**Имена:**  Тервел Радославов Вълков **фн:** *81623***Начална година:** *2017* **Програма:** бакалавър, (КН) **Курс: 4**   
**Тема:** 24.1 Редактор и генератор на учебна програма във ФМИ и съставяне на граф на зависимостите между дисциплините **Дата: 2021-01-22 Предмет: w15prj\_KN \_final имейл:** trvulkov@uni-sofia.bg

**преподавател:** доц. д-р Милен Петров

# ТЕМА: 24.1 Редактор и генератор на учебна програма във ФМИ и съставяне на граф на зависимостите между дисциплините

## 1. Условие

Редактор и генератор на учебна програма във ФМИ и съставяне на граф на зависимостите между дисциплините. (например: https://intranet.fmi.uni-sofia.bg/index.php/s/fI5QKFCi1xSQKoQ) - съдържа списък от елементи около 3 таблици + няколко списъка + различни параметри на плана; да се добави и таблица за зависимости (от една страна този предмет от кои предмети зависи, и от друга - кои предмети зависят от този - и да се генерира визуализация).

## 2. Въведение

Изготвянето на учебна програма е същинска част от образователната дейност във ФМИ и включва много части и трудности. Този проект не се фокусира толкова върху разпределянето на предметите по зали, а по-скоро върху разписанието, както и зависимостите между предмети. Освен предметите в първите семестри на различните специалности, и някои избираеми дисциплини, почти всеки предмет във ФМИ има предварителни изисквания – други предмети, които студентите трябва да са завършили, и други знания, които те трябва да са усвоили. Поради това съществен момент при подготвянето на учебната програма е подреждането на предметите според тези изисквания, така че студентите да учат даден предмет само след като са удовлетворили предварителните му изисквания. Нещо което може да помогне за това е изграждането на граф на зависимостите – структура, в която възлите са предмети, а ребрата представят зависимостите между тях. Така от възела на даден предмет ще излизат ребра към предметите, които го имат като предварително изискване, а към същият възел ще сочат ребра от предметите, които той изисква.

## 3. Теория

Първият (презентационен) слой се реализира чрез HTML, като за всяка основна функционалност има отделен .html файл. Тези файлове се стилизира с CSS (разделен на 2 файла – form.css за стилизирането на формите, което е общо за повечето страници, и timetable.css за по-специфичното стилизиране на разписанието), и към тях има включени и скриптове на JavaScript. Вторият слой е базата данни, a третият слой (бизнес логиката) е реализиран чрез PHP.

## 4. Използвани технологии

Тествано на конфигурация:

* сървър – Microsoft Windows 10, Version 10.0.19041, Build 19041, x64
* сървърен език – PHP 7.4.10 (ZTS Visual C++ 2017 x64)
* уеб сървър – XAMPP v3.2.4 (Apache HTTP Server 2.4.46.0)
* сървър за БД - 10.4.14-MariaDB

Други технологии:

* HTML5
* JavaScript
* CSS

Използвани библиотеки:

* vis.js network – библиотека за визуализация на графи [1]

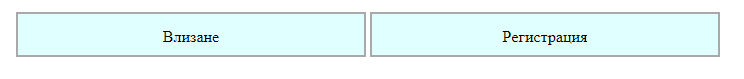
## 5. Инсталация и настройки

За приложението е необходим предварително инсталиран XAMPP. Предоставеният архив трябва да бъде разархивиран в папката htdocs на XAMPP (обикновено C:\xampp\htdocs). За пускането му, от браузър трябва да се навигира до <http://localhost:PORT/FOLDER/>, където PORT е заменено със съответния порт на който слуша Apache Web Server-а, а FOLDER е съответната папка в която е поставен проекта (например 81623\_exam\_final/project) . Тъй като проектът изисква и база данни, тя трябва да бъде създадена – на <http://localhost:PORT/phpmyadmin> трябва да бъде създадена нова база данни, чрез import-ване на предоставения с проекта файл timetable\_editor.sql. Класът Config (/libs/Config.php) съдържа настройки за хоста, името и потребителя/паролата за базата данни, така че ако тя е създадена с име различно от timetable\_editor, там ще трябва да бъде променена съответната настройка.

## 6. Кратко ръководство на потребителя

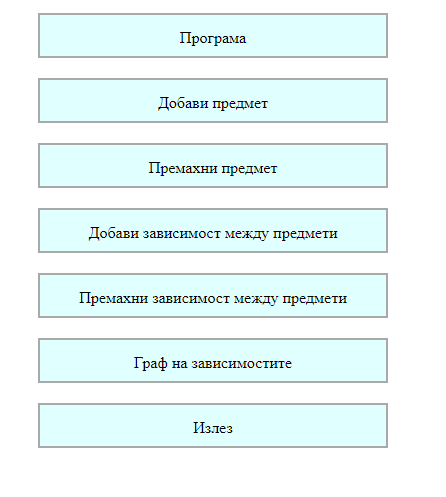
Първо потребителят трябва да стартира XAMPP на своя компютър, след което да достъпи проекта на съответния адрес в localhost и да премине към влизане в самата система.

Системата на този етап поддържа само администраторска роля, тъй като се предполага че е предназначена за хора във ФМИ, които ще подготвят учебната програма, но не и за хора които само ще я преглеждат без да имат привилегии за редактиране. На първоначалния екран има две опции – за влизане със съществуващ акаунт или регистрация на нов.



След влизане се стига до страница, съдържаща всичките опции на приложението. Добавянето на предмети и зависимости става чрез обикновена HTML форма, в която се попълват съответните данни. Премахването на предмети и зависимости отново става чрез форма, в която в dropdown меню са изредени предметите/зависимостите които в момента съществуват в базата данни. Бутонът „Програма“ зарежда разписанието, което първоначално е празно, но чрез форма може да бъдат заредени предметите за дадена специалност и семестър, след което чрез drag-and-drop да бъдат поставени в различни часове и дни на седмицата. Има опции за export на текущото разписание до .csv формат, както и за import на разписанието от същия формат. Бутонът „Граф на зависимостите“ просто зарежда съответният граф.

фигура : Екран с опциите на системата



фигура 1: Екран за вход/регистрация в системата

## 7. Примерни данни

Примерни данни за логване – *admin* с парола *admin*.

Във включеният към архива .sql файл е export-ната базата данни, която ползвах докато разработвах проекта. Тя освен структурата на самите таблици включва и някои примерни данни – 17 примерни предмета (основно от учебния план на Компютърни науки за първите няколко семестъра), 15 примерни зависимости между предмети, и частичен учебен план (обхващащ само първите 2 семестъра на КН и първият семестър на СИ).

## 8. Описание на програмния код

По-голямата част от проекта се състои от тройки файлове - <име>.html, <име>.js и <име>.php (например addSubject.html, addSubject.js и addSubject.php), като .html и .php файловете са разположени в главната папка на проекта, а .js файловете са в под-папката /scripts.

Тук .html файлът съдържа съответната страница на проекта, .php файлът обикновено прави някаква връзка с базата данни (в някои случаи за добавяне на данни, в друг за премахване, в трети за извличането им) и връща резултат в JSON формат, а .js файлът съдържа код за пращането да данни от съответния .html до съответния .php файл, получаването на резултат в JSON формат, и извършване на някакви действия с него.

Например за добавянето на предмети - addSubject.html съдържа форма, в която се задават име и лектор на предмета. Скриптът addSubject.js добавя event listener, който при submit на данните от формата ги изпраща към addSubject.php в съответния формат, където данните се получават и се извършва създаването и добавянето на предмета към базата данни. След това се връща резултат в JSON формат, който съдържа информация за успеха на операцията, която се приема от event listener-а в addSubject.js, който според този успех предприема други действия.

Подобна последователност на действия се случва при addPrerequisite и register. За removeSubject и removePrerequisite логиката е малко по-различна – вече в .php файловете се извършват две различни действия с базата данни. При GET се извличат и изпращат съответните предмети/зависимости, докато при POST се получават данни за предмет или зависимост и се извършва премахването им от базата данни. В съответните .js файлове поради това вече има и допълнителен listener, който е отговорен за зареждането на данните чрез GET заявка към съответния .php файл.

Тройката login файлове извършва по-различни действия – тук .php файлът проверява дали такъв потребител наистина има, и започва сесия. Началната страница homepage.html не е обвързана с PHP логика, тъй като тя само съдържа препратки към другите функционалности на проекта.

По-специфични действия има при timetable и graph – timetable.js скриптът има допълнителна логика за създаването и попълването на таблиците на разписанието и предметите, както и за import и export функционалността, докато graph.js се позовава на функциите от библиотеката vis.js (по-точно нейната компонента network, която се съдържа във файла vis-network.min.js в папката /scripts) за да създаде и визуализира графа на зависимостите.

Всичките .js файлове обвързани с дадена страница също така извършват проста проверка за това дали в момента има влязъл потребител, като за това се позовават до файла checkLogin.php.

Освен тези файлове, има и няколко допълнителни .php файла в папката /libs. Те включват класове за потребител, предмет и предварително изискване на предмет, които се ползват основно за извършване на валидация на данни, получени от презентационния слой. Класът ValidationException се ползва когато валидацията е неуспешна, а класът Db съдържа всичката логика свързана с базата данни. При конструиране се създава връзка към нея, а чрез различните публични функции се извършват различни действия по самата база данни, като селектиране, добавяне и премахване на данни.

Накрая, logout.php просто унищожава текущата сесия и връща потребителя до страницата за вход или регистрация.

## 9. Приноси на студента, ограничения и възможности за бъдещо разширение

В текущото си състояние приложението не поддържа различни роли за потребители – всички са с администраторски привилегии. Тази част може да се разшири, така че обикновени потребители да могат само да разглеждат програмата и зависимостите, докато привилегированите потребители да могат да правят редакции.

Разписанието сега е в опростен вид – то поддържа само лекции и предполага че всяка специалност се състои от един поток. Освен това, тъй като при лекциите обикновено няма разделение по групи, приложението не ги представя в програмата. В бъдеще тази част може да бъде разширена, като към програмата се добави възможност за добавяне на упражнения, и редовете за всеки ден бъдат разделени на под-редове за различните групи на съответната специалност.

За сега добавянето на предмети и зависимости става чрез прост интерфейс – форма, в която се задават съответните данни за предмета или зависимостта. Самото представяне на предметите също е опростено – те имат само име и лектор. В бъдеще, представянето на предметите може да бъде усложнено, така че да включва всичките детайли които истинските предмети във ФМИ имат – хорариуми, кредити, анотации, конспекти и т.н. Добавянето също може да бъде подобрено, като се направи по-интерактивно – например директно в самия граф на зависимости да може да бъдат добавяни нови предмети (възли) и зависимости (ребра).

Също така, на текущия етап при премахването на предмет няма каскадно премахване на зависимостите обвързани с него. Това най-вероятно би могло да се оправи с настройка на базата данни, ползвайки foreign key и ON DELETE CASCADE.

Относно структурирането на кода – на доста места се получава повторение на подобни секции код. Например в Db.php има функции, които извършват подобни действия, но върху различни таблици в базата данни, и съответно имат доста подобен код. Най-вероятно това повторение може да се елиминира, или поне намали, чрез по-добра организация на функциите. Има повторение на подобна логика и при .js скриптовете – например доста от event handler-ите извършват подобни действия (пращат данните на съответния .php файл, получават обратна връзка и според успеха и предприемат действия). Може би е възможно и тези повторения да бъдат намалени, но поради липса на опит с JavaScript не съм сигурен за най-добрия начин това да се постигне.

Най-накрая – в текущото си състояние системата предоставя доста проста обратна връзка на потребителя за успеха или неуспеха на действията му. В бъдеще това може да се подобри, така че съобщенията да са по-добри, и по-добре стилизирани.

## 10. Какво научих

Научих доста относно динамичното генериране на HTML елементи чрез JavaScript – например за генериране на таблици и попълването им с данни. Също така научих и някой детайли за свързването на презентационния слой и бизнес логиката. На доста места ми се наложи чрез PHP код да правя заявки до базата данни, резултатите от които след това да пращам обратно до презентационния слой в JSON формат, например за да може JavaScript код да попълни select елемент със съответни данни.

## 11. Списък с фигури и таблици

Списък на фигурите:

1. Екран за вход/регистрация в системата
2. Екран с опциите на системата

## 12. Използвани източници

[1] vis.js [https://visjs.org/]

[2] Документация на vis.js network [https://visjs.github.io/vis-network/docs/network/]

[3] HTML, CSS и JavaScript tutorial-и от w3schools.com [https://www.w3schools.com/]

[4] Документация на PHP [https://www.php.net/manual/en/]

[5] Valentino Gagliardi, Back To The Basics: How To Generate a Table With JavaScript [https://www.valentinog.com/blog/html-table/]

[6] Eric Bidelman, Rachel Andrew, Using the HTML5 Drag and Drop API [https://web.dev/drag-and-drop/]

[7] CodexWorld, Export HTML Table Data to CSV using JavaScript [https://www.codexworld.com/export-html-table-data-to-csv-using-javascript/]

[8] Stack Overflow [stackoverflow.com]

Предал (подпис): ………………………….

/*81623, Тервел Радославов Вълков, КН, група 1*/

Приел (подпис): ………………………….

/доц. *Милен Петров*/