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

РОЗРОБКА API ДЛЯ СИСТЕМИ ЗАПИСУ НА ВИБІРКОВІ ДИСЦИПЛІНИ

Текстова частина до курсової роботи

Виконала студентка 4 курсу
БП «Інженерія програмного забезпечення»
Андрусів Соломія Ігорівна

Керівник курсової роботи
старший викладач
Вовк Наталія Євгенівна

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем,

доцент, к. ф.-м. н. О. П. Жежерун

_____

(підпис)

“____”_____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студента Андрусів Соломії Ігорівни факультету інформатики 4 курсу

Тема: Розробка API для системи запису на вибіркові дисципліни

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Календарний план

Зміст

Перелік умовних позначень

Вступ

Розділ 1 Аналіз предметної області. Постановка завдання курсової роботи

Розділ 2 Теоретичні відомості

Розділ 3 Опис розробки програмного продукту

Висновки

Перелік використаних джерел

Дата видачі “___” _____ 2021 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи

Тема: Розробка API для системи запису на вибіркові дисципліни

Календарний план виконання роботи:

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
2.1.	Отримання завдання на курсову роботу.	10.10.2020	
2.2.	Ознайомлення з існуючою інформацією по темі	05.11.2020	
2.3.	Ознайомлення з існуючими системами-аналогами роботи	30.11.2020	
2.4.	Початок створення практичної частини	15.02.2021	
2.5.	Початок написання теоретичної частини	15.03.2021	
2.6.	Подання проміжної версії практичної частини	29.03.2021	
2.7.	Аналіз практичної частини; її коригування	31.03.2021	
2.8.	Остаточне завершення написання теоретичної частини роботи та розробки практичної частини; коригування	01.04.2021-09.04.2021	
2.9.	Створення презентації	10.04.2021	
2.10	Захист курсової роботи	19.04.2021-25.04.2021	

Студент Андрусів С.І.

Керівник Вовк Н.Є.

“_____” _____

ЗМІСТ

Перелік термінів та умовних позначень

Модуль – блок коду, логічно відокремлений від інших, зазвичай знаходиться в окремому файлі.

Бібліотека – це сукупність попередньо скомпільованих модулів коду, які виконують певну функцію. Їх можна використовувати за потреби при розробці власних програм.

Фреймворк – це платформа для розробки програм, призначена для спрощення процесу розробки. Може включати в себе заздалегідь визначені класи та функції та має свою архітектуру.

API (application programming interface) - набір програмного коду, що забезпечує передачу даних між програмними продуктами та містить умови передачі цих даних. В даній роботі API реалізовано з використанням RESTful-архітектури, він здійснює передачу даних між базою даних САЗу та клієнтською частиною програми KMA Schedule.

САЗ (система автозапису) – веб-сайт НаУКМА, на якому знаходиться уся основна інформація, необхідна для студентів університету, їх індивідуальні плани, розклади, оголошення, також тут проводиться запис на дисципліни та у групи.[1]

Темплейт – синонім до слова шаблон, зазвичай уже містить весь необхідний код, що може бути модифікований за потреби. Тут темплейт використовується також у значенні повторюваного блоку HTML-коду.

Рендеринг – процес парсингу HTML-коду та перетворення HTML-об’єкта у графічний.

Фронт-енд –клієнтська частина веб-застосунку.

ВСТУП

KMA Schedule – додаток, створений на основі бази даних системи автозапису на вибіркові дисципліни НаУКМА, призначений надавати доступ до триместрових розкладів занять в Національному університеті «Києво-Могилянська академія». На момент проведення дослідження розклади з дисциплін в університеті створюються методистами деканатів відповідних факультетів та розміщуються на сайті САЗу у вигляді файлів. У кожного факультету свій формат розкладів, а для перегляду змін в розкладах студентам необхідно завантажувати ці файли. Тому система, яка б дозволяла здійснювати усі операції з розкладами безпосередньо на одному веб-сайті, без використання Google Docs як сховища усіх файлів, є актуальною на даний момент.

Основною метою роботи є спрощення процесу створення розкладів для методистів НаУКМА шляхом його автоматизації та узагальнення формату і вигляду розкладів для усіх факультетів. Також, студентам буде надана можливість переглядати зведений розклад за індивідуальним навчальним планом, щоб полегшити формування власного розкладу на етапі запису в групи.

Завданням курсової роботи є розробка API, що буде взаємодіяти з базою даних my.ukma.edu.ua та написання клієнтської частини зі зручним та інтуїтивно-зрозумілим інтерфейсом, яка буде надавати користувачам можливість перегляду розкладів, виконання операцій над ними та завантаження їх у Excel-форматі.

Об’єктом дослідження є система створення розкладів в НаУКМА на даний момент, відмінності у форматі розкладів на різних факультетах та способи передачі їх студентам, а також проблеми пошуку вибіркових дисциплін у розкладах інших факультетів.

Предметом дослідження є способи вдосконалення та автоматизації процесу створення розкладів.

Основною мовою програмування при розробці став JavaScript. Серверну частину реалізовано на Node.js з підтримкою архітектури REST API, клієнтську частину – з використанням фреймворку Vue.js та графічної бібліотеки Bootstrap. Всі дані містяться у базі даних MySQL.

Курсова робота складається з вступу, трьох основних розділів та висновків.

У першому розділі «Аналіз предметної області. Постановка завдання курсової роботи» розглядається стан питання на момент початку дослідження та визначаються кроки для реалізації проекту.

У другому розділі «Теоретичні відомості» зібрана інформація, необхідна для кращого розуміння роботи результуючої програми та причин вибору тих чи інших шляхів розробки.

У третьому розділі «Опис розробки програмного продукту» вказані деталі реалізації веб-застосунку.

У висновках підбиваються підсумки результатів дослідження.

Посилання на наукові роботи та інші джерела інформації містяться у списку використаної літератури. Всі матеріали, що доповнюють текст, зокрема зображення, схеми та програмний код знаходяться у «Додатках».

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

1.1. Аналіз сучасного стану питання та обґрунтування теми

В НаУКМА є шість факультетів, у кожного з яких – свої спеціальності. В університеті є бакалаврська та магістерська навчальні програми, тривалістю 4 та 2 навчальні роки відповідно. На кожен з трьох триместрів для всіх спеціальностей згідно з їх навчальним роком потрібно підготувати як мінімум два розклади – розклад занять на початку триместру та розклад заліково-екзаменаційної сесії, що надається зазвичай після 8 навчального тижня.

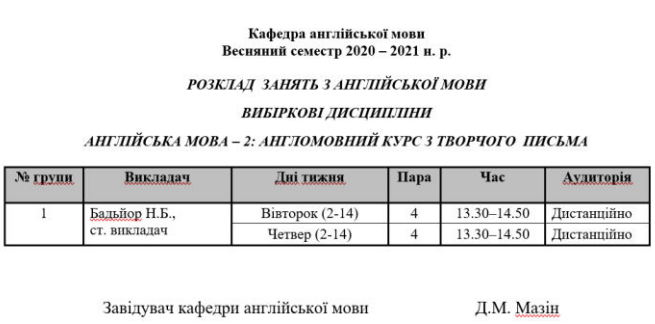
Розклади занять, що надаються на початку навчального триместру, розміщені на сайті САЗу у вигляді посилань на файли Google Docs, доступні для завантаження та перегляду. В залежності від факультету файли мають різні розширення, загалом їх чотири види - .xls, .xlsx, .doc та .docx. Також варто зазначити, що не всі файли відкриваються у режимі перегляду. На факультеті економіки для всіх спеціальностей одного навчального року прив’язано один і той же файл розкладу(збігається навіть назва), у якому спеціальність, для якої проводиться предмет, вказана у дужках після назви дисципліни. Також більшість розкладів мають неінформативні назви файлу – 1, 2, 3, 4, 1, 2 до кожного навчального року відповідно. В свою чергу, на основному сайті НаУКМА (ukma.edu.ua) також розміщені розклади, але у pdf-форматі.

Розклади заліково-екзаменаційної сесії надаються деканатом старостам спеціальностей, які в свою чергу поширюють їх серед студентів. На сайті САЗу немає інформації щодо цього виду розкладів, вони контролюються виключно деканатами відповідних факультетів.

Однією з особливостей НаУКМА є наявність вибіркових дисциплін. Зазвичай такі предмети можна знайти в розкладах спеціальностей, для

яких вони є професійно-орієнтованими, а якщо таких немає – тоді на одному з років навчання одного з розкладів відповідного факультету. Іноді доводиться витратити немало часу, щоб завантажити усі розклади локально та знайти серед них потрібний курс, а перегляд файлів не завжди працює, залежно від браузера та розширення документа.

Також, є предмети, які не містяться у таблицях розкладів. До прикладу, предмет «Англійська мова 2: Англомовний курс з творчого письма» знаходиться в окремій таблиці як додаток до розкладу 3 курсу спеціальності «Філологія (германські мови та література)» (*рис. 1.1*). Можу сказати на власному досвіді, що знайти його практично неможливо, якщо не знаєш, де шукати. Щоб дізнатись розклад цього курсу, я зверталася в деканат факультету гуманітарних наук в минулому навчальному році. Також, розклад даної дисципліни не відображається у телеграм-боті KMAScheduler, адже він не враховує предмети, що не знаходяться в основних таблицях розкладів.



Кафедра англійської мови
Весняний семестр 2020 – 2021 н. р.

РОЗКЛАД ЗАНЯТЬ З АНГЛІЙСЬКОЇ МОВИ

ВИБІРКОВІ ДИСЦИПЛІНИ

АНГЛІЙСЬКА МОВА – 2: АНГЛОМОВНИЙ КУРС З ТВОРЧОГО ПИСЬМА

№ групи	Викладач	Дні тижня	Пара	Час	Аудиторія
1	Бадьйор Н.Б., ст. викладач	Вівторок (2-14)	4	13.30–14.50	Дистанційно
		Четвер (2-14)	4	13.30–14.50	Дистанційно

Завідувач кафедри англійської мови Д.М. Мазін

Рисунок 1.1

Проект KMA Schedule спрямований в першу чергу на вирішення розбіжностей між форматами розкладів у різних факультетів та проблеми розміщення розкладів, що не входять до жодної спеціальності, як-от «Англомовний курс з творчого письма», а також на надання можливості зручного перегляду усіх розкладів в любому веб-браузері.

1.2. Аналіз предметної області

Для проведення дослідження мені надано тестову базу даних САЗу, що містить інформацію про факультети, кафедри та спеціальності, а також перелік курсів на 2020-2021 навчальний рік. У базі даних міститься зокрема така інформація про кожен курс, як унікальний код курсу, його назва, академічний рік, навчальна програма та навчальний рік, факультет та кафедра, яким належить предмет, викладач, кількість годин та кредитів. В окремих таблицях також знаходиться інформація про студентів та дисципліни, на які вони записані; про спеціальності, для яких курс є професійно-орієнтованим, нормативним чи вільного вибору; вид контролю для курсу під час заліково-екзаменаційної сесії. Переліченої інформації достатньо для того, щоб працювати з розкладами.

Проаналізувавши всі таблиці розкладів, що містяться на САЗі, виділила 3 основні типи розкладів, що будуть представлені у застосунку:

- розклад спеціальності – той, який надається кожному курсу студентів перед початком навчального триместру та можна завантажити на САЗі;
- розклад сесії – дати заліків, консультацій та екзаменів, що надаються старостам деканатом;
- розклад за кафедрою – варіант для таких дисциплін, як «Англомовний курс з творчого письма», що не мають своєї спеціальності, адже кожен курс в базі даних прив’язаний до кафедри.

Проглянувши версії розкладів усіх факультетів, я обрала за зразок для загального шаблону розкладу таблицю в Microsoft-Excel, як це зроблено на факультетах інформатики та правничих наук, з новішим розширенням .xlsx. Формат таблиць у всіх розкладах однаковий, містить стовпці «День», «Час», «Дисципліна, викладач», «Група», «Тижні», «Аудиторія». Для

розкладів сесії за зразком факультету інформатики буде додано стовпець «Вид контролю» та замінено колонку «Тижні» на «Дату».

Також, студентам буде надана можливість переглядати власний розклад на основі індивідуального плану з детальною інформацією про кожен курс, що мінімізує проблему пошуку студентами вибіркових дисциплін у розкладах інших факультетів.

1.3. Постановка завдання

Як було зазначено вище, у НаУКМА немає спільного шаблону для збереження розкладів. Тому ідеєю мого курсового проекту було зробити такий зразок та надати методистам можливість створювати розклади у веб-браузері з пошуком існуючих дисциплін.

Основними завданнями моєї курсової роботи є:

1. Дослідження технологій, що використовуються у веб-розробці та вибір із них достатньо популярних та широковживаних для реалізації необхідного функціоналу власного проекту.
2. Аналіз предметної області та всієї доступної інформації про курси і розклади в КМА, процес їх створення та передачі студентам, на основі яких формування вигляду основного шаблону таблиці розкладу та визначення їх основних типів.
3. Аналіз веб-сайту САЗу НаУКМА, а зокрема його графічного інтерфейсу. Використання цієї інформації як зразок для дизайну прототипу власного веб-застосунку, що буде візуально подібним до вже існуючого з метою спрощення користування для методистів та студентів.
4. Власне розробка застосунку, що включає в себе модифікацію наявної бази даних, написання API та реалізацію клієнтської

частини. Програма повинна задовольняти такі функціональні вимоги:

- поділ розкладів на три типи, зазначені у п. 1.2;
- перегляд розкладів у веб-браузері;
- завантаження розкладів у форматі .xlsx;
- відображення зведеного розкладу за індивідуальним планом на сторінці профілю студента;
- коли доступний розклад сесії, надання його аналогічно до попереднього на сторінці профілю студента;
- на сторінці профілю методиста показувати всі розклади факультету, якому належить методист;
- створення та видалення методистами розкладів власного факультету;
- редагування методистами будь-якого існуючого розкладу власного факультету;
- пошук та автодоповнення назви та коду дисципліни в процесі заповнення таблиці розкладу;
- надання методистам можливості закривати та відкривати розклад для перегляду студентам(тобто можливість створювати чорнові варіанти розкладу).

РОЗДІЛ 2. ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1. Загальні відомості про Vue.js

Робота більшості веб-сайтів базується на клієнт-серверній взаємодії. У цій схемі клієнт - пристрій кінцевого користувача, що має доступ до візуальної частини застосунку (перегляд сторінки в Інтернеті). Щоб на стороні клієнта відобразились нові дані, він відправляє запит на сервер, який в свою чергу надає клієнту нові дані.

У виконанні усіх процесів на стороні сервера є негативний аспект. Необхідність подолання кожним запитом шляху від клієнта до сервера і назад призводить до зайвих витрат часу. Щоб уникнути таких затримок, сьогодні розробники все більшу частину коду переносять на фронт-енд. Одним із важливих прикладів використання такого підходу є рендеринг динамічних веб-сторінок на стороні клієнта. В результаті міграції виконання процесів на фронт-енд, виникає необхідність структурувати увесь JavaScript код.[6]

Фреймворк Vue.js на даний момент є одним із трьох найпопулярніших фронтенд-фреймворків, на ряду з React та Angular, призначених для розширення можливостей клієнтської частини застосунку та зведення до мінімуму взаємодії з сервером. Він забезпечує користувача необхідними методами і компонентами, а також заздалегідь визначеною архітектурою програми. Одна з особливостей Vue полягає в його прогресивності, тобто його підтримку можна легко додати до будь якого вже існуючого проекту, розширивши цим його можливості. Також даний фреймворк легко інтегрується з будь-якими додатковими бібліотеками. [8]

Vue.js є реактивним фреймворком. Це означає, що при будь-якій зміні даних він автоматично змінює відображення інформації на сторінці. Vue.js є component-based фреймворком. Він дозволяє розділити весь код на

компоненти, що можуть використовуватись повторно. Кожен компонент яких містить власний темплейт HTML-елемента, може містити також блок JavaScript коду та CSS-класи. Останні можуть бути як локальними, так і глобальними, тобто доступними для всіх компонентів, вкладених в даний, незалежно від рівня вкладеності, якщо забрати з тегу <style> атрибут «scoped».

Кожен Vue-компонент має параметри, що відповідають за роботу з даними:

- data – функція, що повертає об'єкт, полями якого є деякі статичні дані компоненту, не є реактивними, тому нові поля в data не можуть добавлятись після рендерингу компонента;
- computed – об'єкт з даними, що потребують деяких обчислень чи змін та перераховуються тільки при зміні реактивної залежності;
- methods – тут знаходяться усі функції компонента;
- watch – спостерігає за зміною елементу з data; використовується, якщо при зміні одного елементу повинен змінюватись інший; зазвичай спостерігачі використовують, якщо немає можливості синхронізувати зміну даних через властивість computed;
- props – масив, що містить значення атрибутів, переданих в компонент з батьківського.
- emits – події, що викликаються для передачі даних з дочірнього компонента до батьківського.

Директиви також є однією з основних особливостей Vue.js. Це – спеціальні атрибути для HTML-тегів, що надає можливість керувати відображенням елементів шаблону, щоб здійснювати певні операції над ними. Основними директивами фреймфорку є:

- v-bind – призначений для динамічної прив'язки даних з <script> до HTML-елементів, може використовуватись з будь-якими існуючими атрибутами;

- v-if – рендерить елемент, тільки якщо він задовольняє вказаній умові;
- v-else – директива «спрацьовує», тільки якщо тег, розташований безпосередньо перед даним, містить директиву v-if, а елемент рендериться, якщо попередній елемент не задовільнив умові, вказаній у v-if;
- v-for – директива призначена для ітерації по масиву та циклічного створення однакових елементів, використовує також додатковий атрибут «key» з унікальними значеннями (значення повинні бути унікальними не в межах даного тегу, а в межах усього DOM); варто зазначити, що v-if v-for не можуть використовуватись в одному HTML-тегу;
- v-model – використовується для динамічної передачі даних між HTML-елементами та полями даних з data чи computed;
- v-on – зв’язує елемент з слухачем подій;
- v-once – рендерить елемент тільки один раз та потім не оновлює його при оновленні батьківського;
- v-show – показує та приховує елемент в залежності від умови;
- v-text – змінює textContent елементу.[2]

Кожен компонент має свій життєвий цикл – від моменту створення до моменту знищення. Vue надає функціонал для роботи з компонентом на різних його стадіях: так звані хуки життєвого циклу (lifecycle hooks):

- beforeCreate – перший метод, що викликається при зразу після ініціалізації компоненту, на цьому етапі ще немає доступу до даних;
- created – на цьому моменті відбувається ініціалізація усіх методів та властивостей компонента та спостереження за даними;
- beforeMount – викликається після компіляції темплейта, але перед створенням його графічного відображення;

- mounted – викликається зразу після включення новоствореного компоненту в DOM, є найбільш широковживаним хуком; варто зазначити, що перебування компоненту в даному стані не означає, що всі його дочірні компоненти також закінчили рендеринг;
- beforeUpdate – відбувається повторний рендеринг HTML-елементу та його порівняння з уже існуючим в DOM, при виявленні розбіжностей елемент в DOM оновлюється;
- updated – викликається після рендерингу змін;
- beforeDestroy – на цьому етапі видаляються усі методи, спостерігачі та внутрішні компоненти;
- destroyed – викликається, зразу після знищення елементу.[7]

2.2. Бібліотеки Vue: Vuex-store, Vue-router

За внутрішню маршрутизацію у Vue відповідає офіційна бібліотека Vue-router. Її використання надає застосунку ряд нових можливостей:

- створення SPA з переходом між вкладеними маршрутами;
- зручна анімація переходів між сторінками-компонентам Vue при зміні маршрутів;
- доступ до даних маршруту, зокрема параметрів url-адреси тощо;
- можливість настроювати режим прокрутки сторінки;
- спрощення навігації.[4]

Vuex-store - офіційна бібліотека з управління станом Vue-застосунку. Вона є глобальним сховищем даних на стороні клієнта, доступитись до яких можна з будь-якого компонента. Store містить правила, що вказують, яким способом можна змінити ці дані. Він складається з таких частин:

- state – стан даних в даний момент часу;
- getters – за потреби, можна повертати у компонент дані з стану;

- actions – дії, що відповідають за зв’язок з бекендом та запускають необхідні мутації.
- mutations – частина store, що відповідає за зміну стану даних, власне правила, що вказують, яким способом дані можуть бути змінені;
- modules – за потреби можна створити модулі з власним простором імен, що будуть складатися з тих же полів, що і основний store, та мають зв’язок один з одним через основний store.

Взаємодія компонентів зі Vuex-store відбувається наступним чином:

1. Компонент має нові/змінені дані, які необхідно записати у базу даних та відобразити на сторінці;
2. За допомогою команди dispatch, у компоненті викликається функція з блоку actions;
3. У функції відбувається надсилання запиту на бекенд та отримання результатів;
4. В цій же функції викликається необхідна мутація;
5. В мутації дані стану оновлюються;
6. Vue.js – реактивний, тому нові дані автоматично відобразяться на сторінці.[3]

Якщо потрібно змінити дані виключно на стороні клієнта, виконуються всі ті ж кроки, тільки без відправки запиту на бекенд.

2.3. Основні принципи архітектури REST API

RESTful API – програмний інтерфейс, що підтримує REST-архітектуру. Вона в свою чергу містить ряд правил та обмежень, відповідно до яких девайси «спілкуються» між собою у мережі через протокол http-протоколу. Він може бути впроваджений у застосунок багатьма шляхами.

Основними концепціями REST-архітектури є:

- Клієнт-серверний зв’язок – клієнтський застосунок робить запит на певну url-адресу, запит надсилається до сервера, який у свою чергу повертає необхідну інформацію чи рендерить HTML-сторінку.
- Передача даних здійснюється у невеликих об’ємах в стандартних форматах(XML, JSON тощо).
- Повинна бути реалізована підтримка кешування.
- Незалежність від мережевого прошарку – клієнт, відправляючи запит, не повинен знати, через скільки посередників проходить запит, перш ніж він досягне сервера.
- Не зберігає стан – два ідентичні запити, надіслані послідовно, повинні отримати однаковий результат.[9]

Запит(request) складається з таких компонентів:

- url на сервері, на який надсилається запит;
- http-заголовки – містять додаткову інформацію про запит, параметри передачі даних тощо;
- http-метод: get, put, post або delete;
- body – тіло запиту, в якому передаються дані в одному зі стандартних форматів.

Відповідь(response) містить такі поля:

- статус - 200, якщо запит успішний, та ряд інших, якщо у ході виконання запиту виникли проблеми;
- заголовки відповіді;
- тіло відповіді – дані у одному з вказаних вище форматів.

РОЗДІЛ 3. ОПИС РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

3.1. Аналіз технічного завдання

Веб-застосунок KMA Scheduler призначений для перегляду, створення та редагування розкладів занять в НаУКМА.

Застосунок підтримує два типи зареєстрованих користувачів: методистів деканату та студентів. Більшість інформації доступна для перегляду неавторизованим користувачам, зокрема:

- перегляд головної сторінки;
- список усіх доступних розкладів;
- перегляд будь-якого розкладу зі списку;
- завантаження будь-якого розкладу зі списку.

Студенти мають також можливість переглядати власний профіль, на якому розташовані:

- зведений розклад у формі календаря за індивідуальним планом студента, що включає в себе всі нормативні, професійно-орієнтовані та предмети вільного вибору, на які записаний студент в поточному триместрі у формі;
- детальна інформація про кожен з курсів, що відображається при наведенні курсору на дисципліну в таблиці;
- зведений розклад заліково-екзаменаційної сесії студента, якщо він уже доступний;
- повідомлення про те, що розклад сесії ще недоступний в іншому випадку.

Найбільший функціонал веб-застосунку доступний авторизованим методистам деканату. Вони мають можливість:

- переглядати весь список доступних для редагування розкладів у власному профілі;

- редагувати будь-який з розкладів власного факультету;
- створювати розклади трьох доступних типів: спеціальності, сесії та кафедри;
- видаляти будь-який розклад власного факультету;
- зберігати чорновий варіант розкладу, та відкривати і закривати його для студентів у списку в профілі.

Для спрощення користування сайтом, веб-застосунок надає методистам такі можливості:

- автоматична синхронізація спеціальностей в залежності від вибраної навчальної програми(бакалаврат, магістратура);
- автодоповнення при заповненні рядків таблиці курсів за назвою чи кодом, з вибірки усіх доступних дисциплін відповідно до вибраної програми, спеціальності, триместру та академічного року;
- в залежності від обраного предмету, також визначення зазначеної в базі даних кількості груп та представлення їх у випадаючому списку;

Веб-застосунок повинен також підтримувати зворотній зв’язок з користувачами, що виражається зокрема у виведенні повідомлень про успішну операцію та попереджень про помилку, а також відображенні відповідної анімації при загрузці даних.

3.2. Обґрунтування алгоритму та архітектура програми

Проект реалізовано у вигляді веб-сайту з двома типами користувачів. Він складається з трьох основних частин: клієнту, сервера та бази даних. Для виконання проекту мені було надано тестову базу даних САЗу, до якої в ході написання застосунку було додано дві таблиці. Детальніше модифікації з нею будуть розглянуті у п. 3.5.

Серверна частина являє собою RESTful API та написана на Node.js. Вона поділена на чотири модулі, що здійснюють запити до бази даних та повертають інформацію у відповідні модулі на стороні клієнта. Застосунок підтримує рендеринг динамічних сторінок на стороні клієнта, тому з бекенду на фронтенд відправляються об’єкти з відповідною інформацією, а не HTML-сторінки. Обидві частини застосунку написані на мові програмування JavaScript, з огляду на це передача даних між ними відбувається у форматі JSON.

При розробці клієнтської частини програми було використано популярний JavaScript фреймворк Vue.js. Його перевагами є наявність власного роутера Vue-router, що відповідає за переключення між сторінками, компонентів, на які розбивається увесь HTML-код та Vuex-store, що використовується як глобальне сховище даних на клієнтській частині.

Програма передбачає коректну навігацію між сторінками та відображення відповідних даних. User-flow веб-застосунку зображено на рисунку 3.1.

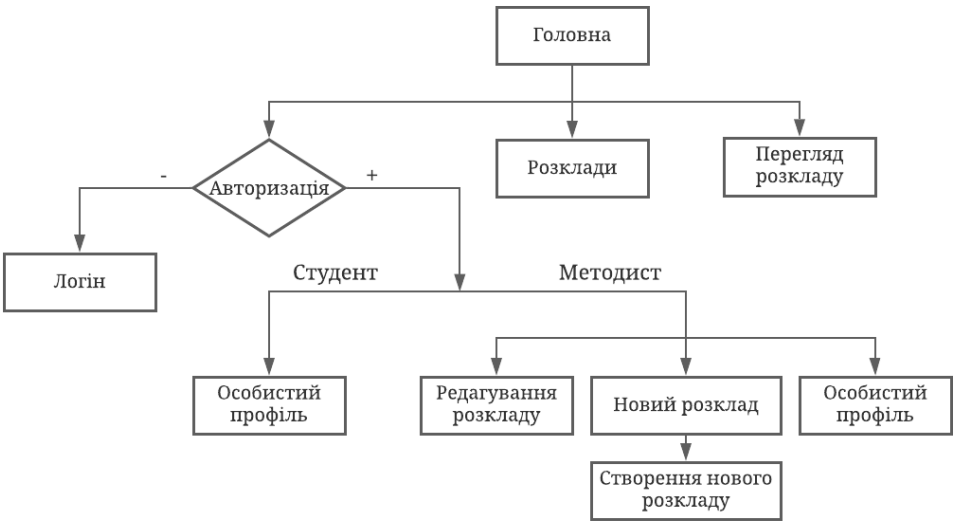


Рисунок 3.1 Блок-схема потоку користувача

Vue Router здійснює генерацію url-адрес та відображення відповідних сторінок. Об’єкти, зображені на блок-схемі, відповідають компонентам сторінок Vue Router:

- «Головна» (Main.vue) – сторінка, на якій міститься основна інформація про проект.
- «Розклади» (Schedules.vue) – відображається список усіх розкладів, розділений за типами, факультетами, спеціальностями та курсами аналогічно до такого ж списку на сайті САЗу.
- «Перегляд розкладу» (View.vue) – сторінка, що відображає власне інформацію про розклад та таблицю з датами і курсами.
- «Особистий профіль» (Profile.vue) – має різний вигляд залежно від типу користувача, містить мінімальну інформацію про користувача, зведений розклад для студентів і список усіх доступних для редагування розкладів методиста.
- «Редагування розкладу» (Edit.vue) – сторінка, в якій вся інформація про розклад та таблиця є доступними для редагування.
- «Новий розклад» (New.vue) – форма, що дає можливість вибрати тип розкладу, який необхідно створити.
- «Створення нового розкладу» (Create.vue) – в залежності від обраного типу на попередній сторінці, відображаються порожні поля даних про розклад та таблиця з відповідними стовпцями.
- «Логін» (Auth.vue) – сторінка авторизації, містить форму для входу в систему, в майбутніх планах також додати авторизацію через Office-365.

Відповідно до стандартної архітектури Vue-проекту, всі компоненти-сторінки знаходяться у папці views, а прив’язка їх до відповідних адрес відбувається у файлі router.js.

3.3. Засоби розробки

Проаналізувавши популярні технології розробки веб-проектів та структуру веб-сайту САЗу, було обрано наступні засоби для реалізації застосунку. Проект написаний на мові програмування JavaScript та складається з трьох основних частин – клієнтської, серверної та бази даних.

Для написання програмного коду було використано середовище розробки WebStorm від JetBrains, що забезпечене відповідними засобами для роботи з Node.js, фреймворком Vue.js та синхронізацією з системою контролю версій Git. Як сховище коду було використано GitHub. Розробка прототипу застосунку виконана у векторному графічному додатку Figma.

Для управління базою даних використано СКБД MySQL, доступ до таблиць здійснювався у редакторі MySQL Workbench.

Серверна частина призначена для виконання запитів до бази даних та надсилання інформації до клієнта у JSON-форматі:

- Node.js – середовище з відкритим кодом, що дозволяє використовувати JavaScript на стороні сервера. Працює на різних платформах. Node.js, на відміну від того ж php, обробляє запити так, що не потрібно чекати результату першого запиту для того, щоб виконати наступний. Сервер на Node.js є однопотоковим та неблокуючим, що дуже ефективно для пам’яті, а його функції виконуються асинхронно, що у даному випадку є перевагою, адже зменшує час очікування інформації зі сторони клієнта.
- Express – фреймворк для застосунків, написаних на Node.js, що надає широкий спектр функцій для полегшення роботи з http-запитами, наприклад використання роутера.
- Body-parser – стандартний npm-модуль, призначений для даних, що передаються з клієнта у тілі http-запиту.

- Jest – бібліотека для тестування, використовує асинхронний запуск тестів.
- MySQL2 – бібліотека, призначена для взаємодії сервера з базами даних MySQL.

У зв’язку з тим, що більшість функціоналу застосунку знаходиться на стороні клієнта, список необхідних засоби для взаємодії з ними містить більше елементів. Він включає в себе:

- Vue.js – прогресивний фронт-енд фреймворк, простий у використанні та інтеграції, структурований на основі шаблону MVVM-архітектури, та базується на поділі коду веб-застосунку на відповідні компоненти.[2]
- Vue-router – основна бібліотека маршрутизації для Vue.js, що дозволяє створювати SPA з вкладеними маршрутами та автоматичним переходом між сторінками.[4]
- Vuex-store – одна з бібліотек Vue.js, що імплементує патерн для управління станом програми. Вона призначена для зберігання даних та передбачає існування конкретних правил, згідно з якими ці дані можна модифікувати.[3]
- Vue-axios – Vue-бібліотека, що інтегрує у Vue-код популярну бібліотеку axios, яка в свою чергу призначена для надсилання http-запитів з клієнта на сервер, є асинхронною та у результаті виконання повертає promise; використовується як альтернатива стандартної JavaScript функції fetch().
- Autocomplete-Vue – бібліотека, що містить власний Vue компонент, реалізований у вигляді HTML-тегу для вводу тексту з передбаченими функціями автодоповнення зі списку на основі введених символів.
- Bootstrap-Vue – бібліотека, в якій класи стандартної UI-бібліотеки Bootstrap представлені у вигляді компонентів Vue.[5]

- VueExcelXlsx – бібліотека, що містить власний Vue компонент, який відповідає за збереження рядків розкладу як таблиці у файлі з форматом .xlsx.
- FontAwesome – JavaScript та css бібліотека для відображення іконок.
- SASS-loader – бібліотека, що відповідає за коректну компіляцію файлів, написаних з допомогою css-препроцесора sass/scss;

3.4. Опис розробки програми

Однією з основних характерних рис фреймворка Vue.js є наявність компонентів, які є своєрідними шаблонами HTML-елементів, їх можна використовувати в інших компонентах довільну кількість разів. Всі компоненти знаходяться у папці components. Компоненти, створені в результаті написання програми, вдалось логічно розділити на три категорії – базові, вкладені, та елементи таблиці. До перших було включено компоненти, що використовуються не більше 2х разів, а також ті, які забезпечують навігацію – Header.vue, Footer.vue та Breadcrumbs.vue. До других – елементи, що використовуються багато разів, наприклад рядок списку чи компонент заголовку. У третій – всі компоненти з табличними тегами. Схему використання компонентів на сторінках застосунку можна знайти на рисунку А1 у додатку А(пунктиром позначено компоненти, що використовуються на кожній сторінці).

Vuex-store слугує глобальним сховищем даних у клієнтській частині застосунку, відповідає за відображення інформації про поточний стан та здійснює комунікацію з сервером. Він складається з таких модулів:

- state.module.js – зберігається інформація про авторизованого користувача, а також поточна сторінка та елементи навігації.

- university.module.js – міститься статична інформація про університет: список факультетів, кафедр та спеціальностей, кількість пар, триместрів і навчальних рівнів тощо.
- student.module.js – модуль використовується, якщо користувач авторизований як студент та відповідає за відображення його зведеного розкладу.
- list.module.js – відповідає за списки розкладів відповідно до факультетів та кафедр, також надає можливість методистам змінювати варіант відображення розкладу.
- schedule.module.js – основний модуль для роботи з поточним розкладом і його даними.
- edit.module.js – модуль, що відповідає за синхронізацію інформації при редагуванні даних про розклад і таблиці, та надсилання змін на сервер.
- download.module.js – призначений для перетворення даних таблиці розкладу у формат, необхідний для збереження в excel-файлі.

Усі сервіси на бекенді поділено на 4 модулі, що виконують запити до бази даних:

- university – здійснює запити на отримання інформації з бази даних, що повертають статичну інформацію про структуру університету: списки факультетів, спеціальностей, кафедр та курсів;
- schedules – здійснює get-запити до таблиці «schedule» та повертає необхідні списки розкладів без детальної інформації про курси;
- schedule – CRUD для таблиць «schedule» та «course_schedule»;
- user – повертає дані про дисципліни студента та дані користувача про авторизації.

Для забезпечення коректного доступу до даних, здійснення асинхронних запитів відбувається з використанням синтаксису async-await.

Передача даних між клієнтом та сервером здійснюється задопомогою vue-axios.

3.5. Опис файлів даних

В процесі розробки проекту я працювала з тестовою базою даних САЗу НаУКМА (рис. Б1 додатку Б). До цього у ній не передбачалось збереження розкладів, тому до існуючих було додано дві сутності. Основними таблицями, використаними у проекті, були:

- «course» - таблиця містить основну інформацію про курс, зокрема: код курсу, назву, анотацію, кількість тижневих годин та кредитів, ініціали викладача, кафедру та статус;
- «faculty», «subfaculty», «speciality» - таблиці містять назву факультету, кафедри та спеціальності відповідно, дві останні таблиці також містять посилання на факультет;
- «course_register» - з'єднує між собою таблиці курсів та студентів, також тут знаходиться інформація про тип дисципліни для відповідного студента (професійно-орієнтована, нормативна чи вільного вибору);
- «course_season» - вказує семестр курсу, вид контролю на заліково-екзаменаційній сесії, а також кредити, кількість лекційних та практичних годин на тиждень;
- «_user» - таблиця містить унікальний код користувача, а також його реквізити для логіну, ім'я та роль(студент, методист, методист деканату, адмін тощо);
- «_methodist» - дозволяє визначити, до якого факультету та кафедри належить методист.

Також, було додано дві нові таблиці.

«Schedule» - таблиця, призначена для зберігання загальної інформації про розклад. Містить такі атрибути:

- id – унікальний ідентифікатор;
- code – унікальний згенерований код розкладу;
- title – заголовок розкладу;
- faculty_id – посилання на факультет, якому належить розклад;
- schedule_type – тип розкладу, один із трьох: speciality, session, subfaculty;
- subfaculty_id – посилання на кафедру, якій належить розклад;
- speciality_id – посилання на спеціальність, якій належить розклад;
- academic_year – навчальний рік;
- season – номер навчального триместру (1, 2, 3 відповідно);
- level – навчальний ступінь, 1 - бакалаврат, 2 - магістратура;
- study year – рік навчання (1 - 4);
- draft – boolean-тип, який визначає стан розкладу; поки розклад знаходиться у чорновому варіанті, він не відображається у списку всіх розкладів.

«Course_schedule» - таблиця, рядки якої збігаються з рядками розкладу. Містить такі атрибути:

- id – унікальний ідентифікатор;
- schedule_code – посилання на розклад;
- day_id – день тижня, коли відбувається курс (1-6 відповідно)
- pair_id – номер пари;
- course_cdoc – унікальний код курсу;
- teacher – незважаючи на те, що ПІБ викладача міститься у таблиці даних про курс, є багато випадків, коли лекції та семінари ведуть різні люди, тому атрибут конкретизує викладача на поточне заняття;
- group – номер групи, 0 якщо лекція;
- exam_type – вид контролю, якщо ця рядок із розкладу сесії;
- weeks – тижні проведення дисципліни чи дата заліку/екзамену;

- classroom – номер аудиторії.

Дані, необхідні для коректного відображення числових значень з бази даних на сайті, містяться на клієнтській частині у папці assets/data у файлі static.js. До них входять назви днів тижня, триместрів, освітніх програм, типів контролю на заліково-екзаменаційній сесії, час, що відповідає номерам пар. Немає потреби зберігати ці рядки в базі даних, адже вони не є громіздкими та слугують виключно коректного відображення інформації користувачеві.

3.6. Вигляд інтерфейсу програми

Перед написанням програмного коду було розроблено дизайн прототипу застосунку у векторному редакторі Figma.[10] В ході реалізації програми загальний вигляд дещо змінився, але основні елементи відповідають макету.

Основним прикладом для створення користувацького інтерфейсу був веб-сайт САЗу НаУКМА. Проаналізувавши UI та UX, було виявлено такі закономірності:

- використання UI-бібліотеки Bootstrap;
- абсолютна відсутність модальних вікон у будь-якій частині веб-застосунку;
- використання сповіщень замість модальних вікон;
- використання UI-елементу «акордеон» при відкритті списків розкладів для назви факультетів та навчальних років.

У ході розробки дизайну з веб-сайту САЗу було запозичено такі елементи інтерфейсу:

- система навігації у складі меню та «хлібних крихт», структура сторінок;

- кольорова гама;
- розмір та шрифт основного тексту, заголовків, елементів таблиці тощо;
- розташування та вигляд елементів, що відображають код курсу та номер групи у рядку таблиці;
- схема використання Bootstrap-сповіщень замість модальних вікон.

Дещо також було добавлено чи змінено:

- Замінено елемент «акордеон» на звичайний колапсивний список, щоб можна було відкривати для перегляду кілька факультетів та курсів одночасно(рис. 3.2).

Рисунок 3.2 Розгорнутий список розкладів

- Так як застосунок передбачає можливість зміни в одній частині сторінки без її повного перезавантаження, добавлено каркасні елементи загрузки (skeleton loaders), зокрема для елементів таблиці(рис. 3.3).

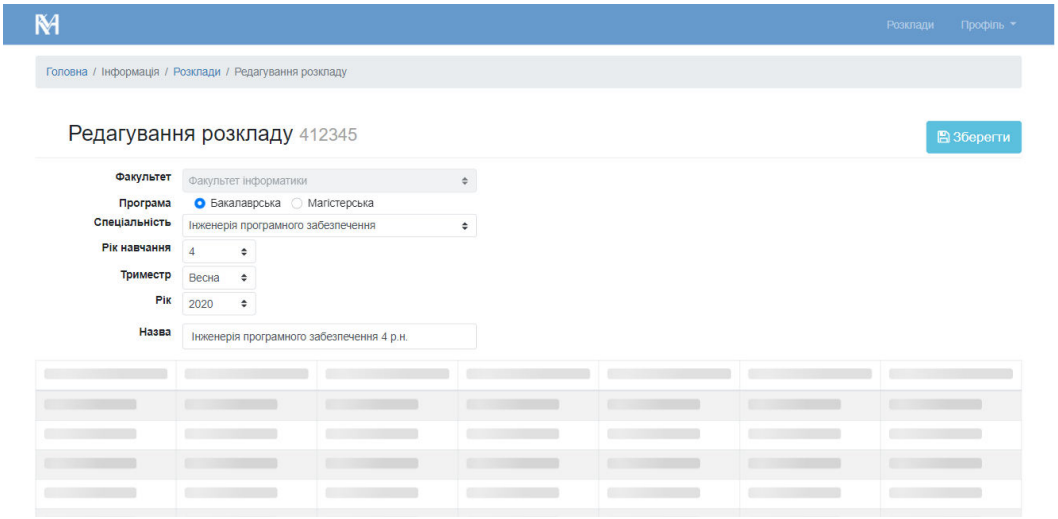


Рисунок 3.3 Каркасна загрузка таблиці

- Система не передбачає перевірку коректності введених даних в input-елементах, але відповідна кнопка залишається неактивною, поки всі необхідні поля не заповнені.
- Назва розкладу не може мати довжину менше 5 символів. Це зроблено з метою уникнення наявних зараз ситуацій, коли розклад має назву «1», «2» тощо.
- Усюди, де це можливо, inputs замінено на випадаючі списки чи вдосконалено автодоповненням(рис. 3.4).

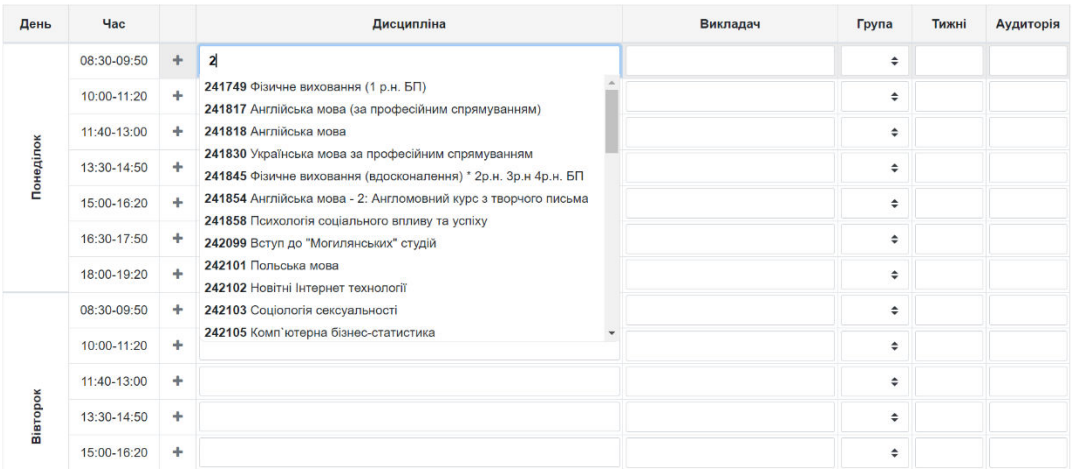


Рисунок 3.4 Приклад автодоповнення

- Добавлено посилання на відповідні сторінки веб-сайту САЗу зокрема для всіх назв курсів. Посилання автоматично відкриється в новій вкладці поточного вікна браузера.
- Для зручнішого перегляду розкладу студентами, його відображено у формі календаря, тобто дні тижня - по горизонталі, а номери пар – по вертикалі.

3.7. Тестування та результати виконання програми

В результаті написання курсового проекту було створено веб-застосунок, що реалізовує повний функціонал для роботи з розкладами в НаУКМА.

Для незареєстрованих користувачів є можливість:

- Переглядати головну сторінку (рис. В1 Додатку В)
- Переглядати список всіх розкладів (рис. В3 Додатку В)
- Відкривати сторінку з будь-яким розкладом (рис. В4 Додатку В)
- Завантажувати будь-який розклад у форматі Excel (рис. В2 Додатку В)

Студенти також можуть переглядати додаткову інформацію у себе в профілі(рис. В5, В6 додатку В).

Методисти мають наступний додатковий функціонал:

- редагування розкладу(рис. В7 додатку В);
- видалення розкладу(рис. В8 додатку В);
- створення нового розкладу(рис. В9 додатку В);
- зміну видимості розкладу(рис. В10 додатку В);

Усі авторизовані користувачі також можуть розлогінитись.

Тестування бекенду здійснено у вигляді Unit-тестів з використанням бібліотеки Jest. Загалом написано 15 тестів, а покриття становить 81% (рис. Г1 Додатку Г).

Тестування фронтенду виконувалось вручну. Основним критерієм було задовільнення програмою функціональних вимог, вказаних у п. 3.1 та наявність усіх ключових елементів, що містяться у прототипі.

ВИСНОВКИ

Отже, в ході виконання курсового проекту створено веб-застосунок на основі бази даних САЗу НаУКМА, що реалізовує функціонал для роботи з розкладами занять в університеті.

Досліджено технології, що використовуються при розробці веб-застосувань. З них вибрано достатньо популярні та широковживані, що дозволять повністю реалізувати функціонал веб-застосунку.

Проведено аналіз предметної області. На основі отриманих результатів визначено основні функціональні вимоги до застосунку та узагальнено вигляд таблиці розкладів залежно від його типу.

Проаналізовано UI та UX веб-сайту САЗу, отримані закономірності використано при розробці дизайну прототипу власного веб-застосунку.

Розроблено веб-застосунок, в якому реалізовано функціонал для створення, перегляду, редагування та видалення розкладів, а також можливість перегляду зведеного розкладу за індивідуальним планом для студентів.

У результаті виконання проекту створено веб-сайт з інтуїтивно-зрозумілим інтерфейсом, наближеним до САЗу, у якому методисти деканатів можуть здійснювати усі необхідні операції над розкладами занять. Проект є унікальним, адже на момент дослідження формування розкладів та передача їх студентам відбувалась різними способами на кожному факультеті.

Проект має багато можливостей для подальшого вдосконалення, зокрема:

- Стилізування Excel-файл з завантажуваним розкладом.
- Завантаження розкладу в інших форматах, наприклад у PDF.
- Внесення до бази даних усіх існуючих аудиторій та на основі розкладів виведення списку вільних приміщень для проведення факультативних занять чи інших потреб.
- Реєстрація задопомогою Office-365

Список використаної літератури

1. Система автозапису на дисципліни НаУКМА [Електронний ресурс]. Режим доступу: https://my.ukma.edu.ua
2. Офіційний веб-сайт Vue.js [Електронний ресурс]. Режим доступу: https://vuejs.org/v2/guide/
3. Офіційний веб-сайт Vuex [Електронний ресурс]. Режим доступу: https://vuex.vuejs.org/
4. Офіційний веб-сайт роутера Vue-Router [Електронний ресурс]. Режим доступу: https://router.vuejs.org/
5. Офіційний веб-сайт Bootstrap-Vue [Електронний ресурс]. Режим доступу: https://bootstrap-vue.org
6. Клієнт-серверна взаємодія у веб-застосунках[Електронний ресурс]. Режим доступу: https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/
7. Хуки життєвого циклу Vue.js[Електронний ресурс]. Режим доступу: https://habr.com/ru/company/mailru/blog/350962/
8. Особливості Vue.js та порівняння з іншими фреймворками[Електронний ресурс]. Режим доступу: https://stfalcon.com/ru/blog/post/vue-js-guide-to-tech
9. Введення в REST API[Електронний ресурс]. Режим доступу: https://habr.com/ru/post/483202/
10. Прототип застосунку[Електронний ресурс]. Режим доступу: https://www.figma.com/file/6OCNTl2qn9eS7SZOTWu3Vj/ScheduleMaker?node-id=0%3A1
11. Програмний код серверної частини[Електронний ресурс]. Режим доступу: https://github.com/solja484/schedule-api
12. Програмний код клієнтської частини[Електронний ресурс]. Режим доступу: https://github.com/solja484/schedue-vue

ДОДАТКИ

Додаток А

(довідниковий)

Діаграма Vue-компонентів

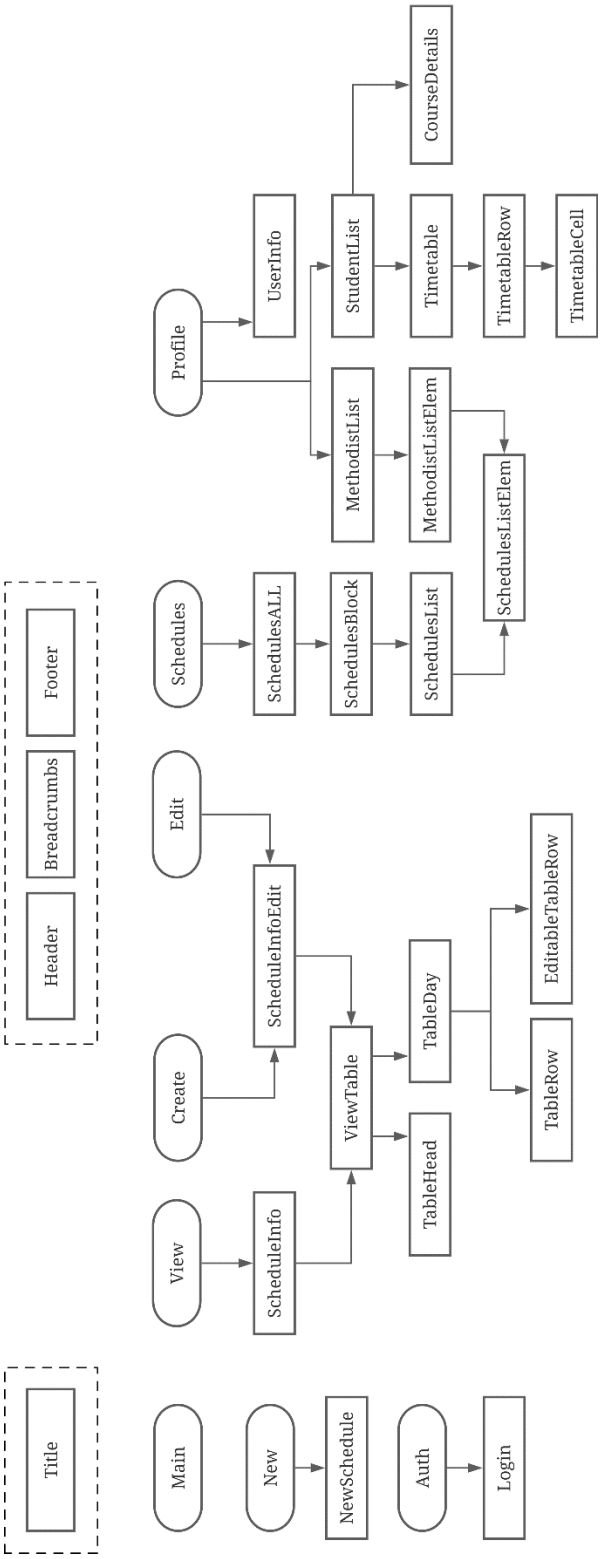


Рисунок А1 Діаграма Vue-компонентів

Додаток Б

(довідниковий)

Схема моделі бази даних САЗу

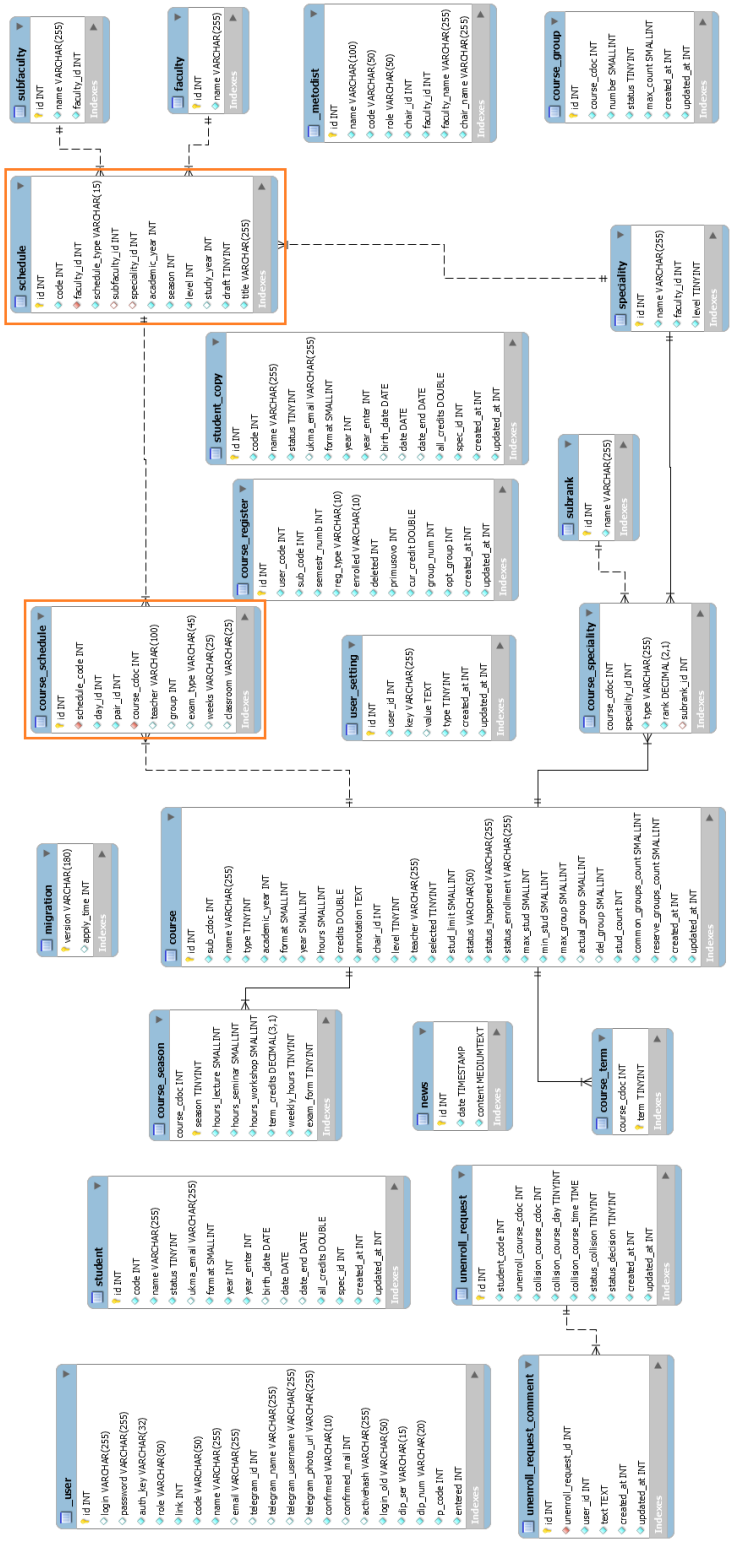


Рисунок Б1 Схема ER-моделі

Додаток В

(довідниковий)

Графічний інтерфейс веб-застосунку

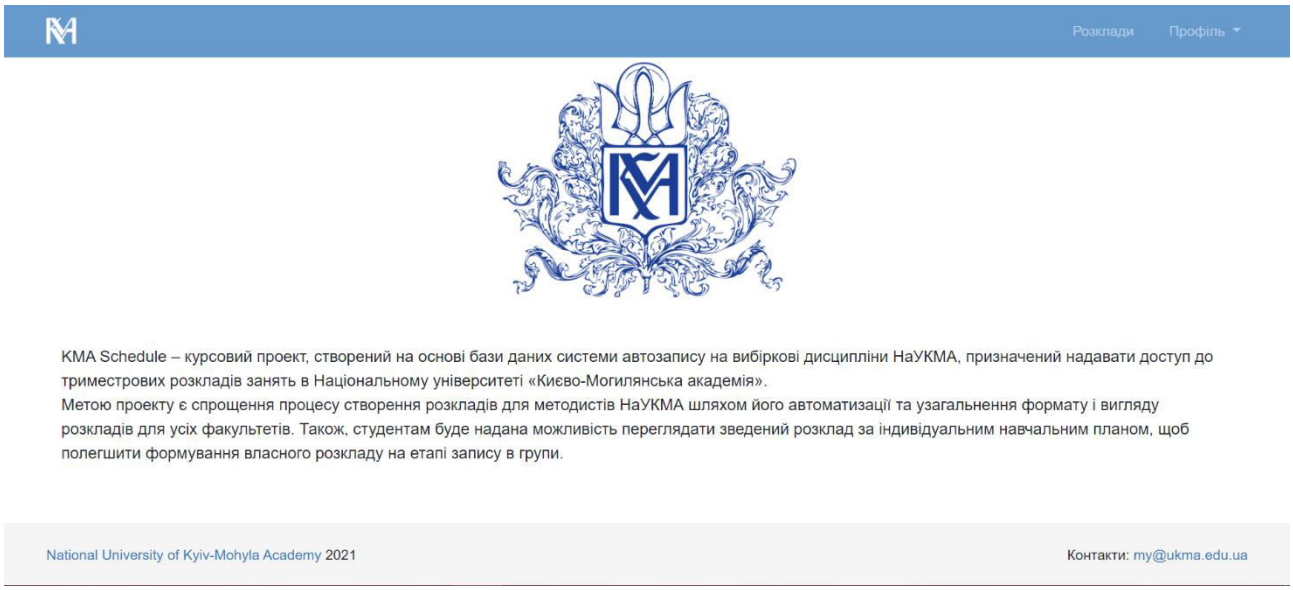


Рисунок В1 Головна сторінка

	A	B	C	D	E	F
1	День	Час	Дисципліна, викладач	Група	Тижні	Аудиторія
2	Понеділок	08:30-09:50				
3		10:00-11:20	Інтелектуальні системи Жежерун О.П.	Лекція	1-7,10-12	дистанційно
4		11:40-13:00	Інтелектуальні системи Жежерун О.П.	1	1-7,10-13	дистанційно
5		13:30-14:50	Інтелектуальні системи Жежерун О.П.	2	1-7,10-13	дистанційно
6		15:00-16:20	Інтелектуальні системи Жежерун О.П.	3	1-7,10-13	дистанційно
7		16:30-17:50	Інтелектуальні системи Жежерун О.П.	4	1-7,10-13	дистанційно
8		18:00-19:20				
9	Вівторок	08:30-09:50	Прикладне програмування мобільних систем на основі ОС Android	Лекція	1-7,10-12	дистанційно
10		10:00-11:20	Інтелектуальні системи Жежерун О.П.	5	1-7,10-13	дистанційно
11			Комп`ютерна графіка	1	2-7,9-13	дистанційно
12		11:40-13:00	Комп`ютерна графіка	2	2-7,9-13	дистанційно
13		13:30-14:50	Комп`ютерна графіка	3	2-7,9-13	дистанційно
14		15:00-16:20				
15		16:30-17:50	Прикладне програмування мобільних систем на основі ОС Android	1	1-7,10-13	дистанційно
16		18:00-19:20	Прикладне програмування мобільних систем на основі ОС Android	2	1-7,10-13	дистанційно
17	Середа	08:30-09:50	Забезпечення якості програмних продуктів	Лекція	1-7	дистанційно
18			Прикладне програмування мобільних систем на основі ОС Android	3	1-7,10-13	дистанційно
19		10:00-11:20				
20		11:40-13:00	Технологія веб-програмування Ruby on Rails	Лекція	2-7,9-12	дистанційно

Рисунок В2 Вигляд завантаженого розкладу

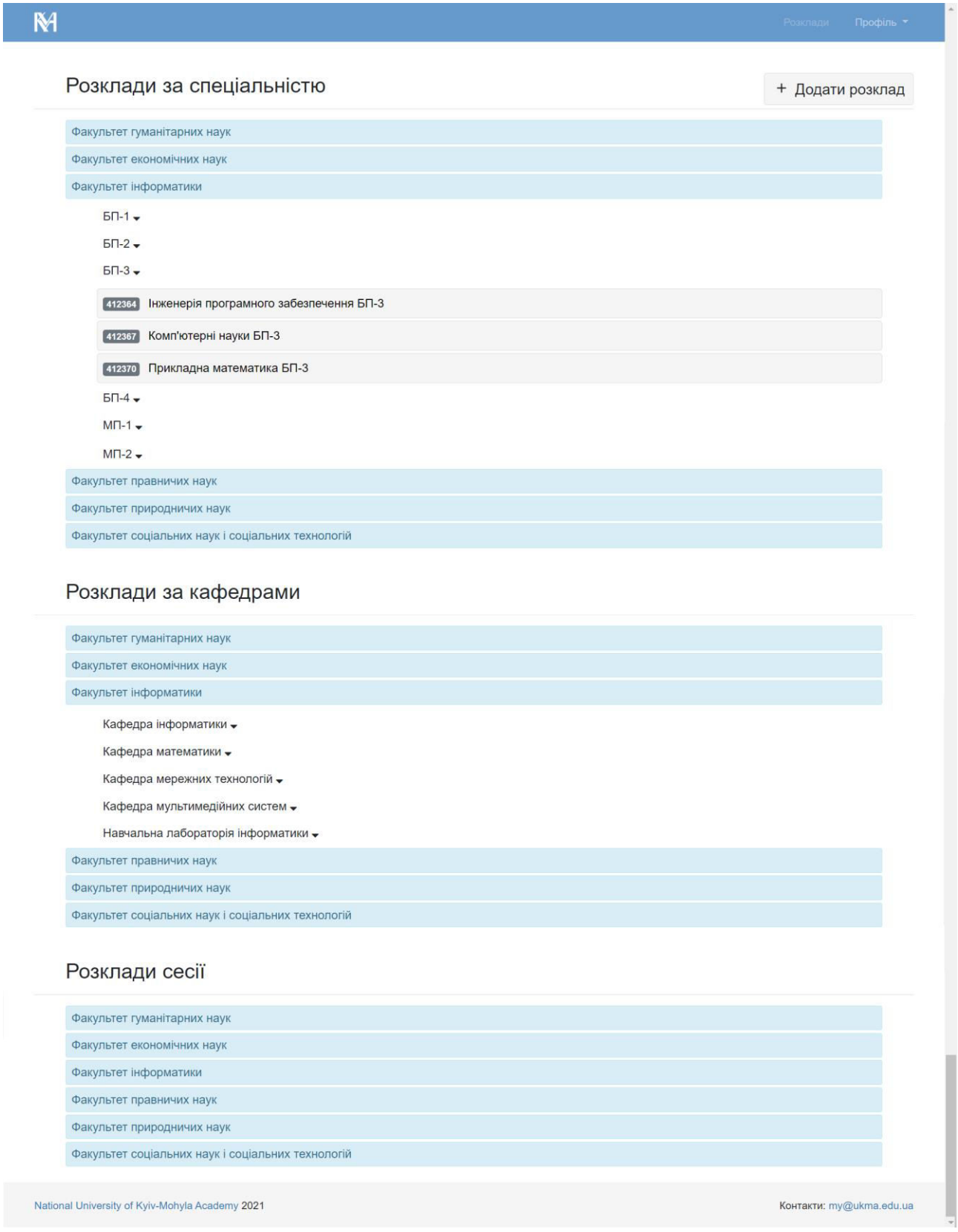


Рисунок В3 Список розкладів

День	Час	Дисципліна, викладач	Група	Тижні	Аудиторія
Понеділок	08:30-09:50				
	10:00-11:20	242154 **Інтелектуальні системи** *Жежерун О.П.*	Лекція	1-7,10-12	дистанційно
	11:40-13:00	242164 **Інтелектуальні системи** *Жежерун О.П.*	1	1-7,10-13	дистанційно
	13:30-14:50	242164 **Інтелектуальні системи** *Жежерун О.П.*	2	1-7,10-13	дистанційно
	15:00-16:20	242154 **Інтелектуальні системи** *Жежерун О.П.*	3	1-7,10-13	дистанційно
	16:30-17:50	242164 **Інтелектуальні системи** *Жежерун О.П.*	4	1-7,10-13	дистанційно
	18:00-19:20				
Вівторок	08:30-09:50	242192 **Прикладне програмування мобільних систем на основі ОС Android**	Лекція	1-7,10-12	дистанційно
	10:00-11:20	242154 **Інтелектуальні системи** *Жежерун О.П.*	5	1-7,10-13	дистанційно
		242130 **Комп`ютерна графіка**	1	2-7,9-13	дистанційно
	11:40-13:00	242130 **Комп`ютерна графіка**	2	2-7,9-13	дистанційно
	13:30-14:50	242130 **Комп`ютерна графіка**	3	2-7,9-13	дистанційно
	15:00-16:20				
	16:30-17:50	242192 **Прикладне програмування мобільних систем на основі ОС Android**	1	1-7,10-13	дистанційно
	18:00-19:20	242192 **Прикладне програмування мобільних систем на основі ОС Android**	2	1-7,10-13	дистанційно
Середа	08:30-09:50	242141 **Забезпечення якості програмних продуктів**	Лекція	1-7	дистанційно
		242192 **Прикладне програмування мобільних систем на основі ОС Android**	3	1-7,10-13	дистанційно
	10:00-11:20				
	11:40-13:00	242161 **Технологія веб-програмування Ruby on Rails**	Лекція	2-7,9-12	дистанційно
	13:30-14:50	242151 **Технологія веб-програмування Ruby on Rails**	1	2-7,9-13	дистанційно
	15:00-16:20	242161 **Технологія веб-програмування Ruby on Rails**	2	2-7,9-13	дистанційно
	16:30-17:50	242161 **Технологія веб-програмування Ruby on Rails**	3	2-7,9-13	дистанційно
		242192 **Прикладне програмування мобільних систем на основі ОС Android**	4	1-7,10-13	дистанційно
	18:00-19:20				
Четвер	08:30-09:50	242192 **Прикладне програмування мобільних систем на основі ОС Android**	5	1-7,10-13	дистанційно
	10:00-11:20				
	11:40-13:00	242141 **Забезпечення якості програмних продуктів**	3	1-7,9-15	дистанційно
	13:30-14:50	242141 **Забезпечення якості програмних продуктів**	4	1-7,9-15	дистанційно
	15:00-16:20	242192 **Прикладне програмування мобільних систем на основі ОС Android**	6	1-7,10-13	дистанційно
	16:30-17:50	242192 **Прикладне програмування мобільних систем на основі ОС Android**	7	1-7,10-13	дистанційно
	18:00-19:20	242192 **Прикладне програмування мобільних систем на основі ОС Android**	8	1-7,10-13	дистанційно
П'ятниця	08:30-09:50	242192 **Прикладне програмування мобільних систем на основі ОС Android**	9	1-7,10-13	дистанційно
		242196 **Структура програмних проектів**	Лекція	1-7	дистанційно
	10:00-11:20	242141 **Забезпечення якості програмних продуктів**	2	1-7,9-15	дистанційно
		242196 **Структура програмних проектів**	1	1-7,9-15	дистанційно
	11:40-13:00	242141 **Забезпечення якості програмних продуктів**	1	1-7,9-15	дистанційно
		242196 **Структура програмних проектів**	2	1-7,9-15	дистанційно
	13:30-14:50	242141 **Забезпечення якості програмних продуктів**	5	1-7,9-15	дистанційно
		242196 **Структура програмних проектів**	3	1-7,9-15	дистанційно
	15:00-16:20	242196 **Структура програмних проектів**	4	1-7,9-15	дистанційно
	16:30-17:50	242192 **Прикладне програмування мобільних систем на основі ОС Android**	10	1-7,10-13	дистанційно
		242196 **Структура програмних проектів**	5	1-7,9-15	дистанційно
	18:00-19:20				
Субота	08:30-09:50	242130 **Комп`ютерна графіка**	Лекція	1-7,9-11	дистанційно
	10:00-11:20	242130 **Комп`ютерна графіка**	4	2-7,9-13	дистанційно
	11:40-13:00	242130 **Комп`ютерна графіка**	5	2-7,9-13	дистанційно
	13:30-14:50	242130 **Комп`ютерна графіка**	6	2-7,9-13	дистанційно
	15:00-16:20				
	16:30-17:50				
	18:00-19:20				

Рисунок В4 Перегляд розкладу

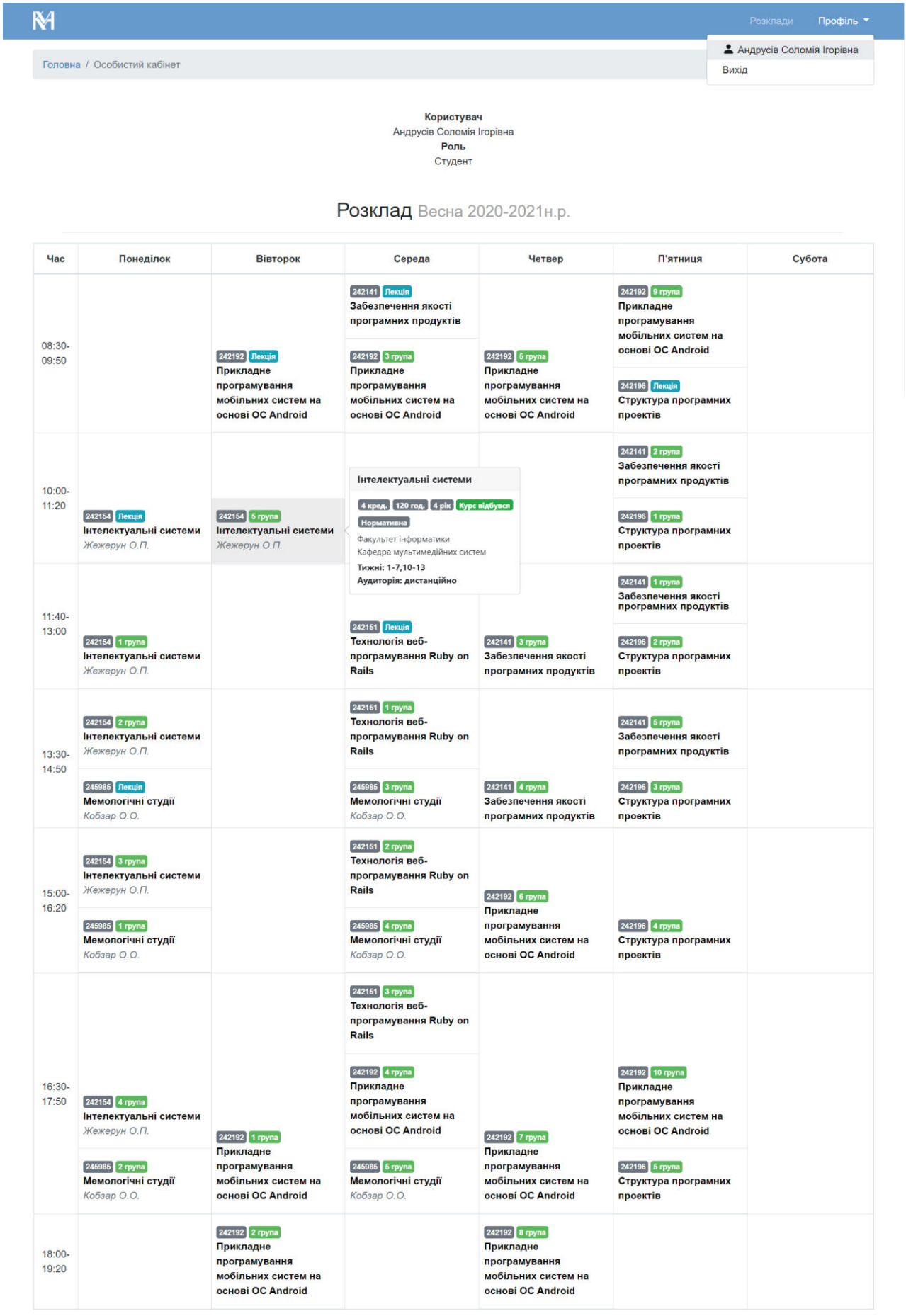


Рисунок В5 Зведений розклад студента

Розклад сесії Весна 2020-2021н.р.

Час	Понеділок	Вівторок	Середа	Четвер	П'ятниця	Субота
08:30-09:50		242192 2 група 20.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**	242192 3 група 21.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**	242192 5 група 22.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**	242192 9 група 23.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**	
10:00-11:20	242154 1 група 26.04.2021 **Інтелектуальні системи**	242154 Всі групи 27.04.2021 **Інтелектуальні системи**		242141 1 група 29.04.2021 **Забезпечення якості програмних продуктів**	242141 Всі групи 30.04.2021 **Забезпечення якості програмних продуктів**	242196 1 група 24.04.2021 **Структура програмних проектів**
11:40-13:00	242154 2 група 26.04.2021 **Інтелектуальні системи**		242151 3 група 21.04.2021 **Технологія веб-програмування Ruby on Rails**	242141 2 група 29.04.2021 **Забезпечення якості програмних продуктів**		242196 2 група 24.04.2021 **Структура програмних проектів**
13:30-14:50	242154 3 група 26.04.2021 **Інтелектуальні системи**		242151 1 група 21.04.2021 **Технологія веб-програмування Ruby on Rails**	242141 3 група 29.04.2021 **Забезпечення якості програмних продуктів**		242196 3 група 24.04.2021 **Структура програмних проектів**
15:00-16:20	242154 4 група 26.04.2021 **Інтелектуальні системи**		242151 2 група 21.04.2021 **Технологія веб-програмування Ruby on Rails**	242141 4 група 29.04.2021 **Забезпечення якості програмних продуктів** 242192 6 група 22.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**		242196 4 група 24.04.2021 **Структура програмних проектів**
16:30-17:50	242154 5 група 26.04.2021 **Інтелектуальні системи**	242192 1 група 20.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**	242192 4 група 21.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**	242141 5 група 29.04.2021 **Забезпечення якості програмних продуктів** 242192 7 група 22.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**	242192 10 група 23.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**	242196 5 група 24.04.2021 **Структура програмних проектів**
18:00-19:20				242192 8 група 22.04.2021 **Прикладне програмування мобільних систем на основі ОС Android**		

Рисунок В6 Зведений розклад сесії студента

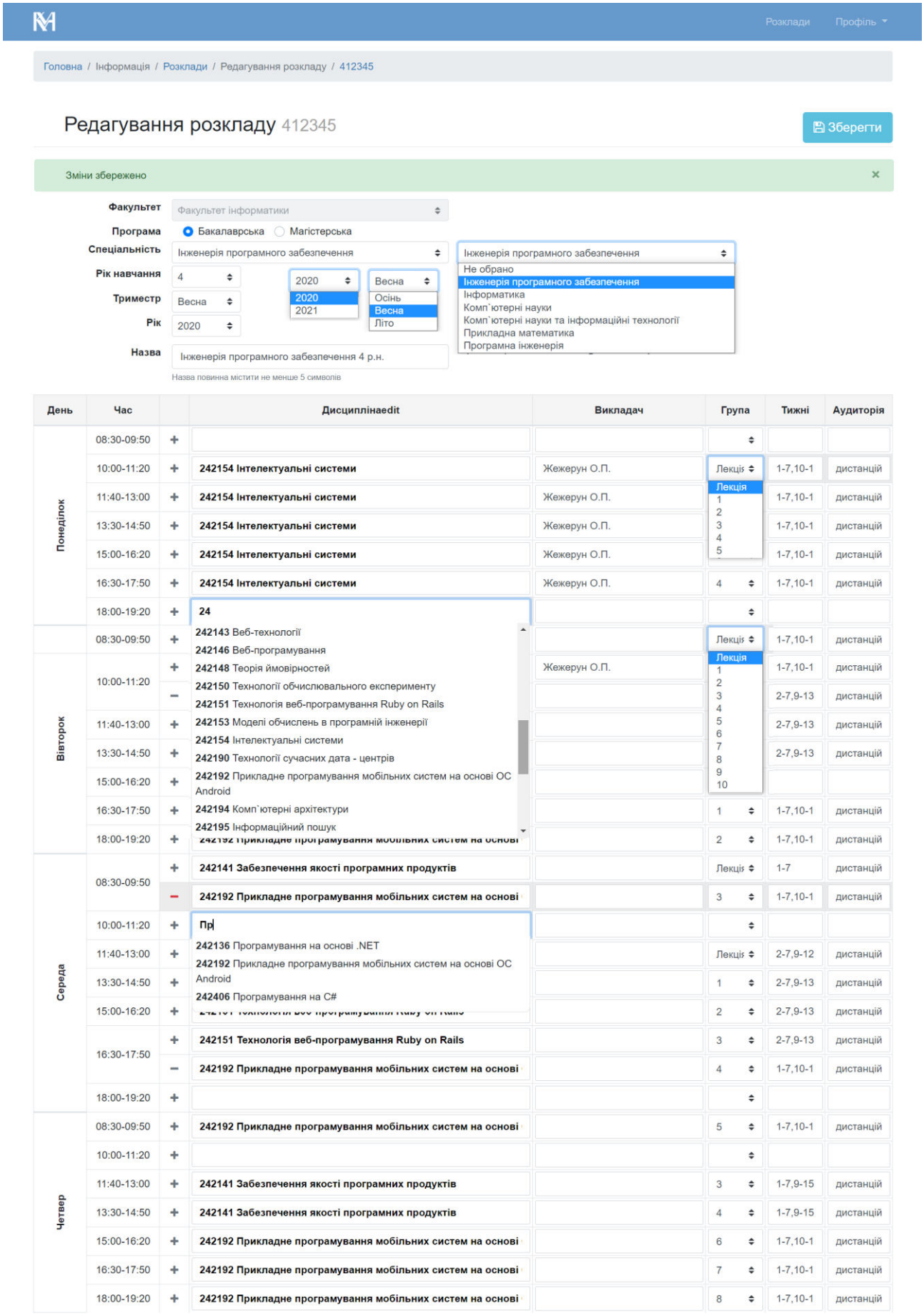


Рисунок В7 Функціонал редагування розкладу

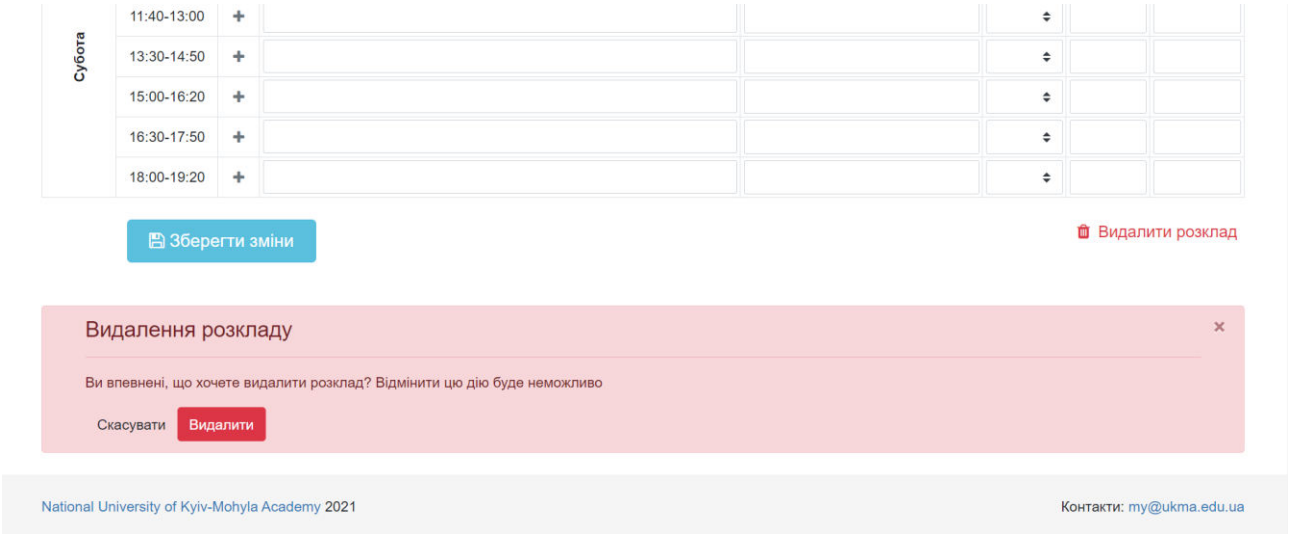


Рисунок В8 Видалення розкладу

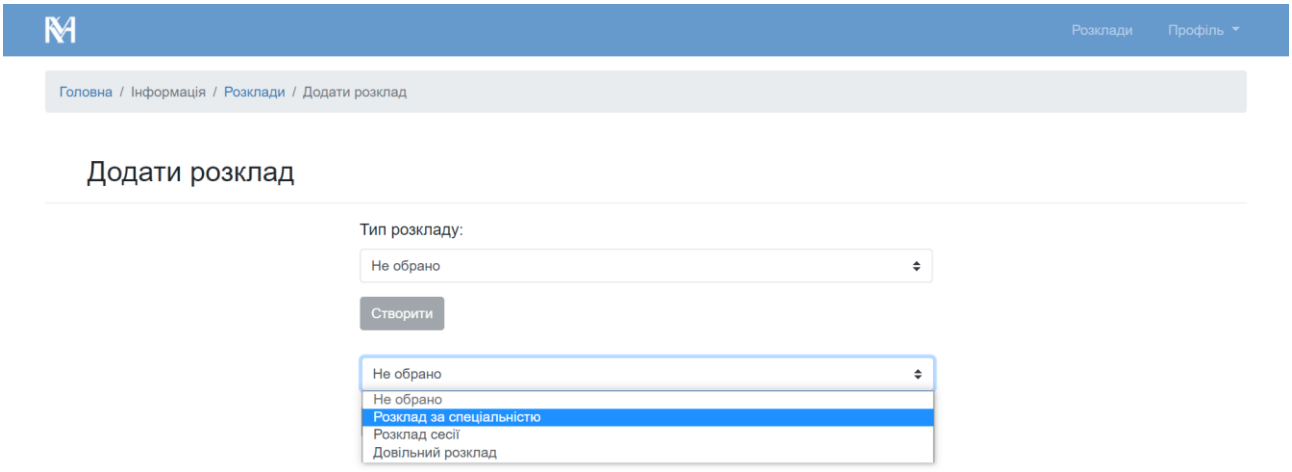


Рисунок В9 Створення нового розкладу

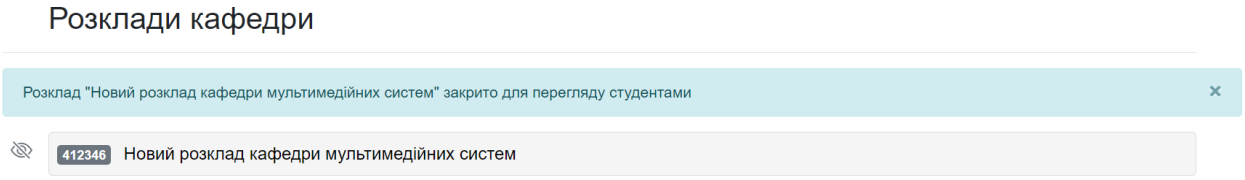


Рисунок В10 Зміна варіанту розкладу з основного на чорновий

Додаток Г

(довідниковий)

Звіт по тестуванню серверної частини

```
----------------------|---------|----------|---------|---------|-------------------------------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------------------------------
All files             |   80.52 |      100 |     100 |   80.52 |
 schedule-api         |     100 |      100 |     100 |     100 |
  config.js           |     100 |      100 |     100 |     100 |
  setup.js            |     100 |      100 |     100 |     100 |
 schedule-api/routes  |   64.29 |      100 |     100 |   64.29 |
  router.js           |   64.29 |      100 |     100 |   64.29 | 13-14,22-23,31-32,43-44,53-54,62-63,72-73
 schedule-api/services |     100 |      100 |     100 |     100 |
  db.js               |     100 |      100 |     100 |     100 |
  schedule.js         |     100 |      100 |     100 |     100 |
  schedules.js        |     100 |      100 |     100 |     100 |
  university.js       |     100 |      100 |     100 |     100 |
  user.js             |     100 |      100 |     100 |     100 |
----------------------|---------|----------|---------|---------|-------------------------------------------

Test Suites: 5 passed, 5 total
Tests:       15 passed, 15 total
```

Рисунок Г1 Покриття бекенду тестами

Додаток Д

(обов'язковий)

Програмний код серверної частини

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const server = express();
const port = 3080;
const router = require('./routes/router');

server.use(bodyParser.json());
server.use(bodyParser.urlencoded({extended: true,}));
server.use(express.static(path.join(__dirname, '../schedule-vue/build')));
server.get('/', (req, res) => {
    res.json({'message': 'ok'});
});

server.use('/api', router);

// Error handler middleware
server.use((err, req, res, next) => {
    const statusCode = err.statusCode || 500;
    console.error(err.message, err.stack);
    res.status(statusCode).json({'message': err.message});
    return;
});

server.listen(port, () => {
    console.log('listening to ' +port)
});
```

Рисунок Д1 Express-сервер

```
const mysql = require('mysql2/promise');
const config = require('../config');

async function query(sql, params) {
    const connection = await mysql.createConnection(config.db);
    const [results, ] = await connection.execute(sql, params);
    connection.end();
    return results;
}


module.exports = {
    query
};
```

Рисунок Д2 Модуль підключення до бази даних

```
const express = require('express');
const router = express.Router();
const university = require('../services/university');
const schedules = require('../services/schedules');
const schedule = require('../services/schedule');
const user = require('../services/user');


router.get('/university/faculty', async function (req, res, next) {
    try {
        res.json(await university.getFaculties());
    } catch (err) {
        console.error(`Error while getting faculties`, err.message);
        next(err);
    }
});

router.get('/university/speciality', async function (req, res, next) {
    try {
        res.json(await university.getSpeciality());
    } catch (err) {
        console.error(`Error while getting specialities`, err.message);
        next(err);
    }
});
```

Рисунок Д3 Express router(частина файлу)

```
const db = require('./db');

async function getFaculties() {
    return await db.query("SELECT * FROM faculty");
}

async function getSubFaculties() {
    return await db.query(`SELECT * FROM subfaculty ORDER BY faculty_id`);
}

async function getSpeciality() {
    return await db.query(`SELECT DISTINCT name, id, faculty_id, level FROM speciality ORDER BY faculty_id`);
}

async function getSpecialityCourses(req) {
    return await db.query(
        `SELECT DISTINCT course.id as id, sub_cdoc as course_code, course.name as name, actual_group,
            season, exam_form FROM course_speciality
            LEFT JOIN course ON course.sub_cdoc=course_speciality.course_cdoc
            LEFT JOIN course_season ON course.sub_cdoc=course_season.course_cdoc
            WHERE course_speciality.speciality_id=?
            AND course.level=? AND course.status_happened=?
            AND course_season.season=? AND course.academic_year=?`,
        [req.speciality, req.level, 'happened', req.season, req.academic_year]
    );
}

async function getFacultyCourses(req) {
    return await db.query(
        `SELECT DISTINCT course.id as id, sub_cdoc as course_code, course.name as name, actual_group,
            season, exam_form FROM course_speciality
            LEFT JOIN course ON course.sub_cdoc=course_speciality.course_cdoc
            LEFT JOIN course_season ON course.sub_cdoc=course_season.course_cdoc
            WHERE course.level=? AND course.status_happened=?
            AND course_season.season=? AND course.academic_year=?
            AND course.chair_id IN
            (SELECT id FROM subfaculty WHERE faculty_id=?)`,
        [req.level, 'happened', req.season, req.academic_year, req.faculty]);
}

module.exports = {
    getFaculties,
    getSubFaculties,
    getSpeciality,
    getSpecialityCourses,
    getFacultyCourses
};
```

Рисунок Д4 Приклад одного серверного модуля

Додаток Е

(обов'язковий)

Програмний код клієнтської частини

```
import Vue from "vue";
import App from "./App.vue";
import router from "./router";
import store from "./store/Store";
import axios from "vue-axios";
import { BootstrapVue, IconsPlugin } from "bootstrap-vue";
import Autocomplete from "@trevoreyre/autocomplete-vue";
import VueExcelXlsx from "vue-excel-xlsx";

Vue.use(VueExcelXlsx);
Vue.use(Autocomplete);
Vue.use(BootstrapVue);
Vue.use(IconsPlugin);

Vue.config.productionTip = false;
Vue.prototype.$http = axios;
new Vue({
  router,
  store,
  render: h => h(App)
}).$mount("#app");
```

Рисунок Е1 Головний файл Vue

```
import Vue from "vue";
import VueRouter from "vue-router";
import Main from "../views/Main.vue";

Vue.use(VueRouter);

const routes = [
  {
    path: "/",
    name: "Main",
    component: Main
  },
  {
    path: "/schedules",
    name: "Schedules",
    component: () => import("../views/Schedules.vue")
  },
  {
    path: "/schedules/create",
    name: "Create schedule",
    component: () => import("../views/Create.vue")
  },
  {
    path: "/schedules/new",
    name: "New",
    component: () => import("../views/New.vue")
  },
  {
    path: "/schedules/edit/:code",
    name: "Edit",
    component: () => import("../views/Edit.vue")
  },
  {
    path: "/schedules/view/:code",
    name: "ViewPage",
    component: () => import("../views/View.vue")
  },
  {
    path: "/student/:code",
    name: "Student schedule",
    component: () => import("../views/Profile.vue")
  },
  {
    path: "/methodist/:code",
    name: "Methodist schedule",
    component: () => import("../views/Profile.vue")
  },
  {
    path: "/login",
    name: "Login",
    component: () => import("../views/Auth.vue")
  }
];

const router = new VueRouter({
  mode: "history",
  base: process.env.BASE_URL,
  routes
});

export default router;
```

Рисунок Е2 Vue-Router

```
import Vue from "vue";
import Vuex, { Store } from "vuex";
import universityModule from "./university.module";
import stateModule from "./state.module";
import studentModule from "./student.module";
import scheduleModule from "./schedule.module";
import schedulesListModule from "./list.module";
import editModule from "./edit.module";
import downloadModule from "./download.module";

Vue.use(Vuex);

const store = new Store({
  modules: {
    university: universityModule,
    state: stateModule,
    student: studentModule,
    schedule: scheduleModule,
    list: schedulesListModule,
    edit: editModule,
    download: downloadModule
  },
  state: {
    faculty: null,
    loading: false,
    currentYear: 2020,
    currentSeason: 2
  },
  getters: {
    faculty: state => state.faculty,
    loading: state => state.loading,
    currentYear: state => state.currentYear,
    currentSeason: state => state.currentSeason
  },
  actions: {
    changeFaculty({ commit }, newFaculty) {
      commit("setFaculty", newFaculty);
    }
  },
  mutations: {
    setFaculty(state, newFaculty) {
      state.faculty = newFaculty;
    },
    setLoading(state, load) {
      state.loading = load;
    }
  }
});
export default store;
```

Рисунок Е3 Основий файл Vuex-store