**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп’ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

**Курсова робота**

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

**Розробка освітнього порталу**

| Виконала студентка 3-го курсу  Анастасія БОРОВИК | (підпис) |
| --- | --- |
| Науковий керівник:  асистент кафедри інтелектуальних програмних систем, кандидат фіз.-мат. наук  Костянтин ЖЕРЕБ | (підпис) |

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

| Студент | (підпис) |
| --- | --- |

**Вступ**

У сучасному світі цифрові технології відіграють ключову роль у сфері освіти. Створення освітнього порталу є важливим кроком для забезпечення доступу до якісних навчальних матеріалів, організації навчального процесу та інтерактивної взаємодії між студентами й викладачами.

Освітній портал — це веб-платформа, яка об'єднує різні ресурси для навчання, включаючи онлайн-курси, тестування. Його розробка передбачає аналіз потреб користувачів, розробку функціоналу, зручного інтерфейсу та інтеграцію з іншими освітніми сервісами.

Мета даної курсової роботи — розробити концепцію та технічне рішення для освітнього порталу, яке забезпечить ефективний доступ до знань, підвищить рівень інтерактивності та сприятиме персоналізованому навчанню.

## **Розділ 3.2.1. Контролер користувачів** (userController.js)

Контролер userController.js відповідає за обробку запитів, пов'язаних з автентифікацією та авторизацією користувачів освітнього порталу. Він надає функціонал для реєстрації нових користувачів, входу в систему та перевірки автентифікації.

### **3.2.1.1. Залежності та допоміжні функції**

**bcrypt**: Бібліотека для хешування паролів. Використовується для безпечного зберігання паролів користувачів у базі даних, унеможливлюючи прямий доступ до них навіть у випадку несанкціонованого доступу до бази даних.

**jsonwebtoken (JWT)**: Бібліотека для створення та перевірки JSON Web Tokens. JWT використовуються для реалізації безстанової автентифікації, дозволяючи серверу перевіряти автентичність запитів без збереження інформації про сесію на сервері.

**ApiError**: Власний модуль для обробки помилок, який дозволяє стандартизувати відповіді сервера у випадку виникнення помилок (наприклад, некоректні дані, відсутність користувача тощо).

**Моделі (User, Personal\_information, Leaderboard, Usertask, Task)**: Імпортуються з ../models/models. Ці моделі представляють структури даних, що зберігаються в базі даних, і використовуються для взаємодії з таблицями бази даних (користувачі, особиста інформація, таблиця лідерів, завдання користувачів, завдання).

**generateJwt**: Допоміжна функція, яка створює JWT токен. Вона приймає id користувача, nickname та email, підписує їх за допомогою секретного ключа (process.env.SECRET\_KEY) та встановлює термін дії токена (24 години). Цей токен потім надсилається клієнту для подальшої автентифікації.

### **3.2.1.2. Основні функції контролера**

#### **3.2.1.2.1. Реєстрація користувача (registration)**

Метод registration відповідає за створення нового облікового запису користувача на освітньому порталі.

**Валідація вхідних даних**: Перевіряє наявність nickname, email та password у запиті. Якщо будь-яке з цих полів відсутнє, повертається помилка ApiError.badRequest.

**Перевірка унікальності**:

Шукає існуючого користувача за nickname. Якщо такий користувач вже існує, повертається помилка "Имя пользователя уже занято".

Шукає існуючого користувача за email. Якщо такий email вже зареєстрований, повертається помилка "Почта уже занята".

**Хешування пароля**: Пароль, отриманий від користувача, хешується за допомогою bcrypt.hash з "сіллю" (salt) розміром 5. Це забезпечує безпечне зберігання пароля.

**Створення записів у базі даних**:

Створюється новий запис у таблиці User з хешованим паролем.

Для новоствореного користувача автоматично створюється запис у таблиці Personal\_information (для зберігання особистих даних) та Leaderboard (з початковим рейтингом 0).

**Ініціалізація завдань**: Отримуюються всі доступні завдання таблиці Task. Для кожного з цих завдань створюється запис у таблиці Usertask для нового користувача зі статусом 'не приступал'. Це гарантує, що кожен новий користувач одразу бачитиме всі доступні завдання та їх початковий статус.

**Генерація JWT**: Після успішної реєстрації генерується JWT токен для нового користувача, який містить його id, nickname та email.

**Відповідь клієнту**: Токен надсилається клієнту у форматі JSON.

#### **3.2.1.2.2. Вхід користувача (login)**

Метод login обробляє запити на вхід зареєстрованих користувачів.

**Отримання даних**: З тіла запиту отримуються email та password.

**Пошук користувача**: Шукає користувача в базі даних за наданим email. Якщо користувач не знайдений, повертається помилка ApiError.internal "Пользователь не найден".

**Порівняння паролів**: Використовує bcrypt.compareSync для порівняння наданого пароля з хешованим паролем, що зберігається в базі даних. Якщо паролі не збігаються, повертається помилка ApiError.internal "Указан неверный пароль".

**Генерація JWT**: У випадку успішного входу генерується новий JWT токен для користувача.

**Відповідь клієнту**: Токен надсилається клієнту.

#### **3.2.1.2.3. Перевірка автентифікації (check)**

Метод check використовується для перевірки валідності токена та оновлення його терміну дії.

**Генерація нового JWT**: Цей метод просто генерує новий JWT токен, використовуючи інформацію про користувача, яка вже міститься в об'єкті req.user (після успішної перевірки проміжним ПО автентифікації). Це дозволяє "оновлювати" термін дії токена, щоб користувачу не доводилося повторно входити в систему після закінчення терміну дії його поточного токена, доки він активно користується системою.

**Відповідь клієнту**: Новий токен надсилається клієнту.

## **Розділ 3.2.2. Контролер особистої інформації (personal\_informationController.js)**

Контролер *personal\_informationController.js* призначений для керування особистою інформацією користувачів освітнього порталу. Він надає функціонал для зміни та перегляду профілю користувача.

### **3.2.2.1. Залежності**

**Моделі (*Personal\_information, User*)**: Імпортуються з *../models/models.* *Personal\_information* представляє таблицю бази даних, де зберігаються детальні особисті дані користувачів.

***ApiError****:* Власний модуль для обробки помилок, що забезпечує стандартизовані відповіді у випадку некоректних запитів.

### **3.2.2.2. Основні функції контролера**

#### **3.2.2.2.1. Зміна інформації профілю (*change\_info*)**

Метод change\_info дозволяє користувачам оновлювати свої особисті дані.

**Отримання ідентифікатора користувача**: userid отримується з об'єкта req.user, який містить інформацію про автентифікованого користувача (зазвичай витягнуту з JWT токена проміжним ПО).

**Отримання даних з тіла запиту**: З тіла запиту (req.body) витягуються такі поля, як firstname, surname, middlename, placeOfStudy, phoneNumber.

**Пошук існуючого запису**: Метод шукає існуючий запис особистої інформації, пов'язаний з userid.

**Оновлення або створення запису**:

Якщо запис знайдено (if (user)), то його дані оновлюються за допомогою Personal\_information.update(). Після оновлення, запис повторно вибирається з бази даних, щоб повернути клієнту найактуальніші дані.

Якщо запис не знайдено (тобто, для цього користувача ще не було створено запис особистої інформації, хоча userController вже створює його при реєстрації, це може бути резервний варіант або для старих записів), створюється новий запис Personal\_information.create() з наданими даними.

**Відповідь клієнту**: Оновлений або новостворений об'єкт Personal\_information надсилається клієнту у форматі JSON.

#### **3.2.2.2.2. Перегляд інформації профілю (view\_lk)**

Метод view\_lk (імовірно, "view personal cabinet" - "перегляд особистого кабінету") дозволяє користувачам переглядати свою особисту інформацію.

**Отримання ідентифікатора користувача**: Аналогічно change\_info, userid отримується з req.user.

**Пошук інформації**: Здійснюється пошук запису в таблиці Personal\_information за UserId.

**Вибірка атрибутів**: Вибираються лише конкретні атрибути (firstname, surname, middlename, placeOfStudy, phoneNumber), що забезпечує конфіденційність та зменшує обсяг переданих даних.

**Відповідь клієнту**: Вибрані дані особистої інформації надсилаються клієнту у форматі JSON.

## **Розділ 3.2.3. Контролер таблиці лідерів (leaderboardController.js)**

Контролер leaderboardController.js відповідає за отримання та відображення даних з таблиці лідерів освітнього порталу. Він надає функціонал для відображення рейтингу користувачів.

### **3.2.3.1. Залежності**

* **Моделі (User, Leaderboard)**: Імпортуються з ../models/models. Leaderboard зберігає інформацію про рейтинг користувачів, а User використовується для отримання імен користувачів, пов'язаних з рейтингом.

### **3.2.3.2. Основні функції контролера**

#### **3.2.3.2.1. Отримання лідерів (get\_leaders)**

Метод get\_leaders отримує список користувачів з найвищими рейтингами для відображення в таблиці лідерів.

* **Вибірка даних**: Використовує Leaderboard.findAll() для отримання записів з таблиці Leaderboard.
* **Вибір атрибутів**: Вибирається лише атрибут rating з моделі Leaderboard.
* **Включення пов'язаних даних**: За допомогою include пов'язується модель User, щоб отримати nickname відповідного користувача. Це дозволяє відображати ім'я користувача поруч з його рейтингом.
* **Сортування**: Результати сортуються за rating у порядку спадання ('DESC'), щоб користувачі з найвищим рейтингом були на початку списку.
* **Обмеження кількості результатів**: limit: 10 обмежує кількість повернених лідерів до 10, що є типовим для відображення "топ-10" таблиці лідерів.
* **Обробка помилок**: Використовується блок try-catch для перехоплення потенційних помилок під час взаємодії з базою даних, повертаючи відповідь 500 Internal Server Error у разі їх виникнення.
* **Відповідь клієнту**: Список лідерів надсилається клієнту у форматі JSON.

## 

## **Розділ 3.2.4. Контролер завдань (taskController.js)**

Контролер taskController.js керує логікою, пов'язаною з навчальними завданнями на освітньому порталі. Він надає функціонал для отримання всіх завдань, отримання деталей окремого завдання, а також для надсилання та перевірки відповідей користувачів на завдання.

### **3.2.4.1. Залежності**

* **axios**: Бібліотека для виконання HTTP-запитів. Ймовірно, використовується для взаємодії з зовнішнім сервісом для перевірки коду або виконання завдань (хоча в наданому фрагменті прямої інтеграції немає, згадка про axios на початку файлу натякає на таку можливість).
* **ApiError**: Власний модуль для обробки помилок.
* **Моделі (User, Leaderboard, Course, Task, Usertask)**: Імпортуються з ../models/models. Task представляє навчальні завдання, Usertask зберігає статус виконання завдань користувачами.
* **sequelize**: Об'єкт Sequelize, що використовується для виконання "сирих" SQL-запитів.

### **3.2.4.2. Основні функції контролера**

#### **3.2.4.2.1. Надсилання відповіді на завдання (giveAnswer)**

Метод giveAnswer обробляє надсилання користувачем рішення завдання.

* **Отримання ідентифікаторів**: Отримує USERID (з req.user), langId (ідентифікатор мови програмування) та taskId (ідентифікатор завдання) з параметрів запиту та тіла запиту.
* **Пошук завдання**: Знаходить завдання за taskId, вибираючи його title, text, rating та inputOutputId.
* **Перевірка наявності завдання**: Якщо завдання не знайдено, повертає 404 Not Found.
* **Отримання вхідних/вихідних даних для тестування**: Виконує "сирий" SQL-запит до бази даних, щоб отримати input та output дані для тестування завдання, використовуючи inputOutputId з моделі Task. Це дозволяє перевірити рішення користувача на відповідність очікуваним результатам.
* **Підготовка даних для зовнішнього сервісу**: stringInput та stringOutput конвертуються в рядки. ***Примітка:*** *Далі в коді видно логіку, яка, ймовірно, мала б відправляти ці дані разом з кодом користувача на зовнішній сервіс (наприклад, Judge0 або подібний) для виконання та перевірки. Однак, у наданому фрагменті коду axios не використовується для відправки запиту, і логіка перевірки рішення відсутня. Це вказує на незавершеність функціоналу або його винесення в інший модуль/шар.*
* **Оновлення статусу завдання**: Після (імовірно) успішної перевірки рішення, статус завдання користувача (Usertask) оновлюється на 'сдано' (здано).
* **Оновлення рейтингу користувача**: Рейтинг користувача в таблиці Leaderboard збільшується на task.rating.
* **Відповідь клієнту**: Повертає успішну відповідь з повідомленням "Answer accepted!".

#### **3.2.4.2.2. Отримання всіх завдань користувача (getAllUserTasks)**

Метод getAllUserTasks повертає список усіх завдань, пов'язаних з конкретним користувачем, включаючи їхній статус виконання.

* **Отримання ідентифікатора користувача**: USERID отримується з req.user.
* **Виконання SQL-запиту з JOIN**: Використовує "сирий" SQL-запит з LEFT JOIN між таблицями tasks та usertasks. Це дозволяє отримати назву завдання, його рейтинг та статус виконання для поточного користувача.
* **Відповідь клієнту**: Список завдань (з їхнім статусом) надсилається клієнту.

#### **3.2.4.2.3. Отримання одного завдання (getOneTask)**

Метод getOneTask надає деталі конкретного завдання.

* **Отримання ідентифікаторів**: Отримує USERID (з req.user) та taskId з параметрів запиту.
* **Пошук завдання**: Знаходить завдання за taskId, вибираючи його title, text та rating.
* **Перевірка наявності завдання**: Якщо завдання не знайдено, повертає 404 Not Found.
* **Відповідь клієнту**: Деталі завдання надсилаються клієнту у форматі JSON.

## **Розділ 3.2.5. Контролер користувачів курсу (user\_of\_courseController.js)**

Контролер user\_of\_courseController.js керує взаємодією користувачів з навчальними курсами на освітньому порталі. Він дозволяє користувачам записуватися на курси та перевіряти своє членство в них.

### **3.2.5.1. Залежності**

* **Моделі (User, Course, Users\_of\_course)**: Імпортуються з ../models/models. Users\_of\_course є допоміжною моделлю, яка реалізує зв'язок "багато-до-багатьох" між користувачами та курсами.
* **sequelize**: Об'єкт Sequelize для взаємодії з базою даних.

### **3.2.5.2. Основні функції контролера**

#### **3.2.5.2.1. Запис на курс (enter)**

Метод enter дозволяє користувачеві приєднатися до певного курсу.

* **Отримання ідентифікаторів**: Отримує UserId з req.user (ідентифікатор автентифікованого користувача) та CourseId з параметрів запиту.
* **Створення запису**: Створює новий запис у таблиці Users\_of\_course, пов'язуючи користувача з курсом.
* **Обробка помилок**: Перехоплює помилки, які можуть виникнути, наприклад, якщо користувач вже записаний на цей курс (якщо в базі даних налаштовано унікальний індекс на парі userId та courseId).
* **Відповідь клієнту**: Повертає повідомлення "You have entered course" у разі успішного запису.

#### **3.2.5.2.2. Перевірка членства в курсі (checkmemb)**

Метод checkmemb перевіряє, чи є користувач вже учасником певного курсу.

* **Отримання ідентифікаторів**: Отримує UserId з req.user та CourseId з параметрів запиту.
* **Пошук члена курсу**: Шукає запис у таблиці Users\_of\_course за наданими UserId та CourseId.
* **Отримання назви курсу**: Додатково отримує назву курсу (coursename) з моделі Course за CourseId. Це корисно для відображення інформації клієнту.
* **Визначення статусу**:
  + Якщо запис знайдено (if (student)), це означає, що користувач вже є учасником курсу. Повертається повідомлення "You already in course".
  + Якщо запис не знайдено (else), це означає, що користувач ще не записаний на курс. Повертається повідомлення "Ok", а також courseid та name курсу, що може бути використано для подальшого запису.
* **Обробка помилок**: Перехоплює потенційні помилки, що виникають під час взаємодії з базою даних.

## **Розділ 3.3. Структура та початкові дані бази даних (detttmi.sql)**

Файл detttmi.sql є сценарієм SQL, який використовується для ініціалізації або відновлення даних у базі даних освітнього порталу. Він містить SQL-запити INSERT для заповнення ключових таблиць системи початковими даними. Це дозволяє забезпечити наявність необхідних тестових або базових даних для функціонування застосунку відразу після розгортання або під час розробки.

### **3.3.1. Заголовки та налаштування SQL**

На початку файлу знаходяться стандартні SQL-коментарі та команди, які зазвичай використовуються в дампах баз даних для забезпечення коректної роботи сценарію в різних середовищах:

* /\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;
* /\*!40101 SET NAMES utf8 \*/;
* /\*!50503 SET NAMES utf8mb4 \*/;
* /\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;
* /\*!40103 SET TIME\_ZONE='+00:00' \*/;
* /\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;
* /\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;
* /\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

### **3.3.2. Дані для таблиці inputoutputs**

Цей розділ файлу містить дані для таблиці inputoutputs, яка, судячи з taskController.js, зберігає вхідні та очікувані вихідні дані для тестування рішень завдань.

* **Призначення**: Кожна запис у цій таблиці представляє набір тестових даних для конкретного завдання. input містить дані, які будуть подані на вхід програмі користувача, а output – очікуваний результат.
* **Приклади даних**:
  + id=1, input='2 5', output='5 2' – Ймовірно, для завдання, яке вимагає обміну значеннями або сортування двох чисел.
  + id=2, input='15 40', output='55' – Для завдання, яке може включати додавання чисел.
  + id=3, input='50 15', output='35' – Для завдання, яке може включати віднімання чисел.
* **Тимчасові мітки**: createdAt та updatedAt автоматично додаються ORM (як Sequelize) і фіксують час створення та останнього оновлення запису.

### **3.3.3. Дані для таблиці tasks**

Цей розділ містить початкові дані для таблиці tasks, яка зберігає інформацію про навчальні завдання.

* **Призначення**: Кожна запис у цій таблиці визначає окреме завдання на освітньому порталі.
* **Приклади даних**:
  + id=1, title='меньшее и большее', text='С клавиатуры вводятся два челых числа напишите какое большее, а какое меньшее', url='123', rating=10, inputoutputId=1
    - title: Назва завдання.
    - text: Детальний опис завдання.
    - url: Може бути посиланням на додаткові матеріали або приклад.
    - rating: Кількість балів, яку користувач отримує за успішне виконання завдання.
    - inputoutputId: Зовнішній ключ, що посилається на відповідний запис у таблиці inputoutputs, забезпечуючи зв'язок з тестовими даними для цього завдання.
  + id=2, title='Сумма', text='С клавиатуры вводятся два челых числа напишите программу, которая найдет их сумму', url='123', rating=15, inputoutputId=2
  + id=3, title='Разность', text='С клавиатуры вводятся два челых числа напишите программу, которая найдет их разность', url='123', rating=12, inputoutputId=3
* **Тимчасові мітки**: Аналогічно inputoutputs, присутні createdAt та updatedAt.

### **3.3.4. Значення та роль файлу detttmi.sql у проєкті**

Файл detttmi.sql відіграє критичну роль у розробці та розгортанні освітнього порталу, забезпечуючи:

1. **Початкові дані**: Надає необхідні дані для завдань та їх тестування, дозволяючи розробникам та тестувальникам працювати з функціональним набором даних без ручного введення.
2. **Відтворюваність**: Дозволяє легко відтворити стан бази даних, що важливо для розгортання на нових серверах, тестування або демонстрації.
3. **Демонстраційні дані**: Містить приклади завдань, які можуть бути використані для демонстрації функціоналу порталу.

## **Розділ 3.5. Маршрутизація (Роути) освітнього порталу**

Система маршрутизації в Node.js застосунку на базі Express.js відповідає за перенаправлення вхідних HTTP-запитів до відповідних обробників (контролерів). Ця організація дозволяє чітко розділити логіку обробки запитів за їх призначенням. Всі маршрути реєструються в головному файлі маршрутизації index.js, який агрегує маршрути з окремих файлів.

### **3.5.1. Головний файл маршрутизації (index.js)**

Файл index.js є центральним пунктом для збору та експортування всіх маршрутів застосунку. Він імпортує окремі роутери для різних функціональних модулів і об'єднує їх під єдиним префіксом або без нього.

**Залежності**: Імпортує об'єкт Router з Express.js та інші специфічні роутери: leaderboardRouter, personal\_informationRouter, user\_of\_courseRouter, userRouter, taskRouter.

**Використання маршрутів**:

* + router.use('/user', userRouter): Всі маршрути, визначені у userRouter, будуть доступні за префіксом /user (наприклад, /user/registration, /user/login).
  + router.use('/user', personal\_informationRouter): Маршрути для особистої інформації також групуються під /user, що логічно, оскільки особиста інформація є частиною профілю користувача (наприклад, /user/change\_personal, /user/lk).
  + router.use(leaderboardRouter): Маршрути для таблиці лідерів доступні без додаткового префікса (наприклад, /leaderboard).
  + router.use(user\_of\_courseRouter): Маршрути для курсів користувачів також доступні без додаткового префікса (наприклад, /:courseId/promo, /:courseId/enter).
  + router.use(taskRouter): Маршрути для завдань також доступні без додаткового префікса (наприклад, /:langId/:taskId/giveanswer, /tasks).

**Експорт**: module.exports = router експортує агрегований роутер, який потім може бути підключений до основного Express-застосунку.

### **3.5.2. Маршрути користувачів (userRouter.js)**

userRouter.js містить маршрути, пов'язані з автентифікацією та авторизацією користувачів.

**Залежності**: userController для обробки логіки та authMiddleware для захисту маршрутів.

**Визначені маршрути**:

router.post('/registration', userController.registration): Маршрут для реєстрації нового користувача. Приймає POST-запити.

router.post('/login', userController.login): Маршрут для входу користувача в систему. Приймає POST-запити.

router.get('/auth', authMiddleware, userController.check): Маршрут для перевірки автентифікації користувача. Вимагає валідного JWT токена (перевіряється authMiddleware). Приймає GET-запити.

### **3.5.3. Маршрути особистої інформації (personal\_informationRouter.js)**

personal\_informationRouter.js визначає маршрути для керування особистою інформацією користувача.

**Залежності**: personal\_informationController та authMiddleware.

**Визначені маршрути**:

* + router.post('/change\_personal', authMiddleware, personal\_informationController.change\_info): Маршрут для зміни особистої інформації. Захищений authMiddleware. Приймає POST-запити.
  + router.get('/lk', authMiddleware, personal\_informationController.view\_lk): Маршрут для перегляду особистого кабінету (профілю). Захищений authMiddleware. Приймає GET-запити.

### **3.5.4. Маршрути таблиці лідерів (leaderboardRouter.js)**

leaderboardRouter.js відповідає за маршрути, що надають доступ до таблиці лідерів.

**Залежності**: leaderboardController.

**Визначені маршрути**:

* + router.get('/leaderboard', leaderboardController.get\_leaders): Маршрут для отримання списку лідерів. Приймає GET-запити.

### **3.5.5. Маршрути завдань (taskRouter.js)**

taskRouter.js містить маршрути для взаємодії з навчальними завданнями.

* **Залежності**: taskController та authMiddleware.
* **Визначені маршрути**:

router.post('/:langId/:taskId/giveanswer', authMiddleware, taskController.giveAnswer): Маршрут для надсилання відповіді на конкретне завдання з вказанням мови та ідентифікатора завдання. Захищений authMiddleware. Приймає POST-запити.

router.get('/tasks', authMiddleware, taskController.getAllUserTasks): Маршрут для отримання всіх завдань, пов'язаних з поточним користувачем. Захищений authMiddleware. Приймає GET-запити.

router.get('/tasks/:taskId', authMiddleware, taskController.getOneTask): Маршрут для отримання деталей одного конкретного завдання за його ідентифікатором. Захищений authMiddleware. Приймає GET-запити.

router.post('/:langId/justtask', taskController.justtask): Присутній, але не описаний у наданому taskController.js. Ймовірно, це тестовий або допоміжний маршрут.

### **3.5.6. Маршрути користувачів курсу (user\_of\_courseRouter.js)**

user\_of\_courseRouter.js містить маршрути для керування членством користувачів у курсах.

**Залежності**: user\_of\_courseController та authMiddleware.

**Визначені маршрути**:

* + router.get('/:courseId/promo', authMiddleware, user\_of\_courseController.checkmemb): Маршрут для перевірки, чи є користувач вже учасником вказаного курсу. Захищений authMiddleware. Приймає GET-запити.
  + router.post('/:courseId/enter', authMiddleware, user\_of\_courseController.enter): Маршрут для запису користувача на вказаний курс. Захищений authMiddleware. Приймає POST-запити.

## **Розділ 3.6. Компоненти клієнтського застосунку (Frontend)**

Клієнтський застосунок освітнього порталу розроблений з використанням React.js, що дозволяє створювати інтерактивні та динамічні користувацькі інтерфейси. Розробка розділена на окремі компоненти, кожен з яких відповідає за певну частину інтерфейсу, що забезпечує модульність, перевикористання коду та легкість підтримки.

### **3.6.1. Компонент анімації (ani.jsx)**

Файл ani.jsx містить React-компонент Top2Component, який реалізує анімацію прояву елемента при прокрутці сторінки.

**Призначення**: Додає динамічний візуальний ефект, який робить інтерфейс більш привабливим. Цей компонент, ймовірно, використовується для поступового прояву контенту, коли користувач прокручує сторінку вниз.

**Механізм реалізації**:

Використовує хук useEffect для додавання та видалення обробника події scroll.

Функція revealTop2 перевіряє позицію елемента з класом top2 відносно вікна перегляду.

Якщо елемент стає видимим у вікні (його верхня межа менша за висоту вікна), до нього додається клас visible, що, імовірно, запускає CSS-анімацію прояву.

Після активації анімації обробник прокрутки видаляється, щоб уникнути зайвих обчислень.

**Взаємодія**: Компонент Top2Component рендерить div з класами top2 та hidden. CSS-стилі (імпортовані з ../styles) визначають початковий прихований стан (hidden) та анімацію при застосуванні класу visible.

### **3.6.2. Компонент бокової панелі/курсу (aside.jsx)**

Файл aside.jsx містить React-компонент AsideIntro, який, ймовірно, представляє бічну панель або розділ, що відображає інформацію про конкретний навчальний курс (наприклад, курс з Python).

**Призначення**: Надає користувачеві огляд курсу, його цільову аудиторію та програму. Включає кнопку для переходу на повний курс.

**Структура**:

Використовує Link з react-router-dom для навігації до сторінки відеокурсу.

Включає інші компоненти: Header (швидше за все, headerPython з імпорту) та Footer (footerPY).

Містить блоки з інформацією: "Что вас ждет:", "Для кого подойдет данный курс?" та "Программа курса:".

**Особливості**: Компонент сильно стилізований за допомогою CSS-модулів (імпортується python\_introduce.module.css), що забезпечує локальну область видимості стилів та уникнення конфліктів.

### **3.6.3. Компонент нижнього колонтитула (footerPY.jsx)**

Файл footerPY.jsx містить React-компонент Footer, що представляє нижній колонтитул (футер) веб-сторінки, ймовірно, специфічний для сторінок, пов'язаних з курсом Python.

**Призначення**: Надає навігаційні посилання (на курси, соціальні мережі), логотип та заклик до дії, характерний для футерів веб-сайтів.

**Зміст**: Містить списки посилань на різні курси (Python, JavaScript, C++), посилання на соціальні мережі (зображення Telegram), логотип порталу та короткий текстовий заклик.

**Стилізація**: Використовує CSS-модулі (footerPY.module.css) для стилізації.

### **3.6.4. CSS-файл для заголовка (headerPY.css)**

Файл headerPY.css містить CSS-стилі, призначені для оформлення заголовка (Header) веб-сторінки, зокрема для сторінок, пов'язаних з курсом Python.

**Призначення**: Визначає візуальний вигляд елементів заголовка: логотипу, навігаційного меню, посилань, зображень користувачів, а також стилі для підменю.

**Вміст**: Включає базові скидання стилів (box-sizing, margin, padding), загальні стилі для списків та посилань, а також специфічні стилі для .header\_\_inner, .header, .user\_\_img, .submenu та інших елементів, які використовуються в компоненті заголовка.

### **3.6.5. Компонент заголовка (headerPY.jsx)**

Файл headerPY.jsx містить React-компонент Header, який представляє верхній колонтитул (заголовок) веб-сторінки, також, імовірно, спеціалізований для курсу Python.

**Призначення**: Забезпечує основну навігацію по сайту, відображає логотип та посилання на ключові розділи, такі як "Каталог курсов", "Задачи", "Таблица лидеров".

**Структура**:

Включає логотип (Detmi.png) та зображення користувача (user.png).

Реалізує навігаційне меню з можливістю розкриття підменю для "Каталог курсов" (PYTHON, JavaScript, C++).

**Стилізація**: Покладається на стилі, визначені у headerPY.css.

### **3.6.6. Компонент навігаційної панелі (Navbar.jsx)**

Файл Navbar.jsx містить React-компонент Navbar, який, судячи з імпортів (@coreui/react, @coreui/icons/cil), використовує бібліотеку компонентів CoreUI.

**Призначення**: Створює бічну навігаційну панель з логотипом, назвою, пунктами меню, що можуть містити іконки та значки (badges), а також розкривні групи.

**Використані компоненти CoreUI**:

CSidebar: Основний контейнер для бічної панелі.

CSidebarHeader, CSidebarBrand: Для заголовка та брендування панелі.

CSidebarNav, CNavTitle, CNavItem, CNavGroup: Для структури та елементів навігації.

CIcon, CBadge: Для відображення іконок та невеликих позначок.

## **Розділ 3.6.8. Модуль для взаємодії з компілятором/перевіркою коду (compilator.jsx)**

Файл compilator.jsx (хоча назва може вказувати на компонент, його зміст є функціональним модулем) містить допоміжні функції для надсилання коду користувача на бекенд для компіляції, виконання та перевірки. Він використовує бібліотеку axios для виконання HTTP-запитів до API вашого освітнього порталу.

**Залежності**:

* + axios: HTTP-клієнт для виконання запитів до бекенду.

### **3.6.8.1. Функція Sendcode**

Функція Sendcode відповідає за надсилання коду користувача для перевірки конкретного завдання.

**Призначення**: Дозволяє користувачеві відправити своє рішення задачі на сервер для оцінки.

**Вхідні параметри**:

* 1. code: Рядок, що містить програмний код, написаний користувачем.
  2. langid: Ідентифікатор мови програмування (наприклад, 1 для Python, 2 для JavaScript тощо).
  3. taskid: Ідентифікатор завдання, для якого надсилається рішення.

**Логіка роботи**:

* 1. **Кодування коду**: Отриманий code кодується в формат Base64 за допомогою btoa(). Це може бути зроблено для безпечної передачі спеціальних символів або для подальшої обробки на сервері.
  2. **Отримання токена автентифікації**: Функція отримує JWT токен з локального сховища браузера (window.localStorage.getItem('token')). Цей токен необхідний для автентифікації запиту на бекенді, оскільки маршрут /api/:langid/:taskId/giveanswer захищений проміжним ПЗ authMiddleware.
  3. **Виконання POST-запиту**: Запит надсилається на ендпоінт http://localhost:5000/api/${langid}/${taskid}/giveanswer.
     + Тіло запиту містить закодований код (code: encodeCode).
     + Заголовки запиту включають Authorization: Bearer ${token} для автентифікації.
  4. **Обробка відповіді**: Функція очікує відповідь від сервера. У разі успіху, вона повертає data.message з відповіді, яка, ймовірно, містить повідомлення про результат перевірки коду (наприклад, "Answer accepted!").

**Зв'язок з бекендом**: Ця функція безпосередньо взаємодіє з маршрутом router.post('/:langId/:taskId/giveanswer' у taskRouter.js та контролером taskController.giveAnswer.

### **3.6.8.2. Функція sendTask**

Функція sendTask відповідає за надсилання тестового коду разом із вхідними та вихідними даними для виконання/перевірки.

**Призначення**: Ця функція, ймовірно, використовується для тестування коду без прив'язки до конкретного завдання (можливо, "пісочниця" або функціонал для викладачів/адміністраторів для додавання нових завдань та їх тестування). З назви маршруту /api/:langId/justtask та параметрів (input, output), можна припустити, що це може бути функціонал для перевірки довільного коду проти наданих вхідних/вихідних даних.

**Вхідні параметри**:

* 1. input: Вхідні дані для тестової програми.
  2. output: Очікувані вихідні дані.
  3. code: Програмний код для виконання.
  4. langid: Ідентифікатор мови програмування.

**Логіка роботи**:

* 1. **Кодування даних**: code та input кодуються в Base64.
  2. **Виконання POST-запиту**: Запит надсилається на ендпоінт http://localhost:5000/api/${langid}/justtask.
     + Тіло запиту містить закодований код, закодовані вхідні дані та очікувані вихідні дані.
  3. **Обробка відповіді**: Функція повертає data.message з відповіді сервера.

**Зв'язок з бекендом**: Ця функція взаємодіє з маршрутом router.post('/:langId/justtask' у taskRouter.js та (якщо реалізовано) методом taskController.justtask.

### **3.6.8.3. Роль у системі**

Модуль compilator.jsx є критично важливим для інтерактивної частини освітнього порталу, що дозволяє користувачам не тільки переглядати завдання, а й активно взаємодіяти з ними, надсилаючи свої рішення для автоматичної перевірки. Це є центральним елементом будь-якої платформи для вивчення програмування.

## **Використані технології та їх обґрунтування**

Для реалізації освітнього порталу було обрано стек технологій, що забезпечує високу продуктивність, масштабованість, гнучкість розробки та легкість підтримки. Вибір ґрунтувався на принципах популярності, наявності великої спільноти та багатих екосистем, що дозволяє швидко знаходити рішення типових проблем та використовувати перевірені підходи.

### **4.1. Архітектура застосунку**

Проєкт реалізовано за архітектурним шаблоном **клієнт-сервер**, де фронтенд (клієнтська частина) та бекенд (серверна частина) розроблені як окремі, взаємодіючі через API, застосунки. На стороні бекенду використовується патерн **Model-View-Controller (MVC)**, що дозволяє чітко розділити логіку застосунку на три основні компоненти:

**Моделі (Models)**: Відповідають за представлення даних та бізнес-логіку. Взаємодіють з базою даних.

**Контролери (Controllers)**: Обробляють вхідні запити, взаємодіють з моделями та формують відповіді.

**Маршрути (Routes)**: Визначають, як HTTP-запити направляються до відповідних контролерів.

### **4.2. Технології Backend (Серверна частина)**

Серверна частина освітнього порталу розроблена з використанням Node.js та фреймворку Express.js.

#### **4.2.1. Node.js**

**Опис**: Node.js – це крос-платформенне середовище виконання JavaScript, що дозволяє запускати JavaScript-код на стороні сервера. Воно базується на двигуні V8 від Google Chrome.

**Обґрунтування вибору**:

**Неблокуючий, подієво-орієнтований підхід**: Node.js ідеально підходить для створення масштабованих мережевих застосунків завдяки своїй архітектурі, що працює з великою кількістю одночасних з'єднань без блокування. Це важливо для веб-порталу, де очікується висока кількість користувачів.

**Єдина мова для фронтенду та бекенду**: Використання JavaScript на обох сторонах розробки (фронтенд та бекенд) спрощує процес розробки, дозволяючи обмінюватися знаннями та навіть деякими фрагментами коду між розробниками.

**Широка екосистема NPM**: Наявність величезної кількості бібліотек та модулів у реєстрі NPM прискорює розробку, дозволяючи використовувати готові рішення для типових задач.

#### **4.2.2. Express.js**

**Опис**: Express.js – це мінімалістичний та гнучкий фреймворк для веб-застосунків Node.js, що надає надійний набір функцій для розробки веб- та мобільних застосунків.

**Обґрунтування вибору**:

* + **Простота та гнучкість**: Express.js не нав'язує жорстких правил щодо структури проєкту, дозволяючи розробнику обирати найбільш підходящий підхід.
  + **Маршрутизація та проміжне ПЗ**: Надає потужні механізми для визначення маршрутів та використання проміжного ПЗ (middleware), що дозволяє легко реалізовувати автентифікацію, авторизацію, логування тощо.
  + **Швидка розробка API**: Добре підходить для створення RESTful API, що є основою взаємодії між клієнтською та серверною частинами.

#### **4.2.3. Sequelize ORM**

**Опис**: Sequelize – це об'єктно-реляційний мапер (ORM) для Node.js, який підтримує різні реляційні бази даних (PostgreSQL, MySQL, SQLite, MS SQL Server). Він дозволяє взаємодіяти з базою даних за допомогою JavaScript-об'єктів, а не безпосередньо через SQL-запити.

**Обґрунтування вибору**:

**Абстракція бази даних**: ORM приховує складність SQL-запитів, дозволяючи розробнику працювати з даними як з JavaScript-об'єктами, що прискорює розробку та знижує ймовірність помилок.

**Підтримка міграцій**: Sequelize надає інструменти для управління міграціями бази даних, що полегшує зміни в структурі БД під час розробки та розгортання.

**Безпека**: Допомагає запобігти таким вразливостям, як SQL-ін'єкції, оскільки запити генеруються ORM.

#### **4.2.4. MySQL**

**Опис**: MySQL – це відкрита реляційна система управління базами даних (РСУБД), яка широко використовується для веб-застосунків.

**Обґрунтування вибору**:

* + **Надійність та стабільність**: MySQL є однією з найпопулярніших та перевірених РСУБД, відомою своєю стабільністю та продуктивністю.
  + **Відкритий вихідний код**: Безкоштовність та відкритість робить її доступним вибором для багатьох проєктів.
  + **Сумісність**: Добре інтегрується з Node.js та Sequelize ORM.

#### **4.2.5. bcrypt.js**

**Опис**: Бібліотека для хешування паролів.

**Обґрунтування вибору**: Забезпечує надійне зберігання паролів користувачів у базі даних, хешуючи їх з використанням солі, що є критично важливим для безпеки застосунку.

#### **4.2.6. jsonwebtoken (JWT)**

**Опис**: Бібліотека для створення та перевірки JSON Web Tokens (JWT).

**Обґрунтування вибору**: JWT використовується для безстаночної (stateless) автентифікації користувачів. Це дозволяє масштабувати застосунок, оскільки серверу не потрібно зберігати сесії користувачів. Токени є криптографічно підписаними та містять інформацію про користувача, що дозволяє швидко перевіряти його ідентичність при кожному запиті.

### **4.3. Технології Frontend (Клієнтська частина)**

Клієнтська частина освітнього порталу розроблена з використанням React.js.

#### **4.3.1. React.js**

**Опис**: React.js – це JavaScript-бібліотека для створення користувацьких інтерфейсів, розроблена Facebook. Вона дозволяє розробникам створювати складні UI з невеликих, ізольованих і повторно використовуваних фрагментів коду, які називаються компонентами.

**Обґрунтування вибору**:

**Компонентний підхід**: Розбиття інтерфейсу на компоненти спрощує розробку, тестування та підтримку. Кожен компонент відповідає за свою частину UI.

**Віртуальний DOM**: React використовує віртуальний DOM для оптимізації оновлень інтерфейсу, що забезпечує високу продуктивність та швидкий відгук застосунку.

**Декларативний підхід**: Дозволяє описувати, як має виглядати UI, а React сам керує оновленнями.

**Широка спільнота та екосистема**: Велика кількість доступних бібліотек, інструментів та активна спільнота розробників прискорює розробку та дозволяє легко знаходити допомогу.

#### **4.3.2. react-router-dom**

**Опис**: Бібліотека для декларативної маршрутизації в React-застосунках.

**Обґрунтування вибору**: Дозволяє створювати односторінкові застосунки (Single Page Applications - SPA) з динамічною маршрутизацією на стороні клієнта, забезпечуючи плавну навігацію без перезавантаження сторінки.

#### **4.3.3. axios**

**Опис**: HTTP-клієнт, що базується на Promise, для браузера та Node.js.

**Обґрунтування вибору**: Використовується для виконання HTTP-запитів з клієнтської частини до бекенд API. Надає зручний інтерфейс для надсилання даних, обробки відповідей та помилок.

#### **4.3.4. CSS-модулі та звичайний CSS**

**Опис**: Для стилізації застосунку використовуються як звичайні CSS-файли (.css), так і CSS-модулі (.module.css).

**Обґрунтування вибору**:

* + **Звичайний CSS**: Використовується для глобальних стилів або стилів, що застосовуються до багатьох компонентів.
  + **CSS-модулі**: Забезпечують локальну область видимості для стилів, запобігаючи конфліктам імен класів між різними компонентами. Це робить стилізацію більш модульною та легкою для підтримки.

#### **4.3.5. CoreUI React (якщо використовується повністю)**

**Опис**: Набір компонентів React, заснованих на Bootstrap, призначений для швидкої розробки адміністративних панелей та користувацьких інтерфейсів.

**Обґрунтування вибору**: (Якщо Navbar.jsx є частиною ширшого використання CoreUI) Прискорення розробки UI за рахунок використання готових, попередньо стилізованих та функціональних компонентів, що дозволяє зосередитися на бізнес-логіці, а не на деталях верстки.

### **4.4. Додаткові інструменти та практики**

**ESLint / Prettier**: Для забезпечення єдиного стилю коду та виявлення потенційних помилок на ранніх етапах розробки.

**Система контролю версій (Git)**: Для ефективної командної роботи, відстеження змін у коді та управління версіями проєкту.

### 