user to find items and contextually relevant information. Additionally, Semantic metadata adds some other factors to this, which are relationships, rules, and constraints to regular metadata. This type of metadata *describes contextually relevant or domain-specific information about content based on a domain specific metadata model or ontology, providing a context for interpretation. In a sense, they capture a meaning associated with the content. If a formal ontology is used for describing and interpreting this type of metadata, then it lends itself to machine process ability and hence higher degrees of automation* [23]. In terms of glass art, for example, it would enable users to search individual or groups of glass art pieces based on their existing properties. For example, it allows in this manner queries regarding a glass art piece's creator, to the country it located in, and so forth. Further in this chapter, a possible structure for Semantic metadata will be shown in greater detail.

[2] Ontologies define the concepts and relationships used to describe and represent an area of concern

[3] *RDF Schema is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources. RDF Schema is written in RDF using the terms described in this document. These resources are used to determine characteristics of other resources, such as the domains and ranges of properties.* [2]

[4] *The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things.* [12]

2.1.3 Linked Data and Linked Open Data

In order to fully conceive a Web of Data like the Semantic Web, it is necessary to have a relevant amount of data on the web, available in a standard format, as previously mentioned in Section 2.1.1, which is reachable and manageable with the usage of Semantic Web tools, such as RDF 2.2.2, which will be explained in much greater detail in the next sections. Not only must this occur, but the relationships that exist amongst the data should also be available. This large collection of interconnected datasets on the web can also be referred to as Linked Data. These seamless created connections between data can serve one or many purposes. Linked Open Data, while inheriting everything previously mentioned, is additionally shared and can be freely used and modified by the users connected to it. In the case of, for example, commercial enterprises deciding to implement Linked Open Data, they can easily connect and integrate the data that they are willing to share with one another, with the purpose of collaboration. In this way, each enterprise can access or integrate the other's data. Provided this example, the benefits are evident in the context of this project. *With Linked Open Data, libraries can increase their presence on the Web, where most information seekers may be found* [13]. In this way, the openness of data can be seen as an oportunity as opposed to a threat.

2.2 Data Management

As previously mentioned, the platform that we want to create is intended to implement Semantic Web and Linked Data principles as to create a machine readable and unified structure for the platform at hand, as well as to provide a common framework that allows data to be shared and reused across the glass art's community's boundaries. Important factors that must be taken into account into this is the type of data that is going to be stored and also how it is going to be represented. In the case of this project, it is known that each piece of glass art stored in the repository must have a description, its own ID, date of creation or a fair estimation of it if there is one, alongst other specific information.

2.2.1 Ontology Languages

Ontology is a term whose meaning has change from its original one when the word first originated, now being an explicit and formal specification of a conceptualization. In general, an ontology consists of a countable list of terms and the relationships that exist between them. These mentioned terms denote important concepts of the domain which they are being worked on. If we take the context of this project for example, this is, in the glass art setting, stained glass, glass jewelry, medieval stained glass are important concepts to be considered. The relationships found within these concepts usually include a hierarchy, which means, considering there is a concept C1 and a concept C2, C1 is a sub-concept of C2 if every object in C1 is also included in C2. Delving further into to the

definition of ontologies, they are very useful for the organization and overall navigation of websites.

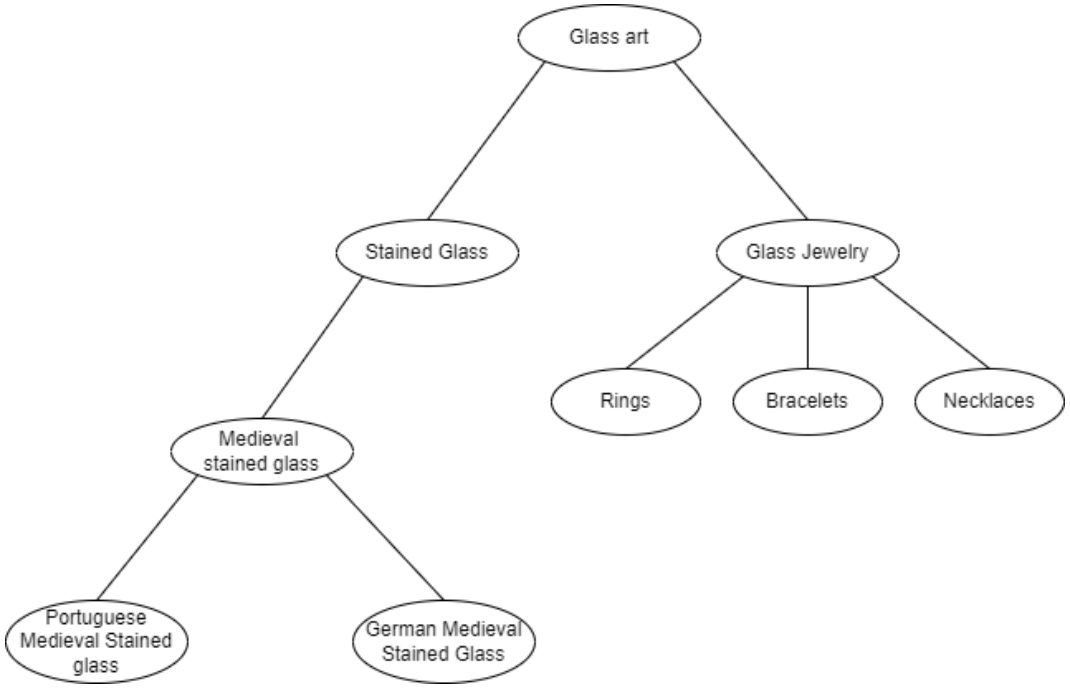


Figure 2.1: A possible hierarchy for the glass art context

Apart from these hierarchical relationships between concepts, ontologies can include information which, as seen in [1], are:

- *properties*
- *value restrictions*
- *disjointness statements*
- *specifications of logical relationships between objects*

There are, however, some problems that must be considered when thinking about the context of the web. Due to the fact of ontologies sharing the understanding of a domain, two applications may use the same term for different meanings. For example, church A may have the same name as the one of a single stained art piece in church B. Although specific, these cases must be taken into consideration and can be solved by *mapping the particular terminology to a shared ontology or by defining direct mappings between the ontologies. In either case, it is easy to see that ontologies support semantic interoperability* [1]. Ontologies, in the context of this thesis, are useful for the following:

- Ontologies improve the organization and navigation within websites as users move from one concept to another in the ontology structure.

- Improving the accuracy of web searches, since the search engine can look for pages respective to a specific concept in an ontology rather than having to collect all pages in which certain, generally ambiguous, keywords occur.
- Web searches can exploit generalization/specialization information, in the sense that, in the case of failure in finding relevant documents when a query is done, there can be suggestions for queries that have more general and broader answers. more general query.

2.2.2 Data models - RDF

Resource Description Framework (RDF) is, in short terms, a framework for representing interconnected data on the web. What makes it special is the way this model can be used to express explicit metadata. Unlike XML, which describes the structure of information at a syntactic level, RDF makes statements about the pieces of information about certain resources. It is, in this way, structured into *triples*, which serve as the way to describe those resources through statements, which specify the properties and the values to those properties regarding the resources. This becomes more clear with an example. Consider the following statement:

"**Stained glass** has a **creation date** whose value is **7th century**".

From this sentence, the RDF terms for the various parts of the statement are:

- The *subject*, which is "**Stained glass**"
- The *predicate* which is the word "**creation date**"
- The *object* which is the phrase "**7th century**"

Knowing how a single RDF triple is structured, we can give a better and more realistic example where URIs are used to identify and represent entities, in table 2.1.

Subject (resource identifier)	Predicate (property name)	Object (property value)
https://www.vitralwiki.com/id/1241256	type	Stained Glass
https://www.vitralwiki.com/edm:object/image.png	publisher	VICARTE

Table 2.1: Structure of RDF triples

A good example of a metadata schema is used in Corpus Vitrearum DE, from which it is possible to take many ideas from, regarding what information is contained inside it. Taking the metadata from a piece to use as an example (Heilige Sippe):

```
                                                                ...
<rdf:Description rdf:about="https://corpusvitrearum.de/id/F3432">
 <Iptc4xmpExt:ArtworkOrObject>
        <rdf:Bag>
            <rdf:li rdf:parseType="Resource">
                <Iptc4xmpExt:AOCircaDateCreated>um 1500</Iptc4xmpExt:
                    AOCircaDateCreated>
            </rdf:li>
        </rdf:Bag>
 </Iptc4xmpExt:ArtworkOrObject>
 <Iptc4xmpExt:DigitalSourceType>digitalisiert von einem Filmnegativ</Iptc4xmpExt:
     DigitalSourceType>
 <Iptc4xmpExt:LocationCreated>
        <rdf:Bag>
            <rdf:li rdf:parseType="Resource">
                <Iptc4xmpExt:City>Adolfseck</Iptc4xmpExt:City>
                <Iptc4xmpExt:CountryName>Deutschland</Iptc4xmpExt:
                    CountryName>
                <Iptc4xmpExt:LocationId>
                    <rdf:Bag>
                        <rdf:li>http://sws.geonames.org/11127701</rdf:li>
                    </rdf:Bag>
                </Iptc4xmpExt:LocationId>
                <Iptc4xmpExt:ProvinceState>Hessen</Iptc4xmpExt:ProvinceState>
                <Iptc4xmpExt:Sublocation>Kapelle</Iptc4xmpExt:Sublocation>
                <Iptc4xmpExt:WorldRegion>Europa</Iptc4xmpExt:WorldRegion>
            </rdf:li>
        </rdf:Bag>
 </Iptc4xmpExt:LocationCreated>
 <exif:GPSLatitude>50.1595</exif:GPSLatitude>
 <exif:GPSLongitude>8.0794805555556</exif:GPSLongitude>
 <cvma:AgeDeterminationEnd>1504-12-31</cvma:AgeDeterminationEnd>
 <cvma:AgeDeterminationStart>1495-01-01</cvma:AgeDeterminationStart>
                                                                ...
```

Listing 2.1: Excerpt of metadata from Corpus Vitrearum DE

As it can be observed from the highlighted terms, it is possible to have an idea on the type of data that can be stored regarding a single piece of glass art, such as the country and city, its GPS coordinates, the rough estimation of the date when the piece was made, to name a few. This metadata was created using RDF/XML syntax, which is standardized by the W3C and is widely used to publish Linked Data on the Web. Despite being machine readable, which is one of the steps towards reaching the Semantic Web, there are some

problems with this syntax. One of which is the fact that it is not as accessible for humans to read or write using this syntax. For this reason, it is not ideal to use this syntax in *data management and curation workflows that involve human intervention, and to the provision of alternative serializations for consumers who may wish to eyeball the data [14].* There are although other serializations which should be considered when data management does involve human intervention, such as RDFa.

2.2.2.1 RDFa

This serialization format embeds RDF triples in HTML documents. Instead of being embedded in comments within the HTML document, which happened in ealier attempts of mixing RDF and HTML, the RDF data is intertwined with HTML *DOM* [5]. RDFa is popular in contexts where data publishers are able to modify HTML templates but have relatively little additional control over the publishing infrastructure. It also must be kept in mind that, *when using RDFa to publish Linked Data on the Web, it is important to maintain the unambiguous distinction between the real-world objects described by the data and the HTML+RDFa document that embodies these descriptions* [18].

One question that can be asked from all of this is how can it help in the Linked Data context. There are a few ways, as stated in [14], where the RDF data model aids in the implementation of Linked Open Data, which are the following:

1. By using HTTP URIs as globally unique identifiers for data items as well as for vocabulary terms, the RDF data model is inherently designed for being used at global scale and enables anybody to refer to anything.

2. Clients can look up any URI in an RDF graph over the Web to retrieve additional information. Thus each RDF triple is part of the global Web of Data and each RDF triple can be used as a starting point to explore this data space.

3. The data model enables you to set RDF links between data from different sources. Information from different sources can easily be combined by merging the two sets of triples into a single graph.

4. RDF allows you to represent information that is expressed using different schemata in a single graph, meaning that you can mix terms for different vocabularies to represent data. This practice can be observed in the figure 2.2.

5. Combined with schema languages such as RDF Schema, the data model allows the use of as much or as little structure as desired, meaning that tightly structured data as well as semi-structured data can be represented.

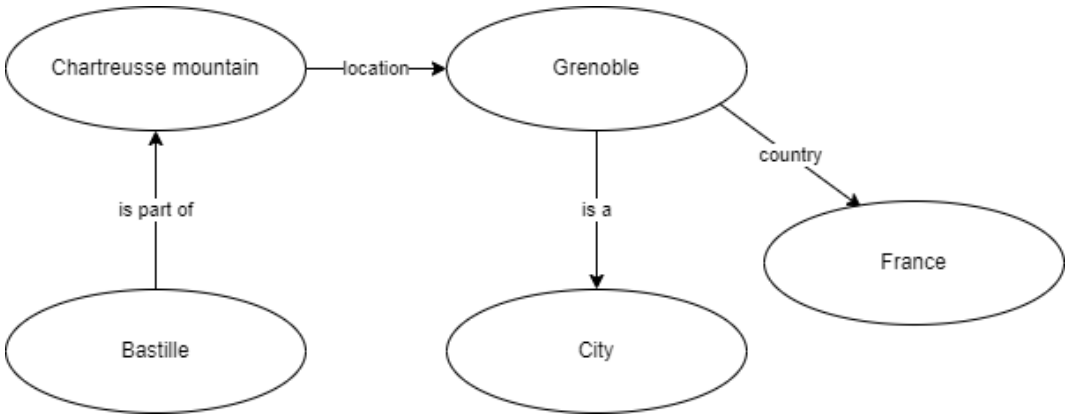


Figure 2.2: RDF Graph

As observed in section 2.2, the labeled nodes are connected by labeled arcs. These labeled nodes represent the entities or objects, whereas the arcs represent the relation of one node to the other. For example, in the case of the node "Grenoble", it is connected and directed towards "City"with the arc labeled as "is a", creating the notion that Grenoble is in fact a city.

2.2.2.2 SPARQL

Related to the RDF data format, which *organizes information in graphs rather than in fixed database tables* [21], SPARQL (SPARQL Protocol And Query Language) serves as semantic query language. That is, it is able to select, extract, and otherwise easily get a hold of particular sections of data which is stored in RDF files. It is specifically designed for this data model and functions similarly to SQL, in the sense both are Declarative Query Languages. This querying language is important in the sense that it allows us to view and analyse data from different domains other than the one our platform will be deployed in. Through SPARQL, if there is a connection between the platform that is being created in this thesis to another endpoint that contains information about stained glass, we can visualise said data given its parameters. Since essentially, SPARQL functions similarly to SQL but directed towards RDF triples, there are some structural differences that must be understood. The translation of a regular SPARQL query into a regular language would be 'I want these pieces of information from the subset of the data that meets these conditions.' These conditions are described triple patterns, which are similar to RDF triples but may include variables to add flexibility in how they match against the data [9]. For example, using the SPARQL query in 2.2 to query an endpoint 'https://www.vitralwiki.com' that contains the data referred in 2.1, the result of the query would be the object, which in this case is the value of the property 'type', where its subject has property 'ID' with value '1241256', which would in fact return the value 'Stained Glass'.

[5]Document Object Model

```
SELECT ?type
WHERE {
   ?id <https://www.vitralwiki.com/id/Property-ID> <https://www.
      vitralwiki.com/id/1241256>
   ?id <https://www.vitralwiki.com/id/Property-type> ?type
}
```

Listing 2.2: SPARQL query example

2.2.3 Georeferencing glass art

Associating a geographic reference to stained glass is an important aspect that this solution wants to achieve, since it is important to give users the ability to locate a determined piece of glass art in a map, giving exact coordinates in relation to the globe.

GIS (Geographic Information System) [4] is a spatial system that creates, manages, analyzes and maps all types of data. It has already been studied and proven that web-geographic information systems have great capabilities in disseminating and in representing cultural heritage data online [17]. This can be observed in further detail in the article [17] where the importance and utility of an architecture that uses Web-GIS [8] based integration of 3D models is explained with great extent. The same also applies for any type of data as previously said, so it applies to glass art as well. Just like cultural heritage data and its associated metadata produced by many cultural heritage institutions is heterogeneous, this can also be said for glass art produced for churches and monasteries in many different countries. Georeferencing stained glass entries is extremely important in the context of Linked Open Data, as having a reference to the location of a piece will let everyone that has access to this piece of information represent the location of a piece in a different way to other platforms that are displaying this said property.

2.3 Relevant tools

Wikis are hypertext pages/publications where their users can publicly add and edit information related to what the publication is about, which could be from objects to places and any subjects at all. What separates regular Wikis from Semantic Wikis is that, while regular Wikis have structured text and untyped hyperlinks, Semantic Wikis provide the ability to capture or identify information about the data within pages, and the relationships between pages, in ways that can be queried or exported like a database through semantic queries [24]. The most well known and used Wiki framework is MediaWiki, which is used to structure the famous encyclopedia Wikipedia. There are other tools, out of which OntoWiki is one that stands out, since it has been used for a long time and has a considerable amount of information online regarding it. These applications serve the main purpose of providing a front-end for the platform which is the focus of this

thesis, giving it a preset of options when it comes to its presentability and aspect. To explain it in a succinct way, these frameworks serve the purpose of allowing the person that is developing, a way to collect and organize knowledge in it and make it available to people. These applications, however, to be able to access to data, must be extended with something that gives the ability to store and query data within the Wiki's pages. In the case of MediaWiki, there are three tools that can make this connection to data, such as Semantic MediaWiki, WikiBase and Cargo. All of them are knowledge management extensions to MediaWiki that let you store data in a structured and queryable way. OntoWiki has an in-built data management tool, named after the Wiki itself. Taking into consideration that Semantic MediaWiki is the most popular extension for MediaWiki, there are some core differences that can be observed from OntoWiki's data management tool, both conceptually and in their respective customizability aspect, as it can be seen in the figures 2.3 and 2.4 respectively.

	Semantic MediaWiki	**OntoWiki**
Managed entities	Articles	Resources
Editing	Wiki markup	Forms
Atomic element	Text blob	Statement

Figure 2.3: Conceptual comparison between Semantic MediaWiki and OntoWiki [11]

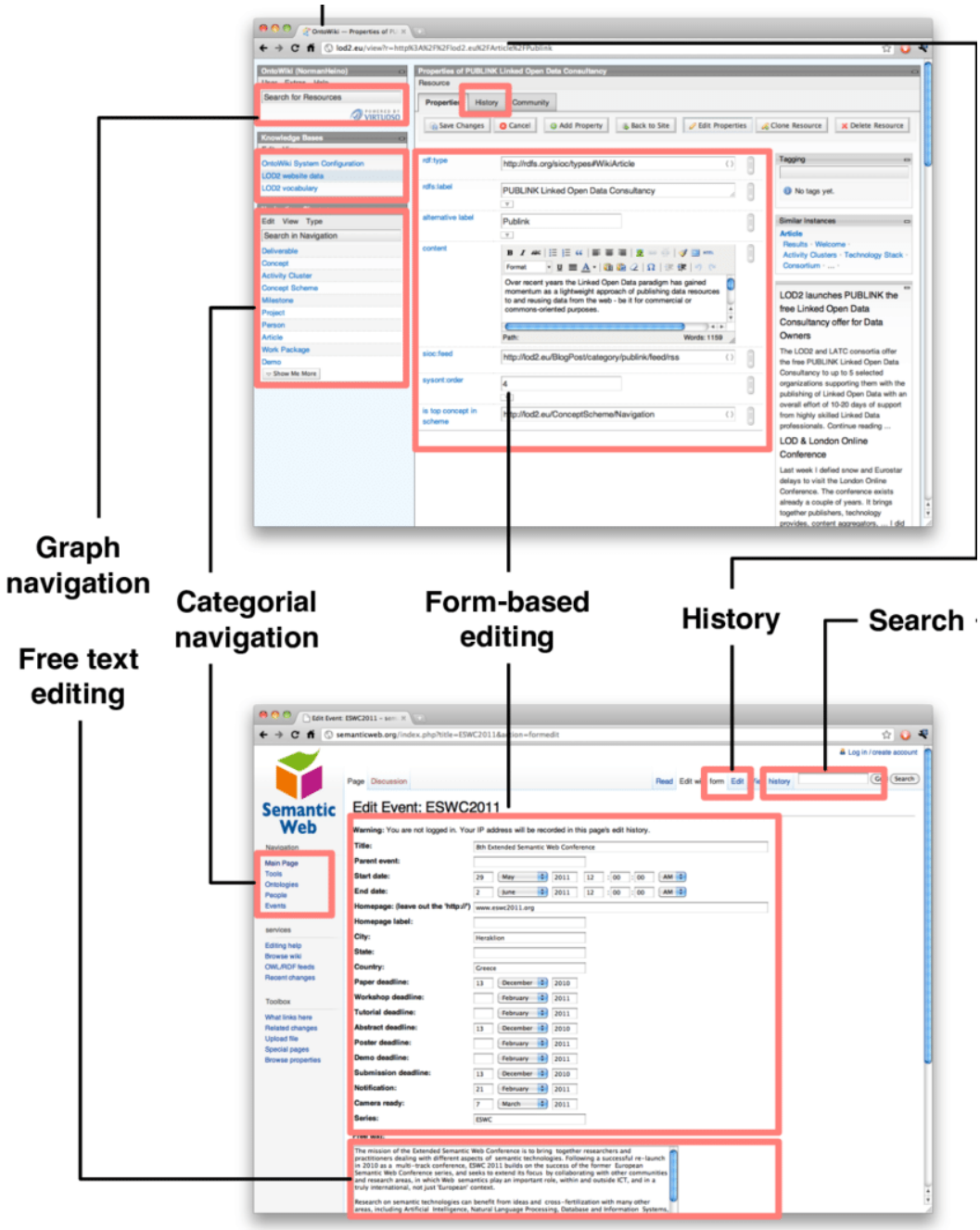


Figure 2.4: Comparison of Semantic MediaWiki (Lower window) and OntoWiki (Upper window) GUI building blocks [11]

2.4 Related Work

As it can be seen from the previously mentioned metadata example in 2.2.2, working with Semantic Web and Linked Data is not entirely anew. Three web platforms that stand out are a Swiss website named Vitrosearch, the British website Corpus Vitrearum Medii Aevi and the German website Corpus Vitrearum DE. What all of these obviously have in common is that they all are designed to present information regarding glass art. Despite this similarity, there are all some differences that distinguish each of these online platforms from one another, each being unique in its own way. In this section, there will be a focus on analysing and studying these three platforms and referring how it is possible to take ideas on how each can support the creation of an online platform that has a repository of stained glass in Portugal.

2.4.1 Corpus Vitrearum DE[6]

Looking firstly into Corpus Vitrearum DE, since it is where the example of the website where the metadata shown in 2.1 was taken from. At a surface level (as seen in figure 2.5), we can see that this website is structured in different sections, where one has the image archives corresponding to each stained glass piece encountered in the platform's repository, and the other sections which give a wider context to the purpose of the website and more information about the research that is being done around it.

[6] https://corpusvitrearum.de/

Figure 2.5: Main page of Corpus Vitrearum DE
Meaning of every word underlined in red: 1. Image Archive; 2. Stained glass in context; 3. The project; 4. Publications; 5. Research; 6. Current

One thing that can be observed also from the metadata in listing 2.1 is that this platform implements the principles introduced in section 2.1. This website provides a large number of glass art pieces encountered throughout the German territory, more specifically stained glass encountered in its churches. In this platform, each glass art piece contains an entry in the website which provides numerous details related to it, as well as an image of it. Along with this, it also contains a map interface, where it is possible to see the exact location where this glass art piece is located in the globe's map.

2.4.2 Corpus Vitrearum Medii Aevi[7]

The Corpus Vitrearum Medii Aevi (CVMA) is the international research project dedicated to recording medieval stained glass in Great Britain. Despite this, it is quite different in the

[7] https://www.cvma.ac.uk/jsp/index.jsp

way glass art is presented. Instead of having the full list of stained glass images straight away, it is necessary to select some filters before being able to access this data, where it is possible to view different glass art collections by the county where they are in Great Britain, and by their location. There is also another way of searching for glass art, by using more complex queries, which lets the user be more specific, in case they are looking for a collection or singular art piece in a more focused search. With this method, along with letting the user selecting the county and/or location, it lets them also filter out the chapel's name where the stained glass can be found, along with the window's name and some other filters. The last searching method provided by the website is by the image's inventory number, in case the user already knows exactly the piece they are looking for. This search engine is quite interesting in terms of the tools it provides in order to specify a user's criteria, giving off fields to compact the amount of resulting images, as can be seen in figure 2.6. It is possible to make a similar approach in the case of this thesis, not only because of the potential large collections of data that could be worked with, but also to make the user's life simpler.

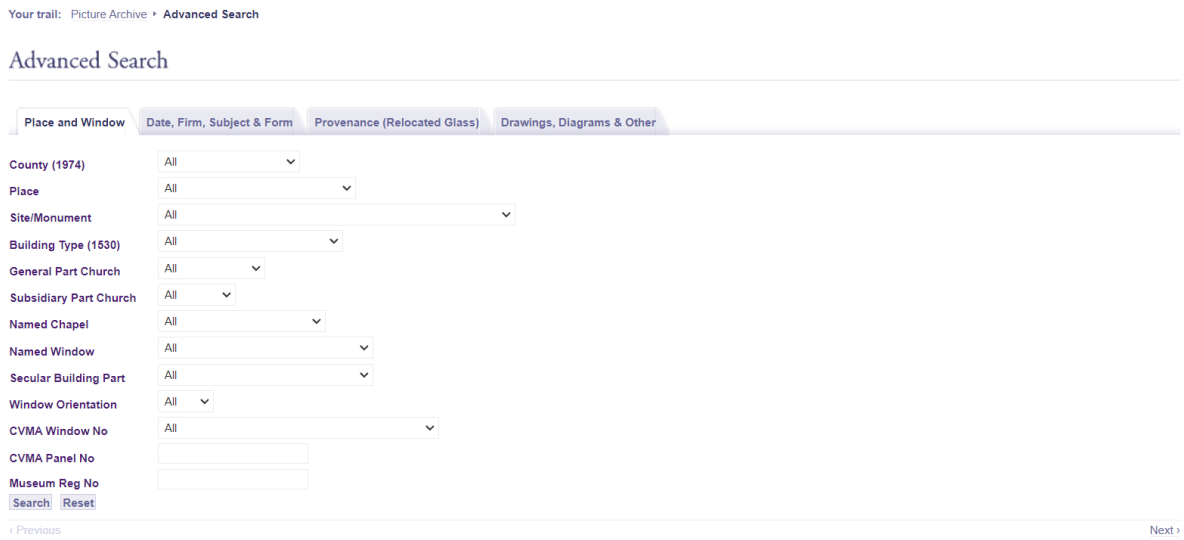


Figure 2.6: Corpus Vitrearum Medii Aevi advanced search section

2.4.3 Vitrosearch[8]

Finally, when looking into this Swiss website, what pops out from the start is how differently it is structured comparatively to the two previously observed platforms in section 2.4.1 and section 2.4.2, as can be seen in figure 2.7. Instead of being separated into sections in the upper part of the website, it lays out its information in the main page's

[8] https://vitrosearch.ch/en

body. There are pros and cons behind this design choice. On one hand, it makes everything reachable at all times from the starting point, where it is possible to search straight away for glass art and to see relevant collections according to the website. On the other hand, this can make using the website a bit more confusing since it is not separated into sections. The main choice in the design that stands out from the rest is that, in the main page, there is already a set of stained glass collections related to certain time gaps or certain artists. This can make access to stained glass collections more direct, without creating the need to make a specified search before.

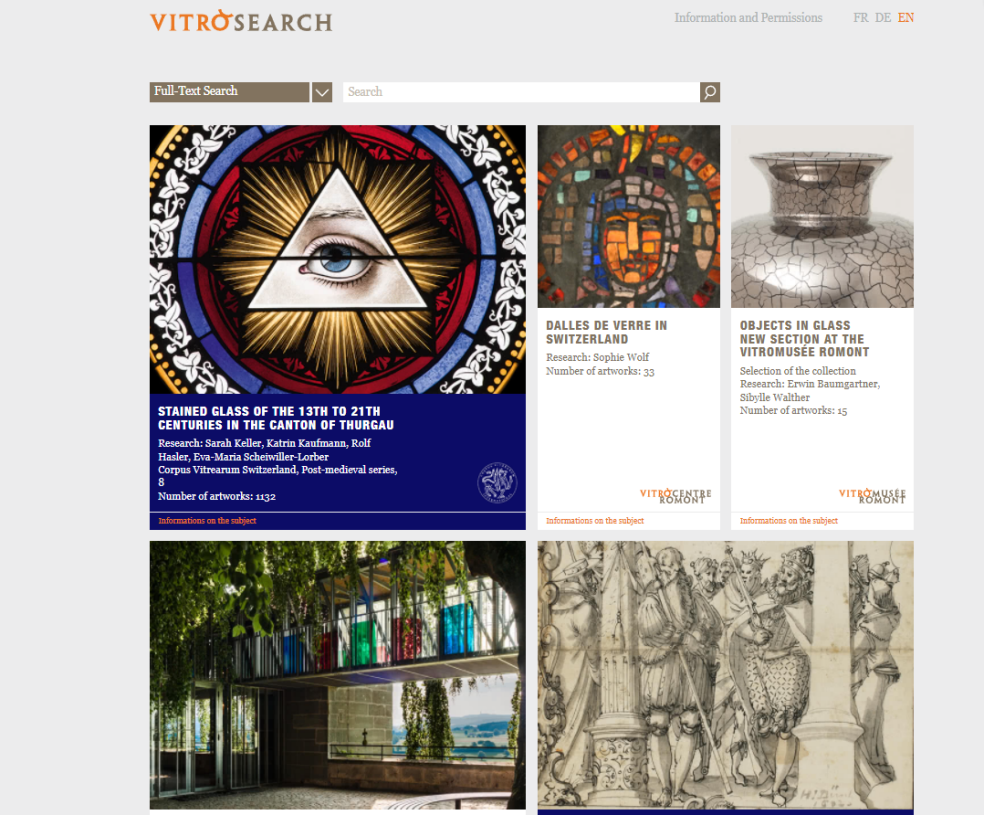


Figure 2.7: Vitrosearch main page

Differently to what is implemented in each of these websites, when constructing a Wiki, depending on the chosen Semantic Wiki data management tool, there are also some APIs that can provide a map interface, which lets one manipulate and look through the globe freely. The main factor that makes the platform planification proposed in this document stand out is that it will let users, which are verified as experts, edit information and add new entries in the glass art repository, by using the created Semantic Wiki. This way, experts that use the platform can add unknown facts or pieces of information related to a certain piece, or correct possible mistakes that could have been made by other users of the platform.

2.5 Choosing a framework

From the tools shown throughout this chapter, it is important to compare them and to see which one provides the most benefit and is most fitting for the context of this project. As said previously, there are some key aspects the are needed in order to create a platform like the one described in section 1.3.

2.5.1 OntoWiki

For it to be possible to come to a fulfilling solution to this problem, it was necessary to test the available tools in order to see which is more fitting in this case. Before referring these tools and how they were tested, it is important to note that this was done on a Virtual Machine of Ubuntu version 20.04. Taking this into account, the first Semantic wiki tool that was tested was OntoWiki. This was one of the two main tools that were considered due to its popularity and available online information regarding its usage and usefulness. The chosen DB system to use in order to store data was Virtuoso. Although there were other choices, Virtuoso stood out due to it having several APIs that would aid in the particular case of this project, such as GeoSPARQL. Unfortunately, this is an outdated wiki, as it has not been updated since 2017. During the installation of this software, several problems occurred, since it required deprecated versions of the subsequent required software to be installed. This is, PHP was needed in order to progress in the installation of OntoWiki but it couldn't be the latest version of it, needing a much older version. For this reason, it was decided not to proceed further in the usage of this Semantic Wiki tool.

2.5.2 MediaWiki

The second semantic tool that was considered and ended up being used in this thesis is MediaWiki. Being much more accessible than the previously tested OntoWiki in its installation since it is a more frequently updated software.

2.5.2.1 Setting up MediaWiki

In order to set up MediaWiki, there was a sequence of steps necessary to implement to have it working. The steps are as follows:

1. Creating a web server to host MediaWiki (using Apache Web server for example);
2. Uploading MediaWiki's folder to said web server after being deployed;
3. Choosing a database system that supports MediaWiki, namely between SQLite, MariaDB/MySQL, and PostgreSQL;
4. Creating a database for this platform and a repository, as a means to store future data (in our case, regarding glass art);

5. Running the configuration script in an online browser.

In step 3, it was decided then to use MySQL, since it is the most popular Open Source SQL database management system for its high efficiency with large amounts of data and for it being one of the most secure database systems in the world. With the conclusion of this setup, it was then possible to finally manage and personalize the resulting front-end side of this Semantic Wiki, with the default main page being the one in figure 2.8.

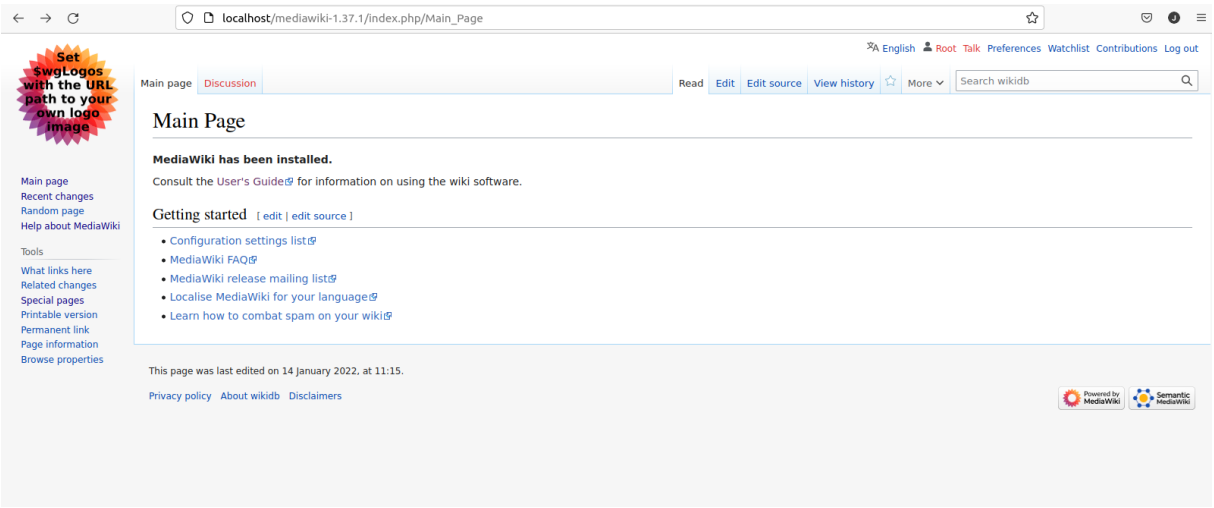


Figure 2.8: Resulting main page after setting up MediaWiki's installation

2.6 Configuring the Wiki

For it to be possible to freely and fully develop the wiki, there are some important tools and programming languages that must be known in order to not only create a functioning platform, but one that also reaches the requirements previously mentioned.

2.6.1 Presentation and UI tools

Given that, in 2.5, it was concluded that the most fitting framework for this project was MediaWiki, it is necessary to understand what is needed to customize and modify the wiki to our desires. There are some programming languages needed to operate with MediaWiki, each serving its specific purpose, whether it is to change how elements are shown in each page, or querying data from a certain endpoint.

2.6.1.1 PHP

Despite being PHP being normally used on server-side scripting, this language is useful for much more and contains the base for which frameworks like MediaWiki are structured.

It is a fairly general language where scripts are loaded by an interpreter, compiled to bytecode, and then executed. Its main role in these frameworks is configuring the wikis themselves, such as user rights and permissions, rules, additional extensions and database settings, to name a few.

2.6.1.2 Wikitext

Wikitext, also know as wiki markup or wikicode, consists of the syntax and keywords used to format a page in wiki frameworks like MediaWiki. This means it is used to configure a good chunk of the front-end of the framework, as it is used to edit what is shown to the users and to organise the data's layout in a page to the developer's desire. This type of text also allows the aggregation of HTML elements, which can be quite useful in formatting the wiki.

2.7 Summary

Before proceeding to the description of the implemented solution, a brief and concise summary of the relevant tools for this work is in order. Looking at the available Semantic Wiki frameworks, it is important to understand why we chose MediaWiki to proceed with this thesis. A large drawback behind using OntoWiki is related to how old this software is. Since it is a fairly old Semantic Data Wiki and Linked Data publishing engine, there is the possibility of version errors emerging due to incompatibility with other programs, since some complementary software is still being updated to this date. MediaWiki, on the other hand, is a widely utilized and regularly updated engine, having a long term maintenance and support. For this reason, it has a wide collection of APIs and extensions that help in the making of a richer and well versed interface. It is well suited for highly active Wikis and supports rich content through specialized syntax.
Taking into consideration the websites stated in 2.4, we can also take many good implementation ideas from these platforms. The two websites which have the most useful ideas in the context of this thesis' theme are the German website Corpus Vitrearum DE, mentioned in section 2.4.1, and the British website Corpus Vitrearum Medii Aevi, mentioned in section 2.4.2. Looking into the Corpus Vitrearum DE, it is divided into different sections, in which one contains the list of glass art pieces with their respective images. In the case of Corpus Vitrearum Medii Aevi, the main design choice that is of great relevance for our platform is its complex and detailed search engine. From it, it is possible to make more complex and specific queries regarding the collections or singular pieces of glass art that we want to analyze with greater detail. In the next chapter, we will discuss how we used MediaWiki to create a solution for this thesis.

3

Implemented Solution

This chapter's focus will lie in describing the implemented solution for this problem, explaining each choice and why it was made, along with the project requirements, such as Wiki and data management frameworks, and the project's architecture.

3.1 System Architecture

Having chosen MediaWiki as a framework to develop our Semantic Wiki for stained glass related content, we must understand how its architecture is constructed. By analysing figure 3.1, we can conclude that MediaWiki is divided into four layers:

1. **User Layer**
   This layer is responsible for managing user interactions with the wiki. It includes the user interface, which allows users to create and edit content, manage their account settings, and interact with other users and it also includes the authentication and authorization systems, which control access to different parts of the wiki based on user roles and permissions.

2. **Network Layer**
   This layer handles the communication between the web server and the user's web browser. It includes components such as HTTP request handling, caching, and response generation.

3. **Logic Layer**
   This layer contains the core application logic that governs the behavior of the wiki. It includes components such as page rendering, search functionality, and user management. The logic layer is also responsible for enforcing the rules and policies of the wiki, such as content moderation and user privacy.

4. **Data Layer**
   This layer is responsible for storing and retrieving data from the database. It includes the database schema, which defines how data is organized and stored and it

also includes components such as the query builder and the data access layer, which provide an abstraction layer between the application logic and the database.

Figure 3.1: MediaWiki's architecture [16].

Having the general architecture of MediaWiki when it is initially setup, throughout this thesis we changed the Data layer the most. This layer is affected every time a page is created and every time an extension is added to provide additional functionalities to MediaWiki, as it will be discussed in 3.1.2. Along with this, we also changed the Logic layer, in which we had to change configurations of MediaWiki, like the user groups and permissions, as we will discuss in section 3.5. Throughout this chapter, it will be clearer to understand how each of this layers is affected by the additions we have made to our platform.

3.1.1 Data structure

As previously mentioned, we opted to use MySQL as the database system that stores information regarding our platform. Since we are using MySQL, the data in VitralWiki is structured into tables, with each table containing information about a specific type of content, such as pages, revisions, categories, and users. For example, the 'page' table contains information about each wiki page, including its title, content, and metadata such as the date it was created and last edited. The structured nature of the data in MediaWiki allows for efficient querying and manipulation of the content, enabling users to easily search, sort, and analyze large amounts of information.

It's worth noting that this process is indeed automatic, and MediaWiki handles most of the database operations and optimizations behind the scenes. For example, every time a new category of pages is created, this is added to the 'category' table, which was created when setting up MediaWiki on any system, during the process described in section 2.5.2.1. This allows users to focus on creating and managing content without worrying about the technical details of how the data is stored in the database. To sum up, when a user creates or edits a page, the data is stored in MySQL in the following way:

1. The page content is divided into sections, each with a unique section number.

2. The section content is stored in the 'text' table, which has columns for the page ID, the section number, and the section content.

3. The page metadata, such as the page title, author, creation date, and modification date, are stored in the "page"table.

4. The revision history of the page is stored in the "revision"table, which keeps track of the page ID, the revision number, the user who made the revision, and the date and time of the revision.

There are still more tables automatically generated, such as the 'user' table, which holds about the wiki users, such as their usernames, passwords, and email addresses. Along with this, as we add more extensions which use the database, like Semantic MediaWiki, as it will be discussed in section 3.1.2.1, more tables and data will be stored in the database automatically. By the end of this thesis, we ended up with 158 tables in our MySQL database.
Having now the skeleton for the Semantic Wiki, as described in section 2.5.2.1, we are able to fully customize and add new pages, being able to change how everything is structured from what is made available by the MediaWiki. MediaWiki also has a lot of working APIs and extensions that can be added to a configuration file that is created during the installation of MediaWiki through the browser, which can serve many different purposes and add more functionalities that enrich the available set of tools and the customizability in its total. For example, one extension that can be useful for this context is *Maps* which, as the name tells, is an extension to help visualize and work with geographical information. Having access to a plethora of tools that MediaWiki provides us, it is important to know which ones will be useful in a platform this one.

3.1.2 Used MediaWiki extensions

A large part of why MediaWiki was used as the framework to build this wiki of glass art relies on the wide selection of extensions provided to customize and expand our online platform. On its own, MediaWiki does not have enough tools to implement what is required for this project. This sub-section will focus on explaining which extensions were used to expand our platform, why they were used, and on how they were used.

3.1.2.1 Semantic MediaWiki

The first and most important extension used in this platform is *Semantic MediaWiki*. As stated in the official Semantic MediaWiki website [22]: 'Semantic MediaWiki is also a full-fledged framework, in conjunction with many spinoff extensions, that can turn a wiki into a powerful and flexible knowledge management system. All data created within Semantic MediaWiki can easily be exported or published via the Semantic Web, allowing other systems to use this data seamlessly.' Briefly, this extension allows the implementation of

Linked Open Data which allows the inclusion of Semantic Web principles to our page. This extension gives us a very simple way of adding Linked Open Data to our repository, as its options can be seen in figure 3.2. The way it does this is by giving the option to

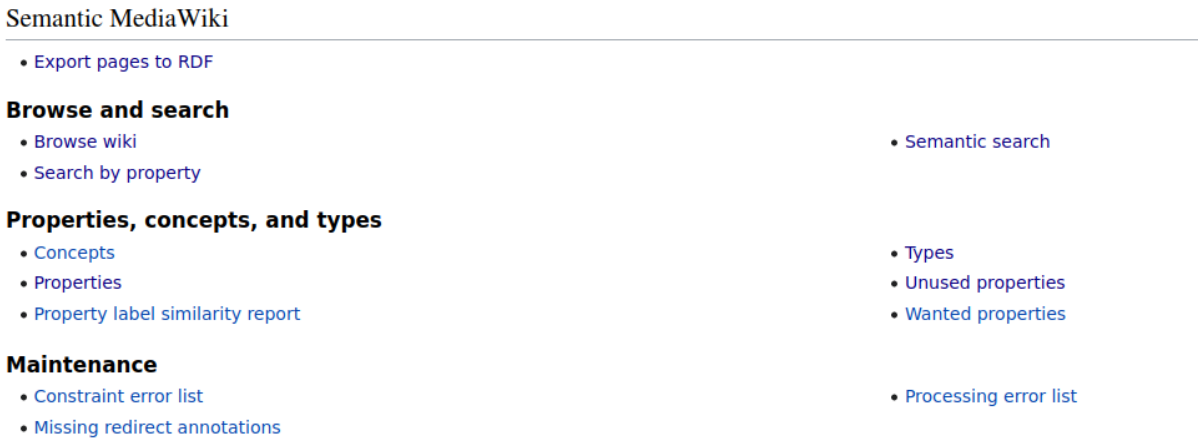


Figure 3.2: Selection of tools provided by Semantic MediaWiki

create properties and to define its data types. These properties like the predicates in RDF and their values are the objects. Having these relations within values, it is possible for users to make Semantic Searches, where they can search for specific stained glass pieces by their properties' values. For example, we can create a property called 'Creation date' of type 'Date' and we can make it a rule for every glass art page to have this property with an assigned value referent to the date when said piece was created. Afterwards, it will be possible for users to search glass art pieces created in a specific date or in an interval of dates. Not only this, but other domains the are connected to our endpoint can also access the data in VitralWiki if they know the names of the created properties.

3.1.2.2 Page Forms

Another extension that can be extremely useful in platforms that use Linked Open Data is Page Forms. Originally created as an offshoot of the Semantic MediaWiki extension, to be able to edit templates that store their parameters via Semantic MediaWiki, this extension allows users to create and edit pages using, as its name suggests, forms. This is, it can allow for the creation of pages without the need of using any programming or the need of having an advanced knowledge on how to work with these kind of platforms.

Since the platform that is being developed in this thesis is designed only for stained glass specialists to change and add new pages regarding stained glass and its different properties, it is difficult to know if the editors of the platform have a reliable way of changing the platform's contents. For these reasons, this extension can close a bridge and make creating and editing pages a far easier task for users who are not familiar with developing MediaWiki platforms than it originally is.

3.1.2.3 Maps

One important feature that was important to be included in our platform consists in representing a stained glass piece geographically. In order to achieve this, the Maps provided a good solution. Along with allowing to represent geographical positions on the globe with a marker, said marker can also be customized, which lets us show an image when the user clicks on the marker, as seen in figure 3.3.

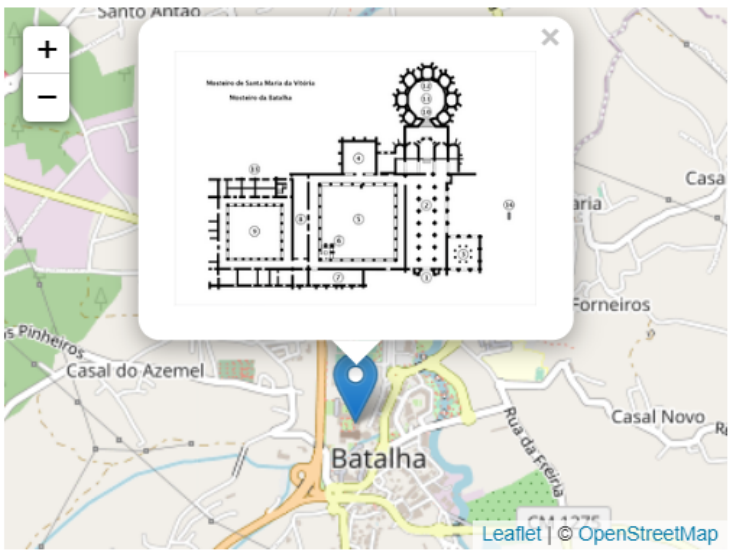


Figure 3.3: Example of the use of the Maps extension in a MediaWiki page

This way, we give the possibility to users registered as specialists to add the blueprint of the building where the stained glass piece is located inside the marker in the map itself,

giving the other users more information about the piece's location.

3.1.2.4 LinkedWiki

This extension serves a very important purpose for the implementation of what was previously mentioned in this thesis. To sum up, LinkedWiki allows the reutilization of Linked Data in the wiki. In other words, with the use of this extension, we are able to access data from different endpoints. We can access this data creating a configuration to a specific endpoint like Wikidata and, after connecting to said endpoint, creating a SPARQL query to represent specific data from that endpoint. In the context of this project, this is extremely useful and helpful as it creates a bridge between our endpoint and several others. When connected to an endpoint and when the query is already written, it is possible to represent the result from the query either in a data table, which contains the names of the properties that are being queried in the columns, and the values for each column in the rows.

The other way of representing the data is in a map. In this way, if the endpoint has entries with geographical coordinates as properties or sub-properties of another property, it is possible to represent each entry and its information in a marker on the map.

There are some limitations that come with querying an endpoint like Europeana's, whose repository hold information, not only about stained glass but also about art, books, films and so on. One of these is related to the fact that not all entries in its repository have values for every property in their data model. This means that if someone wishes to search entries where the property 'dc:subject' holds the value 'Stained glass', there might be quite some stained glass records that are not being shown because they do not hold exactly the same value or they do not have any subject assigned to them at all.

3.1.2.5 Parser Functions

The extension Parser Functions is one that already comes with the regular installation of MediaWiki. This extension allows the usage of several functions while creating and editing pages, such as the usage of *if* operations and other string related operations. Parser Functions can prove to be rather useful in the context of this platform, as many of the properties of our data structure will be of *string* type and manipulating conditions for when data is supposed to be shown and how it is supposed to be presented to the user can also help create a more structured platform with organised pages.

3.1.2.6 Scribunto

Somewhat related to the previously mentioned extension in section 3.1.2.5 in its functionality on a surface level is Scribunto. Also an extension that comes with the default installation of the most recent MediaWiki versions, Scribunto allows for the embedding of scripting languages onto MediaWiki. Although Parser Functions allows us to use programming functions while editing pages on a higher level, it is somewhat limited on the

type of functions we can use with it. Scribunto allows us to program pages at a lower level with its only supported scripting language, Lua. With this extension, it is possible to create *Module* pages, where we can create functions that affect information from another page when invoked in said page. Being able to create much more complex functions where we can manipulate the data of a certain page or more will prove to be extremely useful when dealing with more complex problems that can't be resolved only with *if* and the other simple operations offered by the Parser Functions extension.

There are also some extensions that were tested on a surface level but were not used in the final version of this platform. One of which was RDFIO which is an extension that allows us to import new ontologies into the platform's data structure by uploading RDF files into the platform. Although this feature would be very interesting in the context of this thesis, this extension was not compatible with the newer versions of MediaWiki, not fully working when tested on the platform and proving to be a risk to the platform's integrity.

There is another extension worth mentioning called External Data, which allows, as the name suggests, retrieving data from external sources. This includes Europeana's API, which would allow for simpler and more efficient queries that do not need the usage of SPARQL. The reason to why a solution with this extension was not used in the final version of this platform will be explained in further detail in the section 4.

3.2 Structure of the Metadata

To fully achieve what is intended in this thesis, we need to decide how the metadata in this platform will be structured. As previously mentioned, this project is being developed with the aid of VICARTE, and with their help, we can create a structure for the metadata that will be more fitting for users who either are specialists in glass art or want to know more about it and its details. Along with this, one important aspect of Semantic Web is the interrelation between data and properties within different data sources. Since we want our data to be able to be accessed by other endpoints as much as we want to be able to access data from other endpoints, having common properties to these endpoints is essential for this to work. Being that one of the most important data sources on glass art is Europeana, we decided to adapt our data model to theirs in some ways. Properties that exist in common with the Europeana Data Model, such as Creator (who created the stained glass piece) and Type (type of art), it was decided to create these properties with the same name as the ones in the Europeana Data Model.

3.2.1 Created properties and templates

Having now an idea on how the metadata will be structured, it is possible to create the properties that each stained glass piece will contain. These are composed mostly of the properties that were suggested by VICARTE, along with some important properties required by the Europeana Data Model guidelines. The list of these properties can be found in the Europena Data Model github page [1]. Taking this into account, we created the properties seen in tables 3.1, 3.2, 3.3, 3.4, 3.5, and 3.6.

[1] EDMObjectTemplatesProviders- https://github.com/europeana/corelib/wiki/EDMObjectTemplatesProviders

Name of the property	Description	Data type	Examples
Title (dc:title)	Title of the stained glass piece	Text	The meeting of St. Brendan with the Unhappy Judas
Description (dc:description)	Description of the stained glass piece	Text	This stained glass panel was created in ... by ...
Type (dc:type)	Type of glass art the piece belongs to	Text	Stained Glass
Dataset name (edm:datasetName)	Collection to which the glass art piece belongs	Text	08901_Ag_DE_DISMARC
Identifier in collection (dc:identifier)	Univocal identifier of the piece within its collection	Text	B123
Registry Author	Author of the registry of the art piece in the platform	Text	John Doe
Registry Date	Date of the registry of the art piece in the platform	Date	1943-05-19
Creator (dc:creator)	The creator of the piece	Text	Eugène Dunand
Cartoon designer	The cartoon designer of the panel	Text	Mário Costa
Workshop	The manufacturer of the piece	Text	Ricardo Leone
Place of manufacture	Location where the piece was created	Text	Batalha
ID	Automatically generated internal ID that identifies a specific stained glass piece in the platform	Text	0341203

Table 3.1: General aspects template.

Name of the property	Description	Data type	Examples
Country (edm:country)	Country of origin of the piece	Text	Portugal
State/Region	State/region of origin of the piece	Text	Beira Alta
City	City of origin of the piece	Text	Batalha
Building	Building where the piece is located	Text	Mosteiro de Santa Maria da Vitória
Location details	Details regarding the piece's location	Text	In Storage
Current location (edm: currentLocation)	Current geographic location of the art piece	Geographic coordinates	73.18908,-87.34030
Geographic identifier (edm:Place)	URL the identifies the geographic location where the piece is located	URL	`http://www.geonames.org/8010544`
Direction	Direction of the panel, according with the CVMA notation	Text	N; S; 'North Gallery'
Window number	Number of the window of the glass art piece	Text	I;II;IV
Row	Row number where the piece is located	Number	1; 2
Column	Column number where the piece is located	Number	1; 2
Former/Original Locations	Former/original locations where the piece was located	Text	Frankfurt (Oder), formerly main church St. Marien, east choir, north II, 1c; Anger Museum Erfurt; Thuringia Museum Eisenach
Identifiers of original/former locations	List of URLs of the original/former locations where the piece was located	URL	`http://sws.geonames.org/11153426`

Table 3.2: Location template.

Name of the property	Description	Data type	Examples
Images filenames	Filename(s) of the image(s) presented on the page	Text	Vitral.jpg, Vitral2.png
Images descriptions	Description(s) of the image(s) presented on the page	Text	Description of Vitral;Description of another stained glass panel
Image filename of plan	Filepath URL of the plan where the stained glass piece is located	URL	`https://placeholder./Planta.png`
Image filename of elevation	Filepath URL of the elevation where the stained glass piece is located	Text	`https://placeholder./Elevation.png`
Image URL (edm:object)	URL correspondent to the image of the art piece (If nothing is inputted, it will assume the filepath URL to the *Images filenames property*)	URL	`https://api.europeana.eu/thumbnail/v3/400/71155201ce47eee31245ad7232fbdb2d`
Photographic process	Photographic process behind the piece's photo	Text	Transmitted light front single shot
Digital image source	Origin of the digital image of the piece	Text	Original digital photo
Photographic context	Context of the circumstances where the picture was taken	Text	in situ;expanded
Photograph creation date	Date of when the photo was taken	Date	1984; 1984-05; 1984-05-22
Image author	Author of the piece's picture	Text	Renate Roloff
Holder of rights (dc:rights)	Holder of the piece's photo rights	Text	CVMA Portugal
Publisher (dc:publisher)	Publisher of the piece's photo	Text	CVMA Freiburg - `http://www.corpusvitrearum.de/`, CVMA Potsdam - `http://www.corpusvitrearum.de/`
Credits	Credits of the piece's photo	Text	CVMA Germany Potsdam / Berlin-Brandenburg Academy of Sciences, Photo: Holger Kupfer
License type	Specification of the license on which the image can be used	Text	CC BY-NC 4.0
License description	URL the points towards a description of said license	URL	`https://creativecommons.org/licenses/by-nc/4.0/`

Table 3.3: Photography/image template.

Name of the property	Description	Data type	Examples
Measurements (dcterms:extent)	The measurements of the glass art piece	Text	23cm; 50cm
Date of origin	Date or period when the glass art piece was created	Text	'During 1430'; XVth century
Date of origin(standard - beginning)	Beginning of the period of origin in the standard format	Date	1430-01-1
Date of origin(standard - end)	End of the period of origin in the standard format	Date	1430-12-31
Production technique	Technique behind the production of the glass art piece	Text	Grisaille and silver stain painting on colorless and bulk-coloured glass
Laboratorial PDF	Filepath URL of the laboratory analysis of the glass art piece	Text	`https://placeholder./analysis.pdf`
Laboratorial PDF URL	URL of the laboratory analysis of the glass art piece (If there is nothing is inputted, it will assume the same value as *Laboratorial PDF*)	URL	`https://placeholder.com/analysis.pdf`
Current state of conservation	Text description of the current state of the piece	Text	The window was taken down again in 2006 for cleaning and reinforcement of lead joints.
Restoration and conservation history	Filepath URL of the text file with the history of restoration and conservation interventions to the piece	URL	`https://placeholder./history.pdf`
History related image	Filepath URL of the history related image regarding the piece	URL	`https://placeholder./history.png`
Bibliography/source (dc:source)	Bibliography/source historically associated with the art piece	Text	Lionel Breitmeyer, Saint-Maurice, église catholique romaine de Bernex: étude historique, Genève, 1991

Table 3.4: Panel template.

Name of the property	Description	Data type	Examples
Owner's name	Name of the owner of the piece	Text	Corpus Vitrearum
Owner's identifier	Function of the owner in relation to the object	Text	Conservator

Table 3.5: Owner template.

Name of the property	Description	Data type	Examples
Iconclass notation	Alphanumeric code of the iconclass classification that describes the iconography of the object	Text	73B57
Iconclass keywords	Keywords related to the piece's iconclass	Text	The Virgin with the Child stands on a crescent and holds a white rose in her right hand while Jesus shows a rosary
Signature	Existence of a signature of the author(s)	Bool	True

Table 3.6: Iconography template.

As previously mentioned, these properties were selected with the help of VICARTE, with the addition of the properties *Images filenames*, *Images descriptions*, *Image filename of plan* and *Image filename of elevation*, whose reason to be added will be explained in further detail in section 3.3. It was possible to create these properties and group them in templates correspondent to each table through the usage of Semantic MediaWiki in section 3.1.2.1, as it allows the administrators of the page to create properties and select their data type (e.g.:Number, Text...), along with letting them create templates for groups of properties.

It is possible to create these properties and templates from the Special Pages provided by the Semantic MediaWiki extension. Each property was created in the 'Create a property' special page, as seen in figure 3.4, and afterward, the properties were properly grouped in the 'Create a template' special page, as seen in figure 3.5, where it is possible to accomplish the previously mentioned structure.

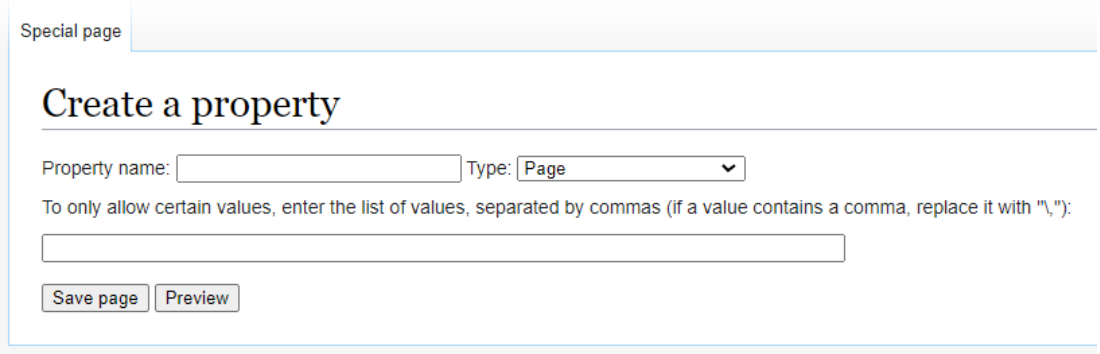


Figure 3.4: Section of the special page where each property was created.

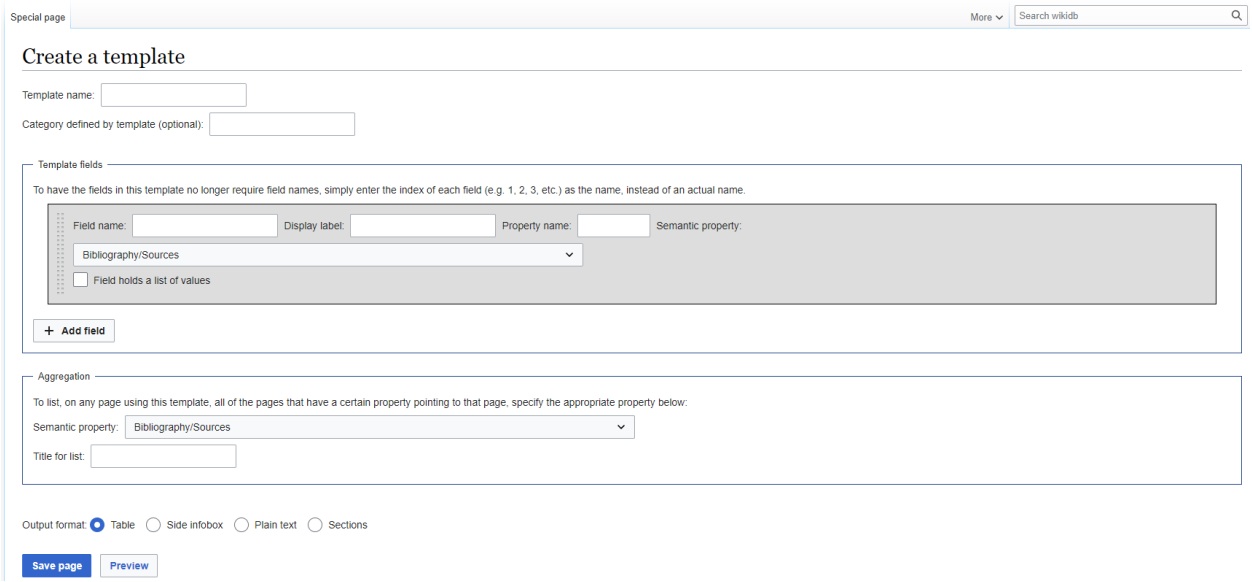


Figure 3.5: Section of the special page where each template is created by assigning it a group of existing semantic properties.

3.3 Creating and editing new pages

One large challenge that MediaWiki provided was making an intuitive and user-friendly way of creating and editing new pages, while still maintaining its Semantic properties. Unfortunately, MediaWiki does not provide a direct way of solving this problem. Although verified users can change the properties' values through a form in the page, the user would still have to make some changes in order for the page to be presentable and according to the rest of the other pages' format.

Ideally, the process of creating a new page should be automated, only requiring specific fields of information for the page to be presentable and with the same structure as all that were previously created and for the properties to be kept in the repository in a way that consequent queries regarding each created property give the proper results.

Upon thorough research, it was found that the best alternative and overall best solution to solve this problem was through the usage of the Page Forms extension of MediaWiki. This extension allows the creation of customizable forms for users to fill specific information about whatever the developer pleases. In this case, we can create a form that allows the user to give the value to each property of the stained glass piece by filling each field with the correct information. This tool, in this way, provides an easier and more intuitive form for users to integrate information about the stained glass piece they want to add to the platform.

With this approach, it is possible to create a page form with the previously created templates from each table of properties referred in section 3.2.1. Similarly to the special pages where each property and template were created, there is also a special page with an interface to create a form, as can be seen in figure 3.6.

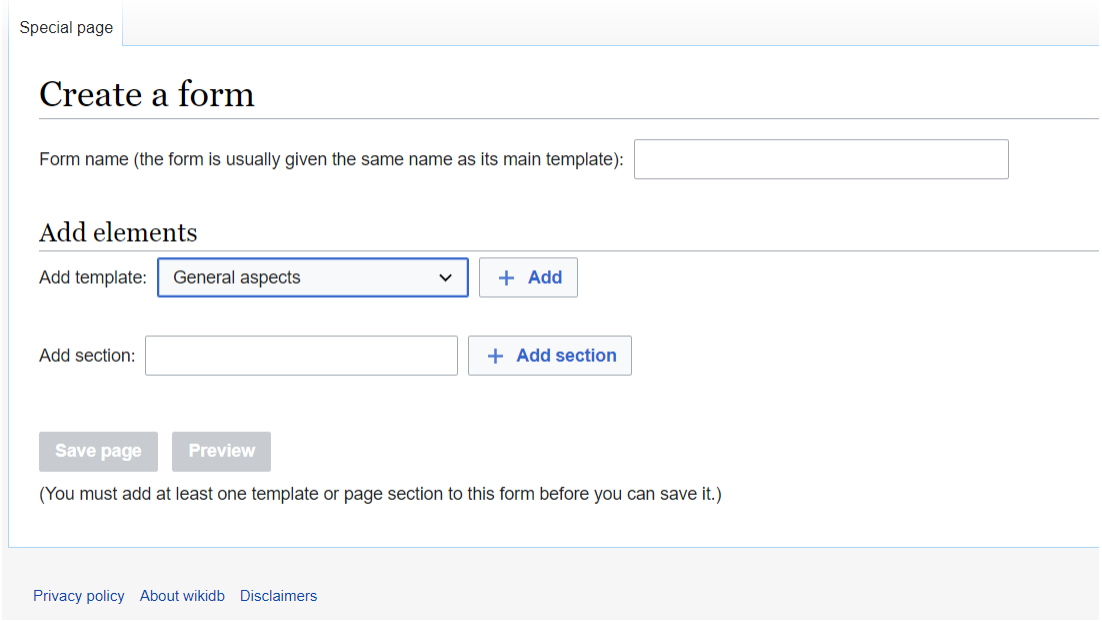


Figure 3.6: Section of the special page where the page creation form was created.

After creating the page form by adding each template, them being the General aspects, Photography/image, Location, Panel, Owner and Iconography templates, we simply have to customise the form and its input fields, in a way that best fits each property type and context. Although most properties can be simply described in the text format, there are other properties, such as *Registry Date* and *edm:currentLocation*, where it would be beneficial to have a different type of input than the 'text' input. Following the Page Forms extension's documentation page, we can select some input types to be used in our stained glass page creation form, which are:

- *textarea*: Input type used for larger excerpts of text, which can be used in the property *dc:description*;

- *dropdown*: Input type used for properties which can only hold a limited selection of values. This input type can be used in the properties *edm:country*, since there is only a limited amount of countries in the world, and *Signature*, as the only values it can hold are either *True* or *False*.

- *datepicker*: This input type is intended for properties of type date, where it lets the user pick the exact date through a Javascript based pop-up calendar. It can be used in the properties *Registry Date*, *Photo creation date*, *Date of origin (standard - beginning)* and *Date of origin (standard - end)*. This input type is useful not only because of how much more intuitive it becomes to pick a date, but it also standardizes the format of the date, making it easier for searches within the platform.

- *leaflet*: This input type displays a leaflet map which lets the user either normally write the pair of coordinates of a location, write the address of the location or click a location on the map to choose the coordinates to be stored in a certain property. This can obviously be used in the property *edm:currentLocation*.

- *combobox*: The combobox input type acts like a regular autocomplete text field, but with the addition of a down-arrow icon that acts like a dropdown. This is especially useful when we want a user to input a value from an already formed group of possible values, which can vastly improve how fast a user inputs a value. This can prove to be quite effective in all properties that hold values related to files that were previously uploaded into VitralWiki, them being *Image filename of plan*, *Image filename of elevation*, *History related image*, *Laboratorial PDF* and *Restoration and consevation history*. Using combobox and changing the possible values to those that belong in the *File* namespace, we can make the filling of this field more efficient.

- *tokens*: This input type is similar to the combobox input type, as it is also possible to restrict the possible values to a smaller selection. The difference is that this input type converts each inputted value into a token, allowing for the addition of multiple values instead of only one into a field. This is useful for the *Images filenames* property, for the same reasons as the ones referred in the combobox, but with the addition of it allowing the attribution of multiple values.

Having these options to use on our Page Form, we need to make a few changes on the templates that use files as properties. Given that we want our platform to be accessible by other endpoints and we want to maintain the Linked Open Data principles, it is important to convert the values inserted on the properties *Image filename of plan*, *Image filename of elevation*, *Laboratorial PDF* and *Restoration and conservation history* into URLs, since if someone is trying to access VitralWiki's data, the filenames will not provide enough information for someone who queries these properties, being much more useful to have access to the files URLs. For this, we can either change the fields in the Page Form to receive URLs instead of the filenames themselves or we can change how the template in which these properties are stored. In this case, the better option is to change the properties themselves inside the template, so that it converts the filename into the URL path to that file inside VitralWiki. This can be simply done by editing a template's code and adding the filepath to the beginning of the inputted value, which will change this property from saving the filenames into saving the filepath URLs.

After sorting out how the properties are stored, we can finally finish creating our Page Form for creating new pages, as a part of the resulting form can be seen in figure 3.7. Given that each resulting page from filling the form contains each of the templates that were mentioned in section 3.2.1, if it is necessary to change how a specific property is saved again, we can simply edit the template of that specific property, which will change all pages that were created with our Page Form, providing a very flexible solution that

allows us to create new properties for a page without compromising the integrity of each stained glass page.

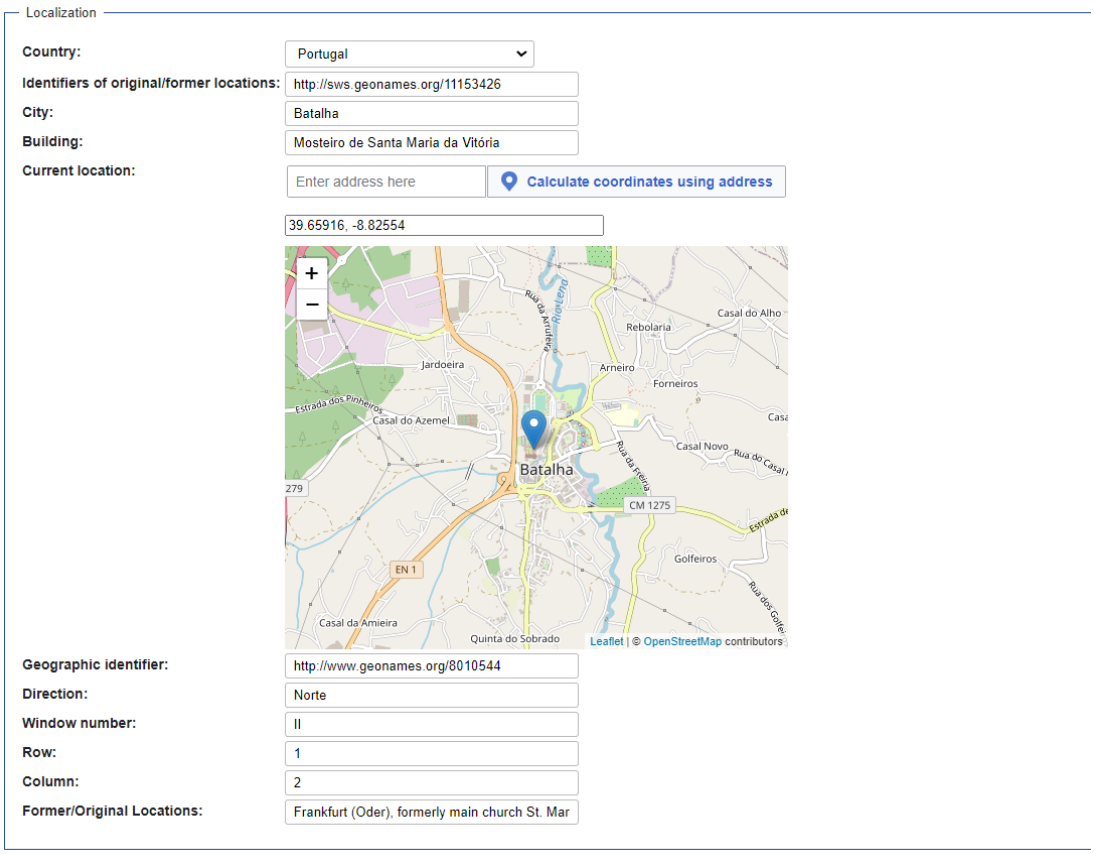


Figure 3.7: Location properties page form section.

However, this brings up one question: After filling out the forms, how is it possible to convert said information into a well-structured page that shares the same structure as all previously created pages and maintains all Linked Open Data properties? And since a page's properties do not need to have values for all properties, how is it possible to adapt to each case when creating a new stained glass page? This is the more challenging part of this approach, as a much more complex solution to our previous one will be necessary in order to integrate the information that the user puts in the form with an actual page. In this proposed solution, we want an automatic way of converting each property into a stained glass page.

After filling the page form, we get a resulting page with data tables, corresponding to the templates that were previously mentioned and each of their properties with the value the was given in said form. One example of one these tables can be seen in image 3.8, which is the end result of filling the properties for the *Location* template in the page form. This results in a rather basic layout, as most users would definitely prefer to, for example, view the image(s) of the stained glass panel and not only having access to those images filenames in the property *Images filenames*. In order to create a more interactive stained

glass page, it is possible to create a template that will receive all data that was inserted in the form and place it in a different structure.

Country	Portugal
State/Region	http://sws.geonames.org/11153426
City	Batalha
Building	Mosteiro de Santa Maria da Vitória
Current location	39.65916, -8.82554
Geographic identifier	http://www.geonames.org/8010544
Direction	Norte
Window number	II
Row	1
Column	2
Former/Original Locations	Frankfurt (Oder), formerly main church St. Marien, east choir, north II, 1c; Anger Museum Erfurt; Thuringia Museum Eisenach
Identifiers of original/former locations	
Localization	

Figure 3.8: Data table with the Location related properties in a stained glass page after filling the page form.

Thankfully, the 'Page Forms' extension provides a way of doing just this. By editing the source of the 'Page Forms' page that is used to create new stained glass pages, we can preload a template that receives each property that is filled in the form and adapts it in the new stained glass page. For this to happen, this new template page needs to be created and customised using wikitext and HTML.

In this way, it was decided to display the following properties in their own section, apart from the already generated data tables from filling the previously mentioned page form on its own. These properties are *Description*, *Current Location*, *Images filenames*, *Images descriptions*, *Image filename of plan*, *Laboratorial PDF*, *Restoration and conservation history* and *History related image*. The reason these properties in particular were chosen to have their specific section in the stained glass pages is related to the presentation and structure of said page. For example, even though one can see the filenames of the images related to the stained glass panel, it would be much better to be able to see the images themselves as well. Having made a decision on which properties will have their own section in every stained glass page, we can start developing our template to process the information in these properties.

In order to create a this template that will process the values of the properties and present them in a new stained glass page, we simply have to create a new page for this and edit its source code to do just this. To do this, we must have a good understanding of Wikitext [7], the 'Parser Functions' extension, which as described in section 3.1.2.5, Semantic MediaWiki's Parser Functions [19] and HTML, so that we can control what is shown in a page, on what conditions that is shown and how it is shown. It is important to understand the syntax and logic behind this template's code in order to also understand how it processes data.

Firstly, we can start by analysing how the properties *edm:country* and *dc:description* are processed in the template in the listing 3.1:

```
{{#default_form:Create a new stained glass page}}
{{#ifeq:{{PAGENAME}}|Preset| WARNING: DO NOT EDIT THIS PAGE |

<div class = "parent" style="width:100%;max-width:100%;max-height:100%;">
{{#ifeq: {{PAGENAME}}|Preset||[[Category:Stained glass]] }}
{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Edm:country}} }} | 0||
[[Category:{{#show:{{PAGENAME}}|?Edm:country}}]]}}
<div class="parent" style="max-width:100%;max-height:100%;">

{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Dc:description}}}}|0||
== Description ==
{{#show:{{PAGENAME}}|?Dc:description}}
<p/>
}}
...
```

Listing 3.1: Section of the stained glass page template's code regarding the properties *dc:description* and *edm:country*.

From this excerpt of code 3.1, it is important to understand how each function works. Since MediaWiki's markup language uses a mix of other markup languages, such as wikitext and HTML. It is also important to know how to call different functions from extensions within MediaWiki, which requires an understanding of those specific extensions. For example, the first line of this section of the page code specifies to the *Page Forms* extension 3.1.2.2, in which it adds the option of editing the page with the specified form, which in this case is the *Create a new stained glass page* form that we previously created. The two most important functions used in the template come from the extensions *Parser Functions* 3.1.2.5 and *Semantic MediaWiki* 3.1.2.1. It is also very important to understand how these functions work in order to control what information is displayed on a page and how this information is displayed. The one and only function used from the *Semantic MediaWiki* extension is the 'show' function. This function is used to display only a single property value for a single page. In this platform, this is extremely useful as we want to display some values of all stained glass pages properties created with the Page Form.

Getting into the *Parser Functions* extension's functions, we can start by explaining the 'ifeq' function. This function is used to compare two input strings. If the two strings are identical, one string is returned. If they are not identical, another value is returned. In this template, this function is often used with the 'len' parser function which, as the name suggests, prints out the length of a specific string. Knowing the purpose of these three functions, it is possible to understand how they are complementary to each other in the context of this platform. We can use the 'ifeq' function to verify if the length, which can be displayed with the 'len' function, of a specific property's value within a specific page, which can be displayed with the 'show' function, is equal to zero. In this way, we can control which information is displayed in each page that contains this template.

As previously mentioned, one of the most important parts of a web page regarding a stained glass piece would be the possibility to view its photo(s) or image(s). Assuming the user has already uploaded some images of the stained glass panel, we must find a way of displaying one or multiple images in the stained glass page that was created through the form. The easiest way that was found for it to be possible to display one or more images in a specific stained glass page started with the user writing the names of the images' filenames in the property *Images filenames*, where if there are multiple images, the user must separate them with a semi-colon. The same also applies to the property *Images descriptions*. In this way, we do not have to limit the user to display a limited number of images and we allow the user to display as many images as they please.

Even though the 'Parser Functions' extension has been proven helpful for this type of scenarios, it will not be sufficient to solve this problem, as this extension does not allow to iterate an array, which is necessary since the user can add as many images and their respective descriptions as they want. For it to be possible to iterate through both the values in the *Images filenames* and the *Images descriptions* properties, we will have to use the 'Scribunto' extension that we previously described in the section 3.1.2.6. In this way, it will be possible to create a *Lua* function that traverses through these properties. To do this, we simply need to create a new VitralWiki page in the 'Module' namespace. In this newly created page, we can create our Lua script that traverses through the properties values and divides them into an array based on a separator, functioning similarly to a 'split' function, and then returns the output in wikitext in a way that it can be inserted into a gallery using its syntax [5]. After this, we simply need to invoke this function in our template, sending *Images filenames* and *Images descriptions* as arguments to this function, as seen in the listing 3.3.

```lua
local p = {}

function p.images(frame)
       local inputstr1= frame.args[1]
       local inputstr2=frame.args[2]
   local t={}
   for str in string.gmatch(inputstr1, "([^"..","..."]+)") do
          table.insert(t, str)
   end
   local u={}
   for str in string.gmatch(inputstr2, "([^"..";"..."]+)") do
          table.insert(u, str)
   end
   local result=""
   for k, v in ipairs(t)
   do
      if u[k] ~= nil then
             result = result.."File:"..t[k].."|thumb|"..u[k].."\n"
      else
             result = result.."File:"..t[k].."\n"
      end
   end
   return result
end

return p
```

Listing 3.2: Lua script created for the properties *Images filenames* and *Images descriptions*.

```
...

{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Images filenames}} }}|0|
{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?History related image}} }}|0|
|[[File:{{#explode:{{#show:{{PAGENAME}}|?History related
    image}}|/|7}}|thumb|500x500px|History related image]]}}
    |{{#ifeq:{{#pos:{{#show:{{PAGENAME}}|?Images filenames}}|, }}||
<div style="float:left" >
[[File:{{#show:{{PAGENAME}}|?Images
    filenames}}|thumb|500x500px|{{#show:{{PAGENAME}}|?Images descriptions}}]]
{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?History related image}} }}|0|
    |[[File:{{#explode:{{#show:{{PAGENAME}}|?History related
    image}}|/|7}}|thumb|500x500px|History related image]]}}
</div>|<div
    style="min-width:500px;max-width:500px;position:relative;float:left">
{{#tag:gallery|{{#invoke:Images|images|{{#show:{{PAGENAME}}|?Images
    filenames}}|{{#show:{{PAGENAME}}|?Images descriptions}}}}}|mode=slideshow|
    widths=500| heights=500 }}
{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?History related image}} }}|0|
    |[[File:{{#explode:{{#show:{{PAGENAME}}|?History related
    image}}|/|7}}|thumb|500x500px|History related image]]}}
</div>
 }}
}}

...
```

Listing 3.3: Section of the stained glass page template's code regarding the properties *Images filenames*, *Images descriptions* and *History related image*.

Now that the properties *Images filenames* and *Images descriptions* have been dealt with, we have to find a way of showing the property's *History related image* value, in the format of an image as opposed to an image URL. Given that all files uploaded in VitralWiki have similar file paths, we can take away the filename from the URL using the 'explode' parser function, as it can be used to split a given string into pieces based on a delimiter. This way, we can use the '/' character as a delimiter and return the filename of the page, so that we can display it using MediaWiki's 'File' format.
Finally, we need to add a way of displaying the remaining properties on each stained glass page. We can do this similarly to what has been previously done, using the 'ifeq' parser function to verify if each property has non-empty values and displaying information based on the non-empty values. In this case, since we want to display a stained glass piece's location on a map, we can use the *Maps* extension, referred in section 3.1.2.3, to display a marker on the map based on the coordinates stored in the property's *edm:currentLocation* value. Along with this, it is also possible to add a tooltip when the marker is clicked on, where we can add the image of the plan of the building or the image of the elevation of

the building, depending on which has a value. The code that processes these properties can be seen in the listing 3.4.

```
...

{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Edm:currentLocation}}}}|0|

{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Image filename of
    plan}}}}|0|{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Image filename of
    elevation}}}}|0||
[[File:{{#explode:{{#show:{{PAGENAME}}|?Image filename of elevation}}|/|7}}
|thumb|500x500px|Elevation of {{#show:{{PAGENAME}}|?Building}}]]}}
|{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Image filename of elevation}}}}|0|
[[File:{{#explode:{{#show:{{PAGENAME}}|?Image filename of plan}}|/|7}}
|thumb|500x500px|Plan of {{#show:{{PAGENAME}}|?Building}}]]
|
[[File:{{#explode:{{#show:{{PAGENAME}}|?Image filename of
    plan}}|/|7}}|thumb|500x500px|Plan of {{#show:{{PAGENAME}}|?Building}}]]
[[File:{{#show:{{PAGENAME}}|?Image filename of
    elevation}}|thumb|500x500px|Elevation of {{#show:{{PAGENAME}}|?Building}}]]
}}
}}
|
<div style="border:5px solid #bada55;border-radius: 10px;box-shadow: 0 0 10px
    rgba(0, 0, 0, 0.5); float:right">

{{#display_map:{{#show:{{PAGENAME}}|?Edm:currentLocation}}
    {{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Image filename of plan}}}}
|0|{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Image filename of elevation}}}}|0|
|~[[File:{{#explode:{{#show:{{PAGENAME}}|?Image filename of
    elevation}}|/|7}}|200x200px]]}}
|~[[File:{{#explode:{{#show:{{PAGENAME}}|?Image filename of
    plan}}|/|7}}|200x200px]]}}
|center={{#show:{{PAGENAME}}|?Edm:currentLocation}}|width=400|height=300}}

</div>
{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Image filename of plan}}}}|0|
    |{{#ifeq:{{#len:{{#show:{{PAGENAME}}|?Image filename of
    elevation}}}}|0|||[[File:{{#explode:{{#show:{{PAGENAME}}|?Image filename of
    elevation}}|/|7}}|thumb|500x500px|Elevation of
    {{#show:{{PAGENAME}}|?Building}}]]}}
}}}}

...
```

Listing 3.4: Section of the stained glass page template's code regarding the properties *Image filename of plan*, *Image filename of elevation* and *Edm:currentLocation*.

3.4 Searching pages within VitralWiki

With the creation of an intuitive and automatic way that lets users create stained glass pages and maintains consistency between all created stained glass pages, it is important to test that we are integrating our platform into the Semantic Web and that its principles are being implemented. A good way to, not only test this, but also add another important tool to our platform is by creating a page where one can make a Semantic Search.

Although Semantic MediaWiki already provides a page for Semantic searches with the Semantic search special page, it is not a very good option for this platform, as using it requires understanding how to make queries upon our platform's properties and an understanding of Semantic MediaWiki's query language [20], which is not ideal given that we want our platform to be as user-friendly as possible. Another option that is provided by Semantic MediaWiki is the Search by property special page, which lets one search for pages based on a specific's property's value, where a user only needs to specify a property and its value. This is also not ideal, since this solution has very limited options and only lets the users search by only just one property, where it would be best to let the users make queries to all the properties they desire.

Knowing that neither of the solutions provided by the Semantic MediaWiki extension 3.1.2.1 are sufficient to accomplish a complete and user-friendly Semantic search interface, we will have to create our own. We will need to create a couple of pages for this to be possible: One page where the user inputs the properties values and submits the search parameters and one page where the query being made and the data is being processed.

For the user interface where the user can input the values for the properties, we can use a similar solution to the one used in the section 3.3. It is possible to create a Page Form 3.1.2.2 similar to the one used to create new stained glass pages, where the user only needs to fill the values in the fields of the properties they want to query. To this, we simply need to copy most of the source code of the Page Form that is used to create new stained glass pages and put in a new Page Form for the purpose of querying stained glass pages within VitralWiki. Of course, we won't need to load our previously created template 3.1 for the new pages in this Semantic search page, as this will serve a much different purpose. Since by default Page Forms' pages are used to create new pages based on the entered values on the fields, in this case we can use the Run Query special page for this newly created form, whose purpose is specifically for running queries based on the values put in the form's fields.

Now that we created our user interface for the Semantic search, we simply need to create a new template where the data that was inserted with said form is queried based on our own criteria. To have a more complete solution, it would also be ideal to have the option of querying with both an *AND* and an *OR*. This means that it is important that users can search pages where all of the conditions have to meet, this is, all properties from the pages returned must have the same values as the ones inputted in each property field

in our Semantic search for, and it is also important that they can search pages where any of the fields values are the same as the properties' values from stained glass pages. For this purpose, additionally to the fields corresponding to each property in a stained glass page, we added a checkbox input called 'Intersection' that decides whether the query is done with an *AND* condition, which happens when the this checkbox is ticked, or whether it is done with an *OR* condition, which happens when the checkbox is not ticked. Having added this additional field to both the Page Form and the page Template, we can analyse the query in listing 3.5.

```
{{#ifeq:{{{Intersection}}}|No|
{{#ask:
[[Category:Stained glass]]
[[Dc:title::~*{{{Title}}}*]]
||[[Dc:description::~*{{{Description}}}*]]
...
||[[Iconclass keywords::~*{{{Iconclass keywords}}}*]]
||[[Signature::{{{Signature}}}]]

|limit=50
|offset=0
|sort=
|order=asc
|format=ul
|searchlabel=No pages found.
|default=No pages found.
}}
|
{{#ask:
[[Category:Stained glass]]
 {{#if: {{{Title|}}} | [[Dc:title::~*{{{Title}}}*]] }}
 {{#if: {{{Description|}}} | [[Dc:description::~*{{{Description}}}*]] }}
 ...
 {{#if: {{{Iconclass keywords|}}} | [[Iconclass keywords::~*{{{Iconclass
    keywords}}}*]] }}
 {{#if: {{{Signature|}}} | [[Signature::{{{Signature}}}]] }}
|limit=50
|offset=0
|sort=
|order=asc
|format=ul
|searchlabel=No pages found.
|default=No pages found.
}}
}}
 ...
```

Listing 3.5: Semantic search template.

In this template's code, we are using the Semantic MediaWiki's parser function 'ask', which can be used to make inline queries upon the properties within our platform. This way, we create two separate queries using the 'ask' function, one that is done when the 'Intersection' checkbox is ticked and one that is done when the checkbox is not ticked. The difference between these queries lies in the conditions of each query. It is also important to understand the purpose of the characters '*' and ' ' in this query to understand exactly what is being returned from each query. If we take a look at line 2 from 3.5, what this line can be translated to in summary is that it searches for pages that do not have an exact match value of the 'Description' value, which is the one that is filled in the page form, in the 'Dc:description' property, but that have a value in the 'Dc:description' property that contains the value of the template parameter. Also note that both of these queries do not return any pages if the fields of the form are empty.

3.5 User rights

Having now a way that allows users to intuitively create and edit stained glass pages, it is immensely important to control which users are allowed or not to do just this. As mentioned throughout this thesis, creating and editing stained glass pages is something that is only destined to recognised specialists of stained glass. Having this in mind, the main goal here is to give the permissions to create and edit pages to specialists users, while only letting registered and unregistered users view these pages and their contents. With this, it is easy to understand there must be two distinct groups with different rights: One group for stained glass specialists, and one group the involves the rest of the users.

Having these two groups decide, it was simple to change the permissions through the 'LocalSettings.php' file. This file is extremely important to set every configuration in the MediaWiki platform, such as which extensions should be loaded, what database system is being used and how data is being stored and of course, which permissions each user group has. With this, assuming that users identified as stained glass specialists belong to the *Administrator* and the following belong either to the *User* group (Registered users) or the * group (Anonymous users), we simply have to remove the ability of creating, editing and deleting pages from the latter groups by using the same syntax as explained in [6].

Having these permissions defined, we only need to find a way to change the user group of the registered stained glass specialists users from the normal user group to the *Administrator* user group. The best way found to differentiate a specialist from a regular user was by sending an email with their username and with proof of being a stained glass specialist to the platform's developer, who can easily change that user's user group by using the 'User rights' special page, as seen in the image 3.9.

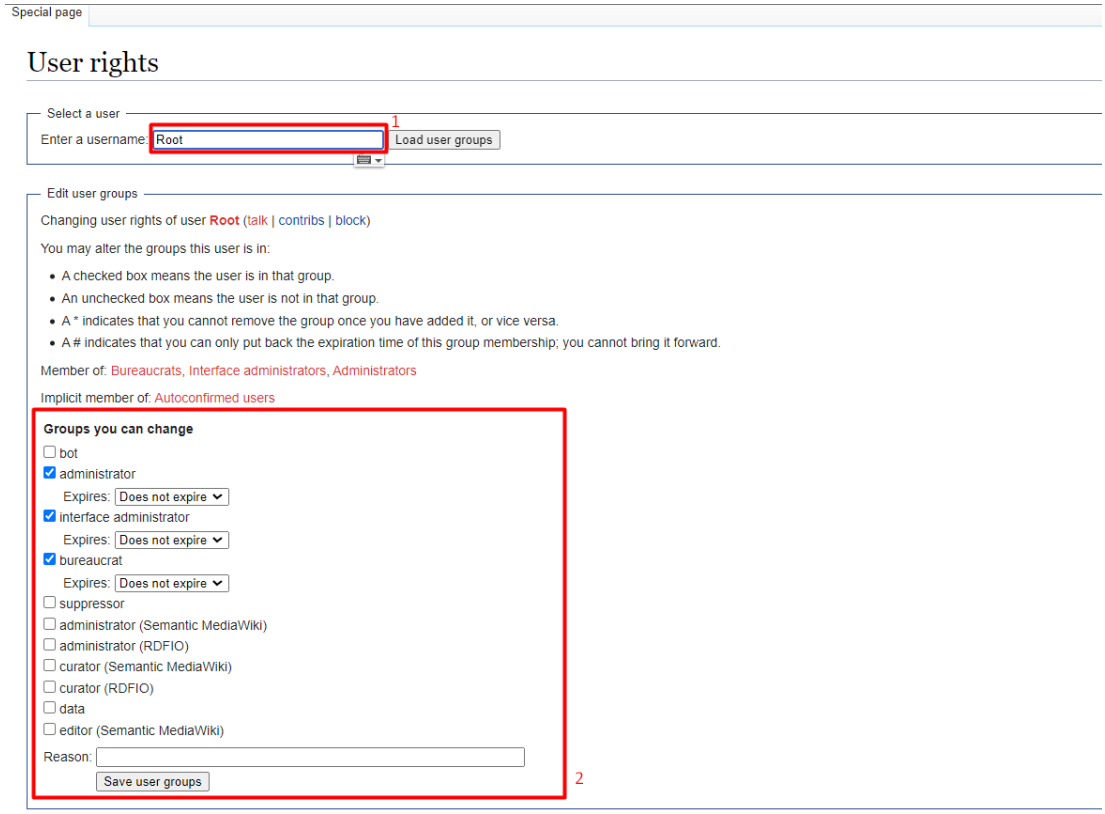


Figure 3.9: Excerpt of MediaWiki's special page where the 'root' user can change a certain user's user group. 1- Field to search for the user's username. 2- User groups the user belongs to

3.6 Linking to other endpoints

Another key aspect of this platform is being able to view data regarding glass art from other endpoints, of each most important is Europeana's. In order to achieve this, as referred in the section 3.1.2.4, we need to create a configuration in the 'LocalSettings.php' file to connect with Europeana's endpoint, so that the RDF triples inside their records can be accessed.

```
$wgLinkedWikiConfigSPARQLServices['http://sparql.europeana.eu'] = array(
        'debug' => false,
        'isReadOnly' => true,
        'endpointRead' => 'http://sparql.europeana.eu',
        'login' => '',
        'password' => '',
        'typeRDFDatabase' => 'mysql',
        'HTTPMethodForRead' => 'GET',
        'storageMethodClass' => 'WikidataStorageMethod',
        'nameParameterRead' => 'query',
        'lang' => 'en'
);
```

Listing 3.6: Configuration to access Europeana's endpoint

After connecting to the said endpoint, it is possible to freely query it through LinkedWiki, as described in section 3.1.2.4. In order to be able to query the RDF triples within their records it is also important to know how the data model itself is structured, as it is impossible to query Europeana's records if we do not know the properties' names. Since we have access to the Europeana Data Model in [10], we can see the hierarchy of its properties and their respective names.

Having now access to both the endpoint and the data model of Europeana, we only have to create queries in order to represent data by the criteria we want. Europeana's records contain much more than what is needed for this platform, as it contains a massive digital heritage collection. Within these records, the ones that are important for the context of this project are the ones related to glass art and stained glass.

One unfortunate limitation that comes with LinkedWiki and querying other endpoints, in general, is that the person that is querying the endpoint and trying to reach its records data has to know how to program in SPARQL, since it is impossible to do so otherwise. For this reason, two pages were created that show Europeana's glass art data and its most relevant properties' values. This way, we created a page that shows Europeana's glass art records in a data table as seen in figure 3.10 and its code in listing 3.7 and another that shows those same glass art records on a leaflet world map, as seen in figure 3.11 and its code in 3.8. Note that due to different records having different properties and values, not all records will hold coordinates to the glass art piece's location since not every record holds these coordinates.

```
{{#sparql:
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?DataProvider ?Title ?Description ?Image ?Date ?Provider
WHERE {

  ?Proxy dc:subject ?subject .
  OPTIONAL{?Proxy dc:title ?Title .}
  OPTIONAL{?Proxy dc:description ?Description .}
  ?Proxy ore:proxyIn ?Aggregation .
  ?Aggregation edm:aggregatedCHO ?ProvidedCHO .
  ?Aggregation edm:dataProvider ?DataProvider .
  OPTIONAL{?Proxy dc:date ?Date .}
  OPTIONAL{?Aggregation edm:object ?Image .}
  OPTIONAL{?Aggregation edm:provider ?Provider .}

  FILTER(?subject ='Stained glass' OR ?subject ='Glasmalerei' OR
  ?subject = 'Glassmaleri' OR ?subject = 'Gebrandschilderd glas' OR ...
  OR ?subject = 'http://data.europeana.eu/concept/base/53')
}
LIMIT 10000
|options=colstyle=col3_img_display:block
!col3_img_max-width:20px!col3_img_border-radius:50%
!col3_img_margin:auto!col3_img_opacity:0.5
|endpoint=http://sparql.europeana.eu
|chart=bordercloud.visualization.DataTable
|log=2
|debug= true
}}
```

Listing 3.7: SPARQL query for Europeana's glass art records in a data table

```
{{#sparql:
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
PREFIX ore: <http://www.openarchives.org/ore/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?lat ?long (COALESCE(?title,'No title present') as ?title)
(COALESCE(?description,'No description present') as ?description)
?ProvidedCHO ?image
WHERE {
   ?p rdfs:subPropertyOf* ?Place ;
      wgs84_pos:lat ?lat ;
      wgs84_pos:long ?long .
   ?Place rdf:type edm:Place .
   ?Proxy ?property ?Place .
   ?Proxy dc:subject ?subject .
   OPTIONAL{?Proxy dc:title ?title .}
   OPTIONAL{?Proxy dc:description ?description .}
   ?Proxy ore:proxyIn ?Aggregation .
   ?Aggregation edm:aggregatedCHO ?ProvidedCHO ;
      edm:object ?image .
  FILTER( ?subject ='Stained glass' OR ?subject ='stained glass'
  OR ?subject ='Glasmalerei' OR ?subject = 'Glassmaleri' OR ...
  OR ?subject = 'Gebrandschilderd glas')
}
LIMIT 10000
|endpoint=http://sparql.europeana.eu
|cellspacing=3
|options=width=100%!height=500px!colstyle=col1_img_display:block!
col1_img_max-width:250px!col1_img_border-radius:50%!
col1_img_margin:auto!col1_img_opacity:0.5
|chart=leaflet.visualization.Map
|log=2
|debug= true
}}
```

Listing 3.8: SPARQL query for Europeana's glass art records in a map

Show 10 entries Search:

DataProvider	Title	Description	Image	Date	Provider
Rijksdienst voor het Cultureel Erfgoed	Rooms-Katholieke Sint Willibrorduskerk	Interieur, gebrandschilderd glas		2007-02-02	CARARE
Rijksdienst voor het Cultureel Erfgoed	Rooms-Katholieke Sint Willibrorduskerk	Interieur, gebrandschilderd glas		2007-02-02	CARARE
Rijksdienst voor het Cultureel Erfgoed	Rooms-Katholieke Sint Willibrorduskerk	Interieur, gebrandschilderd glas		2007-02-02	CARARE

Figure 3.10: Small section of the resulting data table from a SPARQL query to Europeana's endpoint. 3.7

Figure 3.11: Resulting leaflet map from a SPARQL query to Europeana's endpoint. 3.8

As mentioned in section 3.1.2.4, there are some limitations to the information returned from these queries, particularly the one that results in a leaflet map with markers correspondent to each stained glass piece. There are very few markers placed in the map that result from that query due to how data is stored in Europeana. Since Europeana is a virtual library where many entities can add their content to Europeana and because it also contains a large amount of information regarding not only stained glass but many other cultural and art-related content, its stored data is not standardized. For example, in the previous query, we are filtering all results so that the property's *dc:subject* value is 'Stained glass', but since all data comes from mostly different sources, some entities, when uploading stained glass galleries, do not give a value to the property *dc:subject*. These way, it becomes difficult to retrieve all stained glass-related data from Europeana's endpoint, as there is no way to feasibly filter all stained glass-related data, as making additional filters will make the query much more complex and slow. Another reason and also the main reason for there being so few markers on the leaflet map that results from the query

in 3.8 is due to the fact that very few entities give coordinates to the entries they upload in Europeana. For this reason, in addition to the previous, we are left with a very small collection of stained glass retrieved from Europeana's endpoint.

3.7 Summary

In this chapter, a solution for the creation of a platform that stores stained glass-related data and implements the Semantic Web principles was presented and implemented. The resulting solution comprises the creation of a MediaWiki-based platform that stores stained glass-related data using MySQL and as a collection of extensions that help enrich the platform and maintain a Semantic web. This platform lets its users view and analyze stained glass content, where its content is either represented in the pages that were created using a Page Form by verified users who are stained glass specialists or it is displayed in entries that are accessed via SPARQL queries and stored in Europeana's repository. With this, we can proceed to test this platform thoroughly, as will be described in section 4.

4

Evaluation and Results

In this chapter, we will describe how the platform developed in this thesis was tested and how well it performed according to the target users of said platform. It must be taken into account that the usability aspect of this platform was evaluated and, despite not being the main focus of this thesis, it was still important to receive feedback from users regarding this aspect.

4.1 Changing environment

Since the MediaWiki platform we have developed until this point was only tested in a local environment, for it to be possible to fully test the platform, we must deploy it in a different environment. This way, multiple users will be able to test the functionalities of the platform and evaluate how it fares when doing so. For this part of the testing, a Debian 11 server was provided by the NOVA School of Science and Technology's Computer Science department and it was set up with the help of professor João Leitão. This server's specs are listed in appendix B.

4.2 User tests and evaluation

4.2.1 Testing limitations

Our platform was tested by two different groups: A group of people from various backgrounds and a small selection of stained glass specialists from VICARTE, as can be seen in the table in figure 4.1. For the non-specialists' tests and evaluations, we had a group of 14 voluntary participants with ages that range between 19 and 29 years old. In the case of the specialists, there were 3 members of VICARTE testing this platform. Even though VitralWiki was specifically designed for stained glass specialists to edit and create new pages, it was decided to widen the selection of people testing the platform so that we can also have a wider amount of results and feedback. When evaluating the results from the user tests, we must in this way take into account the small amount of participants that was possible for us to invite to test this platform.

Age	Gender	Background
28	Female	Project management
21	Male	Biochemistry
23	Male	Computer Science and Engineering
23	Male	Computer Science and Engineering
29	Male	Web developing
22	Male	Computer Science and Engineering
23	Male	Multimedia manager
24	Male	Programming
20	Male	Electrical engineering
27	Male	Customer support
19	Male	High school student
23	Male	Eletronics
24	Male	Portuguese Airforce member
34	Female	Investigator with Master's Degree and PhD in Stained glass
30	Female	Investigator with Master's Degree and PhD in Stained glass
57	Male	Curator at the Monastery of Batalha

Table 4.1: List of users that tested VitralWiki.

4.2.2 Usability evaluation

In this initial part of the tests where we want to evaluate the usability aspect of VitralWiki, both of the groups participated in this testing. To test the intuitiveness and usability of our platform, we decided to create a group of tasks that each user has to complete. After completing these tasks, the users filled out a form to evaluate the page in various aspects, as will be described later on. To aid users while using VitralWiki, a small manual was provided explaining how to do each task in the platform, as can be seen in A. The tasks asked for the user to perform were as follows:

1. **Create a new account and log into it**
   The user has to access the website where this platform is deployed, create a new account, and log into their account. After this, the user has to inform the developer, so that he can give the new account permission to create and edit pages

2. **Upload the files for a new stained glass page**
   The user has to access the 'Upload files' section of VitralWiki and upload all the stained glass files related to the page that is about to be created

3. **Create a new stained glass page**
   The user is given a set of values to be added to a new stained glass page and is asked to create a new stained glass page by filling out the Page Form we developed and described in our previous chapter;

4. **Edit information on the newly created page**
   The user is asked to edit a property's value inside the page, where the interface is the same as the one used when creating pages

5. **Test the searching stained glass pages with the Semantic Search feature**
   The user is asked to test the Semantic Search page inside VitralWiki, testing its semantic capabilities and integrity, and level of difficulty when searching for pages inside the platform

6. **Test searching and looking into Europeana's stained glass records**
   The user is asked to navigate to the Europeana records section inside VitralWiki and see how fast the data is loaded and displayed in VitralWiki

After finishing through following each task, each user was asked to fill out a form inspired by the System Usability Scale [3], where there is a group of questions to which the user has to give a rating of 1 to 5, based on different criteria. The questions are as follows:

1. **Level of difficulty when uploading files in VitralWiki**
   The user is asked to give a rating from 1 to 5, where 1 is very easy and 5 is very difficult, on the level of difficulty behind the upload of stained glass-related files into VitralWiki.

2. **Level of difficulty when creating pages in VitralWiki**
   The user is asked to give a rating from 1 to 5, where 1 is very easy and 5 is very difficult, regarding the level of difficulty of filling out the Page Form that was mentioned in the previous chapter in order to create a new stained glass page.

3. **Level of difficulty when editing pages in VitralWiki**
   The user is asked to give a rating from 1 to 5, where 1 is very easy and 5 is very difficult, regarding the level of difficulty of editing a stained glass page's properties' values.

4. **Level of difficulty when searching pages by property in VitralWiki**
   The user is asked to give a rating from 1 to 5, where 1 is very easy and 5 is very difficult, on the level of difficulty there is when using the Semantic Search mechanism that was developed to search stained glass pages by property.

5. **Level of satisfaction with the VitralWiki's pages in terms of structure**
   The user is asked to give a rating from 1 to 5, where 1 is very unsatisfied and 5 is

very satisfied, regarding the platform's structure, both in each individual page and the stained glass pages' contents.

6. **Level of satisfaction with the VitralWiki's pages in terms of aesthetics**
   The user is asked to give a rating from 1 to 5, where 1 is very unsatisfied and 5 is very satisfied, regarding the platform's aesthetics, according to how pleasing it is visually.

7. **Level of satisfaction with VitralWiki's responsiveness (satisfaction with how fast the platform is and how well the platform fares in what you want it to do)**
   The user is asked to give a rating from 1 to 5, where 1 is very unsatisfied and 5 is very satisfied, regarding the platform's responsiveness and speed when uploading files, creating new stained glass pages, and retrieving data from Europeana's endpoint through the Europeana's section of VitralWiki.

8. **Comments and suggestions regarding the previous questions**

After each group finished completing the tasks and filling out the evaluation form, the results from the form are as follows:

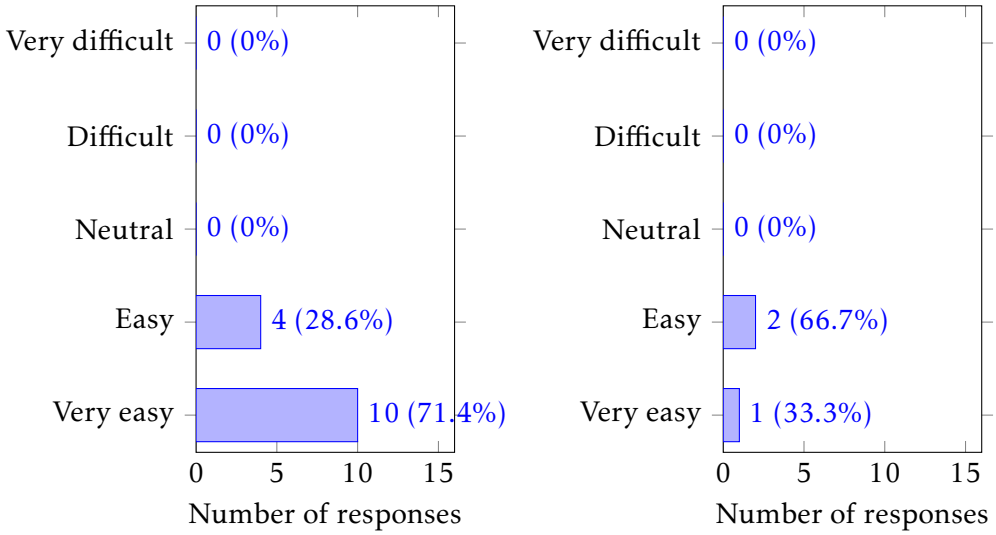


Figure 4.1: Results in the level of difficulty when uploading files for non-specialist and specialist users, respectively

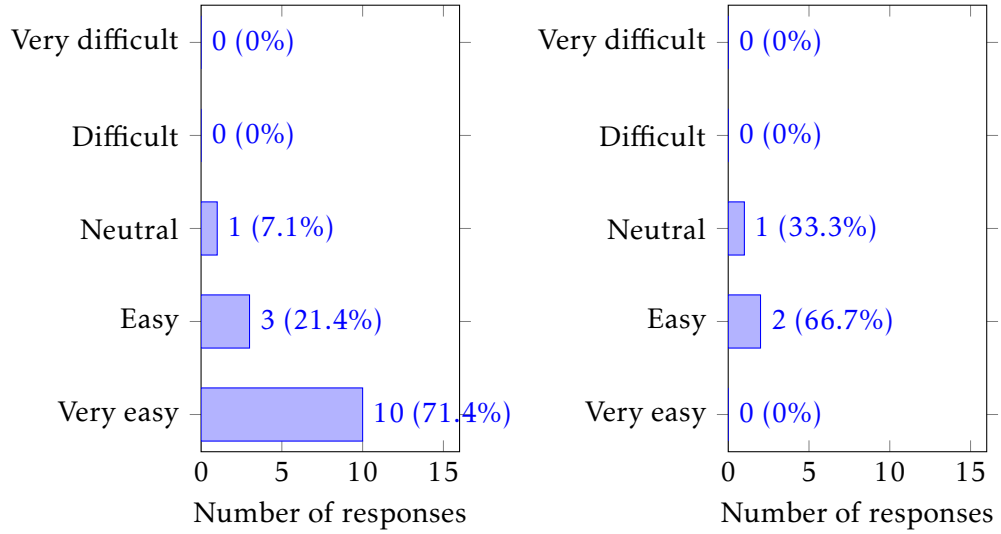


Figure 4.2: Results in the level of difficulty when creating new stained glass pages for non-specialist and specialist users, respectively

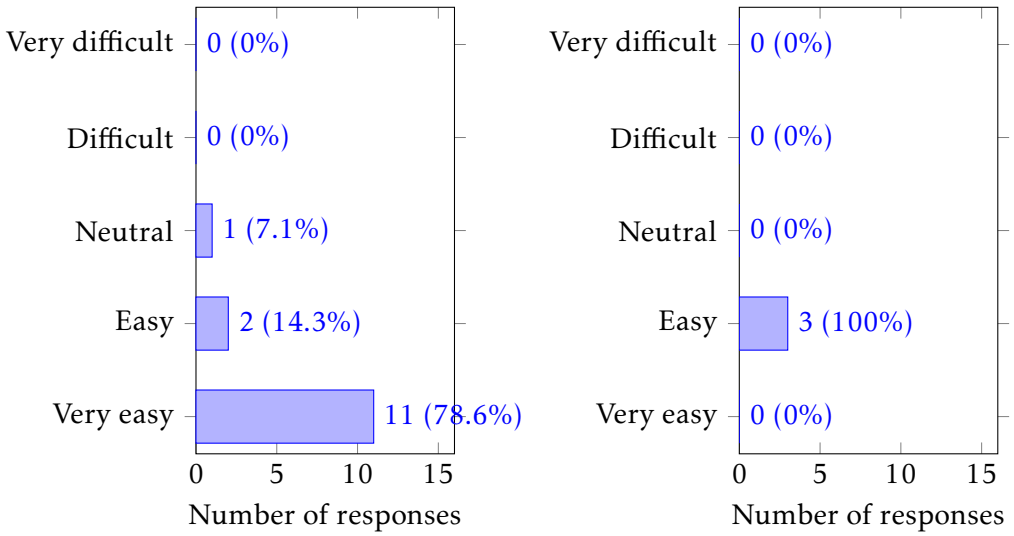


Figure 4.3: Results in the level of difficulty when editing stained glass pages for non-specialist and specialist users, respectively

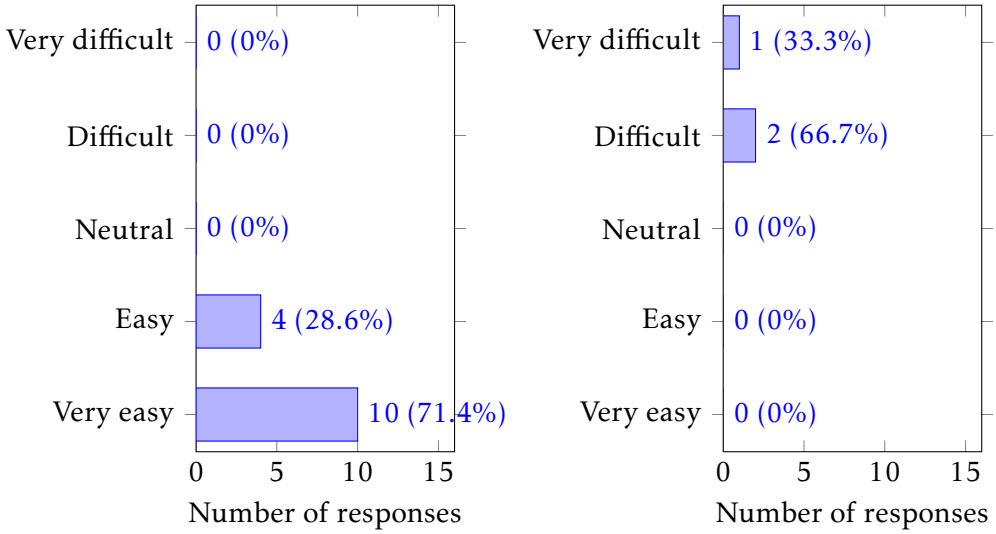


Figure 4.4: Results in the level of difficulty when searching stained glass pages by property for non-specialist and specialist users, respectively

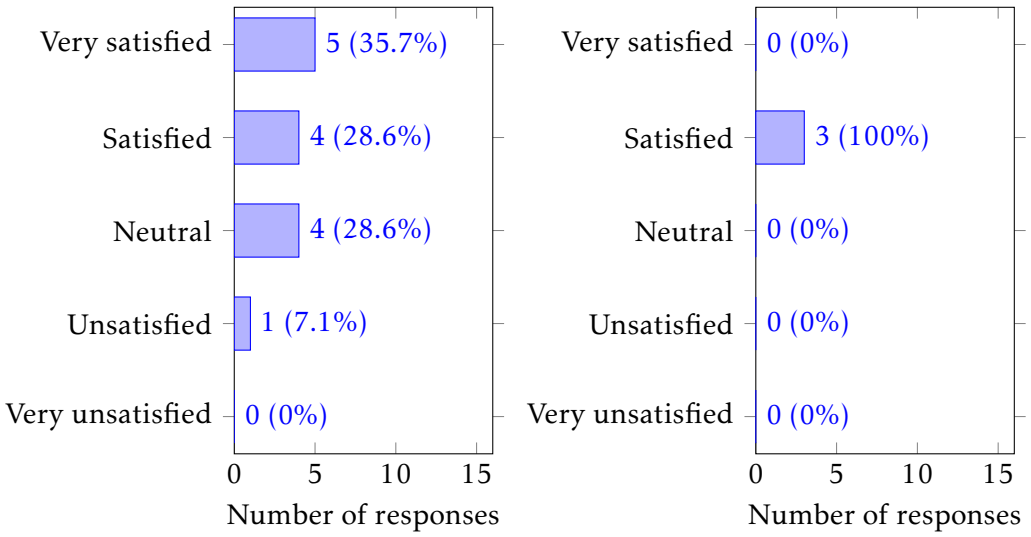


Figure 4.5: Results in the level of satisfaction regarding the structure of the platform's data for non-specialist and specialist users, respectively.

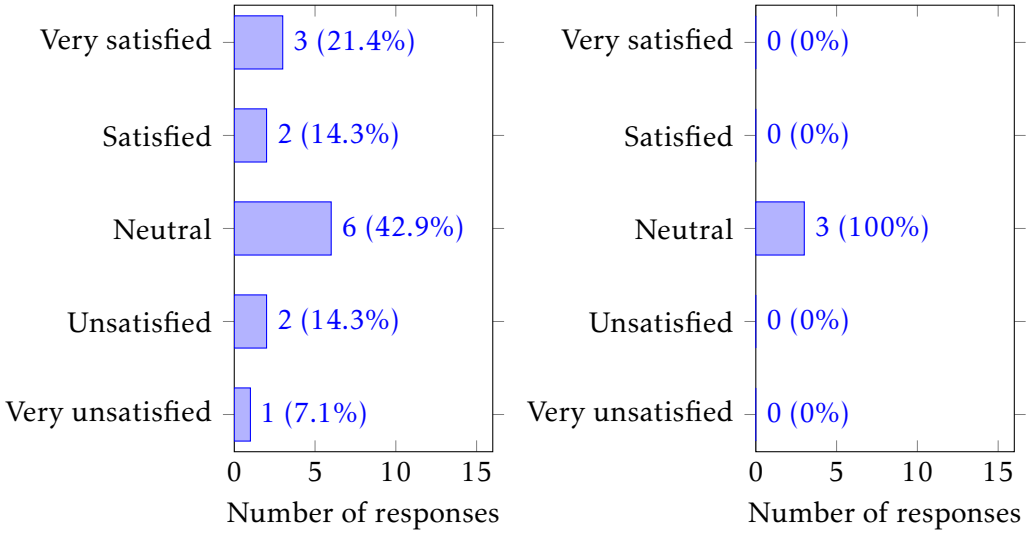


Figure 4.6: Results in the level of satisfaction regarding the platform's aesthetics for non-specialist and specialist users, respectively.

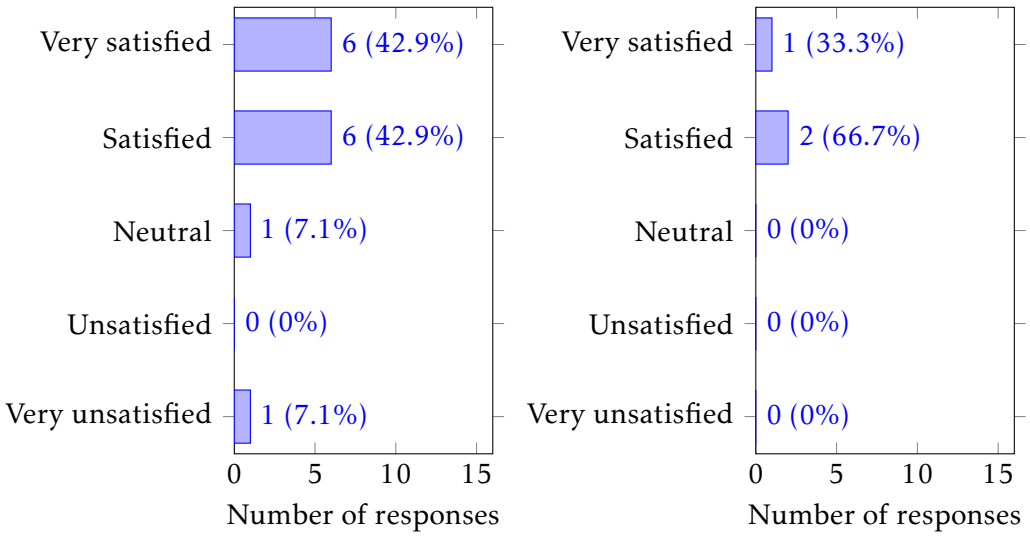


Figure 4.7: Results in the level of satisfaction regarding the platform's responsiveness for non-specialist users

From the responses to the first question, as seen in 4.1, we can see that the process of uploading files into VitralWiki is rather easy. MediaWiki, by default, has the option of only letting users upload one file at a time. This could be rather time-consuming, as stained glass pages can have a large number of files associated with them. This problem was noticed during the beginning of the user tests, so we decided to add an extension to MediaWiki called SimpleBatchUpload, which, to put it shortly, completely removes this problem and allows the user to upload all files they desire at once, instead of one at a time.

By analyzing the second question, we can see that the results in figure 4.2 are very

positive overall when it comes to the difficulty of creating new stained glass pages in VitralWiki. This leads us to conclude that the final solution provided for the creation of stained glass pages and maintaining the integrity of the data in our platform was quite efficient and intuitive, which was our main objective when developing it. Since editing stained glass pages is very similar to creating new pages, the results of the third question are rather similar to the ones in the third as can be seen in figure 4.3, since the interface for editing pages is the same as the one for creating new ones.

The results we got from the fourth question in figure 4.4 are mostly as positive as the results we have gotten so far, with the exception of the specialist users which consider using the Semantic Search feature of our platform difficult. This could be due to it being more complex than the current standard Web search mechanism, as it is a more focused method of searching for content that has more parameters for searching. Despite this, it is still extremely important to implement this search method in VitralWiki, as it is more precise when returning results and it is especially important in the context of this thesis.

By moving into the results of the fifth question in figure 4.5, we can see that the results are once again mostly positive regarding each user's opinion on the platform's structure, although not as uniformly distributed.

By analyzing the sixth question's results in figure 4.6, regarding the platform's overall aesthetics, it can be concluded that, despite the results not being negative, it is the aspect of the platform that users find the weakest on average. This could be due to MediaWiki being an open-source framework to create wikis, which by itself also brings some limitations regarding the way we can change the aesthetics and overall presentation of each page within our platform, bringing a website style very similar to Wikipedia and other Wikimedia websites that use this same framework.

In the figure 4.7 we can see that the level of satisfaction with the platform's performance is positive for the majority of our users. We must also take into account the server's specs in appendix B, as they can affect the server's overall performance to a great extent.

Finally, the responses received from the open text question by the non-specialist users were short comments all regarding the overall aesthetics of each page. It is also important to keep in mind that not all users made comments or suggestions regarding the previous questions. There was a list of suggestions from one of the specialist users which tested VitralWiki, regarding the Semantic search feature. The main suggestion was to create an auto complete feature, such that it already gives all possible options to each field depending on the already created stained glass pages. Although this solution could prove to be rather difficult to implement, it could definitely improve how easy it is to search by property in VitralWiki.

4.2.3 Specialist specific evaluation

When testing with stained glass specialists, we must keep in mind that they are our main target audience for this platform. In fact, once this platform is ready for public use, they

will be the ones who will control the platform's contents in their entirety. Because of this and because of their additional knowledge on the subject of stained glass, it was thought best to create an additional questionnaire to the one the non-specialists users had to fill, in which we ask more technical questions regarding the information that is presented and stored in VitralWiki. In this way, we added three additional questions, which are as follows:

- **Level of relevance of VitralWiki's stained glass pages' content (Properties of the pages)**
  The user has to give a rating of 1 to 5, where 1 is 'Completely irrelevant' and 5 is 'Completely relevant', when it comes to the level of relevance of each property that a stained glass page can have.

- **Level of completeness of VitralWiki's stained glass pages' content (Properties of the pages)**
  The user has to give a rating of 1 to 5, where 1 is 'Lacking' and 5 is 'Complete', regarding the level of completeness of a stained glass page when it comes to what properties a stained glass page can have.

- **Comments and suggestions regarding VitralWiki (e.g., regarding the presented information - completeness, relevance, and form of presentation or more detailed observations on one of the previous questions)**
  Being an open question, the user can give their opinions regarding the previous question, so that it is possible to improve upon what has already been developed.

After each specialist user filled out this form, the results were as follows:

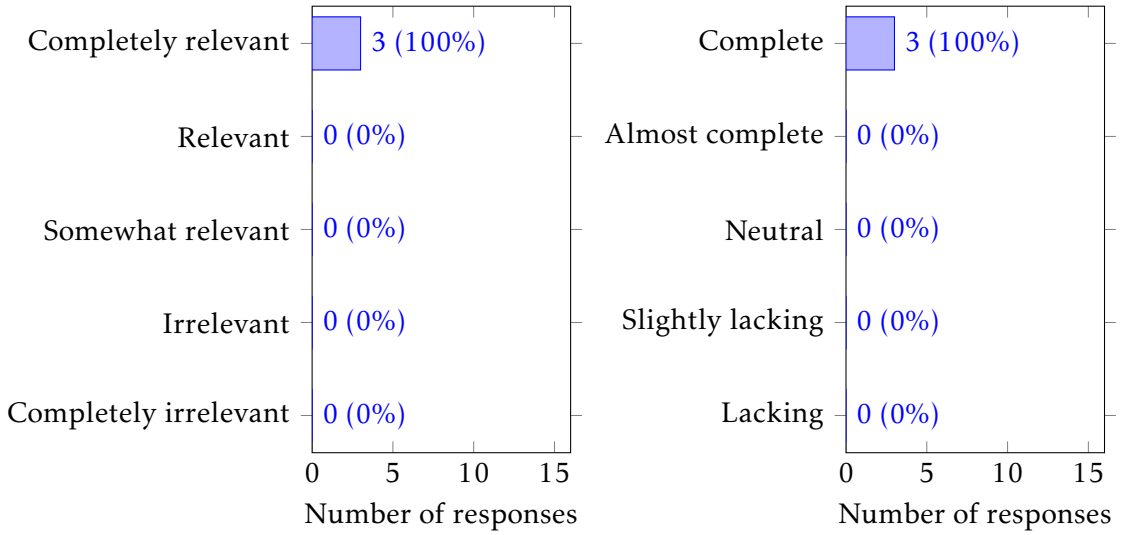


Figure 4.8: Results in the level of relevance of VitralWiki's stained glass pages' content and results in the level of completeness of VitralWiki's stained glass pages' content, respectively

From the specialists' form results seen in figure 4.8, even though we have a very small group of users belonging to this group, we can assume that each stained glass page's content is complete and relevant to those who are knowledgeable in said subject. This is also due to the fact that creation of each property was done with the help of VICARTE, as their knowledge regarding stained glass was essential on making a concise data model for each stained glass page in VitralWiki. Regarding the open text question answers, these are all regarding the overall aesthetics of the stained glass pages. As a single template is being used to display all stained glass pages, there were comments suggesting a different way to align the images that are in each page, which could be changed and tested in the template, since our solution is made in a way that easily allows to make changes to the overall aesthetics of the stained glass pages by only changing the template's code.

5

Conclusion

In this chapter, we will look into the final results of this thesis and analyze how it can be possible to improve and develop further upon what was done.

5.1 Final results

In this thesis, we developed a platform that uses the MediaWiki framework, where users that are specialists in stained glass can create and edit stained glass pages, and their information is stored in a structure that allows and retains the principles of Linked Open Data and integrates into the Semantic Web. We have also succeeded in creating a process for specialists create and edit content without prior knowledge of how to program and work inside MediaWiki or of the principles behind linked open data and the semantic web. Along with this, it is implemented to promote scalability, as it can be easily worked upon without compromising the integrity of the data stored within the platform or the platform itself. We have also managed to create a functioning Semantic Search page within VitralWiki, which lets users make more focused and efficient searches of stained glass content on the platform.

Through the platform that resulted from this dissertation, it is also possible to access stained glass data from other endpoints, given a correct configuration. As a showcase, we accessed stained glass-related content from the Europeana SPARQL endpoint.

5.2 Future plans

There are some aspects of this platform that could be improved or even added, given more time and work on it. Firstly, there were some problems regarding the server in which we tested VitralWiki that stopped us from making additional tests in the platform. One additional test that could have been done to test further our platform regarding its implementation in the Semantic Web, would be creating a SPARQL endpoint, similar to the one we accessed to get data from Europeana. This was not possible during the development and testing of VitralWiki, due to the necessity of changing database systems

to one that stores data in an RDF database, such as Virtuoso. Since we were using MySQL, data was not originally stored in an RDF store and is converted within VitralWiki. In the phase of the tests, it was not possible to change due to the server's permissions and settings regarding its ports, which could not be changed to maintain the server's security.

As seen from the results in section 4.2.2, there were two main aspects which the users found to be weaker in the platform, those being the Semantic search page and the overall aesthetics of the page regarding each stained glass page layout. Taking into account the open text question suggestions referred in that section, it would be possible, given more time, to fix the layout of each stained glass page by changing the CSS of the stained glass pages' template, as it will then also change all previously created stained glass pages to the preferred layout. Regarding the Semantic search, it was suggested to create a dropdown option with all existing values of each property for every field in the Semantic search form so that searching would be easier for both specialist and non-specialist users. Although it might be a difficult solution to implement, in the future it might be useful to analyse this solution with greater detail.

Another aspect that could have added more depth to our platform is the ability to view data both from VitralWiki and Europeana at the same time, instead of showing it on separate pages. There is an extension that could help in achieving this solution called *External Data* which, as the name suggests, allows one to show data from external endpoints. Through this extension, it would have been possible to retrieve data from Europeana's API, as opposed to its SPARQL endpoint, and show it in a selection of formats along with VitralWiki's data. It was possible to test this locally, but it was not possible to do this on the server, once again due to its permissions and problems regarding its ports. However, given more time, this could have been changed given more time, giving us a way to solve both of the previous solutions that could improve this thesis.

Finally, another tool that would have improved our platform would be the ability to import RDFs and convert them to a new stained glass page in MediaWiki. There are tools that can make this possible, but they were not compatible with the version of MediaWiki that was used in the final result of this thesis, which require additional work to adapt.

It is important to mention that this thesis and the platform that resulted from it will also serve as a foundation for a new dissertation that will occur in the semester that will follow the delivery of this document, in which many of the ideas mentioned throughout this section and more can be implemented and perfected.

Bibliography

[1] Grigoris Antoniou, Paul Groth, Frank van Harmelen, and Rinke Hoekstra. *A Semantic Web Primer, 3rd Edition*. MIT Press, 2012 (cit. on pp. 5, 6, 8).

[2] Dan Brickley and R.V. Guha, eds. *RDF Schema 1.1*. Available at https://www.w3.org/TR/rdf-schema/. W3C Working Group Note 25 February 2014, 2014 (cit. on p. 6).

[3] John Brooke. "SUS: A quick and dirty usability scale". In: *Usability Eval. Ind.* 189 (Nov. 1995) (cit. on p. 59).

[4] Jonathan Campbell and Michael Shin. *Essentials of Geographic Information Systems*. Saylor Foundation, 2011. URL: https://open.umn.edu/opentextbooks/textbooks/67 (cit. on p. 13).

[5] MediaWiki Contributors, ed. *Help:Images*. [Online; accessed 30-March-2023]. 2023. URL: https://www.mediawiki.org/wiki/Help:Images (cit. on p. 43).

[6] MediaWiki Contributors. *Manual:User rights*. [Online; accessed 30-March-2023]. 2023. URL: https://www.mediawiki.org/wiki/Manual:User_rights (cit. on p. 49).

[7] Wikipedia Contributors. *Help:Wikitext*. [Online; accessed 30-March-2023]. 2023. URL: https://en.wikipedia.org/wiki/Help:Wikitext (cit. on p. 41).

[8] Michael Dorman. *Introduction to Web Mapping*. [Online; accessed 30-March-2023]. 2021. URL: http://132.72.155.230:3838/js/index.html (cit. on p. 13).

[9] Bob DuCharme. *Learning SPARQL*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2013 (cit. on p. 12).

[10] Europeana. *Definition of the Europeana Data Model v5.2.8*. 2017. URL: https://pro.europeana.eu/files/Europeana_Professional/Share_your_data/Technical_requirements/EDM_Documentation//EDM_Definition_v5.2.8_102017.pdf (cit. on p. 51).

[11] "OntoWiki - An authoring, publication and visualization interface for the Data Web". In: *Semantic Web* 6.3 (2015). Ed. by Philipp Frischmuth et al., pp. 215–240 (cit. on pp. 14, 15).

[12] OWL Working Group, ed. *Web Ontology Language (OWL)*. Available at https://www.w3.org/OWL/. W3C Working Group Note 11 December 2012, 2014 (cit. on p. 6).

[13] W3C Working Group, ed. *Benefits of the Linked Data Approach*. Available at https://www.w3.org/2005/Incubator/lld/wiki/Benefits. W3C Working Group Note 20 September 2011, 2014 (cit. on p. 7).

[14] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011 (cit. on p. 11).

[15] *HISTORY OF GLASS: EARLY PERIODS*. 2019. URL: https://dmglass.com/the-history-of-glass-art-early-periods/ (cit. on p. 1).

[16] MediaWiki. *Manual:MediaWiki architecture — MediaWiki*. [Online; accessed 30-March-2023]. 2023. URL: https://www.mediawiki.org/w/index.php?title=Manual:MediaWiki_architecture&oldid=5852142 (cit. on p. 24).

[17] Ikrom Nishanbaev, Erik Champion, and David A. McMeekin. "A Web GIS-Based Integration of 3D Digital Models with Linked Open Data for Cultural Heritage Exploration". In: *ISPRS Int. J. Geo Inf.* 10.10 (2021), p. 684 (cit. on p. 13).

[18] Guus Schreiber and Yves Raimond, eds. *RDF 1.1 Primer*. Available at https://www.w3.org/TR/rdf11-primer/. W3C Working Group Note 24 June 2014, 2014 (cit. on p. 11).

[19] semantic-mediawiki.org. *Help:Parser functions — semantic-mediawiki.org*. [Online; accessed 30-March-2023]. 2015. URL: https://www.semantic-mediawiki.org/w/index.php?title=Help:Parser_functions&oldid=37942 (cit. on p. 41).

[20] semantic-mediawiki.org. *Help:Semantic search — semantic-mediawiki.org*. [Online; accessed 30-March-2023]. 2018. URL: https://www.semantic-mediawiki.org/w/index.php?title=Help:Semantic_search&oldid=66057 (cit. on p. 47).

[21] semantic-mediawiki.org. *Help:Using SPARQL and RDF stores — semantic-mediawiki.org*. [Online; accessed 30-March-2023]. 2018. URL: https://www.semantic-mediawiki.org/w/index.php?title=Help:Using_SPARQL_and_RDF_stores&oldid=63917 (cit. on p. 12).

[22] semantic-mediawiki.org. *Semantic MediaWiki — semantic-mediawiki.org*. [Online; accessed 30-March-2023]. 2020. URL: https://www.semantic-mediawiki.org/w/index.php?title=Semantic_MediaWiki&oldid=76616 (cit. on p. 25).

[23] Rajiv Shah. *What is Metadata and Types of Metadata in Semantic Web*. [Online; accessed 30-March-2023]. 2023. URL: https://benchpartner.com/what-is-metadata-and-types-of-metadata-in-semantic-web (cit. on p. 6).

[24] Wikipedia contributors. *Semantic wiki — Wikipedia, The Free Encyclopedia*. [Online; accessed 30-March-2023]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Semantic_wiki&oldid=1099057212 (cit. on p. 13).

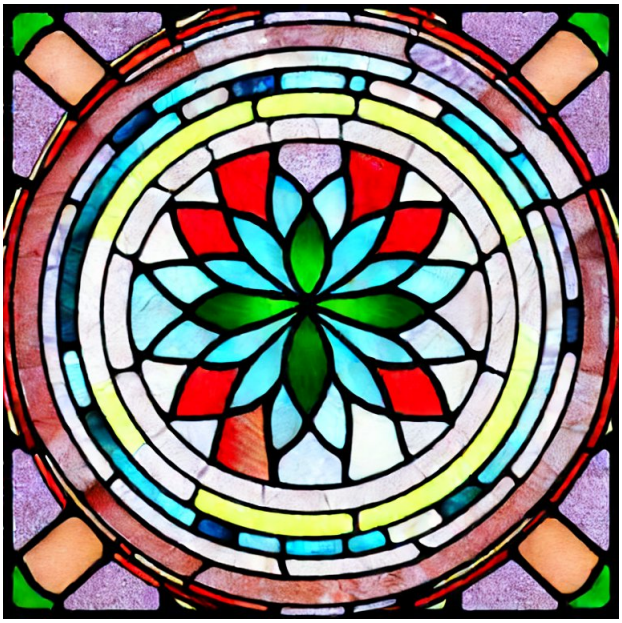
A

Appendix: Basics Manual on How to use VitralWiki

This Appendix shows the user manual for VitralWiki.

Basics Manual on How to use VitralWiki

José Oliveira



A.1 Getting started

A.1.1 Creating an account

In order to be able to fully use VitralWiki, initially you will have to create an account in here, where you will need to fill the obligatory fields presented in the page, as can be seen in image A.1.

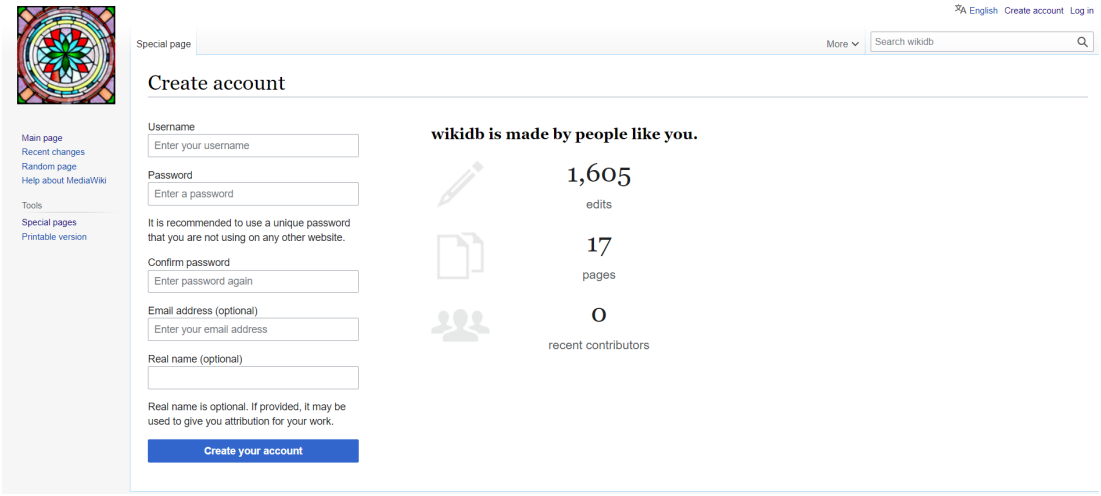


Figure A.1: Page where it is possible to create an account.

A.1.2 Logging into your account

If you are a member from Vicarte, you must inform VitralWiki's developer via e-mail with your account's name. Afterwards, you will be able to login into your account and access VitralWiki as a registered user through this page, where you will only need to fill the fields for your user name and your previously created password as seen in A.2

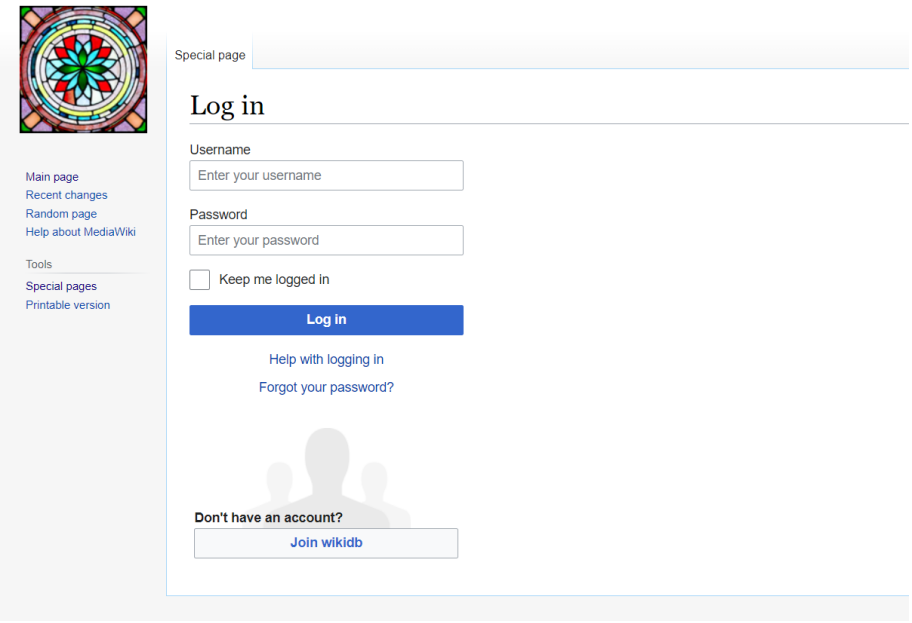


Figure A.2: Page where it is possible to login into an account.

A.2 Using VitralWiki

There are a couple of things you must know before using VitralWiki, especially if you are an administrator. After logging into your account, you should be presented with VitralWiki's main page where you can select from a variety of options. In this section, we will look deeper into each of these options.

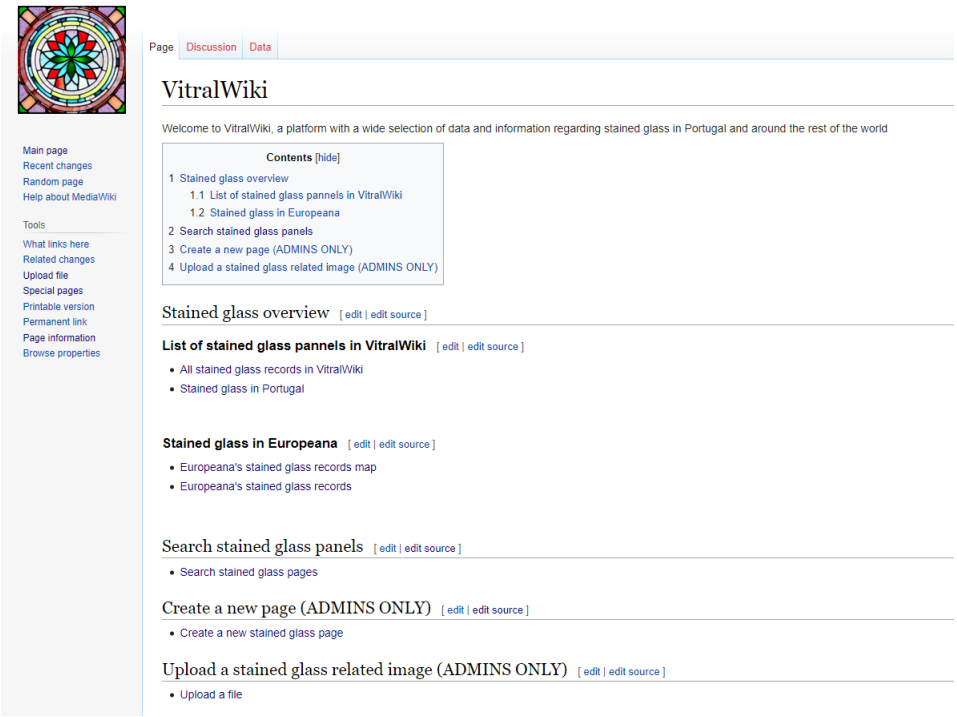


Figure A.3: VitralWiki's Main page.

A.2.1 List of stained glass pieces

In this section, you will be able to select one of two options: All stained glass records in VitralWiki and Stained glass in Portugal. As their names suggest, both of these pages will send you to a list of stained glass pages, where the first one will be a list of stained glass pages in VitralWiki and the second will also be a list of stained glass pages in VitralWiki, but they are also located in Portugal. In both of these pages, there are clickable links with the names of each stained glass page that, when clicked, send you to its page.

A.2.1.1 Stained glass pages

In this type of page, you will be able to see all information regarding a specific stained glass pannel stored in VitralWiki. Every stained glass page will have 6 different tables regarding different aspects of the stained glass pannel: General aspects, Localization, Photography/image, Panel, Owner and Iconongraphy. Each of these tables has different properties with or without values, where you can further details regarding the stained

glass piece. Depending on how much information the stained glass page has, there will probably also 4 different sections other than the tables with properties, them being a section with the description of the pannel, a section with images/photos and the descriptions of these of the pannel, a section with the location of the pannel in a map and a section with the laboratory PDF A.4. Notice that the map might also hold an image of the location's plan, in case you click the marker on said map.

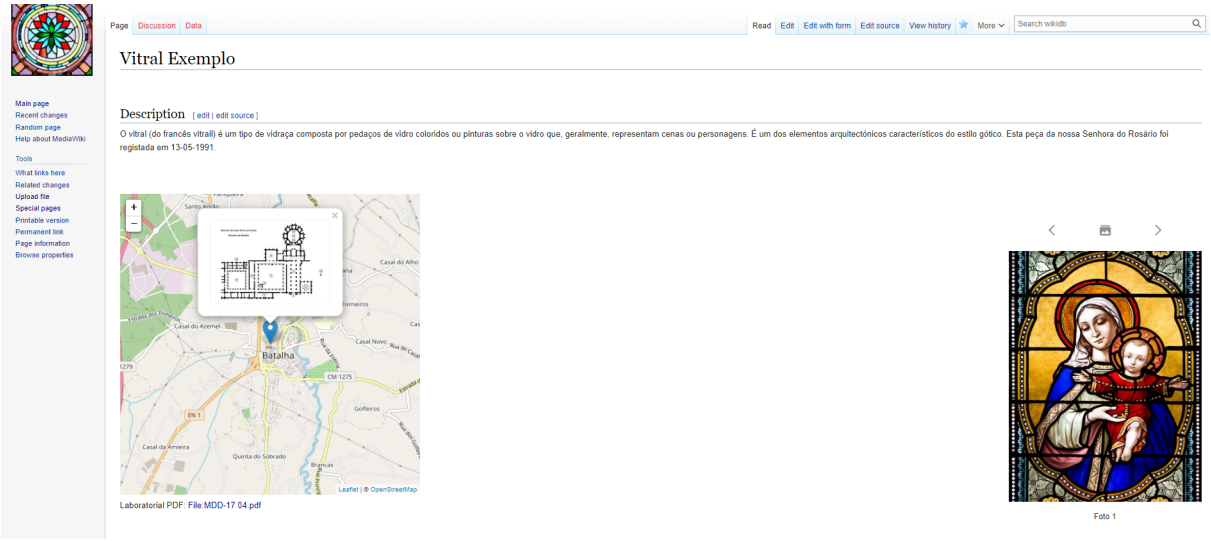


Figure A.4: Section of a sample stained glass page.

A.2.2 Stained glass in Europeana

In this section, you will be able to analyse stained glass pannels from accross Europe, stored in Europeana's repository.

A.2.2.1 Europeana's stained glass records map

As the name suggests, here you will be able to see an interactive map with Europeana's stained glass records, where each marker is clickable and holds information to the stained glass pannel it belongs too.

A.2.2.2 Europeana's stained glass records table

In this subsection, you can see a data table with Europeana's stained glass records where they are listed my the properties: Data Provider, Title, Description, Image, Date (date of revelance associated with the piece) and Provider. It is also possible to search by property in the upper right corner of the page A.5.

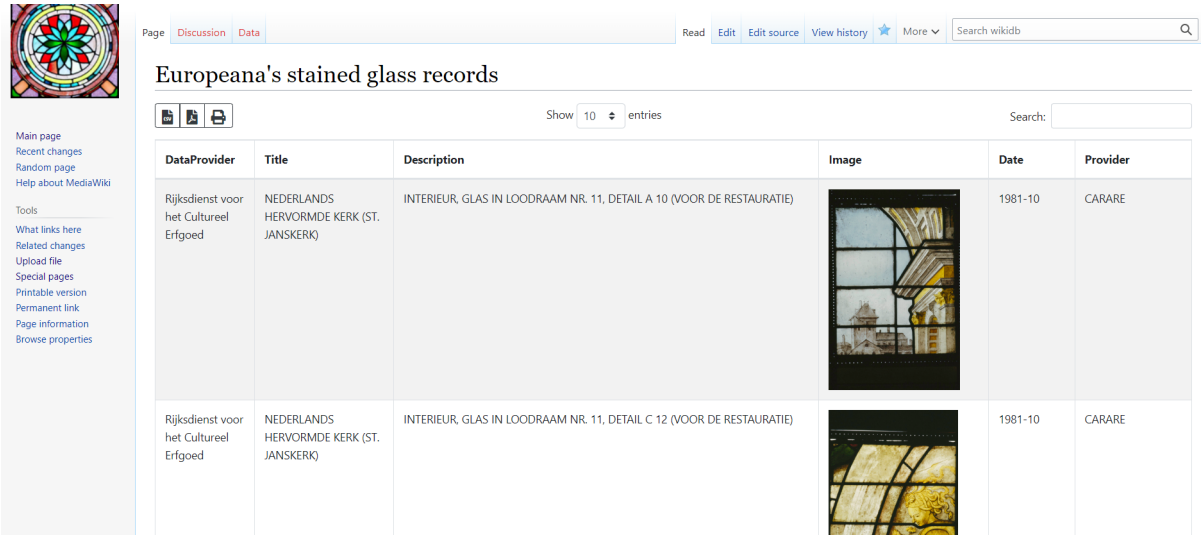


Figure A.5: Europeana's stained glass records table page.

A.2.3 Search stained glass panels

In the Search stained glass pages' page, you can search stained glass pages by filling the page's form and pressing the "Run Query"button. After this, the result is a list of stained glass page links which have the same values in its properties as the values that were filled in the search form. In the form, among with the properties there is a checkbox regarding the type of search that is going to be made. When the checkbox is not checked, the pages returned from the search only need to have at least one property with equal value to the one filled in the form. If "Intersection"checkbox is checked, all pages returned have to have exactly the same values in its properties as the ones that were filled in the search form.

To be efficient at searching, it is good to know the names of the properties that a stained glass page can have and how they are saved within the platform, which are as follows:

Name of the property	Description	Data type	Examples
Title (dc:title)	Title of the stained glass piece	Text	The meeting of St. Brendan with the Unhappy Judas
Description (dc:description)	Description of the stained glass piece	Text	This stained glass panel was created in ... by ...
Type (dc:type)	Type of glass art the piece belongs to	Text	Stained Glass
Dataset name (edm:datasetName)	Collection to which the glass art piece belongs	Text	08901_Ag_DE_DISMARC
Identifier in collection (dc:identifier)	Univocal identifier of the piece within its collection	Text	B123
Registry Author	Author of the registry of the art piece in the platform	Text	John Doe
Registry Date	Date of the registry of the art piece in the platform	Date	19/05/1943
Creator (dc:creator)	The creator of the piece	Text	Eugène Dunand
Cartoon designer	The cartoon designer of the panel	Text	Mário Costa
Workshop	The manufacturer of the piece	Text	Ricardo Leone
Place of manufacture	Location where the piece was created	Text	Batalha

Table A.1: General aspects related properties.

Name of the property	Description	Data type	Examples
Country (edm:country)	Country of origin of the piece	Text	Portugal
State/Region	State/region of origin of the piece	Text	Beira Alta
City	City of origin of the piece	Text	Batalha
Building	Building where the piece is located	Text	Mosteiro de Santa Maria da Vitória
Location details	Details regarding the piece's location	Text	In Storage
Current location (edm:currentLocation)	Current geographic location of the art piece	Geographic coordinates	73.18908,-87.34030
Geographic identifier (edm:Place)	URL the identifies the geographic location where the piece is located	URL	`http://www.geonames.org/8010544`
Direction	Direction of the panel, according with the CVMA notation	Text	N; S; 'North Gallery'
Window number	Number of the window of the glass art piece	Text	I;II;IV
Row	Row number where the piece is located	Number	1; 2
Column	Column number where the piece is located	Number	1; 2
Former/Original Locations	Former/original locations where the piece was located	Text	Frankfurt (Oder), formerly main church St. Marien, east choir, north II, 1c; Anger Museum Erfurt; Thuringia Museum Eisenach
Identifiers of original/former locations	List of URLs of the original/former locations where the piece was located	URL	`http://sws.geonames.org/11153426`

Table A.2: Location related properties.

Name of the property	Description	Data type	Examples
Images filenames	Filename(s) of the image(s) of the stained glass piece presented on the page	Text	Vitral.jpg, Vitral2.png
Images descriptions	Description(s) of the image(s) presented on the page	Text	Description of Vitral;Description of another stained glass panel
Image filename of plan	Filepath URL of the plan where the stained glass piece is located	URL	`https://placeholder./Planta.png`
Image filename of elevation	Filepath URL of the elevation where the stained glass piece is located	Text	`https://placeholder./Elevation.png`
Image URL (edm:object)	URL correspondent to the image of the art piece (If nothing is inputted, it will assume the filepath URL to the *Images filenames property*)	URL	`https://api.europeana.eu/thumbnail/v3/400/71155201ce47eee31245ad7232fbdb2d`
Photographic process	Photographic process behind the piece's photo	Text	Transmitted light front single shot
Digital image source	Origin of the digital image of the piece	Text	Original digital photo
Photographic context	Context of the circumstances where the picture was taken	Text	in situ;expanded
Photograph creation date	Date of when the photo was taken	Date	1984; 1984-05; 1984-05-22
Image author	Author of the piece's picture	Text	Renate Roloff
Holder of rights (dc:rights)	Holder of the piece's photo rights	Text	CVMA Portugal
Publisher (dc:publisher)	Publisher of the piece's photo	Text	CVMA Freiburg - `http://www.corpusvitrearum.de/`, CVMA Potsdam - `http://www.corpusvitrearum.de/`
Credits	Credits of the piece's photo	Text	CVMA Germany Potsdam / Berlin-Brandenburg Academy of Sciences, Photo: Holger Kupfer
License type	Specification of the license on which the image can be used	Text	CC BY-NC 4.0
License description	URL the points towards a description of said license	URL	`https://creativecommons.org/licenses/by-nc/4.0/`

Table A.3: Photography/image related properties.

Name of the property	Description	Data type	Examples
Measurements (dcterms:extent)	The measurements of the glass art piece	Text	23cm; 50cm
Date of origin	Date or period when the glass art piece was created	Text	'During 1430'; XVth century
Date of origin(standard - beginning)	Beginning of the period of origin in the standard format	Date	1430-01-1
Date of origin(standard - end)	End of the period of origin in the standard format	Date	1430-12-31
Production technique	Technique behind the production of the glass art piece	Text	Grisaille and silver stain painting on colorless and bulk-coloured glass
Laboratorial PDF	Filepath URL of the laboratory analysis of the glass art piece	Text	`https://placeholder./analysis.pdf`
Laboratorial PDF URL	URL of the laboratory analysis of the glass art piece (If there is nothing is inputted, it will assume the same value as *Laboratorial PDF*)	URL	`https://placeholder.com/analysis.pdf`
Current state of conservation	Text description of the current state of the piece	Text	The window was taken down again in 2006 for cleaning and reinforcement of lead joints.
Restoration and conservation history	Filepath URL of the text file with the history of restoration and conservation interventions to the piece	URL	`https://placeholder./history.pdf`
History related image	Filepath URL of the history related image regarding the piece	URL	`https://placeholder./history.png`
Bibliography/source (dc:source)	Bibliography/source historically associated with the art piece	Text	Lionel Breitmeyer, Saint-Maurice, église catholique romaine de Bernex: étude historique, Genève, 1991

Table A.4: Panel related properties.

Name of the property	Description	Data type	Examples
Owner's name	Name of the owner of the piece	Text	Corpus Vitrearum
Owner's identifier	Function of the owner in relation to the object	Text	Conservator

Table A.5: Owner-related properties.

Name of the property	Description	Data type	Examples
Iconclass notation	Alphanumeric code of the iconclass classification that describes the iconography of the object	Text	73B57
Iconclass keywords	Keywords related to the piece's iconclass	Text	The Virgin with the Child stands on a crescent and holds a white rose in her right hand while Jesus shows a rosary
Signature	Existence of a signature of the author(s)	Bool	True

Table A.6: Iconography-related properties.

A.3 Using VitralWiki as an Administrator

A.3.1 Uploading files

If you are a VitralWiki administrator, you will probably want to eventually create a new stained glass page and one thing you might want to add onto that page is one or more images of the stained glass piece you want to make a page about. This process is quite simple, as you will only need to click in the "Special:Upload" section in the main page. From there, you will simply need to choose the file from your desktop that you want to upload, add a summary adding context to the image you are uploading and finally clicking the "Upload File"button, as can be seen in A.6

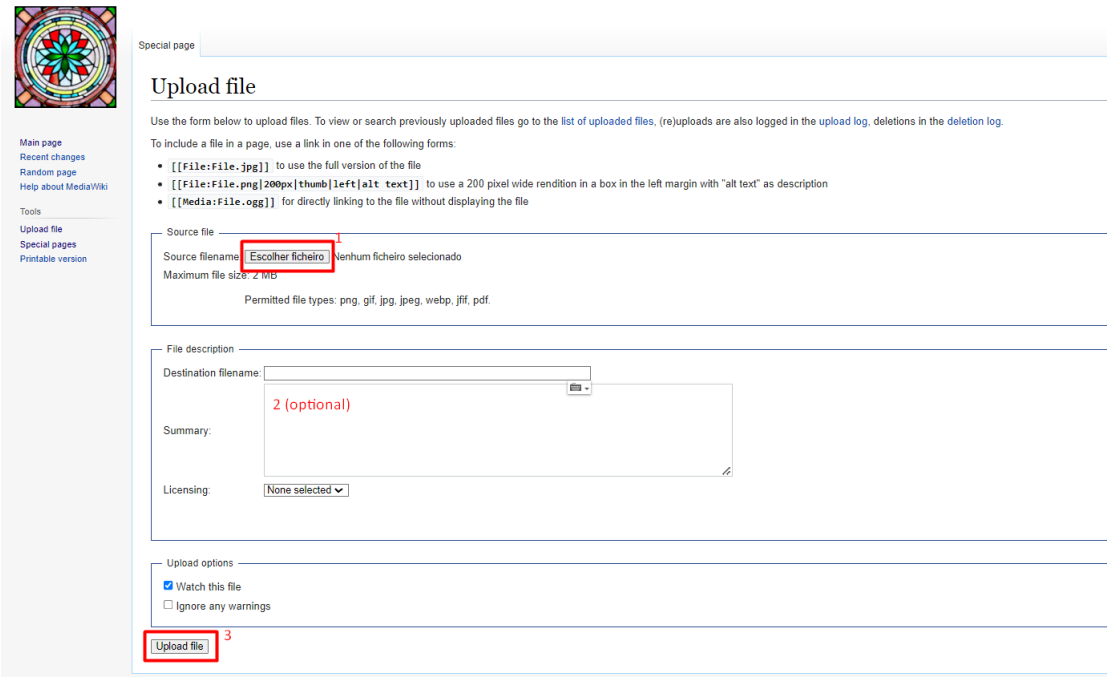


Figure A.6: VitralWiki's page to upload files. 1- Button that leads into choosing the file you want to upload. 2- Text area where you can add context to the file. 3- Button to finish uploading the file

Along with this way of uploading files, it is also possible to upload multiple files at once through the "Upload multiple files" link, where one only needs to drag the files directly into the "Select files or drop them here"button or click and select the files that are going to be uploaded. Then, one only needs to wait until the file's upload is at 100% and afterwards it is possible to proceed to create a new page. Notice: The name of the file will be important for later on.

A.3.2 Creating a new stained glass page

Being an administrator in VitralWiki, you have the ability of adding new stained glass pages to the platform. To do this, you simply have to click the "Create a new glass art

page" section in order to start creating a new page. After deciding on a title for the page and submitting it A.7, you will be fronted with a form.

Figure A.7: VitralWiki's start form page

This form contains each property of the glass art page that you are about to create. Note that none of the properties need to have values. The format of the form is the same as the previously mentioned tables in A.2.3, so do take a look at the tables before creating a new page.

As previously mentioned, the filenames of the images you want to add on the page are important to present said images on the page by filling them in the "Images filenames"and "Image filename of plan"properties. In case you forget the filenames of the images, you can also check the New files page and search for your images there. Even so, the field for the "Image filenames"property will autocomplete based on the already existing files in the platform. Another important thing to notice, in case you want to add multiple descriptions for your images, in the property "Images descriptions"you need to fill in the format "Description 1;Description 2", separating the descriptions with a semi-colon (without a spaces separating the semi-colon).

Note: If no value is attributed to the properties "Image URL"and "Laboratorial PDF URL"but if there are values in the properties "Images filenames"and "Laboratorial PDF", the value will be automatically attributed to these properties after the creation of a page.

A.3.3 Editing and deleting existing pages

To edit an existing page, either to add new values to properties or to change existing values, you simply have to click "Edit with form"in the upper right corner of the page. In case you want to delete the page, you have to press the "More"button also in the upper right corner of the page, and then press the button "Delete"on the dropdown A.8.

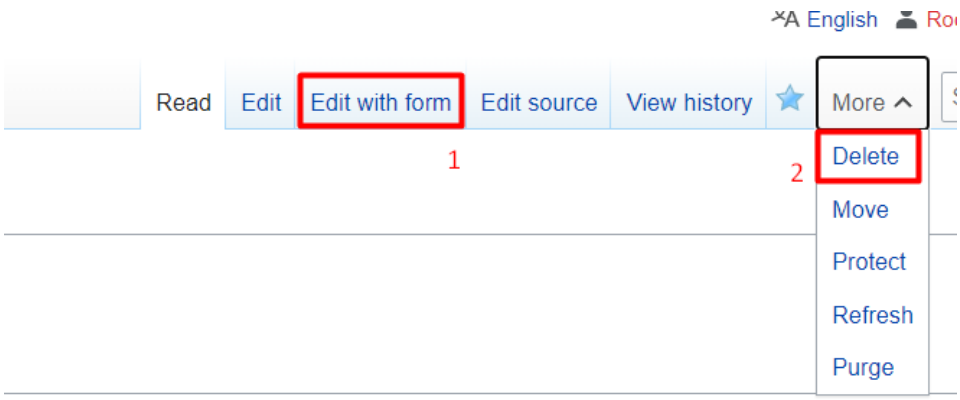


Figure A.8: Section to edit or delete a stained glass page. 1- Button to edit with a form. 2- Button to delete the page

Also notice that you can edit a page by returning to the "Create a new glass art page" and entering the name of the existing page you want to edit and procede.

B

Appendix: Server specs

This appendix contains the specs of the server that was used to test VitralWiki.

```
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   43 bits physical, 48 bits virtual
CPU(s):                          2
On-line CPU(s) list:             0,1
Thread(s) per core:              1
Core(s) per socket:              1
Socket(s):                       2
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           62
Model name:                      Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz
Stepping:                        4
CPU MHz:                         2194.711
BogoMIPS:                        4389.42
Hypervisor vendor:               VMware
Virtualization type:             full
L1d cache:                       64 KiB
L1i cache:                       64 KiB
L2 cache:                        512 KiB
L3 cache:                        50 MiB
NUMA node0 CPU(s):               0,1
Vulnerability Itlb multihit:     KVM: Mitigation: VMX unsupported
Vulnerability L1tf:              Mitigation; PTE Inversion
Vulnerability Mds:               Mitigation; Clear CPU buffers; SMT Host state u
                                 nknown
Vulnerability Meltdown:          Mitigation; PTI
Vulnerability Mmio stale data:   Unknown: No mitigations
Vulnerability Retbleed:          Mitigation; IBRS
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled v
                                 ia prctl and seccomp
Vulnerability Spectre v1:        Mitigation; usercopy/swapgs barriers and __user
                                  pointer sanitization
Vulnerability Spectre v2:        Mitigation; IBRS, IBPB conditional, RSB filling
                                 , PBRSB-eIBRS Not affected
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Not affected
Flags:                           fpu vme de pse tsc msr pae mce cx8 apic sep mtr
                                 r pge mca cmov pat pse36 clflush dts mmx fxsr s
                                 se sse2 ss syscall nx rdtscp lm constant_tsc ar
                                 ch_perfmon pebs bts nopl xtopology tsc_reliable
                                  nonstop_tsc cpuid pni pclmulqdq ssse3 cx16 pci
                                 d sse4_1 sse4_2 x2apic popcnt tsc_deadline_time
                                 r aes xsave avx f16c rdrand hypervisor lahf_lm
                                 cpuid_fault pti ssbd ibrs ibpb stibp fsgsbase t
                                 sc_adjust smep arat md_clear flush_l1d arch_cap
                                 abilities
```

Figure B.1: Specs of the server's CPU