СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) факультет інформаційних
(повне найменування інституту, факультету)
технологій та електроніки
Кафедра інформаційних технологій та програмування
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА
до кваліфікаційної випускної роботи

освітній ступінь бакалавр
(бакалавр, магістр)
спеціальність 126 „Інформаційні системи та технології“
(шифр і назва спеціальності)
спеціалізація „Інформаційні системи та технології“
(назва спеціалізації)

на тему „Інформаційно-програмний комплекс розкладу занять вищого навчального закладу“

Виконав: студент групи ІСТ-19д _____ В.І. Борзикін
(підпис) (ініціали і прізвище)

Керівник _____ Д.В. Ратов
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ В.О. Лифар
(підпис) (ініціали і прізвище)

Рецензент_____

Київ - 2023

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІСТ-19д Борзикін В.І.

Науковий керівник

Доцент, к.т.н. _____ Ратов Д.В.

Оцінка наукового керівника: _____

Рецензент _____

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

_____ Лифар В.О.

підпис

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) факультет інформаційних
(повне найменування інституту, факультету)
технологій та електроніки
Кафедра інформаційних технологій та програмування
(повна назва кафедри)
Освітній ступінь бакалавр
(бакалавр, магістр)
спеціальність 126 „Інформаційні системи та технології“
(шифр і назва спеціальності)
спеціалізація „Інформаційні системи та технології“
(назва спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТП

“____” _____2023року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Борзикін Вадим Ігорович
(прізвище, ім’я, по батькові)

1. Тема роботи Інформаційно-програмний комплекс розкладу занять вищого навчального закладу

Керівник роботи Ратов Денис Валентинович, к.т.н доцент,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “ 26 ” 04 2023року № 240/15-ОД

2. Строк подання студентом роботи 19 червня 2023
3. Вихідні дані до роботи Об’єктом даної роботи є процес створення інформаційно-програмного комплексу

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області, вибір та обґрунтування програмних засобів, створення архітектури додатка, розробка додатка, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 24.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	24.03.2023	
2	Укладання і погодження з керівником плану і етапів виконання роботи	02.04.2023	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	8.04.2023	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	9.04.2023	
5	Розробка та тестування програмного продукту	21.05.2023	
6	Укладання, оформлення та погодження пояснювальної записки з керівником	24.05.2023	
7	Здача готової пояснювальної записки на кафедру	27.05.2023	
8	Укладання доповіді і презентації	30.05.2023	

Студент _____ В.І. Борзикін
(підпис) (ініціали і прізвище)

Керівник роботи _____ Д.В. Ратов
(підпис) (ініціали і прізвище)

РЕФЕРЕАТ

Робота містить: 45 сторінок основного тексту, 11 сторінку додатків, 8 рисунків, 0 таблиць, 14 використаних джерел.

Метою випускної кваліфікаційної роботи є розгляд питань розробки кросплатформних додатків та створення оптимальної архітектури побудови інформаційної системи.

Було проаналізовано різні стеки побудови кросплатформенних додатків, їх переваги та відмінності, обрано стек React Native/Expo.

В результаті виконаної роботи, було побудовано інформаційну систему у вигляді кроссплатформенного застосунку.

Система реалізована відповідно до всіх вимог, запропонованим у технічному завданні. Зроблений детальний огляд процесу розробки та продемонстровано результат розробки програмного засобу.

ЗМІСТ

ВСТУП

Актуальність. Мобільні пристрої нині стали невід'ємною складовою нашого повсякденного життя, і без них сучасна людина вже не може собі уявити комфорт. Телефони втратили свою первісну функцію простого засобу зв'язку і перетворилися на справжні інформаційні комбайни та головне джерело інформації для людей в сучасному світі.

Враховуючи все більшу популярність мобільних пристроїв і зростання їх функціональних можливостей, мобільні додатки стають необхідними для задоволення потреб користувачів. Вони дозволяють підприємствам, організаціям і індивідуальним розробникам ефективно взаємодіяти зі своєю аудиторією, надавати послуги та розвивати свої внутрішні-процеси.

Навчальні процеси у закладі вищої освіти є не менш важливими за інші інформаційні процеси, тому важливо надавати користувачу мобільного телефона легкий доступ до інформації розкладу вищого навчального закладу у вигляді кросплатформного додатку.

Об’єкт дослідження: Процес розробки інформаційно-програмного комплексу розкладу вищого навчального закладу.

Предмет дослідження: Кросплатформний додаток розкладу вищого навчального закладу.

Мета дослідження: розробка кросплатформного додатку розкладу вищого навчального закладу.

Задачі дослідження:

1. Проаналізувати наявні знання та попередній досвід розробки з теми побудови інформаційних систем

2. Обрати стек технологій для розробки та цільову платформу проекту.

3. Створити список завдань та підзавдань для досягнення поставленої мети

4. Розробити додаток згідно з поставленими завданнями

1 АНАЛІТИЧНИЙ ОГЛЯД

В сучасному цифровому світі, де смартфони стали невід'ємною частиною нашого повсякденного життя, мобільні додатки здобувають все більшу популярність. Особливо в університетському середовищі, де ефективне управління розкладом занять є критично важливим для студентів і викладачів. У зв'язку з цим, написання мобільного додатку для розкладу занять в університеті стає актуальною задачею.

Мотивація та необхідність мобільного додатку:

Зручність і доступність: Мобільні додатки дозволяють студентам та викладачам швидко та зручно переглядати розклад занять без необхідності використання комп'ютера або перегляду паперового розкладу. Це надає зручність і гнучкість для користувачів, що забезпечує більш ефективне управління часом та ресурсами.

Швидкість та актуальність інформації: За допомогою мобільного додатку, розклад занять може бути оновлений в режимі реального часу. Це особливо важливо у випадку змін у розкладі або інформації про аудиторії. Студенти та викладачі можуть швидко отримати оновлену інформацію без необхідності перевіряти електронну пошту або стежити за оголошеннями.

Написання кросплатформних додатків вимагає серйозного підхожу до планування майбутніх функцій та можливостей програми, бо кожна із обраних цільових платформ, як теоритично, так і на практиці може накладати свої обмеження обумовлені як особливостями архітектури цільової платформи, так і додатковими обмеженнями висунутими компанією-розробником та/або компанією-власником цієї платформи.

Наприклад, створення програм для платформи Android є складним технологічним процесом, який вимагає детального планування та має свої особливості. Однією з них є різноманітність мобільних пристроїв, на яких додаток буде запускатись. При розробці необхідно враховувати, що користувачі мають різні технічні характеристики своїх пристроїв, а також використовують різні версії операційної системи Android.

Також обмеженням для додатків на платформі Android може бути підвищені ризики безпеки на пристроях, які за відома користувача пристрою, або без його відома, мають розблокований завантажувач та доступ до прав Суперкористувача. Це створює обмеження для створення додатків з особливо чутливими даними. Наприклад додатки з роботою з грошима та з чутливими даними користувача потрібно буде будувати із методами забезпечення максимального підвищення безпеки, які були б надмірними у разі використання додатка на «безпечному» варіанті ОС Android та на інших цільових платформах, де подібних проблем безпеки немає за замовчуванням, через іншу архітектуру.

В операційних системах Unix і Unix-подібних, таких як Linux і Android, звичайному користувачеві присвоюється роль Гостя, що дозволяє виконувати звичайні дії, такі як програвання музики, перегляд відео, гра. Однак, Гість не має можливості видаляти або змінювати системні файли або розширювати функціонал операційної системи. Ці дії доступні лише Адміністратору, який також називається Суперкористувачем. Root (або Суперкористувач) є особливим обліковим записом, володар якого має повні права на здійснення будь-яких операцій.

Платформи від компанії Apple навпаки відомі своєю підвищеною безпекою, яка досягається введеними правилами розробки, розміщення, та встановлення додатків на ці платформи. Також на безпеку працює загальна

закритість сімейства операційних систем та обмеженість моделей пристроїв, на який вони використовуються. Кожен додаток, що подається в App Store, проходить обов'язкову перевірку перед тим, як стати доступним для користувачів. Модератори ретельно переглядають додатки, щоб переконатися, що вони відповідають встановленим критеріям якості, безпеки та захисту даних користувачів. Недоліки або порушення правил можуть призвести до відхилення додатка та вимоги виправити проблеми перед подачею повторно. Цей процес перевірки спрямований на забезпечення якості та безпеки додатків, що пропонуються користувачам App Store.

З точки зору користувача все це однозначно гарно, а для розробників це як плюс, бо не треба додатково перейматися через питання безпеки даних, так і мінус, бо магазин додатків надає ряд жорстких вимог. Навіть створення тестових збірок додатків не проходить без чіткого контролю компанії-розробника платформи: для кожної збірки потрібен акаунт розробника, під час створення якого потрібно вказати велику кількість даних та пройти «грошовий ценз» у вигляді оплати річного аккаунту розробника за 99$.

Тому майбутній додаток розкладу занять вищого навчального закладу, буде являти собою кросплатформний мобільний додаток, що дозволить користувачам використовувати його на будя-якій операційній системі (ОС), незалежно від місця та часу. Зараз найбільш популярними мобільними ОС є Android від компанії Google та iOS від Apple. Підтримка цих платформ може буди досягнута за допомогою використання стека технологій на базі фреймворку React Native, розробленого компанією Facebook для внутрішнього використання і згодом випущеного публічно. Оскільки Android є лідером серед мобільних ОС за поширеністю, він був обраний як головна платформа для дослідження.

Основні завдання розробника кросплатформиних додатків включають:

1. Створення ТЗ (технічного завдання) на розробку додатку.
2. Обговорення з замовником етапів і процесу роботи над проектом. Пункт обов’язковий лише тоді, коли сам замовник не є розробником.
3. Побудова архітектури додатку.
4. Програмування основної логіки додатку додатку.
5. Співпраця з дизайнерами або самостійне ствренння дизайну для розробки інтерфейсу
6. Підтримка та доробка додатків.
7. Співпраця з тестувальниками або самостійне тестування для налагодження додатку.
8. Допомога у створенні інструкцій по роботі з готовим додатком за бажанням замовника.
9. Оформлення документації.
10. Розміщення додатків в магазинах мобільних додатків, таких як Apple App Store, Google Play Market, Amazon Appstore та у Microsoft Store/iTunes у випадку розробки десктопного додатку або у власних каналах розповсюдження

1.1 Протоколи

Протоколи відіграють важливу роль у передачі інформації між комп'ютерами та пристроями. Один з них - HTTP, дозволяє кожному сайту бути доступним у браузері, запитуючи веб-сторінку з сервера і отримуючи відповідь з HTML, CSS і JavaScript. Інший протокол - DDP, використовує веб-сокети для забезпечення спільної зв'язку між клієнтом і сервером. Це дозволяє оновлювати сайт в режимі реального часу без необхідності оновлювати весь

браузер. DDP, як правило, використовується для API і має стандартні методи, такі як GET, POST і PUT, які дозволяють обмінюватися інформацією між додатками.

Для цілей, поставлених в рамках технічного завдання для майбутнього додатку вирішено використовувати HTTP-запити на сервер для отримання даних.

1.2 API

API (інтерфейс прикладного програмування) дозволяє іншим розробникам використовувати функціональні можливості програми без заглиблення у внутрішній код системи, що забезпечує обробку даних. Користувачі API отримують доступ до певних кінцевих точок, які надають розробники, і цей доступ контролюється за допомогою ключа API. Також додаткову безпеку забезпечують такі технології як CORS (Cross-origin resource sharing).

Cross-Origin Resource Sharing (CORS) - це механізм безпеки в сучасних клієнтах та серверах, який дозволяє веб-додаткам та звичайним додаткам отримувати доступ до ресурсів з інших доменів або IP-адрес. Правильне налаштування механізму CORS може забезпечити необхідний рівень безпеки. Для реалізації цього механізму, сервер з якого завантажуються дані додає в HTTP-заголовок окремий тег, в якому перераховано домени, з яких додаток може отримувати дані.

HTTP-клієнт додатка, який отримав таку відповідь від сервера, дозволить додатку запитувати та отримувати дані тільки від двох серверів:

- У випадку веб-додатку, з якого веб-додаток був завантажений та з тих, що були вказані в тегу відповіді.
- У випадку окремого додатку, лише з тих доменів, які є у тегу.

1.3 Архітектурний шаблон побудови мережевих додатків

Серед переліку архітектур було обрано класичний варіант клієнт-сервер

Така архітектура ідеально підходить до ситуацій, коли є загальні ресурси та сервіси, до яких потрібно забезпечити доступ великої кількості розподілених клієнтів.

В підході «клієнт-сервер» компоненти взаємодіють між собою за такими правилами:

1. Клієнти ініціюють запити до сервера, відправляючи їм відповідні запити.
2. Сервер отримує запити від клієнтів і обробляє їх.
3. Сервер генерує відповідь на запит і надсилає її клієнту.
4. Клієнт отримує відповідь від сервера і обробляє її для подальшого використання.

Цей підхід забезпечує взаємодію між клієнтами і серверами, де клієнти виконують роль ініціаторів запитів, а сервери відповідають на ці запити, надаючи необхідні ресурси чи інформацію.

Також сервер може бути обмеженою точкою щодо продуктивності, оскільки він обробляє запити від багатьох клієнтів одночасно. Якщо сервер не може витримати велику кількість запитів або обробити їх швидко, це може призвести до зниження продуктивності системи.

Крім того, сервер є єдиною точкою відмови, оскільки він відповідає за обробку запитів і надання відповідей. Якщо сервер вийде з ладу або не зможе відповісти на запити, то це може призвести до недоступності системи для користувачів.

Однак, змінювати прийняте рішення про розміщення функціональних можливостей (на клієнті або на сервері) після створення системи може бути складно і коштовно. Тому важливо правильно спланувати розподіл функціональності від початку розробки, з урахуванням прогнозованого обсягу роботи, продуктивності сервера і потреб користувачів.

Для забезпечення доступу та контролю якості обслуговування для великої кількості розподілених клієнтів можна використовувати такі загальні ресурси та сервіси:

1. CDN (Content Delivery Network): Це мережа серверів, розташованих у різних географічних місцях, яка дозволяє ефективно доставляти контент клієнтам. CDN забезпечує швидку доставку великого обсягу даних, знижує завантаження на сервер та покращує продуктивність.
2. Балансування навантаження (Load Balancing): Це техніка, коли трафік розподіляється між різними серверами або вузлами, щоб забезпечити оптимальне навантаження та високу доступність. Це дозволяє уникнути перевантаження серверів та забезпечити швидку обробку запитів.
3. Системи кешування (Caching): Кешування дозволяє зберігати копії часто запитуваних даних на проміжному рівні, що знижує час доступу та забезпечує швидкий доступ до ресурсів. Це особливо корисно для статичного контенту, який рідко змінюється.
4. Системи керування доступом (Access Control Systems): Ці системи дозволяють контролювати доступ до ресурсів та сервісів, встановлювати права доступу для різних користувачів або груп користувачів. Наприклад, можна

використовувати систему авторизації та аутентифікації, щоб забезпечити доступ лише авторизованим користувачам.

1.4 Формати даних

Формат даних - це визначений спосіб представлення та організації даних. Він визначає структуру, синтаксис та правила, за якими дані повинні бути збережені, передані та інтерпретовані комп'ютерними системами.

У світі програмування і обробки даних існує багато різних форматів даних, таких як текстові файли (наприклад, CSV, JSON, XML), бінарні файли, бази даних (наприклад, SQL), спеціалізовані формати (наприклад, документи Microsoft Word або таблиці Excel) і багато інших.

Кожен формат даних має свою внутрішню структуру та правила, які визначають, як дані повинні бути організовані і як їх можна читати та обробляти програмними засобами. Вибір відповідного формату даних залежить від потреб і вимог конкретного застосування або середовища.

XML - текстовий формат для представлення структурованих даних, який дозволяє створювати структуровані документи, в яких дані представлені у вигляді тегів та значень. Кожен елемент даних має відповідний тег, і його значення може бути вкладеним у тег або в атрибут. XML документ може також містити коментарі, інструкції обробки та інші метадані. Одна з ключових особливостей XML полягає у його розширюваності та гнучкості. XML є платформо-незалежним форматом даних, що дозволяє ефективно передавати та обробляти різноманітні дані між різними системами.

YAML - текстовий формат структурованих даних. Одна з особливостей YAML - це його читабельність людиною. Він не є мовою розмітки, але

набагато простіший у використанні, ніж інші формати, такі як XML або JSON. YAML дозволяє використовувати простий інтуїтивний синтаксис для представлення даних.

YAML підтримує можливість коментування даних та включення посилань на інші частини документа, що дозволяє уникнути дублювання даних та полегшує підтримку складних структур даних.

Окрім цього, багато реалізацій YAML також підтримують синтаксис JSON як синтаксичну підмножину YAML. Це означає, що валідний JSON також є валідним YAML. JSON-сумісний синтаксис YAML дозволяє легко перетворювати дані між цими двома форматами та використовувати переваги обох.

JSON - це легкий та популярний структурований текстовий формат даних. JSON базується на синтаксисі JavaScript, але він є незалежним від мови програмування і підтримується багатьма мовами програмування. Дані в JSON форматі представляються у вигляді тексту, який може бути легко зрозумілим для людини та легко оброблюється комп'ютерами.

Із зазначених форматів було обрано JSON через контекст використання та вимоги проекту.

1.5 Технічне завдання

Результатом виконання даного дипломного проекту повинена стати розробка кросплатформного додатку розкладу занять вищого навчального закладу. Додаток повинен бути зручним, простим у використанні для користувача та інтуїтивно зрозумілим. Додаток призначається для користування як студентами, так і викладачами.

При розробці додатка додатка потрібно виконати такі задачі:

1. Зробити аналіз сучасних методів кросплатформної розробки
2. Зробити аналіз середи розробки
3. Зробити аналіз архітектур навігації в додатках
4. Написати коду додатку
5. Провести тести додатку

2 МОДЕЛЬ ПРОЕКТУ. ТЕХНОЛОГІЇ ТА АРХІТЕКТУРА

Кросплатформність (багатоплатформність, мультиплатформність) – це властивість програмного забезпечення, що дозволяє працювати на різних програмних та апаратних платформах, таких як операційні системи. Технології кросплатформної розробки надають засоби для створення програм, які можуть бути запущені на різних платформах без необхідності переписування коду для кожної окремої платформи.

Є кілька причин, які спричиняють виникнення явища кросплатформної розробки:

- **Різноманітність платформ**: На ринку існує багато різних мобільних та веб-платформ, таких як Android, iOS, Windows, macOS, Linux і багато інших. Кожна платформа має свої вимоги та специфіку. Кросплатформна розробка дозволяє зменшити зусилля, необхідні для створення окремих версій додатків для кожної платформи.
- **Економія часу і ресурсів**: Розробка окремих версій додатків для кожної платформи може бути часо- та ресурсозатратною задачею. Кросплатформна розробка дозволяє використовувати загальний код, що зменшує зусилля та витрати на розробку, тестування та підтримку.
- **Масштабованість та поширення**: Кросплатформні додатки можуть бути легко масштабовані та поширювані на різні платформи. Це дозволяє підприємствам та розробникам швидше досягати більшої аудиторії та забезпечувати доступ до своїх продуктів на різних пристроях.
- **Зручність розробки та обслуговування**: Використання кросплатформних фреймворків та інструментів спрощує процес розробки, тестування та обслуговування додатків. Розробники можуть використовувати

відомі мови програмування та інструменти, а також поділити загальний код між різними платформами.

- **Швидкість випуску на ринок**: Кросплатформна розробка дозволяє швидше випускати продукти на ринок, оскільки розробники можуть працювати над всіма платформами одночасно. Це може бути важливо в умовах швидкозмінного ринку та конкуренції.

2.1 Сучасні методи кросплатформної розробки

Сучасні методи кросплатформної розробки дозволяють розробляти додатки, які працюють на різних платформах з використанням загального коду. Ось кілька популярних методів кросплатформної розробки:

1. **Фреймворки гібридної розробки**: Фреймворки, такі як Apache Cordova (PhoneGap), React Native, Xamarin та Flutter, дозволяють розробляти додатки, які запускаються веб-переглядачі або використовують нативні компоненти платформи. Ці фреймворки використовують загальну базу коду, що дозволяє ефективно розробляти для різних платформ.
2. **Прогресивні веб-додатки (PWA)**: PWA - це веб-додатки, які використовують сучасні веб-технології для створення досвіду, схожого на нативний додаток. Вони можуть працювати в автономному режимі, отримувати сповіщення та мати доступ до пристройових функцій. PWA дозволяють розробляти додатки один раз і запускати їх на різних платформах, використовуючи веб-браузери. Технологія PWA не універсальна і має низку недоліків: не всі пристої та ОС підтримують повний набір функцій PWA, обмежена робота додатку офлайн, низка обмежень на iOS.
3. **Розробка з використанням нативних мов програмування**: Деякі мови програмування, такі як Kotlin, Swift та C#, можуть бути

використані для розробки нативних додатків для різних платформ. З використанням спеціальних фреймворків, таких як Kotlin Multiplatform, можна поділити загальний код між платформами, зменшуючи дублювання роботи.

Вибір підходу до розробки залежить від потреб замовника, технічних вимог та навичок розробника.

2.2 Xamarin

Xamarin є платформою для кросплатформної розробки мобільних додатків. Вона дозволяє розробникам використовувати мову програмування C# та .NET-фреймворк для створення нативних додатків для різних платформ, включаючи Android, iOS та Windows. Xamarin надає зручний інструментарій, що дозволяє перевикористовувати код, включаючи бізнес-логіку та інтерфейс, для розробки додатків, які працюють однаково на різних платформах. Вона забезпечує доступ до нативних API та функціональності платформ, що дозволяє розробникам створювати потужні та ефективні додатки. Xamarin може бути використана в поєднанні з різними інтегрованими середовищами розробки, такими як Visual Studio, а також має активну спільноту розробників, яка підтримується Microsoft.

Xamarin займає близько 30 відсотків від ринку кросплатформних мобільних застосунків

Деякі з недоліків використання Xamarin можуть включати:

1. **Складність налаштування**: Налаштування Xamarin може бути відносно складним, оскільки вимагає налаштування середовища розробки та підключення до різних платформ.

2. **Розмір додатку**: Додатки, розроблені з використанням Xamarin, можуть бути більшими за розмір, оскільки включають в себе додатковий код та бібліотеки для кросплатформної підтримки.

3. **Обмеження доступу до платформеного API**: У деяких випадках можуть бути обмеження доступу до нових функцій або платформеного API, оскільки Xamarin повинен підтримувати кросплатформену сумісність.

4. **Потреба в вивченні не тільки мови C#, а й нативних мов Android та iOS**: Хоча розробники платформи і позиціонують її як кросплатформну, все одно може бути присутня необхідність написання модулів додатку нативною мовою для обраної платформи

2.3 Flutter

Фреймворк Flutter був розроблений компанією Google у 2017 році і швидко набув популярності, конкуруючи з React Native. Flutter використовує мову програмування Dart і дозволяє розробляти мультиплатформні мобільні додатки, які можуть працювати на різних операційних системах, включаючи Android та iOS.

Основною особливістю Flutter є власний движок рендерингу, який дозволяє створювати швидкі та привабливі інтерфейси користувача з використанням віджетів. Фреймворк надає багато готових компонентів і інструментів для розробки інтерфейсу, а також доступ до нативних API пристроїв. Flutter також має гарну продуктивність і здатність до гарячого перезавантаження, що дозволяє швидко бачити зміни у додатку під час розробки.

Фреймворк Flutter зарекомендував себе як популярне рішення для розробки красивих та швидких кросплатформних додатків, і його використовують багато компаній та розробників по всьому світу.

Одні з основних переваг Flutter - це продуктивність та графічні можливості. Він забезпечує швидке відтворення і відзначається високою продуктивністю завдяки своїй архітектурі, яка включає в себе власний рушій для малювання елементів і використовує аппаратний прискорювач для 2D та 3D графіки.

Ці особливості роблять Flutter привабливим вибором для розробки додатків, які вимагають високої якості графіки та плавного інтерфейсу користувача.

Особливості Flutter відносно Apache Cordova та React Native:

1. **Висока продуктивність**: Flutter використовує власний двигун рендерингу Skia, що дозволяє досягти високої продуктивності та швидкості роботи додатків. У порівнянні з Apache Cordova, який використовує WebView, та React Native, який використовує місткість між JavaScript та нативним кодом, Flutter може працювати швидше та більш ефективно.
2. **Повна кросплатформовість**: Flutter дозволяє створювати один код для різних платформ, таких як iOS та Android, який буде повністю сумісним між різними платформами. Це означає, що розробники можуть уникнути окремої обробки функцій та властивостей кожної з ОС, що згодом могло б стати складним та затратним завданням у підтримці. У порівнянні з Apache Cordova, який використовує веб-технології, та React Native, який вимагає окремого коду для кожної платформи, Flutter забезпечує більшу ефективність розробки. Але це має і негативні сторони наприклад як сильно обмежений список підтримуваних платформ. В даний момент Flutter підтримує лише мобільні платформи, такі як iOS та Android, тоді як Apache

Cordova та React Native можуть бути використані для створення додатків для більш широкого спектру платформ, включаючи веб та настільні. Також мінусом є відсутність повного доступу до нативного API. Хоча Flutter надає можливість викликати нативний код за допомогою платформозалежних каналів, він все ж не має такої повної і безпосередньої підтримки нативного API, яку можуть забезпечити Apache Cordova та React Native. Це може бути проблемою для додатків, які вимагають специфічної функціональності, що доступна лише через нативний API.

3. **Красивий інтерфейс користувача**: Flutter має вбудовану бібліотеку для створення привабливих та красивих інтерфейсів користувача, яку називають "Material Design". Ця бібліотека надає широкі можливості для створення естетичного та інтуїтивно зрозумілого дизайну. У порівнянні з Apache Cordova, який використовує стандартні веб-елементи, та React Native, який має менш розвинену систему дизайну, Flutter дозволяє створювати більш привабливий інтерфейс користувача. Але це суттєво впливає на розмір додатку. Додатки, розроблені на Flutter, можуть мати більший розмір, особливо для простіших додатків, порівняно з Apache Cordova та React Native.

2.4 Apache Cordova

Apache Cordova, раніше відомий як PhoneGap, є фреймворком для розробки мобільних додатків з використанням веб-технологій, таких як HTML, CSS і JavaScript. Він дозволяє розробляти кросплатформні додатки, які можуть працювати на різних мобільних платформах, таких як Android, iOS і Windows.

Основна ідея Cordova полягає в тому, що весь додаток виконується веб-браузері, який вбудований в нативний контейнер додатка. Це дозволяє

використовувати веб-технології для створення інтерфейсу користувача та роботи зі стандартними функціями пристрою, такими як камера, геолокація, контакти і т.д.

Особливості Apache Cordova в порівнянні з Flutter та React Native:

1. **Використання веб-технологій**: Apache Cordova використовує HTML, CSS та JavaScript для створення інтерфейсу та логіки додатків. Це означає, що розробники з веб-досвідом можуть легко перейти до розробки мобільних додатків без необхідності вивчення нових мов програмування. Недоліком цього підходу є низька продуктивність. Так як Apache Cordova використовує WebView для відображення інтерфейсу додатка, це може призвести до зменшення продуктивності порівняно з Flutter та React Native, які використовують власні двигуни рендерингу нативних компонентів.
2. **Більша спільнота розробників**: Apache Cordova має велику та активну спільноту розробників, що забезпечує доступ до багатьох різноманітних плагінів, розширень та інструментів. Це полегшує розробку та розширення функціональності додатків. Мінусом є висока залежність від плагінів для доступу до різноманітних функцій та можливостей платформи, які додатково потребують установки та керування. Це може призвести до складнішої конфігурації та управління залежностями проекту.
3. **Доступ до нативного API**: Apache Cordova надає простий доступ до нативного API платформи, що дозволяє розробникам використовувати специфічні функції та можливості, які надаються платформою. Це особливо важливо для додатків, які потребують глибокої інтеграції з операційною системою. Мінусом цього є те, що при поглибленому використанні нативних компонентів обмежуються можливості кросплатформної розробки та може призвести до обмежень та різниці в зовнішньому вигляді та функціональності додатків на різних платформах.

4. **Легка інтеграція з веб-сервісами**: Завдяки використанню веб-технологій, Apache Cordova дозволяє легко інтегрувати додатки з різноманітними веб-сервісами, API та іншими онлайн-ресурсами.

5. **Відсутність готового набору UI-елементів**: Apache Cordova не надає готового набору інтерфейсних елементів, як це роблять Flutter та React Native. Це означає, що розробники повинні самостійно стилізувати інтерфейс та використовувати стандартні веб-елементи для побудови інтерфейсу додатка.

6. **Відсутність гарячого перезавантаження**: Apache Cordova не підтримує гаряче перезавантаження, що є корисною функцією для швидкої перевірки та впровадження змін під час розробки. Це може призвести до більшого часу витраченого на збірку та розгортання змін.

2.5 Ionic

Ionic - це відкритий фреймворк для розробки кросплатформних мобільних додатків. Він базується на веб-технологіях, таких як HTML, CSS та JavaScript, і дозволяє створювати додатки, які працюють як на iOS, так і на Android.

Основною перевагою Ionic є його набір готових компонентів і інструментів для побудови користувацького інтерфейсу додатків. Він надає доступ до стандартних елементів уявлення, таких як кнопки, списки, форми і т.д., що робить розробку інтерфейсу простою і швидкою.

Ionic також поєднується з Cordova або Capacitor, що дозволяє отримати доступ до нативних API пристроїв, таких як камера, геолокація, пуш-сповіщення та багато інших. Це дозволяє створювати додатки з більш широкими функціональними можливостями.

Ionic Capacitor є одним з компонентів фреймворку Ionic, який використовується для розробки кросплатформних мобільних додатків. Він є альтернативою Cordova і надає розробникам більш сучасну та гнучку платформу для побудови нативних функцій додатків.

Переваги Capacitor над Cordova:

1. Архітектура: Ionic Capacitor використовує більш сучасну і гнучку архітектуру, яка дозволяє розробникам використовувати нативні SDK та інструменти для кожної платформи окремо. Apache Cordova використовує підхід, де веб-додаток запускається всередині веб-браузера, що обмежує доступ до деяких нативних можливостей.

2. Підтримка платформ: Capacitor підтримує широкий спектр платформ, включаючи Android, iOS, Electron та Progressive Web Apps (PWA). Cordova також підтримує багато платформ, але можливості можуть бути обмежені або не так потужні, як у Capacitor.

Недоліки використання Ionic Capacitor в порівнянні з React Native та Flutter:

1. Обмежена функціональність: Ionic Capacitor може мати обмежені можливості порівняно з React Native та Flutter, особливо в тих випадках, коли потрібно використовувати специфічні функції або API платформи. React Native та Flutter надають більш прямий доступ до нативного API та можуть мати більший набір функцій.

2. **Швидкодія**: У порівнянні з React Native та Flutter, Ionic Capacitor може мати меншу продуктивність та швидкодію. Це пов'язано з тим, що Ionic Capacitor використовує WebView для відображення інтерфейсу, що може призвести до меншої продуктивності порівняно з використанням власних двигунів рендерингу.

3. **Розмір додатку**: Завдяки використанню WebView, Ionic Capacitor може мати більший розмір додатку порівняно з React Native та Flutter, які

можуть бути компактніші завдяки власним двигунам рендерингу та компіляції.

4. **Спільнота розробників**: Хоча Ionic має певну спільноту розробників, вона може бути меншою та менш активною порівняно з спільнотами React Native та Flutter. Це може призвести до обмеженого доступу до допомоги, розширень та сторонніх бібліотек.

5. **Дизайн та інтерфейс користувача**: Хоча Ionic надає кросплатформений дизайн за допомогою Material Design та iOS-стилю, він може виглядати менш нативним порівняно з React Native та Flutter, які надають більшу контроль та можливості для створення природного інтерфейсу користувача.

2.6 React Native

React Native - це відкрита платформа для розробки кросплатформових мобільних додатків, розроблена компанією Facebook. Вона базується на React, популярній бібліотеці для створення веб-інтерфейсів. Основні тези про React Native такі:

1. **Кросплатформовість**: React Native дозволяє розробляти мобільні додатки для iOS та Android з використанням одного коду. Це дозволяє зменшити час та зусилля, потрібні для розробки та підтримки додатків на різних платформах.

2. **Використання JavaScript**: React Native використовує JavaScript для розробки додатків. Це робить його доступним для веб-розробників, які вже мають досвід у JavaScript. Знання React також полегшує процес вивчення React Native.

3. **Нативний інтерфейс**: Завдяки використанню компонентів, які маплються на нативні елементи інтерфейсу платформи, додатки, розроблені з

використанням React Native, мають нативний вигляд та поведінку. Це допомагає створювати високоякісні додатки, що інтегруються з платформовими можливостями.

4. **Швидкість розробки**: React Native пропонує широкий вибір готових компонентів, бібліотек та інструментів, які сприяють прискоренню розробки. Гаряче перезавантаження (Hot Reload) дозволяє миттєво бачити зміни в додатку без перезапуску, що полегшує та прискорює ітеративний процес розробки.

5. **Велика спільнота розробників**: React Native має активну спільноту розробників, яка надає багато ресурсів, плагінів, сторонніх бібліотек та підтримки. Це робить його легким для вивчення, а також забезпечує доступ до різноманітних рішень та інструментів для розробки.

В цілому, React Native є потужним інструментом для розробки кросплатформових мобільних додатків, забезпечуючи швидкість розробки, нативний вигляд та велику спільноту розробників.

Подібно до React для вебу, в програмах React Native використовується комбінація JavaScript і XML-подібна розмітка, яка відома під назвою JSX, що дозволяє описувати інтерфейси користувача. В рантаймі React Native використовує рідні API для відображення компонентів у мовах Objective-C (для iOS) або Java (для Android). Це означає, що програми використовують реальні компоненти мобільного інтерфейсу, а не веб-сторінки, і мають нативний вигляд та поведінку. Крім того, React Native надає JavaScript-інтерфейси для платформових API, що дозволяє програмам отримувати доступ до функціональності платформи, такої як камера або геолокація користувача.

React Native використовує подібні принципи роботи, що й React, але з певними відмінностями. У відміну від React, який використовує Virtual DOM, React Native не працює з DOM. Замість цього, він працює безпосередньо на кінцевому пристрої у фоновому режимі, де виконується інтерпретація

JavaScript, написаного розробниками. React Native взаємодіє з рідною платформою, передаючи серіалізовані дані через асинхронний та пакетний міст.

React Native підтримує розробку для наступних платформ:

1. **iOS**: Дозволяє створювати мобільні додатки для iPhone та iPad. Офіційно підтримується командою розробки React Native.
2. **Android**: Забезпечує можливість створювати додатки для пристроїв, що працюють на ОС Android. Офіційно підтримується командою розробки React Native.
3. **Windows**: Дозволяє створювати додатки на базі React Native для десктопних та мобільних пристроїв під управлінням ОС Windows. Існує мінімум два варіанти реалізації підтримки цієї платформи: react-native-windows, що офіційно підтримується партнерами команди розробки оригінальної платформи та варіант з використанням фреймворку Electron, що підтримується спільнотою.
4. **macOS**: Дозволяє створювати додатки на базі React Native для десктопних пристроїв під управлінням macOS. Підтримка забезпечується силами партнерів команди розробки React Native.
5. **Web**: За допомогою спеціальних інструментів, таких як React Native Web, можна також створювати веб-версії додатків, які можуть працювати у веб-браузерах.

Завдяки такій широкій підтримці платформ, React Native дозволяє розробляти крос-платформні мобільні додатки, які можуть працювати на різних пристроях та операційних системах, з використанням спільного коду.

Загалом, використання React Native для розробки мобільного додатку розкладу занять вищого навчального закладу є раціональним вибором, оскільки цей фреймворк, в купі з вже наявним досвідом розробки, може забезпечити широкі можливості, продуктивність та кросплатформовість, що

дозволяє створити потужний та зручний інструмент, який полегшить організацію навчального процесу, забезпечить зручність та ефективність взаємодії з розкладом для всіх учасників університетського середовища.

2.7 Фреймворк Expo

Expo – це набір інструментів, побудованих на базі React Native, який спрощує та прискорює розробку кросплатформних мобільних додатків для Android та iOS. Додатково можна зібрати додатки під інші платформи, що підтримуються спільнотою React Native . Expo містить готові компоненти та бібліотеки, що дозволяють швидше та ефективніше розробляти додатки. Крім того, Expo надає набір інструментів, які спрощують процес розробки, допомагають збільшити швидкість розробки та дозволяють зосередитись на створенні функціональності застосунку замість вирішення проблем, пов'язаних з налаштуванням робочого середовища.

Expo складається з трьох основних частин:

- **Expo Go** – це мобільний додаток, який дозволяє запускати та тестувати додатки, створені з використанням Expo. Він доступний для платформ iOS та Android і надає доступ до різноманітних функцій, що пропонує Expo SDK для розробки мобільних додатків. Використання Expo Go дозволяє розробникам переглядати вигляд та функціональність своїх додатків на мобільних пристроях, що сприяє швидкому виявленню та виправленню помилок.
- **Expo SDK** - це набір інструментів та компонентів, який надається разом з фреймворком Expo для розробки мобільних додатків з використанням React Native. Expo SDK розширює можливості React Native, додаючи нові

функції та інтерфейси для взаємодії з різними аспектами мобільних пристроїв, такими як камера, геолокація, пуш-сповіщення, аудіо та багато іншого.

- **Expo CLI** - це прикладна програма командного рядка, що є основним інтерфейсом між розробником та інструментами Expo. Крім того, Expo CLI має веб-інтерфейс, який відкривається у вашому веб-браузері під час запуску проекту. Ви можете використовувати графічний інтерфейс замість командного рядка, якщо вам не зручно працювати з терміналом або якщо ви віддаєте перевагу графічним інтерфейсам. Обидва варіанти мають подібні можливості.

Платформа Expo має і деякі обмеження:

- Додатки Expo не підтримують фонове виконання коду. Це означає, що ви не можете, наприклад, запустити код, який слідкує за змінами місцезнаходження, коли додаток закрито.
- Додатки Expo обмежені нативними API, які підтримує Expo SDK. Це означає, що якщо ваш додаток має дуже специфічні завдання, наприклад, обмін даними з пристроєм Bluetooth, єдиний варіант для реалізації такого функціоналу - це використання простого React Native або написання власного коду з використанням ExpoKit.
- Автономні виконувані файли додатків Expo можуть бути побудовані лише при підключенні до мережі. Expo надає інструмент командного рядка під назвою Expo Application Services (EAS). Це дозволяє розробникам запускати процес збірки проекту на серверах Expo. Після завершення збірки буде доступне посилання URL для завантаження файлу .apk або .ipa. Варто зазначити, що розробники Expo вже працюють над можливістю збирати додаток локально, але ця можливість ще є експериментальною та не рекомендується до використання. Також при такому типу збірки постає питання платформозалежності деяких видів збірок. Наприклад, зібрати файл

.ipa для встановлення на iOS, за умовами компанії Apple, можливо лише на пристроях з ОС macOS.

Додатки, що створюються з використанням фреймворку Expo мають декілька шляхів для тестування:

1. Тестування в мобільному клієнті Expo Go
2. Створення збірок в режимі розробки. Можна розглядати збірку для розробки як власну версію клієнта Expo Go. Якщо для проекту потрібен спеціальний нативний код, плагін конфігурації, спеціальна версія середовища виконання або зменшений розмір пакета програми, можна перейти від використання Expo Go для розробки до окремої збірки додатка для розробки.
3. Створення збірок попереднього перегляду дозволяє команді розробки протестувати додаток перед наступним релізом так, наче це фінальна збірка проекту. Відрізняється від безпосередньо фінальної збірки лише методом розповсюдження, технічних відмінностей мінімум.

2.8 Архітектура додатку

Додаток є комплексним, тобто відповідає шаблону архітектури клієнт-сервер, що є одним із архітектурних шаблонів програмного забезпечення та стала домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними.

Схему клієнт-сервер показано на рисунку 2.1

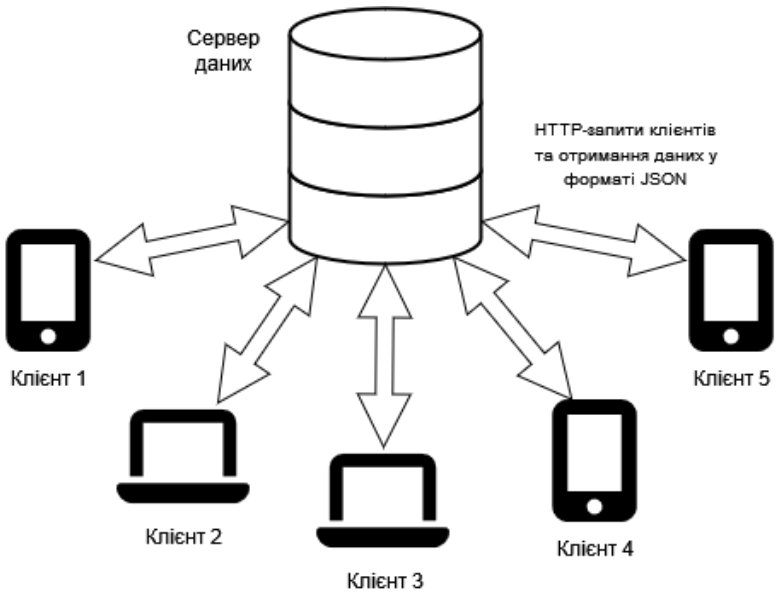

Рис. 2.1

Риси, характерні для серверної частини системи, створеної за архітектурою клієнт-сервер:

- **Управління даними**: Серверна частина додатку розкладу занять університету відповідає за збереження та управління розкладом, інформацією про заняття, групами студентів, викладачами та іншими даними.
- **Обмін даними через API**: Серверна частина повинна надати API (інтерфейс програмування додатків), який дозволяє клієнтській частині додатку отримувати та надсилати дані. Це може включати запити для отримання розкладу занять, оновлення даних про заняття або взаємодію з іншими функціями додатку.
- **Безпека та захист даних**: Централізована серверна частина може запропонувати покращений контроль за безпекою даних, ніж розподілена система, де кожен елемент може мати свої вади безпеки.
- **Масштабованість та продуктивність**: Оскільки додаток розкладу занять університету може мати велику кількість користувачів і обробляти значну кількість даних, серверна частина повинна бути

масштабованою і продуктивною. Це означає, що сервер повинен бути здатний обробляти багато запитів одночасно і швидко відповідати на них, а також мати механізми кешування і оптимізації запитів до бази даних. Масштабованість також включає можливість легко розширювати серверну інфраструктуру, наприклад, додавання додаткових серверів або використання хмарних сервісів для забезпечення надійності та доступності додатку. Це дозволить забезпечити гладку роботу додатку навіть при зростанні навантаження та кількості користувачів.

Риси, характерні для клієнтської частини системи, створеної за архітектурою клієнт-сервер:

- **Інтерфейс користувача**: Клієнтська частина повинна мати зручний інтерфейс, що дозволяє студентам та викладачам легко переглядати розклад занять, вибираючи групу чи викладача через пошук, переглядати деталі занять та зміни. Важливо, щоб інтерфейс був інтуїтивно зрозумілим та забезпечував зручну навігацію по додатку.
- **Локальне збереження налаштувань**: Клієнтська частина може зберігати налаштування та персоналізовані параметри користувача локально на пристрої. Це дозволяє зберігати об’єкт користувача, предпочитані налаштування та інші персоналізовані дані без необхідності постійно їх запитувати з сервера.
- **Взаємодія з сервером**: Клієнтська частина взаємодіє з сервером через API, отримуючи дані про розклад, списки викладачів і груп та іншу інформацію.
- **Мобільні можливості**: Клієнтська частина може використовувати мобільні можливості пристрою, такі як геолокація, камера, сповіщення та інші. Наприклад, студентам можна надавати можливість перегляду розташування

аудиторій на мапі, а викладачам - можливість завантаження матеріалів до кожного заняття.

- **Локальна обробка даних**: Клієнтська частина може виконувати деяку обробку даних локально на пристрої без необхідності постійного звернення до сервера. Наприклад, вона може здійснювати фільтрацію розкладу за певними критеріями.
- **Візуальне оформлення та дизайн**: Клієнтська частина повинна мати приємний та привабливий для використання дизайн.

3 ПОБУДОВА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Середовище розробки

Visual Studio Code (VS Code) - це безкоштовний і відкритий текстовий редактор, розроблений компанією Microsoft. Він підтримує багато мов програмування і надає розширені можливості для редагування коду. У ньому є підсвічування синтаксису, автодоповнення коду, інструменти навігації по коду та інші корисні функції, які полегшують процес розробки.

Крім того, VS Code має вбудовану підтримку системи контролю версій, можливість відладки коду, розширені можливості інтеграції з інструментами розробки та слугує міцним фундаментом для інтеграції розширень, що дозволяє сильно розширити та функціональність та збільшити зручність використання редактора відносно своїх потреб. Список доповнень різноманітний. Є доповнення, що автоматизують рутинні процеси, наприклад автозакриття та автоперейменування тегів HTML або JSX. Є доповнення, які допомагають з форматуванням, наприклад Prettier та ESlint. Є доповнення, які допомагають налаштувати зовнішній вигляд так, щоб нічого не заважало виконувати поставлені завдання.

На рисунку 3.1 показано вигляд налаштованої робочої області Visual Code із встановленими доповненнями

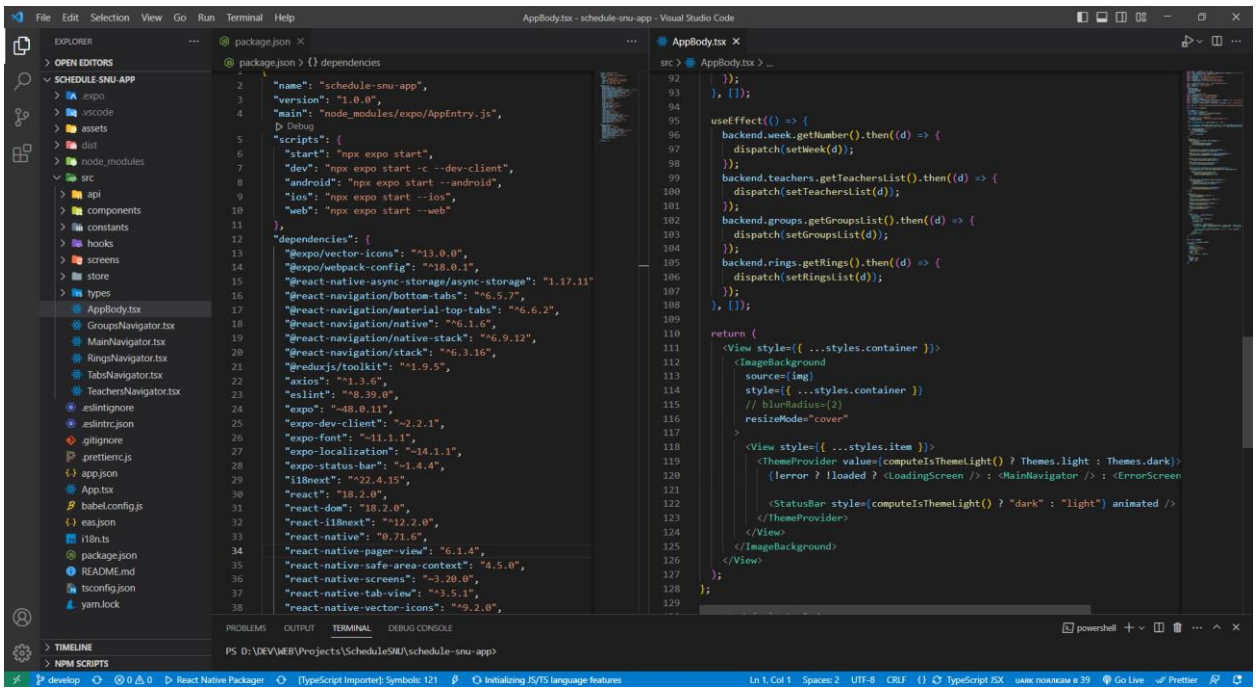

Рисунок 3.1 Вигляд робочої області програми Visual Code

Переваги та особливості VS Code:

- Глибока та всеосяжна база доповнень, яка дозволить створити середовище розробки практично під будь-які завдання.
- VS Code надає вбудовану підтримку багатьох популярних мов програмування, таких як JavaScript або TypeScript, C++, C#, Go Java, Dart/Flutter та багатьох інших. Він пропонує такі функціональності, як автозавершення коду, можливість налагодження коду, підсвічування визначень функцій, управління пакетами та багато іншого.
- VS Code має вбудований віртуальний термінал, що дозволяє виконувати команди прямо в редакторі. Це дозволяє запускати скрипти, виконувати консольні команди та взаємодіяти з іншими інструментами, не виходячи з редактора.
- Кроспратформність: Середа розробки VS Code доступна як на десктопних платформах як наприклад Windows, macOS та різних

дистрибутивах Linux, так і на незвичних платформах: Android або навіть у якості веб-редактору коду, як це зроблено в GitLab.

3.2 Вибір мов програмування для обраного стеку

Оскільки було обрано стек технологій кросплатфрмної розробки на базі сучасних веб-технологій, постає питання про вибір мови програмування із двох найпопулярніших нині наявних наявних: JavaScript, TypeScript.

JavaScript - це високорівнева, інтерпретована мова програмування, яка використовується для розробки веб-додатків та динамічних веб-сторінок. Вона є однією з основних мов програмування для веб-розробки разом з HTML і CSS.

Основні риси JavaScript:

- **Веб-орієнтовані можливості**: JavaScript є основною мовою для програмування веб-сторінок. Вона дозволяє динамічно змінювати вміст сторінки, реагувати на події користувача, взаємодіяти з сервером та маніпулювати DOM (Document Object Model).
- **Інтерпретованість**: JavaScript виконується безпосередньо в браузері без необхідності компіляції. Це дозволяє розробникам швидко експериментувати та вносити зміни в код без додаткового кроку компіляції.
- **Об'єктно-орієнтоване програмування**: JavaScript підтримує об'єктно-орієнтований підхід, що дозволяє розробникам створювати класи, об'єкти та спадковість.
- **Функціональне програмування**: JavaScript також підтримує функціональний підхід до програмування, що дозволяє використовувати функції як значення, передавати їх як аргументи та повертати з них результати.

- **Широке застосування**: На додаток до веб-розробки, JavaScript також використовується для розробки серверних додатків (за допомогою Node.js), мобільних додатків (за допомогою фреймворків, таких як React Native та Ionic) та навіть десктопних додатків.

JavaScript має багато бібліотек і фреймворків, які розширюють його можливості та спрощують розробку. Він є важливим інструментом для веб-розробки та дозволяє створювати інтерактивні, сучасні та потужні додатки.

TypeScript — компілювана мова програмування, що була розроблена компанією Microsoft та являє собою засіб розробки веб-застосунків, що розширює можливості JavaScript. Фактично є надмножиною для мови JavaScript.

TypeScript виник через ймовірні недоліки JavaScript при використанні його у великомасштабних додатках як у компанії Microsoft, так і в інших користувачів JavaScript. Проблеми з розробкою складних програм на JavaScript призвели до необхідності полегшення розробки компонентів мови.

В значній мірі простежується вплив таких мов як JavaScript, C#, Java и CoffeeScript. Вплив мови C# на TypeScript цілком пояснюється тим фактом, що ці обидів мови розробив Андерс Гейлсберг, який раніше також зробив Turbo Pascal і Delphi.

Код компілятора, для транслювання коду TypeScript у вигляд JavaScript, поширюється під відкритою ліцензією Apache, що дає змогу вести розробку публічно. Цьому сприяє і відкриті специфікації мови, що опубліковані в рамках угоди Open Web Foundation Specification Agreement.

TypeScript є зворотньою сумісним з JavaScript. Деякі режими компілятора взагалі дозволяють використовувати чистий JavaScript-код як валідний код TypeScript, але це не має сенсу через втрату ключових переваг мови. Фактично,

програму на TypeScript після трансляції коду компілятором можна виконувати в будь-якому JavaScript-рушії. Наприклад у сучасному браузері або використовувати у таких платформах як Node.js, Deno або новітнього Bun.

Описані переваги мови TypeScript та те, що ця мова фактично майже стала індустріальним стандартом, мотивує на те, щоб обрати цю мову у якості основної мови реалізації додатку.

У якості мови серверного додатку було обрано мову PHP.

Існує кілька переваг використання PHP в якості мови бекенду:

1. Широке поширення та підтримка: PHP є однією з найпопулярніших мов програмування для розробки веб-додатків. Вона має велику спільноту розробників, багато документації та сторонніх бібліотек, що робить розробку бекенду на PHP зручною та ефективною.
2. Простота вивчення та використання: PHP має простий та зрозумілий синтаксис, що полегшує вивчення мови та розробку веб-додатків. Багато новачків в програмуванні обирають PHP як свою першу мову, оскільки вона легко доступна та має велику кількість ресурсів для самостійного навчання.
3. Швидкодія: PHP зазвичай працює досить швидко, особливо при використанні оптимізованого коду та кешування. Багато фреймворків PHP мають вбудовану підтримку кешування та інші механізми для покращення продуктивності додатків.
4. Розширюваність: PHP має велику кількість розширень та бібліотек, які дозволяють розширити його функціональність та надати підтримку різних сторонніх сервісів та протоколів.
5. Інтеграція з базами даних: PHP має добру підтримку для роботи з різними базами даних, включаючи MySQL, PostgreSQL, SQLite та інші. Це дозволяє легко зв'язувати ваш бекенд з потужною базою даних.

Загалом, PHP є потужною мовою програмування для розробки бекенду веб-додатків, забезпечуючи широкий вибір інструментів, простоту використання та гнучкість.

Код серверної частини модна побачити у додатку А.

3.3 Архітектура серверної частини

При кожному запиті до API методом GET або POST проходить перевірка на метод запиту та отримується список аргументів, серед яких є токен безпеки та назва методу, який викликається.

Використання токенів безпеки в API має кілька переваг. По-перше, вони забезпечують аутентифікацію користувачів, дозволяючи встановити їх ідентичність при кожному запиті. Це дозволяє обмежити доступ до API лише автентифікованим користувачам і запобігти несанкціонованому доступу. По-друге, використання токенів безпеки підвищує рівень безпеки, оскільки їх можна генерувати з використанням сильних алгоритмів шифрування а також вони можуть мати обмежений термін дії. Це робить ускладненим злам доступу до облікових записів користувачів та забезпечує захист конфіденційної інформації. Крім того, використання токенів безпеки сприяє масштабованості системи, оскільки їх можна видавати та перевіряти без необхідності зберігання стану на сервері. Це дозволяє розподіленому середовищу ефективно обробляти автентифікацію без необхідності спільної пам'яті або сесійного управління.

Кожен з методів, що призначені для виклику через АПІ, мають область видимості public та відповідають за формування і повернення свого конктерного набору даних. Наприклад метод getListGroup відповідає за

повернення списку груп, а метод getLessonTime формує та повертає розклад дзвінків.

Кожен з методів робить формування рядку запиту мовою SQL та виконує сам запит для отримання певних даних з бази даних mySQL, де зберігаються всі дані і повертає сформаовані у формат JSON дані.

3.4 Структура навігації клієнтської частини

Клієнтський додаток розкладу з точки зору навігації побудований відразу на двох типах навігації: стековій та двох видах табової навігації. Їх поєднання в одному додатку дає можливість одночасно використовувати плюси обох типів навігації, уникаючи обмеження кожного із типів.

Особливості стекової навігації:

- **Ієрархічна структура**: Стекова навігація побудована на основі ієрархічної структури екранів. Кожен екран відображається у вигляді стеку, де нові екрани додаються на вершину, а при переході до попереднього екрану відбувається видалення останнього екрану зі стеку. Це дозволяє легко виконувати навігацію вперед і назад між екранами.
- **Навігаційні дії**: Стекова навігація надає вбудовані навігаційні дії, такі як перехід на новий екран, повернення до попереднього екрану і зміна стека екранів. Це спрощує виконання типових дій навігації без необхідності вручну керувати станом навігації.
- **Анімації переходів**: Стекова навігація дає непоганий потенціал створення анімації переходів між екранами, що робить навігацію більш привабливою для користувача. Наприклад, можна використовувати затемнення, зсуви або зміну розміру екранів під час переходу.

Частину коду з використанням стекової навігації представлено у додатку Б.

Навігація вкладками (табова навігація) також є одним із популярніших підходів до організації навігації в мобільних додатках. Особливості табової навігації включають:

- **Множинні екрани**: За допомогою табової навігації можна реалізувати майже паралельну роботу відразу на декількох екранах, кожен з яких представляє певний функціонал або категорію. Користувач може переключатися між вкладками, щоб отримати доступ до різних функцій додатку.
- **Постійна видимість**: Вкладки зазвичай розміщуються у нижній або верхній частині екрану і залишаються постійно видимими навіть при переходах між екранами. Це дозволяє швидко перемикатися між вкладками без необхідності повертатися до головного екрану.
- **Незалежність вкладок**: Кожна вкладка має свій власний стек екранів, що дозволяє незалежно керувати навігацією в межах кожної вкладки. Користувач може переходити між екранами вкладки без впливу на стан і навігацію інших вкладок.
- **Візуальне відображення активної вкладки**: Активна вкладка зазвичай відображається з видимою підсвіткою або іншим візуальним відмітком, щоб користувач завжди знаходився в обраній категорії та мав зрозумілу відповідність між вкладками і відображеними екранами.
- **Гнучкість в конфігурації**: Табова навігація може бути налаштована на основі потреб додатку. Вона може включати фіксовану кількість вкладок або динамічно змінюватися в залежності від контексту. Наприклад якщо користувач це викладач, то першою буде вкладка розкладу цього викладача, а якщо студент, то першою буде вкладка групи.

Частину коду з використанням табової навігації представлено у додатку В.

Загальну структуру навігації показано на рисунку 3.2

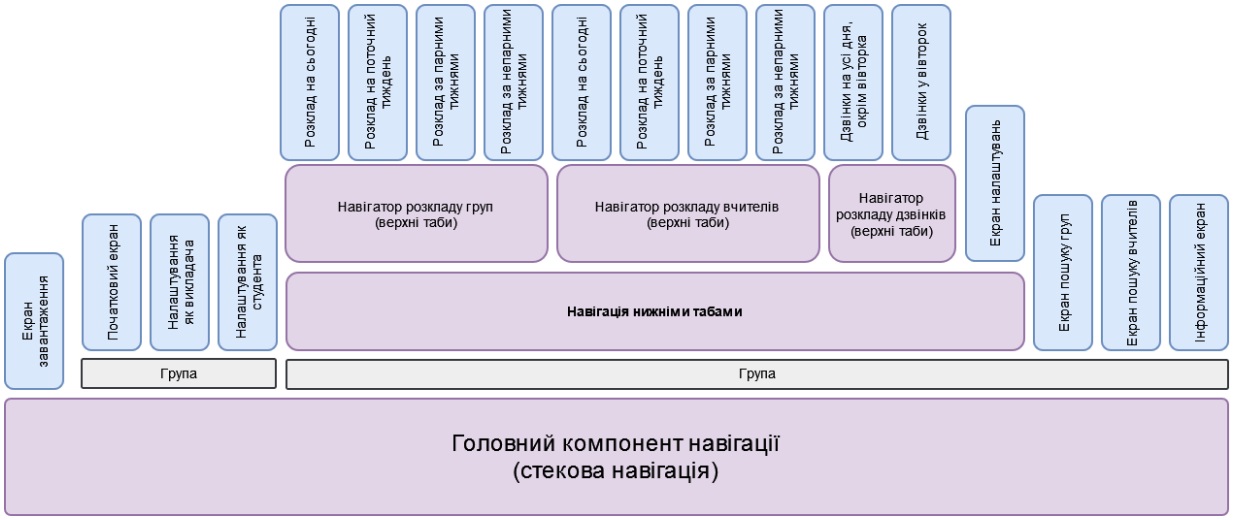

Рис. 3.2 Структура навігації

3.5 Структура даних клієнтської частини

З точки зору даних додаток додаток можна поділити на такі частини: API, Сесійне сховище даних, Асинхронне сховище даних та компоненти відображення даних.

Отримання даних через API є важливою складовою частиною клієнтського додатку у моделі клієнт-сервер. Цей процес включає в себе наступні кроки:

1. Формування посилань на апі з відповідними GET-параметрами в залежності від потреб.
2. Відправлення запиту

3. Очікування відповіді: Після відправки запиту, додаток очікує відповідь від API. В даній реалізації це виконано як асинхронний процес.

4. Отримання відповіді та береження даних: згідно з поточною архітектурою, сервер повертає дані в форматі JSON, що не потребують додаткової обробки. Після отримання даних додаток зберігає дані у відповідному місці, тобто відправляє їх у сесійне сховище даних, для подальшого використання в додатку.

Сесійне сховище даних, реалізоване за допомогою бібліотеки Redux Toolkit, є важливою складовою частиною мобільного додатку. Воно використовується для збереження тимчасових даних та стану додатку протягом сесії користувача. Основні особливості сесійного сховища включають:

- **Централізоване управління станом**: Сесійне сховище реалізоване таким методом можна використовувати для централізованого управління станом додатку. Це дозволяє зручно зберігати та оновлювати дані, які використовуються в різних компонентах додатку.
- **Нормалізація даних**: також таке рішення дозволяє нормалізувати дані в сесійному сховищі, що полегшує роботу зі складними структурами даних. Це допомагає покращити ефективність та читабельність коду, а також забезпечує консистентність та уніфікований доступ до даних.
- **Мутації та екшени**: сесійне сховище використовує мутації та екшени для зміни стану даних. Це дає можливість контролювати зміни в сесійному сховищі та спрощує відлагодження та відстеження змін стану.
- **Селектори**: Redux Toolkit надає можливість використовувати селектори для вибору певних даних зі сесійного сховища. Селектори дозволяють зручно отримувати підмножини даних та виконувати розрахунки або фільтрацію даних перед їх використанням.

- **Поділення сховища на слайси**: слайси даних є наборами функцій та редукторів, які визначають, як дані повинні бути збережені, оновлювані та витягувані з сесійного сховища. Кожен слайс містить визначення початкового набору даних, редуктори для даних і експорт функцій та селекторів, які надають зручний інтерфейс для використання та взаємодії з даними.
- **Розширюваність**: Сесійне сховище на базі Redux Toolkit дозволяє легко розширювати функціональність та внедрювати додаткові мідлвари та плагіни. Це надає можливість додавати додаткові функції до сесійного сховища, такі як логування, аналітика, аутентифікація або кешування даних. Такий підхід дозволяє зручно розширювати функціональність додатку без значних змін у сесійному сховищі, що сприяє підтримці та розробці додатку на протязі часу.

Також в даній реалізації зроблена система запису в асинхронне сховище даних тільки через екшени запису в сесійне сховище. Тобто дані в обох сховищах будуть узгоджені. Таким чином архітектура унеможливлює ситуації, коли дані можуть бути там, де їх не повинно бути та навпаки.

Асинхронне сховище даних, яке використовується в додатку, є незашифрованою, асинхронною, постійною системою зберігання ключ-значення, яка є глобальною для додатку. Вона є аналогом LocalStorage в браузері.

Асинхронне сховище даних дозволяє зберігати дані ключ-значення в мобільному пристрої, забезпечуючи постійне збереження даних навіть після закриття додатку. Воно працює асинхронно, що дозволяє уникнути блокування інтерфейсу користувача під час збереження або отримання даних.

У контексті додатку розкладу занять університету, асинхронне сховище даних може використовуватись для збереження тимчасових даних, таких як налаштування користувача, мови та об’єкта користувача. Воно забезпечує

постійне збереження цих даних, що дозволяє користувачам зручно отримувати доступ до них при кожному запуску додатку.

Потоки даних у додатку показано на рисунку 3.2

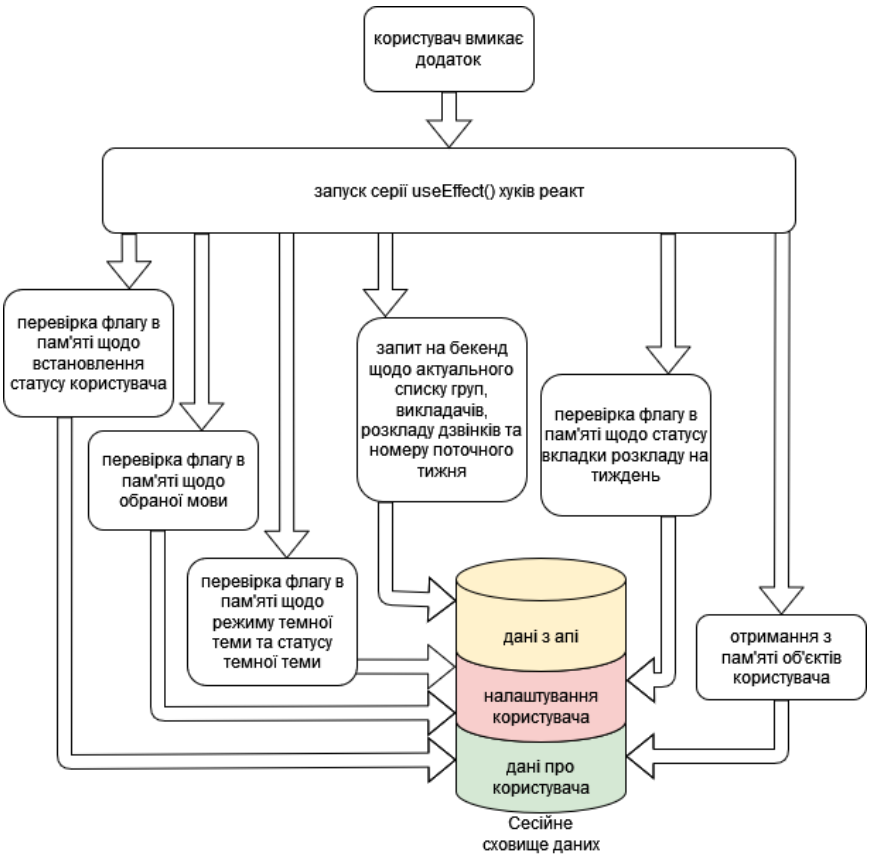

Рис. 3.3 Потоки даних у додатку

З точки зору інтерфейсу користувача (UI), додаток розкладу занять університету можна поділити на різні компоненти відображення даних, які спрощують взаємодію користувача з додатком.

3.6 Опис інтерфейсу користувача

При першому завантаженні користувача зустрічає група стартових екранів. Тут можна обрати тип користувача та групу або викладача зі списку.

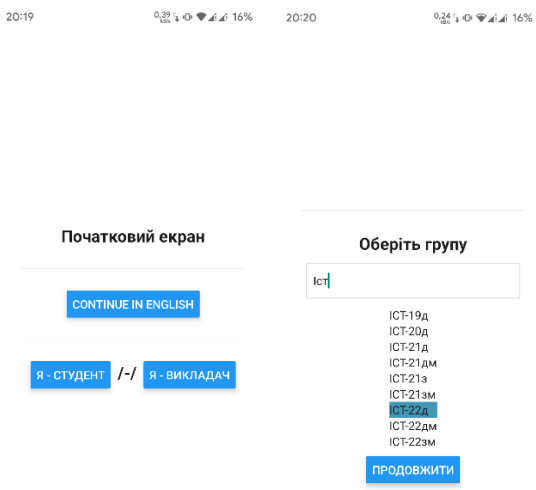

Рис. 3.4

Після входу для користувача стає доступним основний табовий компонент навігації з чотирма основними екранами: студенти, викладачі, дзвінки і налаштування.

Якщо користувач ввійшов як студент, то першим екраном буде розклад групи, на другому пошук викладачів, на третьому дзвінки та налаштування на четвертому. Тоді якщо користувач ввійшов у якості викладача, то першим буде розклад викладача, а другим пошук груп.

Дані щодо розкладу виводяться в табличному вигляді з можливістю перегляду детальніше по натисканню на рядок.

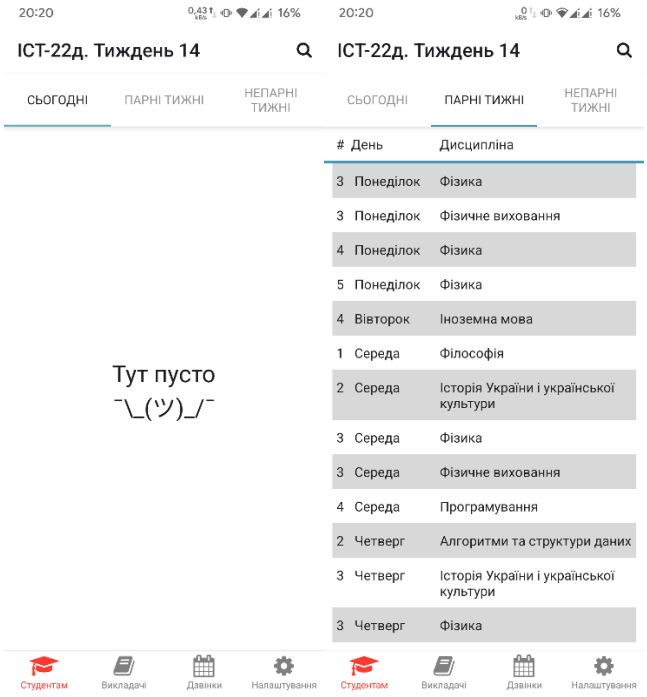

Рис.3.5

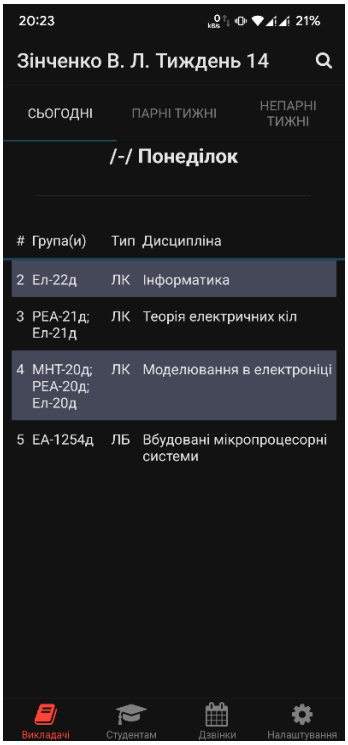

Рис. 3.6

Для покращення досвіду користувача (UX), реалізована можливість вмикати або вимикати тему, як в залежності від налаштувань пристроюю так і

локально в додатку. Також покращенню досвіду користувача сприяє можливість перемикання мови додатку. Початкова архітектура вже підтримує українську та англійську, а модульність архітектури дозволяє легко додавати нові локалізації.

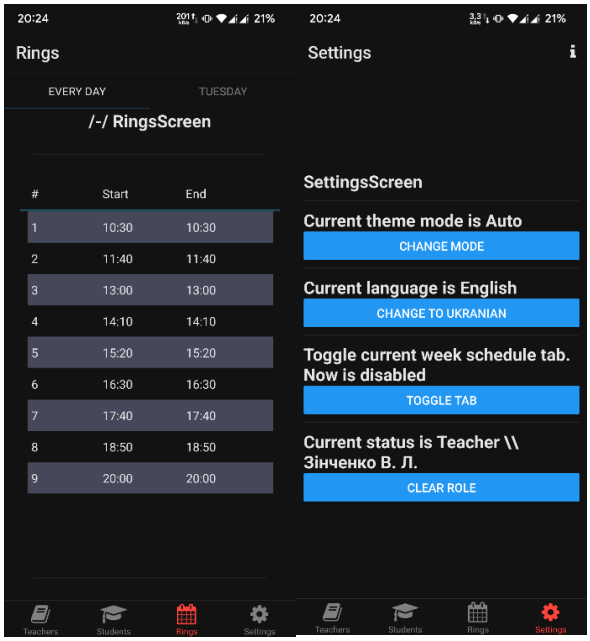

Рис. 3.7

ВИСНОВКИ

В процесі виконання даної дипломної роботи було оброблено питання надання швидкого та зручного доступу до інформації розкладу вищого навчального закладу.

Виходячи з наявних знань та досвіду розробки, було обрано стек розробки та цільову платформу проекту. Також було створено список завдань для досягнення поставленої мети.

На основі результатів аналізу було створено оптимальну архітектуру мобільного додатку та розроблено сам додаток.

Обрана методологія ведення розробки, дозволила проводити тестування додатку у моменти додавання нових функцій та можливостей. Тестування користувачами дозволило знайти декілька незначних відхилень від запланованого функціоналу.

У результаті розробки мобільного додатку розкладу занять для університету було досягнуто значних переваг у зручності та доступності для студентів та викладачів. За допомогою додатку, користувачі мають можливість швидко та зручно переглядати розклад занять на своїх мобільних пристроях, що дозволяє уникнути необхідності використовувати комп'ютер або переглядати паперовий розклад. Це надає гнучкість та ефективне управління часом та ресурсами. Крім того, обрана архітектура дозволяє отримувати завжди актуальну інформацію про розклад, що особливо важливо при змінах у розкладі. Мобільний додаток дозволяє отримувати оновлену інформацію швидко і зручно без перевірки електронної пошти або стеження за оголошеннями. В результаті впровадження мобільного додатку розкладу в університеті, студенти та викладачі зможуть більш ефективно планувати свій

час, а університет отримає зручний і сучасний інструмент для інформаційної підтримки навчального процесу.

Окрім цього в ході створення та написання пояснювальної записки з’являлися нові ідеї щодо подальшого розвитку розробленої системи для покращення ефективності та зручності роботи.

В результаті процеса створення додатка було виконано такі задачі:

1. Зроблено аналіз сучасних методів кросплатформної розробки
2. Зроблено аналіз середи розробки
3. Зроблено аналіз архітектур навігації в додатках
4. Проведено процес розробки додатку
5. Проведено процес тестування розробленого додатку

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ratov D. Architectural paradigm of the interactive interface module in the cloud technology model. Applied Computer Science. 2020. Vol. 16, no. 4. P. 48–55.

2. Ратов Д.В. Програмний контролер автоматизації формування документів з обмеженням несанкціонованого доступу. Наукові праці ДонНТУ. Серія «Інформатика, кібернетика, обчислювальна техніка». Покровськ, 2021. № 1(32). С. 49–56.

3. Ратов Д.В. Модель модуля інтерфейсу користувача інформаційної web-системи. Математичні машини і системи. Київ, 2020. № 4. С. 74–81.

4. Ratov D. Integration with the software interface of the com server for authorized user. Applied Computer Science. 2021. Vol. 17, no. 2. P. 5–13.

5. Ратов Д.В., Іванов В.Г., Лигіна Л.А. Створення мережевої системи авторизації для програмного забезпечення. Математичні машини і системи. Київ, 2021. № 2. С. 35–44.

6. React Native - Learn once, write anywhere. URL: https://reactnative.dev/docs/getting-started (дата звернення: 10.05.2023)

7. Expo - Create amazing apps that run everywhere. URL: https://docs.expo.dev/get-started/expo-go/ (дата звернення: 10.05.2023)

8. React Navigation - Routing and navigation for Expo and React Native apps. URL: reactnavigation.org/ (дата звернення: 10.05.2023)

9. Kotlin Multiplatform - Kotlin Documentation. URL: https://kotlinlang.org/docs/multiplatform.html (дата звернення: 15.05.2023)

10. Out-of-Tree Platforms - React Native. URL: https://reactnative.dev/docs/out-of-tree-platforms (дата звернення: 15.05.2023)

11. Expo Application Services (EAS). URL: https://expo.dev/eas (дата звернення: 15.05.2023)

12. Expo Go – Expo documentation. URL: https://docs.expo.dev/get-started/expo-go/ (дата звернення: 15.05.2023)

13. vscode.dev - Visual Studio Code for the Web. URL: https://code.visualstudio.com/blogs/2021/10/20/vscode-dev (дата звернення: 15.05.2023)

14. ІНФОРМАЦІЙНО-ПРОГРАМНИЙ КОМПЛЕКС РОЗКЛАДУ ЗАНЯТЬ ВИЩОГО НАВЧАЛЬНОГО ЗАКЛАДУ - Всеукраїнська наукова інтернет-конференція. Вітчизняна наука на зламі епох: проблеми та перспективи розвитку № 88. URL: https://confscience.webnode.com.ua/ (дата звернення: 05.06.2023)

ДОДАТКИ

ДОДАТОК А. КОД КЛАСУ СЕРВЕРНОЇ ЧАСТИНИ

```php
<?php
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept,
Origin, Authorization');
header('Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS');
$arrListParam = $_GET['token'] ? $_GET : $_POST;

// створення екземпляру класу
$objServer = new clServer();

// отримання назви методу запиту API
$funcName = $arrListParam['mode'];
echo $objServer->$funcName();

// реалізація класа
Class clServer {
    public $modulJS = 'Modul_Server';
    private $system = 'TimeTable';
    protected $version = '1.1';
    private $token = 'TOKEN';
    public $arWeekDay = array('неділя', 'понеділок', 'вівторок', 'середа',
'четвер', "п'ятниця", 'субота');

    // конструктор класу з початковою ініціалізацією
    public function __construct($p1) {
        global $arrListParam;
        if ($arrListParam['token'] <> $this->token) DIE("In the process no
access");
        $this->connect = mysql_connect("localhost", "ch15b1_ttime", "TOKEN") or
DIE("Не можу створити з'єднання");

        mysql_select_db("ch15b1_db_ttime", $this->connect) or DIE("Не можу
вибрати базу даних");
        $this->status = $arrListParam['status'];
        $this->fio = $arrListParam['fio'];
        $this->year = $arrListParam['year'];
        $this->semestr = $arrListParam['semestr'];
        $this->rdenne = $arrListParam['rdenne'];
        $this->idFio = trim($arrListParam['idFIO']);
        $this->date = $this->convertionDate($arrListParam['date'], 3);
        $this->idGroup = $arrListParam['idGroup'];
        $this->allGroup = 0;
    }
//_____________________________________________________________________

    // деструктор класу
```

```
    function __destruct() {
        mysql_close($this->connect);
    }
//_____________________________________________________________________

    // конвертація дати
    private function convertionDate($d, $p) {
        if (!empty($d)) {
            switch ($p) {
                case 0: // дата у форматі dd.mm.yyyy
                    $date_elements = explode(".", $d); // парсимо на масив
елементів[yyyy][mm][dd]
                    return strtotime($date_elements[0] . "-" . $date_elements[1]
. "-" . $date_elements[2]); // отримуємо дату у мілісекундах

                break;
                case 1: // дата у форматі yyyy-mm-dd
                    $year = substr($d, 0, 4);
                    if ($year == "0000" || $year == "0001") return "";
                    else {
                        $date_elements = explode("-", $d); // парсимо на масив
елементів
                        return $date_elements[2] . "." . $date_elements[1] . "."
. $date_elements[0]; // отримуємо дату у форматі dd.mm.yyyy

                    }
                    break;
                case 3: // дата у форматі dd.mm.yyyy
                    if (strlen($d) < 10) return '0000-00-00';
                    else {
                        $date_elements = explode(".", $d); // парсимо на масив
елементів
                        return $date_elements[2] . "-" . $date_elements[1] . "-"
. $date_elements[0]; // отримуємо дату у форматі yyyy-mm-dd

                    }
                    break;
                }
            } else return "";
        }
//_____________________________________________________________________

        // конвертування рядка
        private function convert($text, $move = 'cp1251') {
            if ($move == 'cp1251') return mb_convert_encoding($text, 'cp1251',
'UTF-8');
            else return mb_convert_encoding($text, 'UTF-8', 'cp1251');
```

```
        }
//_____________________________________________________________________________

        // запит до бази на отримання даних
        private function queryCursor() {
            mysql_query(sprintf("
      CREATE TEMPORARY TABLE tempTT
      SELECT timetable.*, teacher.fio, teacher.FIO_name, teacher.dol,
        lesson.lesson, groups.name as pgroup, idMoodle,
        LEFT(numLesson, 1) as les1,
        RIGHT(numLesson, 1) as les2,
        IF(dayWeek = 'Вівторок', 2, 1) as dayN_1,
        CASE dayWeek
          WHEN 'Понеділок' THEN 1 WHEN 'Вівторок'  THEN 2 WHEN 'Середа'  THEN 3
          WHEN 'Четвер'    THEN 4 WHEN 'П\'ятниця' THEN 5 WHEN 'Субота'  THEN 6
WHEN 'Неділя' THEN 7
          ELSE '-'
        END as dayN
      FROM timetable
        LEFT JOIN teacher ON timetable.idTeacher = teacher.id
        LEFT JOIN lesson ON timetable.idLesson = lesson.id
        LEFT JOIN groups ON timetable.idGroup = groups.id
      WHERE  year = \"%1\$s\" AND semestr = %2\$d AND NOT (lesType IN (\"Z\",
\"ZV\"))  AND  %3\$s ",
            /*1*/
            $this->year,
            /*2*/
            $this->semestr,
            /*3*/
            $this->status == 't' ? 'timetable.idTeacher = ' . $this->idFio //
пошук даних по ПІБ викладача
             : 'timetable.idGroup IN (' . $this->idGroup . ')' // пошук даних по
групі студента
            ), $this->connect);
            return mysql_query(sprintf("
      SELECT tempTT.*, clock.t1, clock.t2, clock.t3, clock.t4
      FROM tempTT
        LEFT JOIN clock ON tempTT.numLesson = clock.numLesson AND tempTT.dayN_1 =
clock.dayN
      ORDER BY dateL, les2, tempTT.dayN, tempTT.numLesson, lesson
    "), $this->connect);
        }

        // розрахунок номера тижня та номера дня за датою
        public function getNumWeekDay() {
            $text_queryD = sprintf("
      SELECT * FROM dateSemestr
      WHERE year = \"%1\$s\" AND semestr = %2\$d ",
```

```
            /*1*/
            $this->year,
            /*2*/
            $this->semestr);
            $query = mysql_query($text_queryD, $this->connect);
            $recD = mysql_fetch_array($query);
            $wBegin = date('W', strtotime($recD['dateBegin']));
            $wCur = date('W', strtotime($this->date));
            // номер тижня
            $weekNum = $wCur - $wBegin + 1;
            // номер дня
            $weekDay = date('w', strtotime($this->date));
            $pnWeek = (($weekNum % 2) == 0 ? 'п' : 'н');
            return ('{' . '"weekNum":"' . $weekNum . '",' . '"weekDay":"' .
$weekDay . '",' . '"weekPn":"' . $pnWeek . '"' . '}');
        }
//_____________________________________________________________________

        // отримання розкладу по ПІБ
        public function getData() {
            $arData = array();
            $arrCell = array(); // ключовий асоціативний масив осередків

            // запит до бази на отримання даних
            $query = $this->queryCursor();

            // заповнюємо ключовий масив осередків даними з бази
            while ($rec = mysql_fetch_array($query)) {
                $keyBD = $rec['dayN'] . $rec['numLesson'] . $rec['lesson'] .
$rec['numWeek'];
                $arrCell[$keyBD]['idlist'].= trim($rec['id']) . ' ';
                $arrCell[$keyBD]['id'] = $rec['id'];
                $arrCell[$keyBD]['dayN'] = $rec['dayN'];
                $arrCell[$keyBD]['dayWeek'] = $rec['dayWeek'];
                $arrCell[$keyBD]['lesson'] = $rec['lesson'];
                $arrCell[$keyBD]['id_lesson'] = $rec['idLesson'];
                $arrCell[$keyBD]['type'] = $rec['type'];
                $arrCell[$keyBD]['room'] = $rec['room'];
                $arrCell[$keyBD]['les2'] = $rec['les2'];
                $arrCell[$keyBD]['les1'] = $rec['les1'];
                $arrCell[$keyBD]['clock'] = substr($rec['t1'], 0, 5) . ' - ' .
(strlen($rec['t2']) > 0 ? substr($rec['t2'], 0, 5) . PHP_EOL . substr($rec['t3'],
0, 5) . ' - ' : '') . substr($rec['t4'], 0, 5);
                $arrCell[$keyBD]['lesType'] = $rec['lesType'];
                $arrCell[$keyBD]['pp'] = $rec['pp'];
                $arrCell[$keyBD]['numWeek'] = (strlen($rec['numWeek']) > 1 ? ' '
. $rec['numWeek'] : '');
```

```
                //$arrCell[$keyBD]['dateL']   = $this->convertionDate($rec['dateL'], 1);
                $group = ($rec['pgroup']);
                $posG = strripos($arrCell[$keyBD]['pgroupAll'], ' ' . $group . ' ');
                if ($posG === false) $arrCell[$keyBD]['pgroupAll'].= ' ' . $group . ' ';
                if ($this->allGroup == 0) {
                    $posG = strripos($arrCell[$keyBD]['pgroup'], ' ' . $group . ';');
                    if ($posG === false) $arrCell[$keyBD]['pgroup'].= ' ' . $group . '; ';
                } else {
                    $len = strlen($arrCell[$keyBD]['pgroup']);
                    switch ($len) {
                        case 0:
                            $arrCell[$keyBD]['pgroup'] = $group;
                        break;
                        case ($len >= 5 && $len <= 10):
                            $arrCell[$keyBD]['pgroup'].= ' ...';
                        break;
                    }
                }
            }

            // формування контенту таблиці тиждень із розкладом
            foreach ($arrCell as $key => $row) {
                $arData[] = '{' . '"nameGroup":"' . substr(trim($row['pgroup']), 0, -1) . '",' . '"lesson":"' . $row['lesson'] . '",' . '"dayWeek":"' . $row['dayWeek'] . '",' . '"dayN":"' . $row['dayN'] . '",' . '"num":"' . $row['les1'] . '",' . '"chet":"' . $row['les2'] . '",' . '"numWeek":"' . trim($row['numWeek']) . '",' . '"type":"' . $row['type'] . '",' . '"pp":"' . $row['pp'] . '",' . '"room":"' . $row['room'] . '",' . '"dateL":""' . '}';
            }
            return ('{' . '"id":"' . ($this->status == 't' ? $this->idFio : $this->idGroup) . '",' . '"data":[' . implode(', ', $arData) . ']' . '}');
        }
//_____________________________________________________________________________


        // отримання списку ПІБ викладачів
        public function getListFIO() {
            $arData = array();
            $text_query = sprintf("
      SELECT *
      FROM teacher
      WHERE activ =1
      ORDER BY FIO_name
    ");
```

```
            $query = mysql_query($text_query, $this->connect);
            $count = mysql_num_rows($query);
            while ($rec = mysql_fetch_array($query)) {
                $arData[] = '{' . '"id":"' . $rec['id'] . '",' . '"FIO":"' .
trim($rec['FIO_name']) . '",' . '"dol":"' . $rec['dol'] . '"' . '}';
            }
            return ('{' . '"count":' . $count . ',' . '"data":[' . implode(', ',
$arData) . ']' . '}');
        }
//_____________________________________________________________________

        // отримання списку груп
        public function getListGroup() {
            $arData = array();
            $text_query = sprintf("   SELECT groups.id, groups.name, denne,
faculty.name as FacultyName
      FROM groups
        LEFT JOIN faculty ON faculty.id = idFaculty
      WHERE activ = 1
      ORDER BY name
    ");
            $query_G = mysql_query($text_query, $this->connect);
            $count = mysql_num_rows($query_G);
            while ($rec = mysql_fetch_array($query_G)) {
                $arData[] = '{' . '"id":"' . $rec['id'] . '",' . '"group":"' .
$rec['name'] . '",' . '"denne":"' . $rec['denne'] . '",' . '"faculty":"' .
$rec['FacultyName'] . '"' . '}';
            }
            return ('{' . '"count":' . $count . ',' . '"data":[' . implode(', ',
$arData) . ']' . '}');
        }
//_____________________________________________________________________

        // отримання розкладу дзвінків
        public function getLessonTime() {
            $arData_1 = array();
            $arData_2 = array();
            $i = 1;
            $text_query = sprintf("
      SELECT * FROM clock
      WHERE dayN = 1 AND RIGHT(numLesson, 1) = \"н\"
    ");
            $query = mysql_query($text_query, $this->connect);
            while ($rec = mysql_fetch_array($query)) {
                $arData_1[] = '{' . '"num":"' . $i++ . '",' . '"beginTime":"' .
substr($rec['t1'], 0, 5) . '",' . '"endTime":"' . substr($rec['t4'], 0, 5) . '"'
. '}';
            }
```

```
            $i = 1;
            $text_query = sprintf("
      SELECT *, LEFT(numLesson, LOCATE(\"н\", numLesson, 1) - 1 ) as num
      FROM clock
      WHERE dayN = 2 AND RIGHT(numLesson, 1) = \"н\"
      ORDER BY id
    ");
            $query = mysql_query($text_query, $this->connect);
            while ($rec = mysql_fetch_array($query)) {
                $arData_2[] = '{' . '"num":"' . $rec['num'] . '",' .
'"beginTime":"' . substr($rec['t1'], 0, 5) . '",' . '"endTime":"' .
substr($rec['t4'], 0, 5) . '"' . '}';
            }
            return ('{' . '"data_everyday":[' . implode(', ', $arData_1) . '],' .
'"data_tuesday":[' . implode(', ', $arData_2) . ']' . '}');
        }
//_____________________________________________________________________________

    } ?>
```

ДОДАТОК Б. КОД СТЕКОВОЇ НАВІГАЦІЇ ГОЛОВНОГО НАВІГАТОРА

```
import React from "react";
import { createNativeStackNavigator } from "@react-navigation/native-stack";
import StartModal from "./screens/StartModal";
import { TabsNavigator } from "./TabsNavigator";

import SetupTeacher from "./screens/SetupTeacher";
import SetupStudent from "./screens/SetupStudent";
import TeacherSearchScreen from "./screens/Teacher/TeacherSearch";
import GroupSearchScreen from "./screens/Group/GroupSearch";
import StackScreenBase from "./components/StackScreenBase";
import LoadingScreen from "./screens/Loading";
import { useSelector } from "react-redux";
import { AppState } from "./store";
import InfoScreen from "./screens/Info";

export const MainNavigator = () => {
  const RootStack = createNativeStackNavigator();
  const isPassedStart = useSelector((state: AppState) =>
state.userSlice.isPassedStart);
  const groupData = useSelector((state: AppState) => state.dataSlice.groupData);
  const teacherData = useSelector((state: AppState) =>
state.dataSlice.teacherData);

  return (
    <RootStack.Navigator
      screenOptions={{
        headerShown: false,
      }}
    >
      {isPassedStart === undefined && (groupData || teacherData) ? (
        <RootStack.Screen name="loading" component={LoadingScreen} />
      ) : !isPassedStart ? (
        <RootStack.Group>
          <RootStack.Screen name="start" component={StartModal} />
          <RootStack.Screen name="setupTeacher" component={SetupTeacher} />
          <RootStack.Screen name="setupStudent" component={SetupStudent} />
        </RootStack.Group>
      ) : (
        <RootStack.Group>
          <RootStack.Screen name="main" component={TabsNavigator} />
          <RootStack.Screen name="info">
            {() => (
              <StackScreenBase>
                <InfoScreen />
```

```
            </StackScreenBase>
          )}
        </RootStack.Screen>
        <RootStack.Screen name="searchTeacher">
          {() => (
            <StackScreenBase>
              <TeacherSearchScreen />
            </StackScreenBase>
          )}
        </RootStack.Screen>
        <RootStack.Screen name="searchGroup">
          {() => (
            <StackScreenBase>
              <GroupSearchScreen />
            </StackScreenBase>
          )}
        </RootStack.Screen>
      </RootStack.Group>
    )}
  </RootStack.Navigator>
);
};
```

ДОДАТОК В. КОД ТАБОВОЇ НАВІГАЦІЇ ГОЛОВНОГО ЕКРАНУ

```
import React from "react";
import { useSelector } from "react-redux";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import { useNavigation, useTheme } from "@react-navigation/native";

import useTabs from "./hooks/useTabs";
import { IHeaderRight } from "./types/headerRight.interface";
import { NavigationProps } from "./types/navigation.type";
import { AppState } from "./store";

export const TabsNavigator = () => {
  const navigation = useNavigation<NavigationProps>();
  const Tab = createBottomTabNavigator();
  const { colors } = useTheme();

  const teacher = useSelector((state: AppState) => state.userSlice.teacher);
  const group = useSelector((state: AppState) => state.userSlice.group);

  const nav: {
    students?: IHeaderRight;
    teachers?: IHeaderRight;
    settings?: IHeaderRight;
  } = {
    students: undefined,
    teachers: undefined,
    settings: {
      f: () => {
        navigation.navigate("info");
      },
      name: "info",
      size: 22,
    },
  };

  if (teacher.id) {
    nav["teachers"] = {
      name: "search",
      f: () => {
        navigation.navigate("searchTeacher");
      },
      size: 20,
    };
  }
  if (group.id) {
    nav["students"] = {
```

```
      name: "search",
      f: () => {
        navigation.navigate("searchGroup");
      },
      size: 20,
    };
  }

  const { tabs } = useTabs(nav);

  return (
    <Tab.Navigator
      screenOptions={() => ({
        tabBarActiveTintColor: colors.notification,
        tabBarInactiveTintColor: "gray",
      })}
    >
      {tabs.map((item, index) => (
        <Tab.Screen
          key={index}
          name={item.name}
          component={item.component}
          options={item.options}
        />
      ))}
    </Tab.Navigator>
  );
};
```