ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут Інформаційних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступень вищої освіти бакалавр

на тему «**РОЗРОБКА FRONT-END ЧАСТИНИ ДЛЯ LEARNING-ДОДАТКУ “CAMPUS” З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ ANGULAR»**

Виконав: студент 4 курсу, групи ПД-42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Фетько Юрій Іванович
(прізвище та ініціали)
Керівник Коба А.Б.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

КИЇВ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут Інформаційних технологій

Кафедра Інженерії прорамного забезпечення
Ступінь вищої освіти «Бакалавр»
Спеціальність підготовки 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії прорамного
забезпечення
_____О.В.Негоденко
“____” _____ 2021 року

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

Фетьку Юрію Івановичу
(прізвище, ім’я, по батькові)

1.Тема роботи: «Розробка програмного забезпечення інтерактивної системи для вивчення історії за допомогою мови програмування Java»

Керівник роботи Коба Андрій Борисович, с.в.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «12» березня 2021 року №65

2. Строк подання студентом роботи «1» червня 2021року

3. Вихідні дані до роботи: веб-застосунки для дистанційного навчання, методи та засоби проектування та розробки програмного забезпечення, принципи побудови односторінкових додатків.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1 Огляд предметної області

4.2 Вибір засобів реалізації

4.3Проектування веб-застосунку

4.4 Реалізація веб-застосунку

4.5 Тестування веб-застосунку

5. Перелік графічного матеріалу (презентація)

1.

2.

3.

4.

6. Дата видачі завдання__1.06.2021__

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання на бакалаврську роботу	19.04.21	
2	Огляд предметної області	20.04.21	
3	Вибір інструментальних засобів реалізації	25.04.21	
4	Проектування засосунку	28.04.21	
5	Реалізація веб-застосунку	30.04.21	
6	Тестування веб-застосунку	06.05.21	
7	Написання та оформлення пояснювальної записки	10.05.21	
8	Розробка графічних та презентаційних матеріалів	15.05.21	
9	Захист бакалаврської роботи	01.06.21	
10			

Студент _____ Ю.І. Фетько
(підпис) (прізвище та ініціали)

Керівник роботи _____ А.Б. Коба
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 55с., 33 рис., 3 табл., 1 дод., 18 джерел.

ANGULAR, SPA, MATERIAL UI, FULLCALENDAR, GIT, FRONT-END, FRAMEWORK, JEST

Об’єкт дослідження – поліпшення методики ведення дистанційного навчального процесу.

Предмет дослідження – клієнтська частина веб додатку для обміну та аналізу інформації щодо графіка та завдань студентів освітніх установ або за інтересами.

Мета роботи – розробка клієнтської частини веб додатку з метою поліпшення методики ведення дистанційного навчального процесу.

Методи дослідження – принципи SPA (Single Page Application), методи проектування архітектури проекту, методи тестування.

У роботі було проведено аналіз існуючих веб додатків, які були створені для дистанційного навчання. Розглянуто використання цих веб додатків, їх переваги та недоліки. Обрано актуальні програмні засоби, для створення візуальної частини даного сервісу. Проведено опис використаних програмних засобів. Реалізована клієнтська частина на фреймворку Angular, детально розібраний кожен підхід до реалізації окремих частин проекту.

Результат даної роботи полягає в можливості впровадження цього веб сервісу в повсякденну роботу студентів, мінімізувавши таким чином кількість додатків для комунікації. Особливість даного сервісу дозволяє використовувати його починаючи з учнів шкіл і закінчуючи будь-якими бажаючими навчатись дистанційно.

ЗМІСТ

ВСТУП

У час інтернет-технологій багато аспектів нашого життя переноситься в мережу, прискорюючи тим самим темпи розвитку інформаційного суспільства, долаючи географічні бар’єри. Освіта не стає виключенням. Зараз вже не обов’язково знаходитись поруч з викладачем для того, щоб отримувати знання. Інтернет дає змогу розширити їх, зробити заочне навчання справді повноцінним та всебічним.

Дистанційна освіта стає надзвичайно популярною формою навчання в силу своєї зручності й гнучкості. Вона усуває основний бар'єр, що утримує багатьох людей від отримання освіти, позбавляючи від необхідності відвідувати заняття за встановленим розкладом.

Для того, щоб забезпечити ефективну взаємодію, при дистанційному навчанні використовується цілий набір інструментів, включаючи інтерактивні комп'ютерні програми, Інтернет, електронну пошту, телефон та інші.

Тому створення додатку для дистанційного навчання є актуальною задачею, для спеціалістів ІТ профілю, яка дає можливість організувати процес свого навчання у найзручніший спосіб для користувача.

Об’єкт дослідження – поліпшення методики ведення дистанційного навчального процесу.

Предмет дослідження – клієнтська частина веб додатку для обміну та аналізу інформації щодо графіка та завдань студентів освітніх установ або за інтересами.

Мета роботи – розробка клієнтської частини веб додатку з метою поліпшення методики ведення дистанційного навчального процесу.

Методи дослідження – принципи SPA (Single Page Application), методи проектування архітектури проекту, методи тестування.

Наукова новизна даної роботи полягає в наступному:

1) Розроблено алгоритм для покращення системи навчання студентів та викладачів.

2) Встановлено, що використання Angular 10 версії – це вдале рішення для досягнення поставленої мети.

3) Показано, що фреймворк Angular зручний у використанні та надає можливість ефективної розробки клієнтської частини веб застосунку.

Ціль цієї роботи, вивчити потреби користувачів та створити зручну у використанні та зрозумілу для будь-якого користувача клієнтську частину освітнього додатка для дистанційного навчання – “Campus”.

В роботі виконано аналіз існуючих додатків для навчання. Встановлено переваги та недоліки. В результаті аналізу було визначено основні потреби користувачів.

Проаналізовано можливості середовища розробки Campus. Розроблено логіку практичних завдань та загальну концепцію представлення інформації для користувачів додатку.

Практична значущість результатів дослідження полягає у вільному доступі додатка, в можливості використовування будь-якою людиною, котрій важливо організувати і структурувати свій навчальний процес у найбільш зручному форматі. Додаток мінімізує кількість додатків для комунікації та навчання своїми зручними функціями. Універсальність цього сервісу дозволяє використовувати його як учням, студентам, так і викладачам.

1 FRONT-END НА СЬОГОДНІ

1.1 Концепція односторінкового веб-сайту

Перед тим як розпочати розповідати про веб-застосунок “Campus”, важливо заглянути у сучасний Front-end та розібрати його основні принципи, фреймворки та бібліотеки і врешті решт обрати для себе інструмент розробки, який буде актуальний для веб-додатку.

Нині веб-технології розвиваються не по днях, а по годинах. Швидкоплинність розвитку веб-технологій дозволяє створити розробникам, починаючи від звичайного рекламного сайту-візитки, справжній сайт з інтерактивними можливостями, тобто справжні веб-сервіси, які можуть конкурувати з мобільними додатками, змушуючи їх розробників створювати свої власні аналоги в онлайн версії.

Single page application (SPA) – це концепція веб-застосунку, що робить його інтерактивним і подібним на звичайний додаток, основна мета якого працювати в браузері й не перезавантажувати сторінку під час роботи. Ідея SPA не нова, вона почала літати в думках розробників як тільки у світ вийшов реліз технології AJAX, тоді ж вебсайти почали набувати динаміки, і виходити за рамки звичайних статичних вебсайтів. SPA відтворюється в браузері за методикою роботи звичайного настільного додатка. Архітектура таких веб-застосунків побудована так, що при переході на нову сторінку оновлюється лише частина вмісту. Особливість життєвого циклу SPA можна побачити на рис. 1.1.1. З цього можна зробити висновок, що веб-додатку нема потреби повторно завантажувати ті ж самі елементи сторінки. Насамперед це дуже зручно як і для користувачів, так і для розробників. Найбільш популярні приклади веб-застосунків які розроблені за концепцією SPA є сервіси Google, наприклад служба електронної пошти Gmail або картографічний веб-сервіс Google Maps. Саме для розробки SPA використовують одну з найпопулярніших мов програмування – JavaScript.

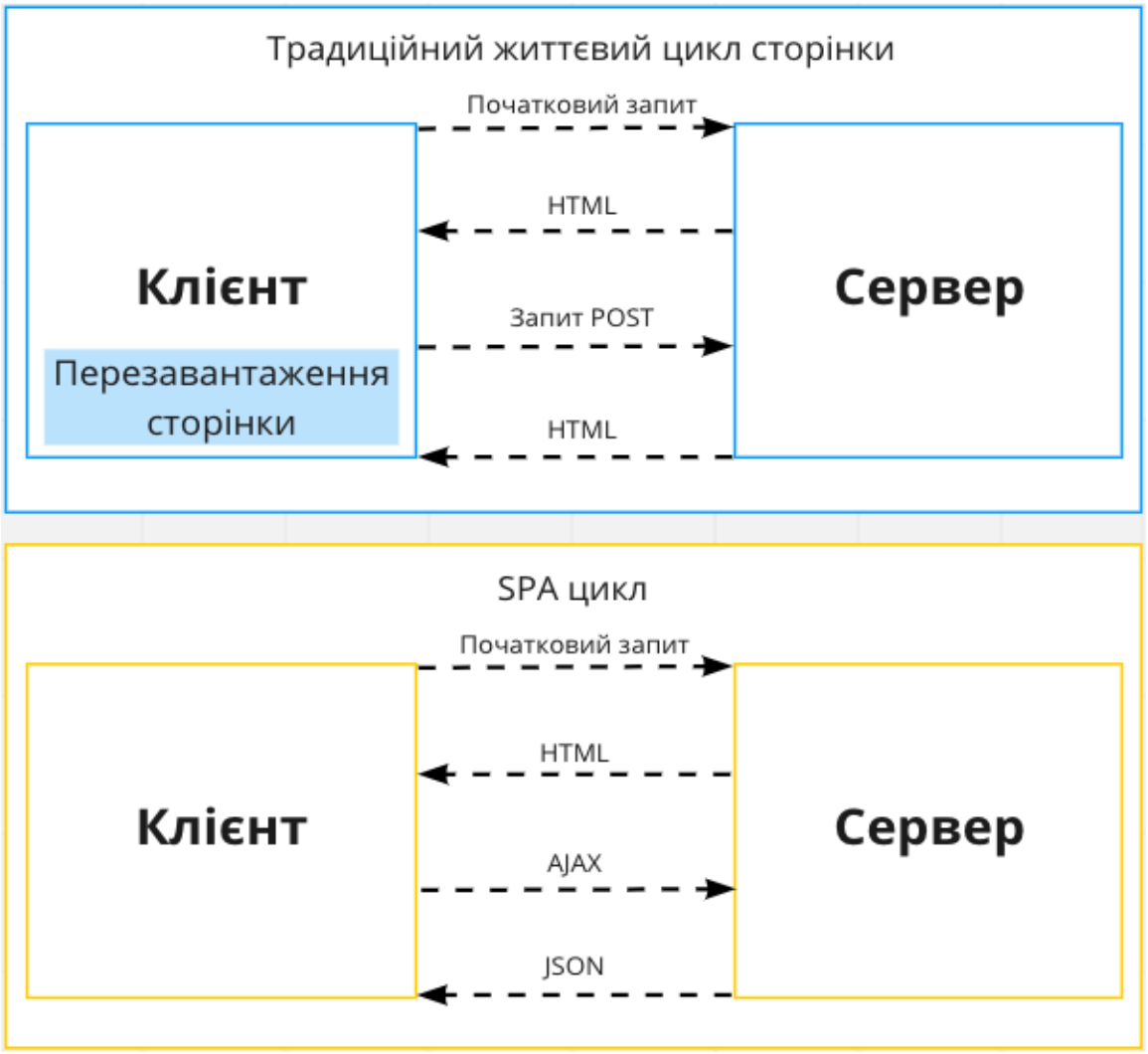


Рисунок 1.1.1 – Особливість життєвого циклу SPA

Веб-додаток може бути розроблений за допомогою бібліотеки jQuery. Але слід зазначити, що jQuery не варто використовувати для розробки великих багатосторінкових проектів. Перш за все, це пов’язано зі структурою коду. jQuery було створення для полегшеної роботи з DOM документа, обробки подій, та простих анімацій[1].

DOM(Document Object Model) – це програмний інтерфейс для HTML і XML документів. DOM надає структуроване уявлення документа і визначає те, як ця структура може бути доступна з програм, які можуть змінювати вміст, стиль і структуру документа. Подання DOM складається з структурованої групи вузлів і об'єктів, які мають властивості і методи. DOM з'єднує веб-сторінку з мовами опису сценаріїв або мовами програмування[2].

Для створення складних проектів рекомендується використовувати більш потужні JavaScript-фреймворки, такі як React, Angular чи Vue.

Фреймворк – програмна платформа, яка визначає структуру програмного забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту[3].

JavaScript-фреймворк – це фреймворк для додатків, написаний мовою програмування JavaScript. JavaScript-фреймворк відрізняється від Javascript бібліотеки потоком управління. Бібліотека містить функції для виклику батьківським кодом, а фреймворк спирається на структуру програми в цілому[3].

Архітектура вище перечислених веб-платформ дозволяє створювати гнучкі веб-застосунки. Також вони включають в собі великий набір готових рішень, що значно прискорює процес розробки. Крім того, на основі даних веб-платформ можна створювати повноцінні мобільні додатки.

Зараз розробники мають достатньо технологій, щоб змінити поведінку традиційного сайту й зробити його схожим на нативний додаток операційної системи. Для цього всі зміни на сторінці веб-сайту роблять за допомогою скриптів, а запити до сервера використовують AJAX.

AJAX (Asynchronous JavaScript And XML) – підхід до побудови користувацьких інтерфейсів веб-додатків, за яких веб-застосунок, не перезавантажуючись, у фоновому режимі за потребою користувача надсилає запити на сервер і сама звідти завантажує потрібні дані[4].

Якщо користувач відкриє такий сайт, то браузеру потрібно лише раз завантажити сторінку, тим самим зміни на сторінці, які не мають необхідності в нових даних, виконуються швидше, оскільки не чекають обробки запитів на стороні сервера.

Отже, незважаючи на назву, суть концепції односторінкового веб-додатку полягає не в обмежені в кількості сторінок веб-сайту, а в наданні сайту логіки роботи як в нативному додатку, за допомогою позбавлення постійних перезавантажень сторінки й перенесенні відтворюючої логіки з сервера до клієнта, водночас зменшуючи залежність роботи сайту від сервера та інтернет з'єднання.

1.2 Порівняння з традиційними веб-сайтами

Односторінковий сайт має такі переваги порівняно з традиційними веб-сайтами:

1) Швидкий і плавний UX (дизайн взаємодії з користувачем).

Легко уявити, як постійне перезавантаження всієї сторінки може завдати шкоди звичайним користувачам. Але неможливо зустрітися з цією проблемою в SPA. JavaScript, CSS та HTML завантажуються лише один раз протягом усього терміну роботи застосунку. Після початкового завантаження, далі сервер надсилає лише нові дані. Це зменшує трафік клієнт-сервер і робить додаток швидким.

У традиційних веб-сайтах сервер повинен генерувати цілі сторінки для тисячі користувачів. Але в SPA рендеринг здійснюється на стороні клієнта. Це означає, що сервер набагато швидше реагує на запити користувачів.

Це робить їх ідеальним кандидатом для перенесення старих настільних програм та рішень корпоративного рівня в Інтернет, не змінюючи їх інтерфейс та робочий процес.

2) Локальне кешування та автономна функціональність.

Після первинного запиту односторонні веб-програми кешують всі дані, які вони отримують від сервера, та зберігають їх локально. Таким чином, SPA можуть продовжувати функціонувати, навіть якщо з’єднання з Інтернетом відсутнє. Коли програма знову підключається до мережі, вона синхронізує локальні дані з сервером і відновлює всю функціональність.

3) Ідеальна мобільність.

Односторінкові веб-застосунки значно легше переносити з Інтернету на мобільні девайси. Як правило, можна використовувати один і той же серверний код як для веб-додатку, так і для версій додатка для iOS/Android, що досить рідко зустрічається за традиційного підходу.

І нарешті, порівняно з традиційним підходом, односторінковий інтерфейс набагато ближчий до інтерфейсу мобільних додатків. Це означає, що не доведеться вносити багато змін у дизайн, адаптуючи свій SPA до мобільного.

4) Повторне використання API (прикладний програмний інтерфейс).

Набагато простіше створити SPA, якщо вже є мобільний додаток. Якщо програма взаємодіє з деякими API на стороні сервера, можна повторно використовувати ті самі API у веб-додатку, не змінюючи їх істотно.

І навпаки, якщо планується запуск власного додатоку, API, який написано для односторінкової веб-програми, стане в нагоді, коли почнеться розробка мобільних пристроїв.

5) Оптимізована розробка.

Розробка додатків на одній сторінці схожа на будівництво будинку за допомогою LEGO. Отримавши компоненти (цеглинки LEGO), можливо швидко побудувати SPA-центр без необхідності додавати однакові фрагменти коду знову і знову. Крім того, не потрібно писати власний код для відтворення HTML-сторінок на сервері.

6) Розділення front-end та back-end інтерфейсів.

У односторінкових додатках інтерфейс користувача відокремлений від даних. Це дозволяє оновити інтерфейс і змінювати інтерфейс, скільки завгодно, не впливаючи на серверну інформацію. Роз’єднання front-end та back-end розробки полегшує пошук помилок у кодовій базі.

Проте, в односторінкового сайту є деякі недоліки:

1) Проблеми з SEO.

Тривалий час Google та інші пошукові системи мали проблеми з індексацією SPA. Їхні боти добре вміли сканувати статичні HTML-сторінки, але були погано підготовлені для відтворення динамічного вмісту. І без можливості виконувати JavaScript, все, що вони бачили в SPA, – це порожній HTML-контейнер.

Як наслідок, односторінкові програми рідко потрапляли до верху результатів пошуку.

За останні пару років Google значно покращив індексацію SPA застосунків.

Проте Google визнає, що іноді не може належним чином індексувати односторінкові програми. Не кажучи вже про те, що інші пошукові системи ще не досягли такої ж майстерності сканування SPA.

2) Повільне перше навантаження.

SPA запускаються повільніше, ніж традиційні програми. Це відбувається через візуалізацію на стороні клієнта. Перш ніж браузер зможе відтворити сторінку, він повинен завантажити громіздкі фреймворки JS. Це може зайняти деякий час, особливо для великих додатків.

Але після першого візуалізації SPA стають набагато швидшими, ніж сайти з традиційним підходом.

На щастя, існують способи прискорити ініціалізацію SPA, такі як динамічне завантаження ресурсів та мінімізація скриптів.

3) Обмежена функціональність із вимкненим JavaScript.

Невеликий відсоток користувачів вимикає JavaScript у своїх браузерах. Причини варіюються від підвищеної швидкості до проблем безпеки. Для таких людей передовий SPA буде абсолютно непридатним.

Частковим рішенням цієї проблеми є використання рендерингу на стороні сервера для першого завантаження сторінки. Однак деякі функції додатка можуть залишатися недоступними, якщо JS вимкнено.

4) Дивна поведінка через асинхронні запити.

SPA керує запитами користувачів асинхронно, і це може призвести до несподіваної поведінки.

Якщо натиснути кнопку або клацнути посилання в традиційній програмі, це зупинить усі поточні дії та зробить новий запит на сервер.

Але в SPA, відповіді сервера надходять асинхронно. Це означає, що вони можуть прибути не в тому порядку, як просили. Отже, якщо натиснути кілька посилань дуже швидко, користувач потрапить на неправильну сторінку. Або можуть виникнути графічні збої, якщо натиснути ту саму кнопку кілька разів.

Розробникам доведеться писати код спеціально для вирішення таких ситуацій.

1.3 Результативна порівняльна таблиця

Виходячи з вищесказаного в підрозділі вище, можна скласти результативно порівняльну таблицю 1.3.1.

Таблиця 1.3.1 – Порівняння SPA з традиційним веб-сайтом

	SPA	Традиційний веб-сайт
Швидкий і плавний UX	+	
SEO оптимізація		+
Мінімальне серверне навантаження	+	
Офлайн функціональність	+	
Мобільна адаптація	+	
Швидкість	+	
Швидкість першого завантаження сторінки		+
Робота без JS		+

Отже за результатами таблиці видно, що SPA має значну перевагу над веб-сайтами з традиційним підходом розробки, навіть якщо він має деякі мінуси, то вони незначні, і з часом вони відійдуть, так як технології не стоять на місці і постійно оновлюються.

Тож в своїй роботі я буду використовувати SPA підхід для розробки front-end частини learning додатку “Campus”.

1.4 Послідовники концепції односторінкових веб-сайтів, їх переваги, недоліки

Angular – фреймворк з відкритим кодом написаний на TypeScript для front-end розробки. Логотип фреймворку зображено на рис. 1.4.1.

Рисунок 1.4.1 – Логотип фреймворку Angular

Angular включає в собі всі основні можливості котрими повинен володіти front-end фреймворк. Він допомагає розробникам керувати користувальницьким інтерфейсом, реагувати на введення з клавіатури, валідувати ці ж введення у формах, керувати маршрутами сторінок, управління станом даних додатку, надсилати запити Ajax, тестувати свій код та багато іншого.

Крім того, існує офіційний CLI, який допомагає вам створювати та керувати проектами Angular, підтримувати їх актуальність, додавати залежності та навіть розгортати.

Angular CLI – це інструмент інтерфейсу командного рядка, який використовується для ініціалізації, розробки, формування та підтримки додатків Angular безпосередньо з терміналу (консолі) [5].

По суті, Angular полягає у створенні повторно використовуваних компонентів користувальницького інтерфейсу, якими потім можна керувати за допомогою Angular, і які можна поєднувати з іншими компонентами, щоб створити цілий користувальницький інтерфейс з цих контрольованих Angular компонентів.

React – це бібліотека JavaScript, розроблена Facebook в 2013 році, яка підходить для створення сучасних односторінкових додатків. Логотип бібліотеки зображено на рис. 1.4.2.

Рисунок 1.4.2 – Логотип бібліотеки React

Там, де Angular дає все, React дає лише одне: бібліотека для рендиренку вмісту DOM і ефективного управління ним після цього. Це також стосується компонентів, а також побудови користувацьких інтерфейсів з компонентів.

Але React не включає вбудовану підтримку перевірки форми. Також не включає маршрутизатор (для відтворення різних компонентів на основі змін URL-адреси) і має власного Http клієнту.

Vue.js – це фреймворк JavaScript із відкритим кодом призначений для розробки користувальницьких інтерфейсів та односторінкових веб-застосунків. Логотип зображено на рис. 1.4.3.

Рисунок 1.4.3 – Логотип фреймворку Vue

Vue знаходиться між React та Angular. Він не такий великий, як Angular, але, безумовно, включає більше функцій, ніж React. Vue надає вбудоване управління станом даних додатку, а також має вбудований маршрутизатор. Однак він не включає перевірку форми чи функціональність клієнта Http.

Вище я провів огляд потужних послідовників розробки SPA, описав їх основні можливості та недоліки. Далі я створив підсумкову таблицю.

Ця таблиця 1.4.1 не включає всі функції вище описаних представників SPA, але замість цього я зосередився на найбільш важливих функціях, які потрібні в багатьох додатках.

Таблиця 1.4.1 – Порівняння послідовників концепції SPA

Особливість	**Angular**	**React**	**Vue**
UI/DOM управління	+	+	+
Управління станом даних	+	+	+
Роутинг	+	-	+
Валідація форм	+	-	-
Http клієнт	+	-	-

Дана підсумкова таблиця показує що Angular володіє найбільшою кількістю основних функцій та можливостей. Приймаючи до уваги, що мета цієї роботи в написанні зручного та розширювального додатку, Angular є припустимим вибором.

1.5 Висновки до розділу

На сьогоднішній день існує безліч фреймворків для комфортної та швидкої розробки односторінкових додатків і кожен з них має свої недоліки й переваги. Я вважаю, що необхідно робити вибір в залежності від цілей та вимог до продукту.

Angular інноваційно підходить до розширення можливостей HTML. У нього велика спільнота і підтримка з боку Google, адже вони є розробниками даного фреймворку, він буде рости і розвиватися, і він добре підходить як для швидкого прототипування, так і для об'ємних проектів.

2 АНАЛІЗ ІСНУЮЧИХ ВЕБ-ЗАСТОСУНКІВ ДЛЯ ДИСТАНЦІЙНОГО НАВЧАННЯ

2.1 Ідея платформ для дистанційного навчання

Без навчання протягом усього життя в сучасному світі не обійтись, адже будь-яка освічена людина має володіти чималим багажем знань. Причому дуже важливо цей багаж постійно оновлювати, інакше важко буде наздогнати стрімкий перебіг життя. А відстати від нього – означає бути не конкурентоспроможним на ринку праці, втратити можливість одержати бажану роботу. Вирішити цю й багато інших проблем допоможе дистанційне навчання[6].

Дистанційне навчання – це спосіб отримання освіти із використанням комп'ютерних та сучасних інформаційних технологій, що надає студентам змогу навчатися на відстані, без відриву від роботи та виїзду за кордон. Серед інших назв дистанційного навчання використовуються і такі, як «відкрита освіта», «електронна освіта», «віртуальне навчання» тощо. Такий спосіб отримання знань передбачає комфортну та зручну для кожного студента обстановку та можливість навчатися без відриву від роботи. На відміну від заочного навчання, з яким часто порівнюють дистанційну форму, остання передбачає не лише постійну самоосвіту та роботу з засвоєння знань, а і постійний контакт як із викладачами, так і з іншими студентами, в той час як заочна форма освіти передбачає спілкування з викладачем лише декілька разів на рік[7].

Дистанційне навчання стало популярним з появою інтернету, відкривши нові можливості розвитку. Спочатку дистанційне навчання сприймалося лише як додатковий спосіб придбання знань або підготовки до іспитів. Зараз можна максимально зручно організувати процес свого навчання за допомогою різних платформ дистанційного навчання.

На теперішній час актуальність поняття платформи дистанційного навчання є невід'ємною частиною нашого життя. Дані платформи зосереджуються на

автоматизації комунікацій та освітніх процесів студентів, учнів, тощо. Наразі студенти можуть використовувати різні платформи дистанційного навчання, деякі для обміну інформацією, деякі для здачі власних лабораторних, складання розкладу і так далі.

Дистанційне навчання має певну кількість переваг перед іншими формами навчання. Так, практично не виходячи з дому чи не покидаючи свого робочого місця, можна підтримувати регулярний контакт з викладачем за допомогою телекомунікаційних технологій, у тому числі відеозв'язку, та одержувати структурований навчальний матеріал, представлений в електронному вигляді. Високий професіоналізм, прагнення до співробітництва, самоствердження і високий рівень комунікації з колегами – це є основними ознаками дистанційного навчання[7].

Проблема полягає в тому, що на всі маніпуляції витрачається багато часу, а саме час на пошук потрібних матеріалів або завдань в різних додатках. Крім того, деякі сповіщення, наприклад, про перенесення заняття, втрачаються в купі інших повідомлень різних платформ. Таким чином виникла ідея про створення зручного додатка, який виконуватиме усі зручні функції для студентів.

2.2 Ідея learning додатку "Campus"

Проаналізувавши велику кількість додатків для дистанційного навчання, було виокремлено багато негативного досвіду користувачів. Саме тому було дуже важливо створити єдиний додаток з усіма зручними функціями для навчання і організації освітнього процесу.

Основна мета даного застосунку – це надати якомога більше користувачам свободи та волі їх думок, зробити освітній процес ефективним та зручним у дистанційному використанні. Однією з особливостей застосунку є доступність. Це означає, що користь з використання даного застосунком можуть винести не тільки викладачі та слухачі в університетах, а й будь-які бажаючі. Наприклад, якщо ви захоплюєтесь програмуванням на фреймворку Angular і ви хочете поділитись своїми

знаннями і спробувати себе в ролі викладача, то ви можете створити в застосунку свій клас і запросити слухачів курсу, серед яких можуть бути ваші друзі або знайомі, а вже далі ви можете ділитись матеріалами, видавати завдання, та покращувати свої навички.

Насамперед я хочу створити повноцінний веб-застосунок. Крім основних функціональних частин в веб-додатку буде присутня реєстрація користувача, вхід у свій профіль, та редагування профілю.

Основні та важливі функції, які вкладаються в розробку даного проекту є реалізація класів по дисциплінах, календаря з подіями, дашборд з основною інформацією для користувача.

Класи по дисциплінах являють собою таку екосистему, де користувач, який створює її, стає автоматично головним в цьому класі, вчителем. Але що ж саме може робити керівник. Насамперед це запрошувати в свій клас слухачів, студентів і так далі. Ділитись з ними матеріалами, видавати домашнє завдання, перевіряти його, і само собою оцінювати.

Також реалізується автоперевірка деяких завдань за допомогою код ранерів – це застосунок, де користувач може запустити свій код виконаного завдання, а система перевірить його та поставить оцінку. Даний код ранер може перевіряти завдання виконані мовою програмування C#.

Функціонал класів по дисциплінам можна розділити на такі функції:

- Видалення класу;
- запрошення слухачів;
- видалення слухачів;
- пошук потрібних класів;
- сортування списку викладачів, студентів за їх критеріями: іменем, оцінками, датою завантаження виконаних завдань;
- створення та перегляд матеріалів лекцій, модулів, практичних завдань;
- перегляд оцінок викладачем, їх фільтрація.

Календар з подіями включає в себе можливість створювати події будь-якого типу, будь це звичайна пара в університеті або нагадування для себе про виконання

якої-небудь задачі. Події може створити кожен користувач, незважаючи на його роль, будь це учень чи викладач. Також в застосунку буде присутня система нагадувань, якщо подія наближається до виконання.

До основних можливостей календаря можна додати ще функціонал, який буде присутній у кожного користувача:

- Видалення події;
- редагування події;
- сортування подій за місяцем, тижнем, днем;
- перетягування події на іншу дату в один клік;
- розширення події одним кліком.

Дашборд з основною інформацією буде включати в себе календар на сьогоднішній день. Нагадування про здачу боргів з дисциплін, якщо користувачі перебувають в класах навчання, останні їх оцінки, та завдання які повинні бути виконаними на цей тиждень.

На сторінці з налаштуванням профілю буде можливість:

- Змінити пароль для аккаунту;
- налаштувати нотифікації;
- змінити інформацію за профілем;
- змінити мову інтерфейсу застосунку.

Вище перелічений функціонал було вкладено у розробку застосунку, далі буде покращуватися, виходячи зі зворотного зв'язку користувачів.

2.3 Популярні аналоги у світі, їх переваги та недоліки.

Дистанційне навчання є сучасною універсальною технологією професійної освіти, орієнтованого на індивідуальні запити студентів та їх спеціалізацію.

Перше, що необхідно зробити – звернути увагу на системи дистанційного навчання, які поширені в усьому світі. ІТ-технології приходять на допомогу викладачам та студентам: існує багато інструментів та продуктів, розроблених для

підтримки онлайн-навчання, і їх можна підключити, не виходячи з дому, маючи лише доступ до інтернету.

На сьогоднішній день у кожного є можливість отримувати знання з найкращих джерел світу, а також зберігати і структурувати їх за допомогою додатків для дистанційного навчання. Проаналізувавши багато додатків, можна дійти висновку, що багато з них не такі зручні у своєму використанні, щоб забезпечити комфорт у навчальному процесі. Багато часу витрачається на пошук необхідної інформації або завдань в різних додатках. Також дуже легко втратити важливі сповіщення в купі інших повідомлень з різних платформ.

Практика віддаленого навчання з'явилася давно і дуже актуальна в наші дні. Така освіта теж є якісною. Розглянемо для прикладу кілька систем. Нижче представлені деякі платформи для дистанційного навчання і порівняння їх з веб-застосунком Campus.

2.3.1 Google Classroom

Google Classroom – це програма управління завданнями, розроблена вчителями та учнями для зв'язку між класами, відстеження їх прогресу. Це онлайн-платформа для дистанційного навчання зав'язана на використанні Google Drive, Google Docs, Sheets and Slides і Gmail. З її допомогою можливо організувати стандартний навчальний процес через інтернет, створювати класи і навчальні групи, додавати до них учнів, завантажувати необхідні навчальні матеріали, відправляти завдання учням, організовувати тематичні обговорення, перевіряти завдання і виставляти оцінки[8].

Переваги сервісу:

- Можливість перевіряти знання слухачів;
- безкоштовність та доступність. У сервісі немає реклами;
- комунікація між учнями та викладачами;
- інтеграцію з сервісами Google.

Недоліки сервісу:

- Відсутність вебінарної кімнати;

- відкрита версії веб-застосунку Google Classroom не включає в себе електронний журнал. Така можливість є тільки для корпоративних користувачів Google Classroom;
- для авторів, які мають особисті акаунти, існують обмеження: кількість учасників курсу не більше 250 і приєднатися до курсу в один день можуть тільки 100 чоловік;
- негнучкий календар (тільки тижневий).

2.3.2 Canvas (LMS)

Canvas – платформа для електронного навчання, спочатку розроблена для навчальних закладів. Це не тільки LMS, але і безліч іншого сумісного ПО. Ця онлайн-платформа дозволяє учням проходити курси, дивитися семінари, лекції та виконувати домашні завдання онлайн. Викладачі та студенти можуть спілкуватися через відео-конференції та популярні месенджери[9].

Переваги платформи можна виділити наступні:

- Адаптивність до потреб користувачів;
- можливість проводити відеоконференції;
- мобільна інтеграція;
- доступна аналітика;
- інтеграція з популярними сервісами, linkedin, twitter.

Недоліки сервісу:

- Перевантажений інтерфейс, часто можна заплутатись у функціоналі веб-застосунку;
- вузька мовна локалізація, в додатку присутня тільки англійська, німецька, португальська та іспанська мова;
- застосунок є платним.

2.3.3 Refreshed Moodle SDN-DUT

Moodle – веб-додаток, що дозволяє створити кастомізовану систему управління навчанням[10].

Переваги:

- Безоплатність;
- необмежена кількість користувачів;
- велика кількість додатків та плагінів;
- можливість установки на будь-який сервер.

Щодо недоліків, хотілось би виділити наступні:

- Адміністрування є складним, заплутаним і не зручним для користувача;
- обмежена звітність;
- для налаштування потрібно мати технічні навички;
- складна документація.

2.3.4 OpenOlat LMS

OpenOlat LMS – це веб-платформа електронного навчання для викладання, навчання, оцінки та спілкування. OpenOLAT включає в себе останні досягнення в галузі освіти, психології навчання, медіа-досліджень і дидактики[11]. Він побудований з використанням найсучасніших технологій, при цьому основна увага приділяється досвіду навчання.

Переваги платформи можна виділити наступні:

- Складна модульна система пропонує авторам курсу широкий вибір дидактичних опцій. Кожна установка OpenOlat може бути індивідуально розширена і, таким чином, адаптована до потреб організації та інтегрована в існуючі IT-структури;
- архітектура, розроблена для мінімального споживання ресурсів, масштабування і безпеки, гарантує виключно надійну роботу;
- є можливість надавати навчальний контент, формувати групи, організовувати користувачів і призначати їх на курси;

Недоліки сервісу:

- Є необхідність мати знання з програмування;
- бракує певних особливостей (У деяких випадках потрібно відрегулювати функції за допомогою кодування) ;

- обмежені можливості підтримки.

2.4 Порівняльна таблиця з аналогами у світі.

В даній порівняльній таблиці 2.4.1 наведено основні можливості застосунку в порівнянні зі схожими додатками у світі. Порівнявши аналоги, було зроблено висновок щодо функцій, яких не вистачає або що можна покращити імплементувавши ці зміни в застосунок “Campus”.

Таблиця 2.4.1 – Порівння з аналогами

Features	**Campus**	**Google Classroom**	**Canvas LMS**	**Moodle**
система push повідомлень	+		+	+
запуск програмного коду у браузері	+			
розподіл дисциплін по віртуальним кімнатам	+	+	+	+
авто-присвоювання оцінки	+		+	
календар з подіями	+	+		+

2.5 Висновки до розділу

В даному розділі було проведено аналіз існуючих додатків для дистанційного навчання, розкриті їх переваги та недоліки. Всі платформи заслуговують право на існування, адже вони значно спрощують процес дистанційного навчання. Моя ж мета зробити корисний застосунок, використовуючи досвід користувачів та зробити це ж навчання зручним та ефективним.

3 КЛІЄНТСЬКА ЧАСТИНА LEARNING ДОДАТКУ “CAMPUS”

3.1 Вибір стеку технологій для веб-застосунку

Отже, як було сказано вище, сучасний фреймворк Angular буде використовуватися для розробки веб-застосунку. Для того, щоб розпочати розробку, потрібно визначитись зі стеком технологій.

Стек технологій – це набір інструментів, що застосовується при роботі в проектах і включає мови програмування, фреймворки, системи управління базами даних, компілятори й так далі.

Для розробки додатку буде використовуватися Angular 10 версії – це одна з останніх версій даного фреймворку. Щоб створити новий проект Angular, я скористуюсь інструментом командного рядка Angular CLI.

Для створення нового проекту, потрібно інсталювати безпосередньо Angular CLI за допомогою команди в терміналі “npm install -g @angular/cli”[12]. Далі прописую команду “ng new campus”, де “campus” це назва проекту. CLI далі буде пропонувати налаштування для проекту, а саме чи потрібен режим “strict”, роутинг, та який формат каскадних таблиць стилів буде використовуватися.

Я не буду використовувати режим “strict” в проекті, тож пропускаю дане питання, роутинг мені потрібен, стилі я буду писати за допомогою препроцесору SASS. Приклад створення нового проекту показано на рис. 3.1.1.

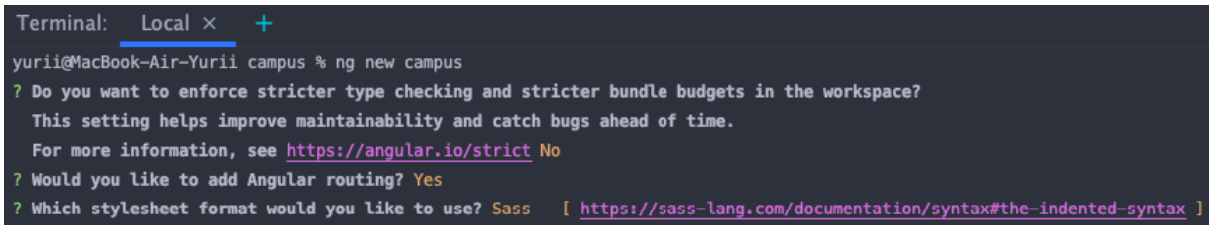


Рисунок 3.1.1 – Створення проекту за допомогою Angular CLI

Після створення проекту, Angular CLI додасть всі залежності до проекту і я можу розпочати розробку застосунку.

Отже, стек основних технологій, які я буду використовувати у веб застосунку це:

- HTML (HyperText Markup Language – мова розмітки гіпертексту) це мова тегів, за допомогою якої здійснюється розмітка веб сторінок в Інтернеті;
- TypeScript – мова програмування, розроблена Microsoft, позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript;
- SASS – це скриптова метамова, яка інтерпретується в каскадні таблиці стилів (CSS) та призначена для підвищення рівня абстракції коду та спрощення файлів CSS;
- Jest – це фреймворк призначений для тестування веб-застосунків.

Саме з вище перечисленими технологіями я можу розпочати розробку learning-додатку “Campus”.

3.2 Початок роботи з Angular

Оскільки встановлено проект за допомогою Angular CLI, тому необхідно звернути увагу на важливі аспектах фреймворку безпосередньо перед розробкою архітектури проекту – а це компоненти, модулі, сервіси та інше.

Компоненти являють собою будівельні блоки додатку. Він містить файл html шаблону, файл каскадних таблиць стилю, тестів, та файл управління станом компоненту[13], приклад компоненту наведено на рис. 3.2.1.

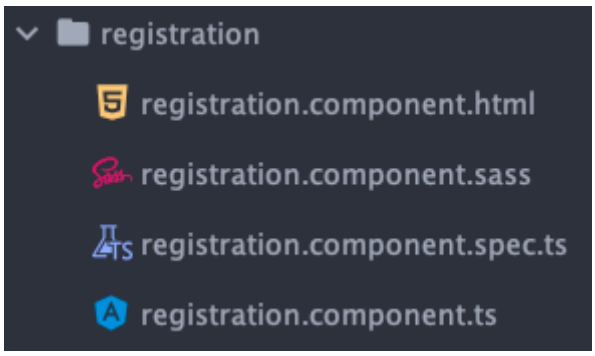


Рисунок 3.2.1 структура компоненту Angular

Файл html-шаблону та стилів зв'язується в контролері компоненту в файлі, який закінчується на ім'я “component.ts” у декораторі “@Component”, як зображено на рис. 3.2.2.

```
@Component({
  selector: 'app-sing-in',
  templateUrl: './sing-in.component.html',
  styleUrls: ['./sing-in.component.sass'],
})
```

Рисунок 3.2.2 – Прив'язка шаблону сторінки та стилів до компонента

На рис. 3.2.2 поле “templateUrl” вказує на шлях до шаблону. А “styleUrls” на файл каскадних таблиць стилю.

Додаток Angular складається з модулів. Вони впорядковують компоненти між собою. Модульна структура дозволяє легко завантажити ті модулі, які необхідні для роботи, і кожен веб застосунок повинен мати хоча б один модуль[14]. Приклад модулю зображено на рис.3.2.3.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
  imports: [BrowserModule, FormsModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule {
}
```

Рисунок 3.2.3 Модуль класу AppModule

Цей модуль, який в цьому випадку називається AppModule, буде початковою точкою в додаток.

За допомогою директиви “import” імпортується ряд потрібних нам модулів. Перш за все, це модуль “NgModule”. Для роботи з браузером також потрібен модуль “BrowserModule”. “FormsModule”, це модуль для роботи з формами. Ось так легко можна підключати додаткові модулі в основний модуль.

Варто зазначити, що в директиві “declarations” підключаються компоненти.

3.3 Діаграма варіантів використання

Перед початком побудови архітектури проекту, доцільно розробити діаграму варіантів використання рис. 3.3.1, тобто опис взаємодії користувача з додатком. В даній схемі буде зображено функціонал зовнішнього інтерфейсу, який доступний користувачу, і як користувач може вплинути на нього, тобто зробити той чи інший крок.

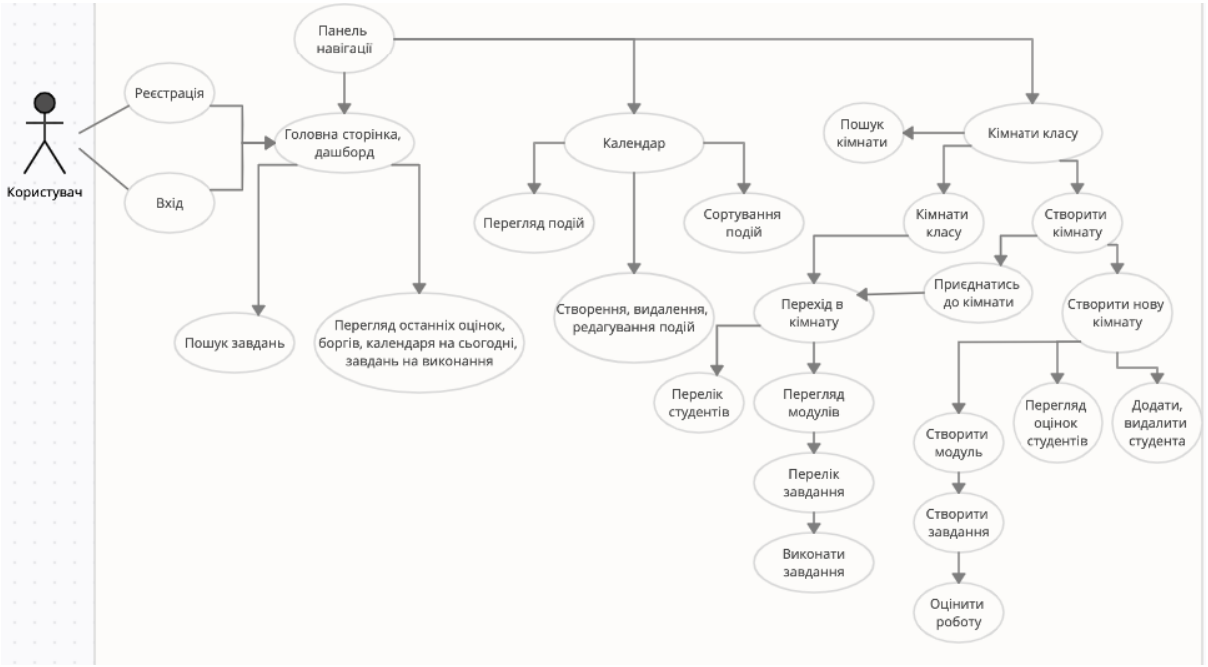


Рисунок 3.3.1 – Діаграма варіантів використання застосунку

3.4 Архітектура застосунку

За допомогою фреймворку Angular з легкістю можна визначити архітектуру веб-застосунку, яка буде логічною та розширювальною. Це можна зробити за допомогою модулів. Архітектуру проекту за модулями показано на рис. 3.4.1.

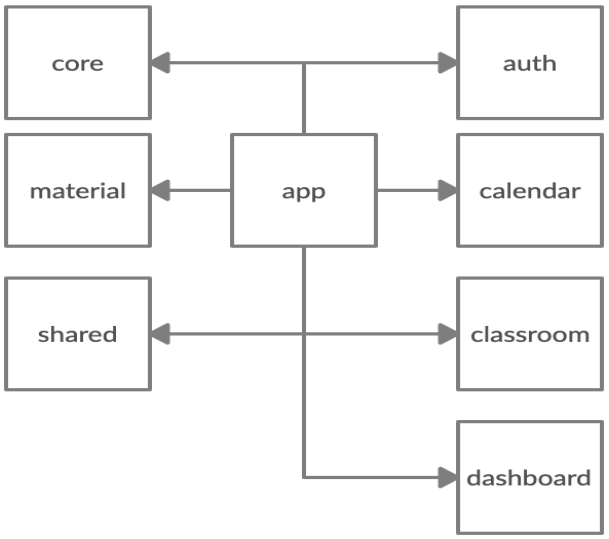


Рисунок 3.4.1 – Архітектура проекту за модулями.

Отже, список основних модулів, які використовуються в застосунку:

- App;
- core;
- shared;
- material;
- auth;
- dashboard;
- calendar;
- classroom.

Модуль app – це кореневий модуль, даний модуль не повинен мати багато вмісту. З нього починається робота додатку та реалізується стартова навігація. В даний модуль доречно інкапсулювати core та shared модуль.

Core – в даному модулі розміщено сервіси, які можуть мати лише один екземпляр, маючи спільний доступ до декількох модулів. Тобто екземпляри служб будуть створені лише в core модулі й не будуть розповсюджуватись в інших функціональних модулях. Ще однією з частин додатку, яка повинна існувати в даному модулі, є компоненти рівня додатка. Це, наприклад компонент навігації.

Shared – це той модуль, де важливо, щоб в ньому було все, що є спільним для всієї програми. Компоненти, модулі та решта частин фреймворку.

Material – модуль, який відповідає за розширення UI компонентів бібліотеки Angular Material.

Angular Material – бібліотека готових UI компонентів для фреймворку Angular.

Модулі, які залишились – це auth, dashboard, classroom та calendar, вони призначені для своїх індивідуальних частин веб застосунку, в них включені всі залежності, які потребує даний сегмент додатку.

Також, щоб зрозуміти основну ідею вище перелічених модулів, потрібно розібратись з маршрутизацією застосунку.

Angular routing – це потужний маршрутизатор JavaScript, побудований і підтримуваний командою Angular, який можна інсталювати з пакета “@angular/router”. Він надає повну бібліотеку маршрутизації з можливістю мати

кілька входів маршрутизатора, різні стратегії узгодження шляхів, легкий доступ до параметрів маршруту та захисту для компонентів від несанкціонованого доступу:.

Однією з переваг розподілу додатку на модулі та використання маршрутизації, є можливість завантажувати певні модулі лише за потребою. При використанні модулів “lazy loading – ліниве завантаження”, завантаження здійснюється лише тоді, коли користувач переходить до маршруту відповідного модуля. З таким підходом веб-застосунок стає значно гнучкішим у використанні та розширюваності додатку.

На рис. 3.4.2 продемонстрована навігація застосунком та показано наявність “lazy loading” модулів.

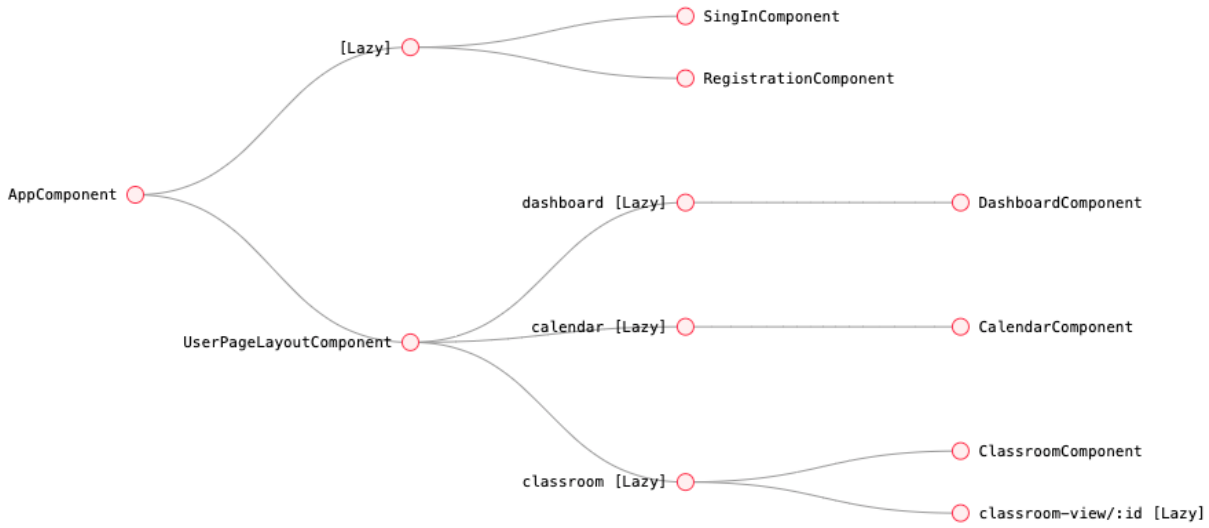


Рисунок 3.4.2 – Схема маршрутизації додатку “Campus”

Тепер загальна структура веб-застосунку виглядає як показано на рис. 3.4.3.

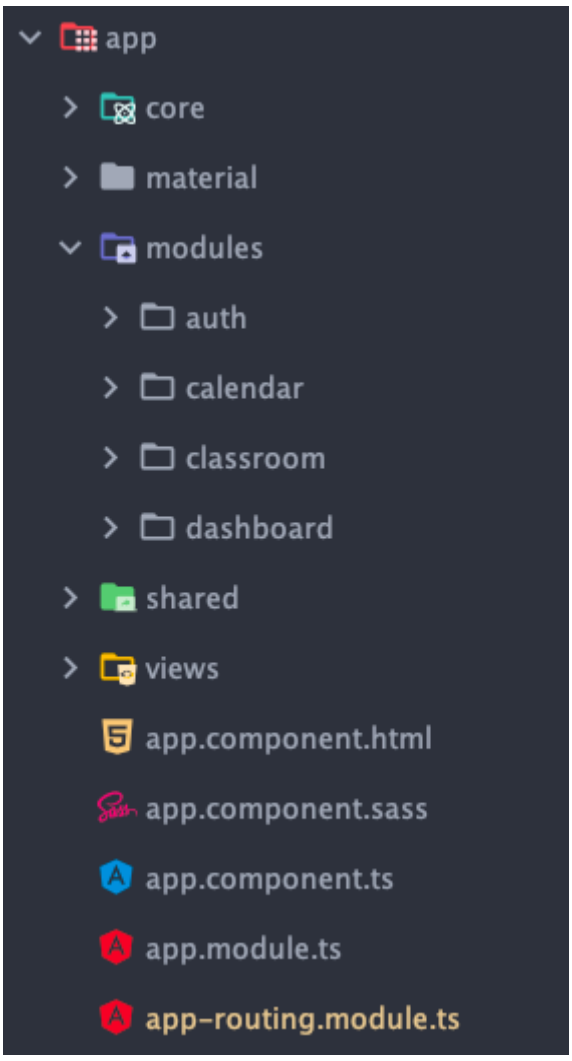


Рисунок 3.4.3 – Структура проекту в середовищі розробки

3.5 Реалізація основних елементів

Так як зі структурою проекту ознайомлено, тепер варто перейти до більш детального опису основних елементів додатку.

3.5.1 Вхід та реєстрація

Компоненти входу та реєстрації знаходять у модулі auth. Даний модуль має лініве завантаження, тобто він викликається за потребою користувача, це означає коли користувач завантажує перший раз застосунок, то шляхом оптимізації та лінивого завантаження модулів сторінка відкривається миттєво, оскільки немає

потреби завантажувати логіку інших модулів поки користувач не аутентифікувався. Приклад реалізації лінивого завантаження реалізовано в модулі app-routing рис. 3.5.1.

```
const routes: Routes = [
  {
    path: '', loadChildren: () => import('./modules/auth/auth.module').then(m => m.AuthModule)
  },
  {
    path: 'campus', component: UserPageLayoutComponent, canActivate: [CanActivateDashboardGuard],
    children: [
      {
        path: 'dashboard', loadChildren: () => import('./modules/dashboard/dashboard.module').then(m => m.DashboardModule)
      },
      {
        path: 'calendar', loadChildren: () => import('./modules/calendar/calendar.module').then(m => m.CalendarModule)
      },
      {
        path: 'classroom', loadChildren: () => import('./modules/classroom/classroom.module').then(m => m.ClassroomModule)
      }]
  },
];
```

Рисунок 3.5.1 – Ліниве завантаження модулів

На вищевказаному рисунку показана основна навігація за модулями, які відповідають за інтерфейс користувача. В об'єкті навігації в значення ключа path вказано шлях до посилання, за яким буде доступний той чи інший модуль, далі наступним параметром для лінивого завантаження вказується стрілкова функція під назвою loadChildren, результатом якої буде шлях до конкретного модуля додатку.

Задля того, щоб навігація працювала доречно, в компонент шаблону кореневого компонента app потрібно додати селектор “router-outlet”, рис. 3.5.2.

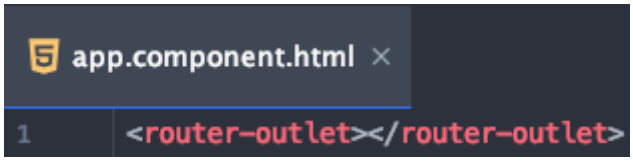


Рисунок 3.5.2 – Селектор “router-outlet”

Принцип роботи даного селектору полягає в тому, щоб динамічно заповнювати html-шаблон на основі поточного стану маршрутизатора.

Сторінка входу та реєстрації має власну валідацію на рівні інтерфейсу користувача, яка реалізована за допомогою влаштованого в фреймворк модуля “FormsModule”[15]. Приклад реалізації валідації показано на рис. 3.5.3.

```
this.form = new FormGroup( controls: {
  email: new FormControl( formState: null,  validatorOrOpts: [Validators.required, Validators.email]),
  password: new FormControl( formState: null,  validatorOrOpts: [
    Validators.required,
    Validators.minLength( minLength: 8),
  ])
});
```

Рисунок 3.5.3 – Валідація форми

Валідація розповсюджується для поштової скриньки та паролю, і за допомогою класу Validators можна додавати тип валідації, який нам потрібен для конкретного поля. В html-шаблоні для поєднання логіки компоненту потрібно до тегу input додати атрибут formControlName та відповідну назву поля для якого прописана валідація. Приклад html-шаблону з полем для валідації зображено на рис. 3.5.4.

```
<div class="form-control" [ngClass]="{
    invalid: form.get('email').touched && form.get('email').invalid}">
  <label for="email" class="email"></label>
  <input id="email"
         type="email"
         placeholder="{{ 'AUTH.SIGN-IN.FORM-TEXTBOX.EMAIL' | translate }}"
         formControlName="email"/>
  <div *ngIf="form.get('email').touched && form.get('email').invalid" class="validation">
    <small *ngIf="form.get('email').errors.required">
      {{ 'AUTH.SIGN-IN.ERROR-MESSAGES.EMAIL.REQUIRED' | translate }}
    </small>
    <small *ngIf="form.get('email').errors.email">
      {{ 'AUTH.SIGN-IN.ERROR-MESSAGES.EMAIL.PATTERN' | translate }}
    </small>
  </div>
</div>
```

Рисунок 3.5.4 – Валідація поля в html-шаблоні

Попередження з приводу валідації на стороні інтерфейсу користувача реалізовані за допомогою структурної директиви “*ngIf”, це директива, яка умовно включає шаблон на основі значення виразу, приведеного до логічного значення. Тож якщо вказана умова, а саме якщо до поля доторкнулися та дані не валідні, то потрібно показати помилку. Приклад роботи валідації продемонстрований на рис. 3.5.5.

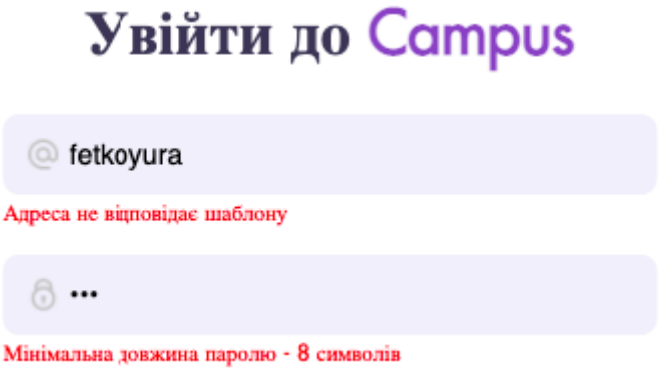


Рисунок 3.5.5 – Валідація в дії

Таким чином можна легко налаштувати валідацію форми за допомогою фреймворку не звертаючись за допомогою сторонніх бібліотек чи написанню власних валідаторів.

Далі важливим аспектом входу чи реєстрації користувача є відправка даних на сервер та обробка відповіді останнього. Як правило для цього в Angular існують сервіси.

Сервіси Angular – це самостійні об’єкти, які створюються лише один раз протягом усього життя програми. Головною метою сервісів є організація та обмін бізнес-логікою, моделями або даними та функціями з різними компонентами додатку Angular.

Приклад роботи “SignInService” сервісу показано на рис. 3.5.6.

```
@Injectable()
export class SignInService {
  constructor(private http: HttpClient) {
  }

  login(user: AuthenticatedUser): Observable<string> {
    return this.http.post<string>(environment.authenticateProfile, user);
  }

  getProfileInformation(): Observable<ProfileInformation> {
    return this.http.get<ProfileInformation>(environment.getProfileInformation);
  }
}
```

Рисунок 3.5.6 – Сервіс класу SignInService

Вище показаний сервіс має два методи, це “login”, який відповідає за вхід у свій профіль, запит відправляється за допомогою модуля “HttpClient”, який доступний в фреймворку. Сервер нам повертає рядок з токеном авторизації. Даний токен призначений для безпеки користувача. В проекті використаний “Bearer Authentication”.

Bearer Authentication (Аутентифікація маркера) – це схема автентифікації HTTP, яка включає маркери безпеки, які називаються токенами-носіями. Маркер – це загадковий рядок, який зазвичай генерується сервером у відповідь на запит на вхід. Клієнт повинен надіслати цей маркер у заголовку авторизації під час надсилання запитів до захищених ресурсів у вигляді “Authorization: Bearer <token>”[16].

Даний токен буде використовуватися у “guard” та “interceptor” класів Angular, які знаходяться в *core* модулі.

Guard (охоронець) дозволяє обмежити навігацію за визначеними маршрутами. Наприклад, якщо для доступу: до певного ресурсу потрібна наявність аутентифікації або наявність інших умов, в залежності від яких користувачеві буде наданий доступ до ресурсу, або навпаки, не наданий.

В застосунку використано guard під назвою “CanActivate”, реалізація методу CanActivate показано на рис. 3.5.7.

```
canActivate(
  next: ActivatedRouteSnapshot,
  state: RouterStateSnapshot): Observable<boolean> {
  return this.signInService.getProfileInformation().pipe(
    map( project: () => {
      return true;
    }),
    catchError( selector: () => {
      const toastStyles = {
        toastClass: 'ngx-toastr server-error-toastr'
      };

      this.router.navigate( commands: ['']);

      zip(
        this.localeService.get( key: 'AUTH.ERROR-TOASTR.NOT-AUTHORIZED'),
        this.localeService.get( key: 'AUTH.ERROR-TOASTR.HEADER')
      ).toPromise().then(([message : string , header : string ]) => this.toastr.error(message, header, toastStyles));

      return of( args: false);
    })
  );
  }
}
```

Рисунок 3.5.7 – Реалізація CanActivate метода.

Даний guard використано для захисту маршрутів, які будуть доступні лише після авторизації користувача. Тобто неавторизований користувач не буде мати можливості перейти на сторінку календаря, класів по дисциплінах, тощо. Замість цього він побачить помилку зображену на рис. 3.5.8, яка прописана у методі.

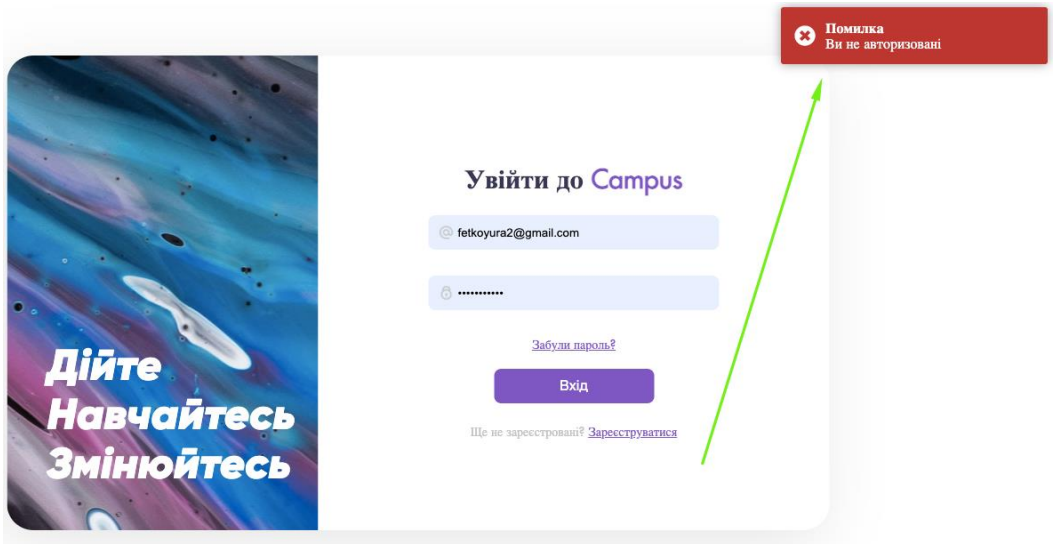


Рисунок 3.5.8 – Робота методу Guard

Angular HTTP Interceptor дозволяє перехоплювати HTTP-запити перед їх відправкою і вносити в них необхідні зміни. В даному випадку перед кожним запитом на сервер буде додано токен користувача для отримання даних. Це потрібно насамперед для безпеки. Приклад реалізації зображено на рис. 3.5.9.

Рисунок 3.5.9 – HTTP Interceptor

В даному підпункті розглянуто основні можливості реалізації компоненту входу та реєстрації застосунку. Продемонстровано роботу валідації форм та інших важливих частин застосунку такі як: guards, interceptors, робота з запитами на сервер та обробка результату.

3.5.2 Календар з подіями

Календар з подіями має свій окремий лінивий завантажувальний модуль. Він містить безпосередньо компонент календаря та компоненти модальних вікон створення, редагування, та видалення події, інтерфейси, якими описано дані подій та сервіс, за допомогою якого проводиться запити та обробка даних на сервер.

Каркас календаря представлений бібліотекою javascript “fullcalendar”.

Створити подію можна різними способами. Перший – натиснути кнопку “Нова подія”. Другий спосіб – це натиснути на будь-яку дату на календарі як показано на рис. 3.5.10.

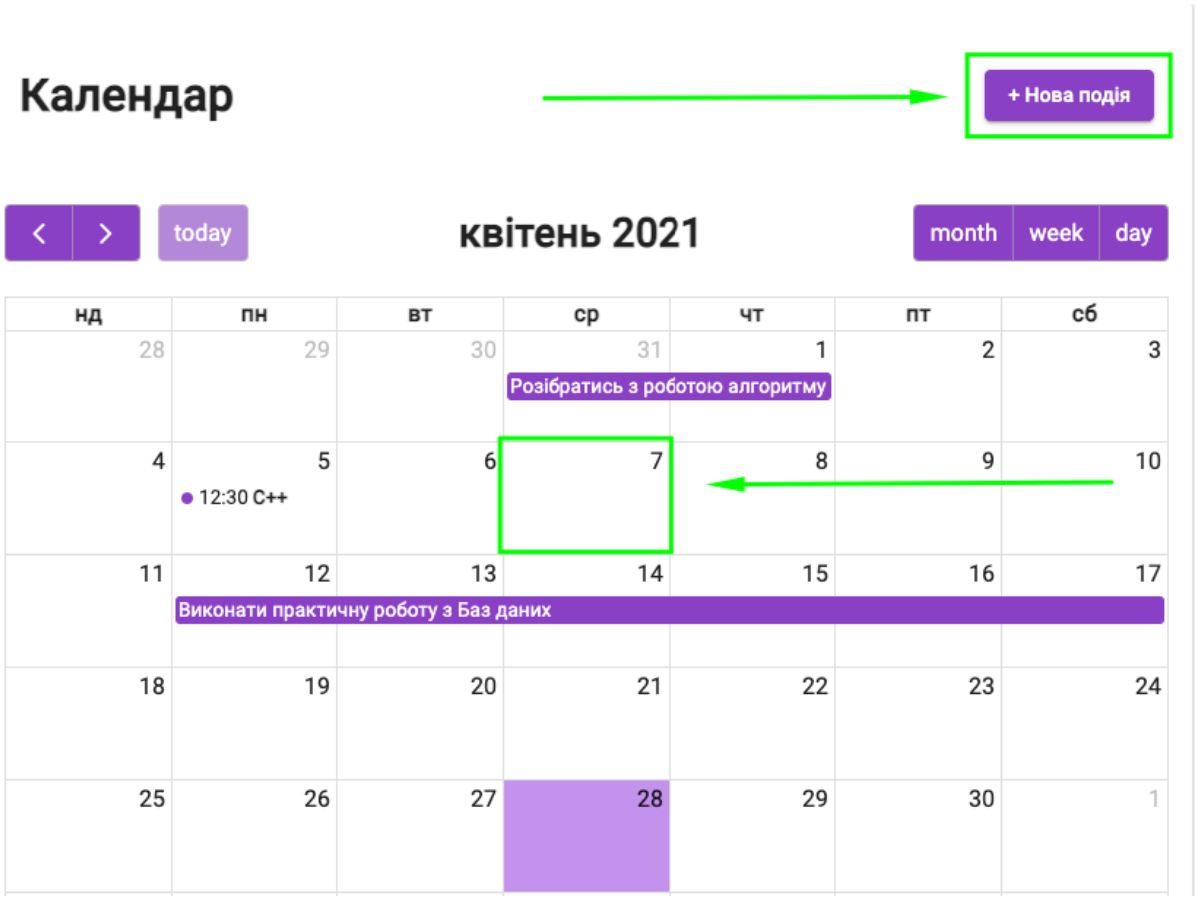


Рисунок 3.5.10 – Способи творення нової події

Далі для користувача відкриється модальне вікно з формою, в якому потрібно ввести основні дані про подію, приклад зображено на рис. 3.5.11.

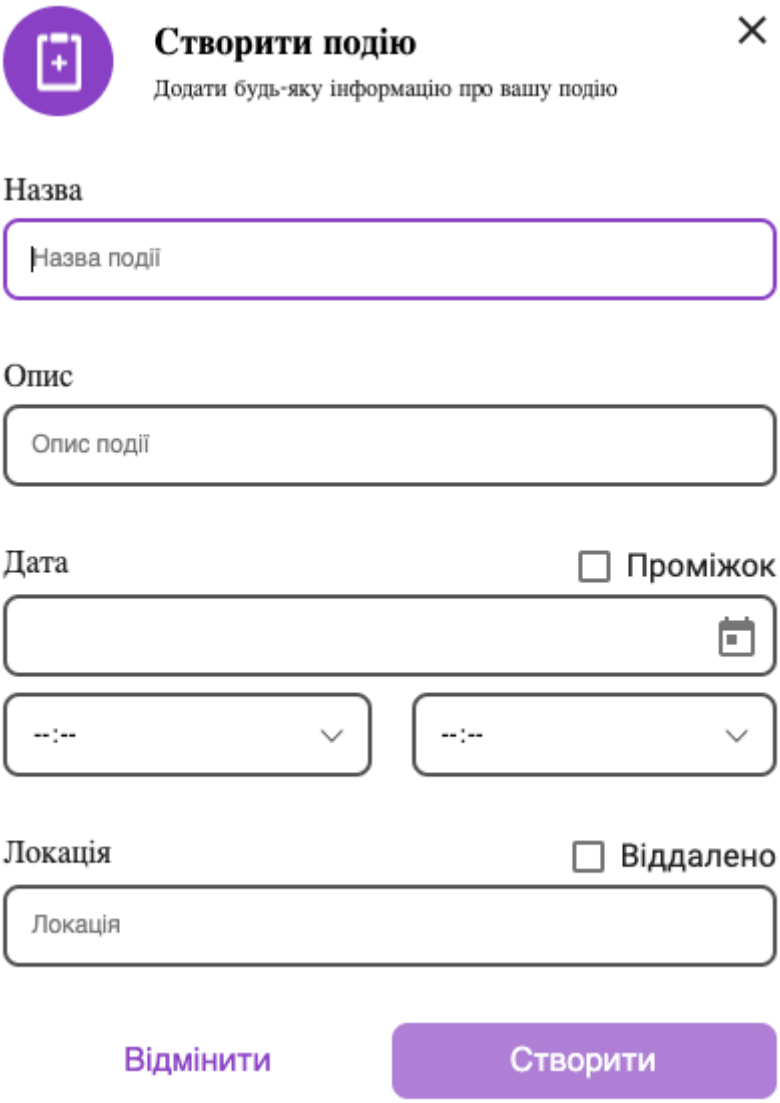


Рисунок 3.5.11 – Модальне вікно створення події

В модальному вікні присутня валідація форми, а саме вказано, які поля обов'язкові й, після заповнення яких, кнопка створення події буде активна. Натиснувши кнопку “Створити” запит відправляється на сервер. Приклад відправки даних на сервер продемонстровано на рис. 3.5.12.

```
const event: CalendarEventForm = {
  title: this.dialogForm.value.summary,
  start: startDate,
  end: endDate,
  location: this.dialogForm.value.location,
  description: this.dialogForm.value.desc,
  allDay: allDayCheck
};

this.addEventSub = this.calendarService.addEvent(event)
  .subscribe( next: resId => {
    this.dialogRef.close( dialogResult: {
      ...event,
      id: String(resId)
    });
```

Рисунок 3.5.12 – Відправка новоствореної події на сервер.

Як показано на рисунку вище у сервісу calendarService викликано метод addEvent, в який передано об'єкт новоствореної події. Далі підписуємось на подію, це метод subscribe, щоб отримати результат від сервера.

Методи класу HttpClient після виконання запиту повертають об'єкт Observable, який визначений в бібліотеці RxJS, яка входить в фреймворк. Ця бібліотека реалізує патерн "асинхронний спостерігач" (asynchronous observable). Позаяк, виконання запиту до сервера за допомогою класу HttpClient виконуються в асинхронному режимі.

Тож після успішної відповіді від сервера, потрібно відписатись від слухача, це потрібно задля того, щоб запобігти витоку пам'яті, тим самим покращити оптимізацію роботу додатку. Через те, що в змінну addEventSub записана підписка слухача події, відписатись можливо, використавши метод unsubscribe() при виході з компоненту як показано на рис. 3.5.13.

```
ngOnDestroy() {
  if (this.addEventSub) {
    this.addEventSub.unsubscribe();
  }
}
```

Рисунок 3.5.13 – Відписатись від слухача події

В підпункті було розглянуто принципи роботи додавання нової події, логіка редагування та видалення тісно переплітається з нею, відрізняючись тільки методами реалізації. Також була продемонстрована на практиці робота зі слухачами події та їх відпискою бібліотеки RxJS.

3.5.3 Кімната класу

Кімната класу, так само як і календар з подіями, має свій окремий модуль лінивого завантаження. В цьому підпункті важливо виділити роботу з навігацією всіх сторінок які відносять до кімнати класу. Адже вона містить сторінки користувачів, завдання, та оцінки студентів, як показано на рис. 3.5.14.

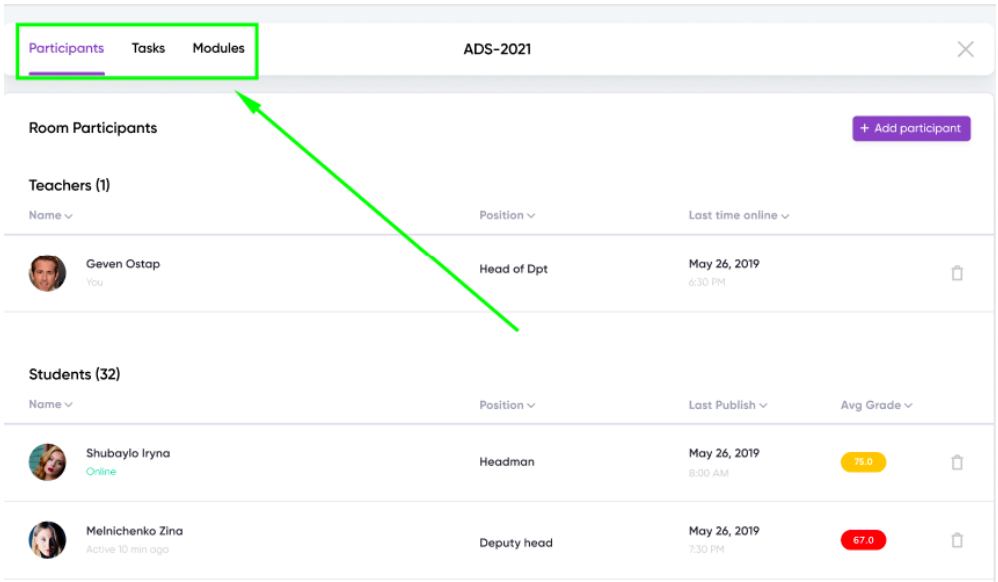


Рисунок 3.5.14 – Навігація кімнати класу

Для оптимізації роботи навігації при натисканні на вище перечислені посилання, потрібно оновлювати контент, а сам блок навігації залишається без змін і використовується лише один раз на сторінці.

Це можна зробити за допомогою модулю навігації, приклад зображено на рис.3.5.15.

```
const routes: Routes = [
  {
    path: '',
    component: ClassroomViewComponent,
    children: [
      {
        path: 'participants',
        component: ClassroomParticipantsComponent
      },
      {
        path: 'modules',
        component: ClassroomModulesComponent
      },
      {
        path: 'grades',
        component: ClassroomGradesComponent
      },
      {
        path: '**',
        redirectTo: 'participants'
      }
    ]
  }
];
```

Рисунок 3.5.15. – Програмна навігація кімнати класу

На рисунку присутній масив об'єктів навігації – children. Він відповідає за додаткову навігацію для данного компонента. Для відтворення контенту в цьому ж компоненті потрібно додати селектор “<router-outlet>” і тепер додатковий контент буде відтворюватись на сторінці кімнати класу за запитом користувача не змінюючи відтворення основної навігації.

Наступним важливим кейсом реалізації логіки роботи кімнати класу є відтворення списку самих кімнат. Через те, що кімнат може бути будь-яка кількість, а нам потрібно відтворити всі, то для кімнати було створено окремий компонент який відповідає за html-шаблон кімнати, а виведення цих кімнати було реалізовано за допомогою структурної директиви “*ngFor”, яка циклічно наповнює html-шаблон даними, в даному випадку кімнатами. Приклад продемонстрований на рис. 3.5.16.

```
<div class="classrooms">
  <ng-container *ngFor="let classroom of classrooms">
    <app-classroom-item [classroomItem]="classroom"></app-classroom-item>
  </ng-container>
</div>
```

Рисунок 3.5.16 – Циклічне виведення кімнат класу

В цикл передано масив кімнат, який отримали від серверу, а всередині контейнеру передаємо компонент, який потрібно відтворити на сторінці. Як результат рис.3.5.17.

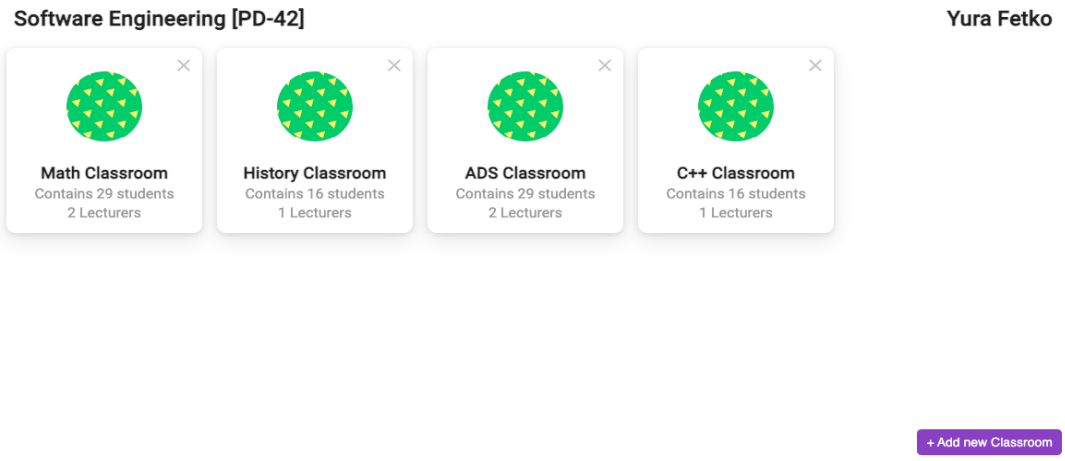


Рисунок 3.5.17 – Кімнати класу

Таким чином не важлива кількість даних в масиві, таким коротким записом можна відтворити будь-які дані.

3.6 Висновки до розділу

В даному розділі була продемонстрована архітектура проекту, показані основі можливості фреймворку Angular та на прикладі learning-додатку “Campus” показана реалізація основних частин застосунку. Реалізації решти компонентів відбувається за такими ж принципами розробки.

4 ТЕСТУВАННЯ ВЕБ ДОДАТКУ

4.1 Unit-тестування

Unit-тестування – це перевірка коректності роботи окремих частин додатку в ізольованому середовищі незалежно один від одного.

На практиці unit-тести пишуться для складних функцій[17], щоб запускати їх при внесенні змін до початкового коду проекту з метою виявлення регресійних помилок.

Регресійна помилка – помилка, яка виникає в роботі функції, при внесенні змін до іншої частини програми.

Для unit-тестування в Angular додатках використано фреймворк Jest.

Jest – це повнофункціональний фреймворк тестування від Facebook. Він має у собі все необхідне для модульного тестування програм JavaScript[17].

Переваги модульного тестування:

− Дозвіл на рефакторинг коду, оскільки наявність тестів дає гарантію, що все працює належним чином, зміни до коду можуть бути внесені з упевненістю, що не отримаємо жодних помилок;

− додавання нових функцій не руйнуючи нічого, якщо додати нову функцію, можна запустити тести на перевірку, щоб переконатися, що будь-якій іншій частині додатку не нашкоджено;

− тести – це перш за все документація та можливість пояснити як код буде працювати в тих чи інших випадках та побачити його зі сторони, щоб зрозуміти його слабкі сторони.

Для того, щоб встановити Jest потрібно запустити в командному рядку проекта команду: “npm install jest jest-preset-angular @types/jest --save-dev”. Після її виконання усі залежності які потрібні для даної бібліотеки будуть встановлені у застосунок.

Перед початком написання тестів потрібно налаштувати конфігурацію фреймворку для тестування. Для цього потрібно в корені застосунку створити файл jest.config.js та задати умови конфігурації як показано на рис. 4.1.1.

```
const {pathsToModuleNameMapper} = require("ts-jest/utils");
const {compilerOptions} = require('./tsconfig');

module.exports = {
  preset: 'jest-preset-angular',
  roots: ['<rootDir>/src'],
  transform: {'^.+\\.tsx?$': 'ts-jest'},
  testMatch: ['**/+(*.)+(spec).+(ts)'],
  setupFilesAfterEnv: ['<rootDir>/src/test.ts'],
  moduleFileExtensions: ['ts', 'tsx', 'js', 'jsx', 'json', 'node'],
  collectCoverage: false,
  clearMocks: true,
  moduleNameMapper: pathsToModuleNameMapper( mapping: compilerOptions.paths || {}, {prefix: {
    prefix: '<rootDir>/'
  })
};
```

Рисунок 4.1.1. – Конфігурація Jest

За допомогою цієї конфігурації jest можна виділити основні права даного фреймворку, а саме, що він може робити з файлами для тестування:

- Шукати всі файли .scec.ts у теці проекту;
- скомпілювати код TypeScript та записати його в пам'ять перед запуском тестів.

4.2 Приклади Unit-тестів веб-застосунку

В даному підпункті я хочу виділити увагу на Unit-тести які написані для проекту, а конкретно для сервісу “sign-in”. Тому, що цей сервіс має важливу роль в проекті, а саме відповідає за спілкування клієнта з сервером, тобто відправляємо дані на сервер, а сервер у відповідь вирішує чи можна даного користувачу авторизуватись в додатку.

Сервіс має два методи це login – який відповідає за вхід користувача в додаток, та метод getProfileInformation – в який відправляється токен доступу: користувача та перевіряємо його актуальність. Для методу login було написано два тест кейси під назвою: “should return token when user logged in”, “should respond with 400 error when

wrong email or password”. Для прикладу розберемо перший тест кейс, який зображено на рис. 4.2.1.

```
test( name: 'should return token when user logged in', fn: () => {
  const mockToken: string = 'token';

  const authenticatedUserMock: AuthenticatedUser = {
    email: 'exmple@example.com',
    password: 'Password321'
  };

  signInService.login(authenticatedUserMock).subscribe( next: response =>
    expect(response).toEqual(mockToken));

  const req = httpTestingController.expectOne( url: '/api/profile/auth');
  expect(req.request.method).toMatch( expected: 'POST');

  req.flush(mockToken);
});
```

Рисунок 4.2.1 – Тест кейс “should return token when user logged in”

На вищевказаному рисунку продемонстрований тест кейс, який перевіряє роботу метод login. Тест повинен повертати токен коли користувач авторизувався.

З самого початку потрібно створити дані для входу і перевірки результату роботи методів. За допомогою методу expect, який визначений в фреймворку jest як очікуваний результат, передано відповідь, яку поверне метод login, а в метод toEqual передаємо очікувану відповідь. Якщо відповідь, яку повернув метод login аналогічна відповіді, яку ми передали в метод toEqual, то алгоритм роботи функції працює вірно.

Наступний метод expectOne, який знаходиться в модулі httpTestingController[18] перевіряє кількість викликів запиту на сервер, а саме, щоб відповідь від сервера повернулась після першого запиту, а в методі expect очікуємо виклик POST запиту на сервер.

В кінці тесту викликається метод flush у результату змінної req, щоб упевнитися що запит відпрацював. Для перевірки коректності роботи всіх юніт тестів необхідно запустити їх та переконатися, що все працює вірно, результат роботи показано на рис. 4.2.2.

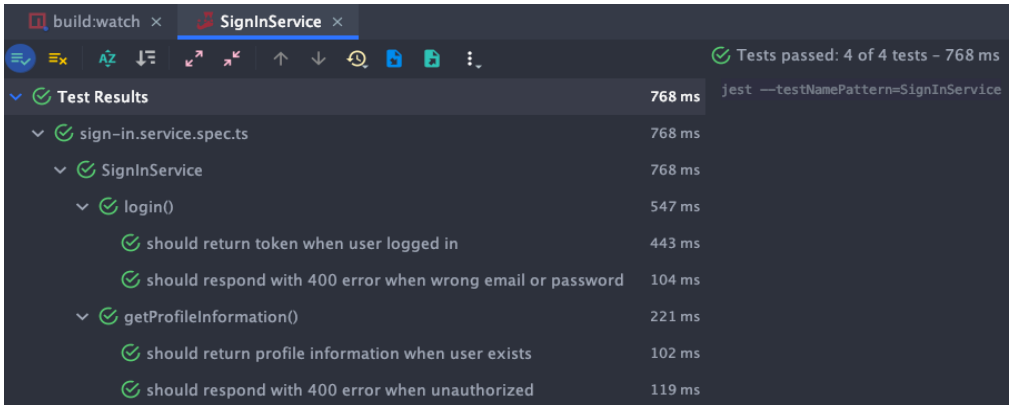


Рисунок 4.2.2 – Запуск тест кейсів сервісу sign-in

Отже, всі тести виконані, тим самим можна бути впевненим в коректності роботи сервісу та застосунку в цілому. Далі за аналогією роботи вищевказаного тесту кейсу пишуться юніт тести для інших частин застосунку.

Однією з особливостей фреймворку для тестування jest є таблиця покриття тестами усіх частин веб-застосунку. Покриття тестами веб-додатку “Campus” показано на рис. 4.2.3.

All files

82.35% Statements 98/119 **50%** Branches 6/12 **70%** Functions 21/30 **79.38%** Lines 77/97

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
app/core/enum	100%	4/4	100%	2/2	100%	1/1	100%	4/4
app/core/services	100%	23/23	100%	2/2	100%	10/10	100%	18/18
app/modules/auth	16.67%	1/6	0%	0/2	0%	0/1	16.67%	1/6
app/modules/auth/registration	59.46%	22/37	0%	0/4	14.29%	1/7	58.82%	20/34
app/modules/auth/shared/services	100%	8/8	100%	0/0	100%	2/2	100%	6/6
app/modules/classroom/shared/components/classroom-item	85.71%	6/7	100%	0/0	0%	0/2	80%	4/5
app/modules/classroom/shared/components/modals	100%	5/5	100%	0/0	100%	2/2	100%	3/3
app/modules/dashboard/dashboard	100%	5/5	100%	0/0	100%	2/2	100%	3/3
app/shared/date-formats	100%	2/2	100%	0/0	100%	0/0	100%	2/2
app/shared/date-formats/directives/date-format1	100%	7/7	100%	0/0	100%	1/1	100%	5/5
app/shared/date-formats/directives/date-format2	100%	7/7	100%	0/0	100%	1/1	100%	5/5
app/shared/pipes	100%	7/7	100%	2/2	100%	1/1	100%	5/5
environments	100%	1/1	100%	0/0	100%	0/0	100%	1/1

Рисунок 4.2.3 – Відсотки покриття unit тестами додатку “Campus”

У вищевказаному звіті зазначено, що він охоплює 82.35% statement, 50% branches, 70% functional та 79.38% Lines. Тепер необхідно розглянути кожне з визначень для розуміння цих термінів:

- Functional coverage (покриття функцій) – показує відсоток викликаних функцій застосунку.
- statement coverage (покриття інструкцій) – відсоток виконаних виразів у додатку.
- branch coverage (покриття логічних виразів) – відсоток покриття кожного кроку програми, наприклад чи виконуються умови оператора “if”, якщо так, то чи виконується його умова “else”.
- line coverage (покриття рядків) – відсоток виконаних рядків у вихідному файлі.

Тож даний звіт покриття тестів допомагає зрозуміти, наскільки ефективні тест кейси та чи охоплюється вся область окремих модулів чи ні.

4.3 Висновки до розділу

В даному розділі була дана відповідь на запитання, що таке unit тести та для чого вони потрібні, наведені переваги їх використання. Розглянута робота фреймворку для тестування Jest, його налаштуванням, наведені приклади реалізації тестів. Показана робота зі звітністю про покриття тестами на прикладі веб застосунку “Campus”.

ВИСНОВКИ

В результаті виконання даної роботи, була досягнута ціль – розроблено learning-додаток “Campus”. Також були вирішені такі задачі:

1) Аналіз наявних застосунків для дистанційного навчання, наведено їх переваги та недоліки.

2) Визначено вимоги та функціональні можливості learning-додатку “Campus”.

3) Проведено аналіз програмних засобів для розробки додатку, визначили, що Angular має багато переваг перед своїми конкурентами.

4) Проектування та розробка веб-застосунку “Campus”, проведено аналіз частин проекту, та розроблено архітектуру проекту яка є оптимальною та може бути розширена в майбутньому.

5) Покриття unit-тестами створеного додатка для підтвердження його коректної працездатності.

Перспективи проекту. Надалі планується покращення працездатності розробленого веб-застосунку та наповнення його новим функціоналом виходячи зі зворотного зв'язку з користувачами.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1) What is jquery [Електронний ресурс] – Режим доступу: https://www.tutorialsteacher.com/jquery/what-is-jquery (дата звернення 29.04.2021). – Назва з екрана.

2) Introduction to the DOM [Електронний ресурс] – Режим доступу: :https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (дата звернення 29.04.2021). – Назва з екрана.

3) What is framewirks [Електронний ресурс] – Режим доступу: :https://hackr.io/blog/what-is-frameworks (дата звернення 29.04.2021). – Назва з екрана.

4) Handbook style web [Електронний ресурс] – Режим доступу: http://style-web.com.ua/handbook-style-web.phtml (дата звернення 29.04.2021). – Назва з екрана.

5) Angular CLI [Електронний ресурс] – Режим доступу: https://angular.io/cli (дата звернення 29.04.2021). – Назва з екрана.

6) Дистанційна освіта [Електронний ресурс] – Режим доступу: :https://ispirina.blogspot.com/ (дата звернення 29.04.2021). – Назва з екрана.

7) Що таке дистанційна освіта: як вона працює? [Електронний ресурс] – Режим доступу: http://www.vsemisto.info/osvita/2355-sho-take-vysha-osvita-jakvona-prazjuje (дата звернення 29.04.2021). – Назва з екрана.

8) Дистанційне навчання з використанням Google Classroom [Електронний ресурс] – Режим доступу: https://evergreens.com.ua/ua/articles/lms-comparison.html (дата звернення 29.04.2021). – Назва з екрана.

9) What is Canvas? [Електронний ресурс] – Режим доступу: https://community.canvaslms.com/t5/Canvas-Basics-Guide/What-is-Canvas/ta-p/45 (дата звернення 29.04.2021). – Назва з екрана.

10) About Moodle [Електронний ресурс] – Режим доступу: https://docs.moodle.org/310/en/About_Moodle (дата звернення 29.04.2021). – Назва з екрана.

11) OpenOLAT [Електронний ресурс] – Режим доступу: https://elearningindustry.com/directory/elearning-software/openolat (дата звернення 29.04.2021). – Назва з екрана.

12) Install the angular CLI [Електронний ресурс] – Режим доступу: https://angular.io/guide/setup-local#install-the-angular-cli (дата звернення 29.04.2021). – Назва з екрана.

13) Ng-book. The complete book on Angular 8 / Nate Murray, Felipe Coury, Ari Lerner, and Carlos Taborda, 2018 – 39-40 с.

14) Angular 2+ Notes for Professionals / Goalkicker.com, 2018 – 55 с.

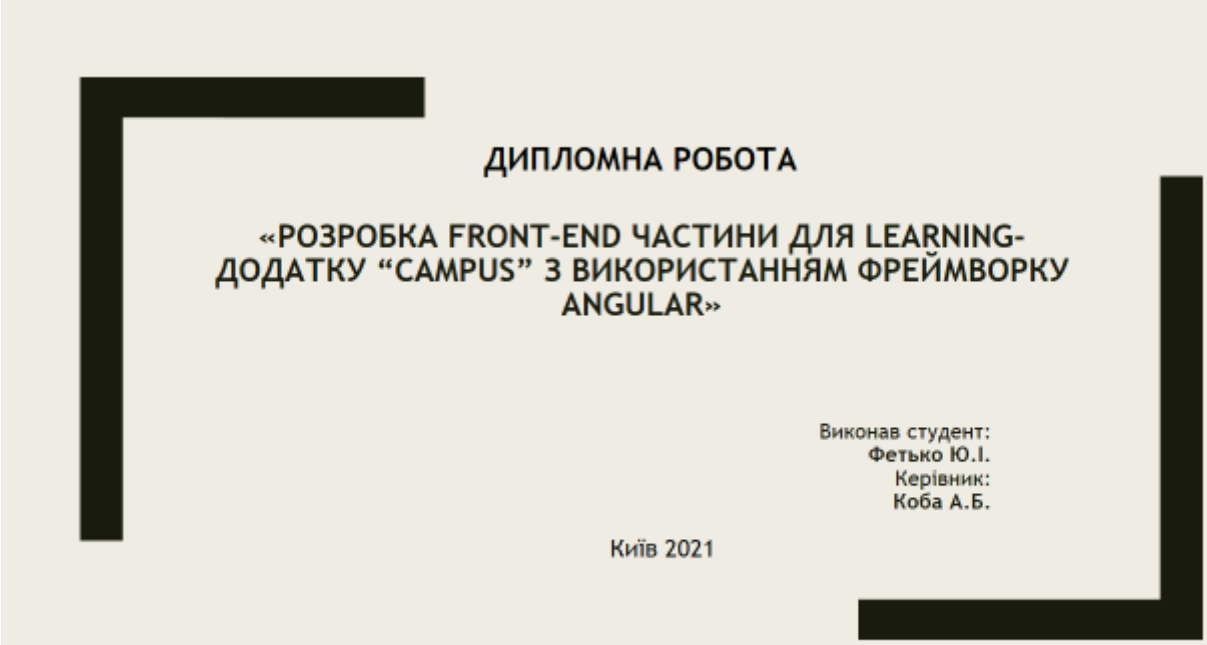
15) Pro Angular 6 Third Edition / Adam Freeman, 2018 – 292 с.

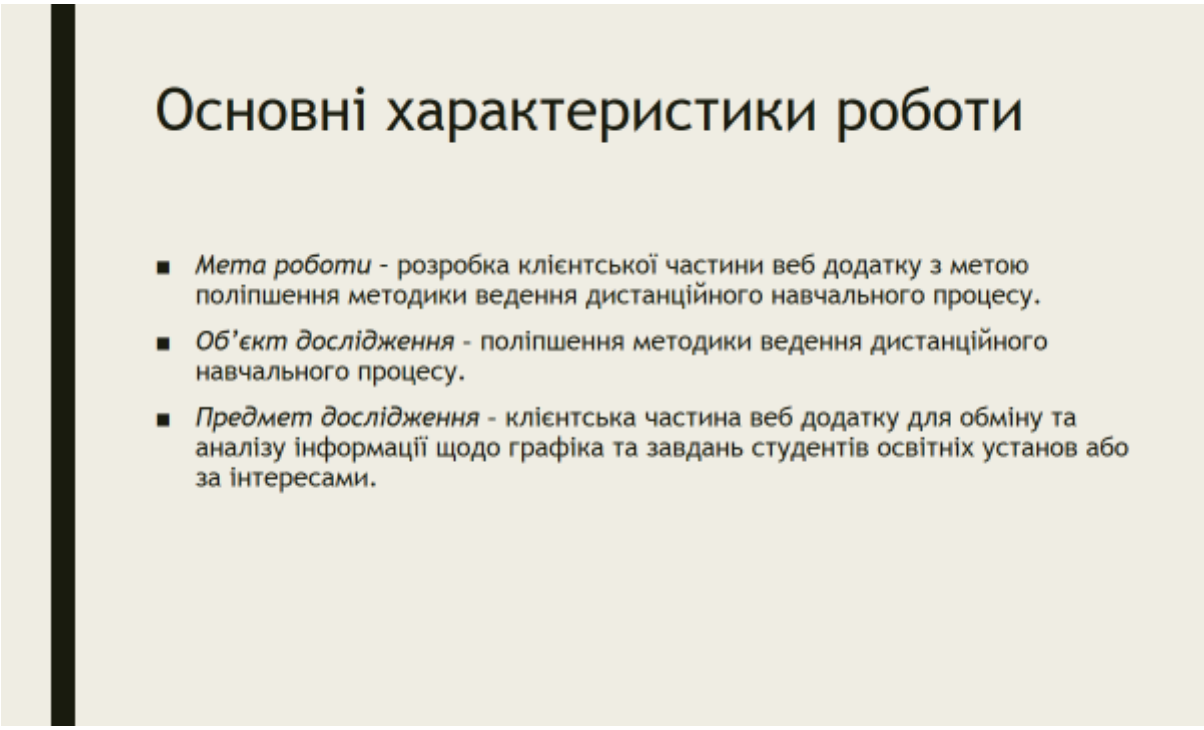
16) Bearer Authentication [Електронний ресурс] – Режим доступу: https://swagger.io/docs/specification/authentication/bearer-authentication/ (дата звернення 29.04.2021). – Назва з екрана.

17) Unit Testing. Principles, Practices, and Patterns / Vladimir Khorilov, 2020 – 20-21 с.

18) Unit Testing Angular Services with HttpTestingController [Електронний ресурс] – Режим доступу: https://www.techiediaries.com/angular-testing-httptestingcontroller (дата звернення 29.04.2021). – Назва з екрана.

ДОДАТОК А


ДИПЛОМНА РОБОТА
«РОЗРОБКА FRONT-END ЧАСТИНИ ДЛЯ LEARNING-ДОДАТКУ “CAMPUS” З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ ANGULAR»
Виконав студент:
Фетько Ю.І.
Керівник:
Коба А.Б.
Київ 2021


Основні характеристики роботи
Мета роботи - розробка клієнтської частини веб додатку з метою поліпшення методики ведення дистанційного навчального процесу.
Об’єкт дослідження - поліпшення методики ведення дистанційного навчального процесу.
Предмет дослідження - клієнтська частина веб додатку для обміну та аналізу інформації щодо графіка та завдань студентів освітніх установ або за інтересами.

Актуальність

- Дистанційна освіта стає надзвичайно популярною формою навчання в силу своєї зручності й гнучкості. Вона усуває основний бар'єр, що утримує багатьох людей від отримання освіти, позбавляючи від необхідності відвідувати заняття за встановленим розкладом.
- Для того, щоб забезпечити ефективну взаємодію, при дистанційному навчанні використовується цілий набір інструментів, включаючи інтерактивні комп'ютерні програми, Інтернет, електронну пошту, телефон та інші.

Аналоги у світі

Features	Campus	Google Classroom	Canvas LMS	Moodle
система push повідомлень	+		+	+
запуск програмного коду у браузері	+			
розподіл дисциплін по віртуальним кімнатам	+	+	+	+
авто-присвоювання оцінки	+		+	
календар з подіями	+	+		+

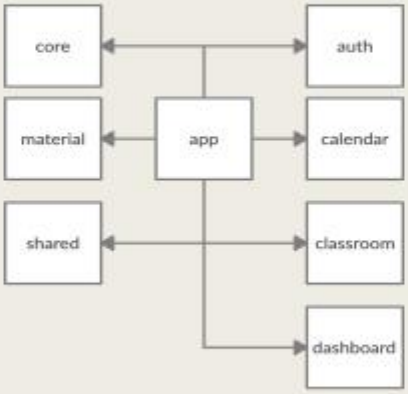
Принцип розробки

	SPA	Традиційний веб-сайт
Швидкий і плавний UX	+	
SEO оптимізація		+
Мінімальне серверне навантаження	+	
Офлайн функціональність	+	
Мобільна адаптація	+	
Швидкість	+	
Швидкість першого завантаження сторінки		+
Робота без JS		+

Послідовники SPA. Порівняння

Особливість	**Angular**	**React**	**Vue**
UI/DOM управління	+	+	+
Управління станом даних	+	+	+
Роутинг	+	-	+
Валідація форм	+	-	-
Http клієнт	+	-	-

Архітектура



Навігація

Вхід та реєстрація

```
this.form = new FormGroup( controls: {
  email: new FormControl( formState: null,  validatorOrOpts: [Validators.required, Validators.email]),
  password: new FormControl( formState: null,  validatorOrOpts: [
    Validators.required,
    Validators.minLength( minLength: 8),
  ])
});
```

```
<div class="form-control" [ngClass]="{
    invalid: form.get('email').touched && form.get('email').invalid}">
  <label for="email" class="email"></label>
  <input id="email"
         type="email"
         placeholder="{{ 'AUTH.SIGN-IN.FORM-TEXTBOX.EMAIL' | translate }}"
         formControlName="email"/>
  <div *ngIf="form.get('email').touched && form.get('email').invalid" class="validation">
    <small *ngIf="form.get('email').errors.required">
      {{ 'AUTH.SIGN-IN.ERROR-MESSAGES.EMAIL.REQUIRED' | translate }}
    </small>
    <small *ngIf="form.get('email').errors.email">
      {{ 'AUTH.SIGN-IN.ERROR-MESSAGES.EMAIL.PATTERN' | translate }}
    </small>
  </div>
</div>
```

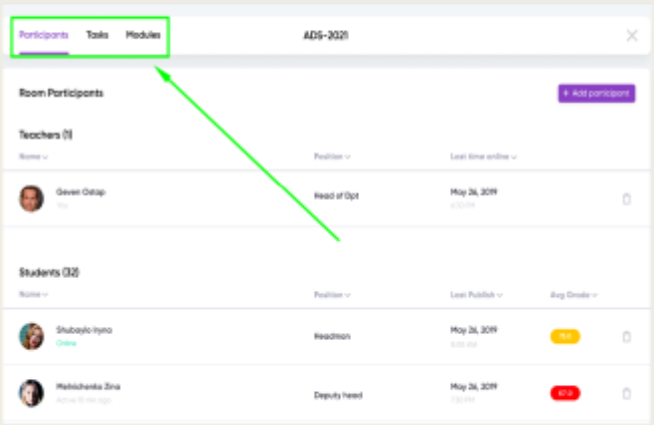
Сервіси

```
@Injectable()
export class SignInService {
  constructor(private http: HttpClient) {
  }

  login(user: AuthenticatedUser): Observable<string> {
    return this.http.post<string>(environment.authenticateProfile, user);
  }

  getProfileInformation(): Observable<ProfileInformation> {
    return this.http.get<ProfileInformation>(environment.getProfileInformation);
  }
}
```

Робота Guard

Кімната класу

```
const routes: Routes = [
  {
    path: '',
    component: ClassroomViewComponent,
    children: [
      {
        path: 'participants',
        component: ClassroomParticipantsComponent
      },
      {
        path: 'modules',
        component: ClassroomModulesComponent
      },
      {
        path: 'grades',
        component: ClassroomGradesComponent
      },
      {
        path: '**',
        redirectTo: 'participants'
      }
    ]
  }
];
```

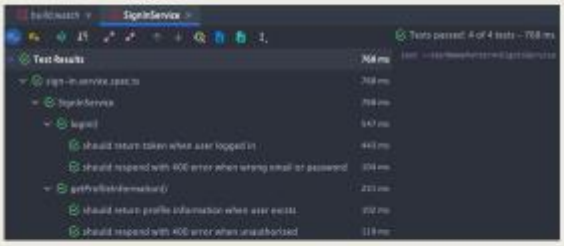


Відтворення кімнат класу

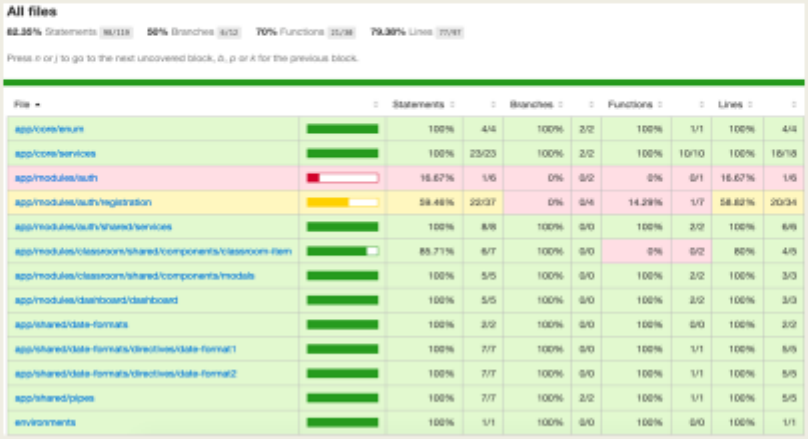
```
<div class="classrooms">
  <ng-container *ngFor="let classroom of classrooms">
    <app-classroom-item [classroomItem]="classroom"></app-classroom-item>
  </ng-container>
</div>
```

Unit-тестування

```
test( name: 'should return token when user logged in',  fn: () => {
  const mockToken: string = 'token';

  const authenticatedUserMock: AuthenticatedUser = {
    email: 'exmple@example.com',
    password: 'Password321'
  };

  signInService.login(authenticatedUserMock).subscribe( next: response =>
    expect(response).toEqual(mockToken));

  const req = httpTestingController.expectOne( url: '/api/profile/auth');
  expect(req.request.method).toMatch( expected: 'POST');

  req.flush(mockToken);
});
```



Покриття тестами



Висновки

В результаті виконання даної роботи, була досягнута ціль - розроблено learning-додаток “Campus”. Також були вирішені такі задачі:

- Аналіз наявних застосунків для дистанційного навчання, наведено їх переваги та недоліки.
- Визначено вимоги та функціональні можливості learning-додатку “Campus”.
- Проведено аналіз програмних засобів для розробки додатку, визначили, що Angular має багато переваг перед своїми конкурентами.
- Проектування та розробка веб-застосунку “Campus”, проведено аналіз частин проекту, та розроблено архітектуру проекту яка є оптимальною та може бути розширена в майбутньому.
- Покриття unit-тестами створеного додатка для підтвердження його коректної працездатності.