FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Implementation of a Portal Dedicated to Higgs Bosons for Experts and the General Public
Student:	Peter Žáčik
Supervisor:	doc. Dr. André Sopczak
Study program:	Informatics
Branch / specialization:	Web and Software Engineering, specialization Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2021/2022

Instructions

After the very successful feasibility study conducted as a Bachelor thesis, the task of this project is to implement a Higgs Boson Portal (HBP). The HBP will be useful to a large number of Higgs boson researchers at CERN and other high-energy physics institutes, phenomenologists and theorists. The HBP will also bring the discovery and the current research in the field of Higgs bosons closer to the general public. This project offers the collaboration in a worldwide renowned international organization and a visit of CERN and the Large Hadron Collider where the Higgs Boson was discovered.Tasks:
1) Familiarize yourself with the results of the feasibility study.
2) Implement the suggested framework.
3) Develop an automated update system for new publications.
4) Implement a categorization system for the publications.
5) Implement a statistical analysis of the publications.
6) Test the HBP and refine the look and feel.

Bonus:
Visualize the development of the measurement precisions.

Electronically approved by Ing. Michal Valenta, Ph.D. on 16 October 2020 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Implementation of a Portal Dedicated to Higgs Bosons for Experts and the General Public

Peter Žáčik

Department of Software Engineering
Supervisor: doc. Dr. André Sopczak

May 13, 2021

Acknowledgements

I would like to thank my supervisor, doc. Dr. André Sopczak, for all the time and resources he dedicated to me and this project.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 13, 2021 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Žáčik, Peter. *Implementation of a Portal Dedicated to Higgs Bosons for Experts and the General Public*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Tato práce prezentuje implementaci webového portálu věnovaného výzkumu Higgsových bozonů. Pomocí API dokumentového serveru CERN-u a metod web scrapingu je vytvořena souhrnná databáze více než 1000 relevantních článků. Databáze je automaticky aktualizována, když jsou dostupné nové výsledky ve výzkumu Higgsova bozonu. S využitím zpracování přirozeného jazyka jsou články automaticky kategorizovány podle vlastností Higgsova bozonu a dalších kritérií. Proces návrhu a implementace portálu „Higgs Boson Portal" (HBP) je v práci detailně popsán. Komponenty HBP jsou nasazeny na CERN Web Services za použití cloudové platformy OpenShift. Webový portál je funkční a dostupný na adrese `https://cern.ch/higgs`.

Klíčová slova Higgsův bozon, CERN, webová aplikace, implementace, automatizace, kategorizace, statistická analýza, React, Flask, MongoDB

Abstract

This thesis presents the implementation of a web portal dedicated to Higgs boson research. A database is created with more than 1000 relevant articles, using CERN Document Server API and web scraping methods. The database is automatically updated when new results on the Higgs boson become available. Using natural language processing, the articles are categorised according to properties of the Higgs boson and other criteria. The process of designing and implementing the Higgs Boson Portal (HBP) is described in detail. The components of the HBP are deployed to CERN Web Services using the OpenShift cloud platform. The web portal is operational and freely accessible on `https://cern.ch/higgs`.

Keywords Higgs boson, CERN, web application, implementation, automatisation, categorization, statistical analysis, React, Flask, MongoDB, OpenShift

Contents

List of Figures

List of Tables

Introduction

Following the proposal of the existence of a new elementary particle in 1964 and a subsequent worldwide search, the Higgs boson was first observed in 2012 by the ATLAS and CMS collaborations using data from the Large Hadron Collider (LHC). The Higgs boson is a part of the Standard Model of particle physics, a theory that describes all known elementary particles and three of the four fundamental interactions. The Standard Model does not explain all phenomena of the universe. It is used as a basis and a gateway to more diverse theories.

Researchers studying the Higgs boson have already produced hundreds of scientific articles, and new discoveries are still being made. Several collaborations participated in the research, two of which are still active today, the ATLAS and CMS experiments. As a result of the many searches for the Higgs boson and a massive technology advancement in the recent years, the information about the particle is scattered.

For scientists, it can be difficult to find all the relevant articles for their specific Higgs boson research, as each experiment has a different way of presenting their results. There is no simple way for staying up to date with the latest news in the field. And nine years after the discovery, the public is still largely unfamiliar with the research being conducted at these facilities. A complete portal dedicated to the Higgs boson could help physicists with their work, as well as spread more knowledge about the Higgs boson to the general public.

This thesis follows on a feasibility study conducted as bachelor's thesis by Martin Kupka: *Feasibility Study of a Portal to Provide Knowledge About the Higgs Boson to the General Public and Experts*. This thesis closely follows Kupka's suggestions and the structure for creating the Higgs Boson Portal (HBP).

The thesis consists of six chapters. The first chapter describes the goals and objectives of this thesis. In the second chapter, the physics behind the Higgs boson is introduced. Necessary parts are explained in more detail. In the third chapter, the feasibility study is inspected and current scientific resources for the Higgs boson research are listed and analysed. A solution to gather information from these sources is proposed. The fourth chapter describes the design and architecture of the Higgs Boson Portal, as well as tools necessary for implementation, testing and deployment. The fifth chapter focuses on technical challenges and the implementation of the portal. In the last chapter the test of the portal and a statistical analysis of the gathered information is presented.

Chapter 1

Goal

The goal of this thesis is to design, implement and test an online portal dedicated to the ongoing research about a discovered particle – the Higgs Boson. The Higgs Boson Portal will be useful to a large number of researchers at CERN, Fermilab and other high-energy physics institutes. Such a portal could also bring the latest news in the Higgs boson field closer to the public. The Higgs Boson Portal will automatically collect and unify all published papers and articles about the boson and present it to the user in an organized manner. Filtering and sorting by various categories will be possible, as well as linking related articles. A brief description of the history of the particle, ongoing experiments and future plans will also be presented to the public in a graphically appealing way.

Firstly, it is necessary to become familiar with the current situation of the Higgs boson research and the results of the feasibility study. The next step is to outline the architecture and design of the portal and decide which tools and services are suitable. The final task is the implementation and deployment of the portal.

Chapter 2

The Higgs Boson

The Higgs boson, named after the the physicist Peter Higgs, is a subatomic particle. It is the manifestation of the Higgs field, a quantum field that is present everywhere throughout the Universe. Certain particles interact with the Higgs field field via the Higgs mechanism and as a result obtain mass. The Higgs boson is the result of an excitation in this field [1, 2].

2.1 The Standard Model

The Standard Model of particle physics is the most successful theory in explaining the phenomena of the universe. It describes six quarks and six leptons. The quarks and three of the leptons couple to the Higgs field and obtain mass. The W and Z bosons, carriers of the electroweak interaction also couple to the Higgs field [3, 4].

Until 2012, the Higgs boson was the last missing piece of the Standard Model. However, the discovery alone did not conclude the research. The Standard model has been rigorously tested and verified but some challenges still remain unsolved. The theory of gravity does not fit into this model and neither does it explain dark matter. To expand our understanding it is necessary to do more research. Precision measurements of the Higgs boson properties could point in the right direction. Observing the behaviour and measuring various properties of the boson leads to new discoveries even today.

2.2 Beyond the Standard Model

There are many theories beyond the Standard Model as theorists believe that the Standard Model is incomplete. The Standard model only contains one Higgs boson. Other theories however predict Higgs bosons with different properties to exist as well, such as lighter, heavier, or charged Higgs bosons. None of these additional Higgs bosons have been discovered yet.

2.3 History of Searches

In 1964, in order to answer the question of why some particles have mass, a mechanism involving a new quantum field was proposed. Particles that would interact with this field would as a result acquire the property of mass. After the discovery of the W and Z bosons and the top quark, the Higgs boson was the last remaining piece of the Standard Model to be found. Observing it would prove the existence of the Higgs field.

This was first attempted at CERN, in the Large Electron Positron (LEP) collider by four collaborations: ALEPH, DELPHI, L3 and OPAL between the years 1989 and 2000. Scientists at LEP excluded a Higgs boson with a mass below 114 GeV[1] with a high confidence level. Theory does not predict the mass of the Higgs boson and therefore it must be measured experimentally.

The search was then extended at the Tevatron collider in the United States, by the CDF and DØ collaborations at Fermilab. The data from the Tevatron experimentally excluded the presence of the Higgs boson in the mass range 147-180 GeV [5].

Finally, in 2012, only three years after the completion of the Large Hadron Collider, a new particle with a mass of approximately 125 GeV was observed by both the ATLAS and CMS collaborations at CERN. A year later it was confirmed that this was the Higgs boson and Peter Higgs and François Englert received a Nobel prize.

2.4 Current Research

Today, the only active experiments today in the Higgs boson research are ATLAS and CMS. There are other experiments at CERN but they are not studying the Higgs boson, for example ALICE or LHCb. ATLAS and CMS study the boson's properties, but also actively look for "new physics", beyond the Standard Model.

When the Large Hadron Collider is active, it produces around 90 petabytes of collision data per year [6]. This data then needs to be filtered and processed. When doing research, only potentially interesting collision events are selected [6]. For example, one may only want to study a certain decay (section 2.6) or production mode (section 2.5) of the Higgs boson. Consequently, when a new article is published, it can be classified in different ways.

Different particle accelerators operate at different energies, also called centre-of-mass energies. The centre-of-mass energy describes the total energy of one collision [7]. It is an important parameter when analysing collision data, because it affects the likelihood of certain events to occur.

The number of collisions per unit time produced in an accelerator is given by the so-called luminosity of the accelerator. Integrating the delivered lumi-

[1] An electronvolt (eV) is a unit of energy/mass most commonly used in particle physics.

nosity over time gives the integrated luminosity. It is a measure of the total number of interactions in a detector [8]. The integrated luminosity is also an important parameter in analytical articles, effectively describing the size of the data set.

2.5 Production Modes

The Standard Model Higgs boson is produced in energetic particle collisions in multiple different ways – production modes. These are distinguished by the particles that fused together to form the Higgs boson or the particles that were produced along with the Higgs boson. Currently observed production modes are listed below, ordered from most common to least common [9].

- Gluon gluon fusion (ggF)
- Vector boson fusion (VBF)
- Vector boson associated production (WH+ZH)
- top-top Higgs production (ttH)

Other production processes might be possible but are not yet verified, such as Higgs boson pair production [10].

2.6 Decay Modes

The Higgs boson is a short-lived particle. It decays almost immediately to lighter particles, which are then analysed to reconstruct the original Higgs boson. Below is a list of observed decay modes, ordered from most probable to least probable [9].

- $H \to bb$ (two bottom quarks)
- $H \to WW^*$ (two W bosons)
- $H \to \tau\tau$ (two tau leptons)
- $H \to ZZ^*$ (two Z bosons)
- $H \to \gamma\gamma$ (two photons)
- $H \to \mu\mu$ (two muons)

Theories beyond the Standard Model suggest the existence of a Higgs boson with more exotic decay modes, but this is still a subject of research in particle physics.

Chapter 3

State-of-the-art

3.1 Scientific Resources

The current state of scientific resources for information about the Higgs boson has been well described in Martin Kupka's bachelor's thesis: *Feasibility Study of a Portal to Provide Knowledge About the Higgs Boson to the General Public and Experts* [11]. In his thesis, Kupka presented multiple options for experts to stay up-to-date with current research. The most prominent for CERN results are the systems hosted by the respective experiments, such as ATLAS [12] and CMS [13]. For the legacy experiments at CERN, the CERN Document Server (CDS) is deemed most useful [11]. Both Tevatron experiments, CDF [14] and DØ [15], host their own respective websites for publishing results.

It is important to make a distinction between the different stages an article can be in. First, it enters the "preliminary" stage, which means that it has not yet been verified by a journal. After submitting it to the journal, it enters the "submitted" phase. This means it has been accepted for review by the journal. When accepted, the journal publishes the article and it enters the "published" phase.

3.1.1 Journals and Preprints

Scientific journals are a good source of information for experts. The most commonly used by particle physicists is *Inspire* [11]. Inspire offers advanced searching options and various additional information about articles. It is dedicated to all high-energy physics, not only to the Higgs boson, and therefore lacks specific categorisation for Higgs boson articles. Preprints are typically first made available in `arXiv.org`.

3.1.2 ATLAS and CMS

The ATLAS and CMS collaborations each host a website for publishing their results from the Higgs boson research. These websites are continuously updated with new publications and preliminary results.

The CMS web portal provides categorisation and filtering for their articles by production mode, decay mode and centre-of-mass energy. However, these filters cannot be combined. The ATLAS web portal provides more categorisation options, such as centre-of-mass energy, minimal luminosity, type of analysis and the decay mode. These filters can be combined with "AND" logic and, in case of the decay mode, also "OR" logic.

Both web portals make a clear distinction between preliminary, submitted and published articles. They include the date of publication, journal reference and a hyperlink to the article's PDF file. They also include all figures and tables with the corresponding caption. The system available for ATLAS publications is generally more preferred by experts [11].

It is important to note that the CERN Document Server is also updated with new results and provides the same information as the ATLAS and CMS portals, but without the categorisation options. However, CDS exposes an API that can be used to gather the information automatically [16].

3.1.3 ALEPH, DELPHI, L3, OPAL

The four LEP experiments took data between 1989 and 2000 and published their final results in the subsequent years. Thus, no new articles are expected to be published. The optimal way to look through the information accumulated by these experiments is via the CDS. CDS provides the title, date of publication, abstract, list of authors and a list of files related to the publication.

3.1.4 CDF and DØ

The CDF and DØ experiments are not related to CERN. They are a part of the Fermi National Accelerator Laboratory (Fermilab), located near Chicago, in the United States. As such, the information is not available through the CDS. Both collaborations host their own website, where new results are presented. These websites are obsolete and provide little categorisation options [11]. The Tevatron accelerator was shut down in 2011 and no more articles related to the Higgs boson are expected to be published.

3.1.5 Development of Measurements

There are many technology improvements of the accelerators and detectors. More data is collected over the years and therefore measurements of various

properties of the Higgs boson improve over time. In search for the Higgs boson, the four LEP experiments continuously updated the lower mass limit of the Higgs boson, up to 114.4 GeV [17]. Experiments at Fermilab developed an upper mass limit of the Higgs boson, finally excluding a mass range between 147 and 180 GeV. [5] After the discovery, ATLAS and CMS continue to improve the precision of the mass measurement of the Higgs boson. The idea of visualising these developments is generally well received by the scientific community [11].

3.2 Outreach to the Public

The general public is mostly familiar with the discovery of the Higgs boson in 2012. The continuing research of the Higgs boson is not well known. One page on the CERN's official website is dedicated to the Higgs boson and some information is available in the Higgs boson's Wikipedia article. Using graphically appealing presentation, the Higgs Boson Portal could help bring the research closer to the general public [11].

Chapter 4

Design, Tools and Architecture

In this chapter, the requirements for the Higgs Boson Portal are listed, and respective solutions proposed. Required methods and tools are chosen and are briefly described. Figure 4.1 outlines the architecture of the HBP using the UML Component diagram.

4.1 Requirements

4.1.1 Functional requirements

The HBP must be able to automatically collect information from the web and update it in reasonable intervals. It must classify collected articles into various categories with high precision and display the classifying information. Articles that are superseeded by newer versions must be marked, and a link to the new version has to be provided.

The portal must be accessible via an adequate User Interface that enables filtering and searching through the collected articles. Users must be able to send feedback to the administrators of the portal.

Administration of the portal must be possible through the web browser. The administration tasks include:

- removing non-relevant articles,
- adjusting categorisation information,
- reviewing and responding to feedback,
- checking the automatic updates to the HBP,
- approving other administrators.

Several *nice-to-have* features are proposed. Users should have an option to save the URL of a specific article to come back to it later. References to

the articles from other sources should be included. Figures, plots and other graphical information related to an article could be displayed directly on the portal.

4.1.2 Non-functional requirements

The HBP has to be available on the Internet with little downtime. Loading times have to be kept as short as possible. The content of the web-portal must be well organised and consistent.

The source code must be organised in a modular manner and easily extensible. Good software development practices and design patterns must be applied.

4.2 Methodology

This section introduces tools and methods that are utilised to fulfill the above mentioned requirements.

4.2.1 Web Scraping

In order to update the HBP periodically, it is necessary to develop an automated update system. Such a system has to scan the web and then download, process and save relevant information. Some data can be collected using the CERN Document Server's API. In case of the Tevatron experiments, scraping (crawling) their websites is the only viable option [11].

"In theory, web scraping is the practice of gathering data through any means other than a program interacting with an API. (or, obviously, through a human using a web browser). This is most commonly accomplished by writing an automated program that queries a web server, requests data (usually in the form of HTML and other files that compose web pages), and then parses that data to extract needed information" [18].

To be effective in web scraping it is important to be familiar with the target website. The CDF and DØ websites are simple HTML pages without dynamically loaded content. All information can be parsed using an XML tree parser from a single HTML file.

Web-scraping can potentially harm the target website by sending many requests in short period of time, therefore it must be applied with caution. An excessive number of requests could overload the server and cause denial of service. Websites usually provide a `robots.txt` file in the root directory of the website, which contains a list of restrictions for search engines, as well as web scrapers. Although these rules are not an enforcement standard, not abiding by them might be illegal in some cases and they should be followed by an ethical scraper [19]. Both CDF and DØ collaborations include a `robots.txt` file, but the restrictions are not limiting to the activity of the HBP.

4.2.2 Natural Language Processing

The raw downloaded data is meaningless without processing. One goal of the HBP is to classify scientific articles into various categories. To the author or an informed reader this classifying information is obvious. Despite that, assigning categories is not a trivial task for the computer [20]. There are many approaches one can take to automatically process human written text.

The naive solution is to manually create rules that determine the categories. This method generally involves searching for keywords in a selected part of the text. Because of the variability of the natural human language, this method often lacks the needed precision [21].

More complex solutions involve artificial intelligence (AI) and Natural Language Processing (NLP). *"Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech"* [20]. NLP is a branch of Artificial Intelligence, which relies on deriving meaning from human languages.

4.2.2.1 Text Classification

To classify articles into predefined categories it is efficient to use a probabilistic classifier. A probabilistic classifier is a classifier that given some input will return the probability of that input being in all of the predefined classes. The class with the highest probability is chosen as the result. Classification is performed using a collection of labeled data, known as the training data. New examples are classified by selecting the class that is most likely to have generated the result. The simplest probabilistic classifier is the *naive Bayes classifier* [22]. The naive Bayes classifier assumes that all features (words) of the text are independent of each other. While this assumption is clearly wrong, the classifier still performs reasonably well compared to keyword based solutions [21].

A text can be expressed as a vector of words – features. This is commonly called the *bag of words* approach [23]. Before classification, it might be beneficial to pre-process the text by removing *stopwords*, which are the most frequently used words in the given language. English examples are "the", "and" or "from". These words do not play any role in classifying the article [24]. The remaining words undergo *stemming*, which involves reducing the words to their root form, e.g. "decaying" to "decay". Context of individual words is lost during the pre-processing. However, this method can still be used to classify the text as a whole.

Articles are often tens of pages long, therefore it is computationally ineffective to apply NLP methods to the full text of the article. In case of the HBP, only the title and the abstract of an article are selected for processing. This also eliminates the need for having access to the full text of the article.

Given a finite vector of features (words) $D = \langle d_i \rangle; i \in \mathbb{N}$ with a length of n and a set of classes $C = \{c_1, c_2, \ldots, c_k\}$, the classification is the assignment of a class $c \in C$ to the vector D, denoted as $c^*(D)$. The likelihood of some class $c \in C$ being assigned to D can be formulated as:

$$P(c \mid D). \tag{4.1}$$

The final result of the classification can be expressed with the following equation, given $1 \leq j \leq k$:

$$c^*(D) = \arg\max_j P(c_j \mid D). \tag{4.2}$$

Using Bayes' theorem, equation 4.1 can be rewritten as [25]:

$$P(c \mid D) = \frac{P(c)\, P(D \mid c)}{P(D)}, \tag{4.3}$$

where $P(c)$ is the likelihood of encountering class c. This can be estimated from the training data. $P(D)$ is the likelihood of encountering the vector D. $P(D)$ is impossible to calculate, but because its value does not depend on the class c, it can be ignored. $P(D \mid c)$ is the likelihood encountering the vector D, given that its assigned class is c. Assuming features $d_i, i \in \mathbb{N}$ are independent of each other (the naive assumption), $P(D \mid c)$ can be calculated as a product of likelihoods of individual features d_i appearing in the vector D, which has been assigned the class c:

$$P(D \mid c) = \prod_i P(d_i \mid c). \tag{4.4}$$

$P(d_i \mid c)$ can be estimated from the training data. Applying the naive assumption to equation 4.3 yields:

$$P(c \mid D) = \frac{P(c) \prod_i P(d_i \mid c)}{P(D)}. \tag{4.5}$$

The assigned class can then be calculated as:

$$c^*(D) = \arg\max_j \frac{P(c_j) \prod_i P(d_i \mid c)}{P(D)}. \tag{4.6}$$

$P(D)$ is positive and identical for every class $c \in C$, therefore it can be dismissed and the final result can be calculated as [26]:

$$c^*(D) = \arg\max_j P(c_j) \prod_i P(d_i \mid c). \tag{4.7}$$

Once the model is trained, this calculation is extremely fast because it consists only of multiplication. In practice, loglikelihoods are used instead of likelihoods to simplify computation and mitigate precision errors.

The Higgs Boson Portal utilises a naive Bayes classifier to distinguish between articles studying the Standard Model of particle physics and articles searching beyond the Standard Model. These two directions of research generally use slightly different wording, which allows for the classification process to be accurate, For example, the word "search" appears more often when searching beyond the Standard Model. It is not necessary to know these differences beforehand.

4.2.2.2 Named Entity Recognition

To extract more concrete information from the articles, e.g. integrated luminosity, centre-of-mass energy, decay products or production modes, a different method must be employed. These entities are dependent on the context in which they are used. For example, in the text: *"Measurements of gluon fusion and vector-boson-fusion production of the Higgs boson in $H \to WW^* \to e\nu\mu\nu$ decays using pp collisions at $\sqrt{s} = 13\,TeV$ with the ATLAS detector"* [27], two production modes are mentioned – gluon fusion and vector boson fusion. The decay mode is expressed using the notation "$H \to WW^*$". The centre-of-mass energy ($\sqrt{s}$) is given as the number "13" combined with the "TeV" unit.

The process of automatically recognising and extracting meaningful words, phrases or numeric values is called Named Entity Recognition (NER). *"Named entity recognition (NER) is the problem of locating and categorizing important nouns and proper nouns in a text"* [28].

NER has to recognize syntactic structures of the text, and therefore is language specific. Most scientific articles are written in English, therefore the HBP uses a NER model pre-trained on the English language. The training can be extended for specific needs of the domain.

After extracting entities from the text, they have to be parsed or categorised. When numeric values are expected, the number and unit are parsed algorithmically using a set of predetermined rules. In case of the decay mode and the production mode, the category is decided by identifying keywords and special characters in the extracted named entity. Identifying keywords and special characters only in the named entity yields better results that searching in the full text, where the context of the keywords are ignored.

4.2.3 Web Services and the Cloud

The gathered information needs to be presented in a conventional and a modern way. Hosting a website with the HBP is the preferred way to provide accessibility for any user with an Internet connection and an Internet browser. Traditionally, to host a website, it is necessary to dedicate hardware to run the web server and then buy the domain dame, e.g. hbp.com. While this provides a lot of control over the infrastructure, it takes more time and money to set up.

An alternative is to use an existing cloud service provider to host the portal. Using such a service, it is possible to set up a production ready web server in a short time and eliminate the need for own dedicated hardware. These services are usually paid, but many offer a starting tier which is free to use. The free services are limited in terms of capacity and performance, but these limits typically far exceed the needs of the HBP. The downsides are the need to rely on a 3rd party to keep the system up and running, and having only limited control over the infrastructure of the system.

The last option is to host the application on the internal hosting service of CERN. The CERN Application Hosting Service allows for deployment of applications with the Platform-as-a-Service (PaaS) paradigm [29]. This service provides the benefits of a cloud-based deployment, while maintaining high control over the application infrastructure. It is therefore recommended that the application is hosted on this service.

4.3 Architecture

This section outlines the top-level architecture of the Higgs Boson Portal.

4.3.1 Database

The collected data needs to be readily accessible to users. Therefore, it must be saved in a database. Traditionally, relational (SQL) databases were used to handle storage and querying. Relational databases rely on a predefined schema, which needs to be updated to propagate a change in the domain model [30]. On the other hand, the HBP is expected to evolve continuously, and the domain model might change during development. Also, the structure of the database is expected to be very simple. Therefore, a NoSQL (Not Only SQL) database is more suited for the task. It does not use a predefined schema and allows for easy changes in the domain [30].

A document-oriented NoSQL database operates with *collections* and *documents*. Collections can be thought of as the tables of a relational database, and documents as the rows [30]. Unlike relational databases, collections do not enforce strict rules for the properties of the documents. Collections typically group together similar documents, e.g. articles, and provide querying and searching functionalities.

The format of a document in a NoSQL database is not as normalised as in relational databases. It varies under different database implementations. For example, a document can be stored in XML, JSON, or a custom format derived from these basic types.

The format of a document is not normalised, therefore a database *driver* is required. The driver is typically a software library, which converts documents into their native representation in the chosen programming language.

Generally, drivers also provide methods for creating, updating, deleting and querying documents in a collection.

A database is fairly simple to set up using an existing database cloud service. Having a cloud-hosted database eliminates the need for a dedicated server and enhances availability, scalability and security [31].

4.3.2 Client-server architecture

The HBP is using multilayered architecture. This architecture is consistent with the "separation of concerns" (SoC) design principle. The SoC principle dictates that separate sections of a software application should address separate concerns [32]. Three layers are implemented. The service layer handles the core logic of the HBP. The controller layer is responsible for transferring data between the server and the client. Finally, the role of the presentation layer is to display the processed content to the user.

The first two layers are the responsibility of the server, and the presentation layer is implemented by the client. This is the client-server architecture [33].

The client-server architecture is the most common type of architecture used in modern web applications. In this environment, personal computers of the users are the clients, and one central computer – server communicates with the clients [33].

The server handles the core business logic of the HBP – fetching articles, categorisation, database management and authentication. It accepts requests from the clients and sends back responses. The server can be deployed to a CERN hosting service, which handles the initial setup and provides the infrastructure for the HBP service to be immediately available.

The client of the HBP is a website, more precisely a Single Page Application (SPA). It consists of a single HTML file and uses Javascript to dynamically fetch content from the server and display it to the user. The client does not handle any computational logic. This separation allows for independent development of the client and the server.

4.3.3 HTTP and REST

The client and server are separate entities and therefore they need to communicate using a selected protocol. The obvious choice is the Hypertext Transfer Protocol (HTTP). *"The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990"* [34]. Specifically, HBP uses the encrypted version – HTTPS.

HTTPS works on the request-response principle and is supported by all modern web browsers. It supports multiple data formats, such as plain text,

JSON, XML, image data and more. For the Higgs Boson Portal, the JSON format is used because of its simplicity.

In order to retrieve information from the server, clients need to access the server Application Programming Interface (API). An API exposes a set of resources and functions that facilitate interactions between the client and the server. The API listens to client requests and responds adequately.

The REST architectural style is commonly applied to web APIs. The REST style outlines multiple constraints that need to be applied in order for an API to be classified as a RESTful API, or REST API. The most important constraint is statelessness. It dictates that the server is not required to memorize the state of the client application. Conforming to the REST principles makes code more readable and testable [35]. The Higgs Boson Portal uses a REST API for interactions between the client and the server.

4.4 Tools

Building the proposed web application from scratch is not feasible. Third party services, tools and libraries have to be used during development [11]. Using adequate external tools also allows future developers to familiarise themselves quickly with the source code without the need for extensive documentation.

4.4.1 Git

All source code of the Higgs Boson Portal is versioned using the Git versioning system and stored remotely in two GitHub repositories. The two repositories are dedicated to the web server and the web client respectively. This allows for independent development of the two components. After completing the project, the source code will be transferred to internal CERN Gitlab repositories.

4.4.2 Python

The server side (back-end) of the Higgs Boson Portal uses Python 3.8 as the main language. Python is a popular programming language, used for web-development, software development, AI, statistics, system scripting and more [36]. Python has been chosen because of its extensive library ecosystem and ease of use. To install and upgrade dependencies, the `pipenv` package is used. The `pipenv` package creates a virtual environment to store project dependencies and therefore does not interfere with the system environment [37].

4.4.3 MongoDB

The HBP uses the MongoDB database. MongoDB is a NoSQL, document-oriented database. A document is stored in a format similar to a JSON object that allows for primitive types, nested object types and array types [38]. There is no predefined schema for a document, therefore documents can be arbitrarily modified or extended. This allows for quick iterations and rapid development [39]. MongoDB offers a powerful query language, aggregations, indexing and file storage [38].

MongoDB offers a cloud-based production database that is free to use for small projects. An alternative is to run the database on the server computer. For development and testing on Linux/Ubuntu, a MongoDB database can be instantiated locally.

4.4.4 Scrapy

It is possible to create a simple HTML parsing tool to scrape the target websites. However, there are already multiple Python libraries and frameworks to choose from, such as *Scrapy* or *BeautifulSoup*. The Higgs Boson Portal utilises the Scrapy framework to collect data from the Tevatron experiments.

"Scrapy is a fast high-level web crawling and web scraping framework, used to crawl websites and extract structured data from their pages. It can be used for a wide range of purposes, from data mining to monitoring and automated testing" [40].

Scrapy uses so-called "spiders" to operate. A spider must be provided with a list of website URLs to download and a function to extract the needed data from the response [40]. The HTML file is parsed using the XPath library, which provides querying methods to extract specific HTML nodes [41].

Scrapy is able handle multiple concurrent scraping tasks, which increases efficiency [40]. This additional speed bonus is required when running Scrapy on the cloud, where computing time and power are limited by the provider.

The result can be either saved to a file or processed further. In case of the HBP, it is saved directly to the database.

4.4.5 NLTK, Scikit-learn and spaCy

After the relevant data is collected, the classification methods are applied. Each item (article) is passed through a pipeline that extracts various information from the text. Multiple open source Python libraries are used for this task, mainly NLTK, Scikit-learn and spaCy.

NLTK stands for Natural Language Toolkit. It is a *"leading platform for building Python programs to work with human language data"* [42]. NLTK is a diverse library, but the Higgs Boson Portal uses it only for text pre-processing, such as filtering stopwords and stemming.

Scikit-learn is a library that provides machine learning utilities for Python. Scikit-learn also offers simple and efficient tools for data analysis [43]. The tool used by the HBP is classification. If given labelled training data, Scikit-learn is able to train a model that can classify new examples. The naive Bayes classifier, described in section 4.2.2.1, is implemented in this library.

Implementing Named Entity Recognition, described in section 4.2.2.2, is achieved using the spaCy library. According to the documentation, *"spaCy is a free open-source library for Natural Language Processing in Python"* [44]. The spaCy library be used to can train a model based manually annotated training data. The trained model is saved to a file, which is then used for recognising entities in new examples [44].

After the HBP extracts all available information from the articles, they are updated in the database to include the categorisation information.

4.4.6 Flask

The HBP server uses Python as the main language. By itself, Python does not provide utilities to easily run a production HTTP server. Typically, a web framework must be used to accept requests from clients and send back responses. A sophisticated web framework abstracts away the technical details and enables developers to focus on the business logic of the web application.

The most popular Python web frameworks include Django, Flask, FastAPI. The required functionality of the Higgs Boson Portal can be achieved using any of these frameworks. Flask is chosen because of its minimalistic architecture and high level of customisability.

Flask is a microframework. *"The "micro" in microframework means Flask aims to keep the core simple but extensible"* [45].

The main responsibility of Flask is the controller layer. There can be multiple controllers in the controller layer. One of the controllers is the REST API. Flask uses a routing mechanism that maps the destination URLs of HTTPS requests (e.g. `/api/articles` or `/api/users`) to Python function calls. Therefore, the service layer can be implemented using pure Python functionalities and other external libraries. The other controller is the command line interface (CLI). This interface is used to trigger commands directly. The CLI commands are also mapped to Python functions. Commands are used to manually trigger the search for articles and the subsequent classification.

To serve a Flask application, an HTTP server must be used. Flask provides a development HTTP server, but it is not suitable for production use. When deploying the application, a production-ready WSGI server must be used along with Flask [45]. The Web Server Gateway Interface (WSGI) is a standard for Python web servers [46]. Web frameworks that implement the WSGI standard can be easily ran using a WSGI server, such as *gunicorn* [47].

4.4.7 React

The HBP client is Single Page Application displayed in a web browser. Web browsers can only display pages written in HTML and CSS. Javascript can be used to implement dynamic loading, advanced animations and content manipulation. It is possible to create a web portal with just these tools, but using a high level front-end framework can increase development efficiency [48].

React is a JavaScript library for creating user interfaces [49]. It can be classified as a model-view-viewmodel (MVVM) framework [11, 50]. React is centered around the concept of *components*. Components are the main building blocks of a React-based application [48]. They encapsulate a specific section and allow it to be reused across the application. Examples of React components are: `Button`, `Navigation`, `DropdownMenu`. The encapsulation adheres to the SoC principle [32].

To use React in production, the source code is compiled into static HTML, CSS and Javascript files. These files can be served from the Flask WSGI server [45]. Serving these static files from the Python server is not efficient, but it allows for both the server API and the client to be hosted under the same domain (origin). This prevents cross-origin safety issues in web browsers [51]. An alternative is to set up a reverse proxy to handle incoming traffic. A reverse proxy could only dispatch specific requests to server and serve the static files directly. The expected traffic on the HBP is small, therefore setting up the proxy is not required.

4.5 Services

This section introduces tools that are not software libraries, but rather services – businesses, that the HBP employs to fulfill the mentioned requirements.

4.5.1 MongoDB Atlas

MongoDB Atlas is a global service offering a fully managed MongoDB cloud database [52]. It provides a browser user interface for creating and managing databases, collections and documents. The user interface also enables querying, searching and basic data analysis. HBP uses the Mongo Atlas as its production database solution.

4.5.2 CERN Web Services and OpenShift

The application source code must be stored to a server, where it is ran without stopping. The CERN Web Services are used to deploy the application to the Internet. Specifically, CERN Web Services utilise the OpenShift platform. The OpenShift platform allows for building, deploying, running and managing applications [53].

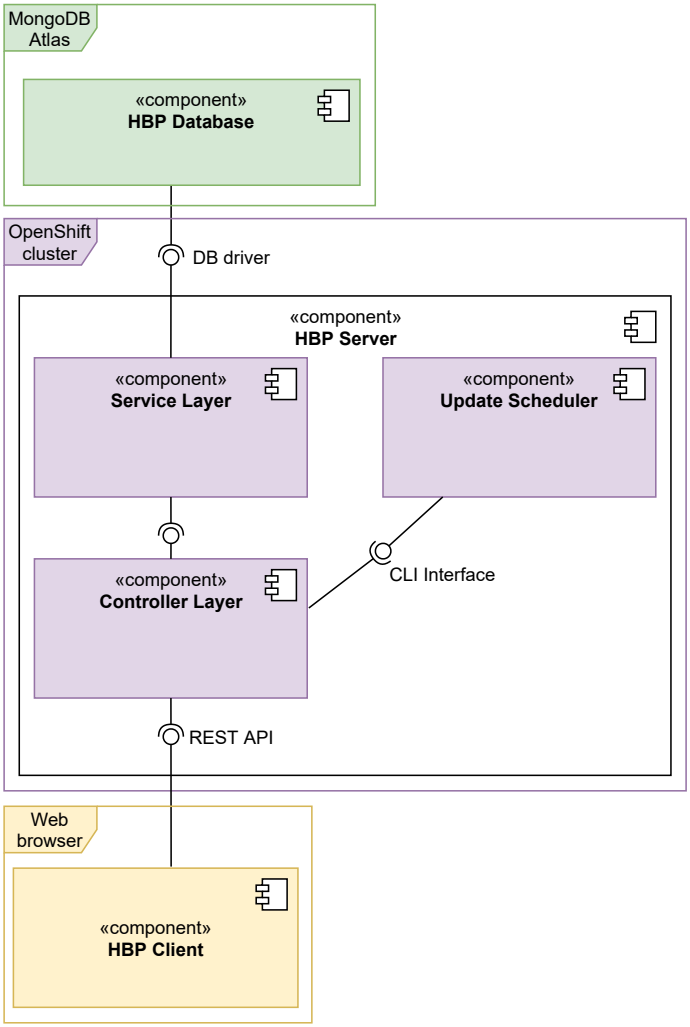

Figure 4.1: UML component diagram of the HBP

The OpenShift platform deploys the components of the applications in customisable Docker containers, which can be ran on multiple physical machines concurrently. OpenShift handles the orchestration of the containers automatically [53].

Applications deployed to the CERN Web Services are accessible within the `cern.ch` domain. By default, this domain communicates using the HTTPS protocol and therefore is secured. Registration of a new TLS/SSL certificate is not required [53].

Chapter 5

Implementation

In this chapter the implementation procedure of the Higgs Boson Portal is described. Technical challenges encountered during development are introduced and the solutions presented.

5.1 Environment

It is important for the system to work predictably on different machines or operating systems. Therefore, the environment is virtualised. This means that rather than relying on system libraries and configurations, a virtual Python environment is created using Pipenv. In order to run commands within the Pipenv environment, the Pipenv shell is used. A command is ran in the Pipenv shell [37] by

```
pipenv run <command>.
```

Alternatively, it is possible to switch to the pipenv shell using the command `pipenv shell`, which ensures that all following commands will be ran in the correct environment [37].

For the client application a similar solution is implemented using the NPM package manager. However, the NPM environment is only necessary for the development phase. The client project is compiled for production and the resulting static files are provided by the server. No external dependencies need to be present in the live environment, because they are already included in the compiled files.

5.2 Flask Server

The Flask application is initialised in the `app.py` file in the root project directory, using the following commands [45] (`__name__` is a built-in Python variable):

```
# app.py
from flask import Flask

app = Flask(__name__)
```

To run this application locally, the Flask development server is used [45]. This server will listen to requests at `localhost:5000`. Execution under the Pipenv environment is achieved by:

```
pipenv run flask run
```

5.3 CLI Commands

Flask provides a functionality to register CLI commands to the controller layer. Multiple commands are employed by the HBP. An example is given for the `update` command. The commands are created in `commands.py` using the `click` package, which is installed along with Flask.

```
# commands.py
import click
from flask.cli import with_appcontext
from service import HBPService
from database import mongo

service = HBPService(mongo)

@click.command("update")
@with_appcontext
def update():
    print("Updating...")
    service.update()
```

The command is registered to the main Flask application:

```
# app.py
from commands import update

app.cli.add_command(update)
```

Running the command is possible from the shell terminal using `pipenv run flask update`. The command is dispatched to the service layer, where the `update` method is defined. The update process consists of multiple steps, which are described in detail in the following section.

5.4 Filling/Updating the Database

5.4.1 Getting Articles via the CDS API

As mentioned in sections 3.1.2 and 3.1.3, the CDS API can be used to fetch articles from the legacy LEP experiments, as well as the ongoing ATLAS and CMS experiments. The CDS API is available in the XML and JSON formats [16]. The data available slightly differs between the two formats. For example, the date of creation of an article in not present in the XML API responses. Both formats are therefore used to extract required information.

The API works similarly to the search page of the CDS. Search parameters are given in the URL of the request to the API. In case of the XML API, the article properties must be specifically requested with *tags*. A tag is three digit number that is used to request specific information from the API [16]. The tags required for the HBP are:

```
tags = {
    "title": "245",
    "supersedes": "780",
    "superseded": "785",
    "abstract": "520",
    "report_number": "037",
    "doi": "024",
    "cds_id": "001",
    "files": "856",
}
```

The complete search parameters are:

- `cc` – category, e.g. ATLAS Papers or CMS Physics Analysis Summaries,
- `of` – the output format, `"xm"` for XML,
- `ln` – language, `"en"` is used,
- `ot` – the output tags, separated by a comma,
- `rg` – number of results, maximum is 200,
- `jrec` – the index of the first result, used for pagination,
- `p1` – the search pattern, `"higgs"` is used,
- `f1` – location of the `p1` pattern, `"title"` is used.

An example of the request is:

```
http://cds.cern.ch/search?cc=ATLAS+Papers&of=xm&
ot=245,780,785,520&ln=en&jrec=1&rg=200&p1=higgs&f1=title
```

The requests are sent using the built-in Python `requests` package. The body of the response is used to create an XML tree object from the `xml` library. The object contains a collection of articles. The CDS JSON API works similarly, except instead of tags, names of the properties must be supplied. The HBP requires the `creation_date` attribute. The API is queried multiple times, because the maximum number of articles per one request is limited.

```
# cds/search.py
import requests
from xml.etree import ElementTree

***

xml_response = requests.get(url, xml_params)
collection = ElementTree.fromstring(xml_response.text)

json_response = requests.get(url, json_params).get()
dates = [item["creation_date"] for item in json_response]
```

5.4.2 Scraping CDF and DØ articles

As mentioned in section 4.2.1, article data from the Tevatron experiments must be collected via a web scraping method. Scrapy is used to simplify the creation of scrapers. The following code is the implementation of the `CdfScraper` class.

```
import re
import scrapy
import urllib.parse as urlparse

# extend the default scrapy.Spider class
class CdfScraper(scrapy.Spider):

    # supply a list of URLs to download
    urls = ["https://www-cdf.fnal.gov/physics/new/hdg/
    Published_files/widget1_markup.html"]

    # method called by Scrapy to start sending requests
    def start_requests(self):
```

```
            for url in self.urls:
                yield scrapy.Request(
                    url=url,
                    callback=self.parse
                )

        def parse(self, response, **kwargs):
            ***
```

The `parse` method is used to extract information from the downloaded HTML file. For example, to get the titles of all articles:

```
def parse(self, response, **kwargs):
    # Select all articles
    for paper in response.xpath('//body/p'):
        title = paper.xpath('.//b//text()').get().strip()
        ***
    ***
```

The same method is applied to the rest of the required data. The web scraping task is ran only once when initially filling the database. During the periodic updates to the HBP, only fetching data from the CDS API is required.

5.4.3 Updating the database

Articles collected by both above described methods are represented with the same format – a Python dictionary. This format is accepted by `pymongo` (MongoDB driver), which converts the articles to the native database format. When querying the database, the articles are converted back to dictionaries.

First, the database connection is initialised, using the `DB_URI` environment variable, which contains a link and a password to the HBP cloud database. The environment variable is not versioned using Git, because it contains sensitive information that grants direct access to the HBP database.

```
import pymongo
from dotenv import load_dotenv

# Load environment variables
load_dotenv()

db_uri = os.getenv("DB_URI")

# Initialise connection to the database
mongo = pymongo.MongoClient(db_uri)
```

```
# Select the "articles" collection
articles = mongo.hbp.articles
```

Inserting new articles into the database is straightforward. However, when updating existing articles, they need to be uniquely identified to prevent duplicates in the database. Only the ATLAS and CMS articles are updated, therefore the identification number of the article within the CDS can be used:

```
for article in articles:
    if "cds_id" in article:
        # Articles from the CDS API
        papers.update_one(
            { "cds_id": article["cds_id"] }, # Filter
            { "$set": article }, # Update query
            upsert=True # Insert if article does not exist
        )
    else:
        # Scraped articles (only present when filling
        # the database for the first time)
        papers.insert_one(article)
```

The first argument of the `update_one` method is the filter that selects the article to be updated. The second argument is the update query. The `$set` operator replaces existing fields with the new fields from the article dictionary. Fields that are not present in the dictionary are not modified. The keyword argument `upsert=True` guarantees that if the filter does not find the article which has to be updated, it is inserted into the database as a new document.

5.5 Categorisation

Categorisation of new articles is triggered after every update to the HBP database. It can be also launched manually with the registered `categorise` command, which updates all articles. This command would be used in case the categorisation rules were to change, without the need to search the Internet for new updates. Registering a command is described in section 5.3.

Articles that need to be categorised are passed through a pipeline that extracts all required information. The categorisation pipeline is defined in `pipeline.py`.

```
# pipeline.py

def categorisation_pipeline(article, pipes):
    for pipe in pipes:
        article = pipe(article)
    return article
```

Any classification task can be expressed as a function (pipe) that modifies the original article dictionary. Usually, the pipe function adds new properties (fields) to the dictionary. For example, the `classify_model` pipe creates the `model` property, which specifies whether an article is describing the Standard Model or searching beyond the Standard Model. Running all classification tasks is achieved by calling the `categorisation_pipeline` function and supplying a list of pipes:

```
# pipeline.py

def categorise(article):
        return categorisation_pipeline(
            article,
            [classify_model, extract_entities,
            extract_luminosity, extract_energy,
            extract_collision, extract_production,
            extract_decay_a, extract_decay_b,
            extract_decay_particles,
            delete_entities, classify_stage]
        )
```

5.5.1 Text Classification

This sections presents the implementation of the naive Bayes classification model, described in section 4.2.2.1.

5.5.1.1 Training Data

The training data in the `.csv` format is prepared first. It has two columns, which represent the text of the article and the physics model. The training data is created manually by labeling the article text (title + abstract), with either "SM" for "Standard Model", or "BSM" for "beyond the Standard Model".

5.5.1.2 Training the Model

First, the texts are pre-processed with the NLTK library, in order the train the model. The pre-processing consists of removing stopwords, pronouns, numbers and punctuation.

Then, a *vectorizer* is created. A vectorizer transforms text into a feature vector. There are two common approaches on how to represent feature vectors. In the simple solution, elements of the vector represent the frequencies of words in a given document.

A more advanced solution should also take into account how often a certain word occurs in the whole collection of documents. Words that occur in too many documents should have reduced deciding power. This is named the Term Frequency - Inverse Document Frequency measure (TF-IDF) [54]. In TF-IDF, elements of the feature vector represent how relevant a word is to a given document. The value is proportional to the frequency of the word in a document, but inversely proportional to the frequency of the word in the whole collection. In a TF-IDF matrix, each column represents one term and each row represents one document.

The TF-IDF measure is implemented by the `TfidfVectorizer` class in the Scikit-learn library. The `fit_transform` method causes the vectorizer to learn the vocabulary and construct the TF-IDF matrix of the training data [54].

```
from sklearn.feature_extraction.text import TfidfVectorizer
from util import preprocess_text

texts = [preprocess_text(text) for text in texts]
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(texts)
```

Lastly, the classifier is trained. A variant of the naive Bayes classifier is used – the complement naive Bayes classifier. This type of classifier outperforms other types when classifying text data [55]. The complement naive Bayes classifier is implemented by the `ComplementNB` class in Scikit-learn. The classifier is trained by the `fit` method. The first argument is the TF-IDF matrix constructed by the vectorizer. The second argument is an array of labels ("SM" or "BSM"), corresponding to the rows of the TF-IDF matrix [54].

```
from sklearn.naive_bayes import ComplementNB

classifier = ComplementNB()
classifier.fit(tfidf_matrix, models)
```

Both the vectorizer and classifier are saved into a file once the training is complete. In order to classify new articles, they are loaded from a file without the need to go through the training process again. It is possible to make a prediction with the following function:

```
def predict_model(text):
    processed_text = preprocess_text(text)
    tfidf_matrix = vectorizer.transform([processed_text])
    model = classifier.predict(tfidf_matrix)

    return model[0]
```

The vectorizer `transform` method accepts an iterable of documents. If only one document is present, it must passed as a singleton array. The classifier makes the prediction with the `predict` method. It returns an array of predictions with a single element.

5.5.2 Named Entity Recognition

A training process is also applied to the NER model, described in section 4.2.2.2.

5.5.2.1 Training Data

The training data for the NER is labelled manually. It consists of text documents, each with a list of corresponding named entities and their positions in the document (annotations).

A training document might look like: *"Search for charged Higgs bosons produced via vector boson fusion and decaying into a pair of W and Z bosons using proton-proton collisions at* $\sqrt{s} = 13\,TeV$*"* [56]. The highlighted phrases are the named entities. Each named entity has a type, offset from the beginning of the document and a length. In this example, the types would be "PRODUCTION", "DECAY" and "ENERGY".

5.5.2.2 Training the Model

The model has to recognise the entities automatically, therefore it must be familiar with the syntax of the English language. In spaCy, a blank language model can be initialised. This model is pre-trained for the syntactical analysis, but does not yet recognise any entities:

```
import spacy

nlp = spacy.blank("en") # initialise blank English model
nlp.add_pipe("ner") # add NER step to the pipeline
```

The model is trained in multiple iterations. Seeing the training data only once is not sufficient for the model to be accurate. The number of iterations cannot be set too high in order to avoid over-fitting [44]. For the HBP, 50 iterations are performed. In each iteration, the training documents are randomly batched into groups of 8. The model is updated after every batch.

```
import random
from spacy.training import Example
from spacy.util import minibatch

# Create examples from training data
examples = [
    Example.from_dict(nlp.make_doc(text), annotations)
    for text, annotations in train_data
]

nlp.initialize()

for i in range(200):
    random.shuffle(examples)

    for batch in minibatch(examples, size=8):
        nlp.update(batch)
```

The trained model is saved to a file to be used later. A similar model is trained to recognise decay products (particles) within a "DECAY" entity, displayed in the example in section 5.5.2.1.

In order to run the recogniser on new articles, the model is loaded from a file and applied to the text of the article. It returns a list of named entities, along with their types. These entities are processed further in the pipeline, e.g. parsing numeric values.

5.6 Web API

This section details the implementation of the HBP API.

5.6.1 Resources

A fundamental concept of RESTful APIs are *resources*. A resource is an object with a type, associated data and a set of methods that operate on it [57]. The methods typically correspond to the GET, POST, PUT, PATCH and DELETE methods of the HTTP protocol.

The core resource of the HBP are the articles. For articles, all API methods are implemented, except POST and PUT, which are used for inserting or completely replacing resource objects. Both these methods are replaced by the automated update system.

The API component is implemented as a Flask blueprint. A Flask blueprint is a module that can be attached to the core application [58]. The API methods are registered in the API blueprint using the Flask routing mechanism [45].

```
# api.py
from flask import Blueprint

api = Blueprint("api", __name__)

@api.route("/papers", methods=["GET"])
def get_papers():
    # Retrieve all papers from the service layer
    ***

@api.route("/papers/<id>", methods=["PATCH"])
def patch_paper(id):
    # Update paper by id
    ***
```

The blueprint is then registered to the main application.

```
# app.py
from api import api

app.register_blueprint(api, url_prefix="/api")
```

All routes within the blueprint are prefixed with the name of the blueprint. For example, to retrieve all articles, the following HTTP request must be sent:

```
GET /api/papers/ HTTP 1.1
```

5.6.2 Authentication and Authorisation

Administrators of the HBP must be able to delete articles and modify article categorisation information manually. These actions are available through the PATCH and DELETE methods of the API. However, these methods require authentication and authorisation to prevent destructive actions.

Administrators can create an account on the HBP via the `/api/register` API route. This account is initially not active, until it is verified by an already existing administrator.

Authentication is implemented using JSON Web Tokens (JWT) [59]. When an administrator logs in with his username and password, he receives a short-lived (one hour) JWT token. This token is sent with every subsequent request to the server, in order to identify the administrator. The JWT token expires after a set time period, after which it must be refreshed. The identity of the administrator can be read from the token and various levels of permissions can be granted.

It is not safe for the administrator to handle the token manually. After logging in, it is automatically stored in the web browser cookies. When sending

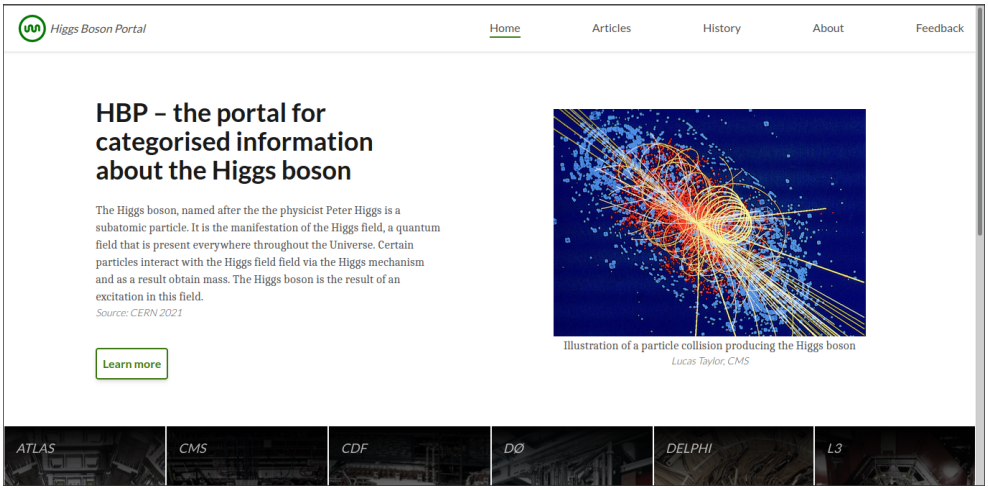

Figure 5.1: HBP Home page

requests, the browser attaches the token to the request. The cookie "SameSite" attribute is set to "Strict", which means it is only available to the HBP domain to prevent cross-site request forgery (CSRF) attacks [60].

Administrator accounts are stored in the database in the "users" collection. A document in this collection consists of the name of the administrator, e-mail and an encrypted hash of his password.

5.7 Web Client

In this section, the process of implementing the web client using the React framework is described.

5.7.1 Environment

A new React application is initialised using the `create-react-app` package. This will create a Node.js environment. Node.js is required to compile the application to Javascript that can be then ran in a web browser. The application uses the NPM package manager to install and manage external dependencies. The application is compiled by running `npm run build`.

5.7.2 Design

The user interface is composed of multiple pages. A brief description of the Higgs boson is given on the "Home" page. Below, both completed and ongoing experiments are listed with a short description. The Home page is displayed in Figure 5.1.

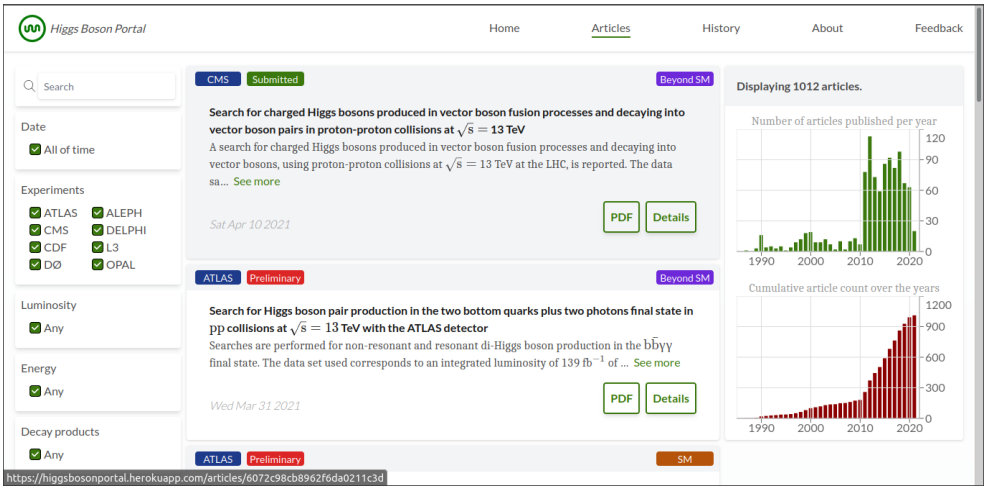

Figure 5.2: HBP Articles page

The main component of the HBP is the database of categorised articles. These can be viewed on the "Articles" page, displayed in Figure 5.2. The articles can be filtered by date, name of experiment, luminosity, center-of-mass energy, decay products, production modes, stage and the physics model (SM or BSM). Any of the filters can be combined. Additionally, searching for keywords in the title or abstract manually is possible.

Detailed information about a specific article, including the categorisation information, attached files and related articles can be viewed by clicking on the article.

The "History" page details the development of various measurements. These include the lower and upper mass limits of the Higgs boson, and the precision of the Higgs boson mass measurement.

The "Feedback" page allows users to send feedback to the administrators.

5.7.3 Administration

A log-in form is available for administrators of the portal. When logged in, administrators are able to perform tasks mentioned in section 4.1.1.

5.8 Deployment

5.8.1 Production Environment

In order to run the server application with a production WSGI server, a `wsgi.py` file is created:

```
# wsgi.py
from app import app

if __name__ == "main":
    app.run()
```

This setup can be then launched using the `gunicorn` package from shell. In order to configure OpenShift to run this application, a Dockerfile must be provided.

```
# Dockerfile
FROM python:3.8

COPY ./requirements.txt ./

RUN pip install -r requirements.txt

COPY . .

EXPOSE 8080

RUN chmod +x start_server.sh

CMD start_server.sh
```

This file is used to create the Docker image of the application. The `CMD` command specifies the process that is ran when the container is launched. The HBP server is launched with the following shell script:

```
# start_server.sh

python update_scheduler.py &
gunicorn --bind 0.0.0.0:8080 wsgi:app
```

The `python update_scheduler.py &` instruction in the above shell script launches a scheduling task in the background. This process triggers a `flask update` command every 24 hours. The second command starts a `gunicorn` web server listening on the port 8080.

In order to import the application to the OpenShift platform, an OpenShift project is first created. A new application is created within this project via the OpenShift command line tool, by using the command

```
oc new-app ssh://git@github.com:zacikpet/hbp.git
    --strategy=docker
    --source-secret=hbp
```

The source repository is private, therefore the `new-app` command must be authenticated using an SSH key pair. The private key is stored in the OpenShift environment and the public key is stored in the GitHub repository.

Running the `new-app` command pulls the code from the repository and builds the application on the OpenShift platform. A set number of containers (replicas) is deployed. The traffic on the HBP is expected to be relatively small, therefore only one replica is necessary.

To make the application publicly available via the domain name, an OpenShift *route* is created, using the following command of the OpenShift CLI:

```
oc create route edge
    --service=hbp
    --insecure-policy=Redirect
    --hostname=higgs.web.cern.ch
```

The `edge` option specifies how the traffic is encrypted. Internal traffic within the OpenShift project cluster is not required to be encrypted.

The `insecure-policy=Redirect` option guarantees that non-encrypted connections will be automatically redirected to HTTPS.

By default, CERN Web Services only allow connections from within the CERN Intranet. The route is made visible to the Internet with the following command:

```
oc annotate route hbp \
    router.cern.ch/network-visibility=Internet
```

5.8.2 Continuous Delivery

The compiled files of the web client must be replaced in the HBP server repository after every update to the client. Instead of manually copying the files to the server repository, an automated workflow is set up with GitHub Actions.

GitHub Actions are able to run pre-defined tasks after publishing a new version to the remote GitHub repository. Pushing a new version to the `hbp-client` repository triggers an automatic build by running `npm run build`. Then, the resulting build folder is copied to the main `hbp` repository. After copying, the files are automatically committed.

The OpenShift project is set to automatically publish new versions from the `hbp` GitHub repository. Therefore, running `git push` in either repository automatically integrates the changes to the production version of the portal.

Chapter 6

Testing

6.1 Automated Tests

The core logic of the HBP is implemented in the service layer. This layer needs to be tested independently of the database implementation and the controller layer.

The service layer follows the Inversion of Control (IoC) pattern. It has one dependency – the database client. The database client instance is passed to the service layer on initialisation. The service layer logic works independently of the database client instance.

When testing, the real database client is substituted by a testing instance from the package `mongomock`. The testing database is an in-memory database, that behaves similarly to a regular MongoDB database. Data in this database is erased on program shutdown. Unit tests of service methods are performed by creating a new instance of the `HBPService` and providing the test database:

```
import mongomock
from unittest import TestCase
from service import HBPService

class TestReadPapers(TestCase):
    def setUp(self):
        self.mongo = mongomock.MongoClient() # Create mock DB
        self.service = HBPService(self.mongo) # Initialise service

    def test_read_all_papers_empty(): # Test empty database
        all_papers = self.service.read_all_papers()
        assert len(all_papers) == 0
```

	TP	FP	FN	precision (%)	recall (%)	F_1-score (%)
Model	26	4	6	86.67	81.25	83.87

Table 6.1: Results from the naive Bayes classification test

6.2 Statistical Analysis

In this section, result of the categorisation accuracy are presented.

6.2.1 Text Classification

The naive Bayes text classifier (implementation in section 5.5.1) that decides whether an article belongs to the "SM" or "BSM" category is trained on a random set of 50 labelled articles. The test is performed on a different set of 50 articles. Results are evaluated with the F_1-score measure, where the F_1-score is the harmonic mean of precision and recall of the model [61]. The quality is assumed to be positive when the article is a search beyond the Standard model. Therefore, the F_1-score measure can be applied by setting:

- True Positives (TP): Correctly identified BSM articles
- True Negatives (TN): Correctly identified SM articles
- False Positives (FP, Type I errors): SM articles incorrectly labelled BSM
- False Negatives (FN, Type II errors): BSM articles incorrectly labelled SM

Precision, recall and F_1-score of the model are calculated as [61]:

$$\text{precision} = \frac{|\{\text{true positives}\}|}{|\{\text{true positives}\} \bigcup \{\text{false positives}\}|} \tag{6.1}$$

$$\text{recall} = \frac{|\{\text{true positives}\}|}{|\{\text{true positives}\} \bigcup \{\text{false negatives}\}|} \tag{6.2}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6.3}$$

Out of the 50 testing articles, 40 were classified correctly, resulting in a success rate of 80 %. Detailed results are displayed in Table 6.1. This result could be potentially improved by training the classifier with a larger data set.

	TP	FP	FN	precision (%)	recall (%)	F_1-score (%)
Luminosity	51	2	7	96.23	87.93	91.9
CoM Energy	51	0	9	100	85	91.9
Production mode	52	8	9	86.67	85.25	85.95
Decay product	71	17	19	80.69	78.89	79.78

Table 6.2: Results from the NER categorisation test.

6.2.2 Named Entity Recognition

The NER model (section 5.5.2) is trained on 100 training articles. The test is performed on 75 testing articles.

Instead of checking the recognised named entities directly, the extracted named entities are first processed algorithmically. The final categorisation results are reviewed. When testing, the entities are processed by the same algorithms that are used in the production environment.

It is possible that one article describes multiple decay modes or production modes. Additionally, multiple data sets with different integrated luminosities or centre-of-mass energies might be analysed in one article. Therefore, the categorisation accuracy is not evaluated per-article, but rather per-category, e.g. how many decay modes mentioned in an article are correctly identified.

The standard F_1-score measure is used for evaluating binary classification models [61]. However, it can be applied to this classification model, by setting [62]:

- True Positives (TP): correctly identified categories,
- False Negatives (FP, Type II errors): undetected categories,
- False Positives (FP, Type I errors): incorrectly predicted categories that are not mentioned in the text.

Precision, recall and F_1-score of the model are calculated identically to the definition in section 6.2.1. Complete results of the categorisation accuracy are displayed in Table 6.2.

Conclusion

This thesis describes how the goal was achieved to implement a web portal dedicated to Higgs boson research. The portal presents categorised information about the Higgs boson useful for particle physicists. Additionally, a description and various facts about the research are provided to make the portal attractive to the public.

The arguments for and against currently available resources were summarised. Requirements for the new portal were outlined, along with other nice-to-have features. A major part of the portal – the automatic categorisation of scientific articles based on the information in the title and abstract, was implemented.

The architecture of the portal is detailed in the thesis. Tools, software libraries and services required for running and maintaining the portal were introduced.

The implementation of the Higgs Boson Portal has been completed and documented. Multiple types of Artificial Intelligence text classifiers were used to categorise scientific articles efficiently. The training process of the classifiers was demonstrated. In this thesis, the front-end of the portal, in the form of a website, is documented.

The portal has been deployed to the Internet using the CERN web hosting services. The integrated database is updated automatically on a daily basis. Manual interventions can be performed using the OpenShift administration tools. Amendments to the web portal itself can be achieved through the synchronisation with Github/Gitlab.

An initial analysis of the performance of the new portal has been performed. The categorisation accuracy is deemed high enough for the portal to be useful, and it can be further improved, for example, by training on a larger dataset.

Concluding, all tasks of the thesis assignments regarding design, implementation, deployment and testing of the new web portal have been completed.

Bibliography

[1] CERN. The Higgs boson [online]. [Cited 2021-03-27]. Available from: https://home.cern/science/physics/higgs-boson

[2] Organtini, G. Unveiling the Higgs mechanism to students. *European Journal of Physics*, volume 33, no. 5, jul 2012: pp. 1397–1406, doi:10.1088/0143-0807/33/5/1397. Available from: https://doi.org/10.1088/0143-0807/33/5/1397

[3] CERN. The Standard model [online]. [Cited 2021-03-28]. Available from: https://home.cern/science/physics/standard-model

[4] Cottingham, W.; Greenwood, D.; et al. *An Introduction to Nuclear Physics*. An Introduction to Nuclear Physics, Cambridge University Press, 2001, ISBN 9780521657334. Available from: https://books.google.sk/books?id=0VIpJPn-qWoC

[5] Sopczak, A. Status of Higgs boson searches at the beginning of the LHC era. *Journal of Physics G: Nuclear and Particle Physics*, volume 39, no. 11, oct 2012: p. 113001, doi:10.1088/0954-3899/39/11/113001. Available from: https://doi.org/10.1088/0954-3899/39/11/113001

[6] CERN. CERN Storage [online]. [Cited 2021-03-30]. Available from: https://home.cern/science/computing/storage

[7] LHC. Relativity, Taking a closer look at LHC [online]. [Cited 2021-03-31]. Available from: https://www.lhc-closer.es/taking_a_closer_look_at_lhc/0.relativity

[8] LHC. Luminosity, Taking a closer look at LHC [online]. [Cited 2021-03-31]. Available from: https://www.lhc-closer.es/taking_a_closer_look_at_lhc/0.luminosity

[9] Sopczak, A. Precision Measurements in the Higgs Sector at ATLAS and CMS. *PoS*, volume FFK2019, 2020: p. 006, doi:10.22323/1.353.0006, hep-ex/2001.05927.

[10] Binoth, T.; Karg, S.; et al. Multi-Higgs boson production in the Standard Model and beyond. *Phys. Rev. D*, volume 74, 2006: p. 113008, doi:10.1103/PhysRevD.74.113008, hep-ph/0608057.

[11] Kupka, M. *Feasibility Study of Portal to Provide Knowledge about Higgs Boson to General Public and Experts*. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology, 2020.

[12] ATLAS. ATLAS Publications [online]. [Cited 2021-04-09]. Available from: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/Publications

[13] CMS. CMS Publications [online]. [Cited 2021-04-09]. Available from: http://cms-results.web.cern.ch/cms-results/public-results/publications/HIG/index.html

[14] CDF. CDF publications [online]. [Cited 2021-04-09]. Available from: https://www-cdf.fnal.gov/physics/new/hdg/Published.html

[15] D0. The DØ Collaboration's Publications [online]. [Cited 2021-04-09]. Available from: https://www-d0.fnal.gov/d0_publications/d0_pubs_list_bydate.html

[16] CDS. Search Engine API [online]. [Cited 2021-04-09]. Available from: https://cds.cern.ch/help/hacking/search-engine-api?ln=en

[17] Barate, R.; Bruneliere, R.; et al. Search for the Standard Model Higgs Boson at LEP. *Phys. Lett. B*, volume 565, no. hep-ex/0306033. CERN-EP-2003-011. CERN-L3-271, Mar 2003: pp. 61–75. 23 p, doi:10.1016/S0370-2693(03)00614-2, 22 pages, 10 figures Report-no: CERN-EP/2003-011. Available from: http://cds.cern.ch/record/610122

[18] Mitchell, R. *Web Scraping with Python: Collecting Data from the Modern Web*. O'Reilly Media, Inc., first edition, 2015, ISBN 1491910291.

[19] Sun, Y.; Zhuang, Z.; et al. A Large-Scale Study of Robots.Txt. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, New York, NY, USA: Association for Computing Machinery, 2007, ISBN 9781595936547, p. 1123–1124, doi:10.1145/1242572.1242726. Available from: https://doi.org/10.1145/1242572.1242726

[20] Chowdhury, G. Natural language processing. *ARIST*, volume 37, 01 2005: pp. 51–89, doi:10.1002/aris.1440370103.

[21] Androutsopoulos, I.; Koutsias, J.; et al. An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal e-Mail Messages. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, New York, NY, USA: Association for Computing Machinery, 2000, ISBN 1581132263, p. 160–167, doi:10.1145/345508.345569. Available from: https://doi.org/10.1145/345508.345569

[22] Mccallum, A.; Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification. *Work Learn Text Categ*, volume 752, 05 2001.

[23] Zhang, Y.; Jin, R.; et al. Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, volume 1, 12 2010: pp. 43–52, doi:10.1007/s13042-010-0001-0.

[24] Wilbur, W. J.; Sirotkin, K. The automatic identification of stop words. *Journal of Information Science*, volume 18, no. 1, 1992: pp. 45–55, doi:10.1177/016555159201800106. Available from: https://doi.org/10.1177/016555159201800106

[25] Joyce, J. Bayes' Theorem. In *The Stanford Encyclopedia of Philosophy*, edited by E. N. Zalta, Metaphysics Research Lab, Stanford University, spring 2019 edition, 2019. Available from: https://plato.stanford.edu/archives/spr2019/entries/bayes-theorem/

[26] Zhang, W.; Gao, F. An Improvement to Naive Bayes for Text Classification. *Procedia Engineering*, volume 15, 2011: pp. 2160–2164, ISSN 1877-7058, doi:https://doi.org/10.1016/j.proeng.2011.08.404, cEIS 2011. Available from: https://www.sciencedirect.com/science/article/pii/S1877705811019059

[27] Aaboud, M.; Aad, G.; et al. Measurements of gluon–gluon fusion and vector-boson fusion Higgs boson production cross-sections in the $H \to WW* \to e\nu\mu\upsilon$ decay channel in pp collisions at s=13TeV with the ATLAS detector. *Physics Letters B*, volume 789, Feb 2019: p. 508–529, ISSN 0370-2693, doi:10.1016/j.physletb.2018.11.064. Available from: http://dx.doi.org/10.1016/j.physletb.2018.11.064

[28] Zitouni, I. *Natural Language Processing of Semitic Languages*. 01 2014, ISBN 978-3-642-45357-1, doi:10.1007/978-3-642-45358-8.

[29] CERN. CERN Web Services [online]. [Cited 2021-05-11]. Available from: https://webservices.web.cern.ch/webservices/

[30] Venkatraman, S.; Fahd, K.; et al. SQL Versus NoSQL Movement with Big Data Analytics. *International Journal of Information Technology and Computer Science*, volume 8, 2016: pp. 59–66. Available from: https://www.semanticscholar.org/paper/SQL-Versus-

NoSQL-Movement-with-Big-Data-Analytics-Venkatraman-Fahd/ff1e0618d9dcad292af5ccf05cd11dfcafecce19

[31] Al Shehri, W. Cloud Database Database as a Service. *International Journal of Database Management Systems*, volume 5, 04 2013: pp. 1–12, doi: 10.5121/ijdms.2013.5201.

[32] Hürsch, W. L.; Lopes, C. V. Separation of Concerns. Technical report, 03 1995. Available from: https://www.researchgate.net/publication/2821402_Separation_of_Concerns

[33] Sulyman, S. Client-Server Model. *IOSR Journal of Computer Engineering*, volume 16, 01 2014: pp. 57–71, doi:10.9790/0661-16195771.

[34] Fielding, R. T.; Gettys, J.; et al. Hypertext Transfer Protocol – HTTP/1.1 [online]. RFC 2616, RFC Editor, June 1999. Available from: http://www.rfc-editor.org/rfc/rfc2616.txt

[35] Masse, M. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, 2011, ISBN 9781449319908. Available from: https://books.google.sk/books?id=eABpzyTcJNIC

[36] Van Rossum, G.; et al. Python [software]. 1991, [Cited 2021-04-08]. Available from: https://courses.minia.edu.eg/Attach/16028python_lecture1.pdf

[37] Pipenv: Python Development Workflow for Humans [software]. [Cited 2021-04-08]. Available from: https://pypi.org/project/pipenv/

[38] MongoDB [software]. [Cited 2021-04-08]. Available from: https://www.mongodb.com/

[39] Chodorow, K. *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. O'Reilly Media, 2013, ISBN 9781449344825. Available from: https://books.google.sk/books?id=uGUKiNkKRJ0C

[40] Scrapy [software]. 2021, [Cited 2021-04-08]. Available from: https://docs.scrapy.org/en/latest/index.html

[41] Clark, J, ed. and DeRose, S, ed. XML Path Language [online]. 2016, [Cited 2021-04-09]. Available from: https://www.w3.org/TR/1999/REC-xpath-19991116/

[42] Natural Language Toolkit [software]. 2021, [Cited 2021-04-08]. Available from: https://www.nltk.org/

[43] Scikit-learn: machine learning in python [software]. 2021, [Cited 2021-04-08]. Available from: https://scikit-learn.org/stable/

[44] Honnibal, M.; Montani, I.; et al. spaCy: Industrial-strength Natural Language Processing in Python [software]. 2020, doi: 10.5281/zenodo.1212303. Available from: https://doi.org/10.5281/zenodo.1212303

[45] Flask [software]. 2021, [Cited 2021-04-08]. Available from: https://flask.palletsprojects.com/en/1.1.x/

[46] Gardner, J. *The Web Server Gateway Interface (WSGI)*. 01 2009, ISBN 978-1-59059-934-1, doi:10.1007/978-1-4302-0534-0_16.

[47] Gunicorn. Gunicorn [software]. [Cited [2021-04-12]. Available from: https://gunicorn.org/

[48] Gackenheimer, C. *Introduction to React*. USA: Apress, first edition, 2015, ISBN 1484212460.

[49] React - A Javascript library for building user interfaces [software]. 2021, [Cited 2021-04-08]. Available from: https://reactjs.org/

[50] Syromiatnikov, A.; Weyns, D. A journey through the land of model-view-design patterns. In *2014 IEEE/IFIP Conference on Software Architecture*, IEEE, 2014, pp. 21–30.

[51] MD Web Docs. Cross-Origin Resource Sharing. [Cited 2021-05-21]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[52] MongoDB Atlas [online]. [Cited 2021-04-09]. Available from: https://www.mongodb.com/cloud/atlas

[53] Redhat, OpenShift. OpenShift Container Platform 4.7 Documentation [software]. [Cited 2021-05-11]. Available from: https://docs.openshift.com/container-platform/3.11/cli_reference/index.html

[54] Buitinck, L.; Louppe, G.; et al. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[55] Scikit-learn. Naive Bayes – scikit-learn 0.24.1 documentation [online]. [Cited [2021-04-10]]. Available from: https://scikit-learn.org/stable/modules/naive_bayes.html

[56] Sirunyan, A.; Tumasyan, A.; et al. Search for Charged Higgs Bosons Produced via Vector Boson Fusion and Decaying into a Pair of W and Z Bosons Using pp Collisions at $\sqrt{s} = 13$ TeV. *Physical Review Letters*, volume 119, no. 14, Oct 2017, ISSN 1079-7114, doi:10.1103/

physrevlett.119.141802. Available from: http://dx.doi.org/10.1103/PhysRevLett.119.141802

[57] RESTful API design: Resources [online]. [Cited 2021-04-12]. Available from: https://restful-api-design.readthedocs.io/en/latest/resources.html

[58] Flask. Modular Applications with Blueprints [online]. [Cited 2021-04-12]. Available from: https://flask.palletsprojects.com/en/1.1.x/blueprints/

[59] Jones, M.; Bradley, J.; et al. JSON Web Token (JWT). RFC 7519, May 2015, doi:10.17487/RFC7519. Available from: https://rfc-editor.org/rfc/rfc7519.txt

[60] MDN Web Docs. SameSite cookies [online]. [Cited 2021-04-21]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite

[61] Chinchor, N. MUC-4 Evaluation Metrics. In *Proceedings of the 4th Conference on Message Understanding*, MUC4 '92, USA: Association for Computational Linguistics, 1992, ISBN 1558602739, p. 22–29, doi: 10.3115/1072064.1072067. Available from: https://doi.org/10.3115/1072064.1072067

[62] Nadeau, D.; Sekine, S. A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*, volume 30, 08 2007, doi: 10.1075/li.30.1.03nad.

Appendix A

Acronyms

API Application Programming Interface

ATLAS A Toroidal LHC Apparatus

CERN Conseil Européen pour la Recherche

CMS Compact Muon Solenoid

CSRF Cross-site Resource Forgery

HBP Higgs Boson Portal

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

IoC Inversion of Control

JSON Javascript Object Notation

LEP Large Electron Positron Collider

LHC Large Hadron Collider Nucléaire (European Council for Nuclear Research)

NER Named Entity Recognition

NLP Natural Language Processing

NoSQL Not Only SQL

REST Representational State Transfer

SoC Separation of Concerns

SQL Structured Query Language

UI User Interface

URL Universal Resource Locator

WSGI Web Server Gateway Interface

XML Extensible Markup Language

Appendix B

Contents of enclosed media

- `readme.txt` ..... intructions on how to run the application locally
- `hbp-server` ..... the HBP server sources
- `hbp-client` ..... the HBP client sources
- `thesis` ..... the directory of LaTeX source codes of the thesis
  - `thesis.tex` ..... the thesis source
- `thesis.pdf` ..... the thesis text in PDF format