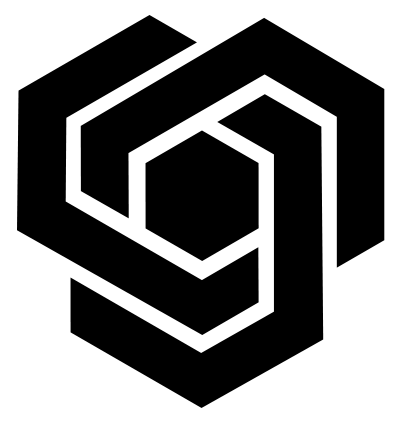
**Технически университет – София**

Факултет по Компютърни системи и технологии



Дипломна работа

Проектиране на уеб-базирана студентска система за връзка с преподавателите

Изготвил*:*

**Грета Петрова Петрова** – фак.№ 121219115

Научен ръководител*:*

**доц. д-р инж Явор Томов**

Увод

Системата “TU-Connect” е разработена с цел да опрости и подобри престоя на студентите по време на тяхното обучение в Техническия университет в София и да улесни взаимодействието на преподавателите с тях. Тази система не само решава единични проблеми, но и се справя с множество предизвикателства, които често възникват през академичната година. Студентите често се нуждаят от контакт с преподаватели поради различни причини - нужда от потвърждение на информация от трета страна, въпроси относно отсъствия, спешни срокове или съвместна работа по дипломни проекти. В такива случаи понякога е трудно да се свържат с преподавателя, тъй като той може да е зает с лекция или да не е в офиса си и последващо да забрави да върне обаждането поради заетостта и отговорностите на професията. Някои преподаватели предлагат само служебен телефон, други споделят общ телефон, а повечето не публикуват своя график на достъпност за студентите. Така, намирането на преподавател понякога се превръща в предизвикателство за студентите.

Системата “TU-Connect” предоставя решения за тези проблеми по няколко начина. На първо място, тя дава възможност на всеки преподавател да въведе своите лекции, упражнения и други ангажименти, които заемат времето му през седмицата. Това позволява на студентите да намерят преподавателя в определена зала или да изчакат завършването на неговия ангажимент, за да не го безпокоят по време на работа. Друга функция на системата е улесняването на организирането на часове за консултации. Всеки преподавател може да въведе информация за своите часове за консултации, включително периода, в който е наличен за дискусии на живо или онлайн, номера на стаята, началния и крайния час, както и допълнителни бележки, напрр линк за онлайн среща. Студентите имат достъп до тези данни и могат да заявят времеви блок за среща с преподавателя. Ако преподавателят одобри срещата, този времеви блок се блокира за други студенти, които търсят среща. Също така, има поле за указване на причината за срещата, което позволява на преподавателя да се подготви с необходимите материали или информация.

Освен това, системата “TU-Connect” предлага решение за улесняване на комуникацията с преподавателите в контекста на дипломните работи, като същевременно осигурява известна степен на анонимност. Студентите могат да публикуват своите идеи за дипломна работа, детайлно описвайки ги, и да ги предложат за разглеждане от преподаватели. Тези публикации са анонимни, за да се избегне лично предубеждение. Когато преподавател изрази интерес към някоя идея и предложи своята помощ като ръководител, системата автоматично изпраща имейл до студента с информация за преподавателя. Студентът може да прегледа всички предложения и да избере преподавателя, който му допада най-много. След приемане на предложението, системата изпраща имейл до преподавателя и разкрива контактите на двете страни, което позволява по-нататъшна комуникация извън системата. Всички други предложения се изтриват от системата. Този подход улеснява процеса на намиране на ръководител и дава възможност на студентите да предлагат множество идеи без притеснение, като същевременно им дава възможност да избягват нежелани преподаватели.

Съдържание

Изложение разделено в глави

1. Преглед на проблема по литературни данни

При разработването на уеб-базираната студентска система за връзка с преподавателите, значително внимание трябва да се обърне към сравняването и анализа на съществуващите образователни платформи като Moodle и Microsoft Teams. Важно е да се разгледат техните ключови функционалности, предимства и ограничения, за да се идентифицират потенциални области за подобрение и иновация.

Moodle, като една от най-разпространените образователни платформи, предлага разнообразие от функции, включително споделяне на учебни материали, публикуване на съобщения и графици за изпити. Тази платформа се е доказала като надежден инструмент за управление на курсове и образователни ресурси. Въпреки това, се наблюдават някои ограничения, особено в областта на директната комуникация между студенти и преподаватели. Например, в Moodle липсва интегрирана система за бързо намиране на помощ от преподаватели или за организиране на консултации, което може да затрудни студентите при търсенето на подкрепа.

От друга страна, Microsoft Teams предоставя отлични възможности за организиране на срещи, благодарение на своя интегриран календар, който улеснява координацията на срещите и управлението на времето. Въпреки това, Teams основно разчита на чат за директна комуникация, което може да доведе до претоварване със съобщения и затруднения в получаването на бързи отговори от преподавателите. Това подчертава необходимостта от по-динамична и ефективна комуникационна платформа в образователния контекст.

В отговор на тези наблюдения, новата система трябва да се стреми към разработването на подобрени и интегрирани функции за комуникация и взаимодействие. Това включва разработването на интуитивен интерфейс за визуализация на графиците на преподавателите, което ще позволи на студентите лесно да идентифицират подходящи времена за срещи или консултации. Също така, важно е да се включи иновативна платформа за споделяне на идеи за дипломни работи, която да предлага възможност за анонимно участие. Тази функция би стимулирала креативността и свободата на изразяване сред студентите, позволявайки им да представят своите идеи без притеснение от лични съдби или предубеждения. В допълнение, системата може да включва модул, който позволява на преподавателите да преглеждат и оценяват предложените идеи, като по този начин се насърчава конструктивният диалог и обратната връзка.

Всичко това ще улесни студентите и преподавателите в ежедневната им работа и ще спомогне за по-ефективното управление на времето и ресурсите.

1. Архитектура на приложението и избор на технологии

Тази глава предоставя ясна рамка за разбиране на целите на системата "TU-Connect", проблемите, които се стреми да разреши, и техническите средства, използвани за постигането на тези цели. Детайлното изложение е от ключово значение за всички заинтересовани страни, включително разработчици, студенти и преподаватели, които вече се възползват от функционалностите на системата.

За тази цел е използван структуриран подход към изложението на задачите и проблемите, които системата има за цел да покрие, както и технологиите, приложени за решаването им. Задачите включват управление на учебните графици, като се предоставя възможност на преподавателите да въвеждат и актуализират графиците си, включващи лекции, упражнения, семинари и други академични ангажименти. Така е облекчена необходимостта от повишена достъпност и прозрачност относно академичните ангажименти на преподавателите.

Организирането на часове за консултации също е сред основните задачи, като се дава възможност на преподавателите да обявяват своите часове за консултации и да предоставят опции за лични срещи и онлайн дискусии. Това е използвано за улесняване на координацията на времето за срещи и консултации. Ефективната комуникация и установяването на директна обратна връзка са също така фокусирани в системата, като се предлагат интегрирани средства за заявяване на срещи, което спомага за улеснената връзка между студенти и преподаватели.

Java е един от водещите и най-използвани програмни езици в света, особено когато става въпрос за изграждане на бекенд услуги за корпоративни приложения. Това е език с висока производителност, който предоставя надеждност и мултиплатформеност, което го прави подходящ за разнообразни операционни системи. Java се отличава със своята устойчивост и скалабилност, което е критично за бизнес приложенията, където обработката на големи обеми данни и паралелната обработка са чести сценарии. Освен това, Java предоставя богат набор от библиотеки и API-та, което разширява възможностите на разработчиците без да се налага да 'изобретяват колелото' за всяка нова функционалност. Стандартната библиотека на Java обхваща всичко от управление на файлове и мрежи до мрежови комуникации и разработка на уеб услуги. Освен това, езикът поддържа многонишково програмиране, което е от съществено значение за изграждането на ефективни и отзивчиви приложения. Платформата Java Enterprise Edition (Java EE) предлага стандартизирана, модулна и мащабируема архитектура, която позволява на разработчиците да създават големи, комплексни бизнес приложения. Java се ползва също заради своята утвърдена защита и сигурност, като предлага редица механизми за криптиране, безопасна аутентикация и авторизация.

От друга страна, Spring Framework е изключително популярен и широко приет в индустрията инструментариум за разработка на приложения в Java. Spring идва като отговор на нуждите за по-лесна разработка и по-добра архитектурна гъвкавост. Той предлага лека инверсия на контрола (IoC) контейнер, който улеснява инжектирането на зависимости и управлението на жизнения цикъл на бийн компонентите. Тази характеристика спомага за декуплирането на компонентите и улеснява тестването и поддръжката на приложенията.

HSQL е въведена като вградена база данни, осигурявайки удобен механизъм за управление на данни без необходимостта от външни сървъри. Това позволява лесна интеграция и бързо тестване в началните фази на разработка, осигурявайки надеждна и ефективна работа.

Postman е използван за тестване на API-тата, като по този начин се осигурява възможност за изпълнение на разнообразни заявки и верификация на коректността на взаимодействието между различните компоненти на системата.

За управлението и автоматизацията на проекта е използван Maven, инструмент, който улеснява конфигурирането, сглобяването и тестването на софтуерни проекти, като по този начин осигурява постоянство в разработката и управление на проектни зависимости.

В разработката на клиентската част на системата са приложени TypeScript и Angular, където TypeScript допринася за типова безопасност и по-удобна разработка, а Angular осигурява структурирана и модулна архитектура за разработка на интерактивни потребителски интерфейси.

Съчетавайки тези технологии, "TU-Connect" предоставя стабилна и функционална платформа, която вече е интегрирана в академичния живот на Техническия университет, облекчавайки ежедневната комуникация и координация между студенти и преподаватели.

//TODO: архитектура

1. Функционалности на приложението

Както името подсказва, основната функция на системата "TU-Connect" е да улесни и оптимизира процесите на свързване и взаимодействие в университетската среда. Част от функционалностите, които предоставя са:

1. Подробен Профил на Преподавателите

Всеки преподавател в системата има свой детайлен профил. Той включва не само основна информация като имена, факултет, катедра и кабинет, но и по-специфични данни като седмичния график на занятията, часове за консултации и предпочитан начин на връзка (например телефонен номер, който е незадължителен). Тази информация е изключително важна за студентите, които се опитват да намерят най-добрия начин да се свържат със своите преподаватели.

1. Изпращане на автоматизирани имейли

При различни процеси в приложението се среща използването на функционалността за изпращане на имейли. Това се случва автоматизирано и потребителите не разбират за него, в повечето случаи кликат бутон и всичко се случва само. В някои случаи се изисква текст за мейла, в други той е шаблонен. Целта е да са кратки, неангажиращи мейли, които да уведомяват без да изтощават с дълго съдържание. Всеки мейл бива изпратен от имейл поща, създадена специфично за приложението, и имейлът на студента бива споделен с преподавателя чак след съгласието на студента, например при приемане на дипломен ръководител.

1. Часове за посещения на преподавател и букване

За да се улесни комуникацията между преподаватели и студенти и да се насърчи срещата на живо, е имплементирана функционалността за часове за посещения. Тя работи като се дава възможност на всеки преподавател да въведе часовете, в които би бил свободен всеки ден да приеме студенти за дискусии или въпроси по различни теми, по които може да бъде полезен. Системата се осъществява чрез времеви блокове. Студентът избира начален и краен час, времеви блок с определена дължина и на заден фон имейл бива изпратен до преподавателят, съдържащ студента, причината на посещението и времевия блок. Той приема срещата или я отменя в зависимост от графика си, като след приемане, времевият блок бива маркиран като зает в профила на преподавателя. Студентът бива уведомен чрез мейл с информация длаи срещата ще се състои или не.

1. Възможност за промяна на време на срещата преди приемане

Ако преподавателят иска да смени нещо по срещата, може да го направи от приложението, може да редактира всичко, включително и кабинета, ако случайно се налага такава промяна. Възможна е и смяна на деня, поради непланирани обстоятелства. Когато е сигурен в срещата, той я приема и нов имейл бива пратен до студента с потвърждение и новите детайли, ако има такива.

1. Преглеждане на седмичен график на преподавател

Системата предлага функционалност за извеждане на седмичния график, попълнен лично от преподавателя, за ориентиране на студента. Данните са въведени с начален и краен час, зала или кабинет и ден от седмицата.

1. Споделяне на идеи за дипломно задание

Приложението има записани идеи за дипломни работи, подадени от студенти. Всеки може да добави своя идея като трябва да опише детайлно проблемите, които решава дипломната, технологиите и всичко, което ще даде представа на преподавателите с какво желае да се занимава дипломантът и дали биха могли да му помогнат. При интерес към някоя, преподавателите могат да заявят желанието си да я поемат, при което се изпраща мейл до студента с информация за преподавателя и евентуални препоръки или идеи за бъдещо развитие на идеята.

1. Поемане на дипломно задание от преподавател

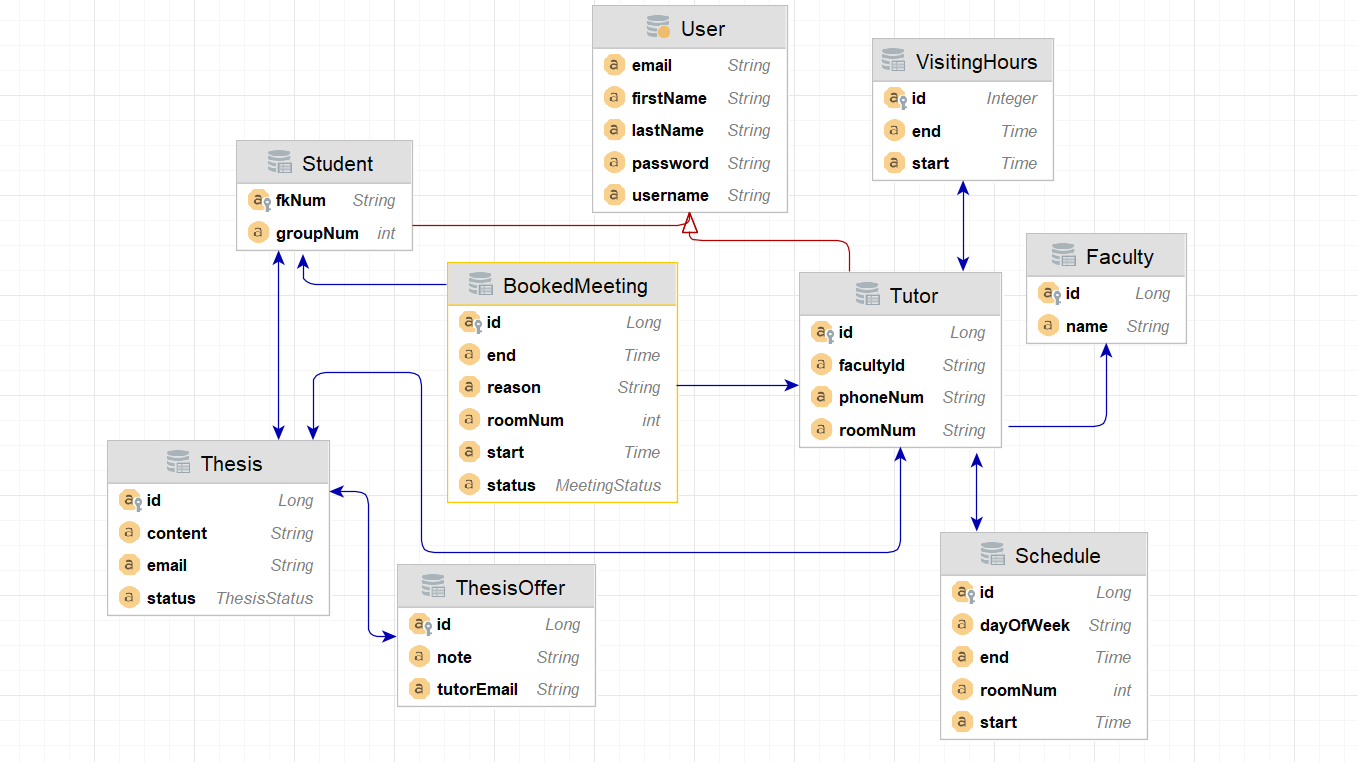
Студентите могат да извадят списък с всички преподаватели, които да изявили желание да поемат идея за дипломна работа. Като всеки студент може да изпрати повече от една идея. Избирайки един от всички, останалите предложения се изтриват, както и самата идея и имейл бива изпратен и до двамата замесени. Личният имейл на студента се изпраща на преподавателя и така той вече знае с кого ще работи по дипломното задание. А студентът получава мейл с потвърждение, че дипломната работа вече има ръоводител и той е уведомен. Оттук комуникацията преминава извън приложението.

1. Анонимност преди приемане на преподавател за ръководител

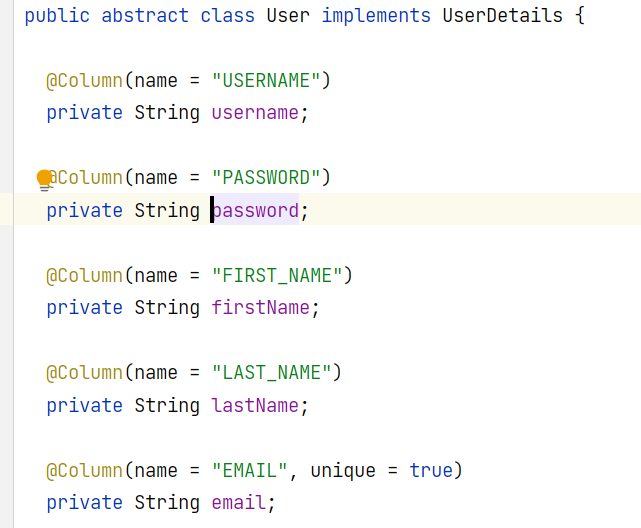
Системата съдържа и функционалност за запазване на личността на студента, докато не бъде приета дипломен ръководител. Целта е да се премахне евентуално притеснение от страна на студените и да могат свободно да изпращат идеи за проекти, които ги вълнуват, както и да се премахне фактора предпочитания и лично отношение от страна на преподвателите. Така цялата комуникация е фокусирана върху самия проект и остава безпристрастна.

1. Бази данни

Схемата по-долу предоставя визуално представяне на структурата на базата данни, която поддържа "TU-Connect". Тя показва как различните ентити се взаимодействат помежду си и формират основата на информационната система, която служи на потребностите на студентите и преподавателите в университета. Връзките са представени със стрелки, показващи отношенията между таблиците - например, еднопосочните стрелки показват, че една запис в родителската таблица може да бъде свързана с множество записи в дъщерната таблица.



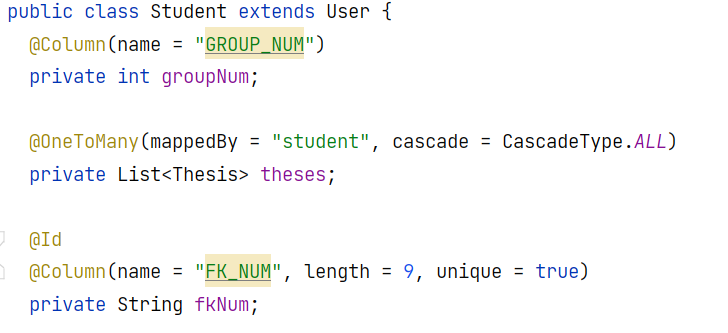
1. *User ->* Наследява обекта UserDetails, съдържа едно поле със задължителна уникалност, email. Имплементациите на UserDetails не се използват директно от Spring Security за цели свързани със сигурността. Те просто съхраняват информация за потребителите, която по-късно се енкапсулира в обекти за автентикация. Това позволява съхранението на информация, свързана с потребителите, независимо от сигурността (като например имейл адреси, телефонни номера и други) на удобно място.

**

Класът съдържа метод за взимане на роля, който е абстрактен и има различна имплементация във съответните деца.



* email (String): Имейл адрес на потребителя.
* firstName (String): Първо име на потребителя.
* lastName (String): Фамилно име на потребителя.
* password (String): Парола за автентикация.
* username (String): Потребителско име за вход в системата.
  1. *Student* -> Наследява User класа, съдържа колекция лист от всички идеи за дипломни работи (Theses), които студентът е предложил пред преподавателите в системата. Има факултетен номер с ограничения за дължина – 9 символа, и уникалност, за идентифициране на студентите. Тук е спестена колоната id и вместо нея се изполва факултетният номер.

**

Имплементацията тук връща ролята на студент.  
A close-up of a computer code

Description automatically generated

* fkNum (String): Факултетен номер на студента.
* groupNum (int): Номер на учебна група.
  1. *Tutor* -> Тук имаме автоматично генериране на id, телефонен номер с лимит 11 цифри, номер на стая с максимална дължина 5, колекциия лист от дипломни рабоит, които преподавателят е поел да управлява като дипломен ръководител, часове за посещение и седмичен график. Тук методът getRole() връща UserRole.TUTORE.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

* id (Long): Уникален идентификационен номер.
* facultyId (String): Идентификационен номер на факултета.
* phoneNum (String): Телефонен номер.
* roomNum (String): Номер на кабинет.

1. *Faculty*

A screenshot of a computer code

Description automatically generated

* id (Long): Уникален идентификационен номер на факултета.
* name (String): Име на факултета.

1. *VisitingHours* -> Тук имаме обект, който пази часовете за посещения на преподавателите. Съдържа 2 времена, начално и крайно и рестрикция за дублирани 2 инстанции с еднакъв период. Изисква проверка при добавяне на преподавател, ако часовете вече ги има, да не се прави опит за добавяне на нова инстанция, а да се преизползва стария обект. Връзката е еднопосочна, защото от страна на този обект не ни интересува кои преподаватели имат съответните часове.

A screenshot of a computer program

Description automatically generated

* id (Integer): Уникален идентификационен номер.
* end (Time): Край на часовете за посещение.
* start (Time): Начало на часовете за посещение.

1. *BookedMeeting*

A screenshot of a computer program

Description automatically generated

A close-up of a white background

Description automatically generated

* id (Long): Уникален идентификационен номер на срещата.
* end (Time): Време за край на срещата.
* reason (String): Причина за нуждата от среща.
* roomNum (int): Номер на стаят, в която ще се проведе срещата.
* start (Time): Време за начало на срещата.
* status (MeetingStatus): Статус на срещата.

1. *Schedule*

* id (Long): Уникален идентификационен номер на графика.
* dayOfWeek (String): Ден от седмицата, в който се провеждат занятията.
* end (Time): Време за край на занятието.
* roomNum (int): Номер на стаята, в която се провежда занятието.
* start (Time): Време за начало на занятието.

1. *Thesis ->* Задължително съдържа content, същността на дипломната работа, както и email, на който да бъдат изпращани заявките за евентуални дипломни ръководители. Статусът е с private достъп, за да може да бъде сменен само чрез методите, дефинирани в класа, като е зададен със стойност по подразбиразбиране ThesisStatus.SUBMITTED. Колекцията лист със заявки от преподаватели се инициализира по същия начин и е възможно само да се добавят към нея заявки, но не и да се присвоява нова колекция.

*A screenshot of a computer program

Description automatically generated*

A screenshot of a computer code

Description automatically generated

* id (Long): Уникален идентификационен номер на темата за дипломна работа.
* content (String): Съдържание или описание на дипломната работа.
* email (String): Имейл адрес за връзка със студента или преподавателя, асоцииран с темата.
* status (ThesisStatus): Статус на дипломната работа.

1. *ThesisOffer ->* Тук задължително поле е мейлът на преподавателя, подал заявката за ръководител, за да може да бъде уведомен при приемане и за да може да му бъде изпратен мейла на студента, чиято дипломна ще ръководи. Незадължитлено поле е note, препоръки или идеи за бъдещо развитие на дипломната идея, които да бъдат добавени като текст на мейла към студента.

*A computer screen shot of a program

Description automatically generated*

* id (Long): Уникален идентификационен номер на предложението за дипломна работа.
* note (String): Бележки или допълнителна информация за предложението.
* tutorEmail (String): Имейл адрес на преподавателя, предлагащ темата.

*Връзки между ентититата:*

User е родителско ентити за Student и Tutor, което подсказва, че студенти и преподаватели са специфични типове потребители с допълнителни атрибути.

Tutor има асоциация с Faculty, всеки преподавател е асоцииран с определен факултет. (@ManyToOne)

Tutor е свързан с VisitingHours, което показва, че преподавателите могат да имат дефинирани часове за посещения, когато са налични за студентите. (@ManyToOne)

Tutor също така е свързан с BookedMeeting, което указва, че преподавателите могат да имат резервирани срещи със студенти. (@OneToMany)

Tutor и ThesisOffer са във връзка, което показва, че преподавателите могат да предложат теми за дипломни работи. (@OneToMany)

Student е свързан с Thesis, студентите могат да предлагат и да работят върху теми за дипломни работи, които имат статус и съдържание. (@OneToMany)

Schedule е свързано с Tutor, показвайки, че преподавателите имат график с определени часове за преподаване в определени дни от седмицата и стаи. (@OneToMany)

1. Програмна реализация, йерархия на класовете

Съдържание

1. API и заявки за тестване

Системата се състои от множество контролери, които формират API (Application Programming Interface) на приложението. Тези контролери управляват различни аспекти на приложението и предоставят интерфейс за взаимодействие между потребителския интерфейс (UI) и сървърната логика. В контекста на тази система, има пет основни контролера: UserController, TutorController, ThesisController, EmailController и BookingController. Всеки от тези компоненти отговаря за различни функционалности на приложението.

1. **UserController**: Управлява всички операции, свързани с потребителските акаунти. Това включва регистрация, автентикация, преглед на профили, актуализации на потребителски данни и управление на потребителски роли.
2. **BookingController**: Управлява резервациите и срещите, като осигурява функционалности за преглед на срещи за студенти и наставници, резервиране на нови срещи, както и потвърждение, модифициране или отказ на вече уговорени срещи.
3. **ThesisController**: Отговаря за управлението на дипломни работи и предложения за тези, като позволява подаване на дипломни работи, добавяне на предложения за тези и приемане на такива предложения.
4. **EmailController**: Фокусира се върху функционалностите свързани с изпращането на имейли, позволявайки на системата да изпраща различни видове електронни писма, базирани на заявки от потребителите.
5. **TutorController**: Управлява взаимодействията с наставниците в системата, включително извличане на информация за всички наставници и получаване на детайли за конкретен наставник по идентификатор.

Заявките ще бъдат описани чрез curl като команден инструмент. Той позволява изпращането на заявки към сървър. Методът на заявката може да бъде разнообразен, като например POST, GET, PUT, DELETE, в зависимост от нуждите на заявката. URL адресът указва към сървъра, към който се изпраща заявката и може да бъде както локален, така и външен. Заглавията в заявката могат да включват Content-Type, който определя формата на данните, например application/json, text/plain, application/xml, и други, както и Cookie или други хедъри за управление на сесии или удостоверяване. Тялото на заявката съдържа данните, които се изпращат към сървъра; при POST заявки често е във формат JSON, докато при GET заявките тялото обикновено е празно и данните се предават чрез URL параметри. Тази структура е гъвкава и позволява адаптация за различни видове заявки и потребности.

/users

Заявкита под /users отговарят за операции свързани с потребителите. Тук има доста възможности за развитие и функционалности, но за момента имаме само 3 разработени заявки, за регистриране, за автентикиране пред приложението и за извеждане на списък със всички потребители.

**POST /users/login**

**За успешна автентикация е нужно подаването на няколко аргумента в тялото на заявка.**

**Задължителни параметри:**

**“username” – това е потребителкското име, с което потребителят е бил регистриран**

**„password“ – това е паролата, с която потребителят е бил регистриран**

**Незадължителни параметри:**

**“role” - това е ролята на профила на потребителя.**

****

**Response:**

**Отговорът, който се очаква от системата при правилни детайли пратени със заявката и правилно поведение на системата е валиден jwt токен, който да може да се ползва от потребителя като се изпрати в хедър заедно с всяка заявка, изискваща автентикиран потребител.******

**Ако се опитаме да се автенитикираме с грешни детайли, например грешно изписана парола, няма да успеем да получим 200 ОК или валиден токен. Правилното поведение на системата ще изисква връщане на отговор със статус 403 и код Forbidden, със Stacktrace и причина за неуспешно минала заявка. Примерен отговор може да се по-долу, както и примерна заявка с грешна парола.**

**Request:******

**Response:******

**POST /users/register**

**Това е заявката, с която потребителят може да се регистрира в системата и да получи достъп до нея и повечето функционалности. Изискват се задължителни параметри за изпращане заедно със заявката.**

**Задължителни параметри:**

**“username” – това е потребителкското име, с което потребителят ще бъде регистриран**

**„password“ – това е паролата, с която потребителят ще бъде регистриран**

**Незадължителни параметри:**

**“role” - това е ролята на профила на потребителя**

****

**Note: Ролята се представлява от енъм, който е възможно да бъде със стойност “STUDENTE” или “TUTORE”, като при липсващ параметър се задава по подразбиране ролята на студент. За да бъде успешна заявката е задължително параметърът да не бъде изписан по какъвто и да е друг начин, а специфично с главни букви.**

**Response:  
При правилна заявка и очаквано поведение на системата, отговорът, който трябва да се върне ще бъде със статус 200 ОК без тяло.**

****

**GET /users**

**Заявка за взимане на колекция от потребители. Тази заявка може да се изпълни само при автентикиран потребител, затова се изисква хедър с jwt токена, който ще каже на бекенда, че съответната заявка е от потребител на системата и ще върне потребителите като резултат. Хедърът трябва да се казва “Authorization” и токенът трябва да бъда енкоднат??**

****

**/api/tutors**

Заявките под този път се занимават с операциите свързани с преподавателите. Чрез тези заявки можем да получим всички преподаватели от базата, конкретен преподавател по подаден идентификационен номер и евентуални бъдещи попълнения към функционалността на контролера, например изтриване на преподавател или промени по данните на преподавател биха могли да се добавят тук.

**get** /api/tutors

**get** /api/tutors/tutor/{id}

/api/thesis

Тези заявки, намиращи се под /api/thesis, се занимават с менажирането на функционалностите около дипломните задания. Тук са имплементирани възможностите за изпращане на дипломна идея от студент за преглеждане от преподавателите, за извеждане на всички дипломни идеи от преподавателите, за заявяване на желание от преподавател да поемен ролята на дипломен ръководител по конкретна дипломна идея и за отговор от страна на студента. Той може да изпрати заявка с потвърджение на ръководството или да откаже, като е включена функционалността, че при приемане на ръководител, всички останали биват автоматично отказани.

**post** /api/thesis/submit

**post** /api/thesis/offer

**post** /api/thesis/accept/offer/{id}

**POST** /api/thesis/decline/offer/{id}

/send-email

Тук имаме само една имплементирана заявка, която основно се изпълнява вътрешно и не се вика външно от фронтенда например. Разбира се при развиване на системата е възможно изпращането на имейли директно от графичния интерфейс на приложението.

**post** /send-email

/api/booking

Тук се намират заявките, които се занимават с функционалността за резервиране на час с конкретен преподавател. Оттук е възможно да се изведе списък с всички уговорени срещи на конкретен студент, всички уговорнеи срещи на преподавател със студенти. Имплементирана е и функционалност за заявяване на среща в конкретен час, която ще бъде резервирана при изпълняването на друга заявка от страна на преподавателя, която ще приеме тази уговорка. В този контролер съществува и заявка, чрез която може да се приеме уговорка от преподавателя с определени промени по предложените от студента, както и да се откаже такава при несътветствие с календара му или изникнали спешни анагжименти.

**get** /api/booking/student/{id}

**get** /api/booking/tutor/{id}

**post** /api/booking/book

**post** /api/booking/accept/meeting/{id}

**post** /api/booking/acceptWithChanges/meeting/{id}

**post** /api/booking/decline/meeting/{id}

1. Подобни приложения, недостатъци, бъдещи подобрения

Централизиран и Достъпен Интерфейс

"TU-Connect" предлага единна, лесна за използване платформа, където цялата информация и функционалности са налични на едно място. Това прави процеса на намиране на информация и свързване с преподаватели бърз и неусложнен.

Спестяване на Време и Ресурси

Освен подобряване на комуникацията, системата спомага за ефективно използване на времето и ресурсите както на студентите, така и на преподавателите. Предвидените функционалности намаляват нуждата от неефективна комуникация и ненужни срещи.

Източници

1. [Technical University of Sofia](https://www.tu-sofia.bg/department/preview/27?dep_id=90)
2. източник
3. източниk
4. [Enabling Cross Origin Requests for a RESTful Web Service](https://spring.io/guides/gs/rest-service-cors/)
5. [Angular - HTTP POST Request Examples](https://jasonwatmore.com/post/2019/11/21/angular-http-post-request-examples)
6. [Angular 9 - Basic HTTP Authentication Tutorial & Example](https://jasonwatmore.com/post/2020/04/29/angular-9-basic-http-authentication-tutorial-example)
7. [Cookie-based JWT Authentication with Spring Security](https://medium.com/spring-boot/cookie-based-jwt-authentication-with-spring-security-756f70664673)
8. [How to Use Cookies in Spring Boot](https://dzone.com/articles/how-to-use-cookies-in-spring-boot)
9. [Github - jsonwebtoken](https://github.com/auth0/node-jsonwebtoken?ref=blog.angular-university.io)
10. [Angular Authentication With JSON Web Tokens (JWT): The Complete Guide](https://blog.angular-university.io/angular-jwt-authentication/)
11. [JWT.IO](https://jwt.io/)
12. [Github - Flex Cheetsheet](https://yoksel.github.io/flex-cheatsheet/#section-flex)
13. [Angular - Common Routing Tasks](https://angular.io/guide/router)