FII Assistant

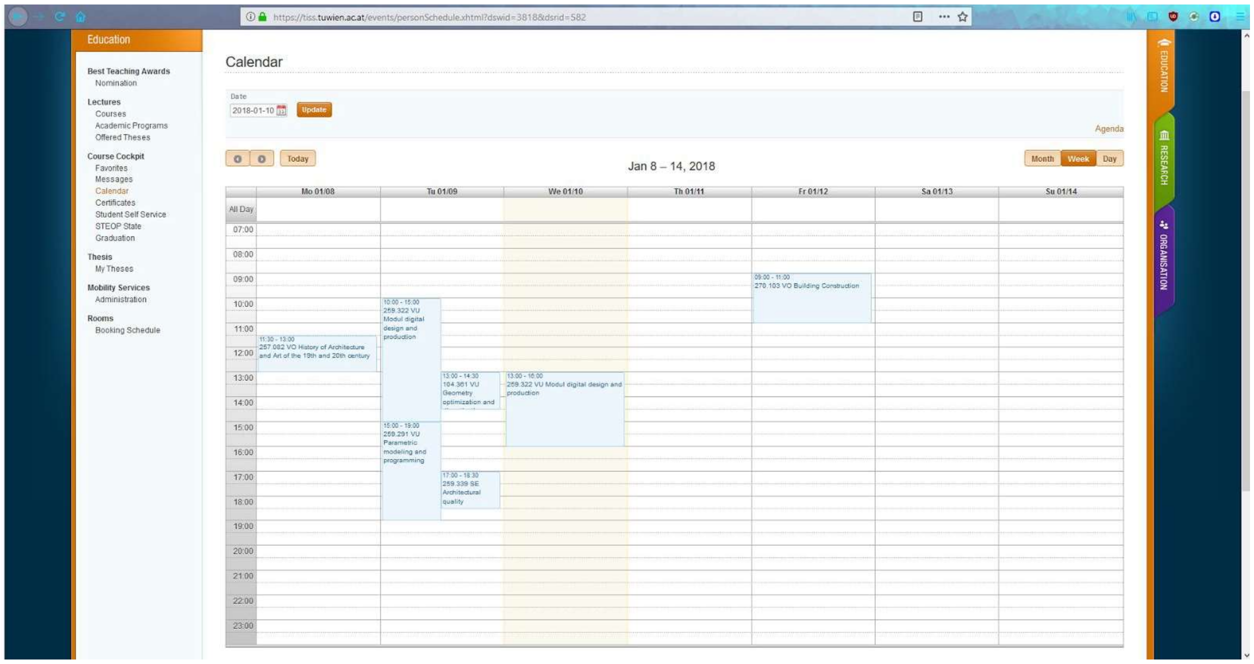
January 2018

Problem Presentation

Our current student portal at FCS (FII), lacks integration with its various services for students. Students cannot login on the main page or access a dashboard-like view. The existing website does not respond to the needs of the students since a lot of the online services are non-existent - they require direct contact with the staff. Other issues include problems with the schedule and the student course selection, as not all options are displayed or optional course selection is not always clear. Furthermore, a lot of the existing course information, such as exams, documentation or even materials are not all congregated in the same place. Seeing as the university has an increasing number new students, and this is their main portal, a lot of these issues cause people to no longer access the webpage and seek information via other means, such as Facebook, and end up misinformed, or end up bothering the staff with trivial questions about things such as the schedule.

State-of the Art

If we are to consider some of the best student portals that exist, we should have a look at british universities. They've basically stamped out a certain design and functionality pattern for their main pages, while each of them has managed, in time, to rebuild and integrate their initial functionality, such as course scheduling or enrollment features. By having a look at student portals like Oxford, http://www.ox.ac.uk/, Manchester, http://www.manchester.ac.uk/, Cardiff, https://www.cardiff.ac.uk/ , universities have a similar design, where they combine the elements of Enrollment, Existing Students Intranet, News/Information, Business/Research and finally Site Map & Search. Having done so, this allows clear navigation through the existing services. And while some may link to some more outdated internal web-pages, having a clear cut pattern, helps solve many of existing users issues. They also have created certain levels of access for staff later on, as to allow professors to view grade students, or the creation of other services that may become necessary in the future. As such, when compared to our initial problem, we have a clear cut method of improving our current features and we have an organisation method presented to us. It is also good to note, that, while UK universities do put an accent on presentation, german/swiss uni, tend to take a more functional design, but keep much of the elements mentioned above. Below: Calendar Schedule for a week for a student.



Jan 8 – 14, 2018

	Mo 01/08	Tu 01/09	We 01/10	Th 01/11	Fr 01/12	Sa 01/13	Su 01/14
09:00					09:00 - 11:00 270.103 VO Building Construction		
10:00		10:00 - 15:00 259.322 VU Modul digital design and production					
11:00	11:30 - 13:00 257.082 VO History of Architecture and Art of the 19th and 20th century						
13:00		13:00 - 14:30 104.361 VU Geometry optimization and	13:00 - 16:00 259.322 VU Modul digital design and production				
15:00		15:00 - 19:00 259.291 VU Parametric modeling and programming					
17:00		17:00 - 18:30 259.339 SE Architectural quality					

Your solution

Our solution was the creation of a .NET application that is based on various microservices using C#. This allows us, or anyone in the future to work and create his own microservice for the existing application that adds various features and functionalities to the website.

Our current system allows us to make user accounts and stay logged in using a token system for two hours. We can see the current course list, that is attributed to our current year, and have a schedule displayed to us, based on those courses. The course list is going to be generated, by parsing the current existing webpage and adding them to a database. The same will be done with the existing schedule. Though currently there is no difference between user types, this can be added as later functionality.

Strictly speaking the current solution is the bareback bones structure that can be used to later build upon other functions.

Results, Evaluation

Currently our application runs smoothly and collects information from the other websites of the university. Two process parse the existing information in the schedule and the existing courses and create a database that is put to use as part of the application. Since it was thought out as a modular application, we can extend its current functionality, and parse other information, having a flexible working model, that integrates old existing elements. It is secured using authentication and authorization with JWT, currently users having tokens generated for each session that lasts two hours.

Server side results

In order to implement the microservice architecture we need a structure that combines and manages communication between modules. This is represented by the server. In order to achieve these results we built the two main components of the server:

- a communication layer
- a reusable **HTTP client.**

The first one (communication layer) provides an endpoint to an existing client application for getting functionality that is built on the microservice layer. Also, it offers a data exchange mechanism between microservices, increasing decoupling in the way that no microservice is dependent on other microservice.

In order to implement the communication layer we composed a **HTTP client.** It is used for providing a generic tool that helps the server module and also the microservices to send their requests. This includes multiple types of HTTP requests (GET, POST, PUT, PATCH, DELETE).

In the end, we managed to build a server side component in a SOLID manner that is able to handle all the client requests. The workflow pattern contains requesting other microservices for providing specific data and also resolving the requests received from the microservices.

Timetable Solution

The timetable module is part of the microservice architecture designed at a project level. This module is separated from all the other modules and returns isolated data. The desired behaviour of this module is to parse the online timetable available for all the groups of students and return the data in a reusable object oriented model.

The business requirement was met by creating a parser as the core of this module which receives a timetable request with basic data like: e.g. group, year. Based on this request type and the type of response wanted (e.g. day timetable, week timetable) the parser requests the content from the online table. Then the content is parsed and is mapped into an appropriate data structure taking into account the structure from the online timetable.

After the parsing is done, the completed data structure is sent to the controller which in turn communicates with the server or other external consumers. In this way this module is completely decoupled from the other functionalities and has a single responsibility.

Comparison with other solutions

Further research is necessary at this stage due to the alpha stage of the application and lack of friendly user interface which allows us to compare the user experience between different solutions.

At a feature level, our architecture allows us to extend the functionality very easily which is an advantage over other tightly coupled solutions.

Future work

As at this stage this is purely theoretical, but with the current way the application is build, we can progress in a manner of directions and create a wide variety of microservices for:

- Secretary Work - Contracts/Paperwork/Payments
- Student Enrollment
- Scheduling Appointments
- Dissertation/Research and Thesis Work - Applying/Proposing/Referencing
- Publishing/Archiving Work
- Intranet - Professor Course Storage
- Online Exams or Assignment Submission
- Partnerships (since our university has loads of connection to companies)
- Foreign Students Portal
- ...and others

But we mentioned a wide variety of things that CAN be done. But since the required work for such an arduous task would require an entire team, having an outline of what can be done, is just as important in order to have a idea of which direction work should progress in. As the program will be developed throughout the next year, we can take our time and make various choices.

Conclusion

As presented earlier, the project developed this semester is only the first step towards an application which will be fully integrated in the students' life.

By starting with a fully modular and reusable infrastructure, we can easily extend the existing platform in the future, being able to scale up to almost all requirements which might appear. The microservice-ready architecture will guarantee that at any time we will be able to migrate to fully independent microservices. This is crucial, since we might not be the ones developing this application in the future: it is easily extensible, without breaking the already existing functionality.

The next logical step in the development of this product will be to create an interactive user interface, with a rich user experience. This will require study of already existing applications in this area, to see which approach is the best when it comes to student portals. However, we will keep in mind the way our students use the portal that is available right now, and start building up a better experience from there.

In conclusion, we strive to give our colleagues at FCS a friendlier approach to all the tools provided by our faculty, in an intuitive and friendly experience.