**Имена:**  Мирослав Дионисиев **фн:**  *62390* **имейл:** [**miro9122@gmail.com**](mailto:miro9122@gmail.com)**Имена:**  Павел Сарлов **фн:**  *62393* **имейл:** [**sarlovpavel@gmail.com**](mailto:sarlovpavel@gmail.com)

**Имена:**  Илия Вълов **фн:**  *62483* **имейл:** [**Iliyavulov@gmail.com**](mailto:Iliyavulov@gmail.com)

**Начална година:** *2019* **Програма:** бакалавър, (СИ) **Курс: 3**   
**Тема: 8.1 Анонимна вътрешна „поща“ за комуникация  
Дата: 2022-06-01 Предмет: w18prj\_SI \_final**

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

**Предаване:** Задачата се предава в архив с попълнен настоящия документ, проекта/проектите с кодовете, README.txt файл, който описва съдържанието на архива; папка с допълнителни компоненти и използваниресурси**, архива да се казва 9999\_project\_final.zip. (Успех!). (Редактирайте маркираните зони в жълто с коректната информация)**

# ТЕМА: 8.1 Анонимна вътрешна „поща“ за комуникация

## Условие

Да се създаде вътрешна поща за комуникация, която да позволява на потребителите да обменят съобщения в режим анонимен-анонимен, неанонимен-анонимен, неанонимен-неанонимен. Също така да има възможност за комуникация в група с няколко на брой потребители. Да може да се създават стаи (пощенски кутии) чрез CSV файл. Съобщения, застрашаващи разкриването на идентичността на анонимен потребител, да бъдат улавяни.

## Въведение

Closed Communication System (CCS) е система за вътрешна комуникация между студенти. Тя се състои от тематични чат стаи. Във всяка чат стая студент може да участва под формата на анонимен или неанонимен участник. Съобщенията на анонимния участник се проверяват за фрази, които ще нарушат неговата анонимност. Такива съобщения се блокират и изпращат до администратор на системата. Той от своя страна може да изтрива или да одобрява уловените съобщения.

Чат стаи могат да се създават от администратори на системата по следните два начина:

* Чрез ъплоудване на CSV файл със следните данни: име на чат, краен срок, ф.н. на студент, роля на студент, като последните две могат да се повтарят нужния брой пъти.
* Ръчно чрез използване на администраторския интерфейс.

## Теория

Софтуерната архитектура, използвана за *backend*-а е *MVC*. За целта е създадена конфигурация за различните endpoint-и на backend-а, характеризирани със следните параметри:

* Метод на заявката (GET, POST, DELETE, PUT и т.н.)
* Път на заявката (URI) – определя се от пътя до текущата директория на конфигурационния файл, конкатениран с пътя на *entry point*-а (зададения от нас *endpoint* за самия ресурс).
* Автентикация
* Оторизация
* Име на съответния контролер
* Име на съответния метод от контролера

## Използвани технологии

HTML5, CSS3, JavaScript ES6,

Apache/2.4.53 (Win64) OpenSSL/1.1.1n PHP/8.0.19, 10.4.24-MariaDB

## Инсталация и настройки

1. Създаване на файл *DatabaseConfig.php* в папка *Backend/Configs*, който да съдържа следната информация:

<?php

define('DB\_HOST', '<име на хост>');

define('DB\_PORT', '<порт>');

define('DB\_USERNAME', '<потребителско име>');

define('DB\_PASSWORD', '<парола>');

define('DB\_NAME', '<име на базата>');

define('DB\_DIALECT', '<диалект (например: mysql)>');

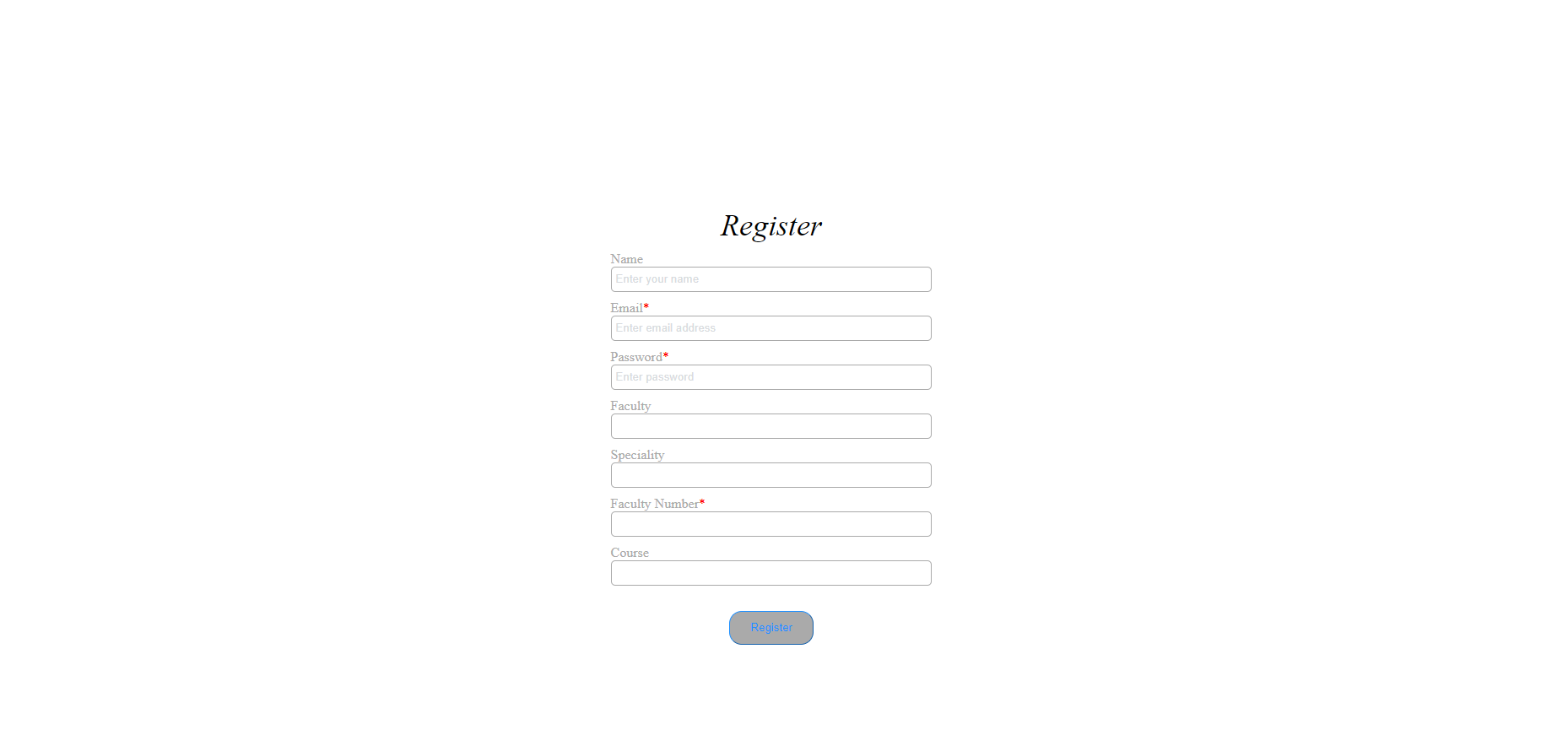
define('DB\_CHARSET', 'utf8');

1. Стартиране на сървърите.
2. Зареждане на съдържанието от файл *Backend/db.sql* и зареждането му в локалната база.

## Кратко ръководство на потребителя

## Регистрация и вход

Потребителите (студентите) се регистрират с имейл и парола, име, факултет, специалност, факултетен номер и курс.



Фигура 1: Регистрация

Потребителите се логват с имейл и парола

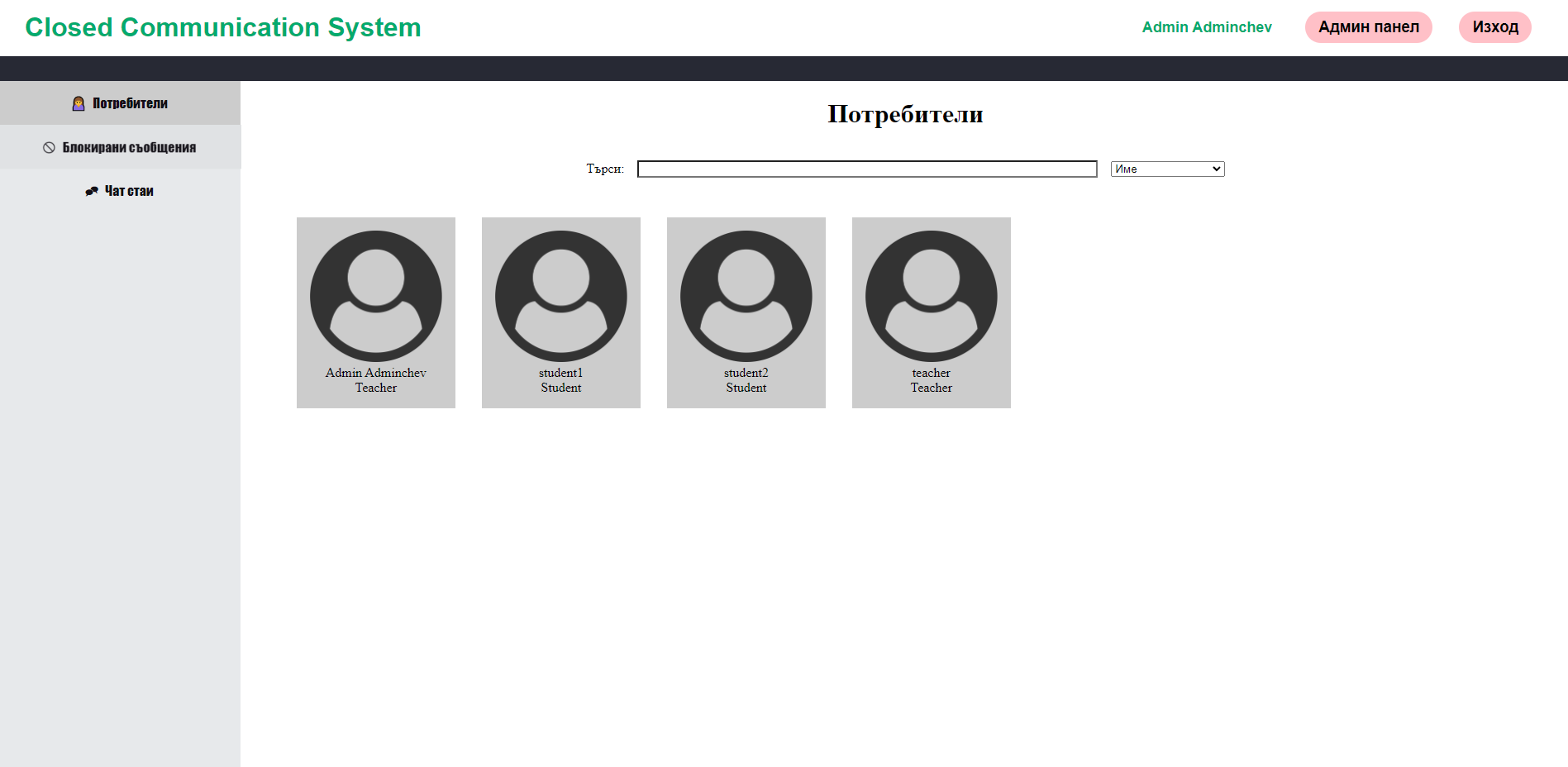


Фигура 2: Вход

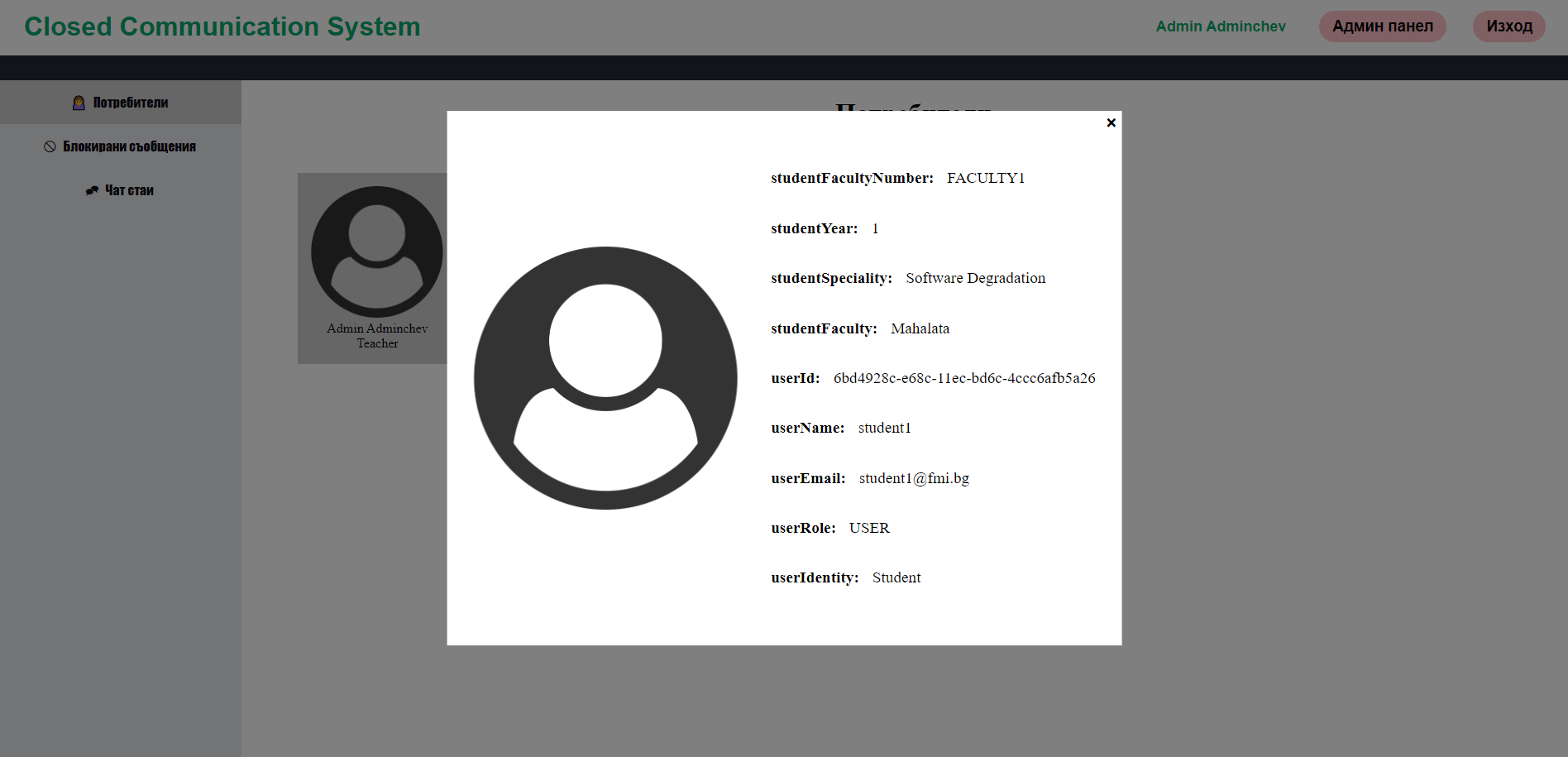
## Администрация

Администратор на системата може да вижда всички потребители, регистрирани в нея.

Фигура 3: Админ панел - списък с потребители.

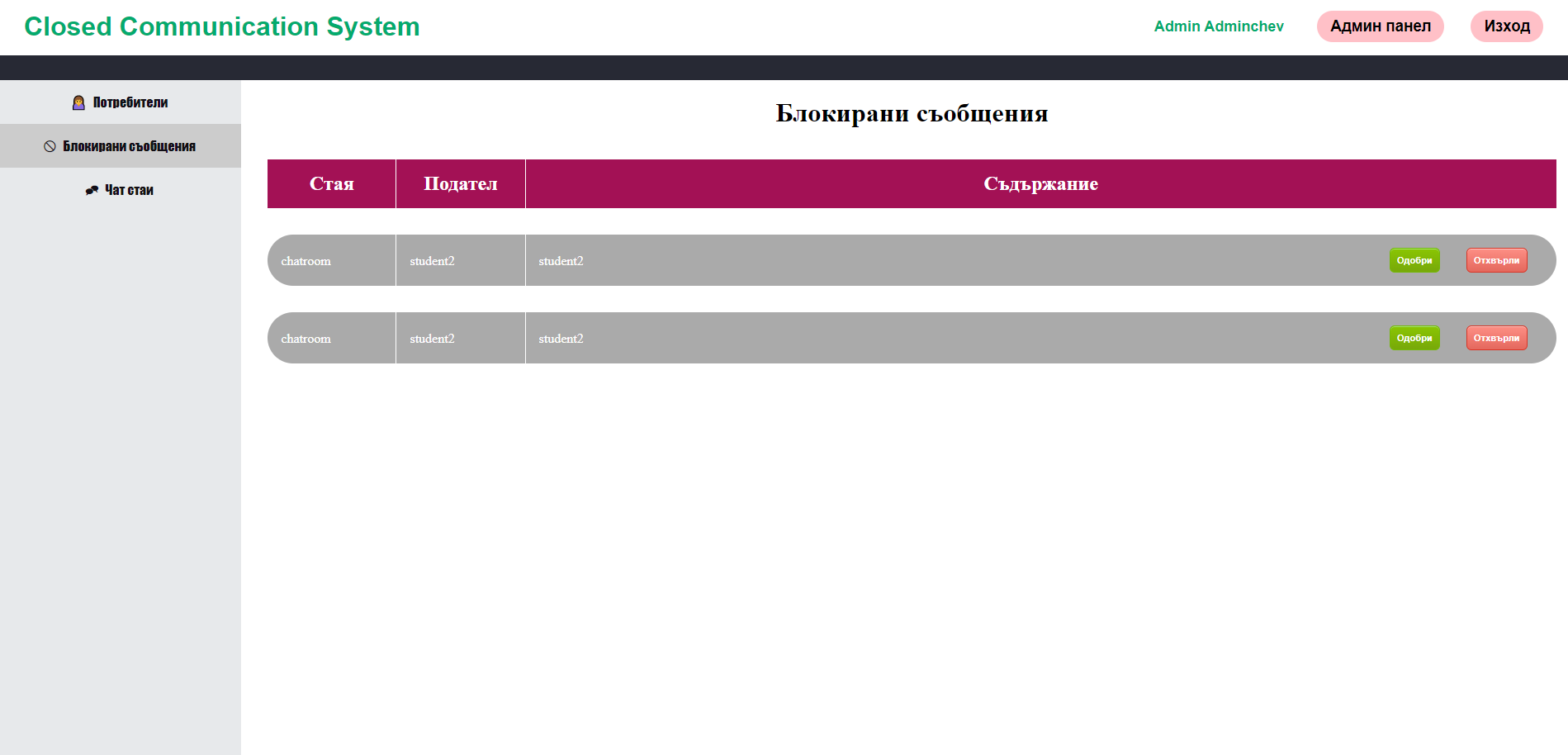


Също така може да вижда данните за всеки отделен потребител.



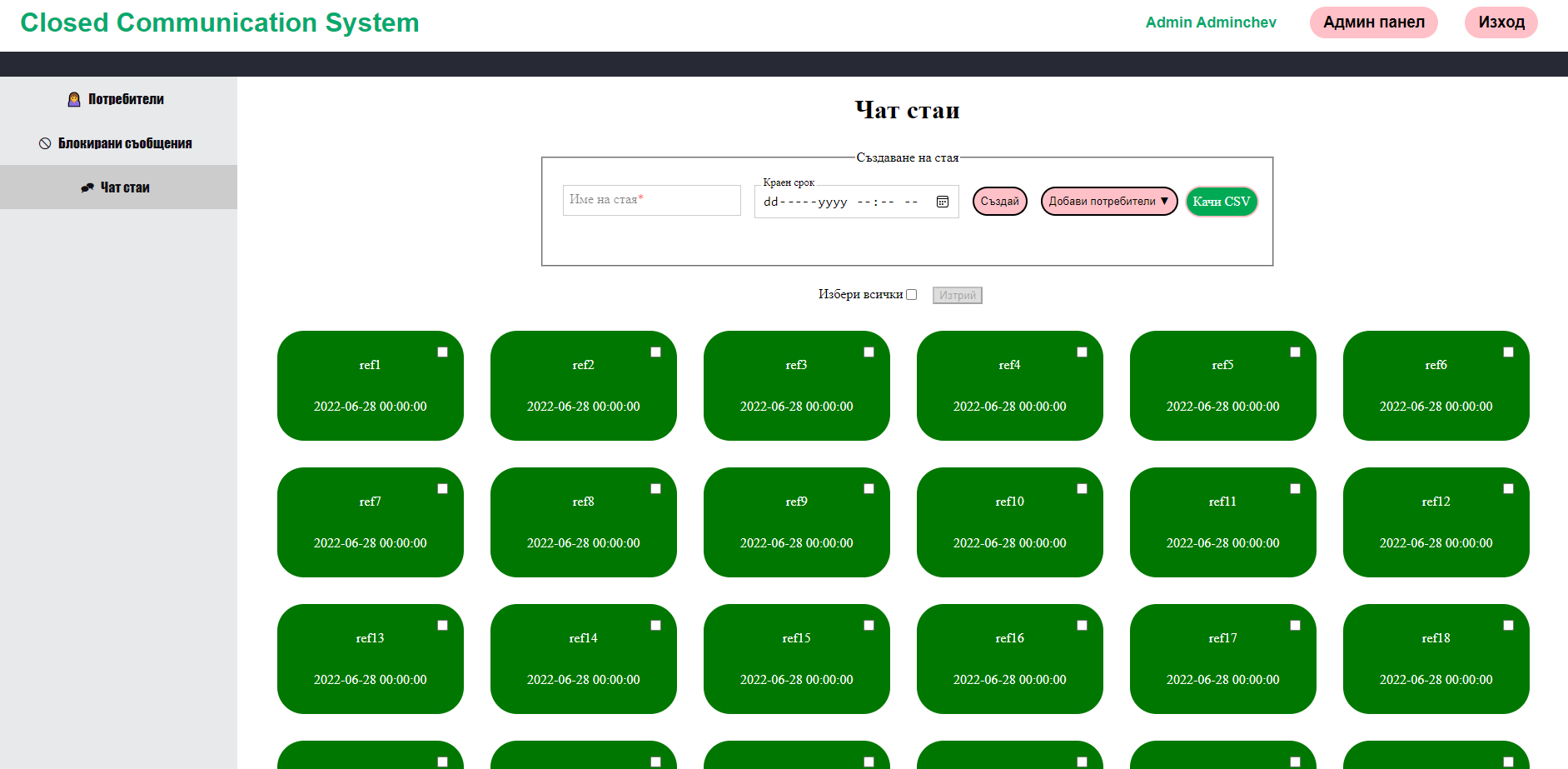
Фигура 4: Админ панел - данни за потребител.

Може да вижда всички блокирани от системата съобщения, които по някакъв начин разкриват самоличността на анонимен потребител.



Фигура 5: Админ панел - блокирани съобщения.

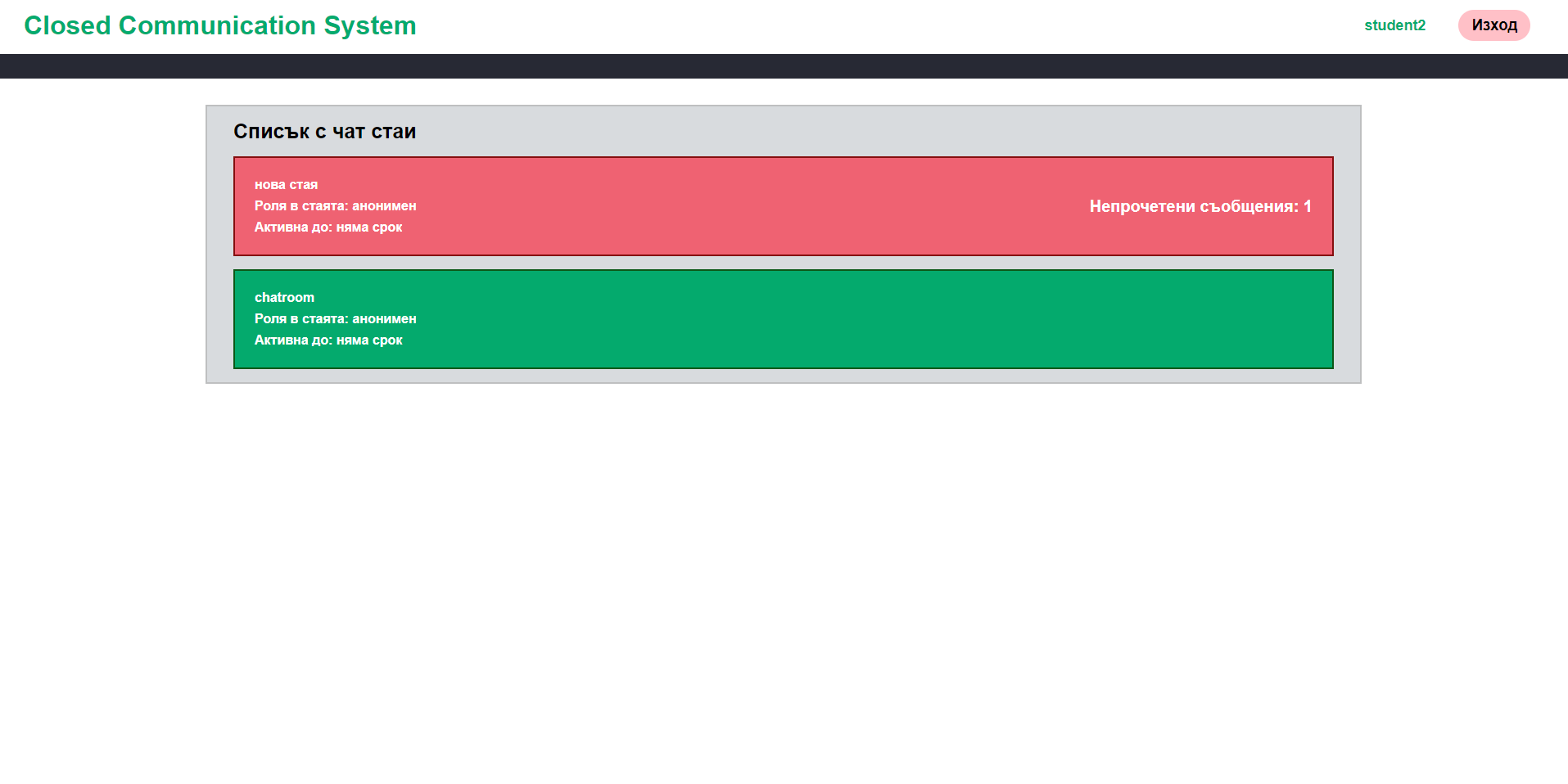
Може да вижда всички създадени чат стаи, да ги премахва, да влиза в тях и да пише съобщения неограничено, да създава нови чат стаи ръчно или да качи CSV файл за тази цел.



Фигура 6: Админ панел - чат стаи.

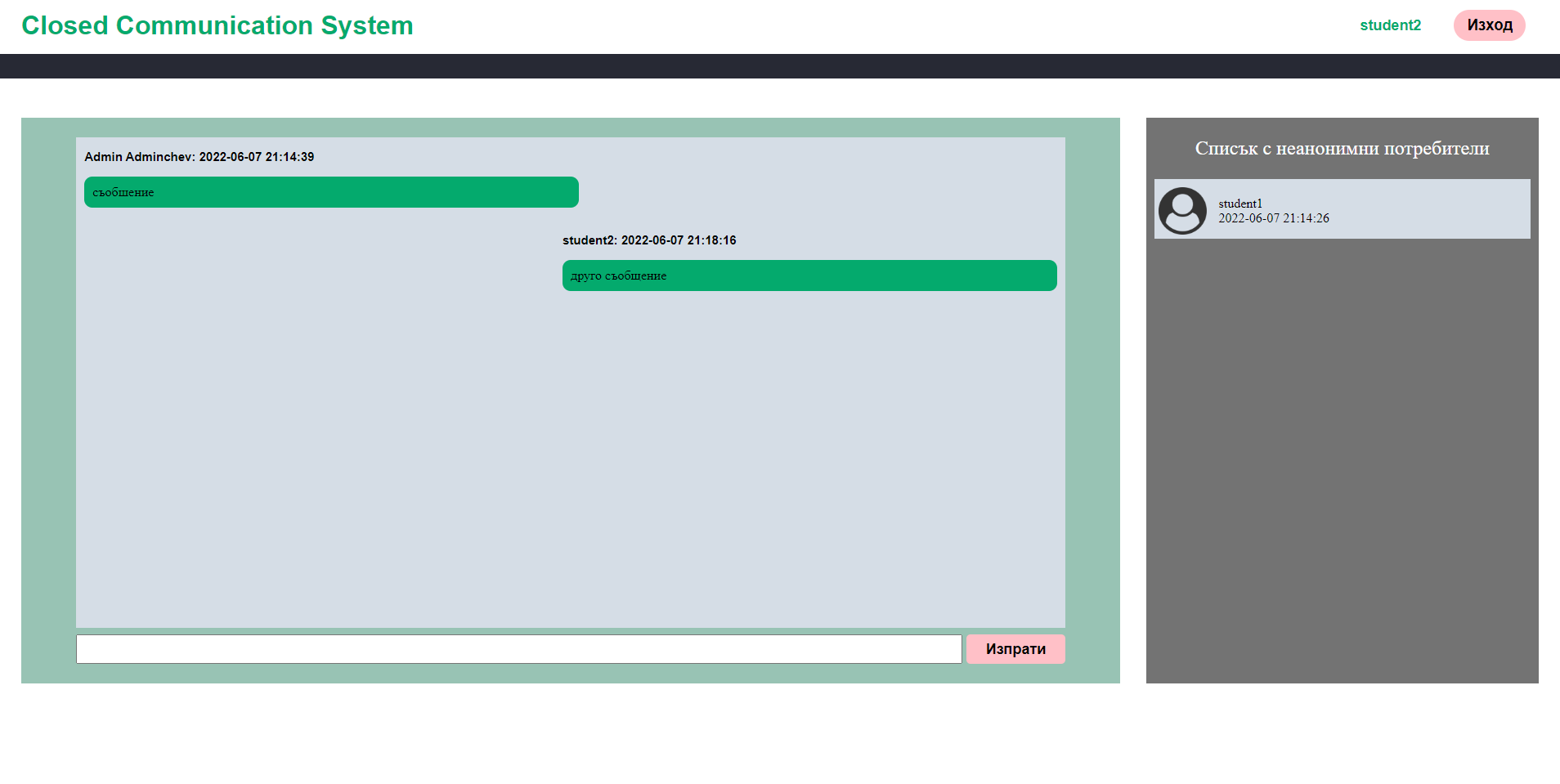
## Потребител

Обикновеният потребител може да вижда всички чат стаи, в които е включен, както и колко съобщения е изпуснал от последното си влизане.



Фигура 7: Потребител - списък с чат стаи.

В самия чат може да вижда всеки друг потребител, включен към него, стига да е неанонимен.



Фигура 8: Чат стая.

## Примерни данни

Във файла за инициализация на базата *db.sql* са предоставени премерни данни за администратор. Отделно във файл *db-populate.sql* има примерни данни за ученици, стая, съобщение.

## Описание на програмния код

Тъй като сме използвали MVC архитектура за целта кода е разделен на:

* Контролери (Controllers)
* Сървиси (Services)
* Хранилища (Repositories)

За иницииране на конекция към базата се грижи *DatabaseConnection* класът, който използва инициалите от конфигурацията. Класът предоставя готов метод за изпълнение на прости заявки *execute*, като отделно може да се вземе и самата конекция с *getConnection* за по-сложни заявки.

Извличането на данни от базата се осъществява от хранилищата, всяко от които отговаря за специфични модели.

Данните са разделени на модели (*Entities* – данни, направо от самата база; *DTOs* – данни за трансфер), като за тяхното трансформиране към обекти и към *Entities*/*DTOs* се грижат *Mapper*-и.

Сървисите отговарят за бизнес логиката на системата, те извличат данни от базата през хранилищата, правят каквото правят и връщат резултат към контролерите.

Контролерите от своя страна отговарят за комуникацията между *frontend*-а и *backend*-а. Т.е. те поемат заявките от потребителите с подадените данни, дават ги на сървисите за обработка и връщат получения отговор на потребителя.

Отделни имаме *Helper*-и, изпълняващи специализирани функции като автентикация/оторизация, управление на съобщенията и управление на глобалните променливи.

*Entry point* на API-то ни е файлът *index.php*, който делегира всякакви заявки към контролерите на *backend*-а спрямо конфигурацията на пътищата в *config.php*.

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

## Мирослав Дионисиев

Изгледите, свързани със стандартния потребител (*User frontpage* и *ChatRoom frontpage*), както и програмната логика, свързана с тях в *JavaScript* и *PHP*.

## Павел Сарлов

Изгледите, свързани с администратора на системата (*Admin frontpage* и съпътстващите го *Users*, *ChatRooms. BlockedMessages*), както и програмната логика, свързана с тях в *JavaScript* и *PHP*.

## Илия Вълов

Изгледите, свързани с вход и регистрация (*Login page, Registration page*), както и програмната логика, свързана с тях в *JavaScript* и *PHP*. Също и качването на CSV файл за съзадаване на стая.

## Ограничения и бъдещо разширение

* Неефективна имплементация на *real-time* чат, в бъдеще може да се ползват WebSocket-и за целта.
* Няма имплементация за ограничаване на възможностите на потребителя да бъде само получател на съобщения.
* Интеграция с puffin.

## Какво научих

Създаване на *MVC* архитектура без *framework*, който да те котка, е много трудно задание. Но пък навлезнахме в идеята на самата структура.

## Използвани източници

[Официална документация на PHP](https://www.php.net/manual/en/)

Предал (подпис): ………………………….

/*62390, Мирослав Дионисиев, СИ, група 5*/

Предал (подпис): ………………………….

/*6293, Павел Сарлов, СИ, група 5*/

Предал (подпис): ………………………….

/*62483, Илия Вълов, СИ, група 1*/

Приел (подпис): ………………………….

/доц. *Милен Петров*/