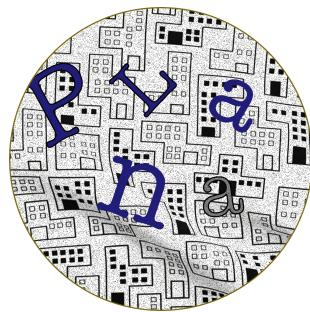


Bern University of Applied Sciences | BFH

Department of Engineering and Information Technology

Bachelor's Thesis (BTI7321) Autumn Semester 2020/21

"Planning of the Assignments for Lecturers(PLANA)" Web Application



Author: Kristina SHIRYAGINA (kristina.shiryagina@bfh.ch)

Supervisor: Prof. Marcel PFAHRER (marcel.pfahrer@bfh.ch)

Expert: Dr. Federico FLUECKIGER

January 13, 2021

Contents

Acknowledgments	4
Abstract	5
1 Introduction	6
1.1 Acronyms	9
1.2 Glossary	9
2 Project Management	10
2.1 Effort Estimation	10
2.2 Scrum	10
i. Scrum Roles	11
ii. Scrum Plan	11
iii. Scrum Artifacts	11
iv. Sprints	13
Conclusion Project Management	13
3 System requirements	13
3.1 Product Backlog and User Stories	13
4 Functional Requirements Specifications	18
4.1 Stakeholder Summary	18
4.2 Actors and Goals	19
4.3 c. User Interface Requirements	19
5 System Architecture and System Design	20
5.1 a. Architecture Styles according business logic	20
5.2 The Plana application Architecture	23
6 User Interface Specification	24
6.1 a. Preliminary Design	24
7 Domain Analysis	35
7.1 Domain Model	35
i. Concept Definition	38
ii. Association Definition	40
8 Technologies	41
9 Creating the Projects	41
9.1 Structure of Projects and Folders	41
9.2 Database and Entity Framework Core	41
9.3 The PLANA App's Relational Database	41

9.4	Modeling Types of Database Relationships	41
	Many-to-many Relationship	41
9.5	Creating Database	44
9.6	Creating a Repository	45
	Creating the Database Migration, Code-First Migration	46
9.7	Creating Seed Data	46
	Configuration of Core Services and Entity Framework	47
9.8	Create a Controller	47
	Complex Data Model	47
	Blazor Server	49
	Configuring ASP.NET Core for Blazor Server	49
9.9	Changing Architecture	52
9.10	Setup for Blazor	52
10	Testing	52
10.1	Test Cases	53
	Testing the ModuleRun Controller	53
	Testing the LecturerModuleRun Controller	54
	Testing the Modules Controller	55
	Testing the Plan Controller	56
	Testing the Plan Lecturer Controller	57
	Testing the Semester Controller	57
11	Conclusions	59
12	Appendix	61
13	Declaration of Authorship	61
References		62
14	Protocol	63

Acknowledgments

Abstract

PLANA (Planning of Assignments for Lecturers) is a web application tool for effective teacher planning. With PLANA, teachers can participate in the planning process, make suggestions about the modules which they teach, and schedule hourly workloads for specific modules. In this thesis, we improve the planning of assignments for lecturers by implementing a web application using ASP.NET Core Blazor. We also describe the application's system architecture and system design and perform domain analysis. Additionally, we justify our choice of implementing PLANA as a web application by exploring the advantages of web applications over Microsoft Excel solutions and desktop applications.

1 Introduction

At our school at the Department of Technology and Computer Science, teacher assignment planning is done using Microsoft Excel. This plan is handled by one person. In the modern world, with the rapid growth of new technologies, it is possible to improve various systems, giving them more and more possibilities, automating many functions and saving a lot of time. This thesis aims to develop an information system that eases assignment planning for lecturers. But unlike the existing system, it should fulfill the following criteria.

- The teachers themselves should be involved in the planning process
- The ability to create groups of modules and groups of teachers
- Increased planning flexibility

This thesis also employs an innovation technique called **product development**. The previous project (project2) and this thesis include all phases of product creation.[1]

- An analysis of new trends in technology
- A study of the client's needs
- An idea of how to create the product based on the selected technologies and concepts
- Product implementation and testing

This thesis contains the following novelties.

- Establishing successful ideas, such as:
 - Involving teachers in the planning process, which reduces planning time and makes the process easier
 - Creation of groups of teachers and modules, making it easier to coordinate planning
- Creation of a specific solution - a product that includes great advantages over the existing product. These advantage are:
 - Possibility of collaborative work, i.e. all system actors can actively participate in the planning process.
 - Data consistency
 - A better overview of the entire system
 - Accessibility regardless of location, etc.

The planning process involves the input of specific data for specific user-defined views for each user, as well as time limits set by the system.

All of these requirements need a more suitable system than Microsoft Excel. In the previous project, we compared a **Microsoft Excel** solution to a generic **Web Application** with respect to several criteria and concluded that the web application meets the requirements of the conceived system.

- The web application is designed to involve **many users**
- It can have a database that gives us **data consistency**
- Also, the data is much **safer** in a database
- The web application gives the best **overview** of the entire system

Comparing a generic **Desktop Application** to a generic **Web Application** also results in the web application being the better option:

- it is **It is accessible from anywhere**
- **No updates required**
- **Costs less**

We decided to make a web application that will meet all the requirements and will be created using suitable modern technologies. ASP.NET Core Blazor Server with Entity Framework Core (EFCore) and MS SQL (Microsoft Structured Query Language) were the database technologies of choice. ASP.NET Core Blazor is a new framework that is gaining popularity. A major advantage of using ASP.NET Core Blazor is that it becomes possible to do the entire application in C# without using JavaScript.

This work is a continuation of project 2, which was completed last semester, where we have prepared the necessary environment for this project by creating a prototype. In this thesis, the system will be expanded. In particular, the following goals are pursued.

- **Involvement of lecturers in the planning process.** Lecturers can create their medium- and long-term plans in form of requests and proposal for the definite plan, which is then approved by the person responsible for the planning
- **Manage planning data.** Each teacher can manage his assignments.
- **Grouping of lecturers.** It should be possible to schedule several lecturers for the same module.
 - Teachers who join a group can independently manage their assignments related to their common module.
 - Each teacher can make a group with other lecturers.

- **Grouping of the modules.** The group of lecturers can choose a group of modules and assign themselves to it. This can be done in the form of a proposal for the definitive plan, which is then approved by the person responsible for the planning.

This thesis has the following structure. First, we will explain how the project was organized and how SCRUM was used to manage it. Then, we will carry out an additional analysis of the system pertaining to the expansion of system requirements. We will also make changes to the domain analysis and extend the System Architecture and System Design chapters. Finally, we present the Project Implementation and Testing.

1.1 Acronyms

Acronyms	Words
EF	Entity Framework
CSS	Cascading Style Sheet
KKK	345

Table 1: Caption2

1.2 Glossary

- **FURPS+**[2] is a system for classifying requirements.
 - Functionality
 - Usability
 - Reliability
 - Performance
 - Supportability
- **SignalIR** is a free and open-source software library for Microsoft ASP.NET that allows server code to send asynchronous notifications to client-side web applications.
- **Blazor** is a free and open-source web framework that enables developers to create web apps using C# and HTML. It is being developed by Microsoft.
- **EF Core**
- **HTML** Hyper Text Markup language
- **SQL** Structured Query Language
- **JS** JavaScript
- **CRUD** Create, read, update and delete
- **UI** User Interface
- **API** Application Programming Interface
- **MS** Microsoft
- **BPMN** Business Process Model and Notation
- **Entity Framework(EE)Core** Is an Object-Relational Mapping (O/RM) framework. It is an enhancement to ADO.NET that gives developers an automated mechanism for accessing storing the data in the database

- **NuGet** Is a free and open-source package manager designed for the Microsoft development platform.
- **DbContext** DbContext is an important class in Entity Framework API. It is a bridge between domain or entity classes and the database. DbContext is the primary class that is responsible for interacting with the database
- **Database** Database is an organized collection of data, generally stored and accessed electronically from a computer system
- **Microsoft SQL Server** Microsoft SQL Server is a relational database management system (RDBMS) that supports a wide variety of transaction processing, business intelligence and analytics applications in corporate IT environments
- **ASP.NET Core APIs** is a free and open-source web framework and successor to ASP.NET, developed by Microsoft and the community. It is a modular framework that runs on both the full .NET Framework, on Windows, and the cross-platform .NET Core.
-
-
-
-
-

2 Project Management

2.1 Effort Estimation

The Bachelor's Thesis is designed as a 12 ECTS module. This corresponds to a workload of 360 hours. To control the hours allotted to work on the project the author used an excel. At the end of the project, by the time the documents were submitted the total working time was... Together with the subsequent preparation for the defense and presentation, the total time will be from 420 to 450 hours.

2.2 Scrum

The foundation of the project organization was Scrum. Some principles of Scrum could not be achieved since they need a group of more than two people. Our work was based on the principles of Scrum like the Empirical Process of Control, the core of Scrum, self-organization, value-based prioritization, etc. The Empirical Process of Control includes three main ideas, namely transparency, inspection, and adaptation.

- Transparency: The work is carried out in full trust of all parties involved. Everyone has the courage to keep each other up to date with both good and bad news.
- Inspection: Inspection is carried out by every one in the Scrum Team. The team openly shows the product at the end of each Sprint.
- Adaptation: The team asks constant questions about the progress of work, whether we are on the right way. Depending on this, we can adapt an existing product.

At the beginning of the project, we have discussed and estimated all the work that needs to be done. Meetings between supervisor and developer are bi-weekly and sometimes weekly. Each meeting includes a discussion about what has been achieved since the tasks have been assigned, what can be improved, and scheduling of future tasks.

i. Scrum Roles

The Scrum Roles have been defined as follows:

- Product Owner: Mr. Pfahrer acts as the owner of the product. He is also responsible for feedback on the work, and overseeing the work process.
- Development Team: Since there is only one developer, this role is assigned to Shiryagina Kristina.
- Scrum Master: Shiryagina Kristina takes on this role.

The Thesis expert Federico Flueckiger takes a part in the meetings thus has the role of stackenolder.

ii. Scrum Plan

To discuss the project, were weekly and biweekly meetings held . They included personal meetings or meetings using Microsoft Teams. The meetings consisted of:

- Sprint Review. It includes a show of work and its discussion.
- Sprint Planning. It includes the scheduling of future tasks.
- Sprint Retrospective. It includes discussion about what went well and what went wrong, what we should do differently.

iii. Scrum Artifacts

Gitlab from the BFH school was used for the project, which makes it easy to figure out what has already been done and what else is in the process of work, as well as make plans for sprints. The figures below shows that with the help of gitlab the Product Backlog and Sprint backlog was created. **Product Backlog** contains everything that product

needs. **Sprint Backlog** is a set of items selected from Product backlog for working on concrete sprint. For each task there is a label "TODO", "DOING" and "DONE", which facilitates the overall control of the work.

Figure 1: Gitlab boards with issues

An Excel document was also used to organize the project. This document displays all data on the deadlines, goals of the sprint, duration of sprints, as well as tasks for a specific sprint, description, duration, and their prioritization.

BaThesis BTI7321 20/21, HS, Project Planning				
Sprint Calendar				
Semester week	MO	FR	sprint N	merge-date
1	14-Sep	18-Sep		
2	21-Sep	25-Sep		
3	28-Sep	2-Oct	1	
4	5-Oct	9-Oct		11-Oct
5	12-Oct	16-Oct		
6	19-Oct	23-Oct	2	
7	26-Oct	30-Oct		
8	2-Nov	6-Nov		8-Nov
9	9-Nov	13-Nov	3	
10	16-Nov	20-Nov		
11	23-Nov	27-Nov		
12	30-Nov	4-Dec		
13	7-Dec	11-Dec		13-Dec
14	14-Dec	18-Dec	4	
Holidays	21-Dec	25-Dec		
Holidays	28-Dec	1-Jan		7-Jan book --> Mr.Pfahrer
Holidays!	4-Jan	8-Jan		10-Jan
16	11-Jan	15-Jan	5	
	18-Jan	22-Jan		21-Jan
	25-Jan	29-Jan		
	1-Feb	5-Feb		
	8-Feb	12-Feb		

Ablauf der Bachelor-Thesis / Milestones

Until this date I have to prepare all documents and also the application
All documents are:
1. Report
2. Final presentation till 1-Feb
3. Small presentation
4. Poster
5. Video
* have to be prepared for the 1st Presentation -"Sale Product" Presentation

Figure 2: Sprints Calendar

Sprint Backlog(Sprint1)							hours per day	4	deliverable_sprint 1:	
ID	Task Name	Task Description	days-	sum of	Priority	Estimated Effort	Actual Eff. Status	hours per day	hs sprint1	Goals
1	Infrastructure	Set up infrastructure GitLab Create project management document	0.4	0.4	High	2	2	done	102.9	
2	Management tools		1.9	2.3	High	10	11	done		
3	Create a user stories	Create and enter a user stories in product backlog	1.0	3.3	High	5	6	done		
4	Create the tasks	Break down user stories to the tasks for sprint	0.4	3.7	High	2	3	done		
5	User stories	Make repositories	3.9	7.6	High	20	22	done		
6	User stories	Add controllers for crud operations	3.9	11.5	High	20	25	done		
7	User story	Make graphical concepts(start)	1.9	13.4	High	10	10	done		
8	Report	Add introduction	1.4	14.8	Medium	7	7	done		
9	Report	Add background to the introduction	0.6	15.4	Medium	3	1	doing		
10	Add Semester View	Start with views	1.4	16.7	Medium	7	8	doing		
11	Frontend-Backend	Add classes to communicate view with controller	1.9	18.7	Medium	10	12	done		
12	Report	Add a document structure	0.3	19.0	Medium	1.5	1.5	done		
13	Milestone	delivery of sprint1	11-Oct	Milestone		0				
1	The main aspects of Sprint 1 were:									
2	1. Creation of project management.									
3	2. Define the user stories									
4	3. Creating of document structure.									
5	4. Creating concept of User Interface									

Figure 3

iv. Sprints

Conclusion Project Management

If to look closely at each sprint in the document https://gitlab.ti.bfh.ch/shirk1/planning-of-the-assignments-for-lecturers-plana/-/tree/master/3project_management", it is east to see that some tasks have been moved from one sprint to the next. This suggests that not all tasks were completed on time, that is, first of all, attention was paid to tasks with high importance. This is due to the fact that some tasks may vary significantly over time. therefore, the prioritization system is in demand in this case. There was a noticeable delay during the implementation of the code for the program, which I could not solve for several days, in this case I turned to a colleague for help, who explained to me what I needed to change in the configuration. And I think that school projects are always associated with a risk that they will not meet the deadline, since there is a constant study of new technologies, and working alone is always inferior to teamwork. On the other hand, dividing work into sprints allows you to have more clear control over the overall situation.

3 System requirements

3.1 Product Backlog and User Stories

In project 2, the main user stories were identified, in this work I supplemented this list in accordance with the new requirements.

Epic			
As a Lecturer I want to be able to create my own medium and long-term assignment plans in form of requests and proposal for the definite plan and manage it so that it will be possible a mutual development of the main assignment plan.			
ID	User Story Name	User Story	Acceptance Criteria
7.0	list modules	As a Lecturer, I want to see the list of modules for a concrete semester, so that I can choose the modules I want to plan in my own plan.	User is able to: <ul style="list-style-type: none"> • navigate to his plan page • able to see the module list
5.0	add modules to my assignment plan	As a Lecturer, I want to be able to add some of the modules to my assignment plan I want to teach in a specific semester or remove it from my plan so that I can participate in the main planning by making suggestions or requests.	User is able to: <ul style="list-style-type: none"> • navigate to his plan page • able to select a module and set himself to it
08	manage my plan	As a Lecturer, I want to be able to manage my plan, so I can modify some data in my plan.	User is able to: <ul style="list-style-type: none"> • navigate to his plan page • able to modify some data of his planning

Table 2: Product Backlog

Epic			
As a Study Director, I want to be able to make a group of lecturers and attach it to a specific module, and also make a group of modules so this will increase planning flexibility. As a Study Director, I want to be able to attach a specific group of modules to a specific group of teachers so that further joint planning of these modules will be easier			
ID	Story Name	User Story	Acceptance Criteria
09	See the requests for the modules	As a Study Director, I want to see proposal for the selected modules so that it will be easier to make assignment plan.	User is able to: <ul style="list-style-type: none"> open a list of modules see the lecturers requests
10	See the requests for the groups	As a Study Director, I want to see teachers suggestions for the group work so that it will be easier to approve specific groups and make assignment plan.	User is able to: <ul style="list-style-type: none"> open a list of lecturers group see the changes made by lecturers
11	See the working hours request for the concrete modules	As a Study Director, I want to see teachers proposal for working hours for concrete module so it will be easier to set working hours to each lecturer.	User is able to: <ul style="list-style-type: none"> open a plan see the hours requests
12	Make groups of lecturers	As a Study Director, I want to be able to make groups of lecturers in the assignment plan.	User is able to: <ul style="list-style-type: none"> open the main planning page select list of teachers select several teachers and save them as a group.
13	Set groups of lecturers to the modules	As a Study Director, I want to be able to set group of lecturers to the assignment plan.	User is able to: <ul style="list-style-type: none"> open the main planning page select the group of lecturers and attach it to a specific module.
14	Make groups of modules	As a Study Director, I want to be able to make groups of modules in the assignment plan.	User is able to: <ul style="list-style-type: none"> open the main planning page select list of modules select several modules and save them as a group.

ID	User Story Name	User Story	Acceptance Criteria
15	Attach Lecturers to the module	As a Study Director, I want to be able to attach lecturer or group of lecturers to the specific module in the assignment plan.	<p>User is able to:</p> <ul style="list-style-type: none"> • open the main planning page • select lecturer • click add lecturer button and save it to the module
16	Attach Lecturers to the module group	As a Study Director, I want to be able to attach group of lecturers to the specific module in the assignment plan.	<p>User is able to:</p> <ul style="list-style-type: none"> • open the main planning page • select and add lecturer to the group of modules and save it
17	Attach Lecturers Group to the Module Group	As a Study Director, I want to be able to attach group of lecturers to the group of modules in the assignment plan.	<p>User is able to:</p> <ul style="list-style-type: none"> • open the main planning page • select and add lecturers group to the group of modules and save it

Table 4: Product Backlog

Epic			
As a Lecturer, who joins a group, I want to be able to independently manage the tasks related to the common module.			
ID	User Story Name	User Story	Acceptance Criteria
14	Manage common modules	As a Lecturer, who joins a group, I want to be able to independently manage the tasks related to the common module.	User is able to: <ul style="list-style-type: none"> • Open page with common modules • Manage his tasks related to him.

Table 5: Product Backlog

The Figure below shows all identified user stories with their prioritization. There is an identifier in priority weight from 1 to 10. A higher priority weight indicates that the corresponding user story is more important for the project's success and for meeting customer needs.

Product Backlog			
ID	Actor	Task Name	Priority
1	Study Director	Create a plan of assignments	10
2	Study Director	Manage plan of assignments	10
3	Study Director	Add new module to the plan	10
4	Study Director	Add lecturer to the module	10
5	Study Director	Publish plan for lecturers	10
6	Study Director	Officially publicize a plan	10
7	Study Director	Set lecturer's hours to for module	10
8	Study Director	See the assignment plan for a specific year	10
9	Study Director	See the requests for the modules	10
10	Study Director	See the requests for the groups	10
11	Study Director	See the working hours requests for the concrete module	10
12	Study Director	Make a groups of lecturers	10
13	Study Director	Set the group of lecturers to the module	10
14	Study Director	Make groups of modules	10
15	Study Director	Attach the lecturers to the module	10
16	Study Director	Attach the lecturers to the module group	10
17	Study Director	Attach the lecturers group to the modules group	5
18	Study Director	See last year plan	3
19	Study Director	Make a copy from last year plan	5
20	Study Director	See hours conflicts	7
21	Lecturer	See the plan of my assignments for concrete year	10
22	Lecturer	Set my hours for concrete module	10
23	Lecturer	Add new modules to my plan	10
24	Lecturer	Add new additional assignments to my plan	10
25	Lecturer	See last year plan	3
26	Lecturer	Publish my plan	10

Figure 4: Product Backlog

4 Functional Requirements Specifications

4.1 Stakeholder Summary

The stakeholders of our system are:

- **Study director**

The first major stakeholder is a study director. The study director can be both the person who makes an assignment plan and also a lecturer. He participates in planning, and is the person who take control and main responsibility for planning.

- **Lecturer**

The lecturer is another major stakeholder of our system. He participates in assignment planning.

4.2 Actors and Goals

The roles of people that will directly interact with the system and their goals.

- actor -Study director
 - Initial Actor
 - goal Uses system to make a plan of assignments for lecturers.
- actor -Lecturer
 - Initiating Actor
 - **goal** Uses the system to participate in the planning of his assignments .
 - **goal** Uses system to see his assignment plan .
 - **goal** Uses system to make a requests for the modules he/she wants to teach.
 - **goal** Uses system to make a request for the teaching hours for a concrete module.

4.3 c. User Interface Requirements

In the project2 I did the main part of the Functional Requirements .In this work I pay special attention to the development the user interface concepts. All requirements are based on consumer needs. There is an identifier in the form of REQ-x and a priority weight from 1 to 10. A higher priority weight indicates that the corresponding requirement is more important for the project's success and for meeting customer needs.

Identifier	Priority Weight	Requirements
REQ-1	1	UI must have a landing page (log in).
REQ-2	10	UI must have study director start page.
REQ-3	10	UI must have a page where the study director can make a new plan.
REQ-4	10	UI must have a page to add a module.
REQ-5	7	UI must have a page to load existing module.
REQ-6	10	UI must have a page to create a new module .
REQ-7	10	UI must have a page to add lecturers or group of lecturers to the module.
REQ-8	7	UI must have a page with copied assignments from last year.
REQ-10	10	UI must have a page to see the lecturer's assignments plan
REQ-11	10	UI must have a page to add a module to the lecturer's assignment plan
REQ-12	2	UI must have a lecturer's page to see the last year assignment plan
REQ-13	5	UI must have a page to see the groups of lecturers to the corresponding module/module group
REQ-14	10	UI must have a page to see and manage the plan after editing of lecturers

Table 6: User Interface Concepts

5 System Architecture and System Design

In project 2 describes the System architecture and design. In this work this chapter is supplemented with new diagrams and descriptions.

5.1 a. Architecture Styles according business logic

Applications are designed to provide a variety of services to solve specific problems. And each problem has a list of rules (business rules). That is, the developer writes the code to implement the business rule. This code is business logic. There are various techniques and patterns for handling business logic. [3]

In this application is used a layered architecture. The pattern is used to make development of database access faster and to make code cleaner. And this pattern based on the domain-driven design(DDD) pattern from Eric Evans. But in this case the business logic is not in the entity classes but in the Business Logic Service layer.

The following list shows the guidelines that build the business logic pattern of Plana application. **The guidelines that build the business logic**

- Low coupling between layers, high cohesion within them.
- Separation of concerns.
- Adaptability: be able to change.
- No circular references between layers
- Lower layers should not depend on upper layers.

[4]

The Figure below shows the structure of application's back-end.

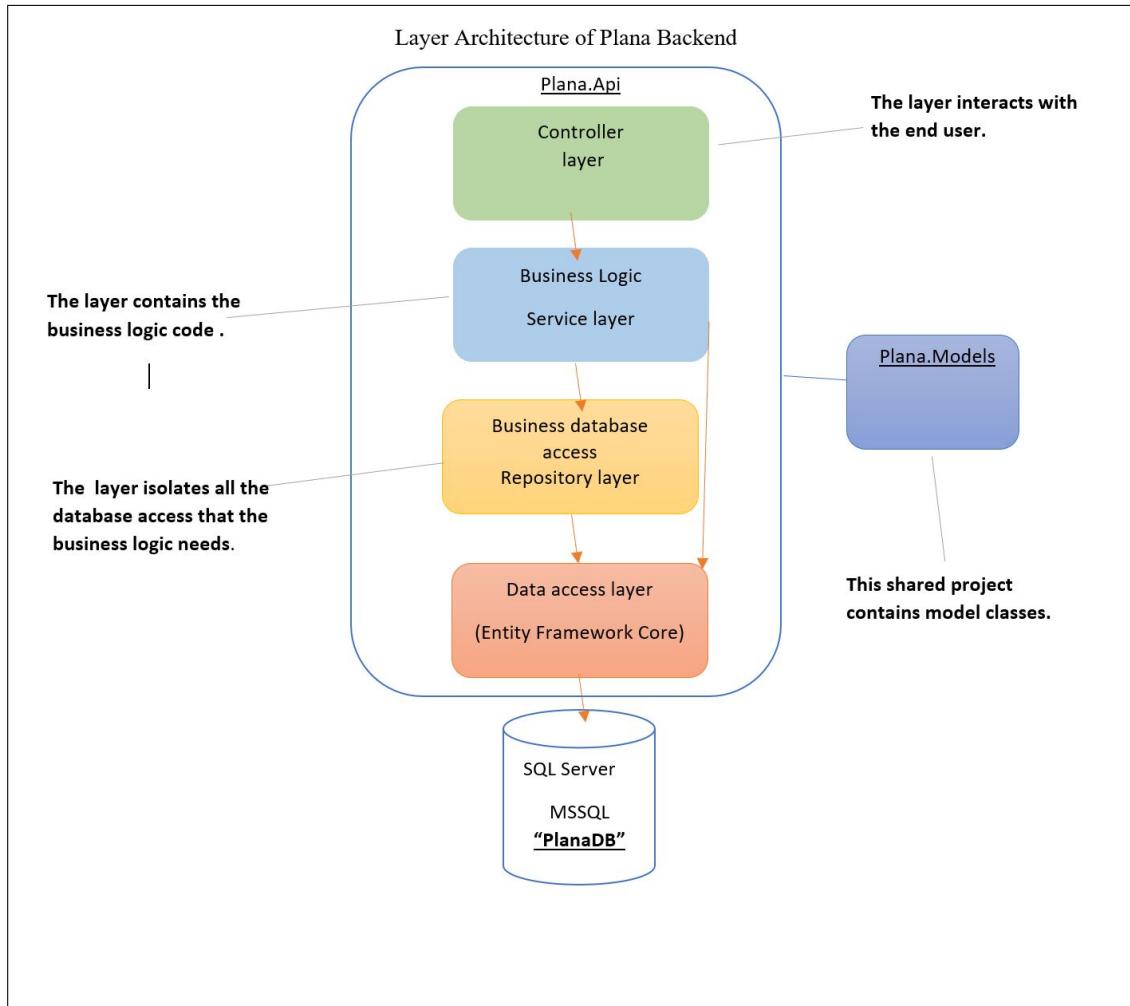


Figure 5: Layer Architecture of Plana Back-end

Layer Name	Description
Data access layer	The layer stores and retrieves the information from a database
Business database access layer	The layer performs database operation using EF Context object.
Business Logic Service layer	The layer makes logical decisions, processes commands
Presentation layer	The layer contains the code for the Web API endpoint logic. It contains a minimal business logic.[5]

Table 7: Layers Description

Layers shall to be loosely coupled and design dependency in only one direction. [6] [3]

5.2 The Plana application Architecture

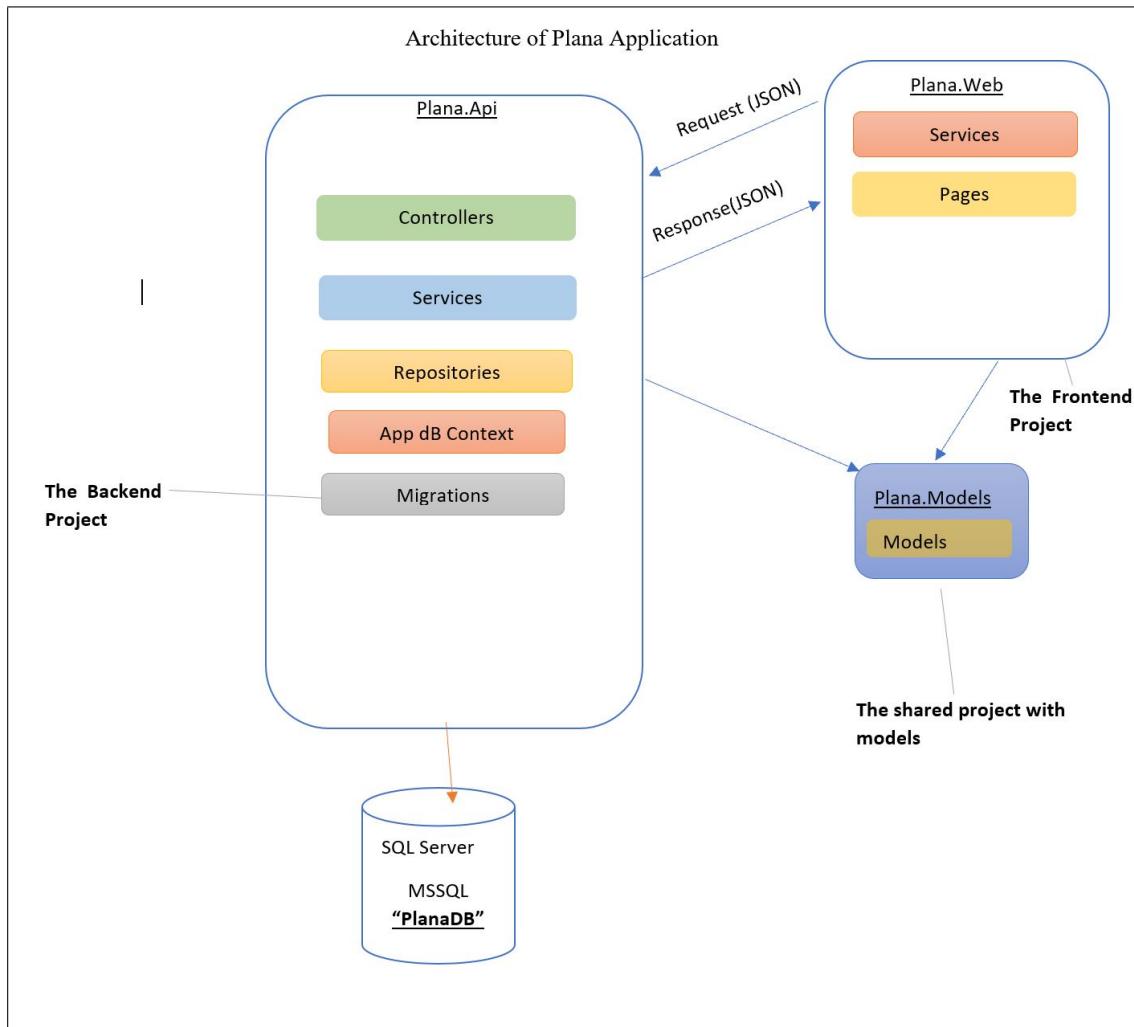


Figure 6: Layer Architecture of Plana Application

Components	Description
Plana.Api Project	
Controllers	Controllers - define the end point into the web api from client application via http requests. [7]
Services	This folder contains classes that are communicate with the data layer and controllers. These classes contain the main business logic
App dB Context	An instance of DbContext represents a session with the database which can be used to query and save instances of your entities to a database. DbContext is a combination of the Unit Of Work and Repository patterns.[8]
Migrations	This folder includes Entity Framework Core generated migration classes
Plan.Web Project	
Pages and Shared	The Pages and the Shared folders contain .razor files, which are the Blazor components [9]
Plana.Models Project	
Models	This folder holds the application domain objects, the Code First classes. [10]
Plana.Dtos Project	
DTO models	in this application DTO has role of the view model ,which is used to transfer data between the domain model and the view, it contains User Interface logic

Table 8: Plana Projects Components

6 User Interface Specification

*Note: only selected Interface designs are presented below because they are the main functions of the application.

6.1 a. Preliminary Design

- i. Login
- ii. Study director's dashboard

Once the study director successfully logged in, the personal dashboard page should be shown directly

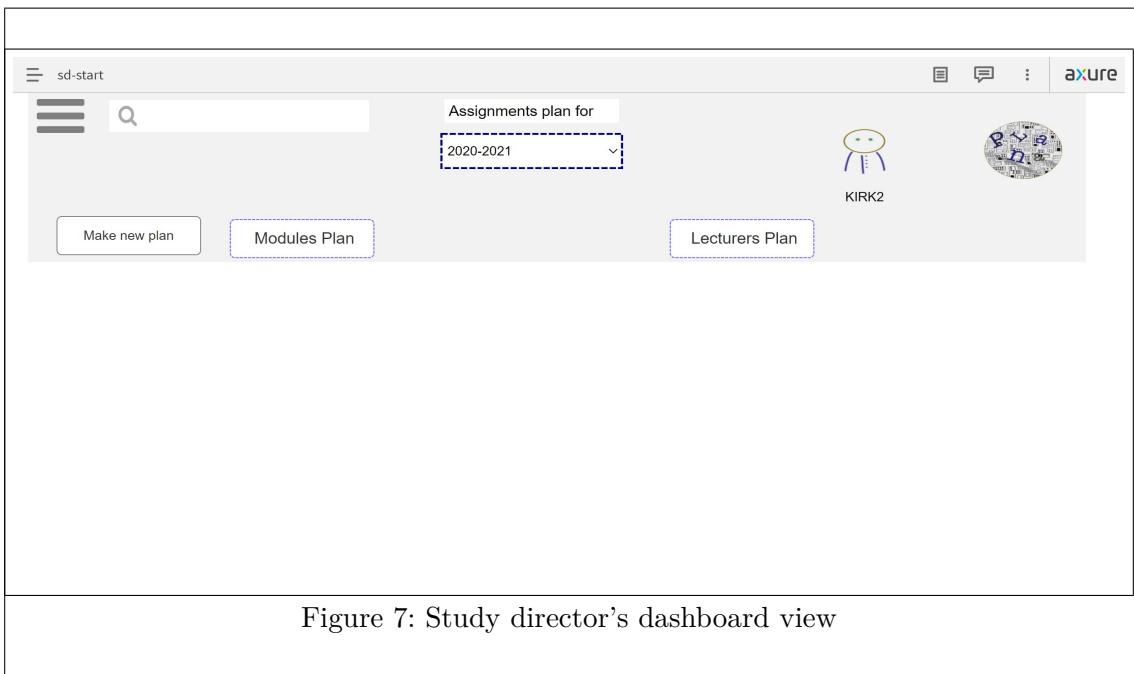


Figure 7: Study director's dashboard view

iii.Create a plan

Once the personal dashboard is displayed, user could click on the "Make a new plan" button, and add new module page will be shown where study director can make a new assignments plan.

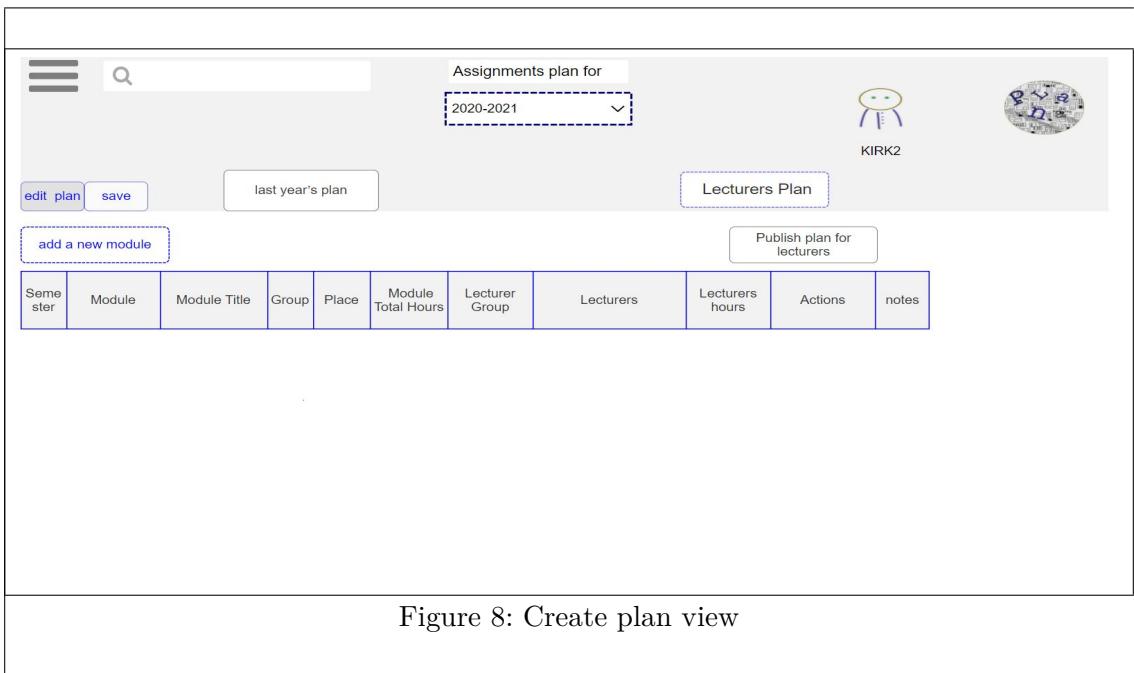


Figure 8: Create plan view

iv. Add a new module

Study director could click on the "add new module" button, and a window with buttons of adding or creating a module will be shown.

The screenshot shows a user interface for managing course modules. At the top, there's a header with a search icon, a dropdown for 'Assignments plan for' showing '2020-2021', and a user profile for 'KIRK2' with a circular icon. Below the header are several buttons: 'edit plan', 'save', 'last year's plan', 'Lecturers Plan', and 'Publish plan for lecturers'. A main table has columns for Semester, Module, Module Title, Group, Place, Module Total Hours, Lecturer Group, Lecturers, Lecturers hours, Actions, and notes. A modal window is open, containing fields for 'add existing module' (with a red 'X' icon), a dropdown for 'Modern Frameworks', a 'save' button, and a 'add new module' button. The entire interface is framed by a large rectangular border.

Semester	Module	Module Title	Group	Place	Module Total Hours	Lecturer Group	Lecturers	Lecturers hours	Actions	notes
----------	--------	--------------	-------	-------	--------------------	----------------	-----------	-----------------	---------	-------

add a new module

Assignments plan for
2020-2021

KIRK2

edit plan save last year's plan Lecturers Plan Publish plan for lecturers

add existing module

Modern Frameworks

save

add new module

Figure 9: Add a new module view

v. Possibility of adding a module or group of modules.

Study director could click on the "add an existing module" button, and a drop down menu with possible modules will be displayed, then he could choose one of them and click button "save" and the chosen module will be displayed in the table. Or study director can choose "add new module button" and the row with an editable fields will be displayed.

Semester	Module	Module Title	Group	Place	Module Total Hours	Lecturer Group	Lecturers	Lecturers hours	Actions	notes
HS	BTI1077	Modern Frameworks	a	Biel	200	<input type="button" value="+"/>	<input type="button" value="+"/>	<input type="text"/>	<input type="button" value="save"/>	note
..

Semester	Module	Module Title	Group	Place	Module Total Hours	Lecturer Group	Lecturers	Lecturers hours	Actions	notes
						<input type="button" value="+"/>	<input type="button" value="+"/>	<input type="text"/>	<input type="button" value="save"/>	note

Figure 11: Add module/ modules group view

vi. Possibility of adding a lecturer or group of lecturers.

Study director could click on the "+" button that represents an add button and the windows with different buttons will be displayed. There are "add existing group" button

to add an existing group of lecturers, "add new group" button to create a new group of lecturers and it to the module/group of modules and the "add lecturer" button to add lecturer.

The screenshot shows a software interface for managing assignments. At the top, there's a header with a search bar, a date selector set to '2020-2021', and user information for 'KIRK2'. Below the header, there are buttons for 'edit plan' and 'save', and links for 'last year plan' and 'Lecturers Plan'. A 'Publish plan for lecturers' button is also visible. The main area is a table titled 'Assignments plan for' with columns for Semester, Module, Module Title, Group, Place, Module Total Hours, Lecturer Group, Lecturers, Lecturers hours, Actions, and notes. Several rows are listed, each with a 'save' button and a 'note' link. A tooltip 'add lecturer' points to a 'GROUP TITLE' input field containing 'LG_10'. Another tooltip 'add new group' points to a 'LECTURERS' input field containing 'MASK1'. A third tooltip 'add existing group' points to a 'LECTURER GROUP' input field.

Semester	Module	Module Title	Group	Place	Module Total Hours	Lecturer Group	Lecturers	Lecturers hours	Actions	notes
HS	BTI1021	Computer Science Basics	a	<input type="text" value="x"/>	...	<input type="button" value="+"/> <input type="button" value="add existing group"/>	<input type="button" value="+"/>		<input type="button" value="save"/>	note
FS	BTI1021	Computer Science Basics	p	<input type="text" value="t"/>		<input type="button" value="+"/> <input type="button" value="add new group"/>			<input type="button" value="save"/>	note
HS	BTI7321	Bachelor-Thesis	p/c	<input type="text" value="p/c"/>		<input type="button" value="+"/> <input type="button" value="add lecturer"/>			<input type="button" value="save"/>	note
FS	BTI17333	New Module	a	<input type="text" value="MASK1"/>		<input type="button" value="+"/> <input type="button" value="save"/>			<input type="button" value="save"/>	note
HS	BTI3002	Project and Training 1	a	Biel	200				<input type="button" value="save"/>	note
FS

Figure 12: Add lecturer/lecturers group view

vii. See and copy last year plan.

The study director could click on the button "last year plan" to see the last year's plan. Then he could copy the whole plan clicking the button "copy whole plan to the actual plan" or he could choose the modules he likes and copy it to the actual plan clicking the button "copy to the actual plan".

Semester	Module	Module Title	Group	Place	Module Total Hours	Lecturer Group	Lecturers	Lecturers hours	Action
HS	BTI1021	Computer Science Basics	a	Biel	200	LG_01	MIWH2 BOPE1	150 50	<input checked="" type="radio"/>
FS	BTI1021	Computer Science Basics	p	Bern	200	LSG_13	SHAW1 MIWH2	150 50	<input type="radio"/>
HS	BTI7321	Bachelor-Thesis	p/q	Bern	28 h/stud	LG_04	KAUG PEGE	28 56	<input checked="" type="radio"/>
HS	BTI3002	Project and Training 1	a	Biel	200	LG_05	VAKI2 DASA1	75 125	<input type="radio"/>
FS	<input type="radio"/>

	Module	Module Title	Modul e-Group	Place	Module Total Hours	Lecturer Group	Lecturers	Lecturers hours	Action	notes
HS	BTI1021	Computer Science Basics	a	Biel	200	LG_01	MIWH2 BOPE1	150 50	<input checked="" type="checkbox"/>	note
FS	BTI1021	Computer Science Basics	p	Bern	200	LSG_13	SHAW1 MIWH2	150 50	<input checked="" type="checkbox"/>	note
HS	BTI7321	Bachelor-Thesis	p/q	Bern	28 h/stud	LG_04	KAUG PEGE	28 56	<input checked="" type="checkbox"/>	note
HS	BTI3002	Project and Training 1	a	Biel	200	LG_05	VAKI2 DASA1	75 125	<input checked="" type="checkbox"/> <input type="checkbox"/>	note
FS	BTI1733	New Module	a	Biel	200	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	note
HS	<input type="checkbox"/>	..

Figure 14: Copy last year plan.

Study director can use this page to make a plan and could publish it for the lecturers by clicking the button "publish on 18 July".

viii. Lecturer's dashboard

Once the lecturer successfully logged in the dash board page is shown. And if the study director has already published the plan, the lecturer's assignment plan will be displayed. The dash board contains also buttons with different functions. Here the lecturer can manage his assignments.

Assignments plan for
2020-2021

LEON3

Archive
My plan

Assignments plan for
2020-2021

LEON3

submit plan
submit until
12 July 2020

add new module
add new additional assignment
My plan
show lecturer groups
last year plan

Semester	Module	Module Title	Group	Place	Module Total Hours	My hours	Actions	TOTAL				
HS	BTI3001	Project and Training 1	a	Biel	300	100		HS	FS	AA	YEAR	
HS	BTI1021	Computer Science Basics	a	Biel	200	150		484	400	800	1684	
HS	BTI1021	Computer Science Basics	p	Bern	200	50		ANNUAL TARGET 1600				
HS	BTI7321	Bachelor-Thesis	p	Bern	28 h/stud	84		BALANCE ACTUAL 84				
HS	BTI7544	Scala	a/b/q/p	Biel	100	100		BALANCE 19/20 -50				
FS	BTI3002	Project and Training	a	Biel	200	150		BALANCE ACCUMULATED 34				
FS	BTI7501	Spieltheorie	a	Biel	100	100						
FS	BTI7533	Praxis Startup	a/b	Biel	100	100						
FS	MG_BTI7187_BTI7 191	Project6_Project7	p/q	Bern	400	50						
AA		aF&E				800						

Figure 16: Lecturer's dash board view with opened plan

30

ix. Add new module

Lecturer could click on the "add new module" button and UI would show the window with possible modules. The lecturer could add him to the module and set desired hours for the module. The lecturer could click the button "add new additional assignment" and the window with possible additional assignment will be displayed. And like with modules he can add desired assignment to his plan and set the hours.

Semester	Module	Module Title	Group	Place	Module Total Hours	Add me	Add group	My hours
HS	BTI7321	Bachelor-Thesis	p	Bern	28 h/stud			<input type="text" value="28"/>
FS	BTI7321	Bachelor-Thesis	a	Biel	28 h/stud			
HS	BTI7089	New Module	a/b	Biel	250			
FS	MG_BTI7981– BTI7982	Project3+Project4	p/q	Bern	400			

Additional Assignment Type	Additional Assignment Title	Add me	My hours
Research	aF&E		<input type="text" value="750"/>
Teaching	FB-Pool		<input type="text"/>

Figure 18: Add a new additional assignment.

x. Lecturers groups

Lecturer could click on the 'show lecturer groups' and the window with lecturers corresponding to the module will be displayed".

The screenshot shows the 'Assignments plan for 2020-2021' interface. On the right, there is a sidebar titled 'Lecturer Group' containing a list of groups: G_pat1_a, G_csb_a, G_csb_p, and several empty slots. A note window is open for group G_pat1_a, listing 'Karim Uglu', 'Peter Villiger', and 'Danielle Peltre'. The main table lists various modules with their details and lecturer groups.

Semester	Module	Module Title	Group	Place	Module Total Hours	My hours	Actions
HS	BTI3001	Project and Training 1	a	Biel	300	100	note
HS	BTI1021	Computer Science Basics	a	Biel	200	150	note
HS	BTI1021	Computer Science Basics	p	Bern	200	50	note
HS	BTI7321	Bachelor-Thesis	p	Bern	28 h/stud	84	note
HS	BTI7544	Scala	a/b/q/p	Biel	100	100	note
FS	BTI3002	Project and Training	a	Biel	200	150	note
FS	BTI7501	Spieltheorie	a	Biel	100	100	note
FS	BTI7533	Praxis Startup	a/b	Biel	100	100	note
FS	MG_BTI7187_BTI7_191	Project6_Project7	p/q	Bern	400	50	note
AA		aF&E				800	note

xi. Lecturer could click the link "note" and the note windows will be displayed.

The screenshot shows the 'Assignments plan for 2020-2021' interface. A note window is open for group G_pat1_a, listing 'Karim Uglu', 'Peter Villiger', and 'Danielle Peltre'. To the right, a summary table titled 'TOTAL' provides an overview of the lecturer's performance.

	AA	YEAR
notes	800	1684
ANNUAL TARGET	1600	
BALANCE ACTUAL	84	
BALANCE 19/20	-50	
BALANCE ACCUMULATED	34	

Figure 20: See the notes view.

xii. Lecturer's last year plan

Lecturer could click on the button "last year plan" and the window with the last year plan will be displayed.

Archive
add new module
add new additional assignment
My plan
show lecturer groups
last year plan
submit until
12 July 2020

Assignments plan for 2020-2021							
Semester	Module	Module Title	Group	Place	Module Total Hours	My hours	Actions
HS	BTI3001	Project and Training 1	a	Biel	300	100	edit
HS	BTI1021	Computer Science Basics	a	Biel	200	150	edit
HS	BTI1021	Computer Science Basics	p	Bern	200	50	edit
HS	BTI7321	Bachelor-Thesis	p	Bern	28 h/stud	84	edit
HS	BTI7544	Scala	a/b/q/p	Biel	100	100	edit
FS	BTI3002	Project and Training	a	Biel	200	150	edit
FS	BTI7501	Spieltheorie	a	Biel	100	100	edit
FS	BTI7533	Praxis Startup	a/b	Biel	100	100	edit
FS	MG_BTI7187_BTI7_191	Project6_Project7	p/q	Bern	400	50	edit
AA		aF&E				800	edit

TOTAL

HS	FS	AA	YEAR
484	400	800	1684

ANNUAL TARGET **1600**
BALANCE ACTUAL **84**
BALANCE 19/20 **-50**
BALANCE ACCUMULATED **34**

TOTAL

HS	FS	AA	YEAR
400	400	750	1550

ANNUAL TARGET **1600**
BALANCE ACTUAL **-50**
BALANCE 18/19 **0**
BALANCE ACCUMULATED **-50**

Figure 21: See last year plan.

xiii. Seeing the data after lecturers input.

The lecturer could submit plan by clicking the "submit plan" button. After that the study director could see the page with the plan after lecturers editing by clicking "Modules Plan" button. In this page he could manage the plan.

The screenshot shows a web-based application for managing assignment plans. At the top, there's a header with a search icon, a dropdown menu for the academic year (set to '2020-2021'), and user icons for 'KIRK2' and a profile picture. Below the header are buttons for 'edit plan' and 'save', and a link to 'Modules Plan'. A red dashed box highlights a button labeled 'Publish on 18 July'.

The main area is a table titled 'Assignments plan for' with the following columns:

- Module
- Module Title
- Module -Group
- Place
- Module Total Hours
- Lecturer's Group
- Lecturers
- Set Lecturers hours
- Actions
- notes

The table contains several rows of data:

- HS BTI1021 Computer Science Basics**: Module Group 'a', Place Biel, Total Hours 200. Lecturer's Group LG_01. Lecturers MIWH2, BOPE1. Set hours 50, 150. Actions include a blue 'save' button and a small edit icon.
- FS BTI1021 Computer Science Basics**: Module Group 'p', Place Bern, Total Hours 200 (to plan 150). Lecturer's Group LG_13. Lecturer MIWH2. Set hours 50. Actions include an edit icon.
- HS BTI17321 Bachelor-Thesis**: Module Group p/q, Place Bern, Total Hours 28 h/stud. Lecturer's Group LG_04. Lecturers IVDO1, KAUG, PEGE. Set hours 84, 56, 28. Actions include an edit icon and a small note icon.
- HS BTI3002 Project and Training 1**: Module Group 'a', Place Biel, Total Hours 200 (to plan 75). Lecturer's Group LG_05. Lecturers VAK2, DASA1. Set hours 125, 0. Actions include an edit icon and a small note icon.
- FS BTI17333 New Module**: Module Group 'a', Place Biel, Total Hours 200 (to plan -75). Lecturer's Group LG_06. Lecturers LEWH3, MADI1, MIYO2. Set hours 125, 75, 75. Actions include an edit icon.
- FS MG_BTI1718_7_BTI17191 Project6_Project**: Module Group p/q, Place Bern, Total Hours 400. Lecturer's Groups LG_08, LG_11. Lecturers HADA1, GELT1, PEBO2, PESA2. Set hours 50, 150, 100, 100. Actions include an edit icon.
- FS ...**: Module Group '...', Place '...', Total Hours '...'. Lecturer's Groups '...', '...'. Lecturers '...', '...'. Set hours '...'. Actions include an edit icon and a small note icon.

Below the table, the text 'Figure 22: The plan view after lecturer's input' is centered.

xiv. Make a note

The study director could make a note clicking note link corresponding to the certain module.

Assignments plan for
2020-2021

KIRK2

edit plan save Modules Plan Lecturers Plan Publish on 18 July

	Module	Module Title	Module -Group	Place	Module Total Hours	Lecturer's Group	Lecturers	Set Lecturers hours	Actions	notes
HS	BTI1021	Computer Science Basics	a	Biel	200	LG_01	[+] MIWH2 BOPE1	50 150	save	
FS	BTI1021	Computer Science Basics	p	Bern	200 to plan 150	LG_13	MIWH2	50		
HS	BTI17321	Bachelor-Thesis	p/q	Bern	28 h/stud	LG_04	IVDO1 KAUG PEGE	84 56 28		
HS	BTI3002	Project and Training 1	a	Biel	200 to plan 75	LG_05	VAK2 DASA1	125 0		
FS	BTI17333	New Module	a	Biel	200 to plan -75	LG_06	LEWH3 MADI1 MIYO2	125 75 75		
FS	MG_BTI718_7_BTI1791	Project6_Project 7	p/q	Bern	400	LG_08 LG_11	HADA1 GETI2 PEBO2 PESA2	50 150 100 100		

Figure 23: Make a note view.

7 Domain Analysis

7.1 Domain Model

The domain model (Figure 1) shows us the important concept classes, associations and multiplicities between them. The model made in the previous project is shown in black. And other colors show new concepts and associations associated with new tasks.

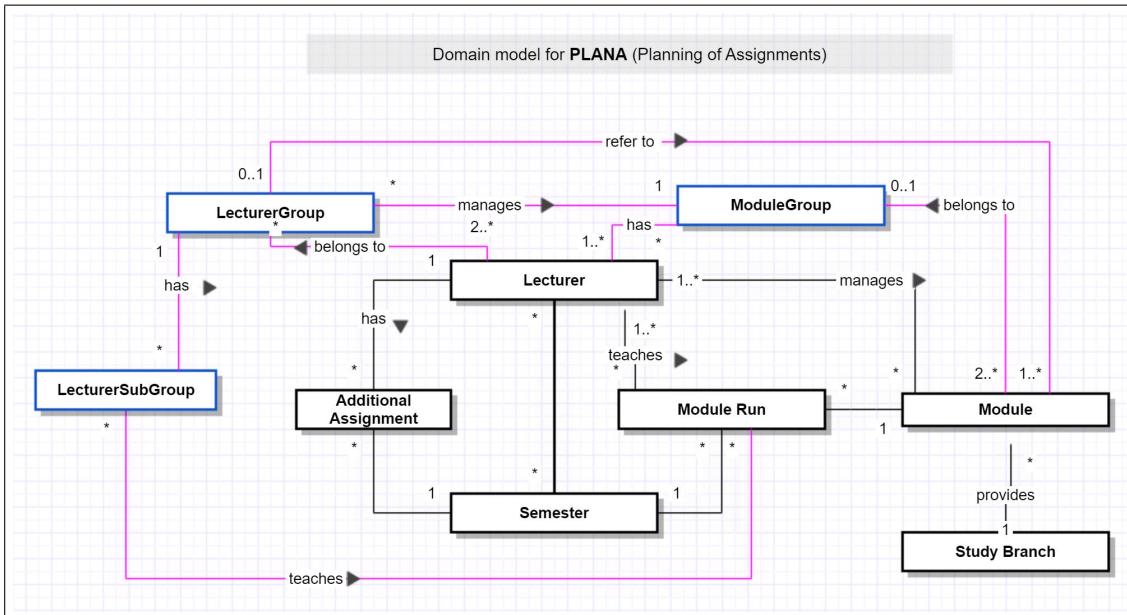


Figure 24: Domain Model for PLANA

In the domain model (Figure 2) we added small changes. When analyzing graphic concepts and the domain model, we found that it is necessary to add associations between the teacher and the teacher subgroup, since they are directly related.

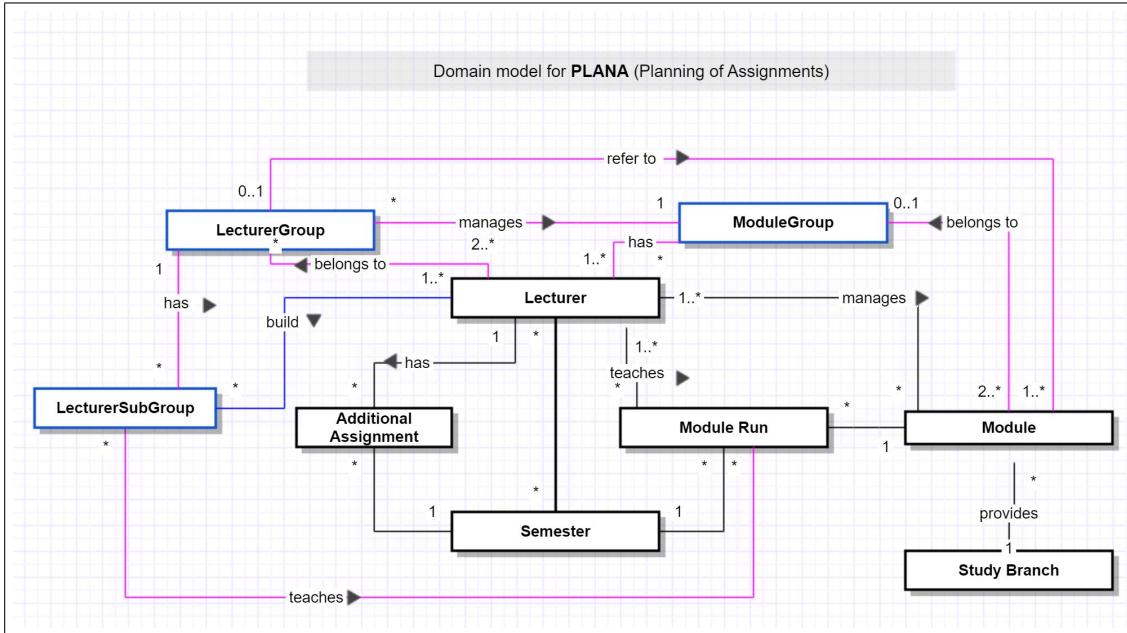


Figure 25: Domain Model for PLANA

The domain model (Figure 3) is the post-analysis Domain model. We have removed

the LecturerSubGroup since we can simplify the application. The LecturerGroup can consist of other LecturerGroups, that is, we create a relationship between them. Also we removed the links between the Module and Module Group and added it between the Module Run and the Module Group, since it is intended to create a groups of the Module Runs.

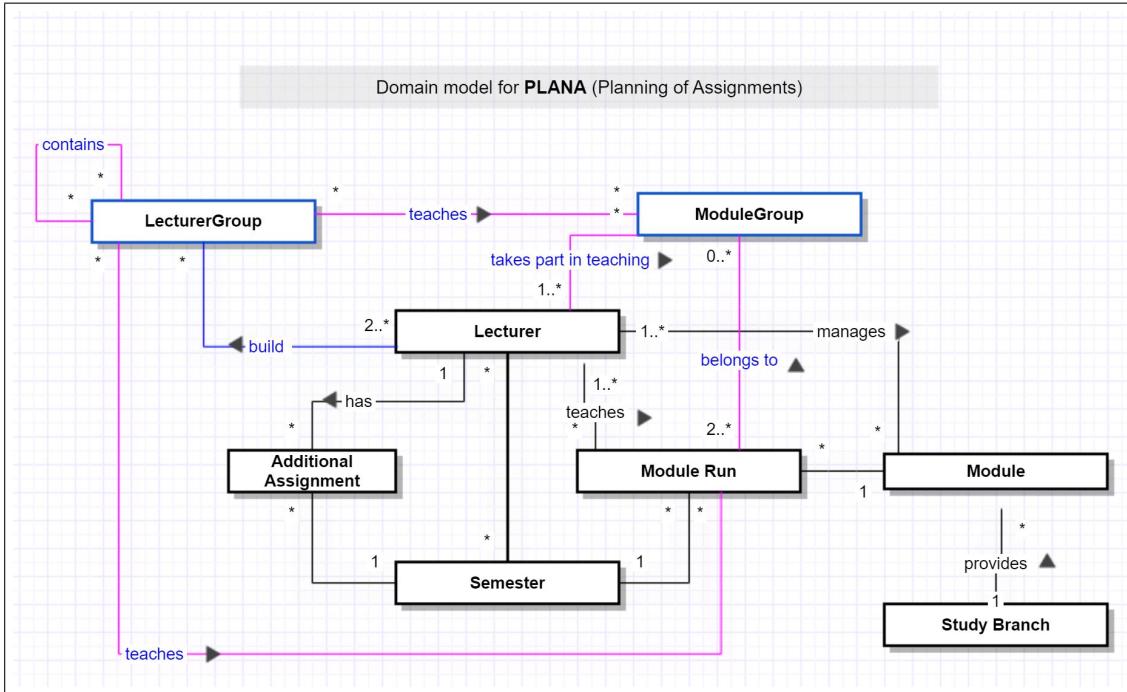


Figure 26: Domain Model for PLANA

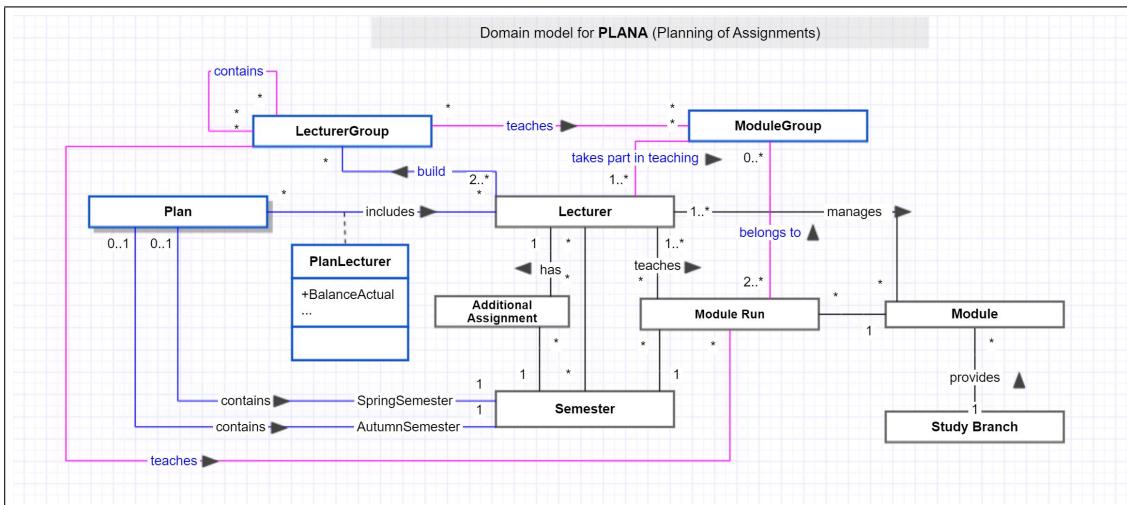


Figure 27: Domain Model for PLANA

In the definition of concepts, new concepts, associations between them and other concepts are highlighted in blue.

i. Concept Definition

- Concept class **Lecturer** models a person who teaches in a school.
- Concept class **Study Branch** models a conceptual subdivision of subjects that form a study program.
- Concept class **Module** models a set of independent units that form a course at the school.
- Concept class **Module Run** models executions of a course in different languages.
- Concept class **Additional Assignment** models a set of independent units that form an additional task for the assignment's plan for the lecturer. There are several types of the Additional Assignments. For example, teaching, research. Each teacher can plan his Additional Assignment and manage it.
- Concept class **Semester** models the periods in the year, during which the lecturer is present in the school.
- Concept class **LecturerGroup** models group of teachers who will jointly participate in teaching one or more modules.
- Concept class **ModuleGroup** models several modules collected in groups for further effective use.

ii. Association Definition

Concept Pair	Association Definition	Association Name
Lecturer -> Module Run	Lecturer can teach zero or more Module Runs . Each Module Run can be taught by one or more Lecturers	teaches
Semester -> Module Run	Semester can include zero or more Module Runs .	includes
Semester -> Additional Assignment	Semester can include zero or more Additional Assignments	includes
Lecturer -> Additional Assignment	Lecturer can have zero or more Additional Assignments	has
Lecturer -> Module	Lecturer can manage zero or more Modules. A module can be managed by one or more Lecturers	manages
Module -> Module Run	Module is executed as many as there are module runs or not executed at all. A Module run is executed for one Module.	executes
Study Branch -> Module	Each Study Branch has many modules. These modules belong to exactly one study branch.	has
Semester -> Lecturer	In each Semester, there are many lecturers that are teaching, and these teachers are teaching in more than one Semester	includes
Lecturer -> LecturerGroup	Each Lecturer build zero or multiple groups. Each LecturerGroup must consist of two or more lecturers.	build
LecturerGroup -> ModuleRun	Each LecturerGroup can have from zero to many ModuleRuns. Each ModuleRun can be teached by zero or multiple LecturerGroup	teaches
ModuleRun -> ModuleGroup	A ModuleRun can belong to zero or one ModuleGroup. ModuleGroup can have from two to many ModuleRuns.	belongs to
LecturerGroup -> ModuleGroup	A LecturerGroup can teach zero or multiple ModuleGroups. A ModuleGroup can have zero to many LecturerGroup. .	teaches
Lecturer -> ModuleGroup	Each Lecturer can take a part in teaching zero or many ModuleGroups. ModuleGroup can be teached by one or many Lecturers .	takes part in teaching
LecturerGroup -> LecturerGroup	Each LecturerGroup can consist of zero or multiple LecturerGroup and each LecturerGroup can be included in zero or multiple LecturerGroup .	consists

Table 9: Association Definition

8 Technologies

This chapter is described in project 2 report-roject2.pdf in Chapter 7, that can be found at the following link <https://gitlab.ti.bfh.ch/shirk1/planning-of-the-assignments-for-lectures-/tree/master/help%20documents>

9 Creating the Projects

In project 2 I started the implementation of the Plana application. I created the initial structure of the project, added domain classes and implemented the crud(Create, Update, Delete) operation for the teacher.

9.1 Structure of Projects and Folders

The structure of Projects as well the Data Models Project, Plana.API project, Plan.Web project as well as Plana Database were described in Project 2 in the document Visual Review In Initial Phase of Project.pdf at the following link <https://gitlab.ti.bfh.ch/shirk1/project2/-/tree/master/report-docs> In this work I added one more sub Project Plana.Shared, which contains the Data transfer object(DTO), the objects that are used to encapsulate data, and send it from one subsystem of an application to another.[11]

9.2 Database and Entity Framework Core

Entity Framework(EF) Core is an object-relational mapper (O /RM). It is designed to make writing code for accessing a database quick and intuitive. There are many good reasons to use EF Core. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with many databases, including SQL Database, SQLite, MySQL, PostgreSQL, and Azure Cosmos DB. book [12] [13]

9.3 The PLANA App's Relational Database

Our database has many types of relationships we can have in EF Core. The types are: One-to-many: Lecturer Many-to-many: One-To-Many Relationship : Lecturer to an Additional Assignment Semester to a Additional Assignment Semester to a Module Run Module to a Module Run Study Branch to a Module Many-To-Many Relationship : Lecturers to Semester Lecturers to Module Lecturers to Module Run

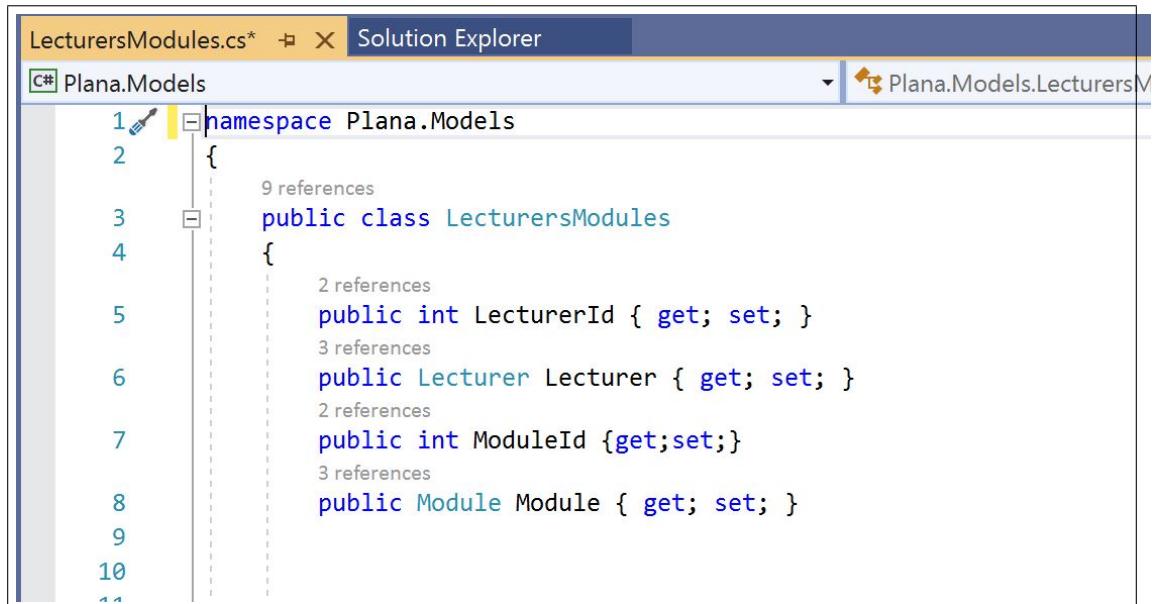
9.4 Modeling Types of Database Relationships

Many-to-many Relationship

Creating many-to-many relationship is little bit different from the one-to-many and one-to-one. We will take as example relation between Lecturer and Module.

In EF Core database doesn't directly implement this kind of relationships. First we have to create class Lecturer and class Module. Then we have to create one more class, we call it LecturersModules. This class links lecturers to their modules.

At the LecturersModules class there are two properties, LecturerId and ModuleId. There are both - primary keys and foreign keys, known as a composite key.[12]



The screenshot shows the Visual Studio Solution Explorer with the file 'LecturersModules.cs*' selected. The code editor displays the following C# class definition:

```
1  namespace Plana.Models
2  {
3      public class LecturersModules
4      {
5          public int LecturerId { get; set; }
6          public Lecturer Lecturer { get; set; }
7          public int ModuleId { get; set; }
8          public Module Module { get; set; }
9      }
10 }
```

The code editor highlights the 'namespace' and 'public class' lines in yellow. Intellisense dropdowns are visible above the 'LecturerId' and 'ModuleId' properties, showing '2 references' and '3 references' respectively. The 'Lecturer' and 'Module' types are also shown with their respective Intellisense dropdowns.

Figure 28: The LecturersModules entity class

The next step is adding necessary code to the AppDbContext class. We add

- **DbSet<Lecturer>**
- **DbSet<Module>**
- **DbSet<LecturersModules>**

The Figure 2 below shows this process. In the OnModelCreating method we add

- **modelBuilder.Entity<LecturersModules>().HasKey(x=> new {x.LecturerId, x.ModuleId};**

```

AppDbContext.cs* X
▼ Plana.Api.Models.AppDbContext ▼ ★ OnModelCreating(ModelBuilder modelBuilder)
  0 references
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Lecturer>().HasQueryFilter(p => !p.IsDeleted);

    modelBuilder.Entity<LecturersModules>()
        .HasKey(x => new { x.ModuleId, x.LecturerId });
}

```

Figure 29: Adding Entity LecturersModules to the AppDbContext class

We need write just this and Entity Framework Core will do the correct implementation that we can see then in the migration files. [14]

```

migrationBuilder.CreateTable(
    name: "LecturersModules",
    columns: table => new
    {
        LecturerId = table.Column<int>(nullable: false),
        ModuleId = table.Column<int>(nullable: false)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_LecturersModules", x => new { x.ModuleId, x.LecturerId });
        table.ForeignKey(
            name: "FK_LecturersModules_Lecturers_LecturerId",
            column: x => x.LecturerId,
            principalTable: "Lecturers",
            principalColumn: "LecturerId",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_LecturersModules_Modules_ModuleId",
            column: x => x.ModuleId,
            principalTable: "Modules",
            principalColumn: "ModuleId",
            onDelete: ReferentialAction.Cascade);
    });

```

Figure 30: Initial Migration File

From Figure 3 we can see that one many-to-many relationship has transformed in two one-to-many and many-to-one relationships.

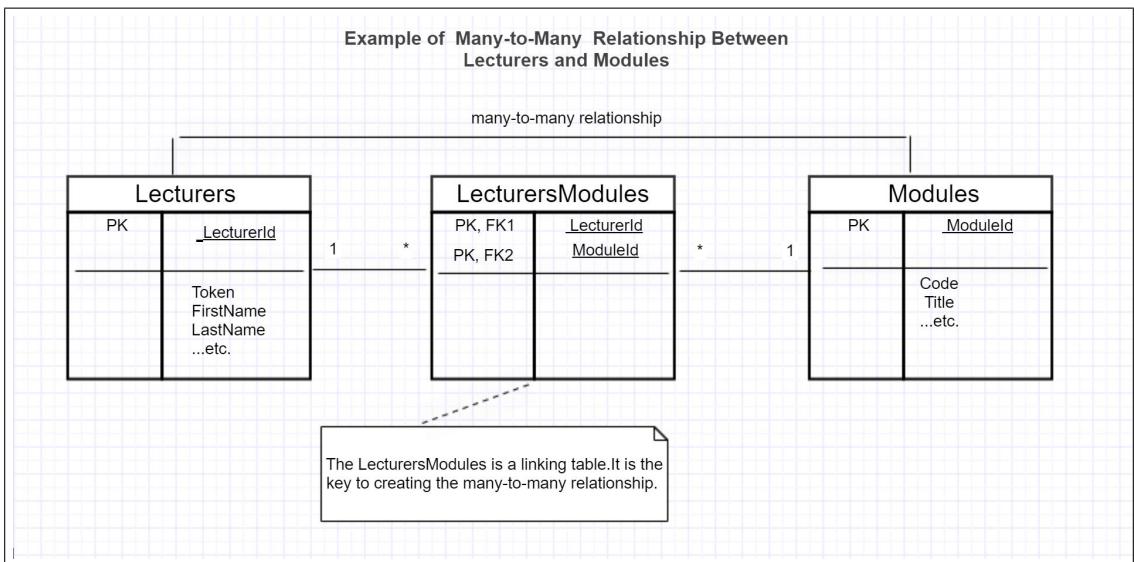


Figure 31: Creating Many-To-Many Relationship

9.5 Creating Database

9.6 Creating a Repository

todo: photo of repository

In our project we create a repository interfaces and implementation classes. We use **IQueryable<T>** and **IEnumerable<T>** interfaces. With **IQueryable<T>** interface the objects can be queried in more efficient way. For example: **public IQueryable<ModuleRun> ModuleRuns => appDbContext.ModuleRuns;** the **ModuleRuns** property in the context class returns a **DbSet<ModuleRun>** object, which implements the **IQueryable<T>** interface.

```
using Plana.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Plana.Api.Models
{
    public class ModuleRunRepository : IModuleRunRepository
    {
        private readonly AppDbContext appDbContext;

        public ModuleRunRepository(AppDbContext appDbContext)
        {
            this.appDbContext = appDbContext;
        }
        public IQueryable<ModuleRun> ModuleRuns => appDbContext.ModuleRuns;
```

Figure 32: The ModuleRunRepository.cs file in the Plana.Api/Models folder

Then we create the Repository Service in the Startup.cs file.

```
public class Startup
{
    0 references
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    2 references
    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the
    0 references
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddDbContext<AppDbContext>(options =>
            options.UseSqlServer(Configuration.GetConnectionString("DbConnection")));

        services.AddScoped<ILecturerRepository, LecturerRepository>();
        services.AddScoped<IModuleRepository, ModuleRepository>();
        services.AddScoped<IModuleRunRepository, ModuleRunRepository>();

        services.AddControllers();
    }
}
```

Figure 33: Creating Services in Startup.cs File

[15]

Creating the Database Migration, Code-First Migration

Entity Framework Core makes it possible to generate schema for the database from the data model classes using **migrations dotnet em migrations add Initial** [15]

9.7 Creating Seed Data

The seed data is the data that is used to populate the database. For seed data we add class SeedData.cs in the Models folder in Plana.Api project [15]. By default, Entity Framework Core uses cascade deletes for depend relationships with non-nullable foreign keys. [12]

..add photo of seed class (give it name The contents of the SeedData.cs class

Configuration of Core Services and Entity Framework

It is necessary to make changes in Startup.cs class in Plana.Api project - configure Entity Framework Core and set up the services that will be used to access the database [12]. The figure below shows all these configurations.

```
using Microsoft.AspNetCore.Builder;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Plana.Api.Models;

namespace Plana.Api
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }
        public IConfiguration Configuration { get; set; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<AppDbContext>(options =>
                options.UseSqlServer(Configuration.GetConnectionString("DbConnectionString")));

            services.AddScoped<ILecturerRepository, LecturerRepository>();
            services.AddScoped<IModuleRepository, ModuleRepository>();
            services.AddScoped<IModuleRunRepository, ModuleRunRepository>();
            services.AddControllers();
        }

        public void Configure(IApplicationBuilder app, AppDbContext context)
        {
            app.UseDeveloperExceptionPage();
            app.UseHttpsRedirection();
            app.UseRouting();
            app.UseAuthorization();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
            SeedData.SeedDatabase(context);
        }
    }
}
```

Figure 34: Startup.cs in the Plana.Api project.Preparing Services and Middleware

9.8 Create a Controller

Complex Data Model

in this section I would like to highlight more complex features of coding and data model structure in asp .net core.

```

public async Task<IEnumerable<Lecturer>> GetLecturers()
{
    return await appDbContext.Lecturers.ToListAsync();
}

```

Figure 35: LecturerRepository.cs file in the Plana.Api project's folder

```

{"lecturerId":12,"photoPath":"images/michele.jpg","lecturersModules":null,"lecturersModuleRuns":null,"lecturersSemesters":null,"additionalAssignments":null,"birthDate":"1986-05-11T00:00:00","gender":0,"workingRate":0.0,"isActive":false,"activeUntil":"0001-01-01T00:00:00","isDeleted":false,"firstName":"Michele","lastName":"Orsi","token":null,"email":"mo@gmx.ch","password":null,"role":0}
]

```

Figure 36

```

public async Task<IEnumerable<Lecturer>> GetLecturersModules()
{
    return await appDbContext.Lecturers
        .Include(m => m.LecturersModules)
        .ThenInclude(mo => mo.Module).ToListAsync();
}

```

Figure 37: ...

An unhandled exception occurred while processing the request.

JsonException: A possible object cycle was detected which is not supported. This can either be due to a cycle or if the object depth is larger than the maximum allowed depth of 32.

System.Text.Json.ThrowHelper.ThrowInvalidOperationException_SerializerCycleDetected(int maxDepth)

Stack Query Cookies Headers Routing

JsonException: A possible object cycle was detected which is not supported. This can either be due to a cycle or if the object depth is larger than the maximum allowed depth of 32.

System.Text.Json.ThrowHelper.ThrowInvalidOperationException_SerializerCycleDetected(int maxDepth)

System.Text.Json.JsonSerializer.Write(Utf8JsonWriter writer, int originalWriterDepth, int flushThreshold, JsonSerializerOptions options, ref

Figure 38

Package Manager Console

PM> Install-Package Microsoft.AspNetCore.Mvc.NewtonsoftJson -Version 3.0.0

Figure 39

```

public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }
    public IConfiguration Configuration { get; set; }

    [scr]
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddDbContext<AppDbContext>(options =>
            options.UseSqlServer(Configuration.GetConnectionString("DbConnectionString")));

        services.AddScoped<ILecturerRepository, LecturerRepository>();
        services.AddScoped<IModuleRepository, ModuleRepository>();
        services.AddScoped<IModuleRunRepository, ModuleRunRepository>();
        services.AddControllers();
        services.AddControllers().AddNewtonsoftJson(options =>
            options.SerializerSettings.ReferenceLoopHandling =
Newtonsoft.Json.ReferenceLoopHandling.Ignore
);
    }
}

```

Figure 40: ...

```

le":0},{"lecturerId":12,"photoPath":"images/michele.jpg","lecturersModules":[{"lecturerId":12,"moduleId":14,"module":
("ects":0,"moduleId":14,"title":"Computer Science Basics","code":"BTI1021","lectPerWeek":4,"totalHours":200.0),"lecturers":
[],"moduleRuns":null,"studyBranch":null}]}],"lecturersModuleRuns":null,"lecturersSemesters":null,"additionalAssignments":null,"bi-
rthDate":"1986-05-11T00:00:00","gender":0,"workingRate":0.0,"isActive":false,"activeTill":"2001-01-
01T00:00:00","isDeleted":false,"firstName":"Michele","lastName":"Orsi","token":null,"email":"mo@gmx.ch","password":null,"role":0
}]
}

```

Figure 41

Blazor Server

Configuring ASP.NET Core for Blazor Server

Call the API from Asp.net Core Blazor

- .. add picture with a blazor page
- ...write about imports
- ... write about registration of http client services

```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    public void ConfigureServices(IServiceCollection services)
    {
        services.AddRazorPages();
        services.AddServerSideBlazor();
        services.AddAutoMapper(typeof(LecturerProfile));

        services.AddHttpClient<ILecturerService, LecturerService>(client =>
        {
            client.BaseAddress = new Uri("https://localhost:44399/");
        });

        services.AddHttpClient<ILecturersModulesService,
LecturersModulesService>(client =>
        {
            client.BaseAddress = new Uri("https://localhost:44399/");
        });
    }
}
```

Figure 42: Registration of Http Client Services in Startup File in Plana.Web Project folder

```

using Microsoft.Extensions.Hosting;
using Plana.Web.Models;
using Plana.Web.Services;

namespace Plana.Web
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddRazorPages();
            services.AddServerSideBlazor();
            services.AddAutoMapper(typeof(LecturerProfile));

            services.AddHttpClient(client =>
            {
                client.BaseAddress = new Uri("https://localhost:44399/");
            });
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Error");
                app.UseHsts();
            }

            app.UseHttpsRedirection();
            app.UseStaticFiles();

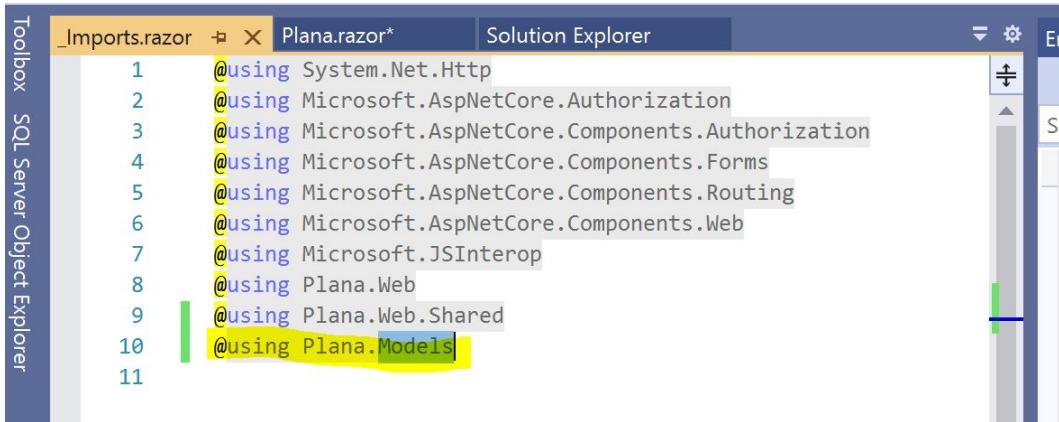
            app.UseRouting();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapBlazorHub();
                endpoints.MapFallbackToPage("/_Host");
            });
        }
    }
}

```

Figure 43: Adding Services and Middleware in the Startup.cs File in the Plana.Web project folder

When we need to use some data from other folders in a blazor files like Razor files(.razor) all necessary imports we include in the partial class **_Imports.razor** using the **@using** directives.



```

_Imports.razor  X  Plana.razor*  Solution Explorer
1  @using System.Net.Http
2  @using Microsoft.AspNetCore.Authorization
3  @using Microsoft.AspNetCore.Components.Authorization
4  @using Microsoft.AspNetCore.Components.Forms
5  @using Microsoft.AspNetCore.Components.Routing
6  @using Microsoft.AspNetCore.Components.Web
7  @using Microsoft.JSInterop
8  @using Plana.Web
9  @using Plana.Web.Shared
10 @using Plana.Models
11

```

Figure 44: Adding required namespaces to the `_Imports.razor` file in `Plana.Web` project folder

9.9 Changing Architecture

After studying the literature about the Framework entity, I came to the conclusion that the Laeyr repositories do not really matter in the application, since Repository just duplicates what the `DbContext` class of Entity Framework already gives us. I deleted the repository layer, so the service class directly contacts with the data access layer.

9.10 Setup for Blazor

To use the Blazor framework it is necessary to install :

- **.NET Core SDK 3.1 or later** from <http://dotnet.microsoft.com/download>
- **Visual Studio 2019** from <https://visualstudio.microsoft.com/downloads/>

The sprints covered a one three-four weeks period. At the end of each sprint, there was a discussion with the supervisor.

10 Testing

The most important was to test REST-API. For this purpose, the Postman application was chosen. The main logic is stored in the Service classes. And they are injected in Controller classes. Thus, were tested Controllers classes in the Api sub project.

There are several HTTP status codes. The status code it is when the server sends back an HTTP code to signal the status of the request. **postman**

10.1 Test Cases

Common HTTP status codes include:

Status Code	Description
200	OK Successful responses
400	Bad_REQUEST
404	NOT_FOUND
500	SERVER_ERROR

Table 10: HTTP status code

All Controller classes and all methods in it are tested. Below are examples of testing certain methods and Response Statuses .

Testing the ModuleRun Controller

Update ModuleRun

The screenshot shows the Postman application interface. The request URL is `https://localhost:44399/api/moduleRun`. The Body tab contains the following JSON payload:

```
1 {  
2   "moduleId": 1,  
3   "code": "b",  
4   "module": null,  
5   "place": "Dionaea"  
6 }
```

The response status is `200 OK`, time: `244 ms`, size: `5.37 KB`.

Figure 45: Test for update method of ModuleRun

Get ModuleRun

The screenshot shows a Postman request for 'GetModuleRun'. The method is 'GET' and the URL is 'https://localhost:44399/api/moduleRun/4'. The 'Params' tab is selected. The response status is 200 OK, time is 445 ms, and size is 528 B. The response body is a JSON object:

```
1 {
2     "moduleId": 4,
3     "code": "a",
4     "module": {
5         "ects": 0,
6         "moduleId": 15,
7         "title": "Projekt 2",
8         "code": "BTI7302",
9         "lectPerWeek": 1,
10        "totalHours": 25.0,
11        "lecturers": [],
12        "studyBranchId": 1
13    }
14 }
```

Figure 46: Update Module Run

Testing the LecturerModuleRun Controller

Get LecturerModuleRun

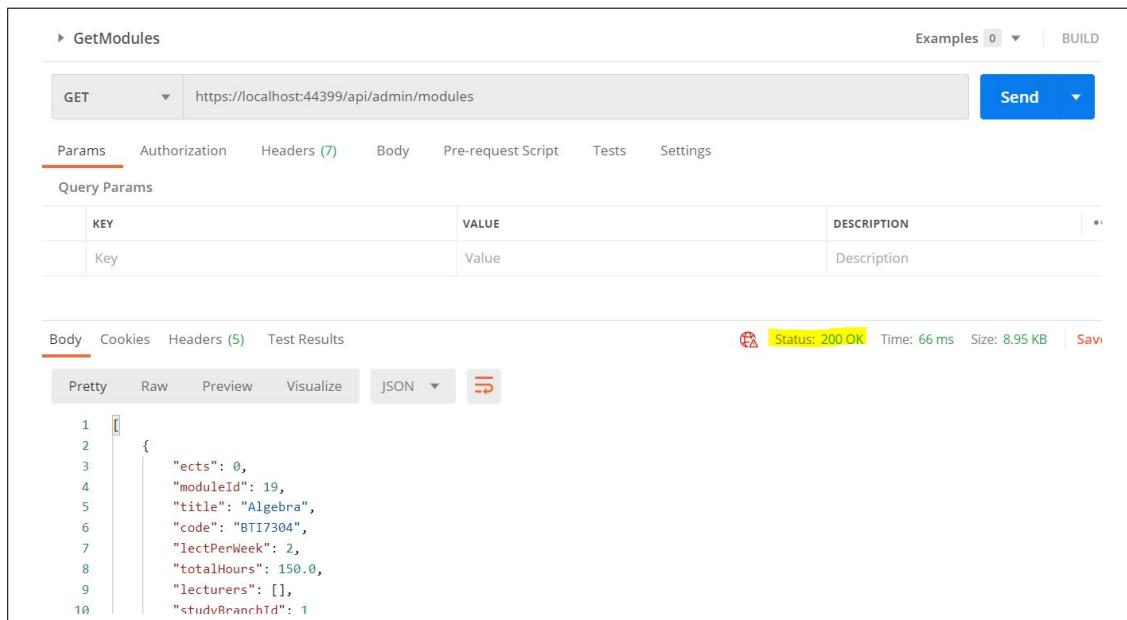
The screenshot shows a Postman request for 'GetLecturerModuleRun'. The method is 'GET' and the URL is 'https://localhost:44399/api/lecturerModuleRun/1/10'. The 'Params' tab is selected. The response status is 200 OK, time is 151 ms, and size is 5.39 KB. The response body is a JSON array:

```
1 [
2     {
3         "lecturerId": 10,
4         "lecturer": {
5             "id": 10,
6             "firstName": "Mike",
7             "lastName": "White",
8             "token": null,
9             "email": "mikewhite@gmx.ch",
10            "confirmEmail": "mikewhite@gmx.ch",
11            "password": null
12        }
13    }
14 ]
```

Figure 47: Test for get LecturerModuleRun method

Testing the Modules Controller

Get All Modules

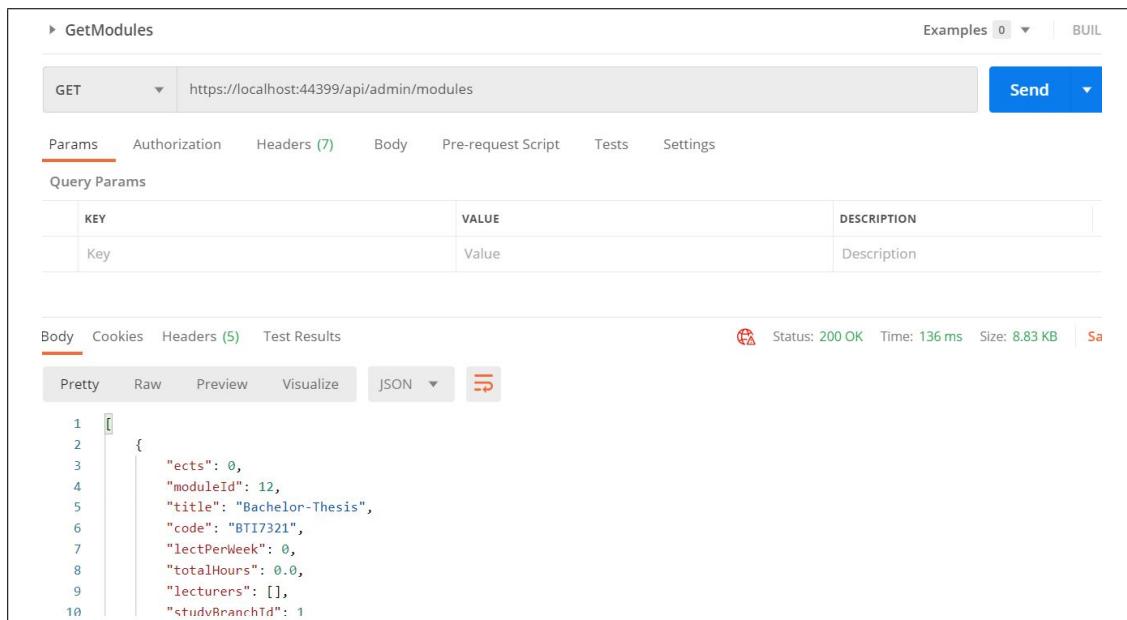


The screenshot shows a Postman test for the 'GetModules' endpoint. The URL is `https://localhost:44399/api/admin/modules`. The response status is 200 OK, time is 66 ms, and size is 8.95 KB. The JSON response body is:

```
1 [ { 2   "ects": 0, 3   "moduleId": 19, 4   "title": "Algebra", 5   "code": "BTI7304", 6   "lectPerWeek": 2, 7   "totalHours": 150.0, 8   "lecturers": [], 9   "studyBranchId": 1 }
```

Figure 48: Test for get all modules

Get module



The screenshot shows a Postman test for the 'GetModules' endpoint with a query parameter 'Key'. The URL is `https://localhost:44399/api/admin/modules`. The response status is 200 OK, time is 136 ms, and size is 8.83 KB. The JSON response body is:

```
1 [ { 2   "ects": 0, 3   "moduleId": 12, 4   "title": "Bachelor-Thesis", 5   "code": "BTI7321", 6   "lectPerWeek": 0, 7   "totalHours": 0.0, 8   "lecturers": [], 9   "studyBranchId": 1 }
```

Figure 49: Test of get a module method

Create module

The screenshot shows a Postman interface for a 'CreateModule' collection. A POST request is made to `https://localhost:44399/api/admin/modules`. The 'Body' tab is selected, showing a JSON payload:

```
1 {
2   "ects": 6,
3   "title": "Algebra",
4   "code": "BT17304",
5   "lectPerWeek": 2,
```

The response status is 201 Created, with a time of 155 ms and a size of 363 B. The response body is:

```
1 {
2   "ects": 6,
3   "moduleId": 19,
4   "title": "Algebra",
5   "code": "BT17304",
6   "lectPerWeek": 2,
7   "totalHours": 150.0,
8   "lecturers": [],
9   "studyBranchId": 1
```

Figure 50: Test of creating a new module

Testing the Plan Controller

Create a plan

The screenshot shows a Postman interface for a 'Post' collection. A POST request is made to `https://localhost:44399/api/plan`. The 'Body' tab is selected, showing a JSON payload:

```
1 {
2   "isFixed": false,
3   "isModifiable": true,
4   "name": "2020_2020"
```

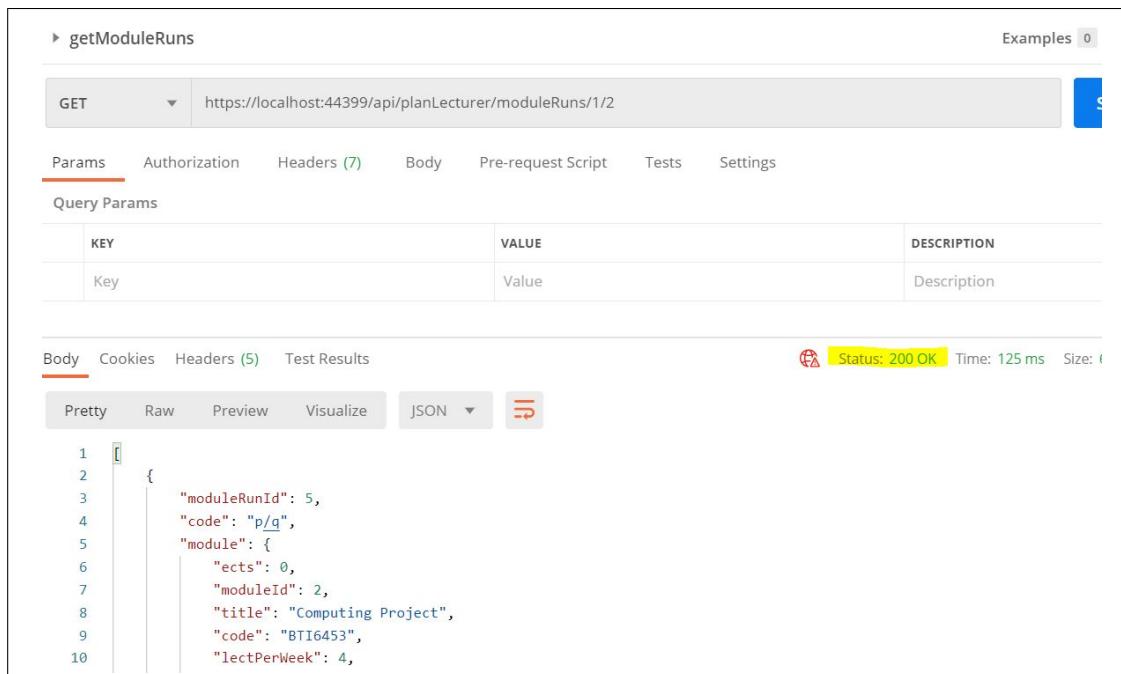
The response status is 400 Bad Request, with a time of 76 ms and a size of 526 B. The response body is:

```
1 {
2   "errors": {
3     "AutumnSemester.Code": [
4       "the Title of Semester must be provided!"
5     ],
6     "SpringSemester.Code": [
7       "the Title of Semester must be provided!"
8     ]
9   },
10  "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
```

Figure 51: Test to create a new plan method

Testing the Plan Lecturer Controller

Get module runs of certain planLecturer



The screenshot shows a POSTMAN interface with the following details:

- Method: GET
- URL: <https://localhost:44399/api/planLecturer/moduleRuns/1/2>
- Headers: (7)
- Body: (empty)
- Pre-request Script: (empty)
- Tests: (empty)
- Settings: (empty)

Query Params table:

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body tab (Pretty view):

```
1
2
3
4
5
6
7
8
9
10
{
  "moduleId": 5,
  "code": "p/q",
  "module": {
    "lects": 0,
    "moduleId": 2,
    "title": "Computing Project",
    "code": "BT16453",
    "lectPerWeek": 4,
```

Figure 52: Test to find module runs for the concrete Lecturer in concrete Plan

Testing the Semester Controller

Delete Semester

The screenshot shows the Postman application interface. At the top, there is a header with a back arrow, the title "SoftDelete", and a "Send" button. Below the header, the method is set to "DELETE" and the URL is "https://localhost:44399/api/semester/1". The "Body" tab is currently selected, indicated by a red underline. Under the "Body" tab, there are several options: "Params", "Authorization", "Headers (7)", "Body", "Pre-request Script", "Tests", and "Settings". The "Body" dropdown menu has items: "none", "form-data", "x-www-form-urlencoded", "raw", "binary", and "GraphQL". A note below the dropdown says "This request does not have a body". In the main content area, there are tabs for "Body", "Cookies", "Headers (5)", and "Test Results". The "Body" tab is active and displays a JSON response: "1 true". To the right of the response, there is a status summary: "Status: 200 OK", "Time: 201 ms", and "Size: 177 B". Below the status summary, there are buttons for "Pretty", "Raw", "Preview", "Visualize", and "JSON".

Figure 53: Test to delete a semester

11 Conclusions

List of Figures

1	Gitlab boards with issues	12
2	Sprints Calendar	12
3	13
4	Product Backlog	18
5	Layer Architecture of Plana Back-end	21
6	Layer Architecture of Plana Application	23
7	Study director's dashboard view	25
8	Create plan view	25
9	Add a new module view	26
10	A boat.	27
11	Add module/ modules group view	27
12	Add lecturer/lecturers group view	28
13	See last year plan view	29
14	Copy last year plan.	29
15	Lecturer's dash board view.	30
16	Lecturer's dash board view with opened plan	30
17	Add a new module view	31
18	Add a new additional assignment.	31
19	See lecturers view	32
20	See the notes view.	32
21	See last year plan.	33
22	The plan view after lecturer's input	34
23	Make a note view.	35
24	Domain Model for PLANA	36
25	Domain Model for PLANA	36
26	Domain Model for PLANA	37
27	Domain Model for PLANA	37
28	The LecturersModules entity class	42
29	Adding Entity LecturersModules to the AppDbContext class	43
30	Initial Migration File	43
31	Creating Many-To-Many Relationship	44
32	The ModuleRunRepository.cs file in the Plana.Api/Models folder	45
33	Creating Services in Startup.cs File	46
34	Startup.cs in the Plana.Api project.Preparing Services and Middleware	47
35	LecturerRepository.cs file in the Plana.Api project's folder	48
36	48
37	48
38	48
39	48
40	49
41	49

42	Registration of Http Client Services in Startup File in Plana.Web Project folder	50
43	Adding Services and Middleware in the Startup.cs File in the Plana.Web project folder	51
44	Adding required namespaces to the _Imports.razor file in Plana.Web project folder	52
45	Test for update method of ModuleRun	53
46	Update Module Run	54
47	Test for get LecturerModuleRun method	54
48	Test for get all modules	55
49	Test of get a module method	55
50	Test of creating a new module	56
51	Test to create a new plan method	56
52	Test to find module runs for the concrete Lecturer in concrete Plan	57
53	Test to delete a semester	58

12 Appendix

Gitlab-Repositories (Bachelor-Thesis): <https://gitlab.ti.bfh.ch/shirk1/planning-of-the-assessment-of-student-plans>

Gitlab-Repositories (Project2): <https://gitlab.ti.bfh.ch/shirk1/project2>

Other sources used: Blazor Tutorials: <https://www.pragimtech.com/courses/blazor-tutorial-for-beginners/>

Guide to a Writing a Thesis: <https://www.ece.rutgers.edu/~marsic/thesis-guide.html>

13 Declaration of Authorship

I hereby certify that I composed this work completely unaided, and without the use of any other sources or resources other than those specified in the bibliography. All text sections not of my authorship are cited as quotations, and accompanied by an exact reference to their origin.

Place, date:

Signature:

References

- [1] D. J.-U. Meyer, *Product development*. [Online]. Available: <https://innolytics-innovation.com/product-development/> (visited on 11/23/2020).
- [2] P. Eeles, “Capturing architectural requirements,” *IBM Rational developer works*, 2005.
- [3] J. P. Smith, *Entity Framework core in action*. Manning Publications Co., 2018.
- [4] M. Flower, *Layeringprinciples*. [Online]. Available: <https://martinfowler.com/bliki/LayeringPrinciples.html> (visited on 01/07/2015).
- [5] C. Woodruff, *Advanced architecture for asp.net core web api*. [Online]. Available: <https://www.infoq.com/articles/advanced-architecture-aspnet-core/> (visited on 06/01/2018).
- [6] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004.
- [7] J. Watmore’s, *Asp.net core 2.2 - role based authorization*. [Online]. Available: <https://jasonwatmore.com/post/2019/01/08/aspnet-core-22-role-based-authorization-tutorial-with-example-api/> (visited on 01/08/2010).
- [8] E. F. Tutorial, *Entity framework core: Dbcontext*. [Online]. Available: <https://www.entityframeworktutorial.net/efcore/entity-framework-core-dbcontext.aspx> (visited on 2020).
- [9] M. Pfahrer, *E-business and web – web technologies, blazor, 2019/20*.
- [10] S. Singh, *Onion architecture in asp.net core mvc*. [Online]. Available: <https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-mvc/> (visited on 06/06/2020).
- [11] stackoverflow, *What is the data transfer object*. [Online]. Available: <https://stackoverflow.com/questions/1051182/what-is-data-transfer-object/> (visited on 2020).
- [12] J. P. Smith, *Entity Framework in Action*. 2018, ISBN: 9781617294563.
- [13] Microsoft.com, *Entity framework core*. [Online]. Available: <https://docs.microsoft.com/en-us/ef/>.
- [14] P. God, *Many-to-many relationship with entity framework core*. [Online]. Available: https://dev.to/_patrickgod/many-to-many-relationship-with-entity-framework-core-4059 (visited on 11/23/2020).
- [15] A. Freeman, *Pro ASP.NET Core 3*. 2020, ISBN: 978-1-4842-5439-4.

14 Protocol

Frequency: (biweekly)

Meeting length: (60 minutes)

Agenda

- Demo and Discuss Deliverable(Demo)
- Planning next Goals(Plan)
- Retrospective
- Date, time of the next meeting(next meeting)

Report from 24.09.20

Plan

Future goals are:

-
-

Retrospective

Next Meeting: 08.09.20

Report from 08.10.20

Plan

Future goals are:

-
-

Retrospective

Next Meeting: 21.10.20

Report from 21.10.20(13:30)

Plan

Future goals are:

- Implementation. Add new concept classes into the application.
- Implementation. Start implementation of graphic concepts we've made.
- Make Study director plan view concept with less elements.

- Put UI concepts to the report.

Retrospective

- In graphical concept i have choice between semester view and year view, but for planning is more important year view.
- Actual state of planing view for study director looks little bit heavy because of many elements in it, would be better to minimize some staffs. In implementation we can manipulate view hiding some part of them.
- Additional Assignments have a category, description and number of hours.

Next Meeting: 04.11.20 (13:30)

Report from 04.11.20

Plan

Future goals are:

- **report** Describe the Additional Assignments. There are several Arts of Additional Assignments. The Lecturer and also Study Director can plan and manage it.
- **report** Describe what is innovative in this work. For example that Lecturer, Study Director and Institute Manager can active collaborate with each other. That is idea of making groups is also innovative. In general, we make a specific solution for problem that we face.
- Put tasks which have to be done until **21-January 2021** into the sprint backlog. These are
 1. Report
 2. Final Presentation
 3. Book
 4. Video
- **Application** Implement the views that we made with Axure tools.

Retrospective

- The graphical concepts looking better now.
- Conflict between number of hours and lecturers that can teach specific module have place in application.

Next Meeting: 18.11.20 (13:30)

Report from 18.11.20

Plan

Future goals are:

- Add business logic into separated project.
- Test it.
- Add layer diagram

Retrospective

- It is important to add the date to the bib file in case i use the article from the internet.

Next Meeting: 02.12.20)

Report from

Plan

Future goals are:

-

Retrospective

-

Next Meeting:)