

Design and development of the Transnational Access web portal for RADNEXT EU-Project

Tymoteusz Ciesielski

Supervisors: Blerina Gkotse and Federico Ravotti

October 14 2021

1. Introduction

During my summer internship at CERN I was a member of the **IRRAD team** [1]. The project that I was realising is strongly connected to the RADNEXT programme [2]. It is an infrastructure programme with the objective of creating a network of facilities and related irradiation methodology for responding to the emerging needs of electronic components and system irradiation.

My job consisted of creating a **web portal** for this programme, which:

- Includes the database with extensive information about the RADNEXT irradiation facilities.
- Provides a place for users to apply for the RADNEXT facilities and submit proposals of their projects. Users can also view their proposals and their status.
- Allows the RADNEXT committee to view all the proposals.
- Supports the decision making process for the RADNEXT committee, suggesting the correct match of users with the facilities. For example, If a user applied to use a muon radiation, it would suggest the facilities with this type of radiation.

It is also worth mentioning that by the time of starting my project, I already had access to the database of AIDA-2020 irradiation facilities. [3] There are around 200 different irradiation facilities from all around the world. RADNEXT facilities are the subset of those facilities (there are around 38 RADNEXT facilities at the moment of writing). My job was also to create a separate database for those 38 facilities, that would extend more precise information compared to the already existing AIDA-2020 Database.

2. Technologies

2.1. Django Framework

The application is written in Django, which is a Python framework for web development. It was chosen jointly by me and my supervisors even before the beginning of the actual internship. I must say that I am a huge enthusiast of it. Among its advantages there are:

- Security
- Relatively rapid development (the motto of the framework is “The web framework for perfectionists with deadlines”)
- Scalability
- Plenty of handy libraries (also thanks to Python)
- Dedicated Django Administrator Panel

There are, however, in my opinion three disadvantages worth mentioning.

- Django Framework has a very high dose of abstraction, which makes it difficult for the newcomers. The learning curve is quite steep.
- There are always multiple ways of implementing some functionality and it requires some experience to know which one should be chosen. For example a programmer can choose between more abstract class-based views or more verbose function-based views. The framework is however very well documented [4] and there are a lot of tutorials.
- The framework is changing very rapidly. The latest official version is 3.2.8. The versions are not fully compatible with each other. It happened to me a couple of times, that I was reading some tutorial created with an older version of Django, and I was confused with the syntax and tools used.

Django also implements the so-called MVC - Model View Controller design pattern [5]. Basically, it means that the app is divided into three interconnected elements:

1. **Model** - directly manages the data, maps the entities in the relational database
2. **View** - representation of the information (models). In our case, views were responsible for what is “viewed” on each site of the web portal. Talking more in the “python language”, views can be implemented either as functions (more verbose, low level approach) or classes (more abstract, implementing the inheritance).
3. **Controller** - manages the input from a user (HTTP requests). It is “the brain” of the application.

Design patterns are sometimes vague when it comes to such big projects as Django. Lots of people argue that Django should be called MVT pattern - Model-View-Template.

2.2. The Database

It was decided at the beginning of the project that there will be two Database Management Systems used for the development of the application.

SQLite

It is a very easy to use Database System [6]. The most important thing concerning it, is the fact that the whole database associated with it, can be packed into one file (which is not the case with other systems). It was therefore used for most of the time in the development process. It is free and **open-source** and it was replaced by MySQL only during the testing.

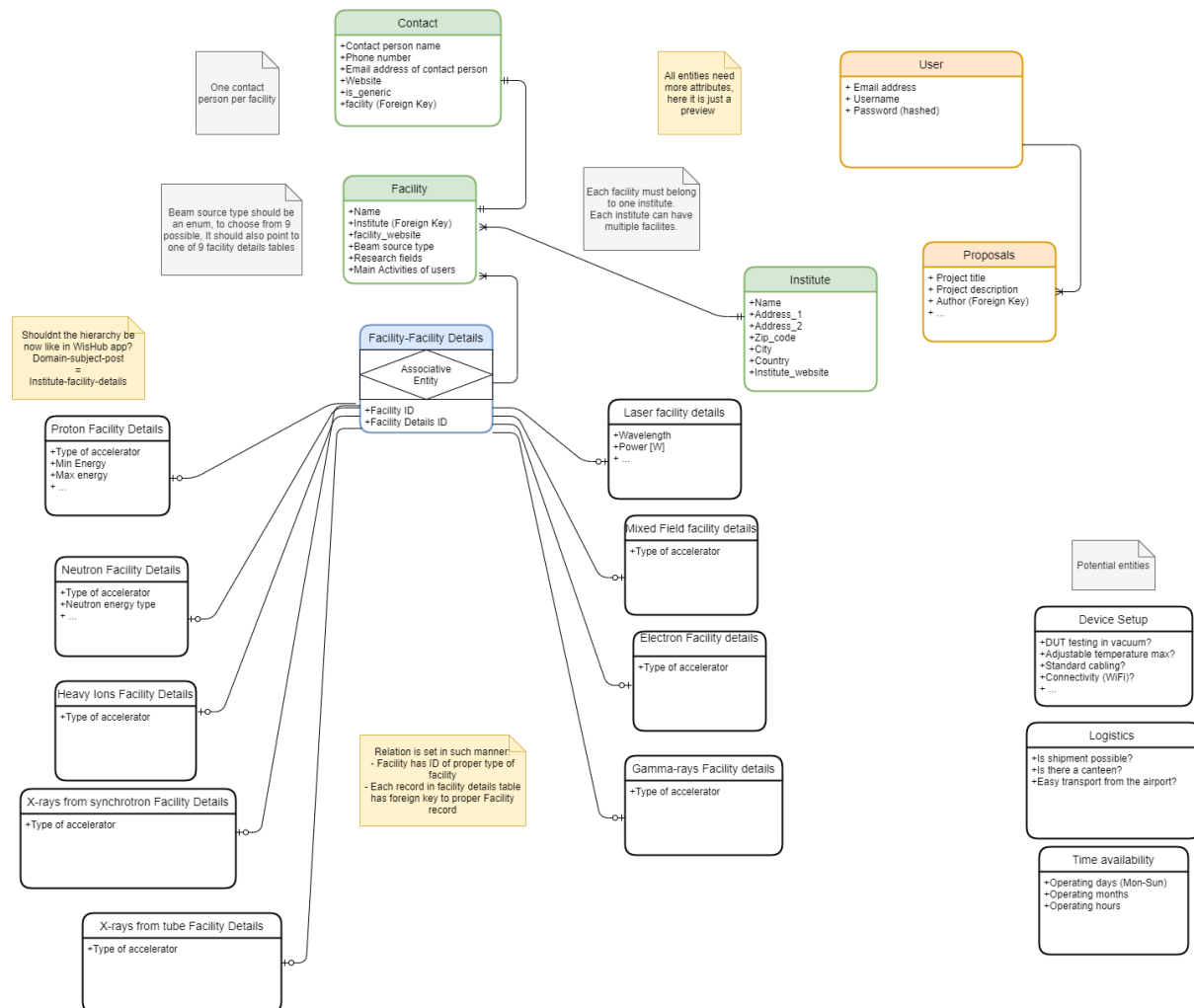
MySQL

This Database Management System is also free and **open-source**. It is considered to be more reliable and used mostly in bigger projects. However, it is more difficult to use than SQLite.

There is also one important tool that I used during my project (Database Administration Tool [7]), and that I wanted to elaborate a little bit about. It is **DataGrip** [8] and it helped me a lot during this project and the previous ones.

It is an easy-to-use but powerful tool, developed by the Czech software development studio - **JetBrains** [9]. The tool itself is neither free, nor open-source. However, thanks to my University I was granted the license to use it. It allows users to easily view and modify information from the database. It is compatible with multiple database systems. It has GUI so you can reach most of the functionalities in a simple couple of clicks, but it also allows you to write raw SQL queries.

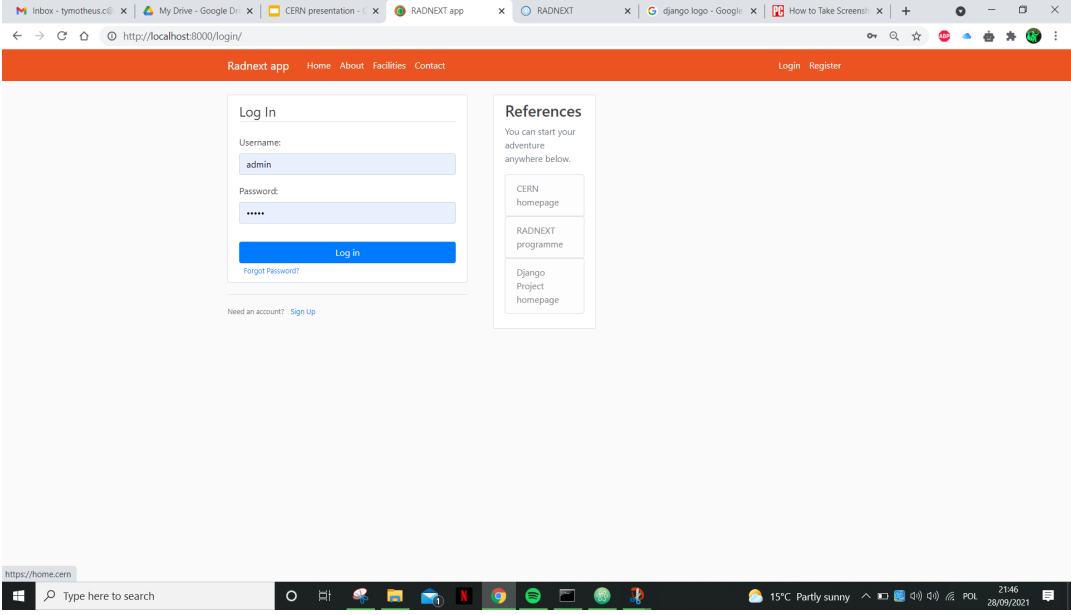
The final Entity Relationship Diagram of the web portal database is shown below.



The main entities are marked in colours. Orange entities are users, and proposals that they submit. In green, you can see facilities, institutes and contacts (mail addresses, phone numbers etc.). Blue colour is for the entities with the detailed information about the facilities. There are also relationships between entities marked in the picture. For example, each irradiation facility belongs to one (and only one) institute, but institutes can have multiple facilities.

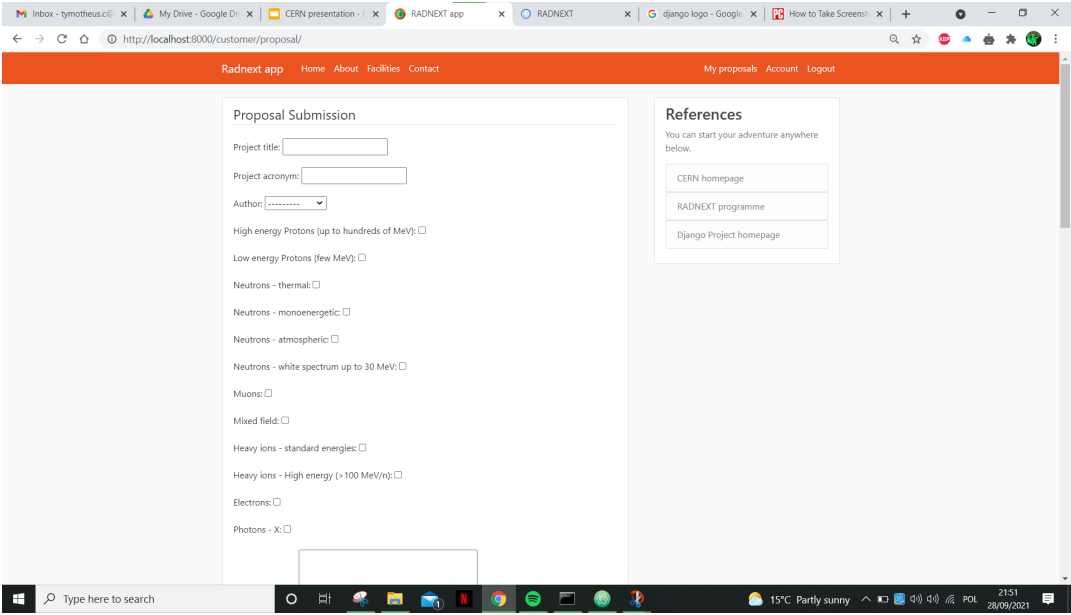
2.3. Frontend

For the look of the web portal, I was using HTML templates and **Bootstrap** [10]. It is a free and open-source CSS framework. In brief, it is a frontend library that allows you to easily implement nice looking buttons, forms and bars in the application. It is a convenient tool, to make the application look good in a relatively quick way. The prototype version of the application is shown below. First a user can log in to it:



The screenshot shows a web browser window displaying the login page of the 'Radnext app'. The browser's address bar shows 'http://localhost:8000/login/'. The page has an orange header bar with 'Radnext app' on the left and 'Login Register' on the right. The main content area is divided into two sections. On the left is a 'Log In' form with fields for 'Username:' (containing 'admin') and 'Password:' (masked with dots), a 'Log in' button, and a 'Forgot Password?' link. Below the form is a link for 'Need an account? Sign Up'. On the right is a 'References' section with the text 'You can start your adventure anywhere below.' and a list of links: 'CERN homepage', 'RADNEXT programme', and 'Django Project homepage'. The Windows taskbar at the bottom shows the date as 28/09/2021 and the time as 21:46.

Later on, a user who wants to apply for the irradiation facility, can make a relevant proposal:



The screenshot shows a web browser window displaying the 'Proposal Submission' page of the 'Radnext app'. The browser's address bar shows 'http://localhost:8000/customer/proposal/'. The page has an orange header bar with 'Radnext app' on the left and 'My proposals Account Logout' on the right. The main content area is divided into two sections. On the left is a 'Proposal Submission' form with fields for 'Project title:', 'Project acronym:', and 'Author:' (a dropdown menu). Below these are several checkboxes for different types of irradiation: 'High energy Protons (up to hundreds of MeV):', 'Low energy Protons (few MeV):', 'Neutrons - thermal:', 'Neutrons - monoenergetic:', 'Neutrons - atmospheric:', 'Neutrons - white spectrum up to 30 MeV:', 'Muons:', 'Mixed field:', 'Heavy ions - standard energies:', 'Heavy ions - High energy (>100 MeV/n):', 'Electrons:', and 'Photons - X:'. On the right is a 'References' section with the text 'You can start your adventure anywhere below.' and a list of links: 'CERN homepage', 'RADNEXT programme', and 'Django Project homepage'. The Windows taskbar at the bottom shows the date as 28/09/2021 and the time as 21:51.

3. Structure of the project

Every Django project consists of one or more apps. There are native Django framework apps and apps created by the developers. Such an approach allows to logically divide functionalities in the project into submodules - they can potentially be used later in different projects. Apps should be named in a manner indicating their purpose. For example, I was using an external **“rest_framework”** app to create REST API for my project. I was not paying a lot of attention to the names of apps created by me, so they can seem a little bit misleading. During the development I created the following apps:

- 3.1. **“customer_forms”** - it is responsible for the interaction of users with the web portal. It stores the information about the data models of proposal submission - for the users applying for the RADNEXT facilities. In the development version it also has a folder storing all the HTML templates.
- 3.2. **“rad_away”** - it is responsible for the data models of facilities and institutes - they have a lot of details and there is a separate folder **“facility_details_models”** with the additional details. It offers REST API to interact with the models of this app. It was the first app in the project developed by me.
- 3.3. **“accounts”** - it stores the information about the users accounts. This app is currently in development. It should implement the custom user model, inheriting from the generic Django User class. It implements signup and login to the portal with the necessary hashing of the passwords.

There is also **“radnext_app”** in the project. It was created by default. It was modified by me, especially the **“settings.py”** file, which holds the information about the external apps that are used in the project.

4. Development process

At first, I wanted to apply **TDD - Test Driven Development** [11], software development methodology, that implies working in the following manner: tests are the core element of the application. Firstly, developer creates software tests that should cover the functionalities that he would like to feature in his application (or in its module). After this part, when the tests are run, they will obviously fail as there is no proper software yet that covers this functionality. Developer has to therefore create software and functionalities that will make his tests pass. When the functionality is properly implemented, the tests are passed. Developer can repeat this process, start writing new tests and incrementally supply new features of the application.

Exhaustive tests require a lot of time - especially when the data models contain a lot of detailed information. Due to the obvious time limitations of the internship, I created basic tests just for the core functionalities and focused more on the software development itself.

Then I received access to the data of some irradiation facilities that were already present, so I could confront them with my application. The data was submitted with google forms. As it was filled by a lot of different users, **the quality of the data was also varied**. I inserted the full data of one irradiation facility, and its contact person. It was working, but I

decided not to add more real-life facilities to my database. Also I was constantly improving the data models that were present in my project.

Then I decided to focus more on the **users of the application**. I created the “customer_forms” app and later the “accounts” app. I added the new data model for the proposals. Also I started working on HTML templates and Bootstrap and I created the first working functionality on frontend - submitting a proposal. It supports all the CRUD operations - creating, reading, updating and deleting. From this time, I was constantly improving the templates to make the portal look better. It could be now tested manually, from the level of the browser.

The final part of my internship consisted of debugging some minor functionalities, improving the templates, together with Bootstrap features and preparing for the final presentations for the IRRAD team and the EP-DT team.

5. Summary

Summer Student internship was an amazing experience for me. Apart from interesting lectures, I had the opportunity to **develop a prototype of a working web portal**. The internship also allowed me to learn:

- Django Framework and a lot of its interesting features
- Irradiation physics - different types of radiation (protons, electrons, muons etc.) in the irradiation facilities all around the world and Europe
- Design of the application - as it is not an easy process, and if done in a correct way - can save a lot of time of coding and debugging
- Design of the database models - I must say it is one of my favourite parts, to design how the models interact with each other and what information they store.
- Basics of Bootstrap - so the app can look decently for the user
- Solving problems - in a both coding way, and time-and-resources management

It is also important to indicate the possible features that could be added or extended in the future of the project:

- Adding the core functionalities - irradiation facilities providers panel, radnext admin panel. It is also associated with different user roles.
- Implementing a decision support engine. There is a lot that can be done here, it could be in fact a subproject. The engine could feature machine learning, or classical algorithms.
- Changing the layout. The web portal could be more eye-pleasing.
- There are also a lot of business intelligence questions to be answered. Such as: what roles and privileges should different users have in the web portal? Should there be different logging methods added (through Google Account)? And much more.

References

- [1] IRRAD facility paper - <https://cds.cern.ch/record/2237333?ln=en>
- [2] RADNEXT programme - <https://radnext.web.cern.ch/>
- [3] AIDA Database - <https://irradiation-facilities.web.cern.ch/>
- [4] Django documentation - <https://docs.djangoproject.com/en/3.2/>
- [5] Model-View-Controller
<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>
- [6] SQLite main page - <https://sqlitebrowser.org/>
- [7] Database Administration - https://en.wikipedia.org/wiki/Database_administration
- [8] DataGrip - <https://www.jetbrains.com/datagrip/>
- [9] JetBrains - <https://www.jetbrains.com/>
- [10] Bootstrap - <https://getbootstrap.com/>
- [11] Test Driven Development - https://en.wikipedia.org/wiki/Test-driven_development