BSc Thesis in Software Technology Engineering

Project Report

Building a website to promote the CERN Academic Training lectures

Franciska-Leonóra Török, 293171 IT

Supervisors:

Kasper Knop Rasmussen
VIA University College

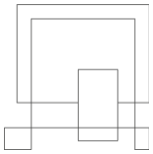
Maria Dimou
CERN IT, Academic Training

43,293 characters

Software Technology Engineering

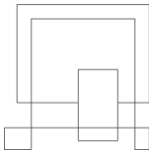
7th Semester

2023



Abbreviations – Acronyms

CERN	European Organization for Nuclear Research
IT	Information Technology (CERN Department)
RCS	Research and Computing Sector (CERN Department)
CDA	Communications, Devices & Applications (CERN Group)
SIS	Scientific Information Service (CERN Group)
ATC	Academic Training Committee
CDS	CERN Document Server
LHC	Large Hadron Collider
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
API	Application Programming Interface
XML	Extensible Markup Language
JSON	JavaScript Object Notation
DBoD	Database On Demand
UI	User Interface
UX	User Experience
SPA	Single Page Application
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
VCRI	Verification and Cross Reference Index
CI/CD	Continuous Integration/Continuous Deployment



Acknowledgments

I wish to express my gratitude to my supervisor at CERN, Maria Dimou, for her continuous support and motivation throughout my time at CERN, especially at times when the project's difficulty seemed to be overwhelming. Furthermore, I would like to acknowledge the technical assistance and advice given by several IT-CDA and RCS-SIS webmasters at CERN, especially Harris Tzovanakis and Salome Rohr. My last and greatest thanks go to Professor Kasper Knop Rasmussen for offering to supervise my BSc thesis and helping me obtain my bachelor's degree. Their assistance was critical to the success of this project.

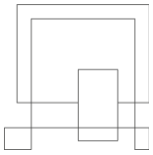
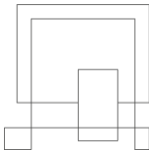


Table of content

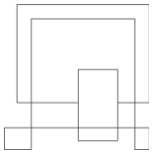
Abbreviations – Acronyms
Acknowledgments
List of figures and tables
Abstract
1 Introduction 1
2 Analysis 3
2.1 Actor Descriptions 3
2.2 Requirements 3
2.2.1 Functional Requirements 3
2.2.2 Non-Functional Requirements 5
2.3 Use Case Diagram 7
2.4 Use Case Descriptions 8
2.5 Domain Model 14
3 Design 15
3.1 Architecture 15
3.1.1 Harvest 16
3.1.2 Backend 17
3.1.3 UI 18
3.2 Technologies 19
3.2.1 Technology alternatives 19
3.3 Entity-Relation (ER) Diagram 21
3.4 Tools 21
3.5 Package Manager 22
3.6 REST API 22
3.7 Cloud Services 23
3.8 Continuous Integration/Continuous Deployment (CI/CD) pipeline 23
3.9 Container Orchestration 23
3.10 UI Design Choices 24
3.11 Frontend Navigation 24
4 Implementation 25
4.1 View lecture functionality 25
4.1.1 View video lecture 25
4.1.2 View non-video lecture 26
4.2 Search lectures functionality 27
4.2.1 Help for search 29



4.2.2 View search results 29
4.2.3 View number of results 30
4.2.4 Sort results by relevance 30
4.2.5 No results 31
4.3 View ATC members functionality 32
4.4 Contact ATC functionality 32
4.4.1 On the header 33
4.4.2 On the footer 33
4.4.3 In the Suggestion Box 34
4.5 Harvesting 34
4.5.1 OAI-PMH mechanism for MARCXML translation 34
4.5.2 CDS Spider 35
4.5.3 XML-JSON conversion 36
4.6 Maintain the database 36
4.6.1 Fields in the database 38
4.6.2 Bleaching the abstract 38
5 Test 39
6 Results and Discussion 40
7 Conclusions 40
8 Project future 41
Sources of information
Appendices

List of figures and tables

Figure 1: Rich Picture – The overall structure of the project
Figure 2: Use Case Diagram
Figure 3: Use Case Scenario 1 – View lecture
Figure 4: Use Case Scenario 2 – Search lectures
Figure 5: Use Case Scenario 3 – View ATC members
Figure 6: Use Case Scenario 4 – View events
Figure 7: Use Case Scenario 5 – Contact ATC
Figure 8: Use Case Scenario 6 – Manage harvesting
Figure 9: Use Case Scenario 7 – Maintain database
Figure 10: Use Case Scenario 8 – Manage navigation
Figure 11: Use Case Scenario 9 – Maintain ATC members
Figure 12: Domain model
Figure 13: Architecture
Figure 14: Class diagram of the Harvest service
Figure 15: Lecture model - Partial class diagram from the Backend service
Figure 16: LectureDocumentViewSet - Partial class diagram from the Backend service
Figure 17: Conceptual class diagram of the UI service
Figure 18: Sequence Diagram
Figure 19: Technology alternatives table
Figure 20: ER diagram
Figure 21: Partial class diagram from the Backend service
Figure 22: Partial class diagram from UI
Figure 23: API Test Case
Figure 24: CDS Test Case



Abstract

You will be guided through the structure of the team, the project requirements, and how these were fulfilled through careful analysis. Furthermore, it will get into technical details about how it was designed and implemented, and tested during the process. This report will also summarize the project's results, conclusions, and future plans.

1 Introduction

CERN's Academic Training Lecture series include recordings and/or downloadable files of material allowing CERN users or interested parties to follow up these lectures through an online platform. The lectures are open to all members of CERN personnel (in particular staff members and fellows, associates, students, users, project associates and apprentices) free of charge. The complete catalogue of Academic Training lectures is **archived since 1968**, however, not all of them have videos.

This project is about **making the lecture views and searches more attractive**.

In this project, the *IT developer* (Franciska Török) and *RSC maintainer* (Harris Tzovanakis and his team) relied heavily on the *Project Initiator* (Maria Dimou), who collected the main requirements from the members of the *ATC*. During the process, several ATC meetings were held with the stakeholders (ATC) [1] and separate meetings between IT and RSC for the development of the website.

CDS [2] and Indico [3] are used to power the Academic Training website, which targets all CERN personnel, whose names are below referred to as *CERN Users*.

In the following rich picture, the solid arrows show direct impact while the dashed lines show dependency.

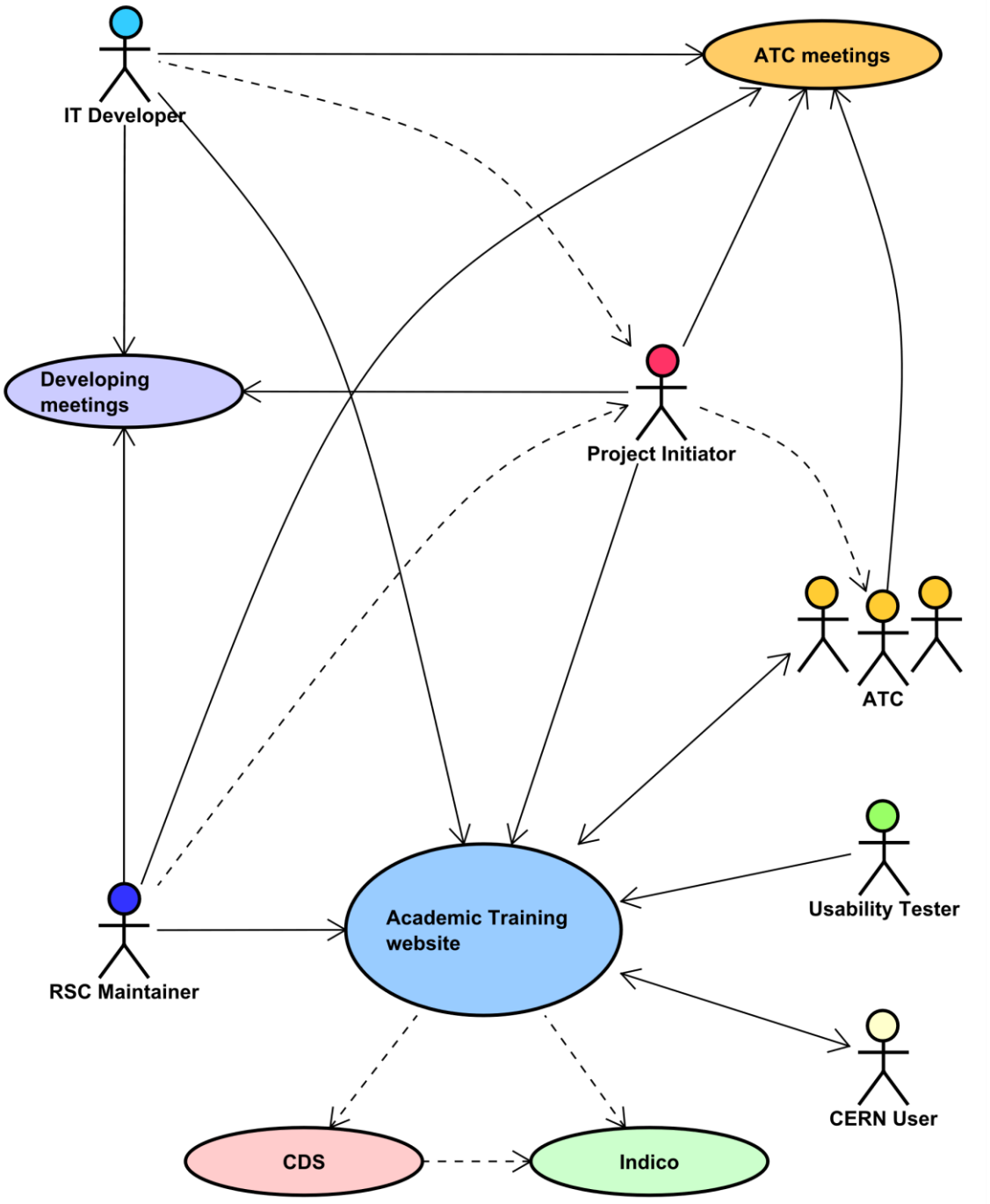
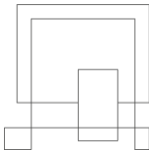


Figure 1: Rich Picture – The overall structure of the project



2 Analysis

Several requirements have been outlined by the ATC members for this project, which will be discussed in more detail below.

2.1 Actor Descriptions

Different types of users will have access to different features and functionality as described in the actor descriptions.

User – User is an actor who uses the platform and performs several actions like watching the video lectures, searching for lectures, accessing lecture files, having the ability to contact the ATC, etc. The user does not have to be registered or logged in in order to perform these actions.

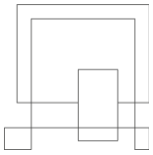
Administrator – The Administrator is the maintainer of the platform who makes sure that the site works as expected and takes responsibility for any changes that arise, such as the need to harvest new lectures or to change members of the ATC.

2.2 Requirements

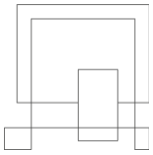
The following requirements are based on the problem defined in the Project Description [4], mandatory elements from CERN's design guidelines [5] and general regulations for the CERN Logo usage [6].

2.2.1 Functional Requirements

1. As a user, I want to see the lectures displayed by *thumbnail picture*, *speaker* and *date*, so I can watch the lecture that I want.
2. As a user, I want a search bar, so I can search lectures by *title, speaker, date, abstract, sponsor,* and *keywords*.
3. As a user, I want to see the search bar permanently, so I can search for a lecture whenever I want.



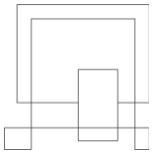
4. As a user, I want the lectures displayed chronologically on the *Main* page and on the *Search* result page, so that I can see the most recent first.
5. As a user, I want a dropdown menu of *Relevance* that sorts the search results by score (*Most relevant*), by date (*Newest first* and *Oldest first*), so I can sort in the order I want to see them.
6. As a user, I want to see the number of results after each search, so I know how many lectures were found for my search term.
7. As a user, I want to see a message displayed when no results are found after a search, so that I know, I must change the search term, and look for something else.
8. As a user, I want a *Contact* button that navigates to the email address of the stakeholders (ATC members), so I can contact them, give feedback or send a suggestion for future lectures.
9. As a user, I want an *Events* button, so that I can access and see the Full programme of the Academic Training Lectures (Indico [3]).
10. As a user, I want a button that navigates to an *About* page, so I can see who the members of the ATC are.
11. As a user, I want the *title, speaker, date, event details, abstract* (description of the lecture), *duration,* and *sponsor* displayed for each lecture, so I can see the details of it.
12. As a user, I want a permanent header on every page top to find the *About, Contact,* and *Events* links, so I do not have to search or scroll for them for too long.
13. As a user, I want a *Help* tooltip next to the search bar, so I know how to use the search engine the most efficient way as possible.
14. As a user, I want to be able to force exact search matches, so I can find the precise lecture that I want.
15. As a user, I want the software to be able to display also *two-channel lectures*, so I can see one for the recording of the speaker and one for the slides used to illustrate the lecture.
16. As a user, I want a *list of files* displayed on the page of a lecture, if video is not present, so I can access and view non-video lectures as well.
17. As an administrator, I want to be able to harvest new lectures from the Academic Training Lecture series, so I can keep the software up to date.



18. As an administrator, I want to be warned if the harvesting fails, so I can avoid risks of the site being out of date.
19. As an administrator, I want to be able to include additional details from the event of the lectures (Indico), if it is not present already in the CDS, so I do not miss details that are needed by the stakeholders.
20. As an administrator, I want to be able to update the ATC members, so I can display up-to-date information about the members and the Committee mission.

2.2.2 Non-Functional Requirements

1. The software must be publicly accessible on the internet.
2. There must be a logo specific to THIS software that gives the brand of the ATC.
3. The Academic Training lectures must be accessed separately from the collection on CDS.
4. The software must display mixed results. While the majority of the lectures are recorded and stored as videos, the old records are archived in PDFs, so they must also be managed.
5. The lecture harvesting should be automatic, performant (compare video loading speed on the site and in CDS; ensure no overhead appears due to the site views/scripts), and persistent to eventual CDS changes.
6. The software must use cold colors (ex. blue, black, white), as this is the general color scheme of CERN [7].
7. The site views, links and navigation must be user-friendly.
8. CERN is bilingual (French and/or English). This software must display views in English because all the lectures are in English.
9. The software must belong to the domain name *.cern.ch*.
10. The software must be designed responsive, so that it can be viewed on smaller screens too, like on tablet and/or mobile devices. Since it is not possible to test it on every single device in the world, it is important to have responsive layout on some of the most popular devices (like iPhone 8, Samsung Galaxy S8, Google Pixel and Nexus 7).



11. The software must be usable from multiple browsers: Google Chrome, Firefox, Samsung Internet, Safari, Microsoft Edge, and Opera.
12. In order to ensure future maintainability, the software must make use of technologies in which CERN and the community have expertise.
13. The CERN toolbar must be displayed at the top of the software in the standard way, including a link to the homepage of CERN [8], to Sign In and to the CERN Directory.
14. The CERN logo [6] should always be used in the footer and be an active link to the homepage of CERN.
15. The CERN logo should *never* be used at the top left corner or in the header.
16. The CERN logo should *not* be animated or have states or scripted interactions.
17. The CERN logo should be *no* less than 60px wide, and its correct proportions must be respected.
18. A descriptive documentation site must be created and written for future maintenance.

Note: Regulations for optional CERN elements are not stated here.

2.3 Use Case Diagram

The following use case diagram represents use cases based on the requirements above.

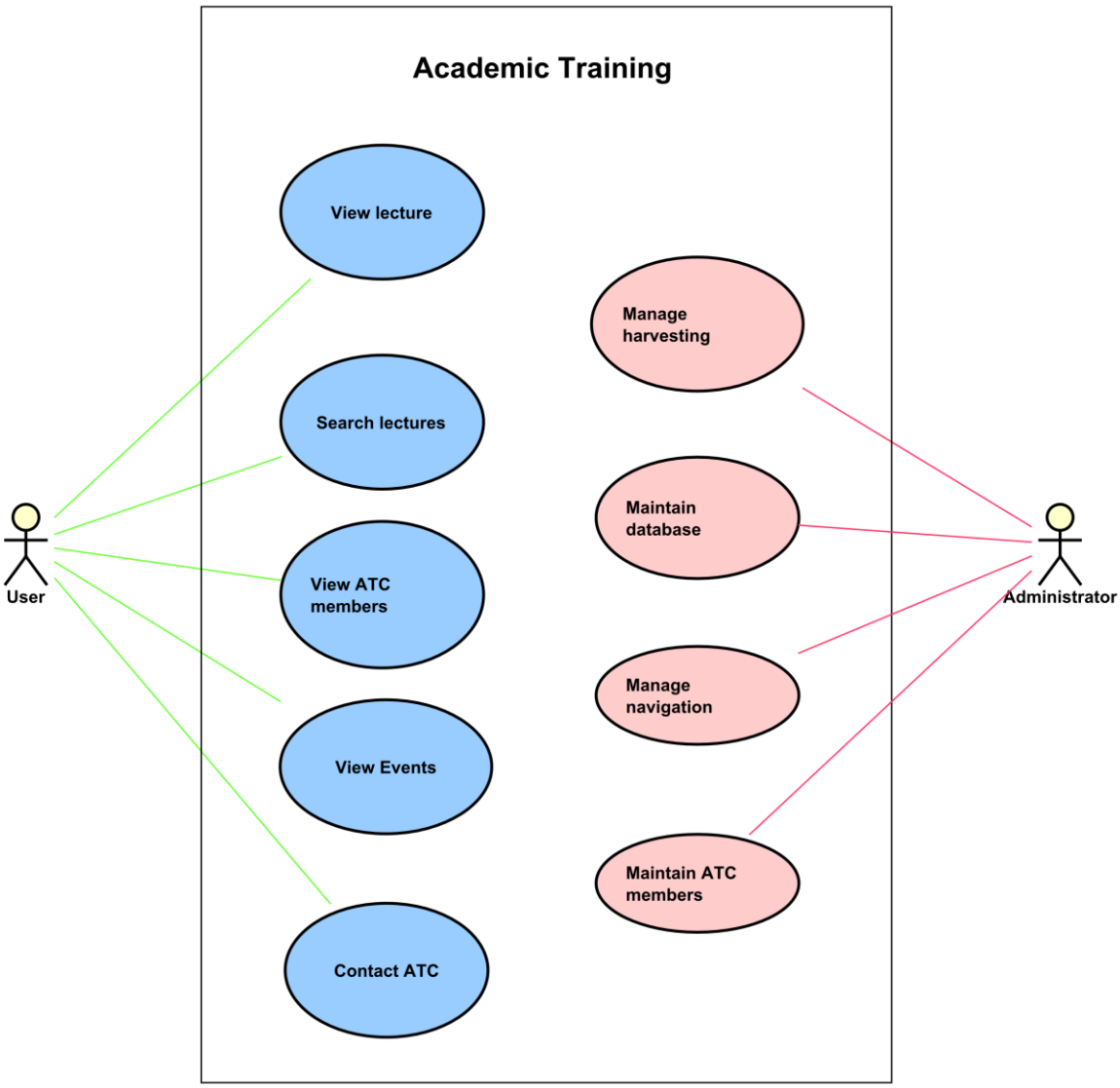
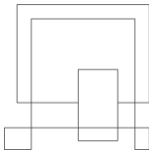


Figure 2: Use Case Diagram

Note: For more visible Use Case diagram visit Appendix D.

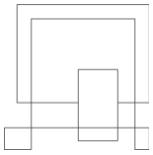


2.4 Use Case Descriptions

Use case	View lecture
Summary	View lecture and its details (title, speaker, date, abstract, event details, sponsor, keywords)
Actor	User
Precondition	● The lecture event exists in Indico ● The lecture has taken place ● The recording has been uploaded to CDS ● The harvesting from CDS was successful ● The connection to Indico API was successful
Postcondition	Harvested from CDS
Base sequence	1. Click on a lecture to watch 2. System talks through HTTP request to the REST API to ask for the called lecture 3. System checks if the lecture exists in the database 4. System sends back an HTTP response 5. System checks if the lecture is a single video, a two-channel video, has files or none 6. System displays the lecture with its details accordingly
Note	If CDS is down, the videos and files cannot be reached (!) – an error message will be displayed that will originate from CDS. Everything else has a copy in the system's separate database (title, speaker, date, abstract, event details, sponsor etc.) that are not dependent on CDS and can be displayed regardless.

Figure 3: Use Case Scenario 1 – View lecture

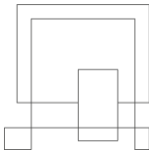
Use case	Search lectures
Summary	Search lectures and/or browse from search results
Actor	User
Precondition	Lectures exist in CDS



Postcondition	Harvested from CDS
Base sequence	1. Click on the search bar to search 2. Fill the search bar with content 3. Hit Enter OR click on the Search Icon 4. System checks if the search value in the Search bar has content a) If not, the system generates an empty search and displays all the lectures available b) If yes, go to step 5 5. System sets the search term with the searched value 6. System navigates to the search route after the query and displays the first page of the results
Note	- Empty search lists all the available lectures. - A guide (?) to help how to search is available next to the search bar: *You can search by title, speaker, date, abstract, sponsor and keywords.* *The default search operator is AND.* *For an OR search, use | between words:* *ex. particle | physics* *For exact match, use double quotes, like this:* *"dark matter"*

Figure 4: Use Case Scenario 2 – Search lectures

Use case	View ATC members
Summary	View the members of the Academic Training Committee (ATC) and their mission
Actor	User
Precondition	Maintainer has the latest ATC members deployed
Postcondition	-
Base sequence	1. Click on the 'About' button to view the members of the ATC a) In the header b) In the footer



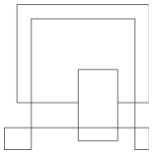
	2. System navigates to the site of the 'About Us' that displays the description of the ATC and the current members of it
Note	-

Figure 5: Use Case Scenario 3 – View ATC members

Use case	View events
Summary	Navigate to and view the full programme of the CERN Academic Training Lecture series
Actor	User
Precondition	Indico category 72 is accessible (Academic Training Lecture Regular Programme)
Postcondition	-
Base sequence	1. Click on the 'Events' button to view the full programme of the CERN Academic Training Lecture series a) In the header b) In the footer 2. System navigates to Indico
Note	If Indico is down, the full programme cannot be reached. However, the rest of the system still can be used.

Figure 6: Use Case Scenario 4 – View events

Use case	Contact ATC
Summary	Contact the ATC through a dedicated email address
Actor	User
Precondition	The *atc-contact@cern.ch* must be configured in the CERN e-groups
Postcondition	-
Base sequence	1. Contact the ATC members either: a) Through the 'Contact' button on the header/footer OR b) 'Submit a suggestion for future topics' button

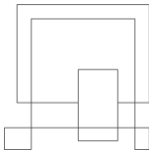


	2. System opens the default email agent with the email address to the ATC
Note	Cancel can be done anytime.

Figure 7: Use Case Scenario 5 – Contact ATC

Use case	Manage harvesting
Summary	Harvest lectures from CDS
Actor	Administrator
Precondition	Lectures exist in CDS
Postcondition	OAI-PMH mechanism for MARCXML translation
Base sequence	1. Run the backend 2. Login in the database with admin credentials 3. Add lecture 4. Migrate lectures a) in case of harvest failure, send an email to the Academic Training site admins' e-group
Note	- The lecture's details are retrieved from a MARCXML file that needs conversion before harvesting the records from CDS to the database. This is necessary because MARCXML has data fields with unique numbers that correspond to a tag, and subfields that correspond to a character. Since it is not easily readable, it must be translated. That is the OAI-PMH mechanism (translation from XML to JSON). - The information that does not exist on CDS is obtained from the lecture's Indico event (Indico API).

Figure 8: Use Case Scenario 6 – Manage harvesting

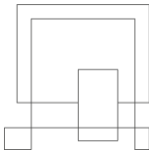


Use case	Maintain database
Summary	Manage database once a change occurs or update is required
Actor	Administrator
Precondition	Lectures harvested
Postcondition	Admin logged in, git repository cloned
Base sequence	1. Run the backend 2. Log in to the database with admin credentials 3. Add/modify lecture fields 4. Migrate lectures a) In case of harvest failure, send an email to the Academic Training site admins' e-group
Note	The abstract must be cleansed because it might have features that are not needed (like HTML tags or CSS attributes).

Figure 9: Use Case Scenario 7 – Maintain database

Use case	Manage navigation
Summary	Manage navigation of the project to render the app at different URLs
Actor	Administrator
Precondition	-
Postcondition	Admin logged in, git repository cloned
Base sequence	1. Add path(s) / URLs 2. Define the route(s) of the path(s) 3. Add outlet to path
Note	A default blank page is added in case of a route that has no endpoint (it leads nowhere).

Figure 10: Use Case Scenario 8 – Manage navigation



Use case	Maintain ATC members
Summary	Maintain the ATC members on the About Us page, update members in case of change
Actor	Administrator
Precondition	The ATC chairperson informs the Administrator
Postcondition	About Us page exists
Base sequence	1. Take the list of ATC members 2. Update or add a new member by the following: a) key b) name c) profile d) position e) department 3. Update About Us page 4. Save changes
Note	The profile pictures of the members must be downloaded and added to the project repository.

Figure 11: Use Case Scenario 9 – Maintain ATC members

2.5 Domain Model

The Domain model presented below describes the Domain concepts and their relations. Users can view the ATC members, view lectures and/or search for these lectures, however, the Administrator can edit the ATC members, harvest these lectures and/or modify their content. The lectures are being harvested from CDS and some additional data is extracted from the lecture's Indico event (such as sponsor and keywords).

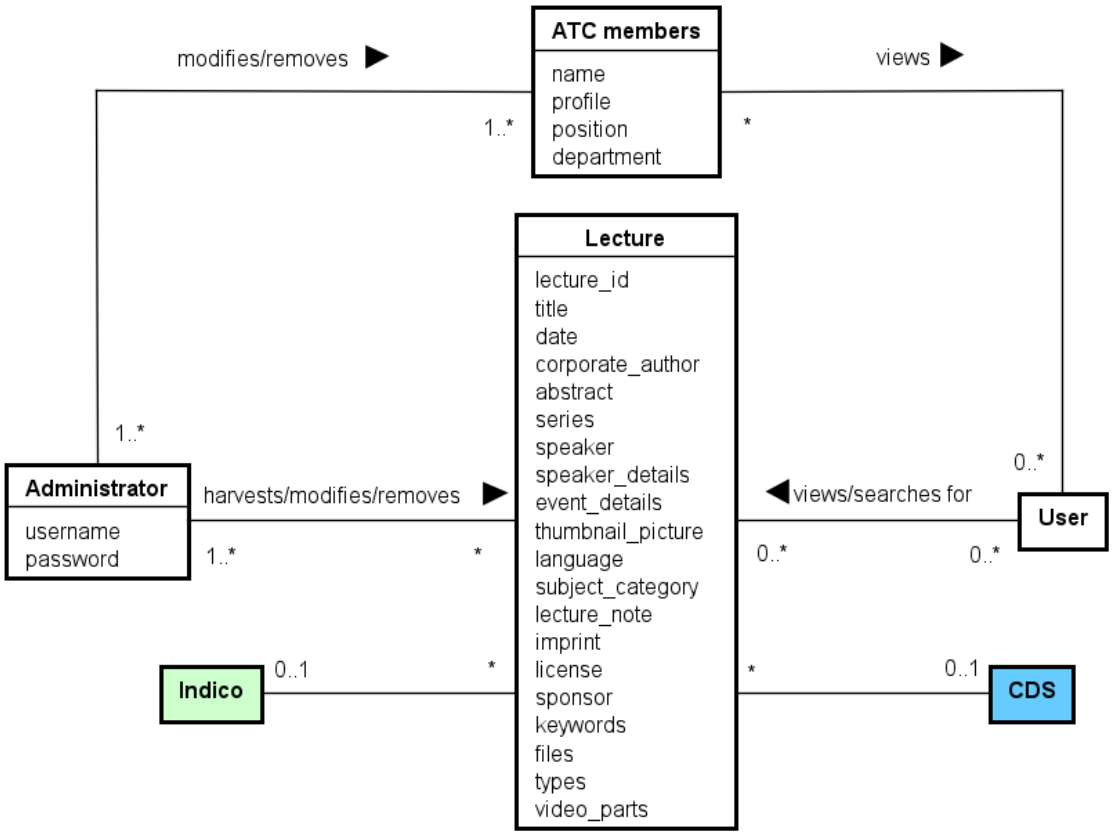


Figure 12: Domain model

Note: For a more visible Domain Model visit Appendix D.

3 Design

The system's main goal is to serve the users with Academic Training lectures, that are easily accessible and pleasant to watch from a user-friendly site.

3.1 Architecture

This chapter will navigate through the architecture of the project.

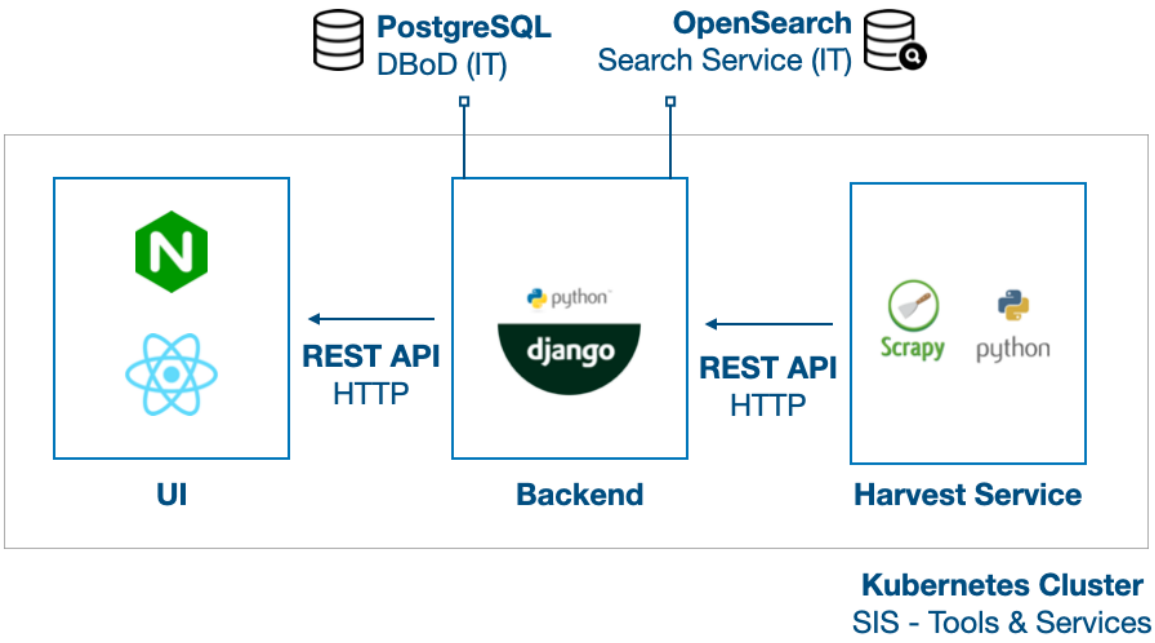


Figure 13: Architecture

CERN SIS - Tools & Services uses its general **Kubernetes** [9] **Cluster** for the Architecture. As can be seen above, the whole architecture consists of **three main services**. These three main services communicate through *HTTP requests* maintained by **REST API** that sets in the Backend.

3.1.1 Harvest

Initially, the CERN Academic Training lectures are collected from CDS. In order to harvest records from CDS, the OAI-PMH mechanism has been used.

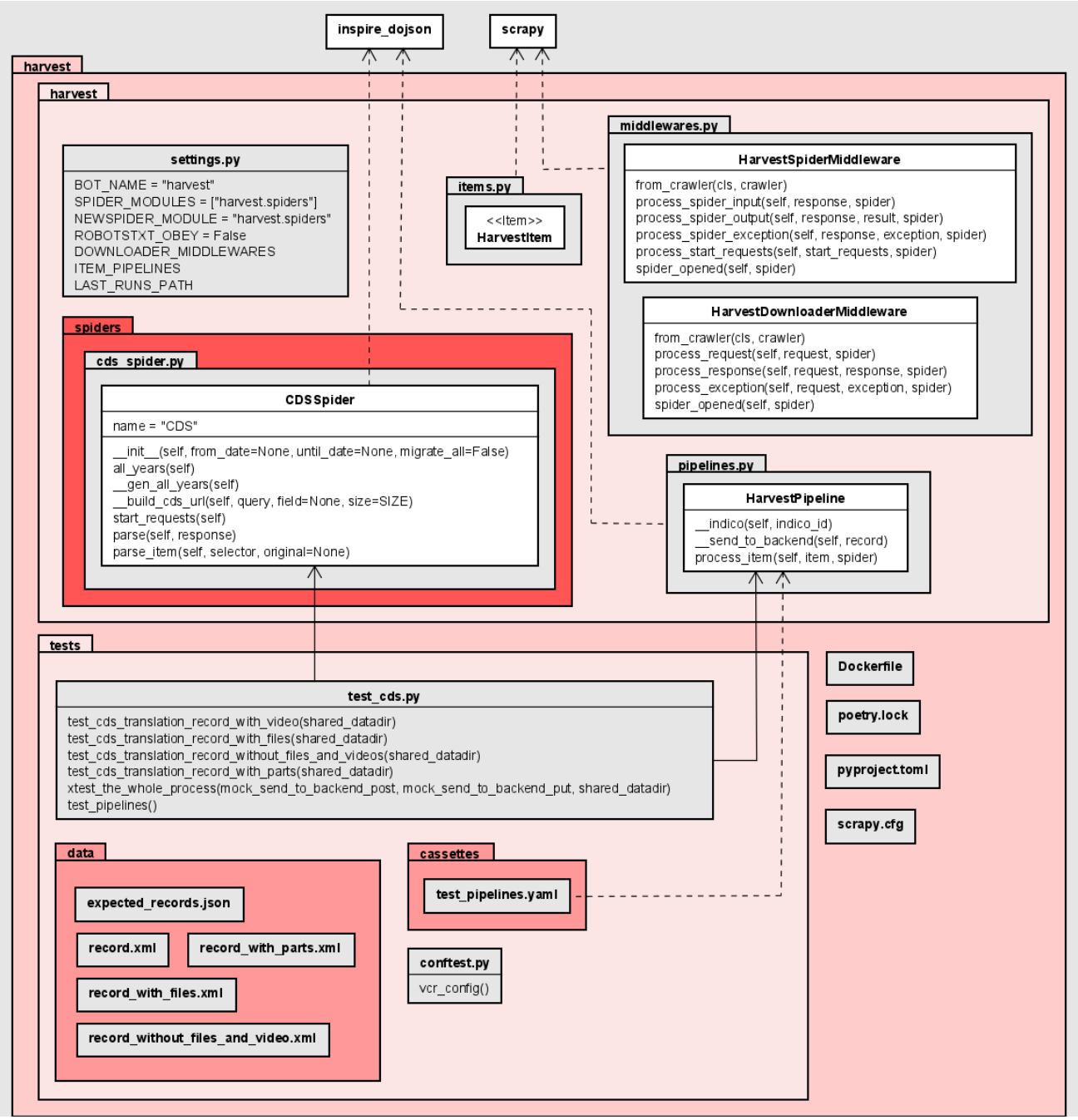
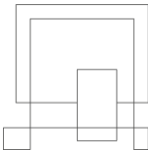


Figure 14: Class diagram of the Harvest service



An already existing harvesting method was reused from **INSPIRE-DoJSON** [10] which contains already the data transformation from XML to JSON format. While ***CDSSpider*** collects all the necessary lectures from CDS from all available years, it also ensures that data conversion happens. The Backend's REST API creates new entries once the harvest has been successful. The CDS harvesting procedure runs every week to obtain the latest records.

3.1.2 Backend

The lectures that are being harvested from CDS are stored in a **PostgreSQL** [11] database (**DBoD** [12]). The lecture fields for the database are established in the Lecture model seen in *backend/cds/models.py*. Another service provides the **Search** of the website (**OpenSearch**) [13].

Note: Some important data are not present in CDS. They are retrieved from the **Indico API**. We will get back to the reasons and details later.

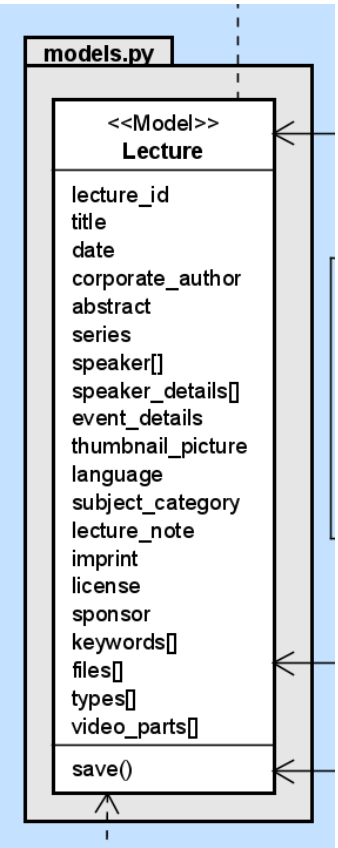


Figure 15: Lecture model - Partial class diagram from the Backend service

Any other rules applied for manipulating the lectures, like filtering, ordering, and prioritizing search results are all added in ***LectureDocumentView*** in *backend/cds/views.py.*

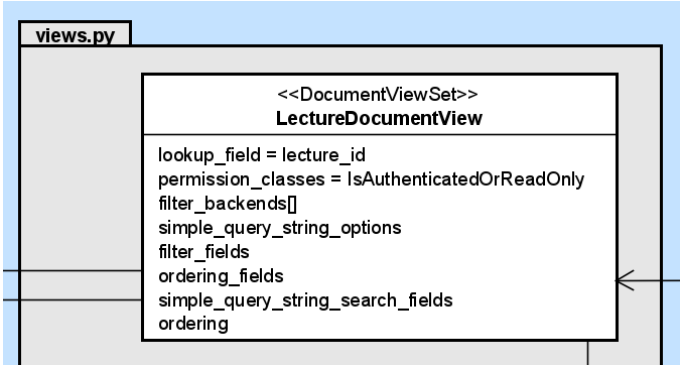


Figure 16: LectureDocumentViewSet - Partial class diagram from the Backend service

3.1.3 UI

React manages the frontend part of the site and its aesthetic design as the main provider of the **UI/UX**. To facilitate long-term maintenance, the **Ant Design** [14] (antd) components were chosen for the UI framework, for which expertise at CERN already exists. The yellow-labeled components seen below are displayed on every screen of the website.

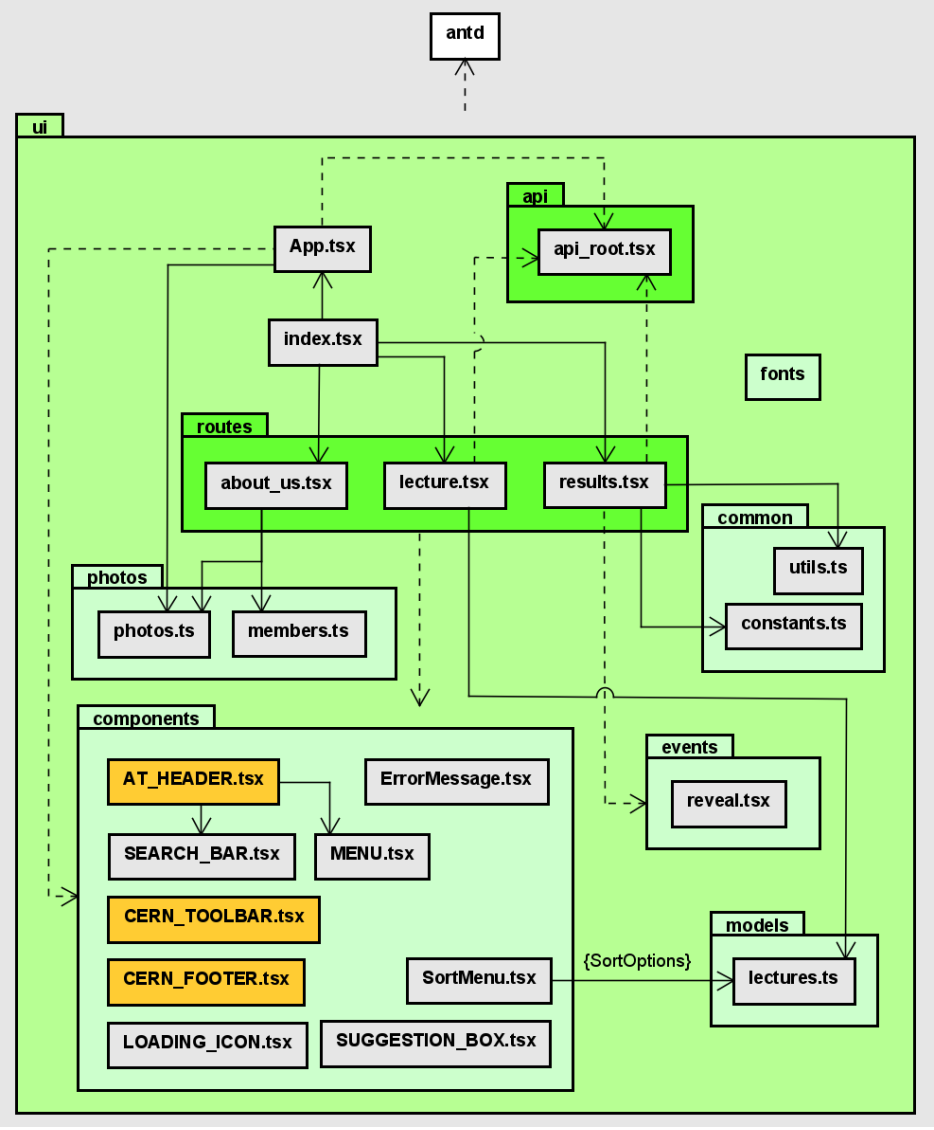


Figure 17: Conceptual class diagram of the UI service

Note: For a more visible Class Diagram visit Appendix D.

3.2 Sequence Diagram

The Sequence Diagram illustrates how data flows between the ***ui***, ***backend***, and ***harvest*** services, involving external servers ***CDS*** and ***Indico***. The **harvest** service collects records from all years by building a *CDS_URL*. It sends requests to CDS, parses records, and simultaneously requests additional data from Indico (sponsor, keywords). The *send_to_backend()* function handles pushing records to the **backend**'s API, either updating existing ones with a PUT request or creating new ones with a POST request. The **ui** can access the data through the API's base URL.

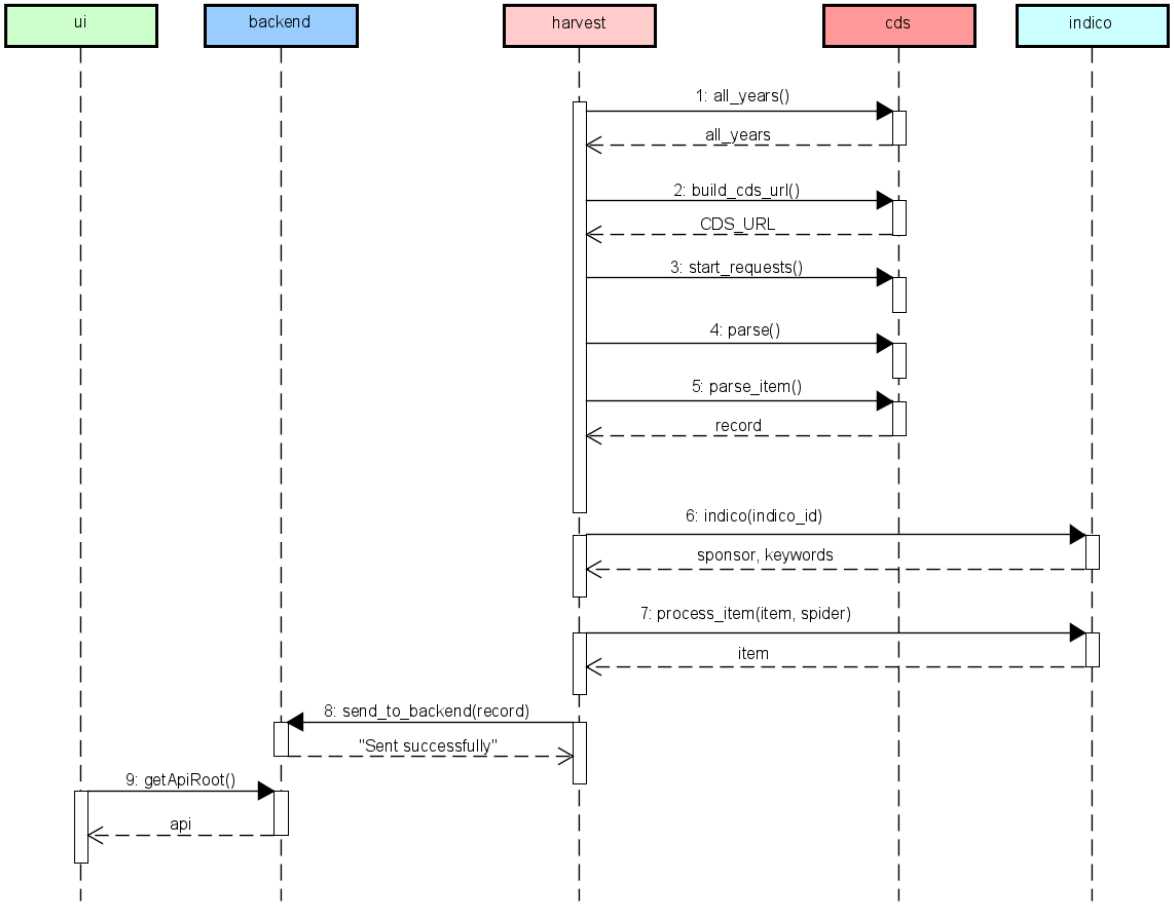
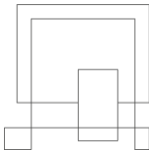


Figure 18: Sequence Diagram



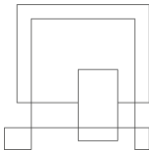
3.3 Technologies

- **Harvest:** Built by **Scrapy** [15] and **Python** [16].
- **Backend:** Built by **Django REST framework** [17] with **Python**.
- **UI:** Built by **React.js** [18] client-side rendered SPA and **Node.js** [19]. Create React App (CRA) toolchain made it easy to set up the React project with minimal configuration by offering a pre-configured development environment with essential features like development server, hot module reloading, and optimized production builds.

3.3.1 Technology alternatives

At the very beginning of the project, we have evaluated the technologies used at CERN in order to defend the solution we would eventually conclude on.

Frontend	Backend	Advantages	Disadvantages
Drupal [20]		Easier maintenance, deployment, authentication, data storage provided by CERN	Only a few lines of code would have been written which would have provided minimal educational value, since most things are already implemented + not much to write in the thesis + detrimental for long-term personal career
React	Django	Provides preferable educational value, the thesis can be written properly + technologies are advantageous for long-term personal	After finishing the Technical Studentship contract at CERN, the project must be maintained by someone else + it is custom (must be built from scratch) and it is more complicated



		career on the developer's side	
WordPress [21]		It is fairly simple for non-experienced developers to create sites	It is not supported at CERN; it is not sophisticated enough for the harvesting
Invenio [22]		It is a CERN hosted application, there are experts at CERN	It is in early development and has the same situation as Drupal

Figure 19: Technology alternatives table

For technology alternatives, **Drupal** was one of the main suggestions for creating this platform, due to its popular use across CERN websites. Drupal is a content-management framework that offers customization and flexibility. It helps to modify websites and content. The home website of CERN is also made by Drupal. CERN has a themed template that can be customized to the developer's choice.

While Drupal would have been easier to maintain, deploy, authenticate, and store data, the development interest would be minimal. Only a few lines of PHP code would have been needed, leading only to a lesser technical experience gained during the project. Since most technical aspects are already implemented, there would have been not much to do, and it would have had minimal educational value. Moreover, the maintainers' team has experience with the technology eventually used, so they adopted the chosen solution, which was **React-Django**. The other options of the table were less considered, and the reasons are on the table itself. The reason why Drupal was extensively evaluated is that there are hundreds of CERN websites built with Drupal.

3.4 Entity-Relation (ER) Diagram

The goal of an ER is to provide an overview of the data that will be managed in the database, as well as reflect the properties and constraints of the database schema.

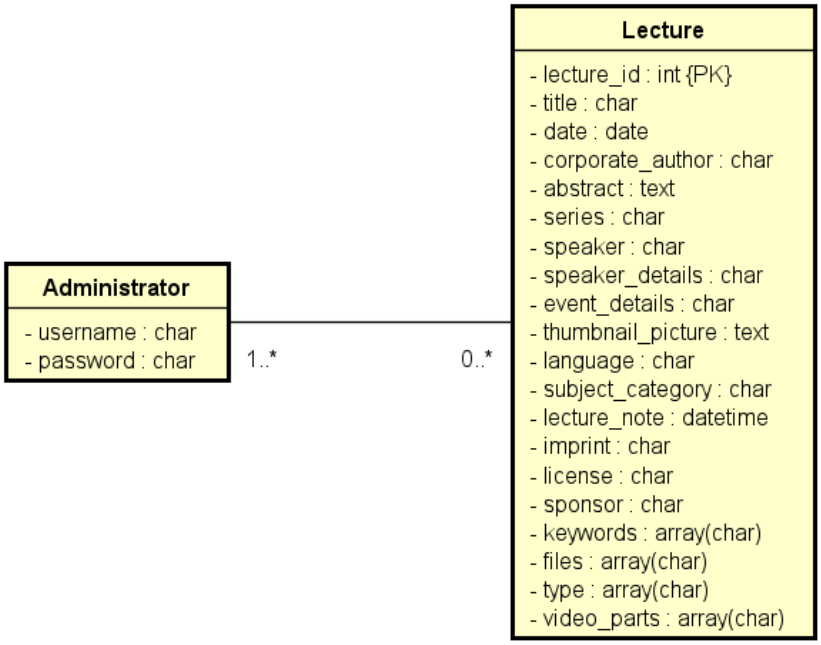


Figure 20: ER diagram

3.5 Tools

Next to the tools that have been used will indicate their purpose in this project:

- **Adobe XD** ® – UI/UX design
- **GitHub** – managing git repositories, team collaboration and code development
- **GitLab** – site documentation
- **Node.js** – running a web server for React app
- **Visual Studio Code** – writing code
- **JIRA** – project management, backlog, and workflow tracking
- **Docker** – packaging the application and dependencies into a container
- **Stack Overflow** – debugging and looking up solutions for bugs

3.6 Package Manager

For this project the used package manager has fallen to ***yarn***[23], which is a software packaging system developed for Node.js JavaScript runtime environment that provides speed, consistency, stability, and security as an alternative to ***npm***[24]. There was no specific reason to pick one over the other.

3.7 REST API

As there was no existing API to send requests for retrieving Academic Training lectures, a new REST API had to be built from scratch in the backend with the help of the Django REST framework. Django app configures the development server with URL paths to the '*/admin*' site, the *'/api/v1'* for accessing the API, and the '*/api-auth'* that authenticates the user with a unique token to the REST framework. These URL paths are defined in *urlpatterns[]* in the *backend/backend/urls.py* file, which collects all the view sets necessary for navigating within our REST API, like LectureViewSet, LectureDocumentView, and UserViewSet (containing only the admin).

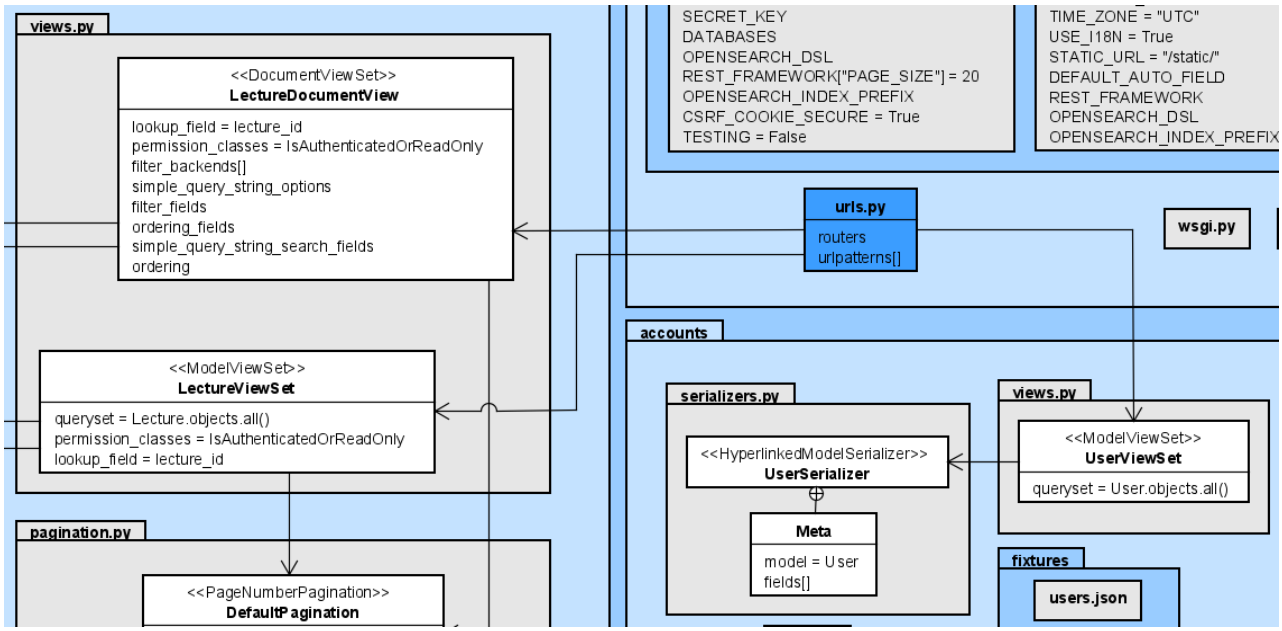
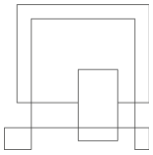


Figure 21: Partial class diagram from the Backend service



3.8 Cloud Services

CERN has its own public cloud services, one of them is OpenShift [25] which was used as Deployment workflow for the site's documentation. As soon as the project documentation [26] was ready, it had to be deployed using a GitLab Registry Image on OpenShift.

3.9 Continuous Integration/Continuous Deployment (CI/CD) pipeline

The GitHub Actions workflow, defined in *'.github/workflows/**test-and-build.yml'***, provides automation every time the code changes. The YAML file is triggered on push or pull requests to the master branch. It orchestrates three parallel jobs that checkout the code, build **Docker** [27] images, run tests for the 'backend', 'ui', and 'harvest' services respectively, and push the Docker images to ***registry.cern.ch*** registry. If the build and test jobs are successful, it then deploys the services to CERN SIS's **Kubernetes** cluster.

3.10 Container Orchestration

This software can be delivered more quickly using **Docker** as it separates applications from their infrastructure. This platform provides the management of the **container lifecycles**, which are the following:

- Developing the application and its supporting components using containers.
- Transforming the container to the unit that distributes and tests the application.
- Deployment of the application into the production environment, as a container.

This application requires some services to run, such as the database and Elasticsearch. The following files are configured in **Docker** and **Docker-Compose** to run these services cross-platform and conveniently: **Dockerfile** builds a fully functional image of the application with all of the static assets it requires, and **docker-compose.yml** that contains and exposes locally the minimal set of service containers needed for developing the instance locally: 'db' (PostgreSQL database) and 'es' (Elasticsearch). When developing and running the instances locally these services can be accessed by the application.

3.11 UI Design Choices

As a direct relation to the UI, only User use cases will be presented. Please find the UI design choices in **Appendix I**.

3.12 Frontend Navigation

The navigation of the project has been accomplished with **React Router** [28] in *index.tsx* which is a client and server-side routing library for React, a JS library for building user interfaces. The used routes have the purpose to lead us to separate URLs of our website, like the main page (*App.tsx*), the detail page of a lecture (*lecture.tsx*), the search results (*results.tsx*), and the page that displays the members of the ATC (*about_us.tsx*). The routes that are needed for the site are defined with their paths the following way:

- **Main page** – "/"
- **Search results page** – "/search"
- **About Us page** – "/about-us"
- **Collection of lectures** – "/lectures"
- **Lecture page** – "/lectures/:lectureId"
- **Blank page** (in case of a route that has no endpoint & leads nowhere) – "/*"

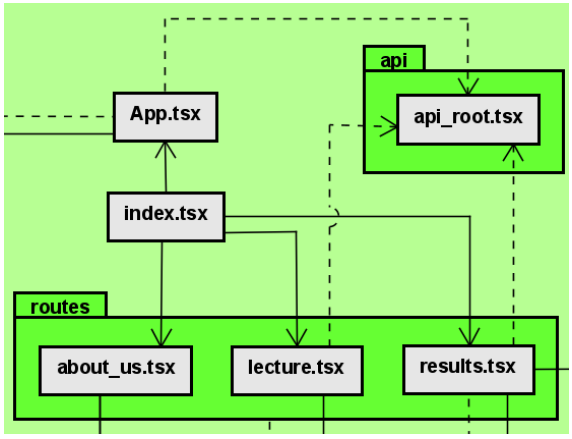
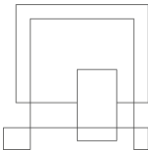


Figure 22: Partial class diagram from UI



4 Implementation

4.1 View lecture functionality

First of all, we establish a connection with the API using **Axios** [29].

/ui/src/api/api_root.tsx

```
const api = axios.create({ baseURL: `/api/v1/` });

export function getApiRoot() { return api; }
```

The code below fetches the lectures from the API root.

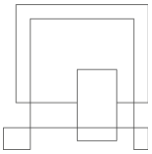
/ui/src/routes/lecture.tsx

```
const fetchLecture = async () => {
  (...)
  const results = await getApiRoot().get(`/lectures/${lectureId}/`);
  (...)
};
```

4.1.1 View video lecture

By selecting the lecture type `video`, I ensure that the video player does not display slides simultaneously with the video.

```
const isVideo = lecture.types && lecture.types.includes("video");
(...)
const displayVideo = isVideo && !displaySlidePlayer;
const displayParts = isParts && lecture.video_parts.length > 0;
```



If it is a single video, it will be displayed in the following manner:

```
{displayVideo && !displayParts && (
    <div className="video-window">
        <iframe
          title={lecture.title}
          src={`https://cds.cern.ch/video/${lectureId}?showTitle=true`}
          allowFullScreen
        />
    </div>
)}
```

4.1.1.1 View two-channel video

It takes the type `slide`. This one requires the `year` and `indicoId` parameters to function properly. The `year` is taken from the `date` while the `indicoId` is taken from the Indico event of the lecture (`event_details`).

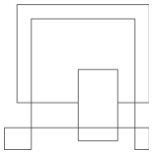
```
const isSlide = lecture.types && lecture.types.includes("slide");
(...)
if (isSlide && lecture.date) { year = lecture.date.slice(0, 4); }
if (isSlide && lecture.event_details) { indicoId = lecture.event_details.split("/")[4]; }
const displaySlidePlayer = year && indicoId;
```

If the video has two channels, it will be displayed in the following manner:

```
{displaySlidePlayer && (
    <div className="video-window">
        <iframe
            title={lecture.title}
            src={`https://mediastream.cern.ch/MediaArchive/Video/Public2/weblecture-
                    player/index.html?year=${year}&lecture=${indicoId}`}
            allowFullScreen
            scrolling="no"
            frameBorder="0"
        />
    </div>
)}
```

4.1.2 View non-video lecture

Some lectures do not have videos or files. These only display the remaining metadata, such as the date, title, speaker, and description (if available).



4.1.2.1 View list of files

This code checks if there are any files available and displays the title for downloading files along with a list of available files.

```
{lecture.files && lecture.files.length > 0 && (
   (...)
         <List
               itemLayout="horizontal"
               dataSource={lecture.files || []}
               split={false}
               renderItem={(item: string, index) => {
                 return (
                   <List.Item key={index}>
                     {index + 1}.{" "}
                         <a title={item} rel="noreferrer" target="_blank" href={item}>
                             {filenameFromUrl(item)}
                         </a>
                   </List.Item>
                 );
               }}
         />
  );
}
```

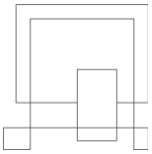
4.2 Search lectures functionality

Let's start with the backend. The backend plays one of the most important roles in search functionality. First of all, in this query set, all the lectures are being called.

/backend/cds/views.py

```
class LectureViewSet(viewsets.ModelViewSet):
    queryset = Lecture.objects.all()
```

The default search operator is set to **AND**, as well as the fields for the search are defined in the **filter_fields** parameter. “None” is an initial value. The **ordering_fields** are defined similarly for sorting purposes.



```python
class LectureDocumentView(DocumentViewSet):
    document = LectureDocument
    ...

    simple_query_string_options = { "default_operator": "and" }

    filter_fields = {
        "lecture_id": None,
        "types": None,
        "keywords": None,
        "series": None,
        "sponsor": None,
        "speaker": None,
        "subject_category": None,
    }
```

To sort the searched lectures, the 'boost' keyword assigns a score to the variables, ensuring that the most relevant lectures are displayed first. The lectures are ordered by their score and date.

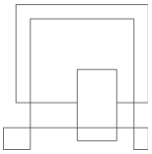
```
simple_query_string_search_fields = {
  "title": { "boost": 5 },
  "abstract": { "boost": 1 },
  ...
}

ordering = ( "_score", "-date" )
```

On the front end, when a key is pressed down, the *onKeyDown* function is triggered.

/ui/src/components/SEARCH_BAR.tsx

```tsx
const onKeyDown = (ev: any) => {
    const searchValue = ev.target.value;
    if (searchValue) {
      navigate(`/search/?search=${ev.target.value}&page=1`);
    } else {
      navigate("/search");
    }
};
```

If the *searchValue* is not empty (which means the user has entered something in the input field), the code utilizes the *navigate* function to redirect the user to a search page. The search



query and page number are included in the URL. The search query is obtained from *ev.target.value*. If the *searchValue* is empty (indicating that the user has cleared the input field), the code will navigate to the default search page.

4.2.1 Help for search

/ui/src/components/SEARCH_BAR.tsx

```
<Tooltip
      (...)
       {helpText}
      <QuestionCircleOutlined />
 </Tooltip>
```

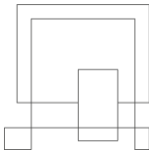
A tooltip has been added next to the search bar to help users learn how to search.

4.2.2 View search results

First, the searched lectures are fetched from the API root. *'searchLectures'* method performs an API request to search for lectures. It uses the *searchValue* variable as a query parameter and updates the *lectures* state with the response data. The function is cached using *useCallback* with *searchValue* as a dependency.

/ui/src/routes/results.tsx

```
const searchLectures = useCallback(async () => {
   ...
      const response = await getApiRoot().get(`/search/lectures/`, {
        params: {
          ...
          search_simple_query_string: searchValue,
          ...
        },
      });
      setLectures(response.data.results);
    ...
  }, [searchValue, ...])
```



4.2.3 View number of results

```
const searchLectures = useCallback(async () => {
  ...
      setTotal(response.data.count);
  ...
  }, [...]);
...
<Title>
   {total} Search {pluralizeUnlessSingle("result", total)}:{" "}
    ...
</Title>
```

When the search is performed, the *total* state gets updated with the response data being counted. The Title component is rendered and displays the total value of lectures next to the 'Search result' text. The *pluralizeUnlessSingle* function is used to handle pluralization based on the value of the total lectures, as it could also retrieve a single lecture during the search.

4.2.4 Sort results by relevance

The *ordering* parameter is set based on the value of the *order* variable. If the *order* is equal to *SortOptions.Default*, the ordering parameter is set to undefined, otherwise, it is set to the value of *order.*

```
const searchLectures = useCallback(async () => {
  ...
      const response = await getApiRoot().get(`/search/lectures/`, {
        params: {
          ordering: order === SortOptions.Default ? undefined : order,
        },
      });
  ...
}, [..., order]);
```

SortMenu component displays a dropdown menu for sorting.

```
<SortMenu sortMethod={order} handleChange={setOrder} />
```

The options for the dropdown menu are included in the SortMenu component.

/ui/src/components/SortMenu.tsx

```
<Select value={sortMethod} onChange={handleChange}>
       <Option value={SortOptions.Default} key={SortOptions.Default}>Most relevant</Option>
       <Option value={SortOptions.Newest} key={SortOptions.Newest}>Newest first</Option>
       <Option value={SortOptions.Oldest} key={SortOptions.Oldest}>Oldest first</Option>
</Select>
```

The lecture interface not only exports the lectures but the sort method type as well.

/ui/src/models/lectures.ts

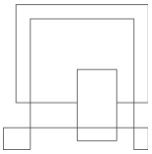
```
export enum SortOptions {
  Default = "relevance",
  Oldest = "date",
  Newest = "-date",
}
```

4.2.5 No results

/ui/src/routes/results.tsx

```
!lectures.length ? (
    <Empty className="empty" description="No results found" />{" "}
)
```

The *Empty* component displays an empty container in case there are no search results.



4.3 View ATC members functionality

Card components have been used for displaying the ATC members. The content of the card can be seen below.

/ui/src/routes/about_us.tsx

```
{MEMBERS["core"].map((member: any) => {
  return (
  ...
      <Title level={3}>{member.department}</Title>
      <Avatar size={120} src={member.profile.default} />
      <Title level={2}>{member.name}</Title>
      <p>{member.position}</p>
  ...
  );
})}
```

The ATC Members were separated by ‘core’, ‘departments’, ‘users’, ‘staff association’, and ‘observers’. They are defined in a separate JSON file as follows:

/ui/src/photos/members/members.ts

```
const MEMBERS: any = {
  core: [
    {
      key: 1,
      name: "Urs WIEDEMANN",
      profile: require("./profiles/Urs_Wiedemann.jpg"),
      position: "Chair",
      department: "TH",
    }, ...
]};
```

4.4 Contact ATC functionality

All of them use a *mailto* identifier that automatically opens the default email agent of the user.

4.4.1 On the header

/ui/src/components/MENU.tsx

```
<Menu>
      ...
      <Menu.Item className="contact-us" key="contact-us">
        <Typography.Link href="mailto:atc-contact@cern.ch" target="_blank">
          <Title level={2} className="contact-us-link">
            Contact
          </Title>
        </Typography.Link>
      </Menu.Item>
</Menu>
```

Then it is built into the header as a part of the Menu. It checks if *screenWidth* is greater than 992 pixels. If the condition is true, it renders a *MENU* component on bigger screens, otherwise, it renders the *Drawer* component, for example on mobile devices.

/ui/src/components/AT_HEADER.tsx

```
{screenWidth > 992 && <MENU />}

{screenWidth <= 992 && (
   <Drawer
       ...
       onClose={toggleCollapsed}
       visible={!state.collapsed}
       destroyOnClose={true}
       closeIcon={<CloseOutlined />}>
                 <MENU />
   </Drawer>
)}
```

4.4.2 On the footer

Similar to the one that is set in the header (See 4.4.1).

4.4.3 In the Suggestion Box

/components/SUGGESTION_BOX.tsx

```
<div className="suggestion reveal">
      <Typography.Link href="mailto:atc-contact@cern.ch" target="_blank">
            <Title className="hover-underline-animation">
                Submit a suggestion for future topics
            </Title>
      </Typography.Link>
</div>
```

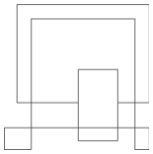
4.5 Harvesting

The process of harvesting occurs through a mechanism called **OAI-PMH**, which is used in our project to convert XML files into JSON format. The basic information for an Academic Training lecture record is obtained from CDS in MARCXML format.

4.5.1 OAI-PMH mechanism for MARCXML translation

There are some tags with numbers in the XML that each of which corresponds to a specific variable, like the title, speaker, date, and so on.

However, we cannot clearly see or understand them clearly from this metadata. Therefore, we will process this metadata by **translating our XML into meaningful data** that is easily readable. In this case, it is converted to **JSON**. This process is the **OAI-PMH** mechanism mentioned earlier, and it will be described below. This method is reused from INSPIRE [10].



4.5.2 CDS Spider

Lecture records are collected from CDS using *Scrapy*, which performs data collection and scraping logic. First, it builds the *CDS URL* with the query, size, and field. The default URL with these fields is included in the variable *CDS_URL*.

/harvest/harvest/spiders/cds_spider.py

```
def __build_cds_url(self, query, field=None, size=SIZE):
        if field is not None:
            return CDS_URL_WITH_FIELD.format(query=query, size=size, field=field)
        return CDS_URL.format(query=query, size=size)
```

It starts requests from CDS:

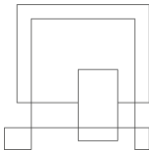
```
def start_requests(self):
        if self.migrate_all:
            self.all_years_gen = self.__gen_all_years
            item = next(self.all_years_gen)
            url = self.__build_cds_url(item["year"], item["field"])
        else:
            query = "->".join(self.query)
            url = self.__build_cds_url(query)
```

This part selects records, then tries to loop through them by getting the parsed records.

```
def parse(self, response):
        ...
        records = response.selector.xpath("//record")
        for record in records:
            ...
                yield self.parse_item(record, original=record.get())
            ...
```

To get all records from CDS, it migrates all records from all available years as follows:

```
try:
     if self.migrate_all and (item := next(self.all_years_gen)):
            LOGGER.debug("Harvesting next page", year=item["year"])
            url = self.__build_cds_url(item["year"], item["field"])
            LOGGER.debug("Harvesting url", url=url)
            yield Request(url, callback=self.parse)
except StopIteration:
            LOGGER.debug("Harvesting all is finished.")
```



4.5.3 XML-JSON conversion

INSPIRE already has a method that transforms data from MARCXML to JSON and vice-versa (**INSPIRE-DoJSON**). This has been reused here. It takes our XML from CDS and translates it to JSON with matching variables. For example, let's say we want to take the ID of the lecture whose tag from the XML is 001. It will be assigned to a new record called *lecture_id* this way:

```
def parse_item(self, selector, original=None):
        ...
        record["lecture_id"] = selector.xpath(".//controlfield[@tag=001]/text()").get()
```

Now here is where the data is passed to create a new record. But first, note that these libraries must be imported:

```
from dojson.contrib.marc21.utils import create_record
from inspire_dojson.cds import cds2hep_marc
```

Last but not least, this part takes the original record and creates a new one by converting the XML record to JSON.

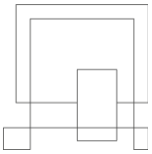
```
data = cds2hep_marc.do(create_record(original))
...
return record
```

When the harvest shows a successful response, the Backend's REST API is used to create new entries.

Note: The harvesting procedure is running daily to get the latest records from CDS.

4.6 Maintain the database

The backend of the software is based on the **Django REST Framework**. After installing the Django REST Framework, the *'rest_framework'* is added to *INSTALLED_APPS*.



/backend/backend/settings/base.py

```
INSTALLED_APPS = [
    ...
    # external apps
    "rest_framework",
    "rest_framework.authtoken",
    ...
]
```

However, since our API is browsable, a file also configures the API's URL paths. It registers default, 'users', 'lectures', and 'search/lectures' routers.

/backend/backend/urls.py

```
router = routers.DefaultRouter()
router.register(r"users", UserViewSet)
router.register(r"lectures", LectureViewSet)
router.register(r"search/lectures", LectureDocumentView, basename="lecturedocument")
```

The *urlpatterns* list routes URLs to views. The '*api/v1/*' is the view for the API, while the '*api-auth/*' is responsible for the REST framework's login and logout views.

```
urlpatterns = [
    path("admin/", admin.site.urls),
    path("api/v1/", include(router.urls)),
    path("api-auth/", include("rest_framework.urls", namespace="rest_framework_auth")),
    path('', include('django_prometheus.urls')),
]
```

Moreover, there is one configuration dictionary in *base.py* named *REST_FRAMEWORK* that contains all global settings for a REST framework API.

4.6.1 Fields in the database

The required lecture fields for the Postgres database like *lecture_id*, *title*, or *date* are defined in the following way:

/backend/cds/models.py

```python
from django.contrib.postgres.fields
from django.db import models

class Lecture(models.Model):
    lecture_id = models.IntegerField(unique=True, db_index=True)
    title = models.CharField(max_length=250)
    date = models.DateField(null=True, blank=True)
    ...
```

4.6.2 Bleaching the abstract

The abstract of the lecture **must be cleansed** before displaying on the UI, because CDS records might have some formatting styles (like HTML tags) that were initially meant for CDS, but they still exist in the metadata. Consequently, they disturb the current UI.

/backend/cds/models.py

```python
from bleach import clean

def save(self, *args, **kwargs):
        try:
            self.abstract = clean(
                self.abstract,
                strip=True,
                tags=["p", "div", "strong", "span", "ul", "li"],
                attributes={"a": ["href"]},
                strip_comments=True,
            )
        except Exception:
            pass
        super().save(*args, **kwargs)
```

As can be seen above, the abstract is taken, the strip will remove entirely invalid markups (strip_comments similarly with built-in comments), and then tags and attributes will only show elements that can be **allowed** within the abstract. For this, a so-called **bleach** [30] Python package has been used, as can be seen above being imported.

5 Test

The system has undergone verification through the utilization of unit tests. In order to evaluate the functionality of our **REST API**, the corresponding tests (POST lecture, GET all lectures, GET single lecture) can be observed in the ***LectureTest*** module located within the *backend/cds/tests.py* file. Additionally, the Harvest service encompasses tests specifically aimed at assessing the efficiency and effectiveness of the harvesting method, as depicted below.

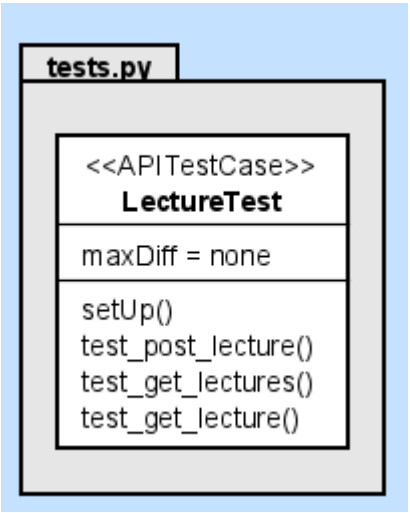


Figure 23: API Test Case

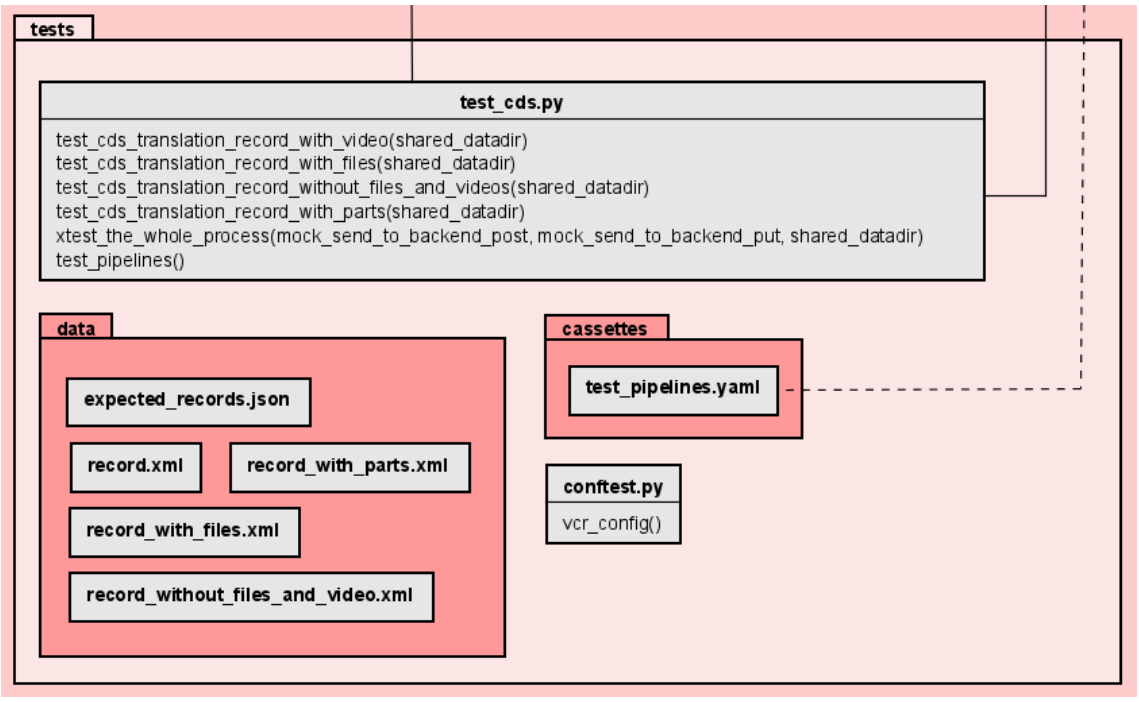
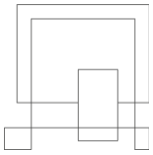


Figure 24: CDS Test Case

Please find the test specifications in **Appendix L**.

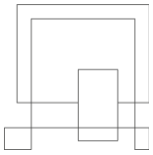


6 Results and Discussion

It was appreciated by the users, we got positive feedback, the project has perspectives for future expansions (see section 8 for details). There is of course a need to publicize and advertise the existence of the site, which only reached operational status in July 2022.

7 Conclusions

The technology evaluation and usage lead to a very good experience for the developer, they also integrated into an existing experience for the maintainer, and they produced an aesthetically attractive and technically functional website. Future collaborations with the CDS team are discussed for other CERN lecture collections to be equally attractively promoted either via separate sites or from within the new CDS. The photos that enhanced the site also gave the opportunity to understand better the life of the laboratory, the types of research, and the experimental installations – also apply skills for photography. There is a challenge now remaining for the ATC to evangelize the site to the user community.



8 Project future

The long-term infrastructure maintenance and evolution, the CDS harvesting supervision will be ensured by the team of Harris Tzovanakis, the CERN SIS - Tools & Services. ATC-related content updates will be done by Salome Rohr.

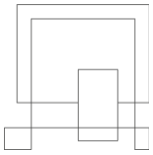
These have been agreed between Maria Dimou and the RCS directorate, project co-founder.

The website's views and especially searches depend on:

- the quality of the recordings,
- the creation of subtitles for the whole backlog of those lectures (after 1989), which contain video, and
- intelligent searches.

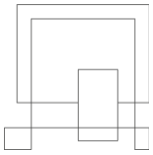
A new dedicated project is needed to harvest searchable strings from the subtitles and the 1968-1989 lectures, which do not contain video.

The CERN Login SSO will be part of the project in the early future, including the possibility to like lectures, add them to watched lists or favorites, and last but not least, a share option to popularize the Academic Training lecture series.

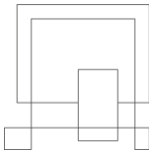


Sources of information

1. Academic Training Committee – About Us. [online] Available at: <https://academictraining.cern.ch/about-us/> [Accessed on 24 July 2022].
2. CERN Document Server (CDS). [online] Available at: <https://cds.cern.ch/> [Accessed on 22 July 2022].
3. Indico – Academic Training Lecture Regular Programme. [online] Available at: <https://indico.cern.ch/category/72/> [Accessed on 5 May 2022].
4. CERN Academic Training website – Project Description. [online] Available at: <https://it-student-projects.web.cern.ch/projects/cern-academic-training-web-site> [Accessed on 5 May 2022].
5. Guidelines for CERN websites. [online] Available at: <https://design-guidelines.web.cern.ch/guidelines/guidelines-cern-websites> [Accessed on 5 May 2022].
6. The CERN Logo. [online] Available at: <https://design-guidelines.web.cern.ch/guidelines/logo> [Accessed on 5 May 2022].
7. The CERN Colours. [online] Available at: <https://design-guidelines.web.cern.ch/guidelines/colours> [Accessed on 5 May 2022].
8. The Homepage of CERN. [online] Available at: <https://home.cern> [Accessed on 5 May 2022].
9. What is Kubernetes? [online] Available at: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/> [Accessed on 26 July 2022].
10. INSPIRE-DoJSON. [online] Available at: <https://github.com/inspirehep/inspire-dojson/> [Accessed on 26 July 2022].
11. What is PostgreSQL? [online] Available at: <https://www.postgresql.org/docs/current/intro-whatis.html/> [Accessed on 26 July 2022].
12. A new web application for the DBoD service. [online] Available at: <https://db-blog.web.cern.ch/blog/baptiste-legouix/2018-09-new-web-application-dbod-service/> [Accessed on 26 July 2022].



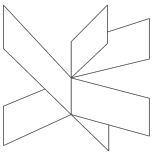
13. Introduction to OpenSearch. [online] Available at: <https://opensearch.org/docs/latest/opensearch/index/> [Accessed on 26 July 2022].
14. Ant Design of React. [online] Available at: <https://ant.design/docs/react/introduce> [Accessed on 26 July 2022].
15. Scrapy documentation. [online] Available at: <https://docs.scrapy.org/en/latest/> [Accessed on 26 July 2022].
16. Python. [online] Available at: <https://www.python.org/about/> [Accessed on 26 July 2022].
17. Django REST framework. [online] Available at: <https://www.django-rest-framework.org/> [Accessed on 26 July 2022].
18. React – Getting Started. [online] Available at: <https://reactjs.org/docs/getting-started.html/> [Accessed on 26 July 2022].
19. About Node.js. [online] Available at: <https://nodejs.org/en/about/> [Accessed on 26 July 2022].
20. What is Drupal? [online] Available at: <https://drupal-tools.web.cern.ch/docs/getting-started/getting-started> [Accessed on 26 July 2022].
21. WordPress. [online] Available at: <https://wordpress.com/> [Accessed on 26 July 2022].
22. What is Invenio? [online] Available at: <https://inveniosoftware.org/about/> [Accessed on 26 July 2022].
23. Introduction – Yarn. [online] Available at: <https://yarnpkg.com/getting-started/> [Accessed on 29 September 2022].
24. What is npm? [online] Available at: <https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/> [Accessed on 29 September 2022].
25. OpenShift. [online] Available at: <https://readthedocs.web.cern.ch/display/MTA/OpenShift/> [Accessed on 16 May 2023].
26. Academic Training website – Documentation site. [online] Available at: <https://academictraining-admin.docs.cern.ch/> [Accessed on 16 May 2023].



27. Docker overview. [online] Available at: <https://docs.docker.com/get-started/overview/> [Accessed on 26 July 2022].
28. React Router. [online] Available at: <https://reactrouter.com/en/v6.3.0/getting-started/tutorial/> [Accessed on 29 September 2022].
29. Axios in React. [online] Available at: <https://www.geeksforgeeks.org/axios-in-react-a-guide-for-beginners/> [Accessed on 26 July 2022].
30. Bleach. [online] Available at: <https://pypi.org/project/bleach/> [Accessed on 26 July 2022].

Appendices

1. Appendix A – Project Description
2. Appendix C – Mockups
3. Appendix D – Diagrams
4. Appendix E – Source Code
5. Appendix F – Source Documentation
6. Appendix G – User Guide
7. Appendix I – UI Design Choices
8. Appendix J – Abbreviations
9. Appendix K – Code Snippets
10. Appendix L – Tests



BSc Thesis in Software Technology Engineering

Process Report

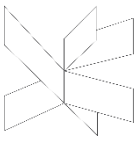
Building a website to promote the CERN Academic Training lectures

Franciska-Leonóra Török, 293171 IT

Supervisors:

Kasper Knop Rasmussen
VIA University College

Maria Dimou
CERN IT, Academic Training



VIA University College

14,940 characters

Software Technology Engineering

7th Semester

2023

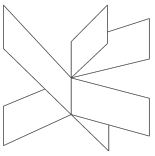


Table of content

1 Introduction 1

2 Group Description 2

3 Project Initiation 3

4 Project Description 4

5 Project Execution 5

5.1 The choice of technology use 6

5.2 The exclusion of categories 6

5.3 The issue with keywords 6

5.4 Different file formats 7

5.5 The postponed CERN SSO login 7

6 Personal Reflection 8

7 Supervision 10

7.1 Separation between the internship work and BSc thesis work 11

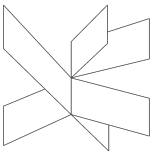
7.1.1 Internship work 11

7.1.2 BSc thesis work 11

8 Conclusions 12

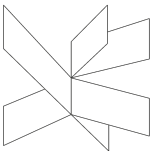
Sources of information

Appendices



1 Introduction

During my Technical Studentship between 1 August 2021 – 31 July 2022 at CERN, the project for the Academic Training website has been completed through several meetings, brainstorming sessions, tasks' recording and follow-up via a formal tracking system, discussions on the development. This document will reflect on the process following the Methodology defined in the Project Description (See **Appendix A**).



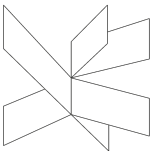
2 Group Description

The IT developer (me) and RCS maintainer (Harris) actively exchanged information and coding improvements during the development, while the project initiator (Maria) occasionally joined the technical sessions and gave valuable inputs for further enhancements. The usability tester (Salome) gave comments mainly regarding CDS, as the lectures were harvested from it. Salome has a wider view of the available Academic Training lectures that are accessible from CDS and must exist on the website.

This restricted development team regularly interacted for feedback with CERN pilot users. Their comments steered the development to more functional directions. This was very positive as an enriching experience and as assurance that the end product satisfied users' expectations.

Details of group communication are mentioned in the **Project Execution** section of this report.

As the laboratory was closed during COVID-19, people had to work from home, so we held our meetings only on Zoom since physical meetings were not allowed. When the restrictions got more relaxed, we returned to physical meetings.



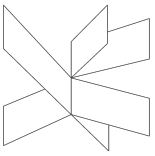
3 Project Initiation

In May 2021, when the opportunity for a one-year Technical Studentship has been confirmed, I have also gotten the offer from CERN to write the BSc thesis on the proposed subject. This had to be agreed on by the Head of Programme and Study Counsellor at VIA, as the first 6 months had to be accepted as an internship, and the rest of the preparation for the thesis.

The Project Description has been already written by the Project Initiator (Maria Dimou) who also proposed the project.

As soon as I arrived at CERN, ideas for the **mockups** came quickly and they were introduced to the ATC members. This presentation received several positive comments that carried professional value.

Note: See **Appendix C** for mockups.



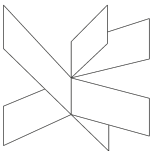
4 Project Description

Maria, the project initiator, had already written the Project Description on the CERN IT student projects' website. This was the proposal that was sent to me by e-mail in April 2021 for a potential BSc subject.

This Project Description later has been adapted to some additional requirements, like the rejection of the Comments, while the rest and newer requirements were accomplished.

The Project Description **for the thesis** I prepared half year later during the course BPR (Bachelor Project Preparation).

Note: See **Appendix A** for the Project Description.



5 Project Execution

The approach to execute this project was determined by:

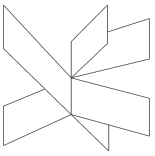
- creating issues in the software **JIRA** under the project named **ATWEB**.
- writing notes in **CodiMD** [3],
- communicating in the team via **Mattermost** and/or **CERNmail,**
- setting meetings on **Indico** and **Zoom** (along with physical ones).

The **Agile-Waterfall hybrid method** has been used throughout the project, which is a combined version of Agile and Waterfall methodology (See **Appendix A** for Methodology). However, we periodically returned to the Requirements and several points have been altered due to frequent changes from the stakeholders.

During my Technical Studentship, I have built the necessary expertise for this project via courses and study that took place mostly during the internship period (the first 6 months). Along with my personal learning path, the project execution happened continuously.

Concerning the Methodology, several discussions and workflow changes led to the decision in favor of an individual method rather than the regular SCRUM [4] or Waterfall [5] ones. Since people working at CERN are busy working on their projects too, setting up meetings and regular checkpoints with them were rather challenging, and required setting appointments sometimes a month or months ahead. Using SCRUM Sprints would not have worked here, but maybe students that are not that busy with other projects at the same time might have ended up well.

Note: See **Appendix H** for a more detailed logbook.



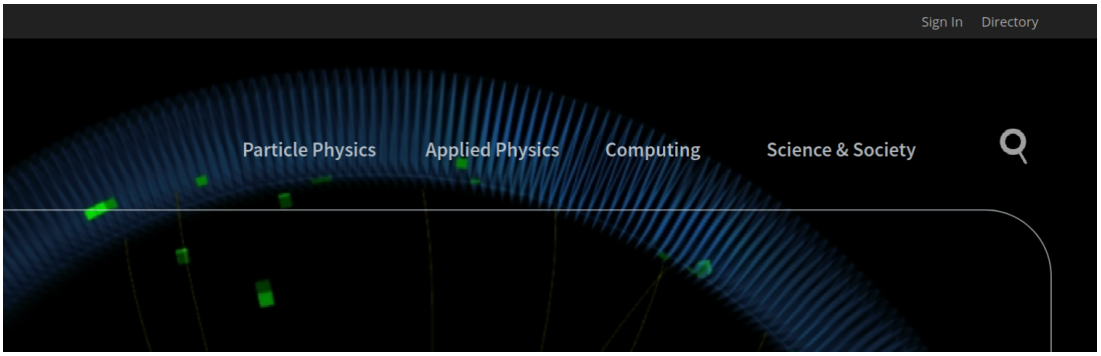
5.1 The choice of technology use

It took several meetings to decide which technology was best to use and how to establish the design for the site. As the majority of CERN sites currently use Drupal, it was initially recommended more. Next, Invenio was suggested as a possible alternative. In the end, React and Django technologies won the debate.

Note: For reasons and arguments see **Technology Alternatives** in the **Project Report**.

5.2 The exclusion of categories

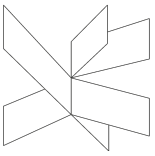
In the early mockups, next to the search engine I have placed several categories that Maria suggested to put in the top banner. While we have been examining the metadata from CDS, we have realized that one piece of data was just missing – the category.



The goal was to fill these categories with lectures, create subcategories, and have the metadata to return relevant results during search. As a Machine Learning algorithm was the only way to improve categorization and searches, which was not a requirement, this has been **omitted**.

5.3 The issue with keywords

The *keywords* data field **did exist** in the Indico API that contained the event information about the lectures, which was eventually built into the software's system. However, due to



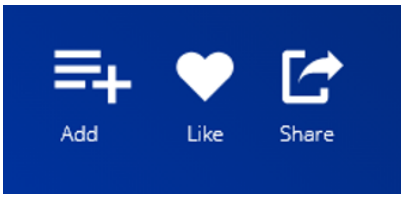
the fact that this field mainly held empty values, it did not have a major impact on the outcome of the search.

5.4 Different file formats

As we handled old records of the Academic Training Lecture series, some limitations came up. Since the CERN Academic Training institution has been around for such a long time, the old records don't have video material, but they do have the knowledge written in files in different file formats. We discovered two types of file extensions: TIF and PDF. Since most browsers' built-in PDF reader makes it easy to display PDF files, only PDFs were handled in this project.

5.5 The postponed CERN SSO login

Earlier, during the design of the mockups, three different buttons were added next to the lecture: **Add**, **Like** and **Share**, as can be seen below.



These buttons were planned to be used for the following goals:

- **Add** – add lecture to a personal playlist or an album (such as Favourites)
- **Like** – like the lecture
- **Share** – share the link of the lecture on several other platforms

Having personal playlists, having the possibility to give likes to lectures would have required to implement CERN SSO login, but it was not a priority. These buttons were additional, optional elements – and hence they have been left out, including the Share button.

While the CERN Design Guidelines required to have the mandatory CERN banner on the top of the site, including the Sign In, which has been indeed displayed. However, knowing that the site viewers do not necessarily need to Sign In in order to have access to the lectures, this has been said to be done by Harris's team after my leave from CERN.

6 Personal Reflection

Figure 1: Portrait of Franciska next to CERN ATLAS Experiment

Franciska

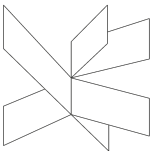
As a Technical Student, the fact that I have been working on a project with a supervisor at CERN for a whole year really helped me a lot to open doors to unique opportunities. I must

admit that I felt extremely lucky to be selected as one of the Technical Students back at the time. I had the chance to gain valuable skills from experts at CERN, both in programming and design, as well as experience in a professional work environment. Maria, my CERN supervisor introduced me to Mónica Bello, the Curator and Head of Arts at CERN. I was invited to different events through her where I had the chance to meet the guest artists of Arts at CERN and take part in interviews and more. I am grateful for my supervisor's continuous attention to my work. She also approved an Adobe packages license for the CERN project and for other photos' editing for publication to the CDS. The photos that I have taken throughout the visits were necessary for the project that I worked on, and I have really enjoyed taking them.

It was an exciting challenge for me to work on the CERN website since web design is something that I enjoy doing. My background in graphic design made preparing the views for the website much faster than constructing the logic. The project was built with React JS + Django from scratch, and although it was a custom solution, I gained valuable experience in coding thanks to it.

While the project itself went smoothly thanks to JIRA and the continuous notes in CodiMD, unexpected errors sometimes came up during the development process, making it difficult to resolve them quickly. There have been a lot of hours, sometimes even days spent searching on Stack Overflow for potential solutions to these problems, but they have eventually been solved. As a result, I gained a broader view of software engineering as I gained experience with a variety of problems.

I caught diseases during the last 2 months of my contract and underwent one surgery after another while writing my thesis and documentation. I had a difficult time getting my health together and setting everything in their final stages. Nevertheless, I ended up with a rewarding experience, was filled with useful knowledge, and I managed to be on time with all deadlines.



7 Supervision

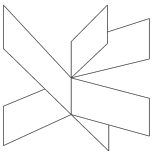
In the beginning, I have been being told that I am not going to get a supervisor from VIA before spring 2022 because I write my thesis from VIA without a group. However, my CERN supervisor, Maria Dimou, heavily suggested contacting one of my professors from VIA for supervision during the preparation of the BSc thesis.

Following the first Zoom meeting with Kasper Knop Rasmussen, Maria and I – Kasper agreed upon supervising the project from VIA. Shortly after, the VIA Engineering Study Counsellor contacted the Head of Programme to reconsider their previous decision about having an individual supervisor from VIA, which later has been confirmed.

Maria closely supervised me at CERN throughout the whole year, and Kasper provided valuable feedback informally during a couple of Zoom meetings between the three of us. We usually set Zoom meetings a month ahead. The backlog of the supervision meetings between Kasper, Maria and I – can be seen below:

Meeting subject	**Date**
Supervision agreement and technology approval for BSc thesis	11 October 2021
CERN Terra Incognita Presentation [6] and BSc thesis plan	2 March 2022
BSc Thesis Draft – Project Description	8 April 2022
BSc Thesis Draft – Final Project Description	19 April 2022
BSc Thesis Draft – Requirements and User Stories	06 May 2022
BSc Thesis Draft – Project Report	17 July 2022
BSc Thesis Draft – Process Report	16 August 2022

Figure 2: Supervision meetings



Me and Maria were able to share our inputs and comments about the project and documentation daily due to the direct supervision I have received at CERN. Being in contact with Kasper was only possible through online sessions since Kasper was in Denmark, while Maria and I were at CERN (Switzerland).

Note: Kasper supervised the project over the year of 2022. A clear separation has been set between the work done in the internship and the work that had to be done for the thesis itself to make sure ***the same work is not credited twice***.

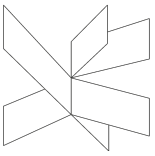
7.1 Separation between the internship work and BSc thesis work

7.1.1 Internship work

August 2021 - January 2022 (6 months)
The learning process of Linux system administration, React, Django, Kubernetes, OpenShift, and the CERN Web Frameworks. I had a CERN Udemy license for any technical and/or managerial course, potentially useful for my future career.
I have also followed the CERN French language courses. Maria approved an Adobe ® packages license for my web development project and for other photos' editing for publication to CDS.

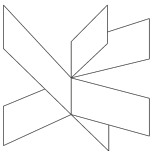
7.1.2 BSc thesis work

February 2022 - July 2022 (6 months)
Application of the acquired knowledge with the above-mentioned tools to implement the website [7] according to the Project Description (See **Appendix A**).
Scripting for video fetching from CDS, development of the Search function, performance testing, entry in operation, documentation [8], and enhancements based on users' feedback.



8 Conclusions

The process we followed at CERN for this project was slightly different from the academic way taught at VIA. The reason was the collaborators' busy schedules and the multiple iterations and input changes of our users. This showed that different work environments and cultures may have different methods to approach a goal. Adaptation is key and this versatile approach allowed us to gain experience and learn from the experts on the job. One of the lessons learned was how to match priorities to colleagues' availabilities. Planning and keeping track of work done made this project a success.



Sources of information

1. Mattermost. [online] Available at: <https://mattermost.com/customers/cern/> [Accessed on 25 July 2022].

2. JIRA. [online] Available at: <https://its.cern.ch/jira/projects/ATWEB/> [Accessed on 25 July 2022].

3. CodiMD – Notes for Franciska's BSc Thesis. [online] Available at: <https://codimd.web.cern.ch/I75TITVDRKyeFEl-yS79uA> [Accessed on 25 July 2022].

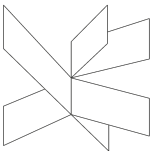
4. What is SCRUM? [online] Available at: <https://www.scrum.org/resources/what-is-scrum/> [Accessed on 25 July 2022].

5. Waterfall Methodology. [online] Available at: <https://www.workfront.com/project-management/methodologies/waterfall/> [Accessed on 25 July 2022].

6. CERN Terra Incognita Presentation – The public web site for Academic Training. [online] Available at: <https://indico.cern.ch/event/1080998/> [Accessed on 24 July 2022].

7. Academic Training website. [online] Available at: <https://academictraining.cern.ch/> [Accessed on 25 July 2022].

8. Academic Training website – Documentation site. [online] Available at: <https://academictraining-admin.docs.cern.ch/> [Accessed on 25 July 2022].



Appendices

1. Appendix A – Project Description
2. Appendix B – Group Contract
3. Appendix C – Mockups
4. Appendix H – Backlog