Міністерство освіти і науки України НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

# Розробка REST API для системи запису на вибіркові

**Текстова частина до курсової роботи**
**за спеціальністю „Комп'ютерні науки" 122**

Керівник курсової роботи

ст.викладач Вовк Н.Є.

_____

*(підпис)*

"____" _____ 2021 р.

Виконала студентка

Лунякіна В.А.

"____" _____ 2021 р.

Київ 2021

ЗМІСТ

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультімедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри мультімедійних систем,

Доцент., к. ф.-м. н. О.П.Жижерун

_____

(підпис)

«____»_____2021 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

на курсову роботу

студентці Лунякіній Валерії Андріївні факультету інформатики 3-го курсу
ТЕМА Розробка REST API для системи запису на вибіркові

Зміст ТЧ до курсової роботи:

1. Індивідуальне завдання

2. Календарний план
3. Анотація

4. Вступ
5. Аналіз предметної області
6. Теоретичні відомості
7. Розробка клієнтської частини

8. Висновки
9. Список використаної джерел

Дата видачі „___” _____ 2021 р.

Керівник _____ (підпис)

Завдання отримав _____ (підпис)

## Календарний план виконання курсової роботи

**Тема:** Платформа для пошуку та замовлення професійної фотозйомки

**Календарний план виконання роботи:**

| № п/п | Назва етапу курсового проекту (роботи) | Термін виконання етапу | Примітка |
|---|---|---|---|
| 1. | Отримання завдання на курсову роботу. | 24.10.2020 | |
| 2. | Ознайомлення з існуючою інформацією по темі | 01.02.2021 | |
| 3. | Початок створення практичної частини | 01.03.2021 | |
| 4. | Початок написання теоретичної частини | 13.04.2021 | |
| 5. | Подання проміжної версії практичної частини | 22.04.2021 | |
| 6. | Аналіз практичної частини; її корегування | 08.05.2021 | |
| 7. | Остаточне завершення написання теоретичної частини роботи та розробки практичної частини; корегування | 14.05.2021 | |
| 8. | Створення презентації | 15.05.2021 | |
| 9. | Захист курсової роботи | З 24.05.2021 | |

Студент Лунякіна В.А.

Керівник Вовк Н.Є.

“_____” _____

Перелік умовних позначень:

1. Айді – id, особистий ідентифікатор
2. REST – Representational State Transfer, передача станів представлення
3. API – application programming interface, програмний інтерфейс застосунку
4. САЗ – система автоматизованого запису

Анотація

У роботі детально розглянуто REST архітектуру, види та побудову запитів. Розроблено API для системи автоматизованого запису на дисципліни, на основі REST підходу за допомогою php фреймворка yii2. Надано приклади запитів та результати виконання.

# ВСТУП

В сучасному світі у розробці програмних застосунків, особливо веб-застосунків, стає дуже популярно використовувати клієнт-серверну архітектуру. Це зрозуміло, бо такий підхід є дуже зручним. Він розділяє інтерфейси, тим самим робить їх простіше та зрозуміліше, надає їм можливість розроблятися незалежно.

Також із усім цим набирає популярності архітектура REST. Такий підхід надає масштабування взаємодії компонентів системи, незалежне впровадження компонентів, проміжні компоненти, які знижують затримку та посилюють безпеку.

Основною метою роботи є розробка REST API для системи автоматизованого запису на дисципліни, який внесе добрий внесок до майбутнього масштабування системи. Це зможе бути залучено до співпраці із іншими системами, вивантаження даних для інших систем, інше. Також послугує добрим розділенням фронт-енд та бек-енд частин системи.

Робота містить три основні розділи, кожен з яких містить власні підрозділи для уточнення та заглиблення у тему дослідження.

Перший розділ містить аналіз вже наявної системи, причину актуальності теми, розбір предметної області  та сформовану постановку завдання.

У другому розділі висвітлюються дослідження про теоретичні відомості за заданою темою. Присутній загальний опис REST підходу, вимоги до розробки використовуючи дану архітектуру, детальний опис структури запиту, бо це є однією із головних частин REST API.

Третій розділ описує реалізацію проекту, надає перелік використаних технологій, наведені деталі розробки. Детально демонструє як користуватися REST API, який має бути результат. Наведені результати тестування усіх можливих запитів.

# РОЗДІЛ 1: Аналіз предметної області

## 1.1 Аналіз сучасного стану питання та обґрунтування теми

На цей час уся взаємодія із інформацією системи виконується через веб-застосунок. Отримання даних, їх зміна, створення та видалення виконуються за допомогою інтерфейсу користувача.
Створення REST API для системи автоматизованого запису буде слугувати гарним кроком до майбутнього масштабування системи. Це дасть незалежне впровадження компонентів, розділення фронт-енд від бек-енд частин системи. Також це буде використовуватися для зв’язку із іншими системами, наприклад, вивантаження даних для інших систем. Ще, через те, що цей інтерфейс є уніфікованим, то буде дуже простий у використанні та майбутніх вдосконалень.

## 1.2 Опис предметної області

### 1.2.1 Загальний опис системи

Система автоматизованого запису на дисципліни (САЗ) – веб-застосунок, за допомогою якого студент має можливість записуватися на дисципліни та групи онлайн. Також система надає можливість подати запит, щоб виписатися із певної дисципліни, переглядати власні індивідуальні плани за всі роки навчання, шукати дисципліни.
Окрім студента, система має ще 5 ролей: адміністратор, методист, методист навчально-методичного відділу, методист деканату, гість.
Кожен користувач має свої певні можливості:

- Гість може переглядати тільки інформацію сайту

- Методист може те саме, що й гість, а також переглядати та змінювати інформацію про курси, групи, реєстрації на курси,
- Методист деканату може те саме, що і попередні користувачі, а також переглядати та змінювати інформацію про методистів, спеціальності, студентів, заявки на виписування
- Методист навчально-методичного відділу може те саме, що і попередні користувачі, а також переглядати та змінювати інформацію про факультети, кафедри.
- Адміністратор може виконувати всі дії, що і інші користувачі, а також переглядати та змінювати інформацію щодо користувачів, новини, керувати налаштуваннями системи.

### 1.2.2 Опис моделей

Система вимагає наявності модель для опису даних, кожній відповідає своя таблиця у базі даних.

Модель Course – описує дисципліну.

Модель CourseGroup – описує групу з дисципліни.

Модель CourseRegister – описує реєстрацію студентів на дисципліни.

Модель CourseSeason – описує сезон курсу та надає інформацію щодо годин та інше.

Модель CourseSpeciality – описує зв’язок дисципліни та спеціальності.

Модель Faculty – описує факультет.

Модель Metodist – описує методиста.

Модель News – описує новини сайту.

Модель Speciality – описує спеціальність.

Модель Student – описує студента.

Модель Subfaculty – описує кафедру.

Модель UnenrollRequest – описує заявку на виписування.

Модель UnenrollRequestComment – описує коментарі заявки на виписування.

Модель User – описує користувача.

Кожна модель має свою структуру, де описано атрибути, їх типи та значення. (Додаток А)

Також присутній вигляд моделей, як таблиць бази даних, та схема бази даних. (Додаток Б)

## 1.3 Постановка завдання

Через те, що система планується масштабуватися, то потрібно розробити RESTful API для системи запису на вибіркові дисципліни.

Основні завдання роботи:

- Зробити можливість отримувати всі данні із бази даних, створювати нові, змінювати, видаляти, робити пошук за айді та бажаними атрибутами;
- Додати спосіб обирати які самі дані ми будемо отримувати;
- Передбачити робити тільки автентифіковані за токеном запити;
- Обмежити дії для кожного зареєстрованого у системі користувача відповідно до його ролі.

# РОЗДІЛ 2: Теоретичні відомості

## 2.1 Визначення REST API

API (application programming interface) – певний набір правил, який дозволяє програмам спілкуватися між собою.
REST (Representational State Transfer) – архітектурний стиль, що складається з набору обмежень, які застосовуються до елементів всередині архітектури.
Коли клієнт робить запит через RESTful API, то інформація подається в одному із декількох форматів HTTP: JSON, HTML, XML, Python, PHP.
JSON (Javascript Object Notation) є одним із найпопулярнішим варіантом для цього, бо він є мовно-агностичним та доступним для читання людям та комп'ютерам.
Архітектура REST сама по собі не прив'язана до конкретних технологій і протоколів. Але зараз майже завжди побудова RESTful API передбачає використання HTTP і JSON або XML. [1]

## 2.2 Вимоги до REST архітектури

Архітектура REST описується 6 обмеженнями. Перший раз вони були представлені Роєм Філдінгом (Roy Fielding) у його докторській дисертації та визначають основи RESTful стилю: [1]

1. Модель клієнт-сервер – система повинна бути розділена на клієнтів та серверів. Відокремлення цих інтерфейсів підвищує мобільність клієнтського інтерфейсу на інші платформи та спрощення серверної частини робить масштабуємість краще.

2. Відсутність станів – сервер не повинен зберігати яку-небудь інформацію про клієнтів. Запит повинен містити тільки всю необхідну інформацію для обробки запиту та, якщо це є необхідним, то й ідентифікацію клієнта. Стан сесії при цьому зберігається на клієнтській частині. Клієнт ініціює відправку запиту, коли він готов перейти у інший стан. Під час обробки запитів вважається, що клієнт перебуває у перехідному стані.
3. Кешування – для кожної відповіді серверу повинно бути зазначено, чи є вона кешованою. Для того, щоб запобігти повторного використання клієнтами застарілих та некоректних даних у відповідь на наступні запити. Правильне використання кешування може прибрати деяку клієнт-серверну взаємодію, чим покращить виробництво та масштабуємість системи.
4. Уніфікований інтерфейс – головна особливість, яка відрізняє REST архітектурний стиль від інших мережевих стилів. Застосовуючи принцип загальності програмного забезпечення до інтерфейсу компоненту, спрощується загальна архітектура системи та покращується видимість взаємодій. Однак це погіршує ефективність, бо інформація передається у стандартизованій формі, а не в тій, яка підходить програмі. Принципи уніфікованого інтерфейсу:

   a. Ідентифікація ресурсів – у запитах всі ресурси повинні бути ідентифікованими. Цей ідентифікатор повинен бути стабільним, тобто він не може змінюватися під час зміни стану ресурсу. В REST цим ідентифікатором є URI.

   b. Маніпуляція ресурсами через представлення – у REST представлення потрібні, щоб виконувати дії над ресурсами. Представлення ресурси є станом цього ресурсу на цей час. Якщо клієнт зберігає у собі представлення ресурсу, то він має достатню

кількість інформації, щоб змінювати чи видаляти ресурс.

c. Само-документуємі повідомлення – запит та відповідь повинні зберігати в собі достатню кількість інформації для їх обробки. Не повинно бути додаткового повідомлення чи кешу для обробки запиту. Це є дуже важливим для масштабування системи.

d. HATEOAS – гіпермедія як засіб зміни стану застосунку. Клієнти можуть змінювати стани системи тільки через дії, які визначені в гіпермедії на сервері. Статус ресурсу передається через тіло запиту, параметри строки запита, заголовки запитів та URI.

5. Система із слоями – можна розділити систему на ієрархію, але тільки так, що кожен компонент може бачити тільки компоненти наступного слоя. Клієнт не може точно визначити, взаємодіє він із сервером або із проміжним вузлом. Використання проміжних серверів може значно підвищити масштабуємість через балансування навантаження та балансування кешування.
6. Код на вимогу – у REST можливо розширити функціонал клієнта через завантаження коду із серверу за допомогою аплетів та сценаріїв.

## 2.3 Структура запиту

Кожен запит повинен складатися із 4 частин:

- Кінцева точка (the endpoint);
- Метод;
- Заголовки (the headers);
- Дані.

### 2.3.1 Кінцева точка

Кінцева точка містить уніфікований ідентифікатор ресурсу (URI), який вказує де і як знайти ресурс в Інтернеті. Найпоширенішим типом URI є URL, яке служить повною веб-адресою.

Кінцева точка має наступну структуру:

*root-endpoint/?*

Де root-endpoint є початковою точкою того API, до якого ми робимо запит.

Наприклад, у Github API вона буде мати наступний вигляд:

*https://api.github.com*

Шлях (path) визначає ресурс, на який ми робимо запит. Ми можемо отримати доступ до шляхів так само, як і до будь-якої частини веб-сайту. Наприклад, щоб отримати список усіх репозиторіїв користувача, потрібно перейти *на https://github.com/userName/repositories*, де username – це ім’я користувача. В цьому прикладі *https://github.com/* є root-endpoint, а */userName/repositories* – шляхом.

Для того, щоб розуміти які шляхи доступні нам, потрібно переглянути документацію API. Наприклад, щоб так само побачити усі репозиторії [https://docs.github.com/en/rest/reference/repos#list-user-repositories]користувача Github, документація пропонує наступний спосіб:

*/users/:username/repos*

Знак “:” у шляху означає, що після двокрапки буде змінна. Нам буде потрібно замінити її на наше значення. Я буду шукати усі свої репозиторії, тому заміню “:username” на “valeriia-lun”. Отже, кінцева точка для отримання усіх моїх репозиторіїв буде наступною:

*https://api.github.com/users/valeriia-lun/repos*

Остання частина кінцевої точки це параметри запиту (query parameters).

Ще їх іноді називають необов’язковими (optional) параметрами, що є першою рисою, яка їх відрізняє від ієрархічних параметрів. Друга особливість полягає в тому, що вони не повинні бути унікальними. Це означає, що ми можемо вказати будь-який параметр декілька раз. Вони дають можливість змінити запит та мають структуру пари ключ-значення. Вони завжди починаються із знаку “?” та потім кожна наступна пара відокремлюється знаком “&”. Виглядає це наступним чином:

*?query1=value1&query2=value2*

### 2.3.2 Методи

HTTP метод описує що саме потрібно зробити із ресурсом. Існує 5 типів запиту:

- GET
- POST
- PUT
- PATCH
- DELETE

Вони використовуються для виконання 4 дій: Create, Read, Update, Delete (CRUD).

| **Назва** | **Значення** |
|---|---|
| **GET** | Використовується для того, щоб отримати наш ресурс із серверу. Він відповідає операції READ. Також, цей метод використовується за замовченням. |

| | |
|---|---|
| **POST** | Він використовується, коли потрібно створити новий ресурс на сервері. Запит цього методу відповідає операції CREATE. |
| **PUT та PATCH** | Використовуються, щоб оновити ресурс на сервері. Вони відповідають операції UPDATE. |
| **DELETE** | Запит цього методу використовують для того, щоб оновити ресурс на серверу. Він відповідає операції DELETE. |

### 2.3.3 Заголовки

Заголовки зберігають у собі інформацію для клієнта і сервера. Зазвичай вони надають дані автентифікації та інформацію про формат відповіді. HTTP заголовки представляють собою пари ключ-значення.

Існує декілька заголовків, які найчастіше використовують:

- Authorization – несе інформацію про автентифікацію клієнта для ресурсу запита;
- WWW-Authenticate – відправляється сервером у випадку, коли потрібна форма автентифікації перед тим, як він зможе надати ресурс у відповідь;
- Accept-Charset – несе інформацію серверу про набори символів, які клієнт може приймати. Встановлюється разом із запитом;
- Content-Type – вказує медіа-тип відповіді сервера клієнту. Це потрібно, щоб клієнт правильно обробляв тіло відповіді;
- Cache-Control – політика кешування, яку визначає сервер для відповіді. Такий тип відповіді може буду збереженим у клієнта і використовуватися повторно до часу, який зазначений у заголовку.

### 2.3.4 Дані

Дані або тіло запиту (body) використовуються для передачі додаткової інформації на сервер. Наприклад, це можуть бути дані, які ми бажаємо додати чи оновити.

За замовченням cURL надсилає дані так, ніби вони надсилаються через поля форми сторінки. Якщо ми бажаємо відправити дані у форматі JSON, то нам потрібно буде встановити заголовок “Content-Type” із значенням “application/json” та звести дані до форми об’єкту JSON, як наведено нижче:

```
{
  "property1":"value1",
  "property2":"value2"
}
```

### 2.3.5 Автентифікація

Інколи клієнт може не мати права виконувати певні запити та у відповідь буде отримувати повідомлення “Requires authentication”. Це означає, що клієнт не авторизован.

Розробники вводять обмеження на виконання певних запитів через міри безпеки. Тому, що запити POST, PUT, PATCH та DELETE змінюють дані у базі, програмісти майже завжди ставлять на них обмеження, щоб убезпечити себе від небажаної втрати чи зміни даних. Іноді виставлення автентифікації вимагає й запит GET. Наприклад, коли потрібно отримати доступ до свого банківського рахунку для перевірки балансу.

Існує два головні типи автентифікації:

- Використовуючи ім’я користувача та пароль
- Використовуючи секретний токен

Щоб спробувати використати перший варіант, можна це зробити за допомогою cUrl, нам потрібно буде використати “-u” та після цього написати ім’я користувача та пароль, щоб це виглядало наступним чином:

*curl -x POST -u "username:password" https://api.github.com/user/repos*

Щоб автентифікуватися за допомогою секретного токена, то потрібно використати HTTP POST метод та ресурс login. Це має виглядати так:

*https://host:port/ibmmq/rest/v1/login*

Та додати ім’я користувача і пароль у тілі запиту:

```
{
   "username" : name,
   "password" : password
}
```

Зберігайте токен, якій повернеться із запитом, у local cookie store. Потім можна автентифікувати запити REST за допомогою цього збереженого токена. Для цього до запиту потрібно додати заголовок ibm-mq-rest-csrf-token.

## 2.4 Коди станів HTTP

Деякі повідомлення з’являються лише тоді, коли із нашим запитом щось не так. Коди станів HTTP дозволяють швидко визначити стан відповіді. Діапазон від 100+ до 500+. Загалом вони відповідають наступним правилам:

- **100+** – Інформація (Information). *Наприклад, 100: Continue*
- **200+** – Успішний запит (Success). *Наприклад, 200: OK, 201: Created, 202: Accepted, 204: No Content*
- **300+** – Запит перенаправлен на інший URL (Redirect). *Наприклад, 301: Moved Permanently, 307: Temporary Redirect*
- **400+** – Помилка на клієнті (Client Errore). *Наприклад, 400: Bad Request, 401: Unauthorized, 403: Forbidden, 404: Not Found*

- **500+** – Помилка на сервері (Server Error). *Наприклад, 500: Internal Server Error, 501: Not Implemented, 502: Bad Gateway, 503: Service Unavailable, 504: Gateway Timeout*

# РОЗДІЛ 3: Розробка клієнтської частини

## 3.1 Аналіз технічного завдання

На меті стоїть розробка REST API для системи запису на дисципліни. Це значно спростить роботу із даними користувачам, допоможе оптимізувати вже наявну систему та послугую гарним вкладенням до майбутнього масштабування сервісу. Також перевага цього REST API в тому, що в ньому присутній розподіл обмежень прав відповідно до ролей користувачів, що покращує структуру взаємодій.

У САЗі наявно декілька типів ролей користувачів: адміністратор, методист НВЗ, методист деканату, методист, гість. Усі методи цих ролей будуть реалізовані відповідно до їх прав доступу.

Технічні вимоги користувачів до функціональної системи:

Гість:

- Переглядати інформацію

Методист:

- Все, що і попередній користувач
- Створювати, змінювати, видаляти блоки дисциплін
- Переглядати всі блоки із їх дисциплінами
- Фільтрувати блоки дисциплін
- Отримувати блок по дисципліні
- Створювати, змінювати, видаляти дисципліни

- Переглядати всі дисципліни
- Фільтрувати дисципліни
- Отримувати кафедру дисципліни
- Отримувати всіх студентів, які записані на дисципліну
- Створювати, змінювати, видаляти інформацію про зв’язок дисципліни й спеціальності
- Переглядати всі зв’язки дисциплін та спеціальностей
- Фільтрувати зв’язки дисциплін та спеціальностей
- Створювати, змінювати, видаляти інформацію про сезон дисципліни
- Переглядати всі сезони дисциплін
- Фільтрувати сезони дисциплін
- Створювати, змінювати, видаляти реєстрації студентів на дисципліни
- Переглядати всі реєстрації студентів на дисципліни
- Фільтрувати реєстрації студентів на дисципліни
- Отримувати студента за реєстрацією на дисципліну
- Отримувати дисципліну за реєстрацією
- Створювати, змінювати, видаляти групи дисциплін
- Переглядати всі групи дисциплін
- Фільтрувати групи дисциплін
- Отримувати курс за групою

Методист деканату:

- Все, що і попередній користувач
- Створювати, змінювати, видаляти спеціальності
- Переглядати всі спеціальності
- Фільтрувати спеціальності

- Отримувати факультет спеціальності
- Отримувати усі дисципліни спеціальності
- Створювати, змінювати, видаляти методистів
- Переглядати всіх методистів
- Фільтрувати методистів
- Отримувати всіх методистів факультету
- Створювати, змінювати, видаляти студентів
- Переглядати всіх студентів
- Фільтрувати студентів
- Отримувати усі дисципліни, на які записан студент
- Отримувати спеціальність та факультет студента

Методист НВЗ:

- Все, що і попередній користувач
- Створювати, змінювати, видаляти заявки на виписування із дисципліни та їх коментарі
- Переглядати усі заявки на виписування та їх коментарі
- Фільтрувати заявки та коментарі
- Отримувати окремо всі коментарі заданої заявки
- Отримувати окремо всі заявки на виписування заданого користувача
- Створювати, змінювати, видаляти кафедри
- Переглядати всі кафедри
- Фільтрувати кафедри
- Отримувати всі кафедри факультету
- Отримувати факультет кафедри
- Створювати, змінювати, видаляти факультети
- Переглядати всі факультети

- Фільтрувати факультети
- Отримувати всі спеціальності факультету

Адміністратор:

- Все, що і попередній користувач
- Створювати, змінювати, видаляти користувачів
- Переглядати всіх користувачів системи
- Фільтрувати користувачів
- Створювати, змінювати, видаляти новини
- Переглядати всі новини
- Фільтрувати новини

Для того, щоб зробити будь-який запит потрібно автентифікуватися. Для цього потрібно ввести свій секретний токен, який вже знаходиться у базі даних. Після цього, відповідно до ролі користувача, йому будуть надані права виконувати його запити.

## 3.2 Обґрунтування структури програми

Оскільки за допомогою REST API ми хочемо працювати із даними бази даних, то буде логічним моделі, які будуть відповідати майже усім таблицям бази даних.
Також я вирішила додати скорочені версії деяких моделей для того, щоб спростити розуміння інформації користувачу.

## 3.3 Обґрунтування вибору засобів розробки

Для розробки REST API мною було обрано мову PHP. Вона є однією із найпоширеніших мов, що використовують у сфері веб-розробок. [2]
Особливості мови:

- Нетрадиційність – мова здається дуже знайомою багатьом розробником, бо багато конструкцій запозичені із мов C та Perl. Код PHP дуже схожий на код типових програм C та Perl. Ця мова поєднує у собі переваги обох цих мов та спеціально спрямована на роботу в Інтернеті. Синтаксис дуже універсальний та зрозумілий.
- Наявність доступного коду та безкоштовність – безкоштовне розповсюдження початкового коду надало неоцінену послугу користувачам.
- Ефективність – ця мова належить до інтерпретованих, що дозволяє обробляти сценарії з достатньо високою швидкістю.
- Наявність інтерфейсів до багатьох баз даних – у PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, SQLite, mSQL, Oracle, Hyperware, Informix, InterBase, Sybase. Через стандарт відкритого інтерфейсу для зв’язку з базами даних (ODBC) можна підключитись до усіх баз даних, до яких існує драйвер.

Разом із мовою PHP я використовую фреймворк Yii2. Він є одним із найпопулярніших фреймворків PHP. Це високопродуктивний об’єктноорієнтований веб-фреймворк, який реалізую парадигму MVC модель-представлення-контролер та має відкритий код. [3]
Можливості фреймворку:

- Висока продуктивність
- Паттерн MVC
- Інтерфейси DAO та Active Record для роботи із базами даних

- Кешування сторінок та окремих фрагментів
- Перехоплення та обробка помилок
- Введення та валідація веб-форм
- Авторизація та автентифікація
- Генерація базового PHP коду для CRUD-операцій
- Міграції бази даних
- Автоматизоване тестування
- Підтримка REST

Для написання проекту мною було обрано середовище розробки PhpStorm від компанії JetBrains. Це IDE для професійної розробки на мові PHP. Завдяки дуже гарному алгоритму аналізу кода, значно спрощується робота із написанням коду. Присутнє автозаповнення, пошук помилок, швидкі виправлення та зручна навігація по коду.

Вбудований термінал надає можливість без проблем встановлювати потрібні бібліотеки, документувати зміни програмного коду за допомогою системи контролю версій.

Для адміністрування СКБД MySQL я використовувала веб-застосунок phpMyAdmin із дуже зручним інтерфейсом для взаємодії із базою даних. Цей застосунок має великий попит серед розробників, бо дозволяє керувати СКБД без безпосереднього вводу SQL запитів.

## 3.4 Опис реалізації

API зроблено у вигляді додаткового модуля, який потрібно буде приєднати до вже існуючого проекту. Модуль містить три діректорії:

- config – відповідає за налаштування модулю
- controllers – містить у собі всі контроллери модулю

- models – зберігає всі моделі, які нам потрібні

Також присутній файл Module.php, який представляє програму, яка сама по собі містить елементи MVC (моделі, контроллери, преставлення тощо).

```php
<?php

namespace app\api;

class Module extends \yii\base\Module
{
    /**
     * @inheritdoc
     */
    public $controllerNamespace = 'app\api\controllers';

    /**
     * @inheritdoc
     */
    public function init()
    {
        parent::init();
    }
}
```

Він має відкриту змінну controllerNamespace, що визначає місцезнаходження усіх наших контролерів у нашому модулі API, та метод init(), який ініціалізує модуль.

У папці config знаходяться два файли api.php та db.php.
У першому знаходяться всі налаштування модуля, а у другому налаштування, щоб підключитися до бази даних.

У папці models знаходяться усі моделі, які нам будуть потрібні для REST API. Там присутні звичайні моделі (наприклад Course.php) та короткі версії (Courseshort.php). Щоб не переписувати вже існуючі моделі у проекті та уникнути дублювання коду я вирішила зробити, щоб усі моделі REST API наслідували їм відповідні моделі із головного проекту. Тому в

оголошенні класу буде написане ключове слово extends та це буду мати наступний вигляд:

```
class Faculty extends \app\models\Faculty{
```

Кожна основна модель наслідує клас \yii\db\ActiveRecord [4]. Він надає об'єктно-орієнтовний інтерфейс для доступу та керування даними, що зберігаються в базах даних. Цей клас пов'язаний з таблицею бази даних, а його екземплях відповідає рядку цієї таблиці, а атрибут екземпляра – певному стовпцю цього рядку.

Це дає нам можливість викликати методи Active Record для доступу та керування даними із таблиць баз даних, замість того, щоб писати запити SQL.

У кожному такому класі повинен бути метод tableName(), який повертає назву таблиці у базі даних, якій відповідає цей клас. Наприклад, цей метод для класу News:

```
public static function tableName()
    {
        return '{{%news}}';
    }
```

Також для кожної моделі існую свій власний клас Query, де будуть запити до бази даних для цієї моделі. Цей клас наслідую \yii\db\ActiveQuery [5] та має два метода. Метод one() повертає один запис, заповнений першим рядком даних. Метод all() повертає всі записи на основі результатів запиту.

Так як майже всі моделі API наслідують свої ж моделі головного проекту, то всі відповідні їм класи Query реалізовані у головному проекті. Але так як я додала модель News, то потрібно було і додати для неї NewsQuery, що я і зробила. У цьому класі реалізовані всі потрібні методи для роботи.

У кожному класі, який наслідує \yii\db\ActiveQuery також присутній метод find(), який повертає наш особистий Query.

Для того, щоб працювати із реляційними даними за допомогою ActiveRecord, спочатку потрібно оголосити відношення в цих класах. Тобто потрібно просто оголосити метод відношення для кожного відношення, яке нам цікаве. Наприклад:

```
public function getSpecialities(){
        return $this->hasMany(Speciality::className(), ['faculty_id' => 'id']);
    }
```

Тут нам потрібно обрати кратність нашого відношення. Це може бути hasMany() або hasOne(). Також вказати ім'я потрібного нам класу та визначити зв'язок між двома класами. Це робиться просто визначивши колонки, за якими вони пов'язані. Значення масиву – це стовпці первинних даних, а ключ масиву – стовпці пов'язаних даних.

Також кожен клас має відкритий метод rules(), який визначає правила валідації атрибутів.

Так як нам не скрізь потрібна вся інформація про наші об'єкти, які мають відношення до головного об'єкту. Наприклад, коли ми хочемо вивести список студентів, а там до кожного студента буде виводитися список його дисциплін. То в цьому випадку нам не потрібна повна інформація про дисципліну, нас цікавлять тільки основні дані. Тому мною було вирішено для деяких моделей створити іншу версію їх, але вона вже буде повертати не таку повну інформацію. Тому, в деяких моделях API я переписала методи відношення даних, використовуючи коротшу версію моделі.

Також, для кожної моделі можна описати конкретні поля, які ми хочемо, щоб вона повертала. Метод fields() за замовчуванням повертає усі поля моделі, але ми можемо перевизначити цей метод для того, щоб вказати власний набір обов'язкових атрибутів моделі для виведення. Також я додала ще метод extraFields(), який за замовчуванням повертає пустий

масив. Але його теж можна перевизначити та вказати які саме атрибути моделі ми би хотіли, щоб були додатковими та їх можна було б вивести за бажанням, вказавши це в запиті. Якраз сама ці два методи надали мені можливість створити коротку версію деяких моделей для зручності, просто вказавши менше атрибутів у методі fields().[6]
Більш детальний приклад моделі. (Додаток В)

У REST API кожній моделі відповідає свій контролер. Кожен такий клас наслідує yii\rest\ActiveController [7]. ActiveController реалізує загальний набір дій для підтримки RESTful доступу до ActiveRecord. Контролер повинен мати поле modelClass, яке потрібно ініціалізувати класом нашої моделі, за яку відповідає наш контролер. За замовченням є наступні дії:

- index – список моделей
- view – деталі певної моделі
- create – створює нову модель
- update – оновлює дані в існуючій моделі
- delete – видаляє існуючу модель
- options – повертає доступні HTTP методи

Також, для того, щоб API міг приймати дані у форматі JSON, потрібно додати конфігурування parsers у компоненту request:

```
'request' => [
            'parsers' => [
                'application/json' => 'yii\web\JsonParser',
            ],
        ],
```

У кожному класі контролера присутній метод behaviors(), який визначає поведінку нашого компоненту. Там можна зазначити різні фільтри, але я використовувала саме authenticator, який забезпечує автентифікацію користувача. Виглядає це наступним чином:

```
public function behaviors(){
        $behaviors = parent::behaviors();

        $behaviors['authenticator'] = [
            'class' => HttpBearerAuth::class
        ];

        return $behaviors;
    }
```

Використовую я фільтр дій HttpBearerAuth [8], він підтримує метод автентифікації на основі HTTP Bearer Token. Такий тип токену є cryptic string, який зазвичай генерується сервером у відповідь на запит логіну. Клієнт повинен надсилати цей токен у заголовку Authentication, коли робить запит захищених ресурсів.

Для роботи цієї автентифікації я ще додала атрибут “access_token” до головної таблиці “_user” бази даних.

Потім ще потрібно імплементувати метод findIdentityByAccessToken() у нашому головному класі User. [9] Це буде мати наступний вигляд:

```
public static function findIdentityByAccessToken($token, $type = null)
    {
        return static::findOne(['access_token' => $token]);
    }
```

Після цього кожного разу при запиту до API запитуємий контролер буде намагатися автентифікувати користувача у своєму методі beforeAction(). Якщо це пройде успішно, то контролер зробить інші перевірки та виконає дію. Інформацію про автентифікованого користувача можна буде отримати через Yii::$app->user->identity. Якщо автентифікація була неуспішною, то буде повернено відповідь із HTTP кодом 401.

Так як мені потрібно зробити доступ користувачам до дій відповідно до їх ролей у системі, то я це зробила наступним чином. Для цього мені потрібно було перевизначити метод checkAccess(). Це потрібно зробити у

кожному контролері, який ми бажаємо обмежити у доступі. Виглядати це буде так:

```
public function checkAccess($action, $model = null, $params = [])
    {
        if ($action === 'view'||$action === 'update' || $action === 'delete'||
$action === 'index'|| $action === 'create') {
            if (!empty(Yii::$app->user->identity->role) && in_array(Yii::$app-
>user->identity->role, [

UserRole::STUDENT,UserRole::METODIST,UserRole::METODIST_NMV,UserRole::METODIST_DEC,
UserRole::GUEST
                                ])){
                                    throw new
\yii\web\ForbiddenHttpException(sprintf('You don\'t have rights to make this
request.', $action));

                                }
        }
    }
```

Тут я заборонила усім ролям, окрім адміністратора, виконувати будь-які дії із таблицею користувачів.

У першій умові я вказала які саме методи запитів я хочу обмежити, а у другій – вказую роль, для якої будуть не доступні ці методи, та потім просто повертаю помилку.

Більш детальний приклад контролера. (Додаток Г)

## 3.5 Опис реалізованих запитів

Тут я опишу які саме методи запитів можна робити до цього REST API та що вони будуть означати.

Види запитів:

1) GET /api/:modelName/ - отримання списку усіх рядків
2) GET /api/:modelName/index - отримання списку усіх рядків

3) GET /:modelName/view? - отримання списку усіх рядків, які відповідають заданим параметрам

Для всіх запитів метода GET можна зробити, щоб нам виводило додаткові поля моделі за нашим бажанням. Для цього у Query Params запиту потрібно додати пару, де ключ буде “expand”, а значення буде назва поля, яке ми бажаємо отримати. Якщо ми хочемо отримати декілька додаткових полей, то потрібно просто написати їх через кому. Наприклад, ми хочемо для всіх користувачів вивести ще додатково їх пароль та токен автентифікації. Це буде пати наступний вигляд:

GET /api/user?expand=password, access_token

Якщо ми хочемо зробити фільтрацію за певними критеріями, то нам потрібно у Query Params запиту внести пару або пари ключ-значення, де ключем буде назва поля, за яким буде здійснюватися фільтрація, а значенням буде значення цього поля. Якщо ми хочемо ввести декілька параметрів, то потрібно писати їх через знак “&”. Наприклад, якщо ми хочемо знайти студента із заданим айді та кодом, то це буде виглядати наступним чином:

GET /api/student/view?id=66928&code=204378

4) POST /api/:modelName/create – створює новий рядок

Також потрібно додати заголовок запиту, де ключем буде “Content-Type”, а значенням – “application/json” та у тілі запиту написати об’єкт, який ми хочемо створити у форматі JSON об’єкту. Так як айді у нас є автоінкремент, то його не потрібно прописувати у тілі. Наприклад, ми хочемо створити нову новину, тому запиту буде виглядати наступним чином:

POST /api/news/create

Тіло запиту:

```
{"content": "Нова новина"}
```

5) PUT /api/:modelName/update?id= - змінює інформацію рядка

У значення айді нам потрібно вписати айді того рядка, який ми хочемо змінити. Також потрібно додати заголовок запиту, де ключем буде “Content-Type”, а значенням – “application/json” та у тілі запиту написати поля, які ми хочемо змінити у форматі JSON об’єкту. Наприклад, ми хочемо змінити дані студента, тоді це буде виглядати наступним чином:

PUT /api/student/update?id=66928

Тіло запиту:


```
{

        "code": 204378,
        "name": "Іван Джміль",
        "status": 1,
        "ukma_email": "p.adamska@ukma.edu.ua",
        "format": 2015,
        "year": 4,
        "year_enter": 2017,
        "birth_date": "2000-05-02",
        "date": "2017-09-01",
        "date_end": "2021-06-28",
        "all_credits": 142
}
```


6) DELETE /api/:modelName/delete?id= - видаляє рядок із заданим айді

У значення айді нам потрібно вписати айді того рядка, який ми хочемо видалити. Наприклад, якщо ми хочемо видалити новину із айді 10, то це буде мати такий вигляд:

DELETE /api/news/delete?id=10

Типи modelName”:

1. course - відповідає таблиці “course”
2. coursegroup - відповідає таблиці “course_group”
3. courseregister - відповідає таблиці “course_register”
4. courseseason - відповідає таблиці “course_season”

5. coursespeciality - відповідає таблиці “course_speciality”
6. faculty - відповідає таблиці “faculty”
7. metodist - відповідає таблиці “metodist”
8. news - відповідає таблиці “news”
9. speciality - відповідає таблиці “speciality”
10. student - відповідає таблиці “student
11. subfactory - відповідає таблиці “subfaculty”
12. subrank – відповідає таблиці “subrank
13. unenrollrequest – відповідає таблиці “unenroll_request”
14. unenrollrequestcomment – відповідає таблиці “unenroll_request_comment”
15. user - відповідає таблиці “_user”

## 3.6 Тестування програми та результати виконання

Тестування REST API проводилось за допомогою Postman [10]. Postman – це інструмент розробки програмного забезпечення. Він дозволяє користувачам дуже зручно тестувати свій API.

Вхідні дані: запити клієнта.

Вихідні дані: відповідь сервера на запити.

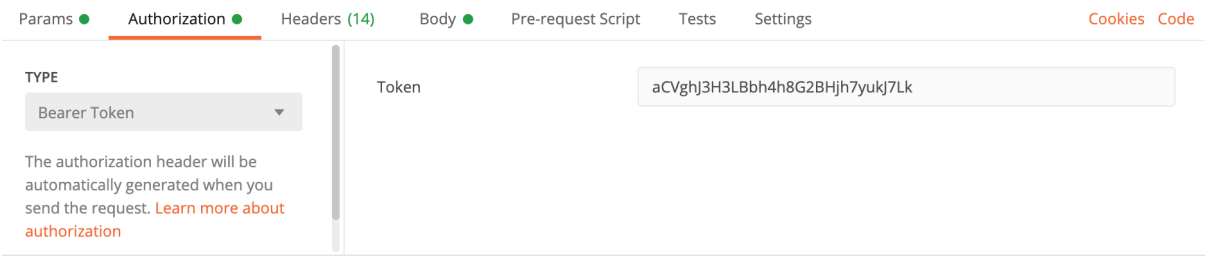
Тестування автентифікації за токеном:

Запит: http://localhost:8080/api/news

Якщо не вводити токен, то сервер поверне наступне:


```
{
    "name": "Unauthorized",
    "message": "Your request was made with invalid credentials.",
    "code": 0,
    "status": 401,
    "type": "yii\\web\\UnauthorizedHttpException"
}
```

У вкладці Authorization я обрала тип Bearer Token та у полі Token вписала значення свого токену:




Відповідь серверу після авторизації:


```
[
    {
        "id": 9,
        "date": "2017-03-21 13:40:15",
        "content": "21 березня 2017 року розпочався великий запис студентів бакалаврату та магістеріуму на вибіркові
            дисципліни.\r\nУ випадку виявлення проблем прохання звертатися до служби підтримки.\r\nE-mail:
            oletsky@ukr.net (Олецький О.В.), slavon07@gmail.com (Горборуков В.В.)."
    },
    {
        "id": 11,
        "date": "2017-03-27 09:49:17",
        "content": "<b>ОГОЛОШЕННЯ:</b> <br>\r\n\r\nДисципліни кафедри \"Могилянська школа журналістики\" призначені
            тільки для студентів спеціальності журналістика. \r\n\r\nСтуденти з інших спеціальностей, які обрали
            дисципліни з цієї кафедри мають скоригувати свої індивідуальні плани та не записуватись на них.
            (Записуватись на ці дисципліни мають тільки, ті для кого вона базова, професійно-орієнтована)"
    },
    {
        "id": 12,
        "date": "2017-03-29 09:34:28",
        "content": "<b>Увага!</b>\r\n<br/>\r\nСтуденти 1, 2, 3 р.н , які з різних причин не змогли записатися через
            систему\r\n\r\nе-запису на вибіркову дисципліну\r\n\r\nФізичне виховання (вдосконалення)\r\n\r\nпросимо
            терміново звернутися на кафедру фізичного виховання\r\n\r\n(3-303 )\r\n\r\nдля вирішення цієї проблеми."
    },
```


Тестування методу GET:

Запит: http://localhost:8080/api/news

Відповідь серверу:

```
[
    {
        "id": 9,
        "date": "2017-03-21 13:40:15",
        "content": "21 березня 2017 року розпочався великий запис студентів бакалаврату та магістеріуму на вибіркові
            дисципліни.\r\nУ випадку виявлення проблем прохання звертатися до служби підтримки.\r\nE-mail:
            oletsky@ukr.net (Олецький О.В.), slavon07@gmail.com (Горборуков В.В.)."
    },
    {
        "id": 11,
        "date": "2017-03-27 09:49:17",
        "content": "<b>ОГОЛОШЕННЯ:</b> <br>\r\n\r\nДисципліни кафедри \"Могилянська школа журналістики\" призначені
            тільки для студентів спеціальності журналістика. \r\n\r\nСтуденти з інших спеціальностей, які обрали
            дисципліни з цієї кафедри мають скоригувати свої індивідуальні плани та не записуватись на них.
            (Записуватись на ці дисципліни мають тільки, ті для кого вона базова, професійно-орієнтована)"
    },
    {
        "id": 12,
        "date": "2017-03-29 09:34:28",
        "content": "<b>Увага!</b>\r\n<br/>\r\nСтуденти 1, 2, 3 р.н , які з різних причин не змогли записатися через
            систему\r\n\r\ne-запису на вибіркову дисципліну\r\n\r\nФізичне виховання (вдосконалення)\r\n\r\nпросимо
            терміново звернутися на кафедру фізичного виховання\r\n\r\n(3-303 )\r\n\r\nдля вирішення цієї проблеми."
    },
```


Запит: http://localhost:8080/api/news/view?id=9

Відповідь серверу:


```
{
    "id": 9,
    "date": "2017-03-21 13:40:15",
    "content": "21 березня 2017 року розпочався великий запис студентів бакалаврату
та магістеріуму на вибіркові дисципліни.\r\nУ випадку виявлення проблем прохання
звертатися до служби підтримки.\r\nE-mail: oletsky@ukr.net (Олецький О.В.),
slavon07@gmail.com (Горборуков В.В.)."
}
```


Запит: http://localhost:8080/api/student/view?id=66930&code=86184

Відповідь серверу:


```
{
    "id": 66930,
    "code": 86194,
    "name": "Аксенова Анастасія Віталіївна",
    "status": 1,
    "ukma_email": "a.aksenova@ukma.edu.ua",
    "format": 2015,
    "year": 4,
    "year_enter": 2017,
    "birth_date": "2000-03-13",
    "date": "2017-09-01",
    "date_end": "2021-06-28",
    "all_credits": 182.5,
    "spec_id": 1467,
    "speciality": {
        "id": 1467,
        "name": "Біологія",
        "faculty_id": 125,
        "level": 1
    },
    "faculty": {
        "id": 125,
        "name": "Факультет природничих наук"
```

Тестування методу POST:

Запит: http://localhost:8080/api/news/create

Тіло запиту:

```
{
    "content":"Hello! NEWS"
}
```

Відповідь серверу:

```
{
    "content": "Hello! NEWS",
    "id": 100004
}
```

Для перевірки зробимо запит GET із цим айді

Запит для перевірки: http://localhost:8080/api/news/view?id=100004

Відповідь серверу:

```
{
    "id": 100004,
    "date": "2021-05-15 23:52:21",
    "content": "Hello! NEWS"
}
```

Отже, такий рядок існує, запит був успішним.

Тестування запиту PUT:

Запит: http://localhost:8080/api/news/update?id=100004

Тіло запиту:

```
{
    "content":"Hello! UPDATED NEWS"
}
```

Відповідь серверу:

```
{
    "id": 100004,
    "date": "2021-05-15 23:52:21",
    "content": "Hello! UPDATED NEWS"
}
```

Для перевірки зробимо запит GET із цим айді

Запит для перевірки: http://localhost:8080/api/news/view?id=100004

Відповідь серверу:

```
{
    "id": 100004,
    "date": "2021-05-15 23:52:21",
    "content": "Hello! UPDATED NEWS"
}
```


Отже, наші зміни зберіглися, запит був успішним.

Тестування методу DELETE:

Запит: http://localhost:8080/api/news/delete?id=100004

Відповідь серверу: пустий рядок

Для перевірки зробимо запит GET із цим айді

Запит для перевірки: http://localhost:8080/api/news/view?id=100004

Відповідь серверу:


```
{
    "name": "Not Found",
    "message": "Object not found: 100004",
    "code": 0,
    "status": 404,
    "type": "yii\\web\\NotFoundHttpException"
}
```


Отже, такого об’єкту не існує, запит був успішним.

Тестування обмеження ролей:

Нам відомо, що користувачів може переглядати тільки адміністратор, тому будемо тестувати це саме на цих запитах.

Авторизувавшись за методиста роблю запит:

Запит: http://localhost:8080/api/user

Відповідь серверу:


```
{
    "name": "Forbidden",
    "message": "You don't have rights to make this request.",
    "code": 0,
    "status": 403,
    "type": "yii\\web\\ForbiddenHttpException"
}
```

Роблю той самий запит, але все авторизуюсь за адміністратора:

Відповідь серверу:


```
[
    {
        "id": 68816,
        "login": "admin",
        "role": "admin",
        "name": "Вовк Наталя Євгенівна",
        "email": "vovknj@ukma.edu.ua"
    },
    {
        "id": 68817,
        "login": "dsv09",
        "role": "metodist_NMV",
        "name": "Думанська Світлана Володимирівна",
        "email": "s.dumanska@ukma.edu.ua"
    },
    {
        "id": 68818,
        "login": "olga",
        "role": "admin",
        "name": "Корольова Ольга Олегівна",
        "email": "korolyovaoo2@ukma.edu.ua"
    },
    {
```

# Висновки

В ході виконання роботи було розглянуто та ґрунтовно проаналізовано предметну область, детально вивчено та досліджено REST архітектуру та, як результат, розроблено RESTful API.

Було реалізовано усі задачі із технічного завдання, а саме: розподілення системи на ролі та присутність токен-аутентифікації, можливість отримувати, змінювати, додавати та видаляти дані у базі даних, регулювання кількості інформації, яку потрібно отримувати.

Усі обрані мною засоби розробки задовільнили технічні вимоги. Обрана мова та фреймворк дали можливість виконати поставлені задачі коректно та зручно. А обрані середовища розробки допомогли розробити програму зручно, вказуючи на помилки, контролювати процеси, керувати та відстежувати версії, зручно оперувати із інформацією у базі даних.

Був розроблений REST API для системи автоматизованого запису на дисципліни, який виконує усі поставлені задачі та слугує гарним компонентом для майбутнього масштабування системи. Отже, можна зробити висновок, що поставлена задача була виконана.

Додаток А

Course:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді дисципліни у системі |
| sub_doc | int | CDOC дисципліни |
| name | string | Назва дисципліни |
| type | int | Тип дисципліни |
| academic_year | Int | Рекомендований рік вивчення |
| format | Int | Формат навчання |
| year | Int | Рік навчання |
| hours | Int | Кількість годин |
| credits | double | Кількість кредитів ЄКТС |
| annotations | string | Опис дисципліни |
| chair_id | int | Айді кафедри |
| lever | string | Навчальний рівень |
| teacher | string | ПІБ викладача |
| selected | string | Чи рекомендовано |
| stud_limit | int | Максимальна кількість студентів |
| status_happened | string | Статус дисципліни |
| status_enrollment | string | Статус запису |
| max_stud | int | Максимальна кількість студентів в групі |
| min_stud | int | Мінімальна кількість студентів в групі |
| max_group | int | Максимальна кількість груп |
| actual_group | int | Кількість груп |

| | | |
|---|---|---|
| stud_count | int | Кількість зареєстрованих студентів |
| common_groups_count | int | Кількість основних груп |
| reserve_groups_count | int | Кількість груп резерву |
| created_at | string | Час створення |
| updated_at | string | Час оновлення |

CourseGroup:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді групи у системі |
| course_cdoc | int | CDOC дисципліни |
| number | int | Номер |
| status | int | Статус |
| max_count | Int | Максимальна кількість студентів |
| created_at | string | Час створення |
| updated_at | string | Час оновлення |

CourseRegister:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді запису у системі |
| user_code | int | Номер користувача |
| sub_doc | int | CDOC дисципліни |
| Ssmester_numb | int | Номер семестру |
| reg_type | string | Тип реєстрації |
| enrolled | string | Чи зареєстрован студент |
| deleted | int | Чи видален студент |

| | | |
|---|---|---|
| primusovo | int | Чи примусово зареєстрован студент |
| group_num | int | Номер групи |
| created_at | string | Час створення |
| updated_at | string | Час оновлення |

CourseSeason:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| course_cdoc | int | CDOC дисципліни |
| season | int | Сезон |
| hours_lecture | int | Години на лекції |
| hours_seminar | int | Години на семінари |
| hours_workshop | int | Години на практичні заняття |
| term_credits | decimal | Кредити за семестр |
| weekly_hours | int | Години навантаження на тиждень |
| exam_form | int | Форма семестрового контролю |

CourseSpeciality:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| course_cdoc | int | CDOC дисципліни |
| speciality_id | int | Айді спеціальності у системі |
| type | string | Тип запису |
| rank | string | Другий ранг |
| subrank_id | int | Третій ранг |

Faculty:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|

| | | |
|---|---|---|
| id | int | Айді факультету |
| name | string | Назва факультету |

News:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді новини |
| date | timestamp | Дата публікації новини |
| content | string | Текст новини |

Speciality:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді спеціальності |
| name | string | Назва спеціальності |
| faculty_id | int | Айді факультету спецівльності |
| level | int | Навчальний рівень |

Student:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді студента |
| code | int | Код студента |
| name | string | Ім’я студента |
| status | int | Статус |
| ukma_email | string | Електронна пошта |
| format | int | Формат навчання |
| year | int | Рік навчання |
| year_enter | int | Рік вступу |
| birth_date | date | Дата народження |
| date | date | Дата початку навчання |

| | | |
|---|---|---|
| date_end | date | Дата закінчення навчання |
| all_credits | double | Кількість кредитів ЄКТС |
| spec_id | int | Айді спеціальності |
| created_at | int | Час створення |
| updated_at | int | Час оновлення |

Subfaculty:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді кафедри |
| name | string | Назва кафедри |
| faculty_id | int | Айді факультету кафедри |

Subrank:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді третього рангу |
| name | string | Назва третього рангу |

UnenrollRequest:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді заявки |
| student_code | int | Код студента |
| unenroll_course_cdoc | int | Код курсу |
| collision_course_cdoc | int | Код курсу, з яким збіг |
| collision_course_day | int | День тижня курсу, з яким збіг |
| collision_course_time | time | Час збігу |
| status_collision | int | Статус збігу |
| status_decision | Int | Статус рішення |
| created_at | int | Час створення |

| | | |
|---|---|---|
| updated_at | int | Час оновлення |

UnenrollRequestComment:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді коментаря |
| unenroll_request_id | int | Айді заявки |
| user_id | int | Айді користувача |
| text | string | Текст |
| created_at | int | Час створення |
| updated_at | int | Час оновлення |

Metodist:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді методиста |
| name | string | Ім’я методиста |
| code | string | Код методиста |
| chair_id | int | Айді кафедри |
| faculty_id | int | Айді факультета |
| faculty_name | string | Ім’я факультета |
| chair_name | string | Ім’я кафедри |

User:

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| id | int | Айді користувача |
| login | string | Логін користувача |
| password | string | Пароль користуваці |
| auth_key | string | Ключ автентифікації |
| access_token | string | Токен доступу |

| | | |
|---|---|---|
| role | string | Роль у системі |
| link | Int | Посилання |
| code | string | Код користувача |
| name | string | Ім’я користувача |
| email | string | Електронна пошта |
| confirmed | string | Чи підтверджений |
| confirmed_mail | int | Чи підтверджена пошла |
| activehash | string | Дійсний хеш |
| login_old | string | Старий логін |

Додаток Б


my_ukma_dev _user
id : int(11)
login : varchar(255)
password : varchar(255)
auth_key : varchar(32)
access_token : text
role : varchar(50)
link : int(11)
code : varchar(50)
name : varchar(255)
email : varchar(255)
telegram_id : int(11)
telegram_name : varchar(255)
telegram_username : varchar(255)
telegram_photo_url : varchar(255)
confirmed : varchar(10)
confirmed_mail : int(11)
activehash : varchar(255)
login_old : varchar(50)
dip_ser : varchar(15)
dip_num : varchar(20)
p_code : int(11)
entered : int(11)
my_ukma_dev speciality
id : int(11)
name : varchar(255)
faculty_id : int(11)
level : tinyint(3)
my_ukma_dev course_speciality
course_cdoc : int(11)
speciality_id : int(11)
type : varchar(255)
rank : decimal(2,1)
subrank_id : int(11) unsigned
my_ukma_dev subrank
id : int(10) unsigned
name : varchar(255)
my_ukma_dev course_register
id : int(11)
user_code : int(11)
sub_code : int(11)
semestr_numb : int(11)
reg_type : varchar(10)
enrolled : varchar(10)
deleted : int(11)
primusovo : int(11)
cur_credit : double
group_num : int(11)
opt_group : int(11)
created_at : int(11) unsigned
updated_at : int(11) unsigned
my_ukma_dev course_group
id : int(11)
course_cdoc : int(11)
number : smallint(6) unsigned
status : tinyint(3)
max_count : smallint(6) unsigned
created_at : int(11) unsigned
updated_at : int(11) unsigned
my_ukma_dev course_term
course_cdoc : int(11)
term : tinyint(3) unsigned
my_ukma_dev course
id : int(11)
sub_cdoc : int(11)
name : varchar(255)
type : tinyint(3)
academic_year : int(4)
format : smallint(6)
year : smallint(6)
hours : smallint(6)
credits : double
annotation : text
chair_id : int(11)
level : tinyint(3)
teacher : varchar(255)
selected : tinyint(3)
stud_limit : smallint(6)
status : varchar(50)
status_happened : varchar(255)
status_enrollment : varchar(255)
max_stud : smallint(6)
min_stud : smallint(6)
max_group : smallint(6)
actual_group : smallint(6)
del_group : smallint(6)
stud_count : int(11)
common_groups_count : smallint(6)
reserve_groups_count : smallint(6)
created_at : int(11) unsigned
updated_at : int(11) unsigned
my_ukma_dev course_season
course_cdoc : int(11)
season : tinyint(3) unsigned
hours_lecture : smallint(6) unsigned
hours_seminar : smallint(6) unsigned
hours_workshop : smallint(6) unsigned
term_credits : decimal(3,1) unsigned
weekly_hours : tinyint(3) unsigned
exam_form : tinyint(3) unsigned
my_ukma_dev faculty
id : int(11)
name : varchar(255)
my_ukma_dev news
id : int(11)
date : timestamp
content : mediumtext
my_ukma_dev subfaculty
id : int(11)
name : varchar(255)
faculty_id : int(11)
my_ukma_dev unenroll_request
id : int(10) unsigned
student_code : int(11)
unenroll_course_cdoc : int(11)
collision_course_cdoc : int(11)
collision_course_day : tinyint(3)
collision_course_time : time
status_collision : tinyint(3)
status_decision : tinyint(3)
created_at : int(11) unsigned
updated_at : int(11) unsigned
my_ukma_dev unenroll_request_comment
id : int(10) unsigned
unenroll_request_id : int(11) unsigned
user_id : int(11)
text : text
created_at : int(11) unsigned
updated_at : int(11) unsigned
my_ukma_dev _metodist
id : int(11)
name : varchar(100)
code : varchar(50)
role : varchar(50)
chair_id : int(11)
faculty_id : int(11)
faculty_name : varchar(255)
chair_name : varchar(255)
my_ukma_dev student
id : int(11)
code : int(11)
name : varchar(255)
status : tinyint(3)
ukma_email : varchar(255)
format : smallint(6)
year : int(11)
year_enter : int(11)
birth_date : date
date : date
date_end : date
all_credits : double
spec_id : int(11)
created_at : int(11) unsigned
updated_at : int(11) unsigned

Додаток В

```php
<?php
namespace app\api\models;

use Yii;
use yii\behaviors\TimestampBehavior;
use yii\db\Expression;
use yii\helpers\ArrayHelper;
use yii\db\ActiveRecord;
use app\models\CourseRegister;
class Course extends \app\models\Course{

    public function fields()
    {
        return ['id','sub_cdoc','name','type','academic_year',
                'format','year','hours','credits',
                'annotation','teacher',
                'stud_limit','status', 'status_happened',
                'status_enrollment','max_stud','min_stud','max_group',
                'actual_group','del_group','stud_count','common_groups_count',

'reserve_groups_count','courseSeasonsList','courseSpecialitiesList',
                'courseGroups','subfacultyName'
            ];
    }

    public function extraFields()
    {
        return ['chair_id','level','selected','courseTerms',
        'courseSpecialities','realMaxStudent','subfaculty','
        courseSeasons','minSeasonValue','created_at','updated_at'];
    }


    public function getStudents()
    {
        return $this->hasMany(Studentshort::class, ['code' => 'user_code'])-
>viaTable(CourseRegister::tableName(), ['sub_code' => 'sub_cdoc']);
    }

}
```

Додаток Г

```php
<?php

namespace app\api\controllers;

use Yii;
use yii\rest\ActiveController;
use app\helpers\BehaviorsFromParamsHelper;
use yii\filters\auth\HttpBearerAuth;
use app\models\enums\UserRole;

class CourseController extends  ActiveController
{
    public $modelClass = 'app\api\models\Course';


    public function behaviors(){
        $behaviors = parent::behaviors();

        $behaviors['authenticator'] = [
            'class' => HttpBearerAuth::class
        ];

        return $behaviors;
    }

    public function checkAccess($action, $model = null, $params = [])
    {
        if ($action === 'view'||$action === 'update' || $action === 'delete'||
$action === 'create') {
            if (!empty(Yii::$app->user->identity->role) && in_array(Yii::$app-
>user->identity->role, [
                                    UserRole::STUDENT,UserRole::GUEST
                                ])){
                                    throw new
\yii\web\ForbiddenHttpException(sprintf('You don\'t have rights to make this
request.', $action));

                                }
        }
    }

}
```

Список використанної літератури

[1] Dissertation of Roy Thomas Fielding
https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#sec_5_1
[2] Стаття про мову PHP https://en.wikipedia.org/wiki/PHP
[3] Стаття про фреймворк php yii2
https://www.yiiframework.com/doc/guide/2.0/en/intro-yii
[4] API Documentation Yii Active Record
https://www.yiiframework.com/doc/guide/2.0/en/db-active-record
[5] API Documentation Yii Active Query
https://www.yiiframework.com/doc/api/2.0/yii-db-activequery
[6] https://www.yiiframework.com/doc/guide/2.0/en/rest-resources
[7] API Documentation Yii Active Controller
https://www.yiiframework.com/doc/api/2.0/yii-rest-activecontroller
[8] HttpBearerAuth https://www.yiiframework.com/doc/api/2.0/yii-filters-auth-httpbearerauth
[9] API Documentation Yii Authentication
https://www.yiiframework.com/doc/guide/2.0/en/rest-authentication
[10] Postman https://www.postman.com