

**Имена:** Росен Андреев Колев, Ралица Атанасова Симова, Георги Николаев Атанасов  
**ФН:** 0MI0800065, 1MI0800135, 3MI0800092

**Начална година:** 2021

**Програма:** бакалавър, (КН)

**Курс:** 4

**Тема:** 4.1

**Дата:** 26.01.2025

**Предмет:** w23prj\_KN\_final

**Имейл:** [rosenandreevkolev1@gmail.com](mailto:rosenandreevkolev1@gmail.com), [ralica.a.simova@gmail.com](mailto:ralica.a.simova@gmail.com),  
[nigosto@gmail.com](mailto:nigosto@gmail.com)

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

# **ТЕМА: w23/4.1 - Alumni Hub - Управление на процес по дипломиране**

## **1. Условие**

Проектът представлява система за управление на процеса по дипломиране. Основната му цел е предоставянето на възможност за контролиране на церемониите за дипломиране и предоставянето на съпътстващата ги информация по удобен начин. Предназначението му е да се превърне в единен портал, където администрацията да предоставя необходимата информация за предстоящи събития и студентите да получават лесно тази информация и да реагират, където това е необходимо. Системата е подходяща за използване от различни университети в България.

## **2. Въведение – извличане на изисквания**

В системата са реализирани три различни роли за потребителите - студенти, администрация и админ. Админът има най-големите правомощия в системата - одобрява заявки за регистрация на студенти и администрация и следи за нередности. Администрацията организира процеса по дипломирането и изпраща покани до студентите, а студентите следят за новини и отговарят на поканите от администрацията.

Основните функционалности, реализирани с този проект са:

- Регистрация на потребител. Възможностите за тип профил са студентски или администраторски. Регистрацията трябва да става с потребителско име, имейл и парола, като името и имейла трябва да са уникални в рамката на системата. За студентските профили е необходим и факултетен номер. След изпращане на заявката за регистрация, потребителя трябва да изчака получаване на одобрение от админа.
- Влизане във вече съществуващ профил. За вход са необходими потребителско име и парола.
- Добавяне на нов факултетен номер към студентския профил. Студентът трябва да има възможност да влиза в системата с различни факултетни номера и да вижда различна информация спрямо факултетния номер, който е избрал.
- Информация за профила. Потребителят трябва да може да види данните за неговия профил като потребителско име и имейл, а за студентските профили е налична и информация за следването на студента - средна оценка, дата на завършване и факултетен номер, както и информация за предстоящи церемонии за дипломирането.
- Импорт на студенти. Администрацията трябва да може да импортира данните за студентите, които ще се дипломират през следващата година, като импорта става от файл в CSV формат.

След като информацията е импортирана, данните за всички студенти са налични за преглед в таблица.

- Експорт на таблици със студенти и церемонии в CSV формат.
- Създаване на церемония - трябва да се предостави дата, година на завършване и да се поканят студентите. Тези покани може да са обикновена покана за церемонията, покана за изнасяне на речи, покана за отговорник за тоги, дипломи или подписи. Церемониите трябва да могат да се редактират. Всяка промяна в церемонията трябва да стане видима за студента след обновяване на страницата.
- Получаване на информация от студентите за предстояща церемония и даване на отговор на поканите. Студентите трябва да могат да отговорят с ДА/ НЕ на поканата за церемонията и ако приемат да присъстват и имат други покани - за речи и за отговорник тоги, дипломи или подписи, да могат да отговорят на тях също.
- Възможност за повторно изпращане на поканите за речи и за отговорник за тоги, дипломи или подписи от администрацията, ако някой от поканените е отказал.
- Студентите трябва да могат да изберат размер за тогата си, като изборът става само между останалите налични размери.
- Възможност за преглед на всички церемонии и на подробна информация за някоя церемония от администрацията.
- Админът трябва да има възможност да одобрява заявките за създаване на профили.

Нефункционалните изисквания, които трябва да се реализират със системата са:

- Сигурност чрез контрол на достъпа спрямо ролите. Преди да успеят да влязат в приложението, потребителите трябва да минават през процес на удостоверяване и овластяване. Контрол на достъпа се осъществява и чрез одобряването на заявките за създаване на профил. Постигане на сигурност при съхранението на паролите чрез криптиране.
- Надеждност се постига чрез валидиране на входните данни и обработка на грешките.
- Използваемост се постига чрез интуитивен дизайн и показване на подходящи грешки на потребителя.
- Възможност за лесна промяна се постига чрез преизползването на общи компоненти и дизайна на архитектурата чрез стила модел-изглед-контролер (MVC)
- Конфигурируемост се постига чрез наличието на конфигурационен файл с променливи на средата.

Ползите от реализацията на тази система са постигането на унифициран и систематизиран процес за дипломирането, събирането на необходимата информация на едно място, достъпно за всички участници в процеса и доставянето на лесен и удобен канал за информация.

### 3. Теория – анализ и проектиране на решението

Реализирана е архитектурата на модел-изглед-контролер (MVC). Моделният компонент управлява поведението и данните на приложението, изпраща информация за неговото състояние и отговаря на инструкции за промяна на състоянието. Изгледът има задължението да управлява представянето на информация пред потребителите. Контролерът управлява взаимодействието с потребителя, валидира входните данни и информира модела или изгледа, за да предприемат подходящи действия.

Като допълнителен архитектурен стил е реализиран Front Controller. Той се състои в използването на централизирана точка за обработка на всички входящи заявки към приложението.

RewriteEngine On

```
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule !^(css|images|files)/ index.php [NC,L]
```

фиг. 3.1 Логиката за пренасочване на заявките към единствената входна точка.

## 4. Използвани технологии

За реализацията на проекта е използвана средата за разработка XAMPP, версия 8.12.2-0, както под операционната система Windows 10 и Windows 11, така и под Ubuntu 22.04. Използваната версия на XAMPP включва:

- интерпретатор на PHP, версия 8.12.2;
- сървър Apache, версия 2.4.58;
- дистрибуция на MariaDB, версия 10.4.32.

По време на разработката е използвана системата за контрол на версиите Git, както и отдалеченото хранилище Github.

## 5. Инсталация, настройки и DevOps

За инсталацията на приложението е необходимо да има инсталирана средата XAMPP, версия 8.12.2-0 или по-висока. Изходният код на приложението трябва да бъде поставен в директорията htdocs (или в някоя нейна поддиректория), която се намира в директорията на инсталация на XAMPP. Например под операционната система Windows, стандартния път до директорията е: C:\xampp\htdocs, а под Linux е /opt/lampp/htdocs. След това трябва през контролния панел на XAMPP да бъдат стартирани Apache Web Server и MySQL Database.

За правилната работа на приложението, трябва да бъде създаден .env файл, който да съдържа променливи на средата. Файлт .env.example съдържа примерната структура на един .env файл, но без конкретни стойности на променливите. Необходимите променливи са име, хост, парола и потребител на базата от данни, както и базовият URL на приложението. Този URL трябва да съдържа като подпътища съответните поддиректории на htdocs, в които е разположен изходния код. Ако кодът на приложението е разположен в директорията C:\xampp\htdocs\alumni-hub, не са правени промени по конфигурацията на MariaDB и името на базата данни е alumnihub, един примерен .env файл би изглеждал по следния начин:

```
DB_NAME="alumnihub"  
DB_HOST="localhost"  
DB_PASSWORD=  
DB_USER="root"  
  
BASE_URL="http://localhost/alumni-hub"
```

фиг. 5.1 Примерен .env файл

След това, приложението ще бъде достъпно на адреса, сочен от BASE\_URL променливата. Преди обаче да бъде готово за употреба, трябва да се изпълнят миграциите върху базата от данни. Това става, като се изпълни следната команда:

```
php scripts/run_migrations.php -m -a
```

Ако е необходимо, могат да се приложат единични миграции, като се изпълни горната команда без аргумента -a и вместо него се окаже името на миграция - директорията, в която се намира. Например за изпълнението на миграцията за създаване на началните таблици може да се използва командата:

```
php scripts/run_migrations.php -m m1_create_tables
```

Ако има нужда да се отменят миграциите може да се използват горните две команди, но вместо аргумента -m се използва -r. Например за отмяна на всички миграции може да се използва командата:

```
php scripts/run_migrations.php -r -a
```

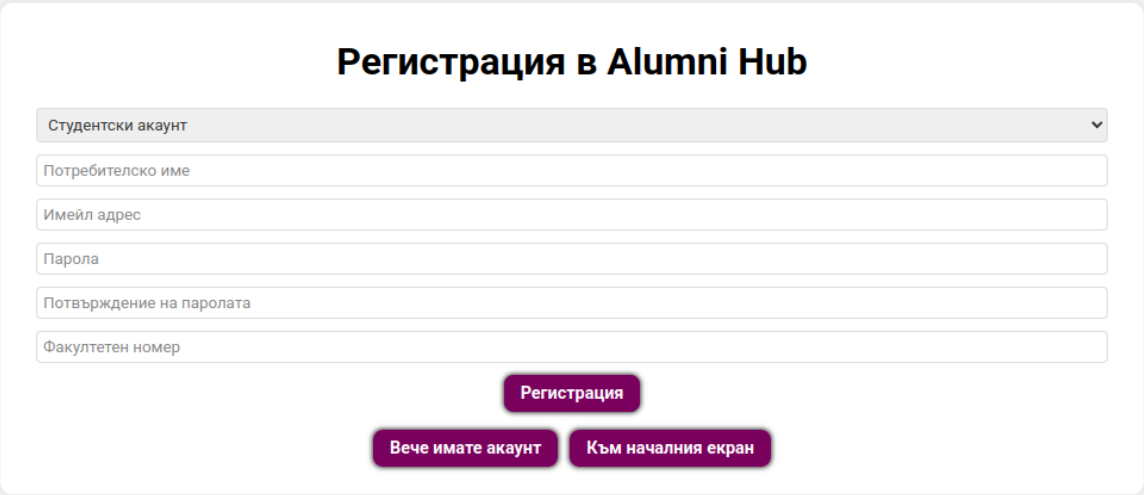
Преди да се започне работа с приложението трябва да се изпълни скрипт, който да зареди началните данни. Това става, като се изпълни следната команда:

```
php scripts/seed_database.php seed
```

След изпълнението на този скрипт, в базата от данни се зарежда потребител с роля админ, както и примерни потребители, студенти и дрехи. Без този скрипт няма как да се добави потребител с роля админ и съответно няма как да бъдат одобрени останалите потребители.

## 6. Кратко ръководство на потребителя

Потребителят е посрещнат от начална страница, но за да достъпи функционалностите, трябва да се регистрира. При регистрация, потребителят може да избере роля на администратор или на студент, като за студентската регистрация трябва да добави и факултетен номер.



The image shows a registration form titled "Регистрация в Alumni Hub". At the top, there is a dropdown menu with "Студентски акаунт" selected. Below it are five input fields: "Потребителско име", "Имейл адрес", "Парола", "Потвърждение на паролата", and "Факултетен номер". At the bottom, there are three buttons: "Регистрация" (highlighted in purple), "Вече имате акаунт", and "Към началния екран".

фиг. 6.1 Форма за регистрация с избрана роля за студент

При успешна регистрация, потребителят трябва да изчака одобрение на профила от админа, преди да може да достъпи функционалностите за избраната роля.

Списък с администратори				
Имейл	Потребителско име	Роля	Одобрен	Действие
maria.ivanova@example.com	Marialvanova	Администратор	Да	Одобри
georgi.georgiev@example.com	GeorgiGeorgiev	Администратор	Да	Одобри
nigosto@gmail.com	nigosto	Администратор	Да	Одобри
milten_petrov@gmail.com	milten	Администратор	Не	Одобри
asen_vasilev@abv.bg	asen4o	Администратор	Не	Одобри

фиг. 6.2 Списък на заявките за одобряване на администратори

След одобряване, администраторът има пълен достъп до всички функционалности на администрацията, но студентът трябва да избере, с кой от одобрените му факултетни номера да влезе, понеже може да има профил с един факултетен номер, докато е бил бакалавър и друг, след като е станал магистър.

The form is titled "Моля изберете факултетния номер, с който да влезете" (Please select the faculty number with which you will log in). It contains a dropdown menu labeled "Избор на факултетен номер" (Select faculty number) with a downward arrow. Below the dropdown is a purple button labeled "Избери" (Select). Underneath is the text "Или добавете нов факултетен номер" (Or add a new faculty number). This is followed by a text input field labeled "Факултетен номер" (Faculty number). At the bottom is a purple button labeled "Добави" (Add).

фиг. 6.3 Форма за избор или добавяне на нов факултетен номер

След избиране на факултетен номер, студентът ще бъде пренасочен към профилната му страница, където може да види информацията за себе си, както и поканите за участие в церемониите.

The page is titled "Добре дошли в Alumni Hub" (Welcome to Alumni Hub). Below the title is a subtitle: "Тук можете да намерите информация за вашия профил" (Here you can find information about your profile). The profile information is listed as follows:  
Потребителско име: nigosto\_student  
Имейл: nigosto@abv.com  
Име: Георги Атанасов  
Степен на образование: Бакалавър  
Факултетен номер: 3MI0800092  
Оценка: 5.45  
Година на завършване: 2025  
Размер на тога: S  
Below the profile information is a section titled "Покани" (Invitations). It contains two entries:  
1. Дата на церемонията: 2025-01-24 23:16:00  
Вие приехте тази покана  
Приемате ли да изнасяте реч: Да Не  
Вие приехте да отговаряте за тогите  
2. Дата на церемонията: 2025-01-25 12:12:00  
Приемате ли поканата: Да Не

фиг. 6.4 Информация за профила на студент

Администраторът от друга страна, може да добавя студенти, да преглежда списъка със студенти и да го експортира.

Импортиране на студенти

Можете да добавите студенти, директно като качите CSV файл, съдържащ информация за тях. Ако имате таблица на Excel с данните за студентите, може да я запазите в CSV формат и да качите съответния файл.

Няма избран файл

Изпращане

фиг. 6.5 Поле за качване на файл, съдържащ информация за студенти

Списък със студенти

Експорт

Факултетен номер	Степен	Имена	Година на завършване	Оценка
0MI0800065	Бакалавър	Росен Андреев Колев	2025	3.78
1MI0800135	Магистър	Ралица Атанасова Симова	2024	4.21
1MI1234567	Бакалавър	Елена Андреева Стоянова	2025	3.01
1MI8901234	Бакалавър	Митко Станиславов Димитров	2026	5
2MI6789012	Бакалавър	Теодора Василева Петкова	2025	5.75
2MI9876543	Бакалавър	Петър Станимирков Колев	2025	6
3MI0800092	Бакалавър	Георги Николаев Атанасов	2025	5.45
3MI4567890	Бакалавър	Красимира Николаева Димитрова	2025	6
3MI7890123	Доктор	Александър Викторов Станев	2025	4.5
4MI6543210	Бакалавър	Николай Радославов Илиев	2025	5
5MI2345678	Магистър	Валентина Валериева Маринова	2025	5.5
6MI8765432	Бакалавър	Стефан Константинов Петков	2025	5

фиг. 6.6 Списък с всички студенти в системата

Освен това, администраторът може да управлява церемониите - да създава церемонии, да кани студенти, да редактира церемонии и да преглежда информацията за церемониите. Това включва както преглед на всички церемонии, така и преглед на подробна информация за конкретна церемония.

Списък със церемонии

Дата на завършване	Дата на церемонията	Студент, изнасящ церемониална реч	Студент отговорник за тоги	Студент отговорник за подписи	Студент отговорник за връчване на дипломи	Действие
2025-01-24 23:16:00	2025	3MI0800092 (Unconfirmed)	3MI0800092 (Confirmed)	1MI0800135 (Unconfirmed)	1MI1234567 (Unconfirmed)	Инфо
2025-01-25 12:12:00	2025	0MI0800065 (Unconfirmed)	0MI0800065 (Unconfirmed)	1MI0800135 (Unconfirmed)	1MI1234567 (Unconfirmed)	Инфо

фиг. 6.7 Списък с всички церемонии в системата

### Създаване на церемония

Дата на церемонията:

Година на завършване:

Студент, изнасящ церемониална реч:

Студент, отговорник за церемониалните тоги:

Студент, отговорник за дипломните подписи:

Студент, отговорник по връчване на дипломите:

[Създаване на церемония](#)

фиг. 6.8 Форма за създаване на церемония

### Списък със студенти за церемония

Дата на церемонията: 2025-01-24 23:16:00

Година на завършване: 2025

[Редактиране](#) [Експорт](#)

Факултетен номер	Степен	Имена	Година на завършване	Размер на тоги	Приел	Покана за изнасяне на реч?	Покана за отговорник?
3MI0800092	Бакалавър	Георги Николаев Атанасов	2025	S	Да	подадена	приета, за отговорник за тоги
9MI5678901	Доктор	Яна Правдомирова Василева	2025	M	Не	няма	няма
1MI0800135	Магистър	Ралица Атанасова Симова	2024	Липсва	Не	няма	подадена, за отговорник за подписи
1MI1234567	Бакалавър	Елена Андреева Стоянова	2025	Липсва	Не	няма	подадена, за отговорник за връчване на дипломи
2MI6789012	Бакалавър	Теодора Василева Петкова	2025	Липсва	Не	няма	няма
2MI0876543	Бакалавър	Петър	2025	Липсва	Не	няма	няма

фиг. 6.9 Списък с информация за студентите, поканени на конкретна церемония

## 7. Примерни данни

Примерните данни за базата се намират в поддиректория на проекта „*database/seed/seed.php*“.

Това е файл, който генерира примерни данни и ги вмъква в базата. Примерните данни се състоят от потребителски профили, студентски профили и дрехи.

Полезни примерни данни, генерирани от „*database/seed/seed.php*“ :

1. За потребител тип „админ“:
  - a. Име: IvanPetrov
  - b. Парола: parola123
2. За потребител тип „администратор“, одобрен от „админ“:

- a. Име: Marialvanova
- b. Парола: qwerty123
- 3. За потребител тип „администратор“, неodobрен от „админ“:
  - a. Име: GeorgiGeorgiv
  - b. Парола: pass456
- 4. За потребител тип „студент“, одобрен от „админ“:
  - a. Име: PetarKolev
  - b. Парола: abc12345
  - c. ФН: 2MI9876543

За изпълнението му е необходимо да се изпълни командата:

```
php scripts/seed_database.php seed
```

Това е скриптов файл, чиято цел е да изпълнява файла за генериране на примерни данни в базата. Аргументът *seed* на последно място обозначава поддиректория на папката *database*, в която се намира файла за генериране на данните, който трябва да се казва *seed.php*. Предназначението е по този начин да могат да се създават много файлове за генериране на данни и да се избира между тях в зависимост от ситуацията.

В допълнение към тези данни, има и предварително подготвен файл:

```
examples/imports/students_import.csv
```

Той може да се използва за демонстрация на администраторската функционалност за вмъкване на студенти в системата посредством файлове в CSV формат.

Полезни примерни данни за студенти, генерирани от *examples/imports/students\_import.csv*:

- 1. Студент 1:
  - a. Имена: Георги Николаев Атанасов
  - b. ФН: 3MI0800092
  - c. Година на завършване: 2025
  - d. Степен: Бакалавър
- 2. Студент 2:
  - a. Имена: Петър Станимирев Колев
  - b. ФН: 2MI9876543
  - c. Година на завършване: 2025
  - d. Степен: Бакалавър
- 3. Студент 3:
  - a. Имена: Петър Станимирев Колев
  - b. ФН: 9MI9876549
  - c. Година на завършване: 2027
  - d. Степен: Магистър
- 4. Студент 4:
  - a. Имена: Ралица Атанасова Симова
  - b. ФН: 1MI0800135
  - c. Година на завършване: 2024
  - d. Степен: Магистър

## 8. Описание на програмния код

Приложението е изградено върху основите на *MVC - model-view-controller (модел-изглед-контролер)*. За по-ясно разяснение на софтуерната архитектура на проекта, ще разгледаме файловата структура, разделена на отделни папки и файлове:



1. *database* - Съдържа скриптов файлове за миграциите на базата данни в процеса на разработка на приложението, както и скриптов файлове за генериране на примерни данни за базата с цел тестване и демонстриране на приложението.

```
<?php
require_once __DIR__ . "/../..../database.php";
require_once __DIR__ . "/../..../config.php";

load_config(".env");

$db_name = $_ENV['DB_NAME'];
$dbase = new Database();
$db_con = $dbase->connection();

$db_con->exec(<<<<CT
CREATE TABLE IF NOT EXISTS Users (
    id INT auto_increment PRIMARY KEY,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(256) NOT NULL,
    username VARCHAR(50) UNIQUE NOT NULL,
    role ENUM('student', 'administrator', 'admin') NOT NULL
);
CT);

$db_con->exec(<<<<CT
CREATE TABLE IF NOT EXISTS Students (
    fn VARCHAR(10) PRIMARY KEY,
    degree ENUM('bachelor', 'master', 'doctor') NOT NULL,
    fullname VARCHAR(100) NOT NULL,
    fig. 8.1 Код за първата миграция на базата данни.
```

```
<?php
require_once __DIR__ . "/../..../database.php";
require_once __DIR__ . "/../..../config.php";

load_config(".env");

$db_name = $_ENV['DB_NAME'];
$dbase = new Database();
$db_con = $dbase->connection();

$db_con->exec(<<<<CT
ALTER TABLE Users
ADD COLUMN approved BOOLEAN;
CT);
?>
```

фиг. 8.2 Код за миграция, добавяща нова колона към таблица *Users*

```

$db_con->exec(<<<CT
INSERT INTO Users (email, password, username, role, approved)
VALUES
('ivan.petrov@example.com', '{$hashed_passwords[0]}', 'IvanPetrov', 'admin', 1),
('maria.ivanova@example.com', '{$hashed_passwords[1]}', 'MariaIvanova', 'administrator', 1),
('georgi.georgiev@example.com', '{$hashed_passwords[2]}', 'GeorgiGeorgiev', 'administrator', 0),

('elena.stoyanova@example.com', '{$hashed_passwords[3]}', 'ElenaStoyanova', 'student', 1),
('petar.kolev@example.com', '{$hashed_passwords[4]}', 'PetarKolev', 'student', 1),
('krasimira.dimitrova@example.com', '{$hashed_passwords[5]}', 'KrasimiraDimitrova', 'student', 1),

('nikolay.iliev@example.com', '{$hashed_passwords[6]}', 'NikolayIliev', 'student', 1),
('valentina.marinova@example.com', '{$hashed_passwords[7]}', 'ValentinaMarinova', 'student', 1),
('stefan.popov@example.com', '{$hashed_passwords[8]}', 'StefanPopov', 'student', 1),

('daniela.angelova@example.com', '{$hashed_passwords[9]}', 'DanielaAngelova', 'student', 1),
('boris.kolev@example.com', '{$hashed_passwords[10]}', 'BorisKolev', 'student', 1),
('yana.vasileva@example.com', '{$hashed_passwords[11]}', 'YanaVasileva', 'student', 1),

('mitko.dimitrov@example.com', '{$hashed_passwords[12]}', 'MitkoDimitrov', 'student', 1),
('teodora.petkova@example.com', '{$hashed_passwords[13]}', 'TeodoraPetkova', 'student', 1),
('alexander.stanev@example.com', '{$hashed_passwords[14]}', 'AlexanderStanev', 'student', 1);
CT);

```

фиг. 8.3 Код за генериране на примерни данни и добавянето им в таблица *Users*

2. *models* - Съдържа класове, моделиращи данните от базата и представлява една част от „*model*“ на MVC архитектурата. Съдържа интерфейс за базов модел и конкретни класове наследници.

```

class Ceremony implements IModel
{
    2 references
    private $date;
    2 references
    private $graduation_year;
    2 references
    private $id;

    4 references | 0 overrides
    function __construct($date, $graduation_year, $id = null)
    {
        $this->date = $date;
        $this->graduation_year = $graduation_year;
        $this->id = $id;
    }

    0 references | 0 overrides
    public function to_array()
    {
        return [
            "id" => $this->id,
            "date" => $this->date->format("Y-m-d H:i:s"),
            "graduation_year" => $this->graduation_year,
        ];
    }
}

```

фиг. 8.4 Код за модела *Ceremony*, базиран на таблицата *Ceremony* от базата данни

3. *services* - Съдържа класове с бизнес логика, които най-често боравят с базата за извличане и попълване на данните. Съдържа клас *DataService* с базови функции за комуникация с базата и конкретни класове наследници, комуникиращи с базата относно определени нейни таблици и базирани върху тях модели.

```
class DataService
0 references | 0 overrides
function __construct(Database $database, $model)
{
    $this->connection = $database->connection();
    $this->model = $model;
}

1 reference | 0 overrides
function insert_many_bulk_query($query, $data)
{
    $stmt = $this->connection->prepare($query);
    $stmt->execute($data);
}

1 reference | 0 overrides
function insert_many_with_query($query, $data)
{
    return $this->execute_in_transaction(function() use ($query, $data) {
        $stmt = $this->connection->prepare($query);

        foreach ($data as $entry) {
            $stmt->execute($entry->to_array());
        }
    });
}
```

фиг. 8.5 Код на класа *DataService* за базова комуникация с базата.

4. *components* - Файлове с произползваеми графични елементи, които могат да се вмъкват в различните страници. Съдържа абстрактен клас за базов компонент и конкретни класове наследници.

```
<?php
8 references | 9 implementations
abstract class Component
{
    97 references | 9 overrides
    public abstract function render();
    60 references | 6 overrides
    public static function get_stylesheets()
    {
        return [];
    }
    7 references | 1 override
    public static function get_scripts()
    {
        return [];
    }
}
?>
```

фиг. 8.6 Абстрактен клас за базов компонент

```

<?php
require_once __DIR__ . '/../component.php';
require_once __DIR__ . '/../models/user.php';

```

---

16 references | 0 implementations

```

class HeaderComponent extends Component
{
    97 references | 0 overrides | prototype
    public function render()
    {
        $base_url = $_ENV["BASE_URL"];

        $navigation_items = $this->render_navigation_items();
        return <<<HTML
        <header id="site-header">
            <h2><a href="$base_url">Alumni Hub</a></h2>
            <nav id="site-navigation">
                <ul>
                    $navigation_items
                </ul>
            </nav>
        </header>
        HTML;
    }
}

```

фиг. 8.7 Компонент, моделиращ заглавната секция на сайта на приложението.

5. *pages* - Съдържа изгледите, моделиращи уеб страниците на приложението. Представява *view* частта на *MVC* архитектурата.

```

<body>
<?php
echo $header->render();
?>

<main class="container">
    <h1>Вход в профила Ви в Alumni Hub</h1>

    <form id="login-form">
        <input type="text" id="username" name="username" placeholder="Потребителско име" required>
        <input type="password" id="password" name="password" placeholder="Парола" required>

        <?php
        $submit_button = new ButtonComponent("Вход", ButtonStyleType::Primary, true);
        echo $submit_button->render();
        ?>
    </form>

    <nav class="nav-buttons">
        <?php
        $link = new LinkComponent("Регистрация", "$base_url/register");
        echo $link->render();
        $link = new LinkComponent("Към началния екран", "$base_url/");
        echo $link->render();
        ?>
    </nav>
    <?php echo $message->render(); ?>
    <?php echo $info_message->render(); ?>
</main>

<?php echo $footer->render();
?>
</body>

```

фиг. 8.8 Отрязък от код за страницата за вписване на потребители.

6. *controllers* - Съдържа класове, играещи ролята на контролери от *MVC* архитектурата, свързващи изгледите и моделите посредством бизнес логиката.

```

class StudentsController
{
    1 reference | 0 overrides
    public function show_students_page()
    {
        $controller = $this;
        require_once __DIR__ . '/../pages/students/index.php';
    }

    1 reference | 0 overrides
    public function show_import_students_page()
    {
        require_once __DIR__ . '/../pages/import-students/index.php';
    }

    1 reference | 0 overrides
    public function import_students($data)
    {
        $students = $this->students_import_service->parse_csv_as_base64($data->file);
        $this->students_service->insert_many($students);
    }

    1 reference | 0 overrides
    public function export_students()
    {
        $students = $this->students_service->find_all();
        $this->students_export_service->export($students);
    }

    0 references | 0 overrides
    public function get_students_data()
    {
        return array_map(function ($student) {
            $values = array_values($student->to_array(true));
            array_pop($values);
            return $values;
        }, $this->students_service->find_all());
    }
}

```

фиг. 8.9 Отрязък от кода на класа *StudentsController* с функции за показване на страница (изглед), за импорт и експорт и за събиране на данни чрез сървис.

7. *middleware* - Съдържа допълнителна междинна бизнес логика, действаща като посредник между маршрутизацията и контролерите.

```

class AuthorizationMiddleware {
    function __construct($user_service)
    {
    }

    3 references | 0 overrides
    public function is_authenticated($next, $check_approval = true)
    {
        return function($params) use ($next, $check_approval) {
            session_start();
            if (!isset($_SESSION["id"])) {
                $base_url = $_ENV["BASE_URL"];
                http_response_code(401);
                echo json_encode(["message" => "Неустановен потребител!"], JSON_UNESCAPED_UNICODE);
                header("Location: $base_url/login");
                return;
            }

            $user = $this->user_service->get_user_by_id($_SESSION["id"]);
            $user = $user->to_array();
            $role = Role::tryFrom($user["role"]);
            $base_url = $_ENV["BASE_URL"];

            if ($check_approval && $role !== Role::Student && !$user["approved"]) {
                header("Location: $base_url/not-approved");
                return;
            }

            if ($check_approval && $role === Role::Student && !isset($_SESSION["fn"])) {
                http_response_code(401);
                echo json_encode(["message" => "Неустановен студент!"], JSON_UNESCAPED_UNICODE);
                header("Location: $base_url/not-approved");
                return;
            }

            $next($params);
        };
    }
}

```

фиг. 8.10 Междинна логика по удостоверяване на потребителите на приложението.

8. *scripts* - Съдържа код с полезни скриптове, имащи разнообразни цели.

```
<?php
if ($argc > 3) {
    throw new Exception("Invalid number of arguments");
}

$option = $argv[2];
$action = $argv[1];
$scripts_locations = __DIR__ . '/../database/migrations/';

$output = null;
$code = 0;

if ($option === "-a" || $option === "-all") {
    if ($action === "-m" || $action === "-migrate") {
        $scripts_locations .= "*/migrate.php";
    } else if ($action === "-r" || $action === "-rollback") {
        $scripts_locations .= "*/rollback.php";
    } else {
        throw new Exception("Invalid action");
    }
}

$files = glob($scripts_locations);
$files = $action === "-r" ? array_reverse($files) : $files;
foreach($files as $file) {
    $migration_file = $file;

    exec("php $migration_file", $output, $code);

    if ($code !== 0) {
        die("Migration $migration_file failed\n");
    }
}

echo "\nMigrations run successfully!\n";
return;
}
```

фиг. 8.11 Отрязък от код на скрипт за изпълнение на миграции на базата от *database*.

9. *utils* - Съдържа файлове с общополезни функции

```
<?php
0 references
function parse_query_params() {
    $query_string = parse_url($_SERVER['REQUEST_URI'], PHP_URL_QUERY);
    $query_params = [];
    parse_str($query_string, $query_params);
    return $query_params;
}
?>
```

фиг. 8.12 Полезна функция за обработка на параметри на заявката.

10. *router.php* - Клас с обща функционалност, отговаряща за регистрирането на пътищата в уеб приложението.

```
<?php
1 reference | 0 implementations
class Router
{
    2 references
    private array $routes = [];

    30 references | 0 overrides
    public function register_route($method, $path, $handler)
    {
        $path = preg_replace('/\{(\w+)\}/', '(<P<\1>[^/]+)', $path);
        $path = '#^' . $path . '$#';

        $this->routes[] = [
            'method' => strtoupper($method),
            'path' => $path,
            'handler' => $handler,
        ];
    }

    1 reference | 0 overrides
    public function dispatch($method, $uri)
    {
        foreach ($this->routes as $route) {
            if (strtoupper($method) === $route['method'] && preg_match($route['path'], $uri, $matches)) {
                $params = array_filter($matches, 'is_string', ARRAY_FILTER_USE_KEY);
                call_user_func($route['handler'], $params);
                return;
            }
        }

        http_response_code(404);
        require_once __DIR__ . "/pages/not_found/index.php";
    }
}
?>
```

фиг. 8.13 Класа *Router*

11. *index.php* - Позовавайки се на контролерите от *controllers* и на класа *Router*, дефинира реалните пътища в уеб приложението със всичките им видове заявки (*GET*, *POST*, *PUT*, *PATCH*, *DELETE*).

```

$router->register_route(
    'GET',
    'ceremony/edit/{id}',
    $authorization_middleware->is_authorized(Role::Administrator,
        function ($params) use ($ceremonies_controller) {
            $ceremonies_controller->show_ceremonies_edit_page($params["id"]);
        }
    ));

$router->register_route(
    'PUT',
    'ceremony/edit/{id}',
    $authorization_middleware->is_authorized(Role::Administrator,
        function ($params) use ($ceremonies_controller) {
            try {
                header('Content-Type: application/json');
                $json = file_get_contents('php://input');
                $data = json_decode($json, true);
                $data["id"] = $params["id"];

                $ceremonies_controller->update_ceremony($data);

                http_response_code(200);
                echo json_encode(["message" => "Success"], JSON_UNESCAPED_UNICODE);
            } catch (Exception $e) {
                http_response_code(500);
                echo json_encode(["message" => "{$e->getMessage()}"], JSON_UNESCAPED_UNICODE);
            }
        }
    ));

```

фиг. 8.14 Дефиниране на пътищата за редактиране на церемония.

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

Ралица Симова, 1MI0800135:

Частите от приложението, които разработих са: регистрация на нов потребител, вход във вече направен профил, добавяне и избор на факултетен номер, информация за потребителя, избор на размер за тоги, валидации и показване на грешки на потребителите.

Георги Атанасов, 3MI0800092:

Частите от приложението, които разработих са: панела за админа, импортиране на студенти, преглед и експортиране на студенти, приемане и отказване на покани за церемонии, сигурността покрай овластяването и одобряването на профилите.

Росен Колев, 0MI0800065:

Частите от приложението, които разработих са: създаване на церемонии, изглед със списък на церемониите, изглед със списък на студенти за церемония, експортиране на студенти за церемония, създаване на базата, мигриране на базата и попълване на примерни данни в нея.



Смятаме, че постигнахме относително завършен вид на проекта и че сме покрили основните функционалности за безпроблемна работа с приложението. Като възможност за бъдещо развитие бихме определили добавянето на още функционалности като:

- Добавянето на датата за церемонията в гугъл календара на потребителя
- Потвърждаване на профила чрез изпращане на имейл
- Възможност за смяна на паролата

## 10. Какво научих

Ралица Симова, 1MI0800135:

За първи път разработвам уеб проект с PHP и това ме принуди да разуча нови технологии и принципи. Научих архитектурния стил MVC

Георги Атанасов, 3MI0800092:

Научих повече за разработката на сървърни приложения, които могат да сервират уеб страници, както и за архитектурния стил MVC.

Росен Колев, 0MI0800065:

Научих повече за разработката на сървърни приложения посредством MVC, за създаване и мигриране на бази от данни, за цялостното интегриране на бизнес логика и изгледи в цялостно приложение чрез PHP и за създаване на междинна логика за удостоверяване и овластяване.

## 11. Използвани източници

1. <https://www.php.net/docs.php>, последно посетен на 26.01.2025
2. <https://webdevetc.com/blog/the-front-controller-design-pattern-in-php>, последно посетен на 26.01.2025
3. <https://developer.mozilla.org/en-US/docs/Glossary/MVC>, последно посетен на 26.01.2025
4. <https://mariadb.com/kb/en>, последно посетен на 26.01.2025

Предал (подпис): .....

0MI0800065, Росен Андреев Колев, КН, група 3

Предал (подпис): .....

1MI0800135, Ралица Атанасова Симова, КН, група 3

Предал (подпис): .....

3MI0800092, Георги Николаев Атанасов, КН, група 1

Приел (подпис): .....

/проф. д-р Милен Петров/