

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

**Розробка та реалізація освітнього ресурсу на основі веб-
технологій**

**Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» 121**

Керівник курсової роботи
к.ф.-м.н., ст.викл. Гречко А. В.

(підпис)

“ ____ ” _____ 2020 р.

Виконав студент 3-го курсу
Бойчук О.Р.

“ ____ ” _____ 2020 р.

Київ 2020

Календарний план виконання курсової роботи

Тема: Розробка та реалізація освітнього ресурсу на основі веб-технологій

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	Жовтень- листопад 2019р.	
2.	Вивчення та аналіз задачі	Листопад- грудень 2019р.	
3.	Розробка архітектури та загальної структури програми	Січень- лютий 2020р.	
4.	Створення веб-застосунку	Лютий- березень 2020р.	
5.	Написання текстової частини	Березень- квітень 2020р.	
6.	Перегляд курсової роботи науковим керівником	Травень 2020р.	
7.	Створення презентації	Травень 2020р.	
8.	Захист курсової роботи		

Студент Бойчук О.Р. _____

Керівник Гречко А.В. _____

“_____” _____ 2020 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	4
АНОТАЦІЯ	5
ВСТУП.....	6
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАННЯ КУРСОВОЇ РОБОТИ.....	8
1.1. Аналіз сучасного стану питання та обґрунтування теми	8
1.2. Огляд існуючих аналогів розробки	10
1.3. Постановка завдання	13
РОЗДІЛ 2 ТЕОРЕТИЧНІ ВІДОМОСТІ	14
2.1. REST.....	14
2.2. Cookies	16
РОЗДІЛ 3 ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ	19
3.1. Обґрунтування алгоритму й структури програми	19
3.2. Обґрунтування вибору засобів розробки	20
3.3. Опис розробки програми.....	22
3.5. Тестування програми і результати її виконання.....	26
ВИСНОВКИ	35
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	36
ДОДАТОК А Код клієнта – профіль викладача	37
ДОДАТОК Б Код сервера – налаштування сервера і підключення до бази даних	41

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

REST – Representational State Transfer.

URL – Uniform Resource Locator.

API – Application Program Interface.

HTTP – Hypertext Transfer Protocol.

PaaS – Platform as a service.

MVC – Model-View-Controller.

СКБД – Система Керування Базами Даних.

JVM – Java Virtual Machine.

NPM – Node Package Manager.

JSON – JavaScript Object Notation.

JWT – JSON Web Token.

АНОТАЦІЯ

Робота присвячена створенню веб-застосунку для вивчення та вдосконалення іноземних мов за допомогою менторства.

Проект створено на клієнт-серверній архітектурі. Рівень клієнта реалізований з використанням Angular фреймворку, а сервер на мові програмування JavaScript з використанням фреймворку Express.js. Для збереження даних використано реляційну базу даних PostgreSQL.

ВСТУП

Освіта – основа розвитку особистості, як культурного, так і духовного, фізичного, вона є запорукою розвитку, сприяє економічному добробуту та успішній соціалізації людини. Особливо гостро питання освіти постає в наш час, коли розвиток технологій дає можливість все більшій частині людей отримувати її, докладаючи при цьому все менше зусиль. Ця тема є актуальною для суспільства, тому що в сучасних реаліях без володіння різними мовами людині важко реалізувати свій потенціал та досягти поставлених цілей. З наукової точки зору актуальність супроводжується у дослідженні можливості передання накопиченого досвіду від людини до групи осіб за допомогою неформального спілкування на певну обрану тему. Практичне значення полягає у тому, щоб показати дієвість такого підходу цим самим привернути увагу до даного методу навчання.

Метою роботи є створення веб-ресурсу, який допоможе удосконалити знання іноземних мов за допомогою менторів. Завдання роботи полягають у наступному:

- 1) З'ясування масштабів проблеми та необхідності створення саме такого веб-ресурсу.
- 2) Аналіз аналогів з детальним виокремленням недоліків та переваг, щоб в подальшому використати ці дані у створенні свого продукту.
- 3) Окреслення цільової аудиторії.

Об'єкт дослідження – процес вивчення та вдосконалення іноземних мов за допомогою веб-ресурсів.

Предмет дослідження – менторство, як запорука успіху в усуненні недоліків та вивченні аспектів іноземної мови.

Для досягнення мети цієї роботи я використав таке програмне забезпечення:

- Front-end: Angular 8
- Back-end: Node JS + Express.js

- Database: PostgreSQL
- PaaS: Heroku

Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

1.1. Аналіз сучасного стану питання та обґрунтування теми

Питання знання мов зараз стоїть чи не одним з перших у галузі освіти та й в загальному житті людей. Одним з найбільш корисних аспектів людського досвіду є наша здатність встановлювати контакт з іншими людьми. Можливість спілкуватися з кимось на його або її мові – це неймовірний дар. Білінгви мають унікальну можливість спілкуватися з більш широким колом людей у своєму особистому та професійному житті. Знання мови робить вас місцевим незалежно від того, де ви знаходитесь, відкриваючи свій світ буквально і фігурально. Ви будете сформовані громадами. Ви будете приголомшені добротою незнайомих. Ви будете будувати дружбу на все життя. І тільки з цих причин ви побачите нагороду за вивчення мов протягом багатьох наступних років.

Згідно з дослідження Обернського університету [1] знання іноземних мов дає чимало переваг, до прикладу:

- Іноземні мови забезпечують конкурентну перевагу у виборі професії;
- Креативність підвищується з вивченням іноземних мов;
- Такі навички, як вирішення проблем, робота з абстрактними поняттями, підвищуються при вивченні іноземної мови;
- Вивчення іноземної мови розширює можливості людини в галузі державного управління, бізнесу, медицини, юриспруденції, технології, військової справи, промисловості, маркетингу і т. д.;
- Аналітичні навички покращуються, коли студенти вивчають іноземну мову;
- Ділові навички плюс знання іноземної мови роблять співробітника більш цінним на ринку;
- Вивчення іноземної мови підвищує навички аудіювання та пам'яті;

- Вивчення іноземної мови створює більш позитивне ставлення і менше упереджень по відношенню до інших людей...

Розширення глобалізації призвело багатьох людей до розуміння переваг вивчення іноземної мови. Багато дорослих повертаються до школи або проходять програму вивчення мови, щоб залишатися добре обізнаними в основних мовах, на яких говорять в глобальному бізнесі. Однак є й особисті причини, за якими люди хочуть вивчати другу або третю мову. Не маючи за плечима іноземної мови, ви ризикуєте втратити цілий світ можливостей. Основною проблемою вивчення мов онлайн є те, що для вивчення мови потрібно говорити нею, бажано з носієм, проте більшість ресурсів не надають такої можливості.

1.2. Огляд існуючих аналогів розробки

На сьогоднішній день існує дуже мало веб-застосунків, які б давали змогу групам людей спілкуватися однією мовою з носіями цим самим удосконалюючи її. При пошуку додатку для вивчення мови користувач найвірогідніше знайде звичайний сайт, що має вузький функціонал по вивченню слів та правопису мови. Розглянемо декілька найпопулярніших застосунків які дають можливість спілкування з менторами.

HiNative – це програма для запитань і відповідей(див. рис. 1) для тих, хто вивчає мову, де ви можете задавати питання носіям мови та культури [3]. Мови: на момент написання дослідницької роботи *HiNative* хвалився підтримкою 113 мов від носіїв мови з 170 країн. Коли ви створюєте обліковий запис ви можете задавати питання про будь-яку мову і отримати відповіді від реальних носіїв мови. Ви також можете надсилати голосові запитання та відповіді за допомогою простого інтерфейсу програми.

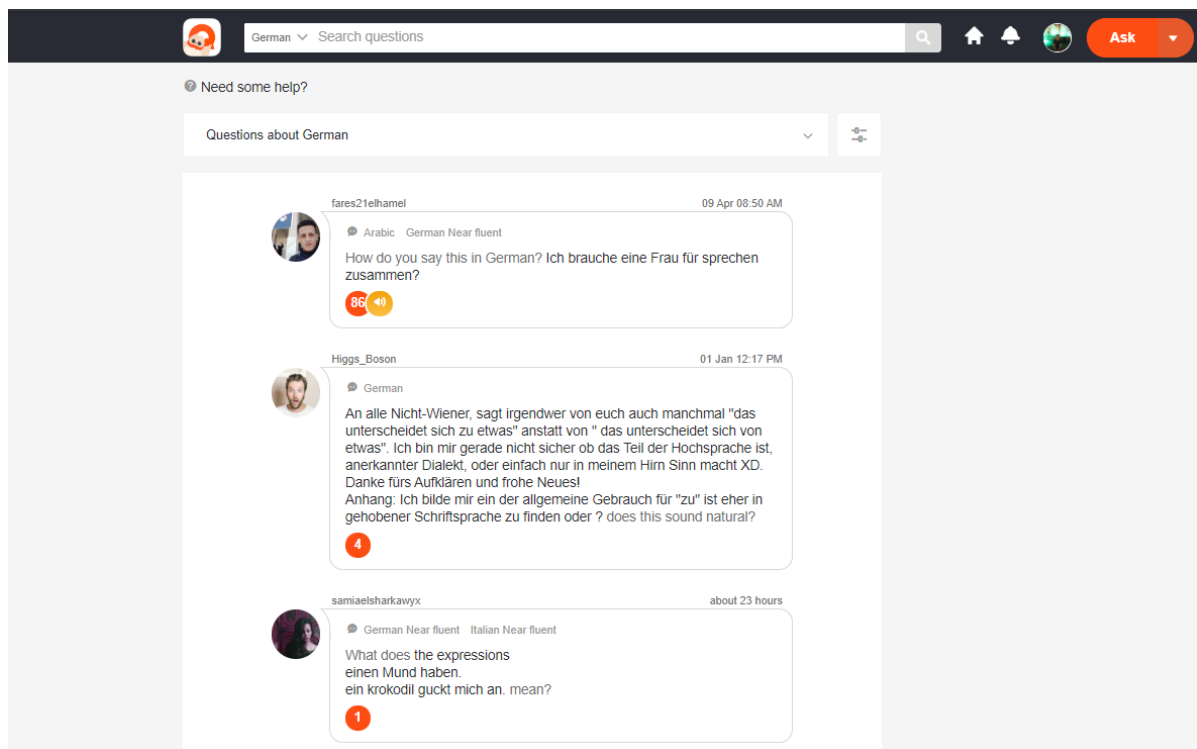


Рисунок 1 - Інтерфейс HiNative

З плюсів даної системи можна виділити простий в дизайні і зручний у використанні інтерфейс, та можливість задати практично будь-яке питання про мову / культуру і швидко отримати на нього відповідь. З мінусів – не всі функції доступні для безкоштовних користувачів.

HiNative хороша для тих, хто вивчає мову, і хоче отримати допомогу з основами, а також для людей, які цікавляться різними культурами.

Polyglot Club – це соціальна мережа мовного обміну, де ви можете спілкуватися з носіями вашої цільової мови, щоб практикувати різні мови онлайн і оффлайн, а також заводити друзів і відкривати для себе нові культури [2]. Мови: на момент написання дослідницької роботи було зареєстровано 894,634 членів з більш ніж 150 країн, так що ви можете вивчити практично будь-яку мову. Ви знаходите партнерів з мовного обміну(див. рис. 2), надсилаєте письмовий текст для виправлення, дивіться відео з вивчення мови, якими діляться члени спільноти, спілкуйтеся в чатах і навіть відвідуєте заходи з мовного обміну клубу і зустрічі з іншими учнями.

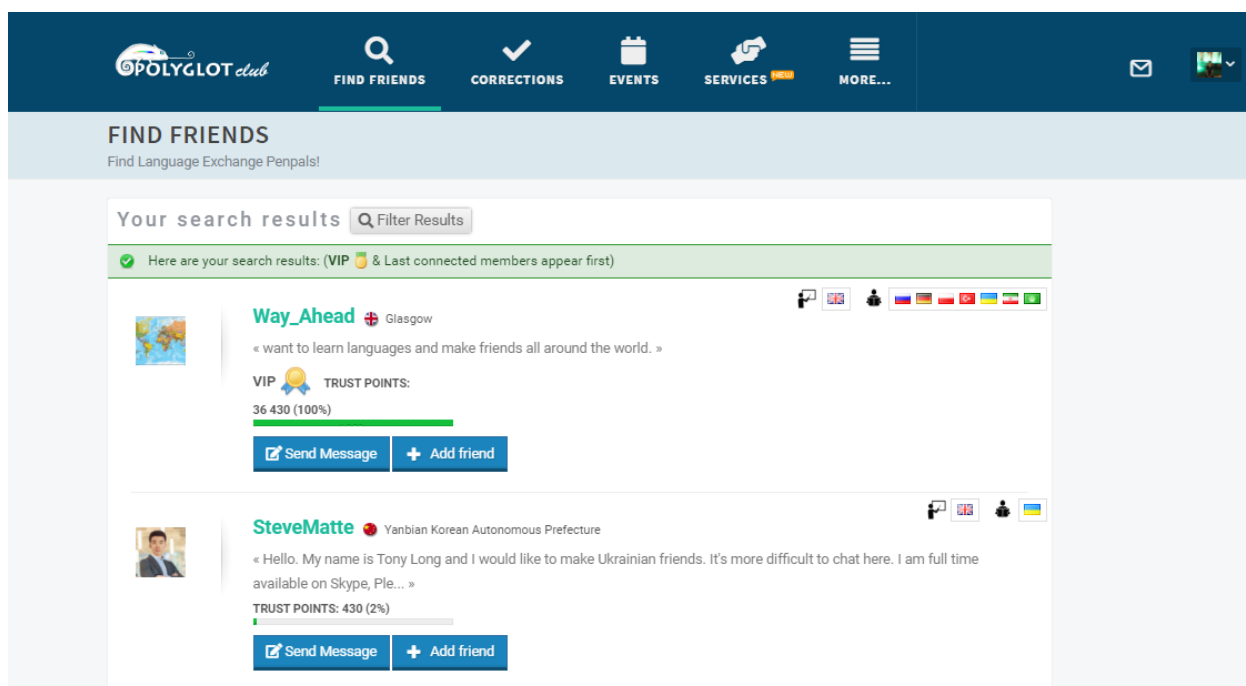


Рисунок 2 - Інтерфейс Polyglot Club

Плюсом є можливість познайомитись з членами спільноти на заняттях, курсах, семінарах, зустрічах, міжнародних вечірках та інших заходах, що

проводяться по всьому світу. Мінус полягає у тому, що знайти когось для довгострокового мовного обміну може бути складно, також інтерфейс програми не інтуїтивно зрозумілий для пересічного користувача.

Polyglot Club – хороше місце для студентів, мандрівників і всіх, хто зацікавлений у вивченні нових мов і різних культур, а також любить знайомитися з новими людьми.

Lang-8 – це платформа для вивчення мови, де носії мови виправляють те, що ви пишете (див. рис 3) [4]. Щоб приєднатися до спільноти, вам потрібно створити обліковий запис. Мови: є носії мов з більш ніж 190 країн, і ви можете знайти партнера з мовного обміну для вивчення 90 мов. Ви пишете текст на мові, яку вивчаєте і носій мови виправляє ваш лист. Також ви можете допомогти іншим людям вивчити вашу рідну мову, виправляючи їхні тексти.

Плюси: можливість покращити свою граматику, можна писати різні тексти від одного речення до цілих творів. Мінусами для безкоштовних акаунтів є обмеження кількості вивчення мов, що складає тільки дві мови, також ви завжди будете бачити рекламу, що може бути досить дратівливим.

Lang-8 – ідеальний сервіс для любителів граматики, які хочуть поліпшити свої навички письма.

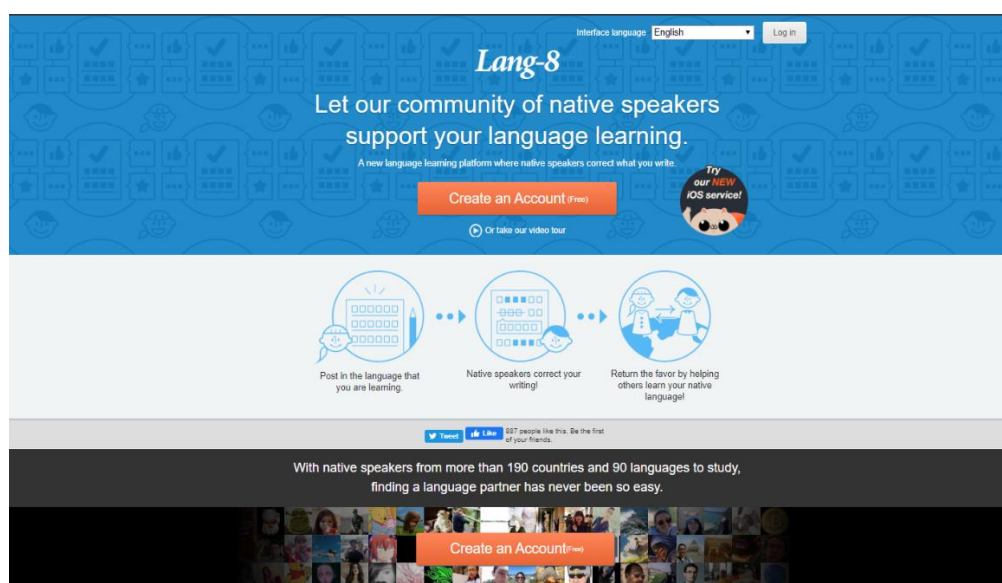


Рисунок 3 - Інтерфейс Lang-8

1.3. Постановка завдання

Створити веб-застосунок для вивчення та вдосконалення іноземних мов за допомогою менторства.

Основні модулі системи:

- 1) Клієнтський застосунок для окремих користувачів(викладачів, студентів), який би задовольняв наступним вимогам:
 - а) Проектування інтуїтивно зрозумілого та простого графічного інтерфейсу користувача.
 - б) Реалізація функціоналу необхідного клієнту для взаємодії з серверною частиною.
- 2) Серверний застосунок для реалізації взаємодії між користувачами ресурсу. Основні вимоги:
 - а) Реалізація функціоналу необхідного серверу для взаємодії з клієнтом.
 - б) Проектування та реалізація взаємодії з СКБД.
 - с) Створення захищеної системи авторизації та автентифікації користувачів.

Функціональні можливості системи:

- 1) Реєстрація та авторизація користувача.
- 2) Редагування та видалення інформації про користувача.
- 3) Створення, редагування та видалення статей.
- 4) Створення, редагування та видалення уроків з вивчення мови.
- 5) Можливість підписуватись та відписуватись на менторів.
- 6) Створення та видалення коментарів до уроку в режимі реального часу, тобто коли один користувач додає коментар у іншого він відображається автоматично без оновлення сторінки.

РОЗДІЛ 2

ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1. REST

Почнемо з основного – архітектури на якій побудований веб-застосунок, у нашому випадку це REST – архітектурний стиль, вперше описаний в докторській дисертації Роя Філдінга з архітектурних стилів і проектування мережових архітектур. На даний момент цей підхід проектування та побудови застосунків є найпопулярнішим і практично витіснив інші. Розберемо основні принципи які сповідує REST архітектура [5]:

- Клієнт-сервер: ми повинні чітко відокремити користувацький інтерфейс від зберігання даних, таким чином покращуємо масштабованість нашого проекту, також це дає змогу розробляти серверну частину незалежно від клієнтської і навпаки.
- Відсутність стану: кожен запит від клієнта до сервера повинен містити всю інформацію, необхідну для розуміння запиту, і не може використовувати переваги будь-якого збереженого контексту на сервері. Таким чином, стан сеансу повністю зберігається на клієнті.
- Кешування: обмеження кешування вимагають, щоб дані у відповіді на запит були неявно або явно позначені як кешовані або некешовані. Якщо відповідь може бути кешована, то клієнтському кешу надається право повторно використовувати ці дані відповіді для наступних еквівалентних запитів.
- Єдиний інтерфейс: визначає інтерфейс між сервером і клієнтом. Це дає змогу відокремити архітектуру, що дає дозвіл окремим частинам самостійно розвиватися.
- Багаторівнева система: стиль багаторівневої системи дає змогу архітектурі складатися з ієрархічних шарів, які обмежують поведінку

компонентів так, що компонент може "бачити" тільки в межах шару, з яким він взаємодіє.

Веб-додаток REST надає інформацію про себе у вигляді інформації про свої ресурси. Він також дозволяє клієнту виконувати дії з цими ресурсами, наприклад створювати нові ресурси (створювати нового користувача) або змінювати існуючі ресурси.

Те, що робить сервер, коли клієнт, викликає один з його API, залежить від 2 речей, які повинні надатись серверу:

- Ідентифікатор ресурсу – URL-адреса ресурсу.
- Операція, яку сервер повинен виконати на цьому ресурсі, у вигляді методу HTTP. Найбільш поширеними методами HTTP є GET, POST, PUT і DELETE.

Ще однією ключовою особливістю додатків REST є використання стандартних HTTP-дієслів і кодів помилок при пошуку або усуненні непотрібних варіацій між різними сервісами.

2.2. Cookies

Розпочнемо з термінології: cookie – термін для пакета даних, який комп'ютер отримує, а потім відправляє назад, не змінюючи і не заміняючи його [6]. При відвідуванні веб-ресурсу, що використовує cookie сервер надсилає файл cookie на комп'ютер, який в свою чергу зберігає його у файлі, що розташований всередині веб-браузера. Це все робиться для того, щоб допомогти веб-сайту відслідкувати активність користувача та відвідування. Основною метою використання cookie в цій роботі є запис та зберігання реєстраційної інформації користувача. Це корисно, бо користувачу не доводиться кожного разу при повторному відвідуванні сайту проходити процедуру авторизації.

```
Set-Cookie: token= eyJhbGciOiJIUzI1NiIsInR5cCI6Ikk
```

Вище наведений приклад заголовку який приходить з сервера та просить клієнт зберегти вказаний cookie. Відповідь (показана нижче), що відправляється браузеру, містить заголовок Set-Cookie, і cookie запам'ятовується браузером.

```
Status Code: 200 OK
```

```
Content-Type: application/json; charset=utf-8
```

```
Set-Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6Ikk;Max-Age=3600;
```

Тепер при кожному новому запиті на сервер, за допомогою заголовку Cookie браузер буде надсилати всі збережені cookie. Приклад такого запиту нижче:

```
Request Method: GET
```

```
Host: poliglout.herokuapp.com
```

```
Cookie: token= eyJhbGciOiJIUzI1NiIsInR5cCI6Ikk
```

Cookies, що використовуються в даній роботі називаються permanent cookies(англ. постійні cookies) – видаляються не при закритті браузера, а

протягом певного періоду, який вказується на сервері за допомогою атрибута Max-Age.

2.3. JSON Web Token

Аутентифікація – це одна з найважливіших складових будь-якої програми. Безпека – це завжди те, що змінюється і розвивається.

JWT – це відкритий стандарт, що дає можливість безпечно передати інформацію між клієнтом і сервером у вигляді об'єкта JSON. Інформація може бути перевірена та довірена, оскільки вона має цифровий підпис.

При автентифікації, коли користувач успішно увійшов у систему, використавши свої дані на клієнт повертається токен. Оскільки в ньому зберігаються дані користувачів, потрібно з обережністю підходити до заходів безпеки та його зберігання. Загалом, ви не повинні зберігати токени довше, ніж потрібно. Також не потрібно зберігати конфіденційні дані сеансу та користувача у сховищі браузера.

Кожного разу, коли користувач хоче отримати доступ до захищених даних на сервері, клієнт повинен надсилати JWT, найчастіше він передається в заголовок запиту.

РОЗДІЛ 3

ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

3.1. Обґрунтування алгоритму й структури програми

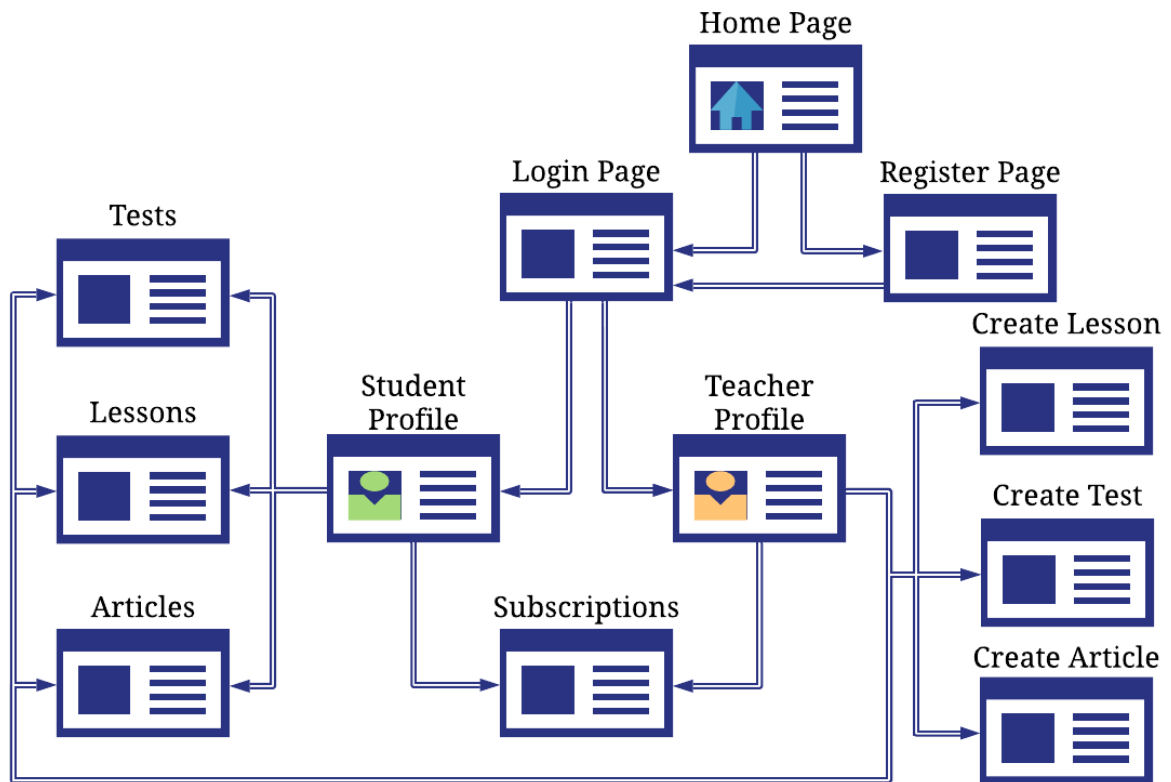


Рисунок 4 - Діаграма структури програми

UML-діаграма(див. рис. 4) демонструє структуру програми та алгоритм її роботи. Виходячи з цього можемо розбити проект на такі модулі:

- Аутентифікація;
- Профіль студента;
- Профіль викладача;
- Урок;
- Стаття;
- Тест;
- Підписки.

Для взаємодії між модулями на клієнті використовуються Angular сервіси , вони потрібні коли дані, що передаються, не прив'язані до якогось певного компонента, а взаємодіють з різними.

3.2. Обґрунтування вибору засобів розробки

Для реалізації клієнтської частини використано фреймворк Angular [10]. Він використовує архітектуру MVC, яка не тільки надає цінність фреймворку при створенні клієнтського додатка, але і створює основу для інших функцій, таких як прив'язка даних і області видимості.

За допомогою архітектури MVC можна ізолювати логіку програми від рівня користувацького інтерфейсу. Контролер отримує всі запити для програми і працює з моделлю, щоб підготувати будь-які дані, необхідні для відображення. Відображення використовує дані, підготовлені контролером, і відображає остаточну відповідь.

Одним з ключових елементів Angular програми є компоненти. Компонент управляє відображенням подання на екрані. Це зроблено для того, щоб їх можна було повторно використовувати. Компоненти в свою чергу складаються в окремі модулі, які можуть взаємодіяти між собою. Нерідко модулі створюються для об'єднання компонентів та інших класів, які працюють з якимось одним аспектом програми. І якщо додаток великий, то правильно буде розбити його на модулі, які виконують різні завдання.

Більшість веб-додатків повинні взаємодіяти з сервером за протоколом HTTP, щоб завантажувати або надіслати дані та отримувати доступ до інших серверних служб. Angular надає спрощений клієнтський HTTP API для Angular-додатків, клас `HttpClient`. `HttpClient` підтримує всі методи HTTP.

Для реалізації серверної частини використана платформа Node.js, яка в свою чергу написана на мові JavaScript, також використовується фреймворк Express [9]. Він надає один з найпростіших, але в той же час і найпотужніших способів створення веб-сервера. Його мінімалістичний підхід зосереджений на основних функціях сервера є ключем до його успіху.

Для розробки веб-додатку була обрана СКБД PostgreSQL. Даний вибір аргументований тим, що PostgreSQL є некомерційною, гарно документованою та широко функціональною системою керування реляційними базами даних. PostgreSQL надає великий запас можливостей для росту і розвитку проекту в цілому та архітектури бази даних зокрема[8].

Серед використаних переваг обраної СКБД можна навести наступне:

- Підтримка найпопулярніших операційних систем (Windows, Mac, Linux)
- Високий рівень продуктивності (performance).
- Широкий набір стандартних типів.
- Наявний механізм підтримки цілісності посилань.
- Підтримка SQL sequence.
- Можливість створювати функції.
- Можливість створювати рекурсивні запити.
- Підтримка check constraint.

IntelliJ IDEA – мультиплатформне інтегроване середовище розробки для мов JVM, що надає можливість розробляти застосунки різних типів: консольні, веб, з графічним інтерфейсом та інші [7]. Також великим плюсом є можливість встановити плагіни для розробки, це дало можливість писати в одному середовищі: серверну і клієнтську частину, та працювати з СКБД. Також присутня інтеграція з системами управління версіями, включаючи Git.

Heroku – постачальник хмарних послуг та платформа для розробки програмного забезпечення, яка забезпечує швидке та ефективне створення, розгортання та масштабування веб-додатків. Цей інструмент є дуже простим у використанні. Крім того, вам не потрібно думати про інфраструктуру, оскільки вона автоматично керується самим програмним забезпеченням.

3.3. Опис розробки програми

Даний продукт має свою структуру(див. рис. 5) і складається з двох частин: клієнта та сервера.

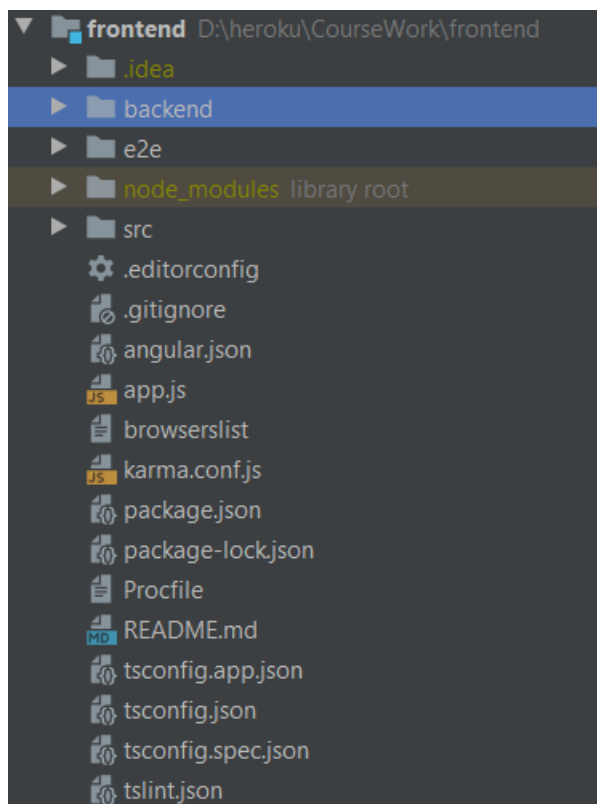


Рисунок 5 - Структура файлів

В ролі сервера тут виступає Node JS, а клієнтською частиною є Angular. Для запуску програми на віддаленому сервері Heroku присутній файл Procfile, що описує порядок дій для розгортання додатку.

Запуск клієнта відбувається завдяки збірці компонентів Angular, яка виконується за допомогою npm. Серверна частина запускається файлом app.js.

3.4. Опис файлів даних та інтерфейсу програми

Для розробки застосунку була використана база даних PostgreSQL. Щоб реалізувати поставлену задачу була розроблена реляційна модель даних(див. рис. 6).

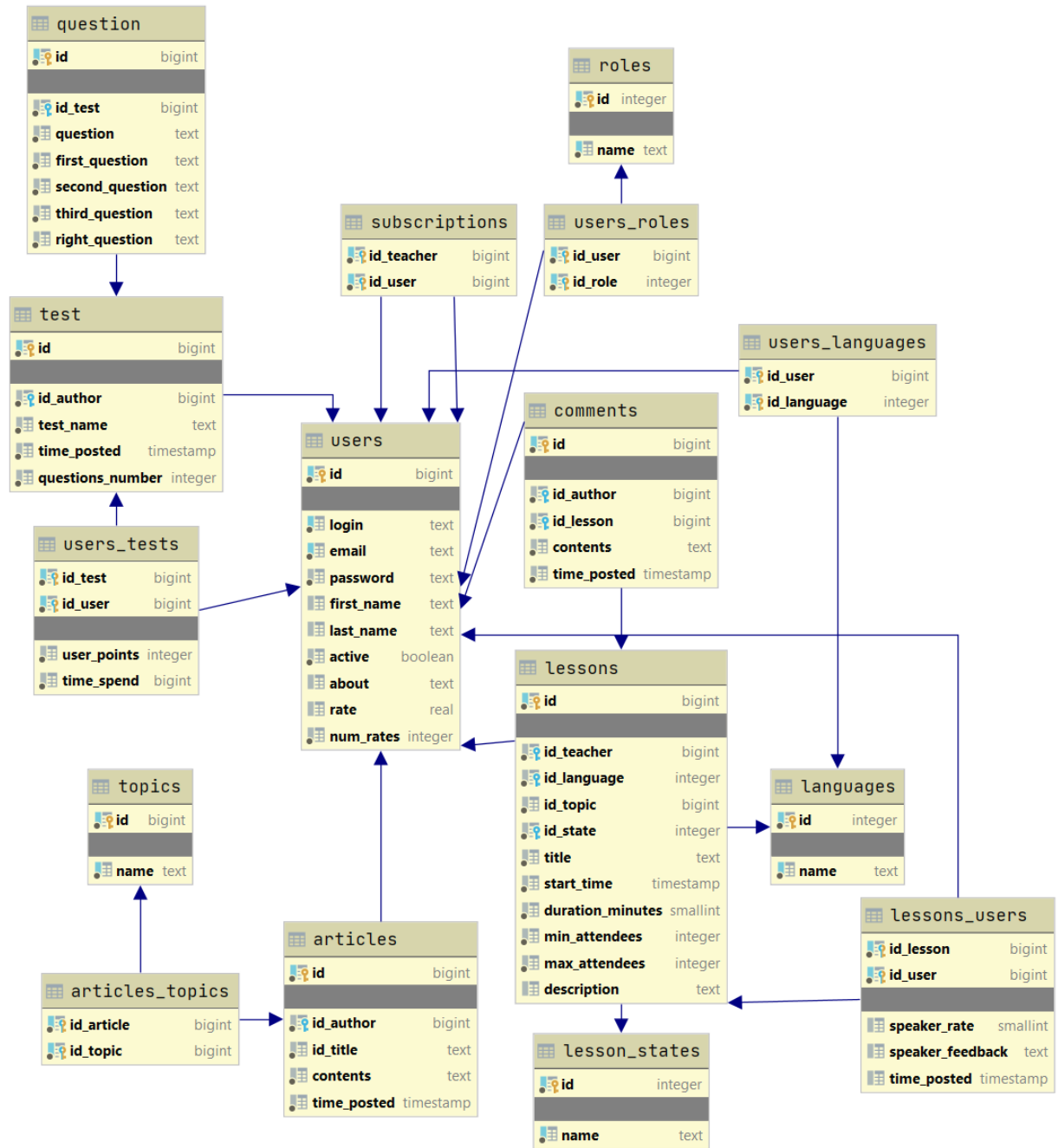


Рисунок 6 - Схема БД системи

Для кожної моделі даних були створені відповідні класи, що визначають їх. Нижче наведений приклад реалізації моделі User:

```
export class UserModel {
    public id: number;
    public role: string[];
    public first_name: string;
    public last_name: string;
    public login: string;
    public email: string;
    public rate: number;
    public num_rates: number;
    public about: string;
    public password: string;
    public languageIds: number[];
}
```

Для того, щоб звернутись до бази даних клієнт використовує вбудований в Angular модуль HttpClient. Звернення відбувається наступним чином:

```
this.userService.getUser().subscribe(res => {
    this.user = res[0];
    this.loading = false;
}, error => {
    this.error = error.error.message;
    this.loading = false;
});
```

```
getUser() {
    return this.httpClient.get<UserModel>('/api/user/profile')
}
```

В компоненті ми звертаємось до сервісу, що призначений для роботи з моделлю User, який в свою чергу посилає GET-запит на сервер для отримання даних про користувача. Код сервера який приймає запит, бере з заголовку токен, розшифровує його та дістає ID користувача і звертається до бази даних:


```
router.route('/api /user/profile')
.get((req, res) => {
  let token = req.header('x-access-token');
  let id = jwt.decode(token).id;
  pool.query(sql.find_user_with_id, [id], (err, result) => {
    if (err) res.status(500).send({message: 'Error on the server.'});
    res.status(200).json(result.rows)
  });
})
```

При успішному виконанні запиту отримані дані відправляються на клієнт у форматі JSON із кодом статусу 200. Якщо ж щось пішло не так користувач отримає повідомлення про помилку на сервері. Це був приклад того як ми отримуємо дані про користувача, давайте розглянемо створення користувача.

3.5. Тестування програми і результати її виконання

Веб-застосунок розміщено на платформі Нероки, а це означає, що для його запуску не потрібно локального розгортання проекту, можна просто перейти за покликанням <https://poliglout.herokuapp.com/>.

При переході на сайт ми потрапляємо на вікно входу в систему(див. рис. 7).

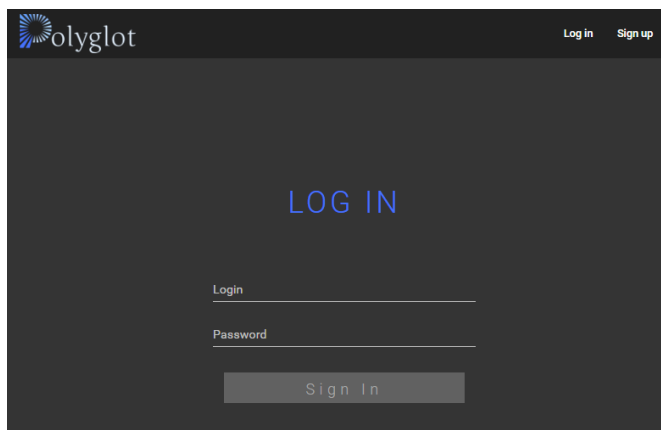
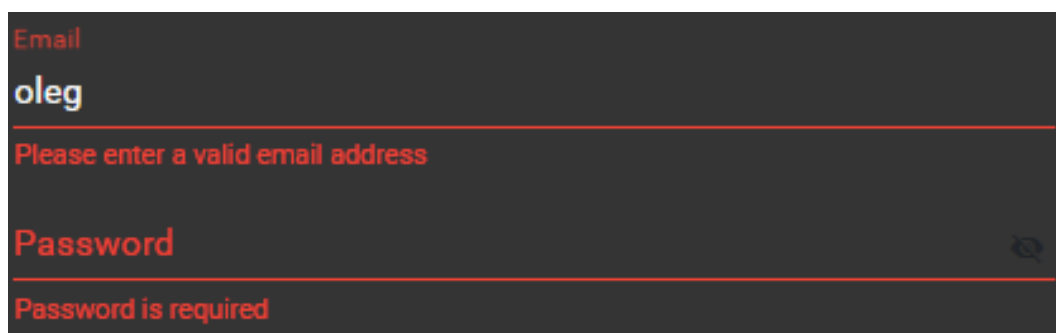


Рисунок 7 - Логін

Зареєструватись можна як ментор(викладач) та студент(див. рис. 8).

Рисунок 8 - Реєстрація

Перевірка на правильність вводу у всіх полях відбувається за допомогою валідатора реактивних форм Angular(див. рис. 9).



Email
oleg

Please enter a valid email address

Password

Password is required

Рисунок 9 - Валідація

Після авторизації користувач, в залежності від своєї ролі, відправляється на сторінку зі своїм профілем.

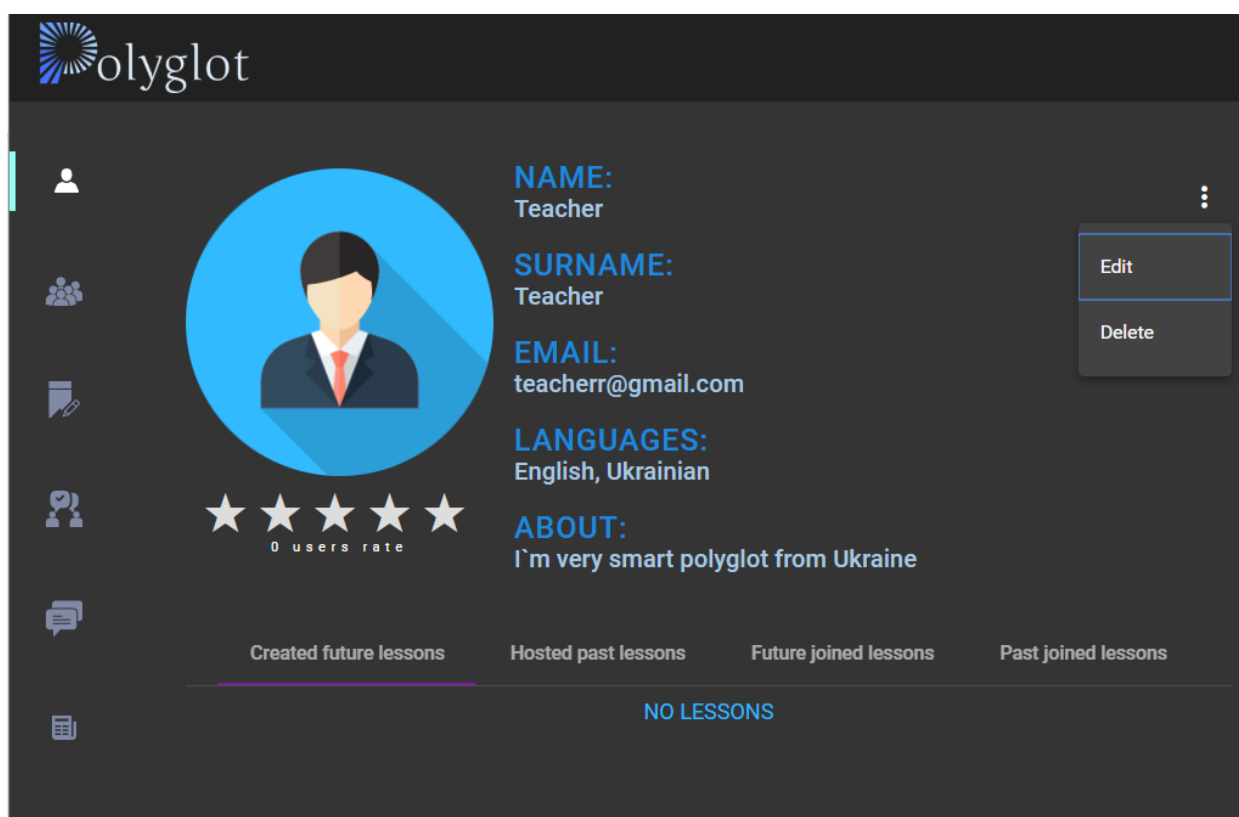
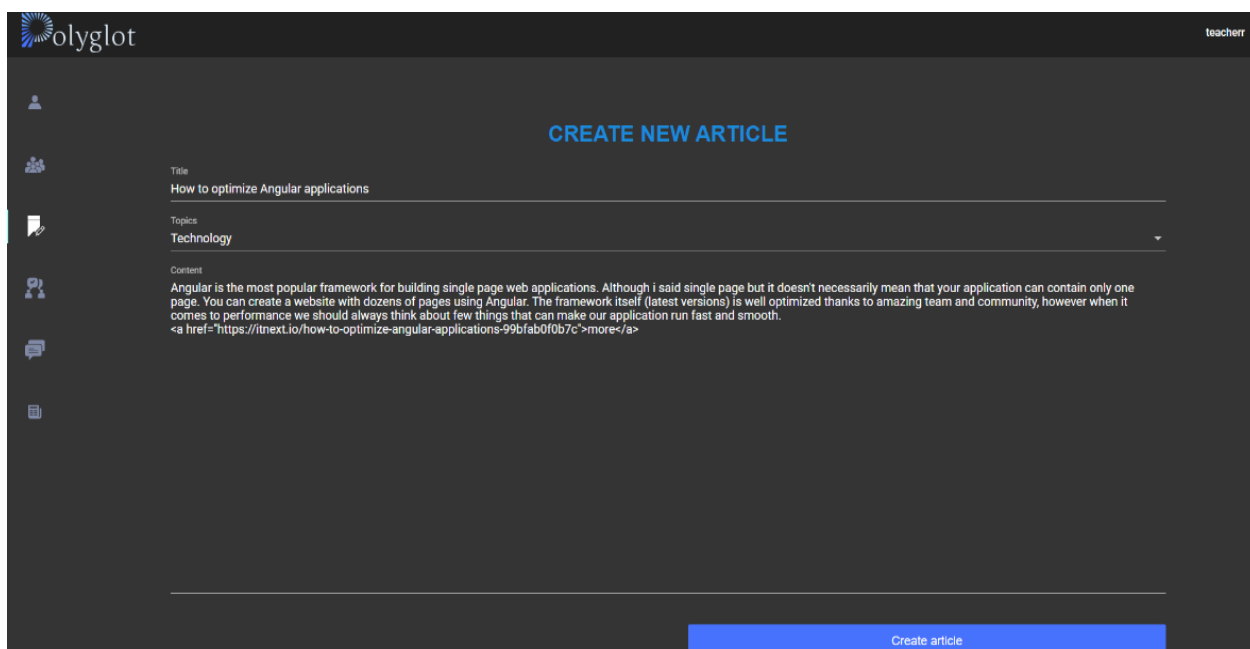


Рисунок 10 - Профіль викладача

Спочатку розберемо роль викладача(див. рис. 10). У викладача на сторінці ми можемо побачити всю інформацію про нього, також є можливість її відредагувати, або ж деактивувати профіль. В низу сторінки можна побачити всі його уроки, які мають відбутись та які вже відбулись та уроки на які він записаний, адже сам викладач також може виступати у ролі учня, якщо він хоче покращити для себе якусь мову. Далі користувач може перейти на такі сторінки як: підписки користувача, створення статті, усі статті, створення уроку, усі уроки.

Розглянемо детальніше створення статті(див. рис. 11).



olyglot teacher

CREATE NEW ARTICLE

Title
How to optimize Angular applications

Topics
Technology

Content
Angular is the most popular framework for building single page web applications. Although i said single page but it doesn't necessarily mean that your application can contain only one page. You can create a website with dozens of pages using Angular. The framework itself (latest versions) is well optimized thanks to amazing team and community, however when it comes to performance we should always think about few things that can make our application run fast and smooth.
more

Create article

Рисунок 11 - Створення статті

При створенні статей ми можемо використовувати HTML-атрибути для оформлення відображення статті. Також можна вибрати декілька тем якщо потрібно. Після створення ми потрапляємо на сторінку з усіма статтями(див. рис. 12).

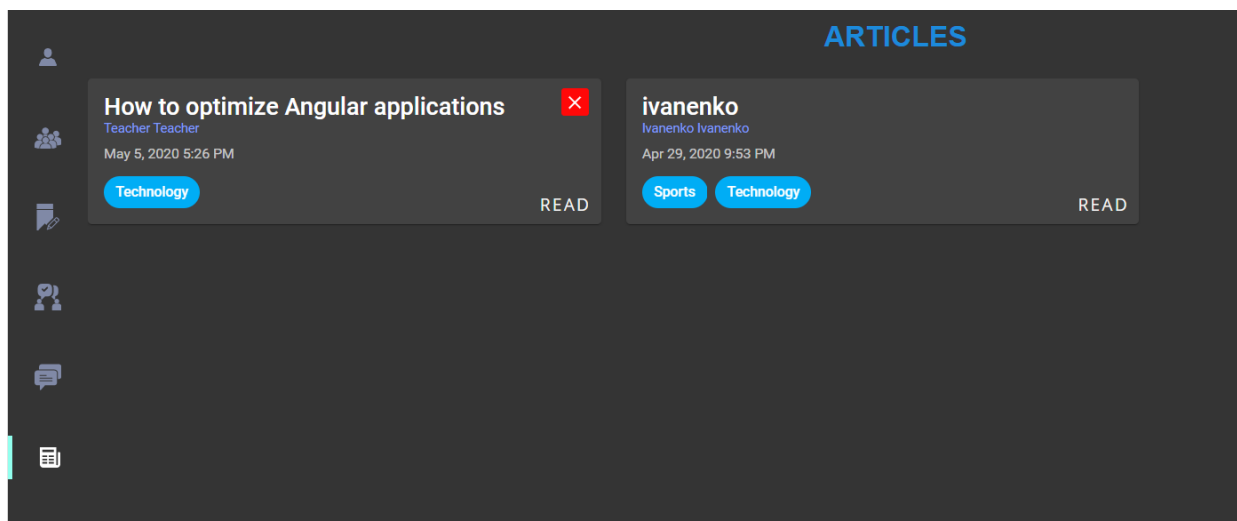


Рисунок 12 - Усі статті

Тут і викладач, і студент можуть побачити всі статті і перейти за посиланням і прочитати всю статтю(див. рис. 13), але викладач також має можливість видалити статті, які він написав.

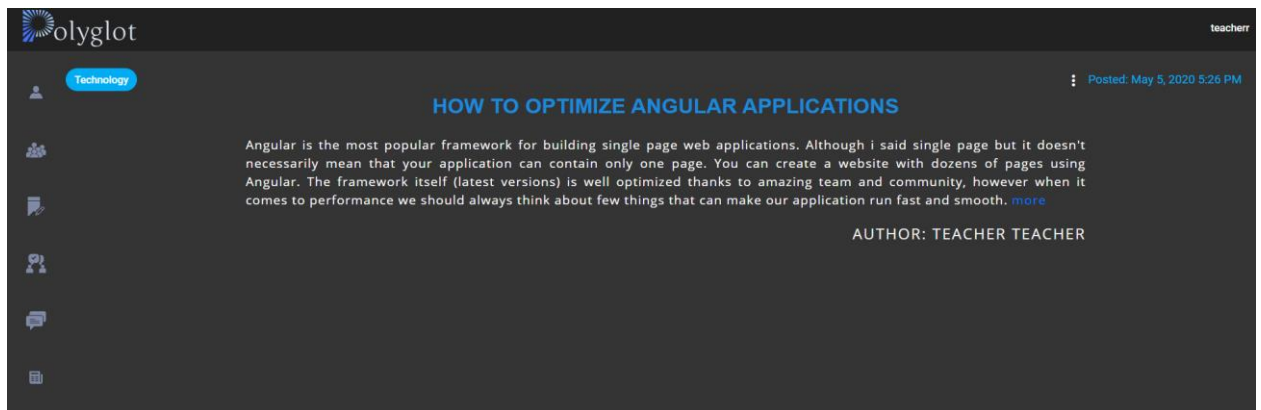


Рисунок 13 - Стаття

Перейдемо на сторінку створення уроку(див. рис. 14).

Рисунок 14 - Створення уроку

Тут як і у всіх формах на сайті відбувається валідація: мінімальна кількість учнів не може бути меншою за максимальну, дата не може бути минулою, всі поля обов'язкові, також викладач може обрати мову тільки ту, якою він володіє.

Після створення уроку ми переходимо на сторінку з усіма уроками(див. рис. 15).

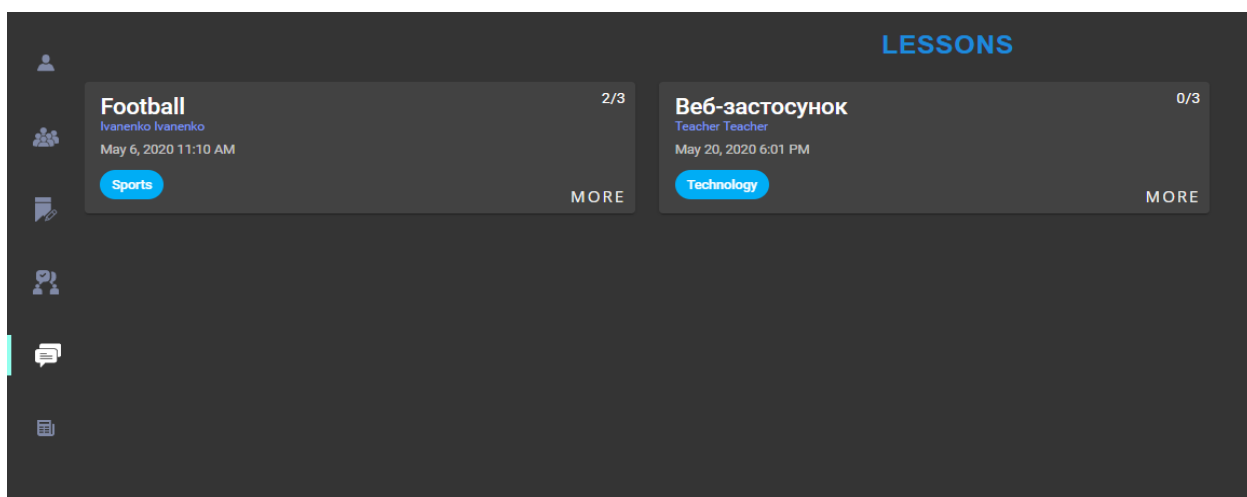


Рисунок 15 - Усі уроки

У верхньому правому кутку уроку ми можемо побачити скільки користувачів зареєструвалось на цей урок з максимальної кількості, а під автором вказана дата коли повинен відбутись урок. Для більш детального ознайомлення з уроком переходимо на сторінку уроку(див. рис. 16).

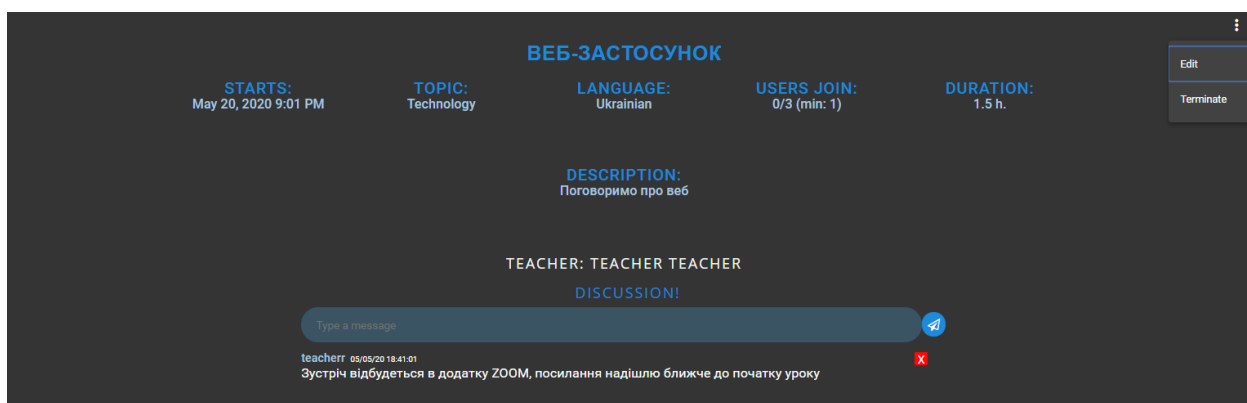


Рисунок 16 - Урок

Тут ми можемо побачити детальнішу інформацію про урок, викладач який створив його може відредагувати, або ж відмінити урок. Також для користувачів які приєдналися до уроку доступна функція коментування, в якій вони і автор можуть уточнювати будь-яку інформацію, або ж висловити свої побажання та пропозиції.

При переході на сторінку іншого викладача(див. рис. 17) користувачі мають можливість підписатись на нього, для швидкого доступу та перегляду уроків цього викладача.

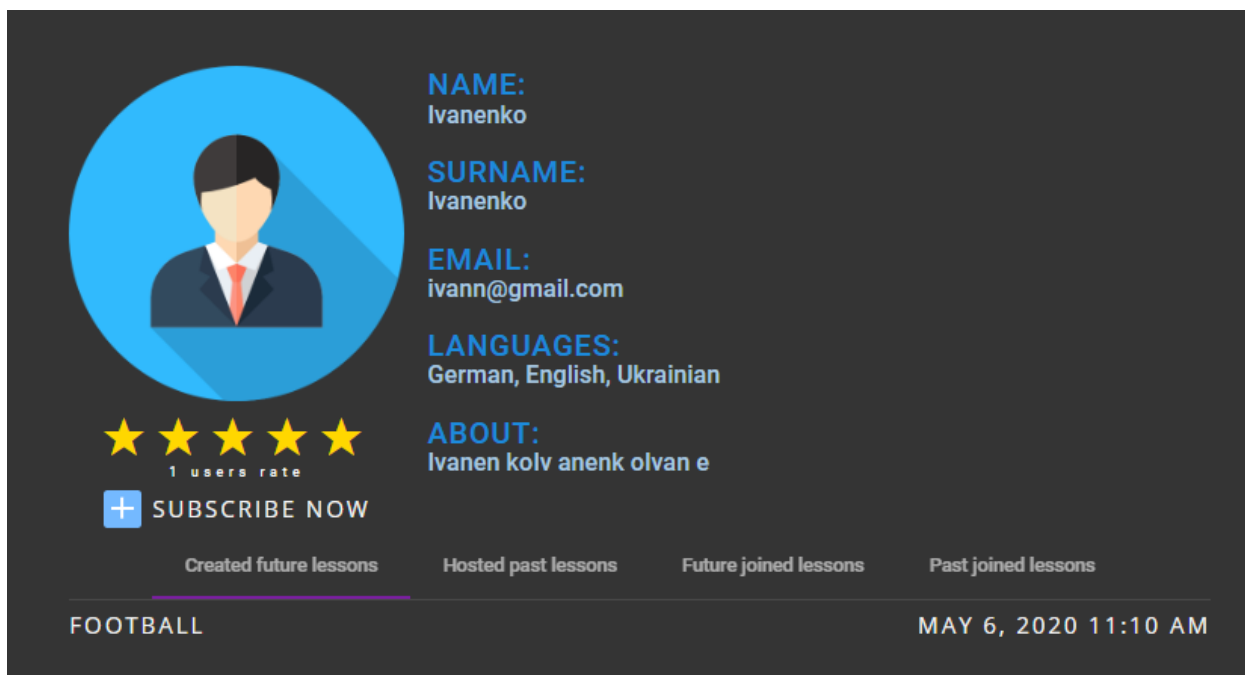


Рисунок 17 - Чужий профіль викладача

А ось як виглядає сторінка з підписками після того коли ми виконали цю дію(див. рис. 18).

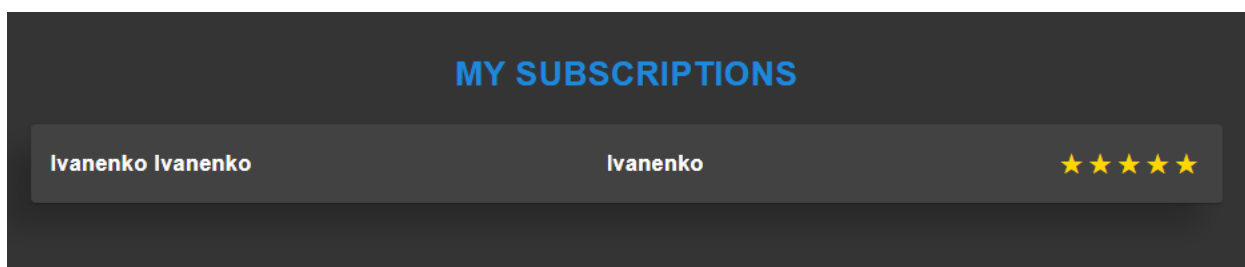


Рисунок 18 - Підписки

Тут відображені прізвище і ім'я викладача, його логін та рейтинг.

Одним з ключових елементів навчального процесу є фіксація та контроль рівня знань. Для цього завдання в застосунку передбачені тести. Створити їх може викладач(див. рис. 19). Перед ним є форма у яку вписується саме питання, три неправильні відповіді і одна правильна, кількість питань тесту викладач визначає на свій розсуд. Після того коли питання додані тест можна перевірити, подивитись чи всі данні введено вірно, є можливість відредагувати або ж видалити певне питання. Після перевірки викладач дає назву тесту і створює його.

Рисунок 19 - Створення тесту

Коли користувач потрапляє на сторінку тесту(див. рис. 20), він бачить топ 5 користувачів(якщо кількість пройдених тест менша 5 то відображаються всі), що пройшли цей тест, кнопку початку та автора тесту.

Name	Points	Time spend
Student Studentivich	7	01 m.

Рисунок 20 - Тест

Після початку тесту користувач потрапляє на вікно з усіма запитаннями(див. рис. 21). Він може вільно переходити між питаннями, та редагувати відповіді, для того щоб завершити тест треба відповісти на всі запитання. Питання та варіанти відповідей при кожному запуску тесту рандомно перемішуються. Скопіювати саме питання чи варіант відповіді не можна, це досягається за допомогою SCSS стилізації форми, тому навіть вимкнувши Java Script у браузері користувач все ще не матиме можливості для спроби шахрайства.

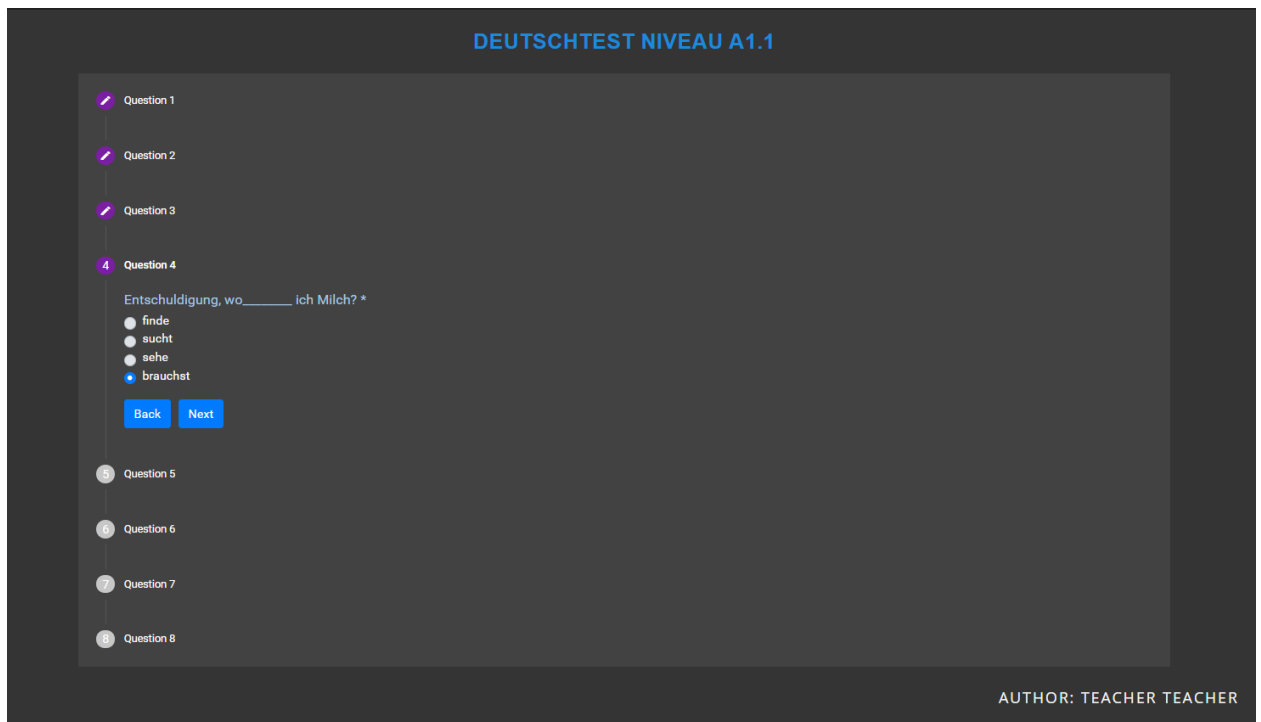


Рисунок 21 - проходження тесту

Після проходження тесту користувач має можливість побачити свої результати: кількість правильних відповідей та час за який він здійснив проходження тесту(див. рис. 22). Також відображаються всі питання з правильними відповідями і помилками(якщо такі є).

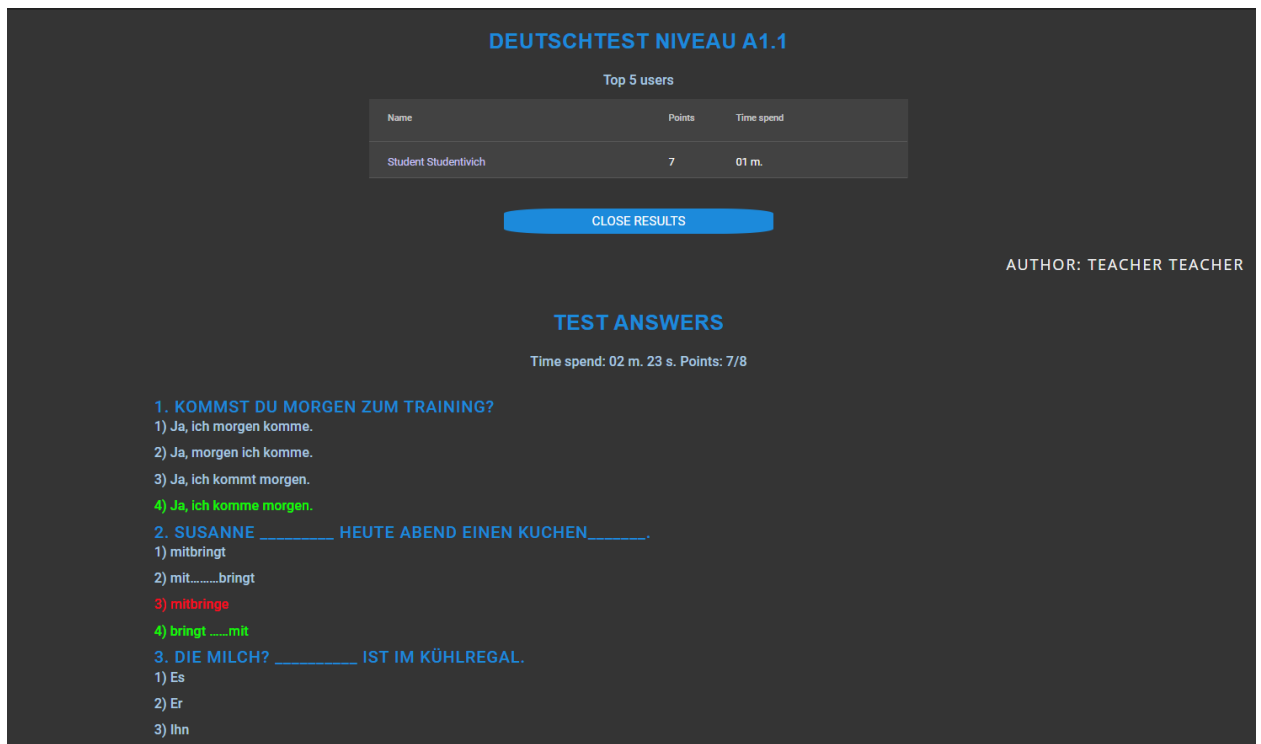


Рисунок 22 - Завершення тесту

Тепер перейдемо до профілю студента(див. рис. 23).

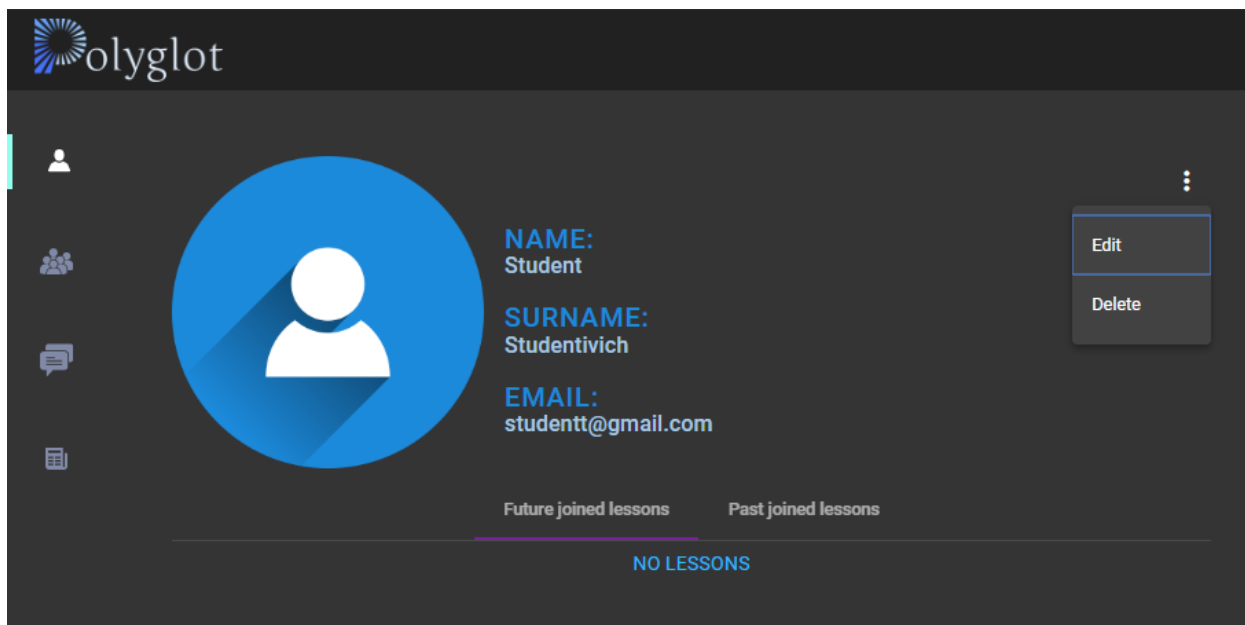


Рисунок 23 - Профіль студента

Він може переглянути уроки на які записаний та які відвідав, також є можливість редагувати дані та деактивувати профіль. Можливості студента є підмножиною можливостей викладача, тому він може тільки читати статті, підписуватись на викладачів, проходити тести, приєднуватись до уроків, та залишати коментарі.

ВИСНОВКИ

Отже, мета проекту досягнута, веб-застосунок відповідає всім тим вимогам, що були описані у завданні. Під час виконання цієї роботи я детально познайомився з архітектурою програмування REST, закріпив знання в розробці Angular застосунків та навчився викладати готовий продукт на хмарне середовище Heroku.

Цей застосунок слід використовувати людям, які мають бажання покращити своє володіння іноземних мов або ж які хочуть поспілкуватись на цікаві для них теми, при цьому допомагаючи іншим освоїти свою мову.

Надалі можна розширити можливості системи додавши можливість надсилати сповіщення про уроки які скоро відбудуться, скаржитись на недобросовісних користувачів, медальки за різного типу досягнення та багато іншого.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Дослідження Обернського університету:
<https://cla.auburn.edu/forlang/resources/twenty-five-reasons/>
2. Polyglot Club: <https://polyglotclub.com/>
3. HiNative: <https://hinative.com/>
4. Lang-8: <https://lang-8.com/>
5. REST API: <https://restfulapi.net/>
6. Cookies:
<https://us.norton.com/internetsecurity-how-to-what-are-cookies.html>
7. IntelliJ IDEA: <https://www.jetbrains.com/idea/>
8. PostgreSQL: <https://www.postgresql.org/>
9. Node JS: <https://nodejs.org/uk/docs/>
10. Angular: <https://habr.com/en/post/348818/#s1>

ДОДАТОК А

Код клієнта – профіль викладача

```
import {Component, OnInit} from '@angular/core';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import {UserModel} from '../models/user.model';
import {UserService} from '../services/user.service';
import {ActivatedRoute, ParamMap, Router} from '@angular/router';
import {LanguagesList} from '../models/languagesList';
import {DialogComponent} from '../dialog/dialog.component';
import {MatDialog} from '@angular/material/dialog';
import * as jwt_decode from 'jwt-decode';
import {LessonModel} from '../models/lesson.model';
```

```
@Component({
  selector: 'app-profile-teacher',
  templateUrl: './profile-teacher.component.html',
  styleUrls: ['./profile-teacher.component.scss']
})
export class ProfileTeacherComponent implements OnInit {
  public changeForm: FormGroup;
  public loading = true;
  public loadingSubs = false;
  public edited = true;
  public user: UserModel;
  public languages: LanguagesList [];
  public langListNames = "";
  public langListIds: number[] = [];
  public star: number;
  public isMyProfile: boolean;
  public userId: string;
  public subscribe = true;
  public subscribeToIdsList: number[] = [];
  public subscribeText: string;
  future_hosted_lessons: LessonModel[] = [];
  future_joined_lessons: LessonModel[] = [];
  past_hosted_lessons: LessonModel[] = [];
  past_joined_lessons: LessonModel[] = [];

  constructor(public dialog: MatDialog,
    private router: Router,
    private formBuilder: FormBuilder,
    private userService: UserService,
    public route: ActivatedRoute) {
  }
}
```

```
ngOnInit() {
  this.loading = true;
  this.loadingSubs = false;
  this.langListIds = [];
  this.langListNames = "";
  this.edited = true;
  this.changeForm = this.formBuilder.group({
    first_name: ['', Validators.required],
    last_name: ['', Validators.required],
    languages: ['', Validators.required],
    about: [''],
    email: ['', [Validators.required, Validators.email]],
  });
}
```

```

this.route.paramMap.subscribe((paramMap: ParamMap) => {
  if (paramMap.has('userId')) {
    this.isMyProfile = false;
    this.userId = paramMap.get('userId');
    this.userService.getUserById(+this.userId).subscribe(res => {
      this.user = res[0];
      this.star = this.user.rate;
      this.userService.getUserLanguagesById(+this.userId).subscribe(lang => {
        for (const i in lang) {
          this.langListNames += lang[i].name + ' ';
          this.langListIds[i] = lang[i].id;
        }
        this.langListNames = this.langListNames.substring(0, this.langListNames.length - 2);
        this.userService.getLanguages().subscribe(
          allLan => {
            this.languages = allLan;
            this.userService.getTeacherSubscribersById(+this.userId).subscribe(sub => {
              for (const i in sub) {
                this.subscribeToIdsList[i] = sub[i].id;
              }
              const userID = jwt_decode(localStorage.getItem('token')).id;
              if (this.userId === userID) {
                this.router.navigate(['/teacher-profile']);
              }
              if (this.subscribeToIdsList.includes(userID)) {
                this.subscribeText = 'UNSUBSCRIBE';
                this.subscribe = false;
              } else {
                this.subscribeText = 'Subscribe Now';
                this.subscribe = true;
              }
              this.userService.getTeacherLessonsById(+this.userId).subscribe(res=>{
                console.log(res);
                this.future_hosted_lessons = res['future_hosted_lessons'];
                this.future_joined_lessons = res['future_joined_lessons'];
                this.past_hosted_lessons = res['past_hosted_lessons'];
                this.past_joined_lessons = res['past_joined_lessons'];
                this.loading = false;
              });
            });
          });
        });
      });
    } else {
      this.isMyProfile = true;
      this.userService.getUser().subscribe(res => {
        this.user = res[0];
        this.star = this.user.rate;
        this.userService.getUserLanguages().subscribe(lang => {
          for (const i in lang) {
            this.langListNames += lang[i].name + ' ';
            this.langListIds[i] = lang[i].id;
          }
          this.langListNames = this.langListNames.substring(0, this.langListNames.length - 2);
          this.userService.getLanguages().subscribe(
            allLan => {
              this.languages = allLan;
              this.userService.getTeacherLessons().subscribe(res=>{
                console.log(res);
                this.future_hosted_lessons = res['future_hosted_lessons'];
                this.future_joined_lessons = res['future_joined_lessons'];

```

```

        this.past_hosted_lessons = res['past_hosted_lessons'];
        this.past_joined_lessons = res['past_joined_lessons'];
        this.loading = false;
    });
    });
    });
    });
    }
    });
}

onEdit() {
    this.edited = false;
    this.changeForm.controls.first_name.setValue(this.user.first_name);
    this.changeForm.controls.last_name.setValue(this.user.last_name);
    this.changeForm.controls.email.setValue(this.user.email);
    this.changeForm.controls.about.setValue(this.user.about);
    this.changeForm.controls.languages.setValue(this.langListIds);
}

onCancel() {
    this.edited = true;
}

onSave() {
    const user = {
        first_name: this.changeForm.get('first_name').value,
        last_name: this.changeForm.get('last_name').value,
        email: this.changeForm.get('email').value,
        about: this.changeForm.get('about').value,
        languageIds: this.changeForm.get('languages').value,
        id: this.user.id
    } as UserModel;
    this.loading = true;
    this.userService.updateUser(user).subscribe(res => {
        this.ngOnInit();
    }, error => {
        this.loading = false;
    });
}

openDialog(): void {
    const dialogRef = this.dialog.open(DialogComponent, {
        width: '320px',
        height: '200px',
        data: {id: this.user.id, first_name: this.user.first_name, last_name: this.user.last_name}
    });
    dialogRef.afterClosed().subscribe(result => {
        console.log('The dialog was closed');
    });
}

Subscribe() {
    if (this.subscribe && !this.loadingSubs) {
        this.loadingSubs = true;
        this.userService.subscribeTo(+this.userId).subscribe(res => {
            this.subscribe = false;
            this.subscribeText = 'UNSUBSCRIBE';
            this.loadingSubs = false;
        });
    }
}

```

```
if (!this.subscribe && !this.loadingSubs) {  
  this.loadingSubs = true;  
  this.userService.unsubscribeTo(+this.userId).subscribe(res => {  
    this.subscribe = true;  
    this.subscribeText = 'Subscribe Now';  
    this.loadingSubs = false;  
  });  
}  
}  
}
```


ДОДАТОК Б

Код сервера – налаштування сервера і підключення до бази даних

```
const {Client} = require('pg');
const csrf = require('csrf');
const cookieParser = require('cookie-parser');
let path = require('path');
const express = require('express'),
    app = express(),
    port = parseInt(process.env.PORT, 10) || 3000;

let UsersController = require('./backend/handlers/UsersController.js');
let LanguagesController = require('./backend/handlers/LanguagesController.js');
let AuthController = require('./backend/auth/AuthController.js');
app.use(express.json());
app.use(express.urlencoded({extended: true}));
app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Methods", "POST, PUT, GET, OPTIONS, DELETE");
    res.header("Access-Control-Allow-Headers", "x-access-token, Origin, X-Requested-With, Content-Type, Accept");
    next();
});
app.use(cookieParser());
app.use('/api', UsersController);
app.use('/api', AuthController);
app.use('/api', LanguagesController);

// Create link to Angular build directory
app.use(express.static(__dirname + '/dist/frontend'));

app.get('/*', function(req,res) {
    res.sendFile(path.join(__dirname+' /dist/frontend/index.html'));
});

app.listen(port, () => {
    console.log('Server started on port 3000');
});

const client = new Client({
    connectionString: 'postgres://user-name:password@host:port/db-name',
    ssl: true,
});

client.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
});
```