Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка
до дипломної роботи

другий (магістерський)
(рівень вищої освіти)

на тему: Інформаційна технологія опрацювання заявок абітурієнтів

Виконав: студент 6 курсу групи КН-61м
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Галюк Т.В.
(прізвище та ініціали)

Керівник Дендюк М.В.
(прізвище та ініціали)

Рецензент Грицюк Ю.І.
(прізвище та ініціали)

Львів – 2021

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Крошний І. М.
"____" _____20___ року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Галюк Тарас Васильович
(прізвище, ім'я, по батькові)

1. Тема роботи **Інформаційна технологія опрацювання заявок абітурієнтів**

керівник роботи **Дендюк Михайло Володимирович, доцент, к.т.н.**,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від **"31" грудня 2021 року № С-593**

2. Термін подання студентом роботи 10 грудня 2021

3. Вихідні дані до роботи Розробити інформаційну систему для опрацювання заявок абітурієнтів. Система дозволяю «адмініструвати» заявки подані вступником в заданий університет. Реалізувати процес реєстрації та авторизації. Створити зручний та інтуітивний дизайн з яким легко працювати.Реалізувати систему сповіщення вступника та можливість передачі документів. Розробити архітектуру бази даних яка буде виступати основним місцем збереження даних. Реалізувати даний функціонал засобами Java та Angular.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Стан проблемної області
Інформаційне та математичне забезпечення
Програмне та технічне забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Підготовка матеріалу до доповіді

6. Дата видачі завдання **18 грудня 2020р**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	28.12.20-20.01.21	
2.	Постановка задачі і її формалізація	20.01.21-28.02.21	
3.	Виконання вхідного етапу технологіями Java та Angular (Розробка архітектури).	01.03.20-15.05.20	
4.	Розробка шаблону застосунку. Зовнішній вигляд.	20.03.21-01.07.21	
5.	Реалізація основного функціоналу. Тестування застосунку.	01.07.21-02.09.21	
6.	Оформлення записки до дипломного проекту.	03.09.21-20.11.21	
7.	Здача пояснювальної записки на рецензування.	10.12.21	
8.	Підготовка доповіді.	10.12.21-20.12.21	

Студент _____ Галюк Т.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дендюк М.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота присвячена розробці та створенню інформаційної системи, яка дозволяє адміністратору опрацьовувати заявки на вступ до університету. У розробленому застосунку роботу реалізовано за допомогою фреймворку Spring Boot, який є легкий у використані та дозволяє досить швидко реалізувати backend частину, в основі якої лежить REST API. Для frontend частини використовувався продукт компанії Google - фреймворк Angular. Даний фреймвок є простим у використанні і має досить великий набір бібліотек, які в свою чергу прискорюють процес розробки і зменшують час, який потрібен для початку роботи. Інтерфейс інформаційної системи є мінімалістичний і досить інтуітивно зрозумілий, що забезпечує легку роботу користувачеві і дозволяє легко використовувати всі можливості данної системи.

Ключові слова: Інформаційна система, Java, Angular, Spring Boot, фреймворк, REST API, backend, frontend, програмне забезпечення, застосунок.

ABSTRACT

Thesis is devoted to the development and creation of an information system that allows the administrator to process applications for admission to the university. In the developed application work with the help of Spring Boot is implemented. This framework is easy to use and allows you to quickly implement the backend part which is based on the REST API. Google's Angular framework was used for the frontend part, this framework is easy to use and has a fairly large set of libraries which in turn speed up the development process and reduce the time required to get started. The interface of the information system is minimalist and quite intuitive, which provides easy work for the user and allows you to easily use all the features of this system.

Keywords: Information system, Java, Angular, Spring Boot, framework, REST API, backend, frontend, software, application.

ТЕХНІЧНЕ ЗАВДАННЯ

Створити інформаційну систему яка дозволяє користувачеві (адміністратору) опрацювати заявки які були подані вступником в певний університет.

Розробити процес авторизації

Створити графічний інтерфейс який буде багатофункціональним і водночас простим у використані

Реалізувати графічні компоненти для відображення списку вступників відповідно до заданих критерії та фільтрів

У випадку якщо адміністратор опрацював заявку вступника як «Рекомендовано до зарахування» потрібно повідомити вступника про те що його заявка на вступ до даного університету успішно опрацьована, надіслати лист на електронну пошту, інтегрувати систему із «Новою Поштою» для можливості відправки документів.

Розробити мессенджер ботів(Telegram, Viber, тд.) які будуть сповіщати вступника про те що його заявка на вступ до даного університету успішно опрацьована.

ЗМІСТ

ВСТУП

Сучасний світ важко уявити без використання інтернету та всіх можливостей які він нам надає, він відіграє важливу роль у житті кожної людини. Це дозволяє нам отримувати інформацію, читати новини та ділитися різними файлами, фотографіями та документами. Зараз люди можуть обмінюватися документами юридичного характеру, подавати документи з електронними печатками, вести банківську справу, не виходячи з дому. Завдяки йому ми можемо спілкуватися з іншими людьми в будь-якій точці світу. Додаток дозволяє абітурієнтам подавати документи на вступ до університету зручно та з доступним інтерфейсом.

Об'єктом дослідження є інформаційна система, яка дозволяє опрацьовувати заявки вступників.

Предметом дослідження є дослідження процесу опрацювання заявок вступника, а саме можливість покращити уже існуючі процеси, які відносять до закладів освіти.

Метою роботи є створення інформаційної системи опрацювання заявок вступника.

Для досягнення поставленої мети необхідно виконати такі завдання:

- Розробити процес авторизації;
- Створити графічний інтерфейс, який буде багатофункціональним і водночас простим у використані;
- Реалізувати графічні компоненти для відображення списку вступників відповідно до заданих критерії та фільтрів;
- Інтегрувати систему із «Новою Поштою» для можливості відправки повідомлення про те, що його заявка на вступ до даного університету успішно опрацьована;
- Реалізувати також надсилання листа на електронну пошту про те, що заявка абітурєнта на вступ до даного університету успішно опрацьована;
- Також розробити мессенджер ботів (Telegram, Viber, тд.), які будуть сповіщати вступника про те, що його заявка на вступ до даного університету успішно опрацьована;
- Протестувати розроблену систему.

Практичне значення полягає в тому, щоб зробити систему опрацювання заявок децентралізованою, тобто кожен заклад освіти опрацьовує тільки заявки, що подані в його університет.

Наукова новизна полягає в зміні підходу для опрацювання поданих заявок та можливості використовувати саме електронні документи та прискорити процес опрацювання заявок.

Результати роботи апробовано на третій науково-практичній конференції студентів, аспірантів та молодих вчених «Комп'ютерне моделювання та інформаційні технології» (Львів, 14-16 жовтня 2021 р.):

1. Haliuk T. Information technology for processing applicants' applications / T. Haliuk, M.V.Dendiuk // Комп'ютерне моделювання та інформаційні технології: матеріали третьої науково-практичної конференції студентів, аспірантів та молодих вчених (Львів, 14-16 жовтня 2021 р.). – Львів: кафедра інформаційних технологій НЛТУ України, 2021. – С. 58-61.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Огляд проблемної області

Термін «абітурієнт» значення якого походить від латинського абітуриуса, маэє значення як «той що повинен піти».

Є й інший етимологічний варіант: згідно з великим енциклопедичним словником, сучасне поняття походить від нового латинського abituriens — «людина, яка збирається піти». Наразі даний термін використовується в більшості країнах саме в формі дослівного перекладу, вважаючи, що конкурсантами є учні шкіл, коледжів і гімназій, які готуються до випускного іспиту і планують залишити школу, а також отримали «конкурсантку» - атестат середньої школи. .

У сучасному тлумачному словнику подано такі слова та речення: абітурієнт - особа, яка вступає до вищого або середнього спеціального навчального закладу.

В українському законодавстві використовується термін «абітурієнт» — особи, які подають заявки на участь у олімпіадах вищої освіти.

1.2. Актуальність розроблення застосунку

Оскільки сучасний світ постійно розвивається і не стоїть на місці, все більше компаній в реальному світі відмовляються від паперового документообігу. На мою думку, для такої ситуації є кілька причин, а саме:

- **Швидкість опрацювання електронних документів**

  Якщо звернути увагу на те, скільки часу займає підготовка звіту та фактична відправка його одержувачу, цей процес займе більше часу, ніж використання електронної версії

- **Зручність і легкість роботи з електронними документами**

Зауважте, що в епоху, коли комп’ютери не були звичними та звичними, вся робота, облік і розрахунки виконувались у паперових періодичних виданнях, які займали місце на столі чи складі й витрачали більше часу на пошуки й роботи. Замість цього можна використовувати електронні документи, які прості у використанні, обробці та пошуку.

- **Актуальність інформації**

  Більше половини друкованих документів втратили актуальність в день їх друку. Потрібен час, щоб інформація була роздрукована та досягнута користувача.

- **Екологія та ресурси**

  Сьогодні суспільство серйозно замислюється над тим, як покращити екологічні умови на нашій планеті. Вони намагаються знайти альтернативи паперу, щоб зупинити вирубку лісів. Тому в сучасних умовах впровадження систем обробки електронних заяв і документів є серйозним досягненням в області обробки інформації та екології.

Висновки до розділу

Оскільки все більше і більше компанії переходять на електроний обіг документів та відмовляються від паперових то створення та використання платформ які працюють тільки з електронними документами дозволяють спростити час на їх опрацювання, згрупувати документи відповідно до певних критерій та відправляти електронні документи набагато швидше ніж їх паперові версії.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Java та Angular

Java – це об'єктно-орієнтована мова програмування, створена Sun Microsystems у 1995 році. З 2009 року компанія Oracle підтримує розробку мови.

Java є основною мовою для курсів та іспитів американських старшокласників. Звичайні студенти швидше вивчають цю мову програмування: зрозуміти основні принципи дуже просто, при цьому можна вирішити практично будь-яку задачу розробки.

Крім того, синтаксис Java легко вивчити, дозволяючи за короткий час вивчити основи програмування. До переваг мови відоситься:

Кросплатформність

Java не є першою мовою для написання кросплатформних програм, але вона стала найпопулярнішою мовою: її головна перевага в тому, що ми пишемо код для программи тільки один раз і через можливості байт коду можемо виконати скомпільований код на інший пристріях які підтримують Java. Програміст Java може написати застосунок на комп’ютері, а потім запустити її на мобільному телефоні, сервері тощо. Якщо пристрій має найнижче середовище запуску, програма буде працювати без проблем.

Активна спільнота

Без гідної підтримки проста і зручна мова програмування не виживе. Спільнота – одна з головних переваг мови та платформи Java: тут багато активних форумів, організацій і відкритий код Stack Overflow. Це дозволяє отримати допомогу від програмістів з будь-якої точки світу.

Angular

Angular – це інтерфейсний фреймворк з відкритим кодом, розроблений командою Google Angular та спільнотою приватних розробників і підприємств. Angular – це AngularJS, переосмислений і повністю переписаний тією ж командою

розробників. Основною особливістю Angular є те, що фреймворк розроблений для роботи на цьому типі мови програмування.

TypeScript мова веб-додатків, що використовується для розширення функціональності JavaScript. Основні переваги цієї мови полягає в тому що на відміну від JavaScript в цій мові реалізована строга типізація, ооп реалізація,

Переваги TypeScript:

TypeScript – це набір скомпільованого JavaScript з наступними перевагами:

- Додатковий тип змінної;
- Тип класу, який забезпечує певні типи переваг без фактичного використання;
- Можливість використовувати особливості мови а саме автодоповнення (IntelliSense).
- Підтримка основних браузерів із стандартами EcmaScript 6 і EcmaScript 7;
- Може бути скомпільований у версію JavaScript, яка підходить для всіх браузерів;

2.2. Можливості Java як серверної частини

Особливістю мови програмування Java є те, що її можна використовувати для написання високонавантажених веб-програм. Java досягла великих успіхів за 25 років і все ще стає популярною.

На данний момент використуться один з двох технічних стеків для написання веб сервісів а саме: Spring Framework або Java EE, від недавнього часу Jakarta Edition. Spring стек активно розвивається спільними зусилями джава розробників що в свою чергу спрощує роботу новачкам та прискорює процес навчання.

Особливістю Spring Framework є те, що основною частиною є контейнер IoC (інверсія управління), який реалізує принцип Dependency Iijection (ін'єкції залежності).Spring Framework складається з декількох основних модулів а саме:

- Spring Boot (Реалізація REST API на основі HTTP протоколу)
- Spring Framework він же Spring Core
- Spring Data JPA (Робота з базою данних)
- Spring Security (аутентифікація та авторизація)

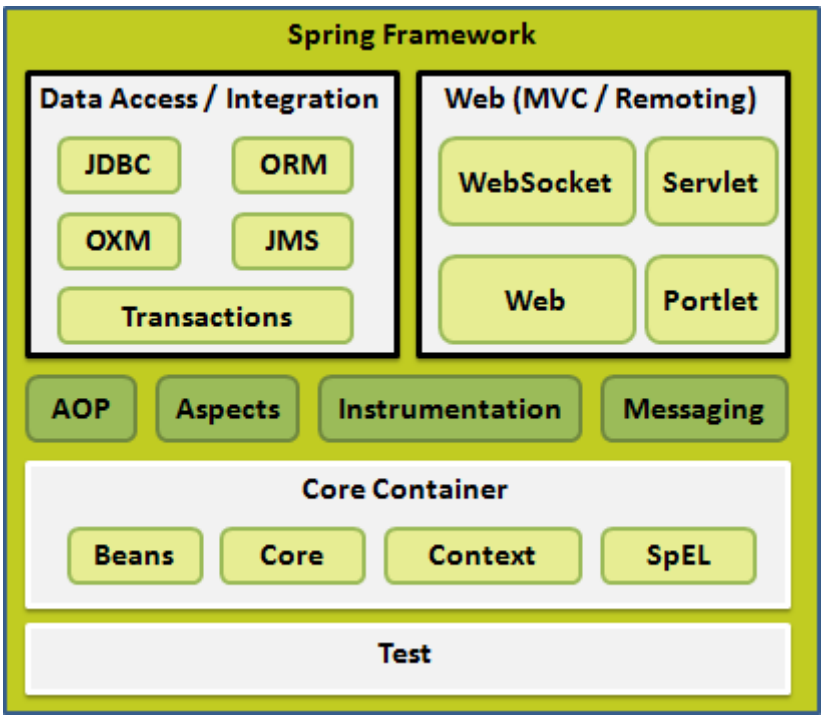


Рисунок 2.2.1 Основні частини Spring Framework

2.3. Особливості Angular як клієнтської частини

Як згадувалося вище, Angular був розроблений Google і відіграє важливу роль у сучасному світі веб-розробки. Для тих, хто не знайомий зі статичною типізацією, цю структуру легко зрозуміти. Розробники Java або C# не зіткнуться з особливими проблемами під час вступу. За допомогою TypeScript розробка на стороні клієнта відбувається набагато швидше, ніж звичайний JavaScript. Завдяки існуванню типів, TypeScript також уникає або обробляє велику кількість помилок під час компіляції.

Цей фреймворк зручний тим, що компанії-розробники намагаються створювати зручну документацію і мають великі можливості використовувати готові рішення.

Завдяки менеджеру пакетів npm ви можете завантажити необхідні бібліотеки однією командою.

Однією з переваг Angular є те, що він має величезну базу готових графічних компонентів (Angular Material), яка проста і гнучка у використанні, а також може налаштовувати компоненти.

До переваг Angular відноситься:

1. Функціональність

Налаштування за замовчуванням надають вам все необхідне для використання вбудованих інструментів Angular. До них належать інструменти маршрутизації, щоб ви могли легко отримати дані, які будуть відображатися в програмі, модулі http і велика кількість інших додаткових бібліотек які часто використовуються в розробці. Попередньо налаштоване середовище Angular не тільки допомагає збільшити швидкість розробки, але й допомагає при тестуванні.

Для того щоб розробляти розробляти на Angular вам не потрібна велика кількість різних бібліотек з різних джерел, не потрібно слідкувати за тим чи данні бібліотеки активно використовуються та чи підтримуються взагалі. Для розробки потрібен тільки основний модуль Angular який уже містить всі потрібні бібліотеки які є частиною екосистеми фреймворка

2. TypeScript

Основна перевага цієї суворо типізованої мови полягає в тому, що вона може допомогти розробникам зберегти свій код чистим і зрозумілим. Оскільки ви можете переглядати поширені помилки під час введення, легше виявляти й виправляти помилки. Це швидше під час налагодження та легше підтримувати великі бази коду, що особливо корисно для великих корпоративних проектів.

3. Послідовність

На відміну від React, Angular є повноцінною системою веб-розробки. Ключовою особливістю є те, що існує зручний спосіб створення компонентів, сервісів або модулів.

На практиці це покращить узгодженість усієї бази коду — важлива мета, яку потрібно переслідувати — і уникнути ситуацій, коли інші розробники витрачають час на розбір коду.

Для подальшого покращення узгодженості команда Angular розробила інструмент CLI, який можна використовувати для створення конкретних повторюваних блоків коду з командного рядка. Крім того, документацію фреймворку можна використовувати як актуальну інформацію для розробників.

Досягнення високого ступеня узгодженості означає, що новим розробникам легше вибрати і зрозуміти структуру програми, і легше підтримувати величезну базу коду, спільно знижуючи витрати та підвищуючи ефективність.

4. Продуктивність

Ідеальним побічним ефектом простого синтаксису є підвищення продуктивності. Розробникам не потрібно витрачати дорогоцінний час на з'ясування функцій, утиліт або компонентів. Коли ви знаєте, як писати компоненти, ви зможете легко писати все інше, дотримуючись тих самих загальних вказівок і структури коду.

Висока читабельність коду також дозволяє новим розробникам легко вводити проекти, які розроблялися протягом тривалого часу.

Система типів дозволяє розробникам раніше виявляти потенційні помилки. Завдяки інтеграції з IDE, такими як код VS і WebStorm, TypeScript поступово перекомпілював і виділяв помилки запуску. Кожен з цих факторів допомагає скоротити час і витрати на розробку.

5. Технічне обслуговування

Коли ви переходите з однієї основної версії на іншу, всі пакунки, пов'язані з певною версією Angular, будуть оновлені одночасно, що означає, що HTTP-модулі, маршрутизація та матеріали Angular також будуть оновлені до відповідної версії, і конфлікти версій будуть відбуватися уникати.

Для завершення оновлення використовуйте лише команду "ng update". Це означає, що вам не доведеться витрачати час на підтримку нових пакетів, вирішення нових проблем із використанням версії або питання, коли оновити пакети з відкритим кодом до останньої версії. Крім того, компоненти можна легко розібрати та замінити для покращеної реалізації.

6. Angular Material

Angular Material — це набір згрупованих компонентів та модулів інтерфейсу користувача, в основі Angular Meterial лежить Google Design Material. Він містить багато компонентів інтерфейсу користувача, таких як навігаційні карти, елементи керування, кнопки та індикатори. Компоненти підходять для різних веб-браузерів, задокументовані та детально написані відповідно до останніх інструкцій.

Ці модулі розроблені спільнотою Angular і спрощують робочий процес команди, дозволяючи розробникам додавати нові елементи та швидко розробляти програми з мінімальним впливом на продуктивність.

Висновок Angular полягає в тому, що він дуже зручний і має багато функцій, які дозволяють розробникам досягати високої швидкості на ринку.

Його екосистема дуже велика, з великою кількістю готових компонентів, що може спростити розширення проекту та зробити його відмінним вибором для корпоративних проектів. Крім того, TypeScript скорочує час налагодження та виявляє найбільш поширені помилки під час розробки.

Вибір правильного інструменту для проекту має вирішальне значення. Хоча вибір веб-фреймворків великий, легко зрозуміти, чому так багато розробників і компаній обирають Angular.

7. Модульна структура розвитку

Модульна природа Angular означає, що розробники можуть ефективно розбивати код на модулі. Це дозволяє вам легко організувати функціональність вашої програми та створювати блоки коду для багаторазового використання, тим самим значно скорочуючи час і витрати на розробку. Спільнота Angular підтримує це за допомогою різних готових компонентів.

Модулі також дозволяють ефективно розподіляти роботу з розробки на основі команд, гарантують, що код залишається чистим і впорядкованим, і дозволяють безперешкодно розширювати програму.

Висновки до розділу

Так як мова Java є строготипізованою та ООП орієнтованою, та з хорошою реалізаціює багатопоточності це дозволяє нам писати серверні застосунки які добре справляються із великим навантаженням запитів.

Фреймворк Angular в основі якого лежить мова програмування TypeScript також строготипізована і дозволяє писати великі веб-застосунки для клієнтської частини.

Як результат ми отримуваємо два інструменти із схожим синтаксисом та однаковим ООП підходом, що в свою чергу дозволяє одній людині легко створювати застосунки

які легко масштабуються, легко підтримуються та дозволяють писати клієнтську та серверну частину без великих затрат.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1Формула пропускної здатності Клода Шеннона

Швидкість передачі може бути оцінена за формулою Клода Шеннона:

$$C = \Delta \mathrm{F} \log_2 (1 + \frac{S}{N})$$

, де B - швидкість передачі інформації, біт/с;

F – ширина смуги пропускання лінії Гц;

S – потужність сигналу;

N - Потужність шуму.

З цього співвідношення видно, що підвищити пропускну здатність можна за рахунок збільшення потужності передавача або зменшення потужності шуму. Близьким до формули Шеннона є відношення Найквіста, для визначення максимально можливої пропускної Можливості лінії зв'язку, але не враховуючи шуму лінії:

$$C = 2\mathrm{F} \log_2 M ,$$

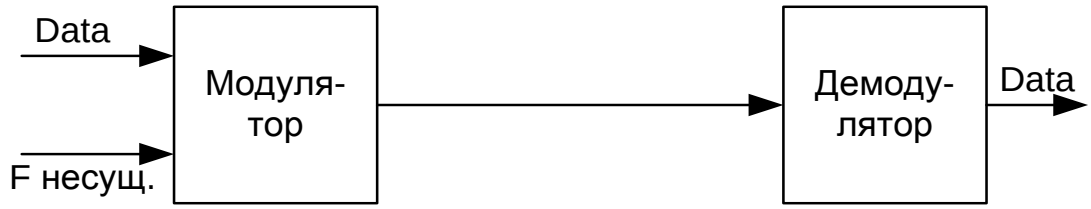
де M — кількість помітних станів інформаційного параметра.

Хоча формула Найквіста явно не враховує наявність шуму, його вплив позначається на виборі кількості станів інформаційного сигналу. Пропускна здатність лінії зв'язку залежить від внутрішніх параметрів: смуги пропускання; зовнішніх параметрів: рівня перешкод та способу кодування даних. Формула Шеннона визначає максимально можливу пропускну здатність лінії зв'язку при фіксованих значеннях смуги пропускання лінії та відношенні потужності сигналу до шуму. Формула Найквіста відображає максимально можливу пропускну здатність лінії зв'язку через смугу пропускання та кількість станів сигналу. Бодом називають швидкість перемикання сигналів. Швидкість передачі в загальному випадку може не збігатися з кількістю бод. Вона може бути як вище, так і нижче, це залежить від способу кодування.

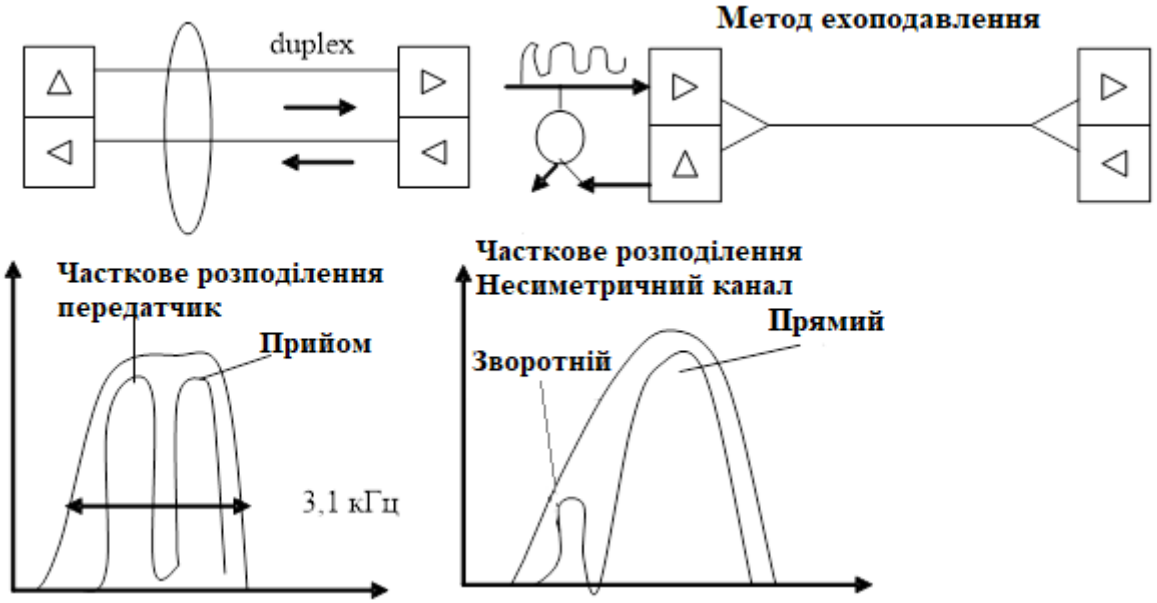
При передачі даних каналами зв'язку застосовуються два основних типи кодування:

1) на основі синусоїдального несучого сигналу. Цей спосіб називається аналоговою модуляцією, кодування здійснюється за рахунок зміни параметрів аналогового сигналу.

2) на основі послідовності прямокутних імпульсів називається цифрове кодування.



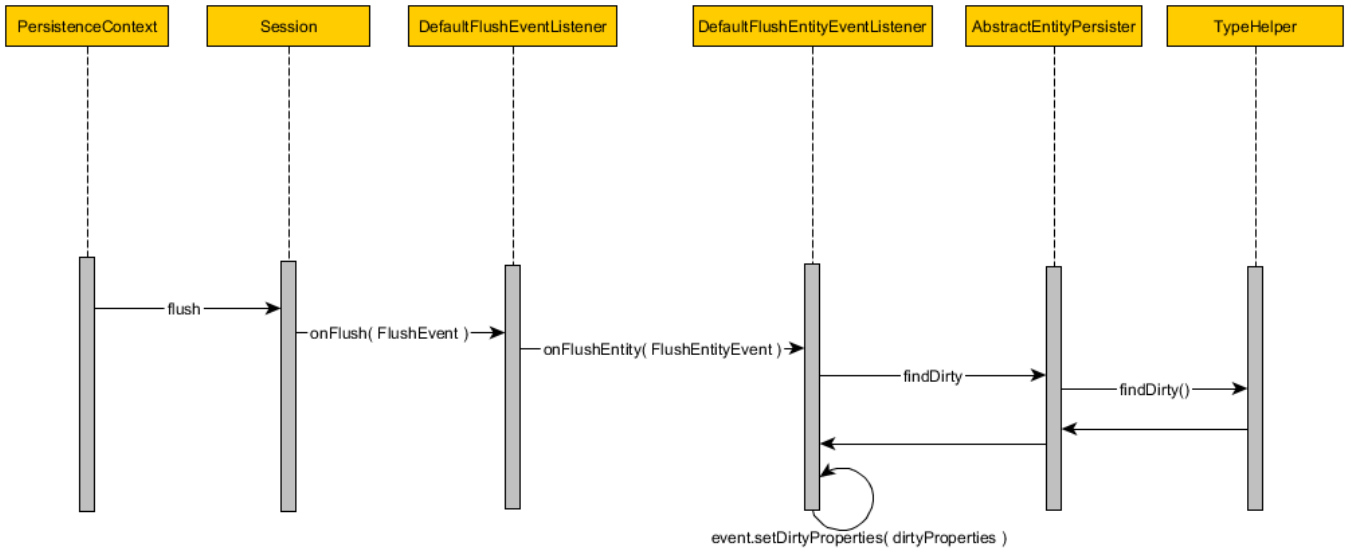
Зазвичай модулятор і демодулятор поєднуються в один пристрій: модем. Модем забезпечує дуплексний режим передачі.



Модеми для організації дуплексного режиму роботи на двопровідній лінії використовують техніку FDM. Модеми, що використовують частотну модуляцію, працюють на чотирьох смугах: 2 частоти - для кодування "1" і "0" в одному напрямку, і 2 інші для зворотного напрямку. При цифровому кодуванні дуплексний режим двопровідної лінії організується за допомогою техніки TDM.

3.2.Стратегія брудної перевірки за замовчуванням

За замовчуванням Hibernate перевіряє всі властивості керованого об’єкта. Щоразу, коли завантажується об’єкт, Hibernate створює додаткову копію всіх значень властивостей об’єкта. Під час очищення кожне властивість керованого об’єкта порівнюється зі значенням моментального знімка часу завантаження:



Отже, кількість окремих брудних перевірок визначається за такою формулою:

$$N = \sum_{k=1}^{n} p_k$$

де

n = кількість керованих об'єктів

p = кількість властивостей даної сутності

Навіть якщо тільки одна властивість однієї сутності коли-небудь змінилася, Hibernate все одно перевірить усі керовані об’єкти. Для великої кількості керованих об’єктів механізм перевірки бруду за замовчуванням може мати значне навантаження на ЦП і пам’ять. Оскільки початковий знімок сутності зберігається окремо, для контексту збереження потрібно вдвічі більше пам’яті, ніж зазвичай займають усі керовані об’єкти.

Висновки до розділу

В даному розділі було описано дві основні зони з якими стикаються будь-яка інформаційна система при розробці, а саме швидкість обробки інформацію та спілкування із базою даних. Було розглянуто принпиц формул для опрацювання пропускної здатності то роботи з базою данних за допомогою entity об’єктів при dirty check.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Мінімальне програмне забезпечення яке потрібне для запуску інформаційної системи:

1. Java 11
2. Gradle - Система автоматизованого збору проекту
3. Node.js: v12 або ж інші мінорні версії (12.1, 12.2)
4. Angular: 10
5. Spring Boot 2.4.3.RELEASE або пізніші версії
6. PostgreSQL: 12

4.1. Опис серверної частини

Серверна частина реалізована інструментов для backend розробки. Основною мовою розробки була вибрана мова програмування Java та інструмент для створення REST API веб-сервісів.

REST — **RE**presentational **S**tate **T**ransfer. В основі якого лежить архітектурний стиль розподілених гіпермедійних систем, який вперше представив Рой Філдінг у своїй знаменитій роботі в 2000 році. Це стиль взаємодії компонентів розподіленого додатка в мережі. REST — являє собою єдиний загальноузгоджений набір обмежень, які враховуються при розробці розподіленої веб-системи. У деяких випадках (інтернет-журнали, пошукові системи, провідні системи) це підвищить продуктивність і зручність використання архітектури. Точніше, компоненти в REST взаємодіють як клієнти та сервери у всесвітній мережі.

Принципи REST

1. Клієнтський сервер. Відокремлюючи проблеми інтерфейсу користувача від проблем зберігання, ми покращуємо переносимість інтерфейсу користувача на кількох платформах і покращуємо масштабованість, спрощуючи компоненти сервера.

2. Дані стану передаються за протоколом HTTP, тому нам не потрібно підтримувати постійний контакт з клієнтом під час сеансу.

3. Архітектура кешу - якщо інформація не змінюється після кількох запитів до сервера, її можна кешувати та повернути клієнту без повторного запиту сервера

4. Доступність представлення даних. Ви можете використовувати два методи форматування як представлення об'єктів: JSON і XML.

5. Єдиний доступний інтерфейс – при доступі до певного API під час фази запиту зрозуміло, яка приблизна відповідь. Наприклад, щоб отримати інформацію про конкретну книгу, нам потрібно перейти до-"/cars.com/car/1"

4 основних типи архітектури REST запиту HTTP:

- **GET** — для отримання (зчитування)
- **POST** — для створення
- **PUT** — для редагування (зміни)
- **DELETE** — для видалення

Основна веб-програма Spring Boot, яка реалізує REST API, складається з кількох основних частин:

1. Модельний шар. Буква «M» у шаблоні MVC. У цей шар ми включаємо класи, на основі цих класів сервер буде формувати і надсилати відповіді. По суті, це об’єкти, які отримає наш клієнт після обробки запиту

2. DAO або рівень сховища. Він складається з класів і методів, які працюють з базою даних. Кожен метод є атомарним і відповідає лише за одну операцію.

3. Сервісний шар. Тут ми маємо кілька класів, які повинні обробляти отриману інформацію. В основному метод обслуговування складається з кількох методів dao і простих або складних операцій над ними.

4. Контролерний шар. Ці класи є точками входу в наш REST API.

Клас складається з методів для обробки конкретних URL-адрес і визначених методів HTTP.

Рисунок 4.1.1 Структура проекту

Отже, приблизний шлях нашого запиту на запит виглядає наступним чином:

1. Ми отримуємо запит клієнта.

2. Клас контролера отримує запит.

3. Далі отриманий запит надходить до класу обслуговування для обробки та модифікації інформації.

4. Потім сформуйте відповідь на основі нашої моделі та передайте її клієнту.

Інформаційна система складається з декількох модулів:

• Config-містить класи для налаштування авторизації, реєстрації та налаштування розсилки електронної пошти

• Контролер це класи які написані по специфікації REST API і використовуються як точки входу до нашої серверної частини.

Нижче наведено приклад опису класу, який містить метод обробки методу http delete для видалення учасника.

```java
@CrossOrigin(origins = "*", maxAge = 3600)
@RestController

public class AbitController {

    @DeleteMapping("/del/{id}")
    public void deleteById(@PathVariable Long id){
        Abiturient one = abitRepos.getOne(id);
        one.setFaculties(null);
        abitRepos.delete(one);
    }
```

Рисунок 4.1.2 Клас-контроллер

• Dto-містить класи, які використовуються для переформатування одного класу в інший

• Entities-класи, на основі яких будуть створені таблиці в базі даних

```java
@Entity
@Table( name = "Abiturients")
public class Abiturient {


    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long idAbitCode;
    private String username;
    private String surname;
    private String poBatkovi;
    private Double avgDiplomaMark;

    @NotBlank
    @Size(max = 50)
    @Email
    private String email;

    @NotBlank
    @Size(max = 30)
    private String password;

    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable( name = "user_roles",
            joinColumns = @JoinColumn(name = "user_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id"))
    private Set<Role> roles = new HashSet<>();
```

Рисунок 4.1.3 Entity Абітурієнти

• На основі моделі на основі класу, який буде формувати об'єкт відповіді сервера

• Payloads включає два пакети. Перший — це «запит», який включає клас для обробки запиту і пакет «відповідь», який надсилає об’єкт відповіді після обробки запиту. Нижче наведено клас «LoginRequest», який містить два поля електронну пошту та пароль, на основі якого буде оброблено запит на вході абітурієнта.

```
public class LoginRequest {
    @NotBlank
    private String email;

    @NotBlank
    private String password;
```

Рисунок 4.1.4 Клас LoginRequest

• Репозиторій - складається з класів зі стандартними CRUD (створювати, читати, оновлювати, видаляти) та іншими атомарними методами.

• Безпека, включаючи класи конфігурації авторизації та реєстрації на основі маркерів JWT.

• Сервіс – містить всю логіку, яка використовується для обробки запиту та класу

• Swagger-документуйте наш API

Процес реєстрації та авторизації заснований на токенах JWT. Це означає, що нам не потрібно використовувати сеанси для підтримки з’єднання з сервером.

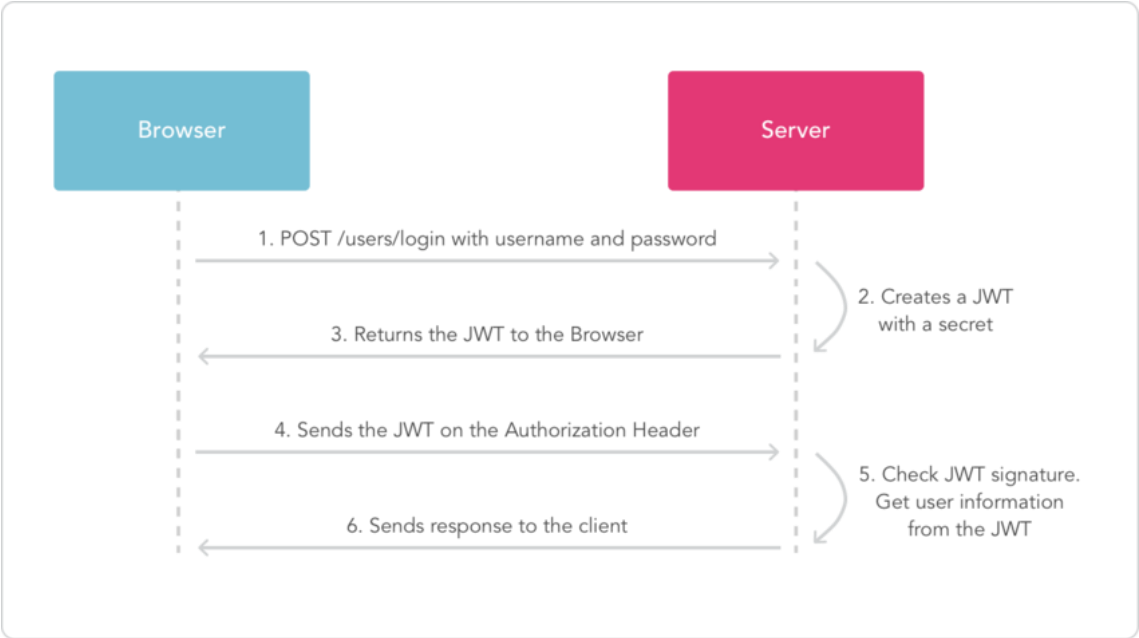


Рисунок 4.1.5 Генерація JWT токена

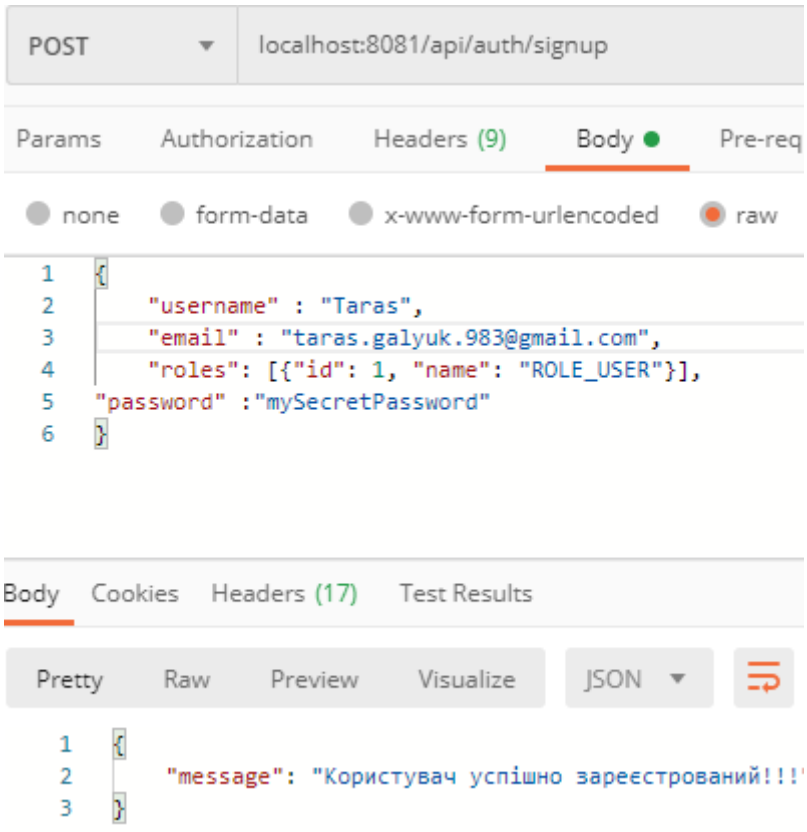


Рисунок 4.1.6 POST запит на реєстрації

Щоб зареєструвати учасника конкурсу на клієнта, необхідно сформувати орган запиту, як показано на малюнку вище. Далі є відповідь на повідомлення «користувач успішно зареєстрований».

Наступний крок – процес авторизації.

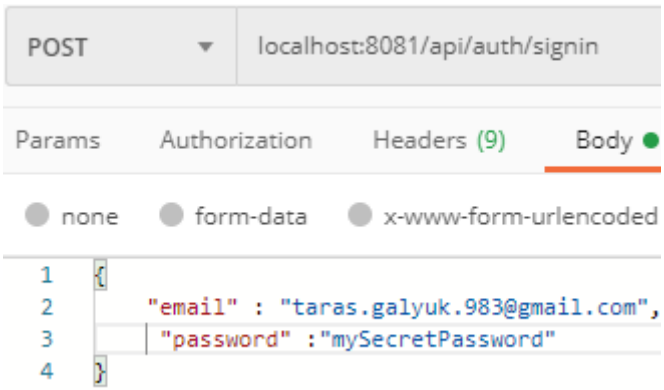


Рисунок 4.1.7 POST запит на авторизацію

У відповідь від сервера ми отримуємо згенерований маркер JWT, який буде додано до заголовків усіх наступних запитів для авторизованих користувачів.


```
{
    "id": 14,
    "username": "Taras",
    "surname": null,
    "email": "taras.galyuk.983@gmail.com",
    "roles": [
        "ROLE_USER"
    ],
    "accessToken":
"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0YXJhcy5nYWx5dWsuOTgzQGdtYWlsLmNvbSIsImlhdCI6MTU5M
jEyOTIxNiwiZXhwIjoxNTkyMjE1NjE2fQ.xBo3iJkhxowUErdZvgznmyxkBJPS1jrSNZR2fxYwGLT5sY8Sm
SX3gdTBDzuYLN4O09bDO10mVALNPkc9SO6Wwg",
    "tokenType": "Bearer"
}
```


Нижче наведено метод класу для обробки запитів POST для авторизації та створення маркерів JWT. Як параметр цього методу він використовує тіло запиту на основі класу LoginRequest, який складається з двох полів.

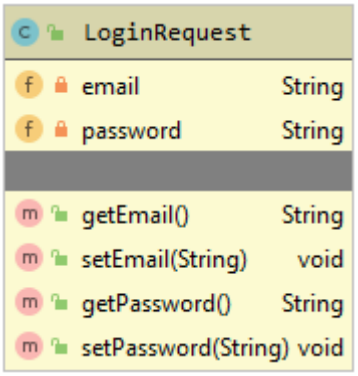


Рисунок 4.1.8 Клас "LoginRequest"

```
@PostMapping("/signin")
public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequest)
{

    Authentication authentication = authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(loginRequest.getEmail(),
loginRequest.getPassword()));

    SecurityContextHolder.getContext().setAuthentication(authentication);
    String jwt = jwtUtils.generateJwtToken(authentication);

    System.out.println(jwt);

    AbitDetailsImpl userDetails = (AbitDetailsImpl) authentication.getPrincipal();
    List<String> roles = userDetails.getAuthorities()
            .stream()
            .map(GrantedAuthority::getAuthority)
            .collect(Collectors.toList());

    return ResponseEntity.ok(new JwtResponse(jwt,
            userDetails.getId(),
            userDetails.getUsername(),
            userDetails.getSurname(),
            userDetails.getEmail(),
            roles
    ));
}
```

Після авторизації заявник зобов’язаний заповнити всю інформацію, необхідну для подальшої роботи.

На основі закінчених предметів, які кандидат склав ЗНО, йому будуть показані всі наявні спеціальності, на які він може претендувати.

```
    @GetMapping("/allFa/{id}")
    List<Faculty> faculties2(@PathVariable Long id) {

        List<Faculty> canPass = new ArrayList<>();

        List<Faculty> all = facultyRepo.findAll();
        Abiturient one = this.abitRepos.getOne(id);
        Set<ZNOOneSubject> subjs = one.getSubjs();
        List<Subject> collect =
    subjs.stream().map(ZNOOneSubject::getSubject).collect(Collectors.toList());

        for (Faculty faculty : all) {
            Set<Specialization>   emtpySpecialization = new HashSet<>();
            Set<Specialization> specializations = faculty.getSpecializations();
            ArrayList<Specialization> specs = new ArrayList<>(specializations);

            Faculty faculty1 = new Faculty();
            faculty1.setFacultyIdl(faculty.getFacultyIdl());
            faculty1.setFacultyName(faculty.getFacultyName());

            for (Specialization spec : specs) {
                if (spec.getNeedSubjects().containsAll(collect) ) {
                    emtpySpecialization.add(spec);
                }
            }

            faculty1.setSpecializations(emtpySpecialization);
            if(faculty1.getSpecializations().size()>0) {
                canPass.add(faculty1);
            }
        }
        return canPass;
    }
```

Інструмент Swagger використовується для документування всього REST API. Використовуючи його, ви можете зручно і легко відображати всі HTTP-запити, оброблені нашим веб-сервером.

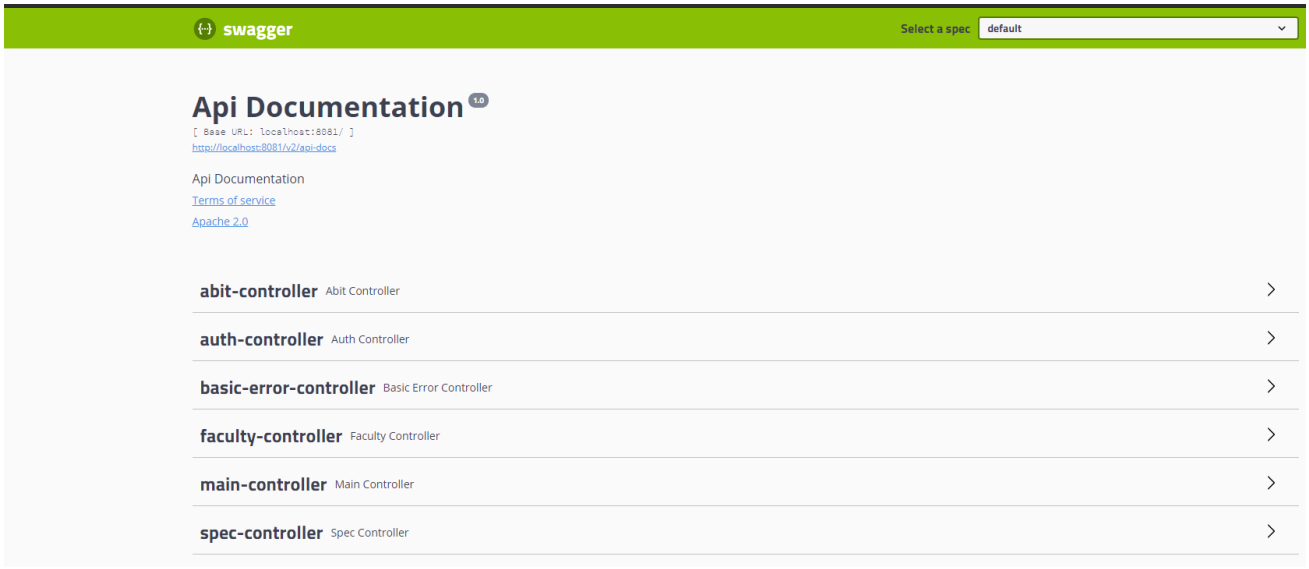


Рисунок 4.1.9 Список контроллерів

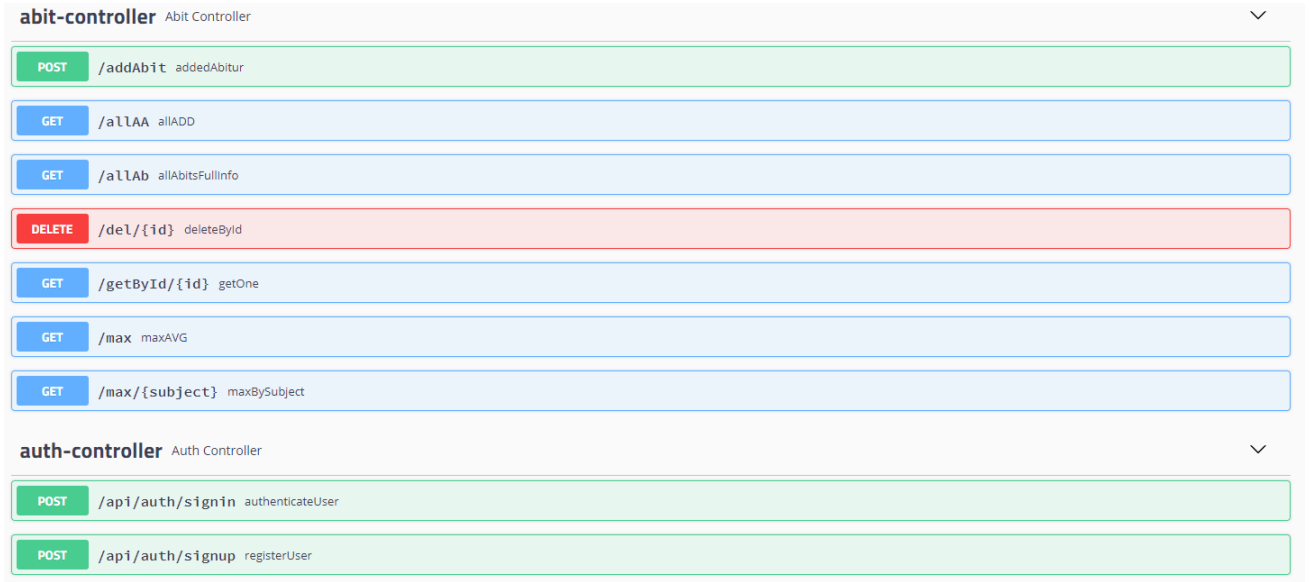


Рисунок 4.1.10 HTTP методи класів

4.2. Проектування бази даних

При розробці бази даних я використовував базу даних Postgres. Postgres — це безкоштовна система управління реляційними базами даних з відкритим вихідним кодом.

Він характеризується стабільністю, простотою використання та високою швидкістю. Він був розроблений для підвищення швидкості обробки великих баз даних. Основні переваги сервера добре продемонстровані в системі UNIX, яка підтримує багатопотоковість, тим самим покращуючи продуктивність системи.

Основна частина бази даних складається з кількох таблиць:

1.Абітурієнти

Таблиця складається з колонок:

- Ід код абітурієнта
- Прізвище, ім’я, по-батькові
- Email
- Середній бал атестату
- Кількість заявок для вступу

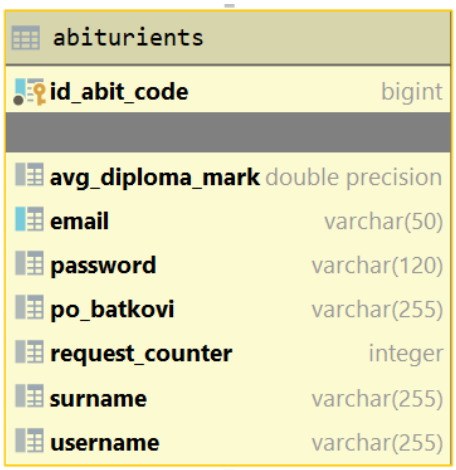


Рисунок 4.2.1 Абітурієнти

2.Факультети:

- Назва факультету
- Опис

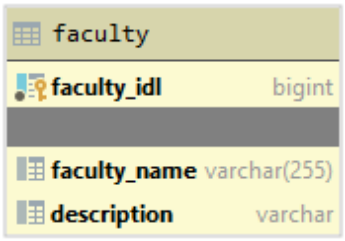


Рисунок 4.2.2 Факультети

3.Спеціальності:

- Код спеціальності
- Назва спеціальності

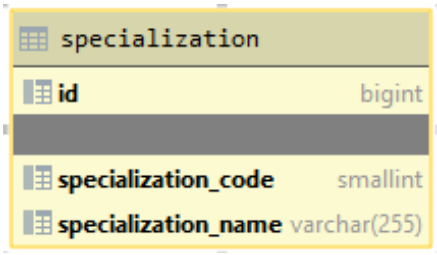


Рисунок 4.2.3 Факультети

4.Запитання вступників:

- Електрона пошта абітурієнта
- Запитання

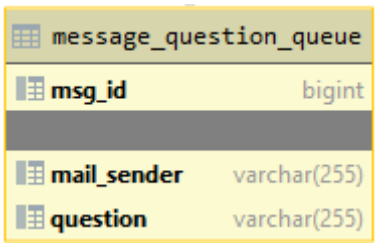


Рисунок 4.2.4 Запитання вступників

5. Ролі

- Ід
- Назва ролі (Користувач, модератор, адміністратор)

Рисунок 4.2.5 Ролі

6. Предмети ЗНО:

- Первинний ключ на ід абітурієнта
- Назва предмету
- Кількість балів з певного предмету

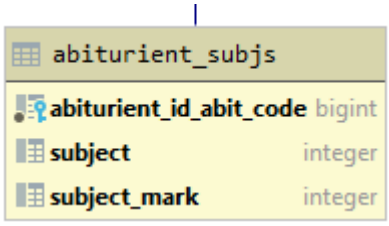


Рисунок 4.2.6 Предмети для абітурієнтів

Схеми БД:

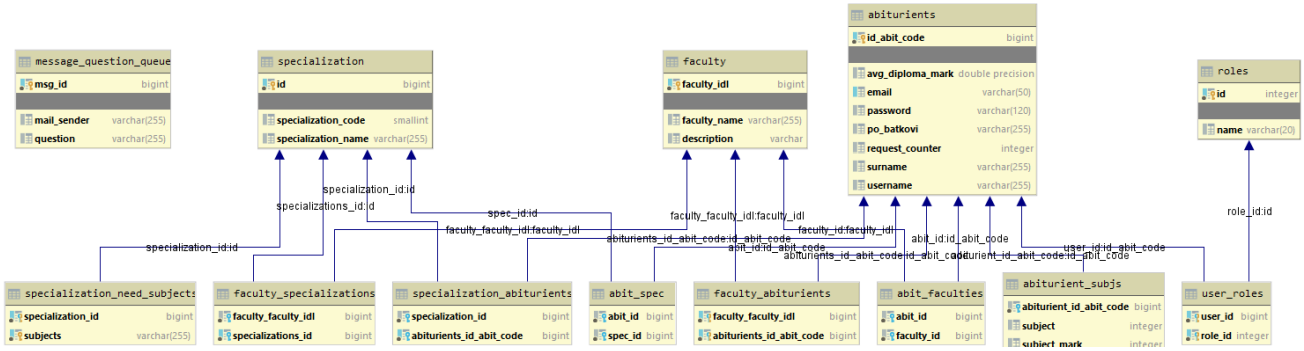


Рисунок 4.2.7 Предмети абітурієнта

Міграція бази даних – у контексті корпоративних програм – означає переміщення ваших даних з однієї платформи на іншу. Існує багато причин, чому ви можете перейти на іншу платформу. Наприклад, компанія може вирішити заощадити гроші, перейшовши на хмарну базу даних. Або компанія може виявити, що певне програмне забезпечення баз даних має функції, які є критичними для її бізнес-потреб. Або система просто застаріла. Процес міграції бази даних може включати кілька етапів та ітерацій, включаючи оцінку поточної бази даних і майбутніх потреб у розробці, міграцію схеми, а також нормалізацію та переміщення даних. Плюс тестування і так далі.

Натомість ці інструменти намагаються зберегти семантику даних або реорганізувати наявні дані, щоб вони відповідали новим вимогам. І оскільки семантика даних зазвичай формально не зберігається, інструменти налаштування зазвичай ручні.

Коли серверна частина запускається вперше за допомогою налаштованого середовища, буде виконано процес запуску сценарію sql для заповнення бази даних.

При подальшому запуску або перезапуску сервера бібліотека спостерігатиме правильну перевірку схеми бази даних

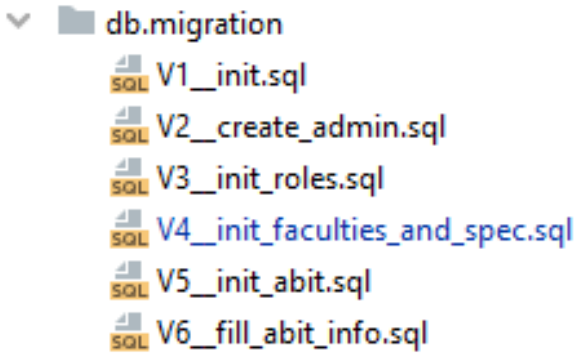


Рисунок 4.2.8 Sql міграції

4.3. Послідовний опис розробки інтерфейсу застосунку

При вході на сайт користувач перенаправляється на головну сторінку на головну сторінку. У заголовку відображається два пункти: «Реєстрація» та «Вхід».

На цій сторінці також відображається статистика заявок, кількість заявників та загальна кількість заявників.

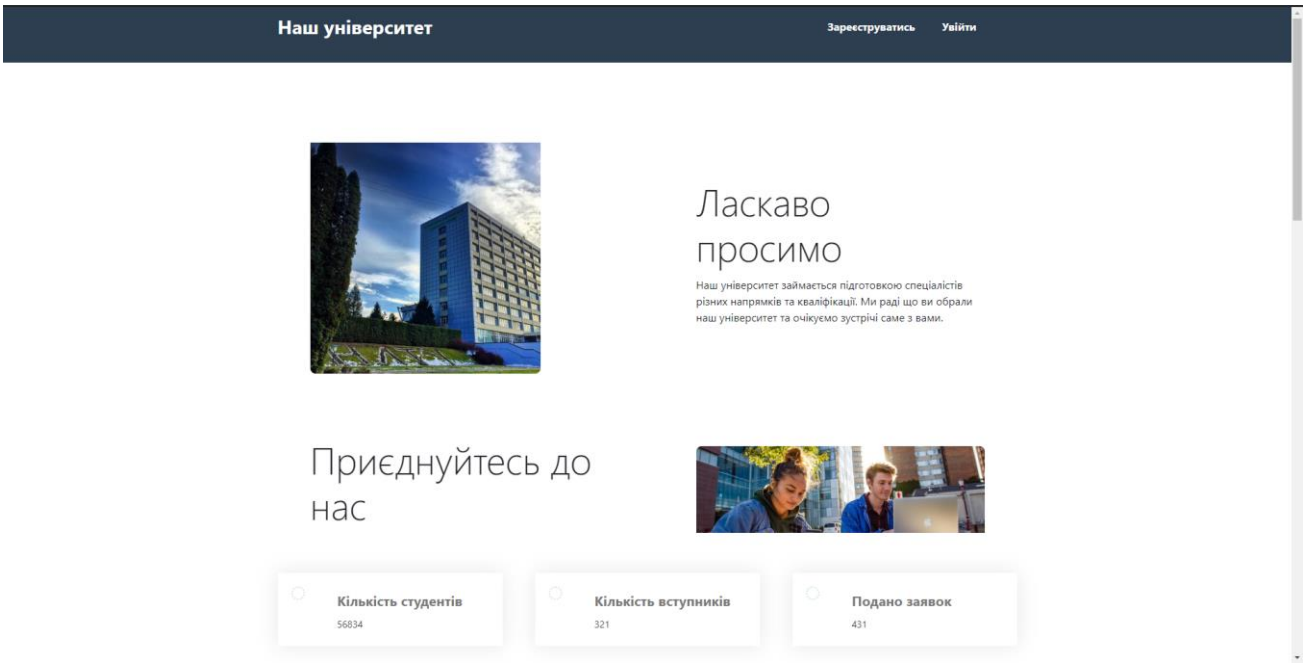


Рисунок 4.3.1 Головна сторінка

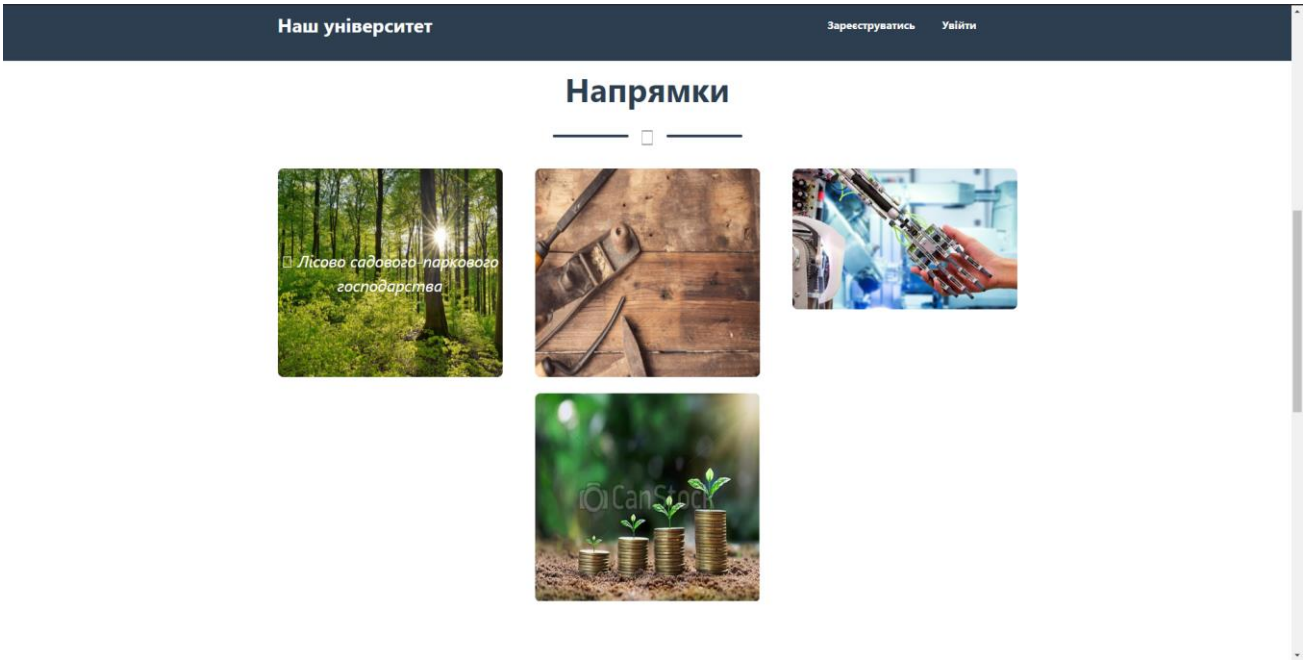


Рисунок 4.3.2 Головна сторінка

Прокручуючи сторінку нижче, користувач побачить список областей, підготовлених учнями. Коли ви наведете курсор миші на один із напрямків на зображенні, з’явиться назва цього напрямку. Клацнувши по одному з напрямків відкриється вікно з детальною інформацією про предмет.

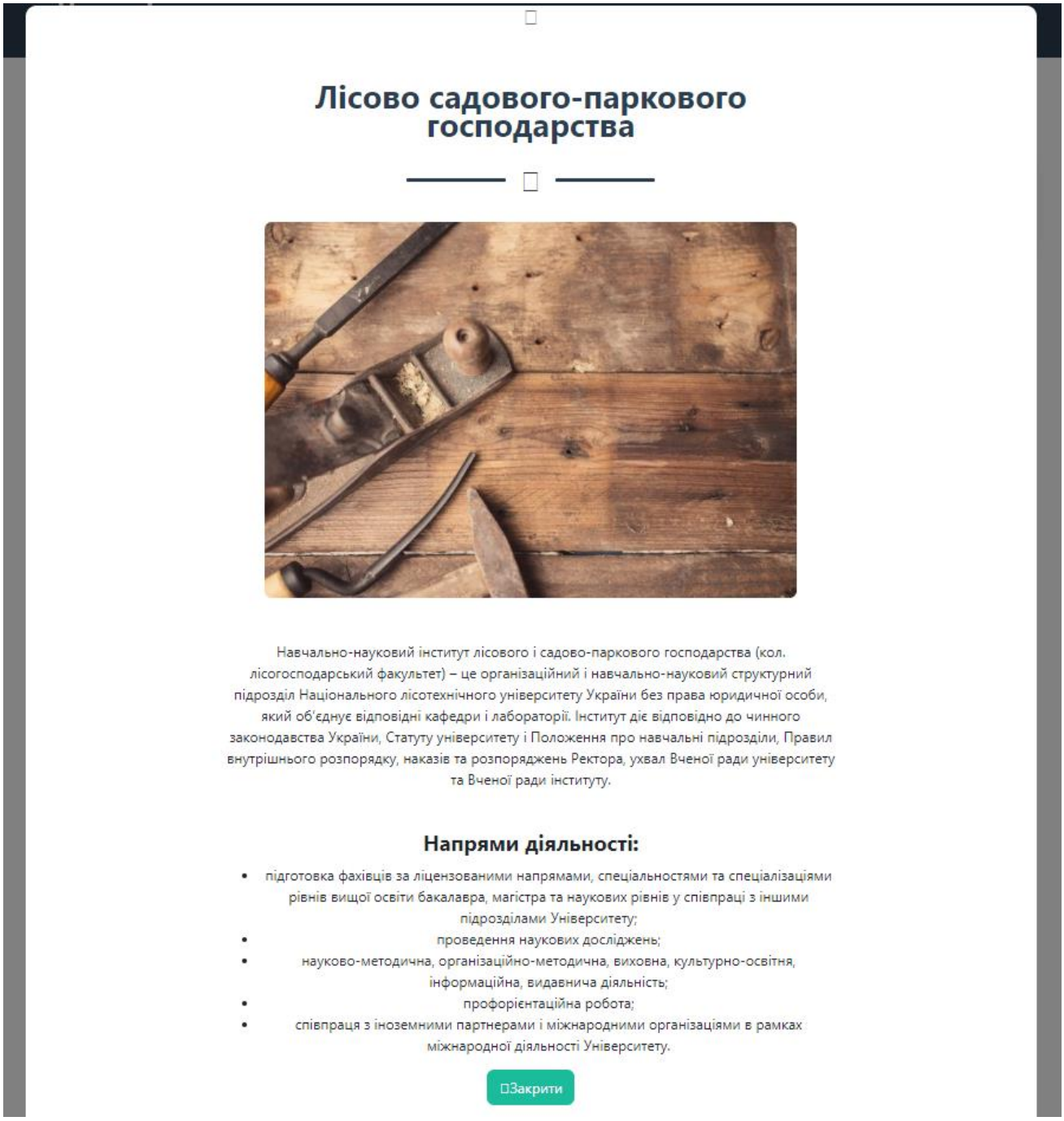


Рисунок 4.3.3 Опис напрямку

Після ознайомлення конкурсантів з професією в наступних розділах будуть описані «Про нас», «Контактна інформація» та «Нижній колонтитул».

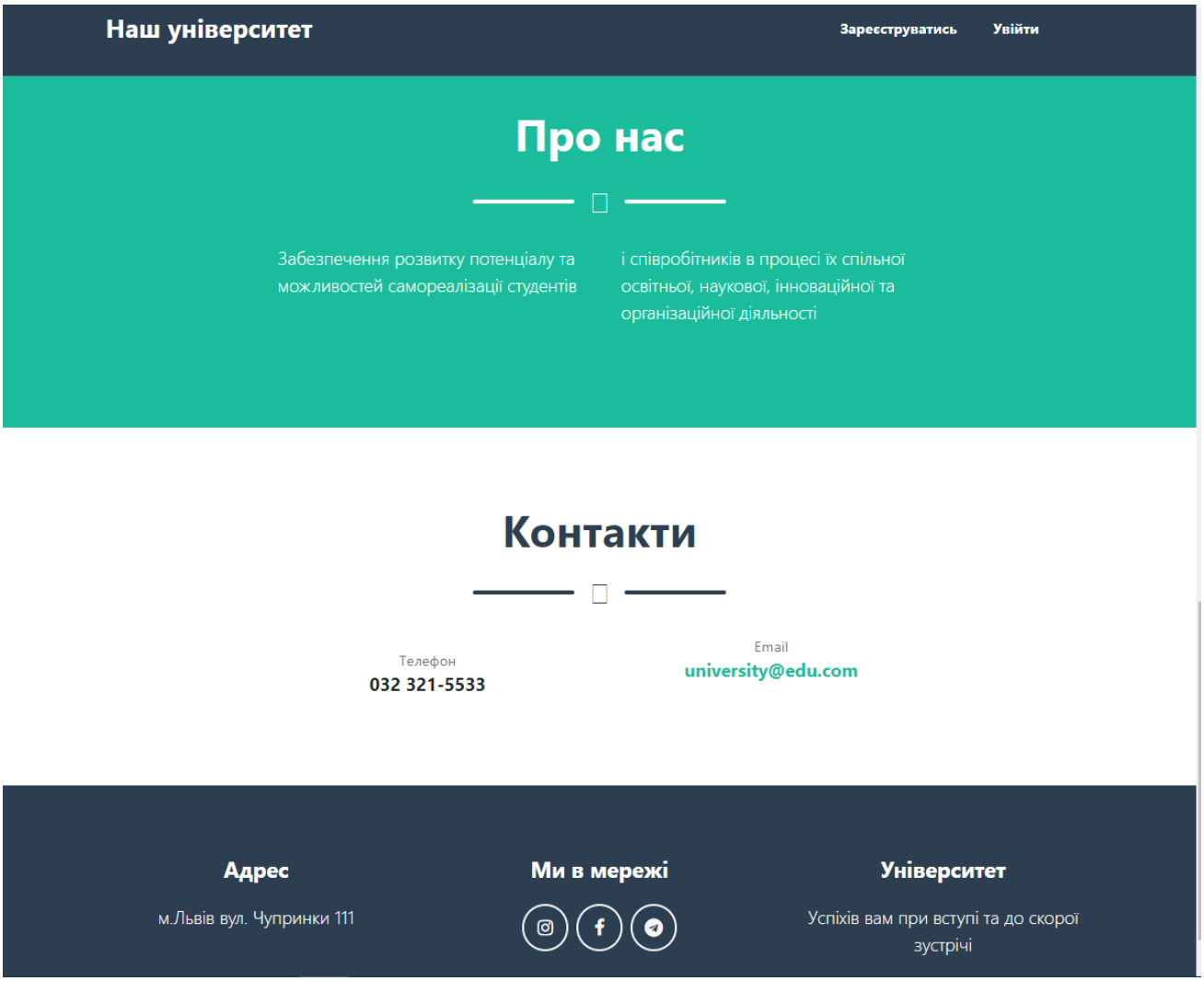


Рисунок 4.3.4 Кінець головної сторінки

Наступний крок – реєстрація конкурсантів. Сторінка складається з форми з кількома полями. Прізвище, ім’я, ім’я батька, електронна пошта, номер мобільного телефону та пароль.

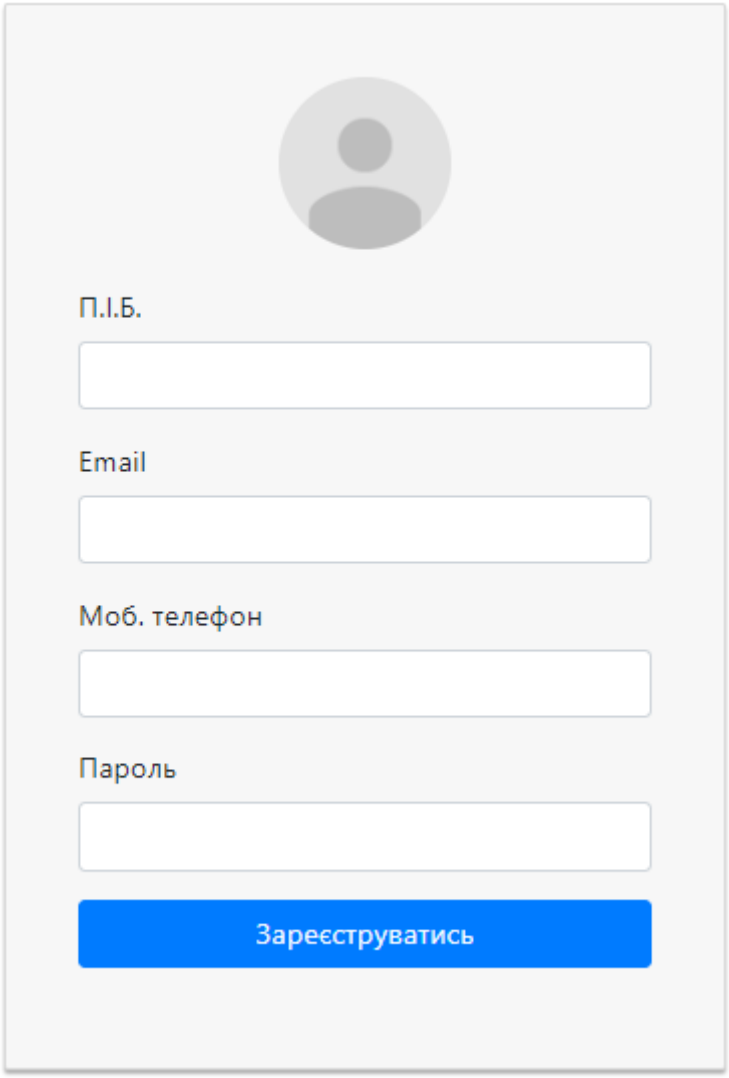


Рисунок 4.3.5 Форма реєстрації

Після заповнення основної інформації користувач отримає повідомлення про успішну реєстрацію, після чого автоматично відкриється вікно входу.

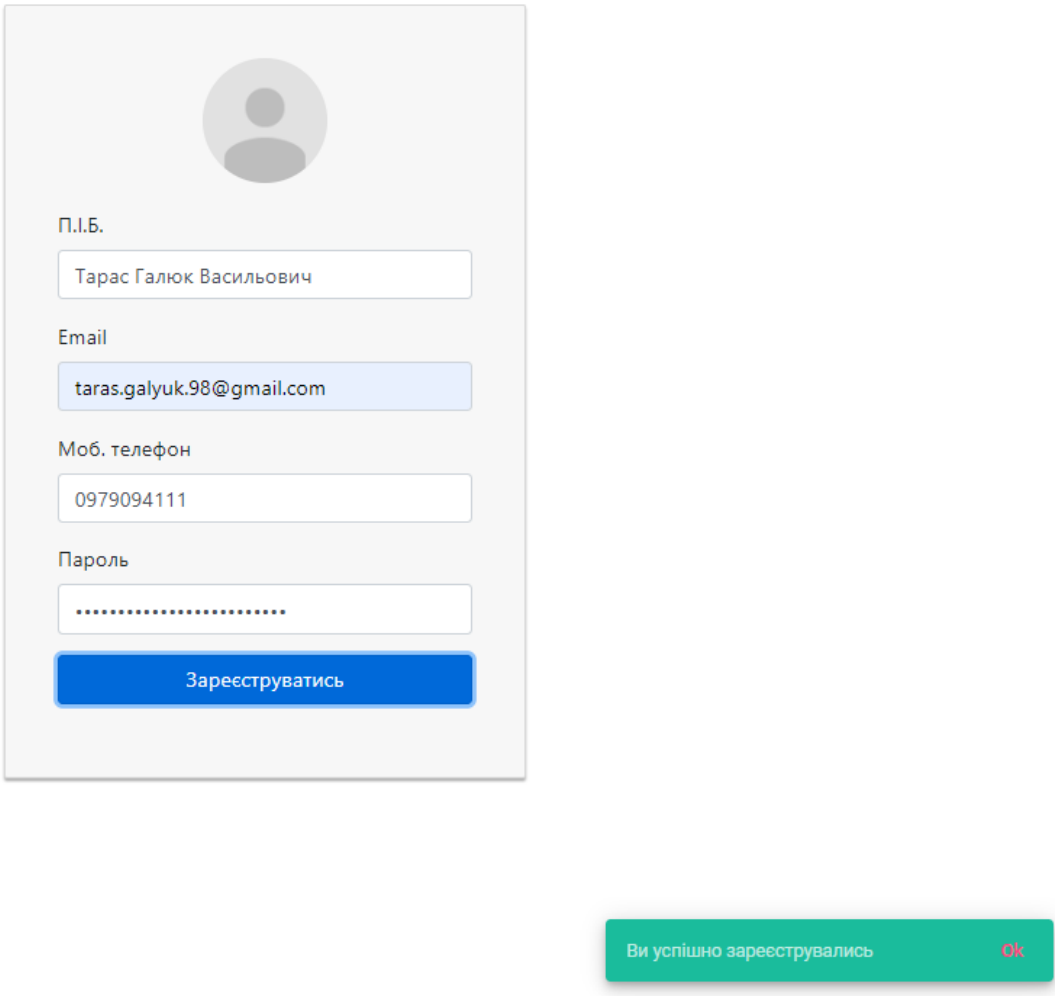


Рисунок 4.3.6 Завершення реєстрації

Після закінчення реєстрації вступника отримує листа на пошту на пошту із вітанням.

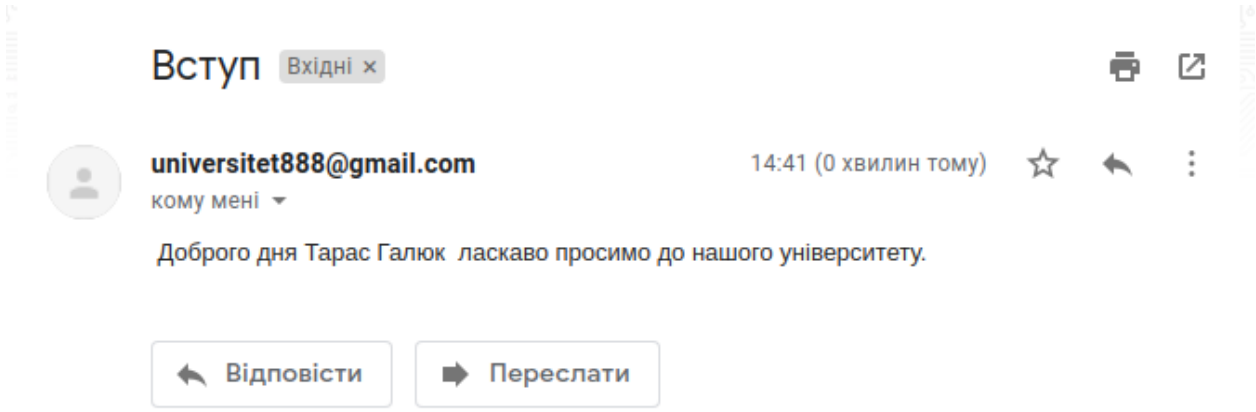


Рисунок 4.3.7 Електронний лист після реєстрації

Після входу в систему учасник може використовувати інші два компоненти на домашній сторінці, які недоступні для неавторизованих користувачів. Перший компонент - це "кроки", а другий компонент - "Довідка бот".

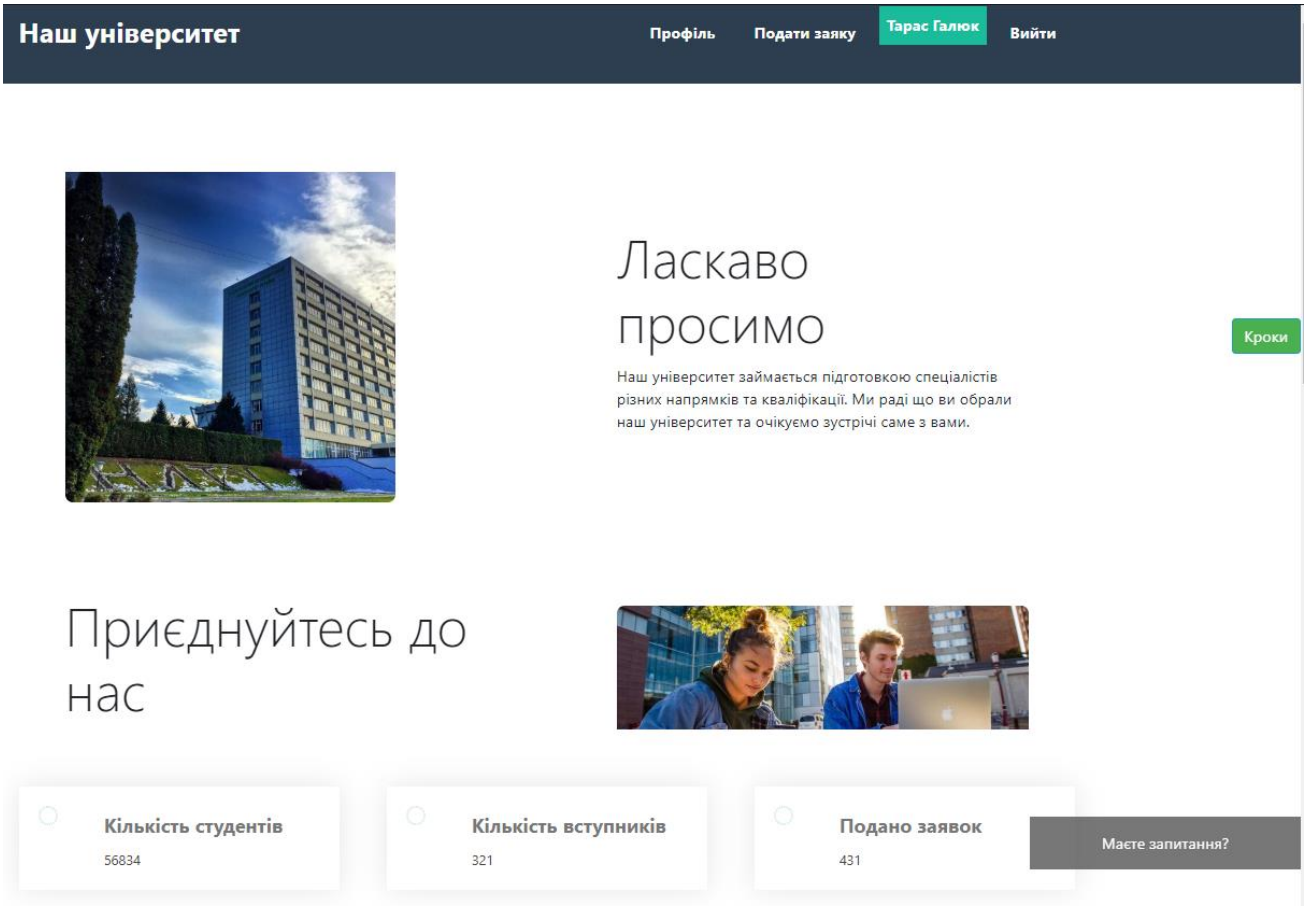


Рисунок 4.3.8 Сторінка авторизованого користувача

«Є питання?» — це компонент робо-порадника. Компонент розташований у нижньому правому куті екрана і має фіксоване положення відносно екрана

Якщо у користувача є проблеми, він може перевірити поширені запитання. Якщо користувач не знайдений у списку питань, які його цікавлять, він може задати своє запитання.

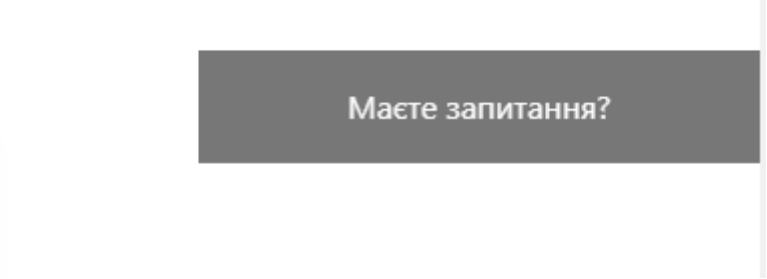


Рисунок 4.3.9 Відображення на сторінці

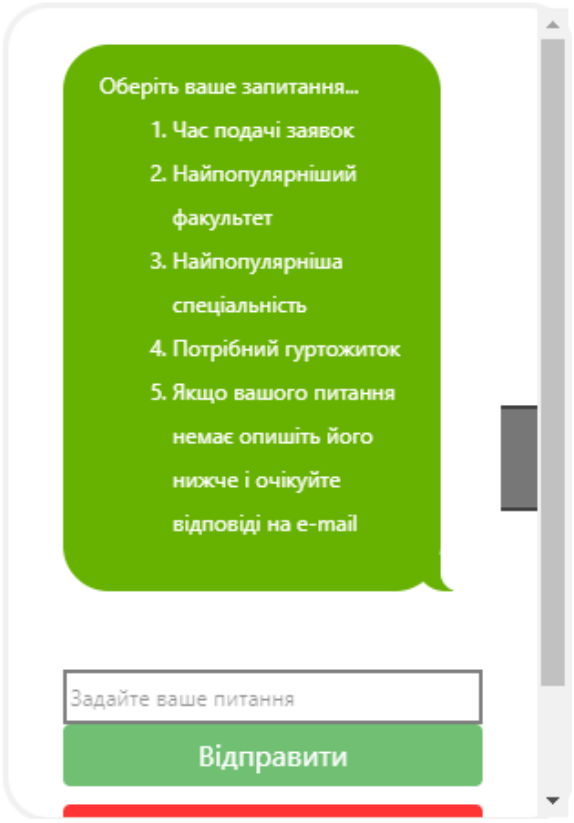


Рисунок 4.3.10 Зовнішній вигляд бота

Бот отримує всю інформацію від сервера. Комунікація заснована на архітектурі REST. Робот надсилає HTTP-запит, а сервер обробляє отриману інформацію і повертає відповідь на запит користувача.

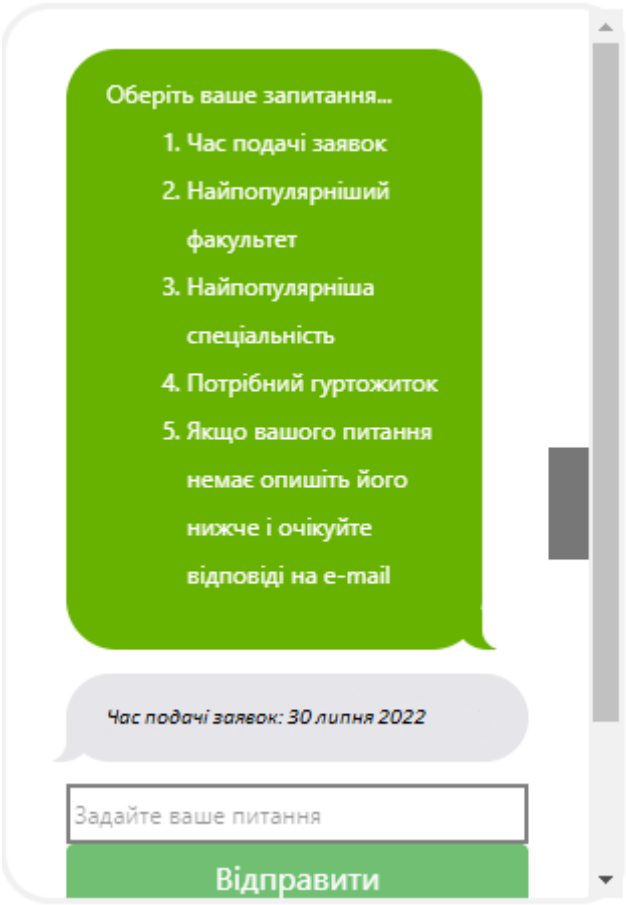


Рисунок 4.3.11 Приклад виконання запиту

Якщо користувач не знайшов запитання, яке його цікавить, він може написати його сам, питання буде відправлено в чергу повідомлень, а відповідь буде надіслана адміністратором та надіслана на електронну пошту, зареєстровану користувачем.

Після того, як користувач запише запитання, на екрані відкриється вікно з інформацією про надісланий запит.

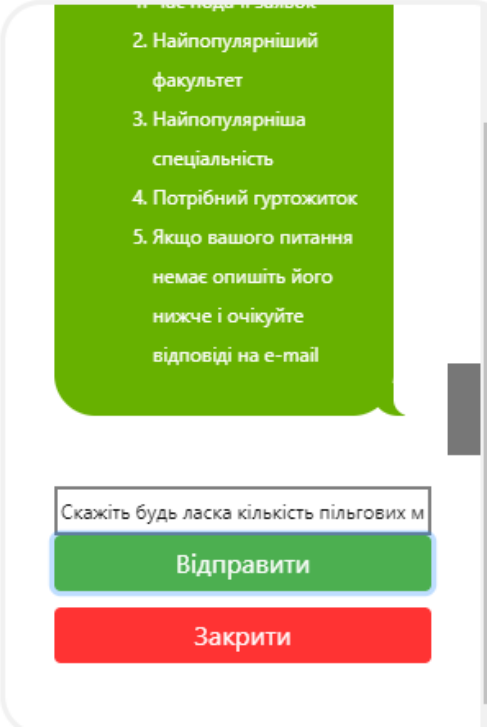


Рисунок 4.3.12 Приклад запитання

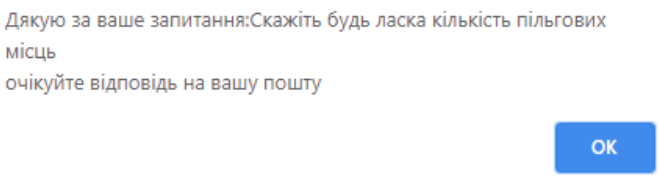


Рисунок 4.3.13 Результат опрацювання запиту

Після обробки запиту питання відправляється в базу даних і питання піднімається в таблиці.

Адміністратори та модератори мають можливість відповісти на запитання. Після того, як адміністратор відповість на запитання, користувач отримає відповідь на електронну пошту.

«Кроки» — це поради, які слід застосовувати крок за кроком. Після першого кроку користувач може натиснути на пункт №2, після чого відкриється нова форма, яку необхідно заповнити.

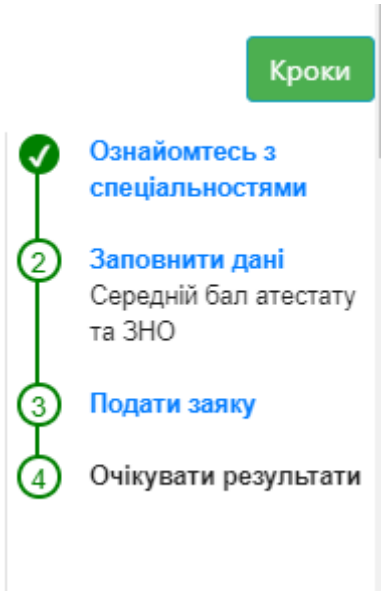


Рисунок 4.3.14 Відкрита компонента

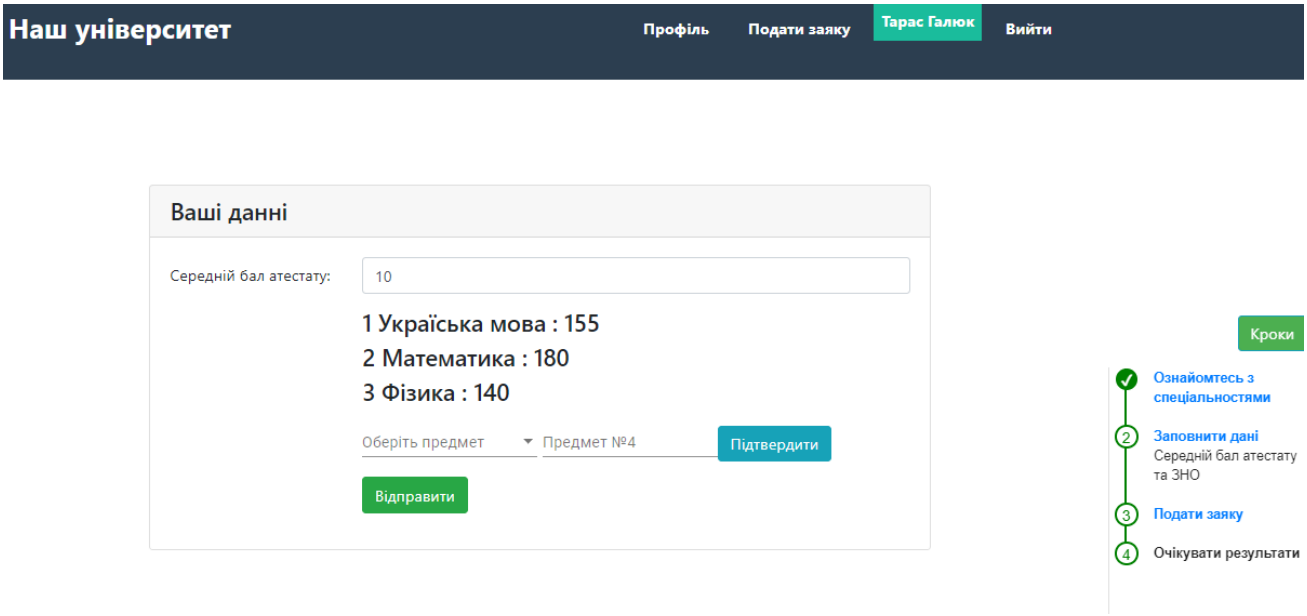


Рисунок 4.3.15 Крок №2

Оскільки наразі є можливість подати заявку на три предмети зовнішнього оцінювання, користувачі мають можливість заповнити власну інформацію на основі трьох-чотирьох предметів.

На наступному етапі після заповнення форми за темою ЗНО відкриється форма зі списком коледжів та спеціальностей, на які можна подати документи.

У користувача є 5 доступних заявок.

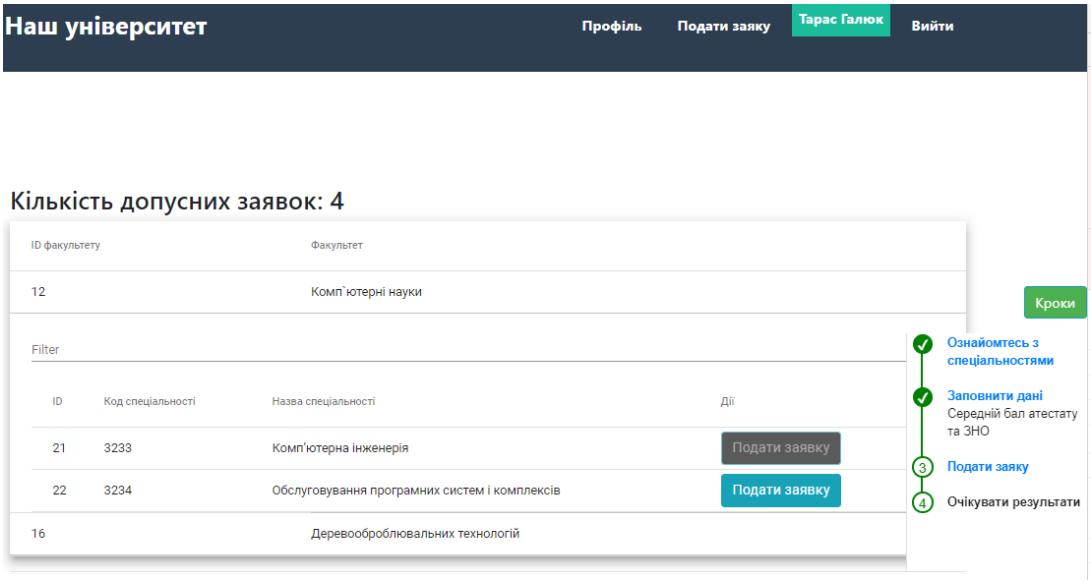


Рисунок 4.3.16 Подача заявки

Після того, як користувач використає всі доступні спроби, користувач не зможе отримати доступ до таблиці але на екрані появиться навий запис із подальшими кроками .

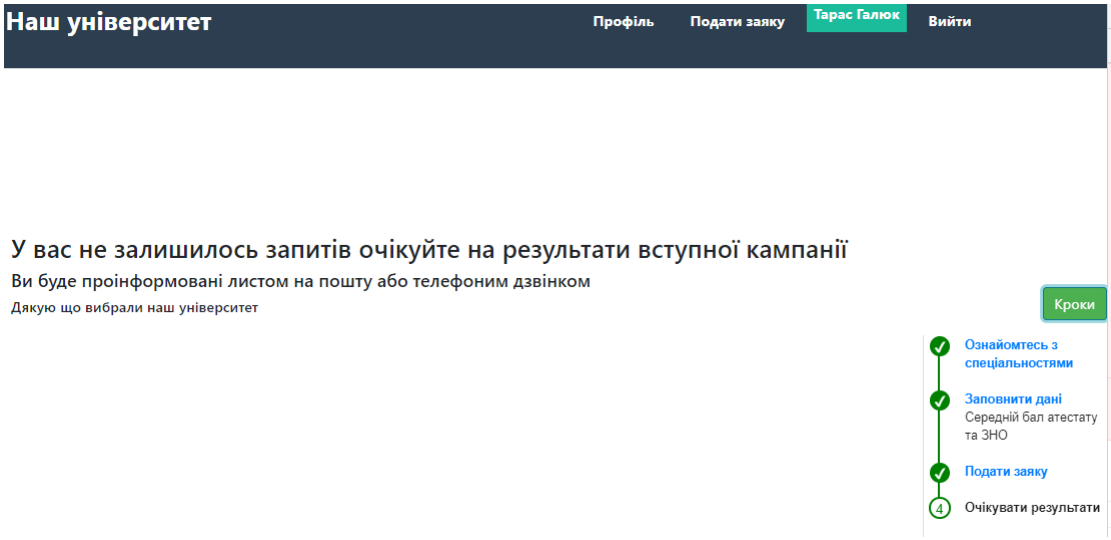


Рисунок 4.3.17 Закінчення подачі заявок

4.4. Тестування проекту

Під час реєстрації форма налаштовується таким чином, що користувач повинен пройти певну перевірку для забезпечення безпеки та правильної обробки даних.

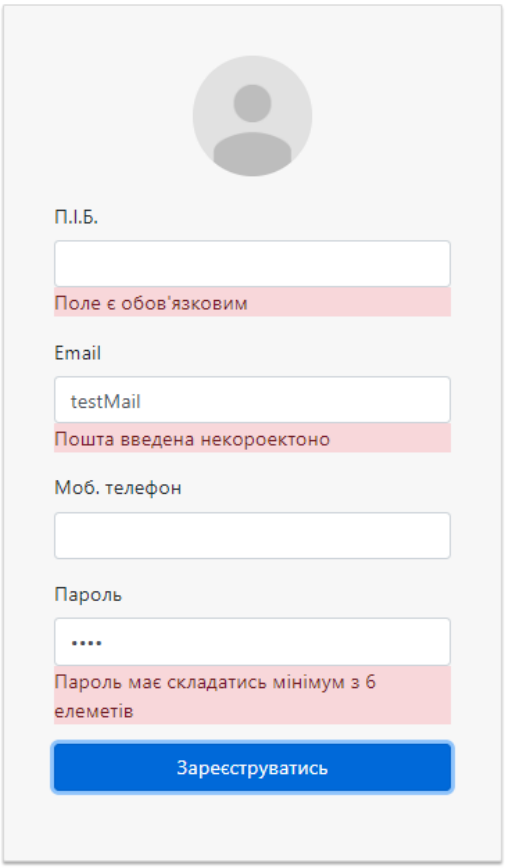


Рисунок 4.4.1 Валідація форми реєстрації

Якщо користувач захоче зареєструватись ще раз то у відповідь він отримає повідомлення що така пошта уже зареєстрована.

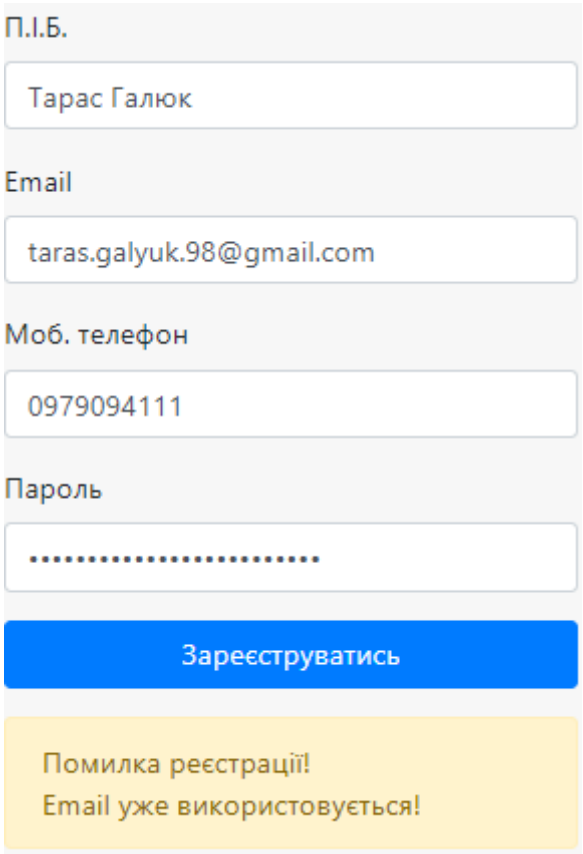


Рисунок 4.4.2 «Помилка реєстрації»

Заповніть форму на предмети зовнішнього оцінювання та ключову інформацію. Якщо поле не заповнене, воно буде пофарбовано в червоний колір, що означає, що воно не заповнене.

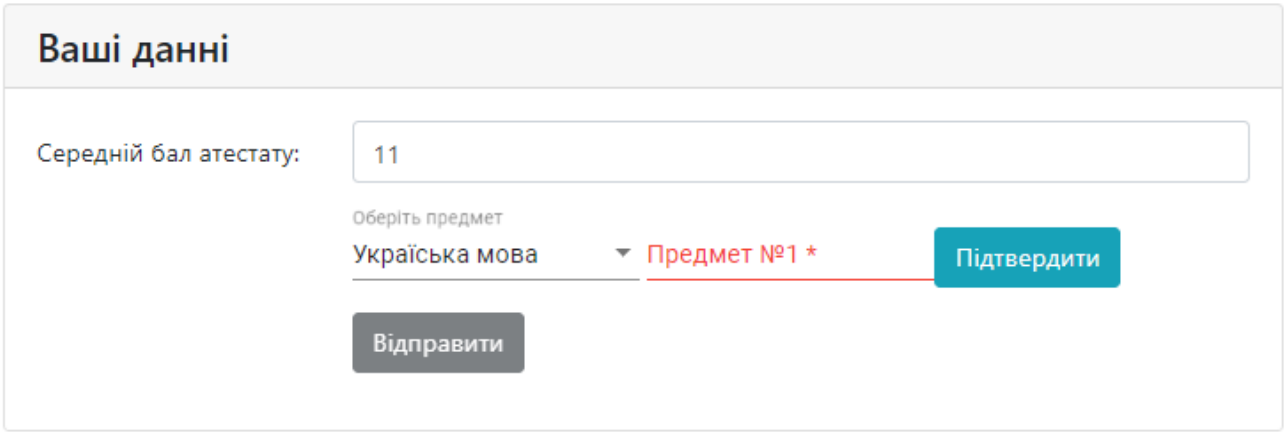


Рисунок 4.4.3 «Валідація форми»

Якщо користувач хоче пропустити крок 3 у списку, відкриється наступне вікно:

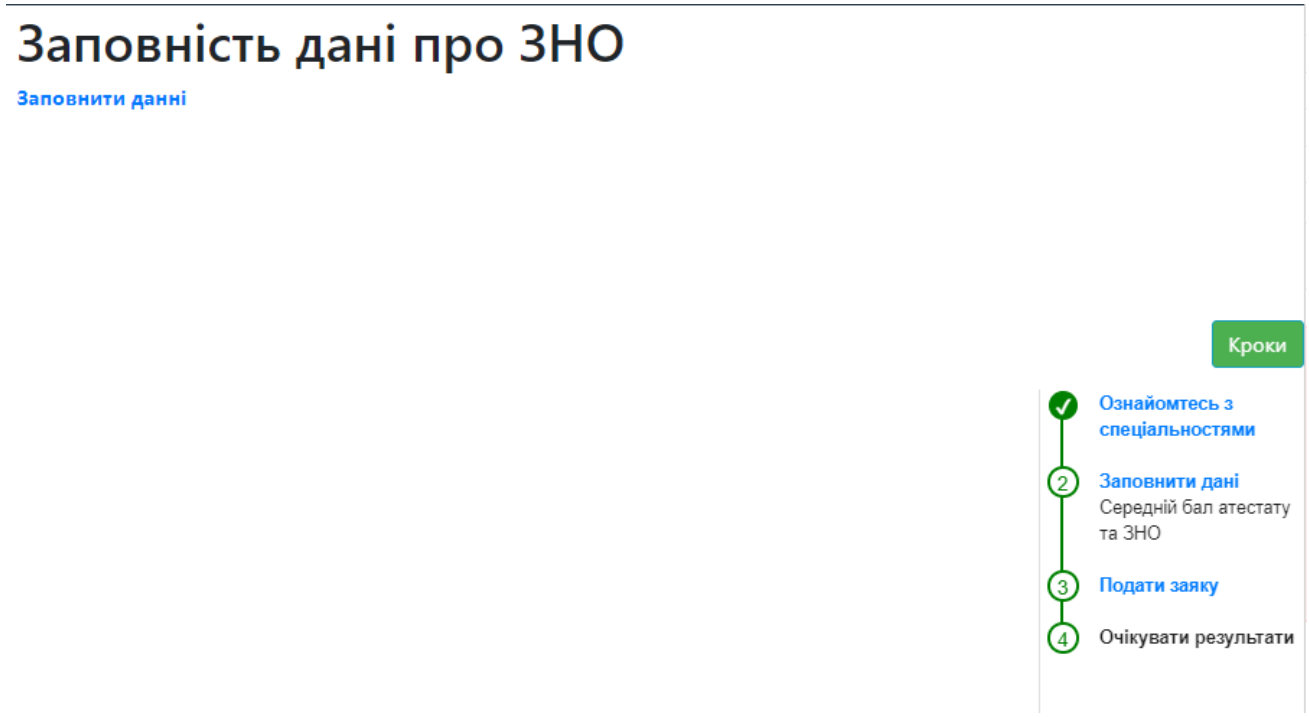


Рисунок 4.4.4 Заповнити попередній крок

Коли ми використовуємо наступне тіло запиту для доступу до REST API, ми отримуємо повідомлення про те, що формат запиту неправильний.

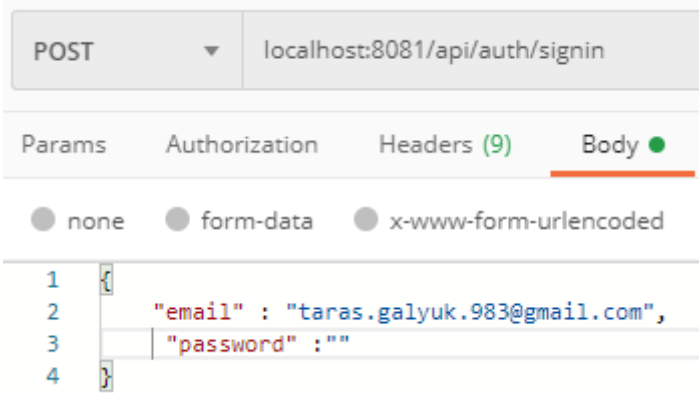


Рисунок 4.4.5 Некоректно сформований запит


```
{
    "timestamp": "2020-06-14T16:47:03.061+0000",
    "status": 400,
    "error": "Bad Request",
    "errors": [
        {
            "codes": [
                "NotBlank.loginRequest.password",
                "NotBlank.password",
                "NotBlank.java.lang.String",
                "NotBlank"
            ],
            "arguments": [
                {
                    "codes": [
                        "loginRequest.password",
                        "password"
                    ],
                    "arguments": null,
                    "defaultMessage": "password",
                    "code": "password"
                }
            ],
            "defaultMessage": "не может быть пусто",
            "objectName": "loginRequest",
            "field": "password",
            "rejectedValue": "",
            "bindingFailure": false,
            "code": "NotBlank"
        }
    ],
    "message": "Validation failed for object='loginRequest'. Error count: 1",
    "trace": "org.springframework.web.bind.MethodArgumentNotValidException: Validation fai
led for argument [0] in public org.springframework.http.ResponseEntity<?> ua.jackson.awsPr
actice.controller.AuthController.authenticateUser(ua.jackson.awsPractice.payload.request.L
oginRequest): [Field error in object 'loginRequest' on field 'password': rejected value []
; codes [NotBlank.loginRequest.password,NotBlank.password,NotBlank.java.lang.String,NotBla
nk]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes
[loginRequest.password,password]; arguments [];",
    "path": "/api/auth/signin"
}
```


Висновки до розділу

Як результат виконаї роботи ми отримуємо готову інформаційну систему для опрацювання заявок, яка є легкою і простою у використанні, із поясненням про подальші кроки, валідацію всіх модливих полів що в свою чергу покращує роботу системи та збільшує її надійність.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

Основна ідея проекту викладена в інформаційній схемі проекту, а її форма показана в таблиці 1.

Вид проекту	Розроблення програмного забезпечення
Назва проекту	Програмна система для вирішення задач навігації мобільних пристроїв на основі використання сигналів базових станцій стільникового зв’язку
Місце розроблення проекту	Національний лісотехнічний університет України
ПІБ автора проекту	Галюк Тарас Васильович
Цілі та задачі проекту	Необхідно реалізувати інформаційну систему яка буде опрацьовувати заявки для вступу в навчальний заклад: • Проаналізувати відомі методи та алгоритми, які будуть використовуватися в системі; • Розробка програмного забезпечення та алгоритмів для розробленої системи; • Проводити експериментальні дослідження проектованої системи та її комп'ютерного моделювання; • Графічне представлення результатів;
Короткий опис проекту	Головна ідея данної платформи полягає в тому щоб система могла опрацьовувати заявки подані вступником в певний навчальний заклад, в залежності від балів та результатів вступу системе видасть список всіх спеціальностей та напрямків які релевантні з вашими балами
Термін виконання	Не повинен перевищувати 6 місяців.
Запланований бюджет	100 000 грн.

Табл.1. Інформаційна карта проекту

5.2 Організаційний та фінансовий плани

Цільова аудиторія

Цільова аудиторія проекту - це державні або приватні навчальні заклади

Основні конкуренти

Аналогів данної платформи у нас в країні немає так як система заточена під використання електронного обігу документів, що на данний час є досить новим в сучасному світі.

Для розробки цього проекту достатньо невеликої групи програмістів або людей з досвідом розробки програмного забезпечення в цій галузі.

Після визначення розробника необхідно визначити фінансовий план проекту за різними фінансовими показниками. Очевидно, що кожен бізнес-план має включати законодавчо затверджені бухгалтерські, фінансові та звітні форми.

Взагалі кажучи, створення будь-якого програмного забезпечення включає три стандартні етапи, такі як: проектування програмного забезпечення, налагодження та тестування. Вартість розробки може бути визначена шляхом додавання вартості всіх ресурсів, які так чи інакше братимуть участь у створенні програмного забезпечення, і визначається відповідно до ринкових цін.У моєму випадку ці ресурси включають наступне:

1. ПК (процесор, блок живлення, жорсткий диск, материнська плата, оперативна пам’ять), монітор, клавіатура і миша = ***25 000 грн.***
2. ОС Windows 10 x64 = ***3 500 грн.***
3. Шість місяців роботи, при 5 робочих днях на тиждень і при 4 годинах роботи на день і при 40,00 грн. за одну годину роботи:

   4 години * 5 днів * 4 тижні * 6місяців * 65,00 грн./год. = ***41 600,00 грн.***
4. 4. Робота на ПК протягом однієї години споживає 340 Вт. Електрика, ціна 1,70 грн. На кіловат електроенергії, т:

   *4 години * 5 робочих днів * 1 місяць(4 тижні) * 6 місяців * 340 Вт. = 163 кВт.*

   *136 кВт * 1,70 грн./кВт =* ***277,00 грн.***

Отже загальна вартість розробки ПЗ буде становити:

25 000 + 3 500 + 41 600 + 277 ***= 70377 грн.***

5.3. Технічне забезпечення

Для використання даної платформи потрібно певне налаштоване програмне середовище а саме:

- Операційна система Linux server(Ubuntu, kubuntu, centOS…)
- Java 8+
- Spring Boot 2.2.2
- Gradle - Система автоматизованого збору проекту
- Node.js: v12
- Angular: 8
- PostgreSQL: 11

Для отпимального використання всіх ресурсіх віртуальної машини або сервера чудово підійте Amazon Web Services, а саме сервіс віртуальних машин EC2. Сервіс дозволяє вибрати відповідні конфігурації для машини, із різними процессорами та різними навантаженями

5.4. Ризики проекту

В процесі створення будь-якого проекту є ризики. Це тому, що накопичення певних подій матиме як позитивний, так і негативний вплив на проект. Крім того, ці події можуть мати різні невизначеності, наслідки або причини.

Оскільки ми не можемо заздалегідь знати, що може статися, ми вводимо значення, наприклад ймовірність ризику. Імовірність можна встановити від нуля до однієї сотої. Оскільки існування невизначеності має непрямий і прямий вплив на проект, ми повинні постаратися якнайкраще описати всі можливі зміни, які можуть відбутися під час реалізації проекту. Як правило, для визначення ймовірності можна використовувати один із двох існуючих методів, а саме: суб’єктивний, заснований на логіці процесу та числових алгоритмах прийняття рішень; мета може бути розрахована зі статичною ймовірністю відповідно до частоти події.

У разі недостатньої інформації слід надати резерв, який може бути виражений як додаткова сума або додатковий період. Такий резерв є дуже важливим інструментом, і він може знадобитися для вирішення проблем, які можуть виникнути в непередбачених обставинах.

У деяких випадках про заплановані зміни можна дізнатися заздалегідь. У цьому випадку у вас повинен бути план реагування, який зменшує ризик змін. Зазвичай ви можете використовувати інформаційне поле, щоб дізнатися про заплановані зміни. Якщо є переходи між різними рішеннями щодо впровадження проекту, ризики проекту також можуть змінюватися. У цьому контексті процедури аналізу інвестиційного ринку дуже важливі.

Висновки до розділу

Як підсумок цього розділу можна виділити деякі пункти: проаналізовано деякі проблеми, які можуть виникнути пристворенні проекту, та способи їх вирішення. Також у цьому розділі було обраховано ціну розробки програмного забезпечення.

Так як аналогів данної платформи немає і платформа націлена на одну з найважливіших сфер кожної держави, а саме освіта то ми впевнено можемо сказати що даний проект буде потрібний і хорошим кроком в розвитку сфери освіти загалом.

ВИСНОВКИ

Результатом магістерської роботи є інформаційна система, яка дозволяє опрацьовувати заявки абітурієнта та інформувати його про те, що заявка успішно опрацьована.

Для цього вступнику необхідно авторизуватися, щоб отримати доступ до розроблених у роботі інтерфесу системи та графічних компонентів для відображення списку вступників відповідно до заданих критерії та фільтрів.

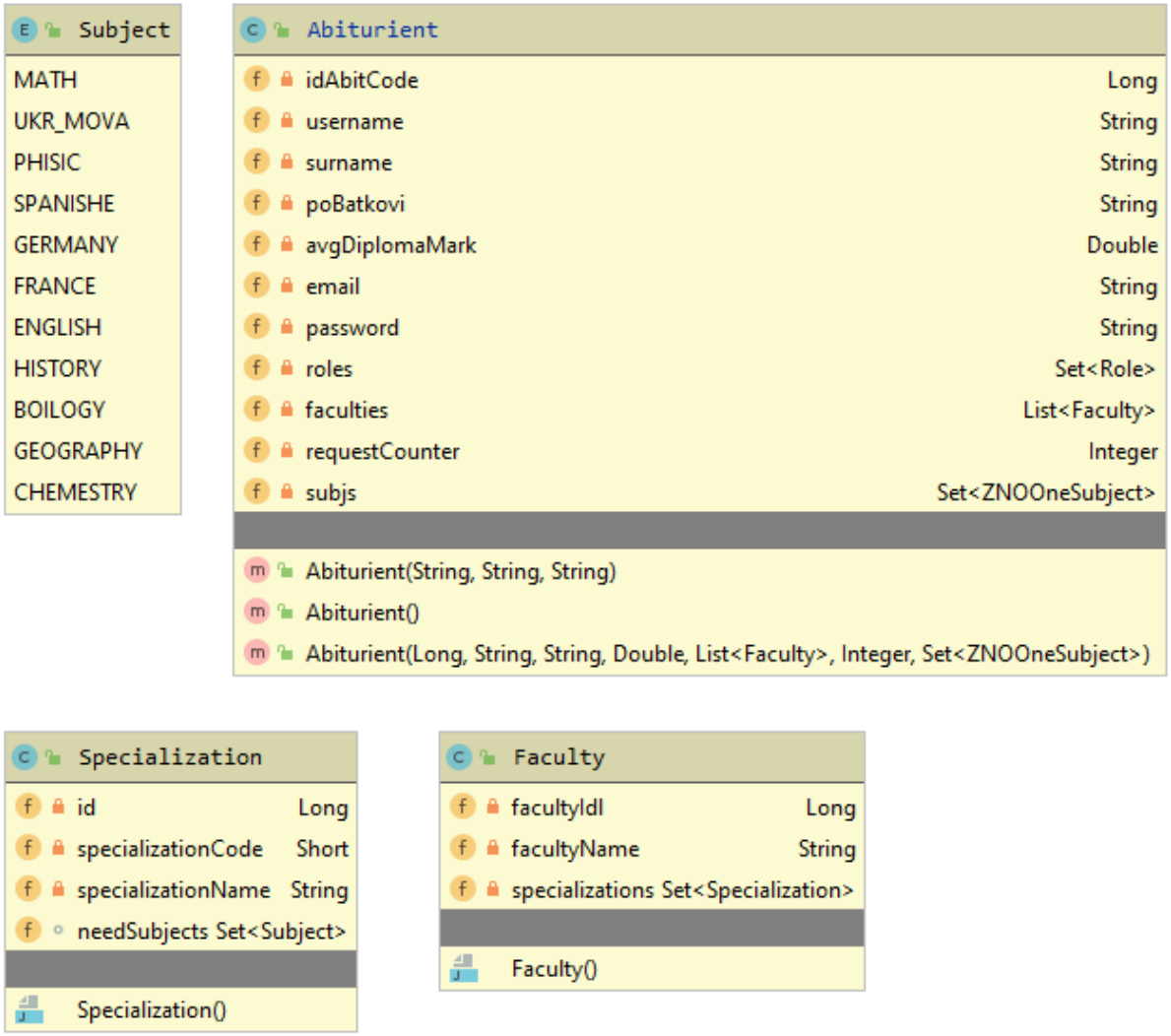
У разі зарахування вступника до даного навчального закладу його буде проінформовано електронним листом, листом на пошту або ж повідомленням у відповідному мессенджері.

Інформаційна система є повністю ізольованою і легко витримує великий обсяг навантажень заявок за одиницю часу що в свою чергу дозволяю платформі витримувати великі навантаження та працювати справно без жодних проблем на стороні сервера або клієнта.

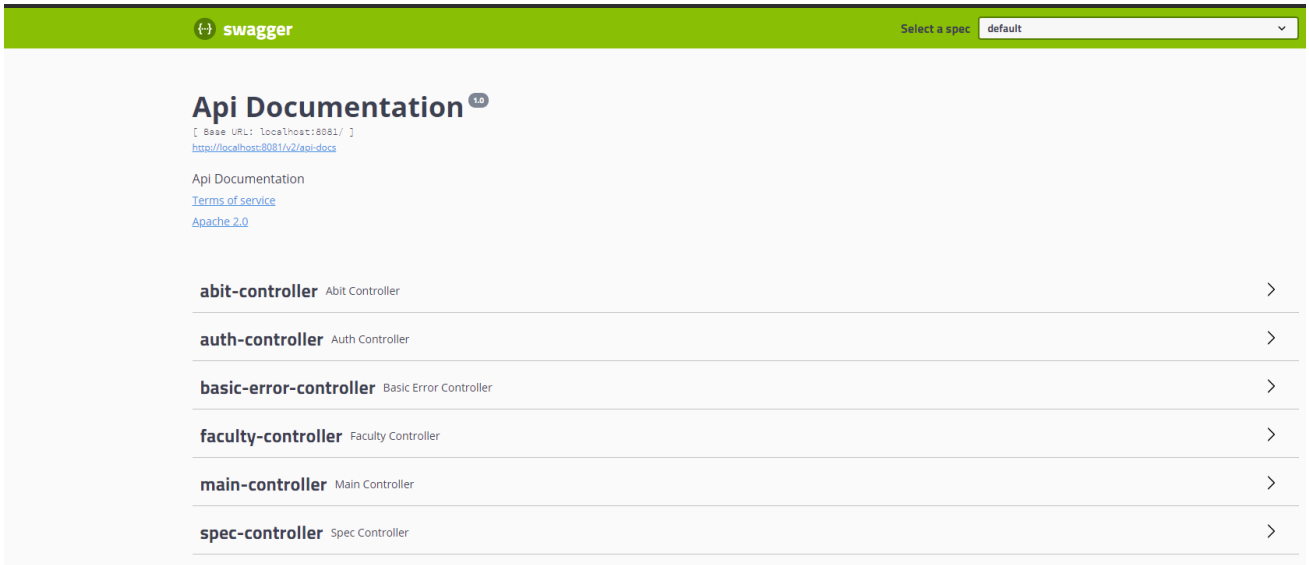
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Haliuk T. Information technology for processing applicants' applications / T. Haliuk, M.V.Dendiuk // Комп'ютерне моделювання та інформаційні технології: матеріали третьої науково-практичної конференції студентів, аспірантів та молодих вчених (Львів, 14-16 жовтня 2021 р.). – Львів: кафедра інформаційних технологій НЛТУ України, 2021. – С. 58-61.
2. Еккель Б . Філософія Java . 4-те повне вид. - СПб.: Пітер, 2018. - 1168 с.
3. Беркунський Є.Ю. Об'єктно-орієнтоване програмування мовою Java: Методичні вказівки для студентів напрямку " Комп'ютерні науки". – Миколаїв: НУК, 2006. – 52 с.
4. Нотон, Патрік. Повний довідник з Java / П. Нотон , Г. Шілдт . - М.: [б.в.]; К.: Діалектика, 1997. - 592 с.
5. Майкл Томас, Пратик Пател, Алан Хадсон, Доналд Болл(мл.) Секреты программирования для Internet на Java.- Ventana Press, Ventana Communications Group, U.S.A.,1996, Издательство "Питер Пресс", 1997. – 878 с.
6. Дейтел Х.М., Дейтел П.Дж. Как программировать на Java. Книга 1. Основы программирования – М.: Бином, 2006. – 848 с.
7. Дейтел Х.М., Дейтел П.Дж. Как программировать на Java. Книга 2. Файлы, сети, базы данных. – М.: Бином, 2006. – 672 с.
8. Хорстманн К. С., Корнелл Г. Java 2. Библиотека профессионала, том 1. Основы, 7-е изд. – М. : Издательский дом "Вильямс", 2006. – 896 с.
9. Хорстманн К. С., Корнелл Г. Java 2. Библиотека профессионала, том 2. Тонкости программирования, 7-е изд. – М.: Издательский дом "Вильямс", 2006. – 1168 стр.
10. Шилдт Г., Холмс Д. Искусство программирования на Java. – М.: Издательский дом "Вильямс", 2005. – 336 с.
11. Эккель Б. Философия Java. Библиотека программиста. 3-е изд. – СПб.: Питер, 2003. – 976 с.
12. Копитко М.Ф., Іванків К.С. Основи програмування мовою Java: Тексти лекцій. – Львів: Видавничий центр ЛНУ ім. Івана Франка, 2002. – 83 с.
13. Брнакевич І.Є., Вагін П.П. Програмування мовою Java: використання фундаментальних класів: Тексти лекцій. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2002. – 75 с.

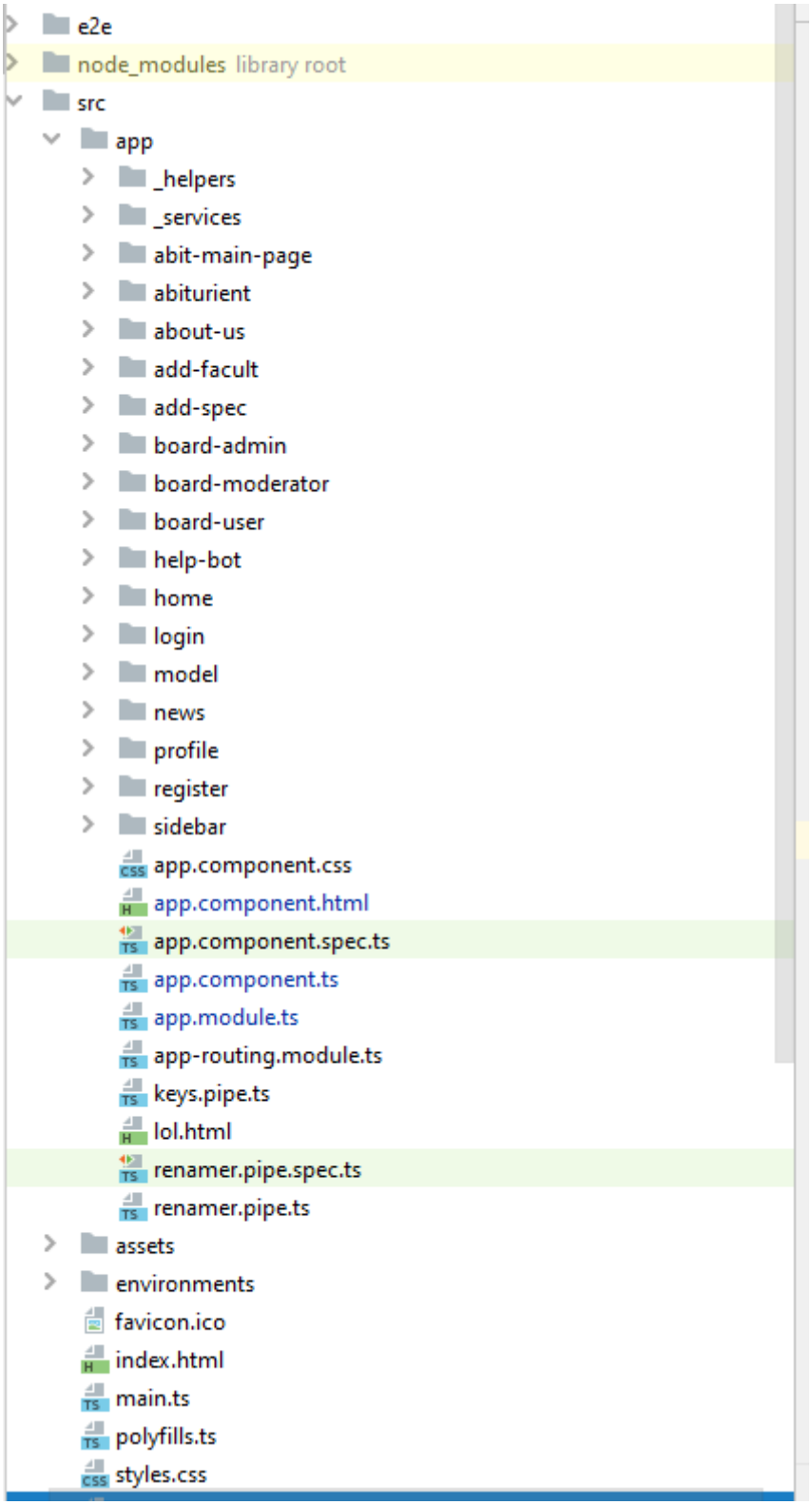
ДОДАТКИ



Діаграма entity класів



Класи-контроллери



Структура клієнтської частини

AbitController.java

```
@RestController
public class AbitController {
    @Autowired
    FacultyRepo facultyRepo;
    @Autowired
    SpecRepo specRepo;
    @Autowired
```

```
    private AbitRepos abitRepos;
    @Autowired
    private AbiturientService abitS;
    @Autowired
    UpdateAbitService updateAbitService;
    @GetMapping(value = "/allAA")
    public List<Abiturient> allADD(){
        return abitS.getAllAbits();
    }

    @CrossOrigin("http://localhost:4200")
    @PostMapping(value = "/addAbit", produces = MediaType.APPLICATION_JSON_VALUE, consumes =
MediaType.APPLICATION_JSON_VALUE)
    public void addedAbitur(@RequestBody UpdateAbitRequest updateAbitRequest){
        this.updateAbitService.updateAbbit(updateAbitRequest);
        System.out.println(updateAbitRequest.toString());
    }
    @CrossOrigin("http://localhost:4200")
    @GetMapping(value = "/allAb")
    public List<Abiturient> allAbitsFullInfo(){
        return abitRepos.findAll();
    }
    @GetMapping("/max")
    public Abiturient maxAVG() {
        return abitS.abitWithMaxAvgMark();
    }
    @CrossOrigin("http://localhost:4200")
    @GetMapping("/getById/{id}")
    public Abiturient getOne(@PathVariable Long id) {
        Abiturient one = this.abitRepos.getOne(id);
        Abiturient abiturient = new Abiturient(
                one.getIdAbitCode(),
                one.getSurname(),
                one.getPoBatkovi(),
                one.getAvgDiplomaMark(),
                one.getFaculties(),
                one.getRequestCounter(),
                one.getSubjs());
        return abiturient; }

    @DeleteMapping("/del/{id}")
    public void deleteById(@PathVariable Long id){ Abiturient one = abitRepos.getOne(id);
one.setFaculties(null); abitRepos.delete(one);}
    @GetMapping(value = "/max/{subject}", produces = MediaType.APPLICATION_JSON_VALUE, consumes =
MediaType.APPLICATION_JSON_VALUE) public List<Abiturient> maxBySubject(@PathVariable Subject subject){
return  abitS.abitWithGreaterMark(subject,"");}}
```

AuthController.java

```
@RestController
@RequestMapping("/api/auth")
public class AuthController {
    @Autowired
    AuthenticationManager authenticationManager;
    @Autowired
    AbitRepos abitRepos;
    @Autowired
    RoleRepository roleRepository;
    @Autowired
    PasswordEncoder encoder;
    @Autowired
    JwtUtils jwtUtils;
    @PostMapping("/signin")
    public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {
        Authentication authentication = authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(loginRequest.getEmail(),
loginRequest.getPassword()));
        SecurityContextHolder.getContext().setAuthentication(authentication);
        String jwt = jwtUtils.generateJwtToken(authentication);
        AbitDetailsImpl userDetails = (AbitDetailsImpl) authentication.getPrincipal();
        List<String> roles = userDetails.getAuthorities()
                .stream()
                .map(GrantedAuthority::getAuthority)
```

```
                .collect(Collectors.toList());
        return ResponseEntity.ok(new JwtResponse(jwt,
                userDetails.getId(),
                userDetails.getUsername(),
                userDetails.getSurname(),
                userDetails.getEmail(),
                roles));}
    @PostMapping("/signup")
    public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest signUpRequest) {
        if (abitRepos.existsByEmail(signUpRequest.getEmail())) {
            return ResponseEntity
                    .badRequest()
                    .body(new MessageResponse(" Email уже використовується!"));}
        Abiturient user = new Abiturient(signUpRequest.getUsername(),
                signUpRequest.getEmail(), encoder.encode(signUpRequest.getPassword())); Set<String>
strRoles = signUpRequest.getRole();
        Set<Role> roles = new HashSet<>();
        if (strRoles == null) {Role userRole = roleRepository.findByName(ERole.ROLE_USER)
                .orElseThrow(() -> new RuntimeException(" Роль не знайдена."));
            roles.add(userRole);} else {strRoles.forEach(role -> { switch (role) {
                    case "admin":Role adminRole = roleRepository.findByName(ERole.ROLE_ADMIN)
                            .orElseThrow(() -> new RuntimeException(" Роль не знайдена."));
                        roles.add(adminRole);break; case "mod": Role modRole =
roleRepository.findByName(ERole.ROLE_MODERATOR).orElseThrow(() -> new RuntimeException(" Роль не
знайдена."));roles.add(modRole); break;default:
                        Role userRole = roleRepository.findByName(ERole.ROLE_USER.orElseThrow(() -> new
RuntimeException(" Роль не знайдена.")); roles.add(userRole); } }); }
        user.setRoles(roles); abitRepos.save(user); return ResponseEntity.ok(new
MessageResponse("Користувач успішно зареєстрований!!!"));}}
```

About-us.ts

```
export class AboutUsComponent implements OnInit {
  name: string;
  facult : FacultiesDTO[];
  abit = new Abit();
  private FacultyPassElement = new FacultiesDTO();
  private counter: number;
  options: FormGroup;
  private reqq2: Abit;
  constructor(fb: FormBuilder,private  abitutientServiceService: AbitutientServiceService,
private cd: ChangeDetectorRef, private httpClient: HttpClient, private
notificationService:NotificationService) {

    this.options = fb.group({
      bottom: 0,
      fixed: false,
      top: 0});}
  @ViewChild('outerSort', { static: true }) sort: MatSort;
  public initTable1 = false;
  ngOnInit() {
    let item = window.localStorage.getItem("auth-user");
    let parse = JSON.parse(item);
    let id = parse.id;
    this.initTable();
    this.abitutientServiceService.updateSpec.subscribe(
      res=>{
        if(res===true && this.facult != null){
          console.log("DWA");
          this.abit.specializations.forEach(value => {
            this.changeChooseStatus(value, this.facult);
            this.abitutientServiceService.updateSpec.next(false);})
          if(!this.initTable1) {setTimeout(()=>{this.printt();},30) this.initTable1 =
true;}}});

    this.httpClient.get<Abit>('http://localhost:8081/getById/' + id, httpOptions).subscribe(
      res => { this.abit = res;
```

```
        this.counter = res.requestCounter;
      });}
  @ViewChildren('innerSort') innerSort: QueryList<MatSort>;
  @ViewChildren('innerTables') innerTables: QueryList<MatTable<Specializations>>;
  dataSource: MatTableDataSource<Faculties>;
  usersData: Faculties[] = [];
  columnsToDisplay = ['facultyIdl', 'facultyName'  ]  ;
  innerDisplayedColumns = ['id', 'specializationCode', 'specializationName', 'requst'];
  expandedElement: Faculties | null;
  private isOpen: boolean = false;
  toggleRow(element: Faculties) {
    element.specializations && (element.specializations as
MatTableDataSource<Specializations>).data.length ? (this.expandedElement = this.expandedElement
=== element ? null : element) : null;
    this.cd.detectChanges();
    this.innerTables.forEach((table, index) => (table.dataSource as
MatTableDataSource<Specializations>).sort = this.innerSort.toArray()[index]);
  }
  applyFilter(filterValue: string) {
    this.innerTables.forEach((table) => (table.dataSource as
MatTableDataSource<Specializations>).filter = filterValue.trim().toLowerCase());
  }
  printt(){
    this.facult.forEach(user => {
      if (user.specializations && Array.isArray(user.specializations) &&
user.specializations.length) {
        this.usersData = [...this.usersData, {
          ...user,
          specializations: new MatTableDataSource(user.specializations)
        }];
      } else {
        this.usersData = [...this.usersData, user];
      }
    });
    this.dataSource = new MatTableDataSource(this.usersData);
    this.dataSource.sort = this.sort;
    this.initTable1 = false;
  }
  decreaseCounter(){
    this.counter--;
  }
  printAll(element: any) {
    element.isChoosen = true;

    let item = window.localStorage.getItem("auth-user");
    let parse = JSON.parse(item);
    let id = parse.id;
    let element1 = element;
    let specc  : SpecializationsDTO[] = [];
    specc.push(element1);
    this.FacultyPassElement.specializations = specc;

    this.httpClient.put("http://localhost:8081/setFaculty/"+id,
      this.FacultyPassElement, httpOptions).subscribe(
      value => {
        console.log(value);
      });
    this.decreaseCounter();
    this.notificationService.success("Запит відправлено")
  }
  isChoosen(element: SpecializationsDTO) {
    return element.isChoosen;
  }
  initTable(){
    let item = window.localStorage.getItem("auth-user");
```

```
    let parse = JSON.parse(item);
    let id = parse.id;
    this.httpClient.get<FacultiesDTO[]>('http://localhost:8081/allFa/'+ id,
httpOptions).subscribe(
      value => {
        if(value != null) {
          this.facult = value;
          console.log(value);
          setTimeout(()=>{
            this.abitutientServiceService.updateSpec.next(true);
          }, 100);
        }})}
  changeChooseStatus(value2: SpecializationsDTO, value: any){
    value.forEach(value1 =>  value1.specializations.forEach(
      value2 =>{
        console.log(this.facult);
        for(let i = 0; i < this.abit.specializations.length; i++) {
          if (this.abit.specializations[i].id == value2.id) {
            value2.isChoosen = true;
            break;
          }
        }

      }));

  }

}
```

Abiturient.java

```
@Entity
@Table(    name = "Abiturients")
public class Abiturient {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long idAbitCode;
    private String username;
    private String surname;
    private String poBatkovi;
    private Double avgDiplomaMark;
    @NotBlank
    @Size(max = 50)
    @Email
    private String email;
    @NotBlank
    @Size(max = 120)
    private String password;
    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(    name = "user_roles",
            joinColumns = @JoinColumn(name = "user_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id"))
    private Set<Role> roles = new HashSet<>();
    @ManyToMany()
    @JsonIgnoreProperties("abiturients")
    @OnDelete(action = OnDeleteAction.CASCADE)
    private List<Faculty> faculties;
    private Integer requestCounter;
    @ElementCollection
    private Set<ZNOOneSubject> subjs = new HashSet<>(4);

public Abiturient(String username, String email, String password) {
    this.username = username;
    this.email = email;
    this.password = password;}}
```

WebSecurityConfig.java

```
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired AbitDetailsServiceImpl abitDetailsService;
    @Autowiredprivate AuthEntryPointJwt unauthorizedHandler;
    @Bean public AuthTokenFilter authenticationJwtTokenFilter() {return new AuthTokenFilter();
}
    @Override public void configure(AuthenticationManagerBuilder authenticationManagerBuilder)
throws Exception {
authenticationManagerBuilder.userDetailsService(abitDetailsService).passwordEncoder(passwordEnc
oder());}
    @Bean @Override public AuthenticationManager authenticationManagerBean() throws Exception {
return super.authenticationManagerBean();}
    @Bean public PasswordEncoder passwordEncoder() {return new BCryptPasswordEncoder();}
    @Override protected void configure(HttpSecurity http) throws Exception {
http.cors().and().csrf()
.exceptionHandling().authenticationEntryPoint(unauthorizedHandler).and()
.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
                .authorizeRequests().antMatchers("/api/auth/**").permitAll()
                .antMatchers("/api/test/**", "/**").permitAll()
                .anyRequest().authenticated();
        http.addFilterBefore(authenticationJwtTokenFilter());}}
```