

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ПОГОДЖЕНО
Декан факультету

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри інформаційних
систем і технологій

(підпис) **Ігор БОЛБОТ**

(підпис) **Михайло ШВИДЕНКО**

« ____ » _____ 2026 р.

« ____ » _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему: «Розробка підсистеми навчального процесу інтегрованої мобільної
системи підтримки студентів»**

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми
д.е.н, професор

(підпис) **Роман РУДЕНСЬКИЙ**

**Керівник бакалаврської
кваліфікаційної роботи**
асистент

(підпис) **Ярослав ПОНЗЕЛЬ**

Консультант
к.п.н., доцент

(підпис) **Тетяна ВОЛОШИНА**

Виконав

(підпис) **Микита СКРИПНИК**

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій**

ЗАТВЕРДЖУЮ
**Завідувач кафедри інформаційних систем і
технологій**

к.е.н., доцент _____ Михайло ШВИДЕНКО
(підпис)

« ____ » _____ 2026 р.

**ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ
Скрипнику Микиті Олексійовичу**

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Розробка підсистеми навчального процесу інтегрованої мобільної системи підтримки студентів»

затверджена наказом від «06» січня 2026 р. № 46 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):

інформація про розклад занять, факультети, групи, корпуси й гуртожитки НУБіП України, координати об'єктів кампусу, профілі користувачів та інтеграція із зовнішніми сервісами через API.

Перелік питань, що підлягають дослідженню:

проектування й реалізація мобільного додатка для перегляду розкладу, навігації кампусом, авторизації користувачів, кешування даних та тестування основних функціональних модулів.

Дата видачі завдання «06» січня 2026 р.

**Керівник бакалаврської
кваліфікаційної
роботи**

(підпис)

Ярослав ПОНЗЕЛЬ

Консультант

(підпис)

Тетяна ВОЛОШИНА

**Завдання взяв до
виконання**

(підпис)

Микита СКРИПНИК

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	5
ВСТУП.....	6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ.....	9
1.1 Актуальність теми та проблематика доступу студентів до актуальних даних.....	9
1.2 Аналіз існуючих інформаційних систем у сфері студентської навігації....	10
1.3 Визначення цілей та завдань дослідження.....	12
1.4 Формулювання вимог до підсистеми навчального процесу мобільної інформаційної системи.....	13
2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	17
2.1 Вибір архітектурного підходу та технологічного стеку.....	17
2.2 Логічна і фізична модель бази даних.....	19
2.3 Моделювання бізнес-процесів за допомогою UML-діаграм.....	22
2.3.1 Діаграма прецедентів.....	23
2.3.2 Діаграма діяльності.....	25
2.3.3 Діаграма послідовностей.....	28
2.3.4 Діаграма розгортання.....	31
2.4 Ролі користувачів і політика доступу.....	34
2.5 Визначення ключових сценаріїв використання системи.....	37
2.6 Інтеграція зовнішніх сервісів.....	39
3 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ.....	42
3.1 Реалізація серверної частини мобільної інформаційної системи.....	42
3.2 Реалізація клієнтської частини мобільного додатка.....	45
3.3 Реалізація модуля перегляду розкладу з використанням каскадної логіки отримання даних.....	52
3.4 Побудова API-взаємодії між frontend і backend.....	55
3.5 Розгортання та підготовка мобільного додатка до експлуатації.....	57
4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ.....	59
4.1 Методика тестування функціональних модулів.....	59
4.2 Тест-кейси та результати перевірки.....	60

4.3 Виявлення недоліків і заходи з їх усунення.....	62
4.4 Оцінка результатів тестування та готовності системи до використання....	62
4.5 Оцінка користі, продуктивності й зручності системи.....	63
4.6 Перспективи масштабування.....	64
ВИСНОВКИ.....	66
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	68
Додаток А.....	70

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface, програмний інтерфейс для взаємодії між програмними компонентами або сервісами.

BaaS – Backend-as-a-Service, модель хмарного сервісу, за якої серверна частина системи надається як готова платформа.

JSON – JavaScript Object Notation, текстовий формат обміну структурованими даними між клієнтською та серверною частинами.

UI – User Interface, користувацький інтерфейс.

SDK – Software Development Kit, набір інструментів для розробки програмного забезпечення.

SQL – Structured Query Language, мова структурованих запитів для роботи з реляційними базами даних.

ETL – Extract, Transform, Load, процес отримання, перетворення та завантаження даних із зовнішніх джерел до бази даних.

UML – Unified Modeling Language, уніфікована мова моделювання для опису структури та поведінки програмних систем.

НУБіП – Національний університет біоресурсів і природокористування України.

RLS – Row Level Security, механізм керування доступом до рядків таблиць у базі даних.

БД – база даних.

HTTPS - HyperText Transfer Protocol Secure, розширення протоколу HTTP для підтримки шифрування.

APK – Android Package Kit, формат файлу додатка для операційної системи Android.

ВСТУП

Ефективність навчального процесу в університеті залежить не лише від якості освітніх програм, а й від того, наскільки швидко студент отримує потрібну організаційну інформацію. Для студентів Національного університету біоресурсів і природокористування України особливо актуальними залишаються два щоденні завдання: доступ до актуального розкладу занять і орієнтування на території кампусу. Розклад, інформація про навчальні корпуси, гуртожитки та маршрути між ними часто сприймаються як допоміжні дані, однак саме від них залежить зручність планування навчального дня.

На практиці студенту доводиться користуватися кількома джерелами інформації. Частина даних розміщується на офіційних вебресурсах університету, частина передається через старост груп або неформальні канали комунікації. Такий підхід створює фрагментарність. Особливо гостро ця проблема проявляється під час зміни аудиторії, пошуку потрібного корпусу або роботи з розкладом у місцях зі слабким інтернет-з'єднанням. Для першокурсників це стає окремим бар'єром. Потрібно швидко знайти корпус, зрозуміти маршрут і не запізнитися на заняття.

Актуальність теми зумовлена потребою у створенні мобільного інструмента, який об'єднує навчальну та навігаційну інформацію в одному середовищі. Нативний мобільний додаток дає змогу швидко відкривати розклад, переглядати перелік корпусів і гуртожитків, працювати з інтерактивною картою та використовувати збережені дані навіть за нестабільного з'єднання. Це відповідає реальним умовам використання системи в межах університетського кампусу.

Метою бакалаврської кваліфікаційної роботи є розробка мобільної інформаційної системи для студентів НУБіП України, яка забезпечує доступ до розкладу занять, навігації територією кампусу та підтримує автономне відображення основних даних.

Об'єктом дослідження є процеси інформаційного забезпечення студентів у мобільних інформаційних системах закладу вищої освіти.

Предметом дослідження є методи, засоби та програмні рішення для реалізації мобільного додатка з функціями перегляду розкладу, навігації кампусом, автентифікації користувачів, кешування та синхронізації даних.

Для досягнення поставленої мети в роботі передбачено розв'язання таких завдань: проаналізувати потреби студентів щодо доступу до навчальної та навігаційної інформації; дослідити існуючі підходи до побудови мобільних інформаційних систем; визначити функціональні вимоги до додатка; спроектувати архітектуру системи та структуру бази даних; реалізувати модулі перегляду розкладу, профілю користувача, списку корпусів і гуртожитків, інтерактивної мапи та побудови маршруту; налаштувати автентифікацію через корпоративну пошту; забезпечити кешування даних для офлайн-доступу; протестувати основні функціональні модулі мобільного додатка.

У процесі виконання роботи використано методи системного аналізу для визначення проблемної області та вимог до системи, методи об'єктно-орієнтованого проєктування для побудови UML-діаграм, методи моделювання баз даних для формування структури збереження навчальної й навігаційної інформації, а також методи функціонального тестування для перевірки коректності роботи програмних модулів. Для реалізації клієнтської частини застосовано React Native та Expo, для серверної взаємодії – Supabase (BaaS), PostgreSQL, Supabase Auth, API-запити та механізми Row Level Security. Для автентифікації використано корпоративну пошту університету через Google OAuth, а для роботи з картою – картографічні сервіси та мобільні компоненти відображення геоданих.

Практичне значення роботи полягає у створенні робочого мобільного додатка, який може використовуватися студентами для швидкого доступу до розкладу занять і навігації територією НУБіП України.

Бакалаврська кваліфікаційна робота викладена на 70 сторінках, складається зі вступу, чотирьох розділів, висновків, списку використаних джерел

і одного додатка. У роботі використано 18 джерел. У першому розділі розглянуто актуальність теми, проблематику доступу студентів до навчальної та навігаційної інформації, а також існуючі інформаційні системи у цій сфері. У другому розділі визначено вимоги до системи, описано архітектурний підхід, технологічний стек, UML-моделі, ролі користувачів, політику доступу й інтеграцію зовнішніх сервісів. У третьому розділі подано реалізацію клієнтської частини мобільного додатка, механізмів кешування, API-взаємодії, автентифікації та підготовки застосунку до розгортання. У четвертому розділі наведено методику тестування, тест-кейси, результати перевірки, виявлені недоліки, оцінку користі системи та перспективи масштабування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність теми та проблематика доступу студентів до актуальних даних

У навчальному процесі студент постійно працює з оперативними відомостями: розкладом занять, аудиторіями, корпусами, часом проведення пар і змінами в організації навчання. Такі дані мають практичну цінність тоді, коли вони доступні у зручному форматі та можуть бути швидко використані під час планування навчального дня.

Навчальна інформація часто розміщується в різних джерелах: на сайті університету, у файлах із розкладом, у повідомленнях академічної групи або передається через старосту. Через це студенту доводиться витратити додатковий час на пошук потрібних відомостей і перемикання між кількома каналами доступу.

Для університету з великою територією важливо, щоб навчальна інформація була пов'язана з навігаційними можливостями. Студенту недостатньо лише побачити аудиторію чи корпус у розкладі; зручніше одразу мати доступ до карти кампусу, розташування навчальних корпусів, гуртожитків і маршрутів між ними. Додаткову складність створює використання навчальних даних зі смартфона, оскільки таблиці розкладу та вебсторінки не завжди зручні для швидкого перегляду на мобільному пристрої.

Актуальність теми полягає в необхідності створення мобільної інформаційної системи, яка централізує доступ до навчальної інформації, забезпечує зручне мобільне подання розкладу та поєднує його з навігацією територією університету.

1.2 Аналіз існуючих інформаційних систем у сфері студентської навігації

Серед існуючих рішень можна виділити кілька груп: офіційні вебпортали університетів, електронні навчальні платформи, студентські інформаційні системи, загальні картографічні сервіси та спеціалізовані системи кампусної навігації. Кожна група закриває окрему потребу.

Офіційні вебсайти університетів залишаються базовим джерелом інформації. На них розміщують новини, контакти кафедр, документи, оголошення, розклад занять і матеріали для студентів. Такий формат підходить для загального інформування, але не завжди працює як швидкий мобільний інструмент. Студент відкриває сайт не для перегляду всієї структури університету. Йому треба знайти конкретну групу, день тижня, аудиторію або корпус.

Електронні навчальні платформи, наприклад Moodle, працюють з іншим типом задач. Moodle App підтримує доступ до курсів, навчальних матеріалів, оцінок, повідомлень, календарних подій і частини функцій в офлайн-режимі, зокрема нотаток, приватних файлів, сповіщень і подій календаря. Це корисно для навчального процесу, але Moodle не розв'язує проблему просторової орієнтації.

Інший приклад – корпоративні студентські інформаційні системи, зокрема рішення Ellucian Student. Такі платформи орієнтуються на повний студентський цикл: вступ, навчальні дані, адміністративні процеси, взаємодію студентів, викладачів і персоналу університету. В окремих університетських впровадженнях мобільні рішення на базі Ellucian Go включають карти кампусу, академічний календар, оцінки та інші студентські сервіси. Це ближче до потрібної моделі. Але така система потребує глибокої інтеграції з внутрішньою інфраструктурою закладу освіти.

Загальні картографічні сервіси, зокрема Google Maps, добре працюють із побудовою маршрутів у міському просторі. Вони допомагають знайти адресу, оцінити відстань і прокласти шлях до потрібної точки. Крім того, Google Indoor

Maps підтримує перегляд планів окремих будівель, пошук об'єктів усередині них і перемикання між поверхами, якщо організація додала такі дані до сервісу.

Проте Google Maps не знає навчального контексту. Сервіс не пов'язує маршрут із розкладом конкретної групи, не показує, яка пара починається наступною, і не пояснює, чому студенту потрібно саме до цього корпусу.

Через це студент часто збирає маршрут вручну: спочатку відкриває розклад, уточнює аудиторію, потім шукає корпус у картографічному сервісі. Якщо заняття перенесли, він перевіряє інформацію через старосту або одногрупників. Такий алгоритм дій викликає низку незручностей для кінцевого користувача.

Спеціалізовані системи кампусної навігації працюють точніше. Наприклад, Navigine пропонує рішення для університетів із підтримкою indoor-навігації, побудови маршрутів, визначення місцезнаходження в реальному часі та інтеграції SDK у вже наявний мобільний додаток закладу. Такі системи добре підходять для великих кампусів, де студентам, викладачам і відвідувачам потрібно орієнтуватися не лише між корпусами, а й усередині будівель. Але точність має ціну. Подібні рішення потребують підготовлених карт, регулярного оновлення планів, технічної підтримки та інтеграції з мобільною інфраструктурою університету. Для базового студентського сценарію іноді достатньо простішої моделі: перелік корпусів, координати об'єктів, карта кампусу й маршрут до потрібної точки.

Для НУБіП України така модель має практичний сенс. Студенту потрібно швидко зрозуміти, де розташований навчальний корпус, як пройти від гуртожитку до аудиторії, що змінилося в розкладі та чи можна відкрити потрібні дані без стабільного інтернету. Особливо це стосується першокурсників. Номер корпусу для них ще не означає фактичне місце на карті.

Аналіз існуючих систем показує, що більшість із них закриває лише частину студентського маршруту. Moodle працює з курсами, але не веде до корпусу. Google Maps будує маршрут, але не бачить розкладу. Ellucian охоплює широкий спектр студентських сервісів, але потребує складного впровадження.

Navigine дає точну кампусну навігацію, проте може бути надмірним для базової задачі.

Для цього дипломного проєкту було обрано компактнішу модель: розклад, профіль користувача, карта кампусу, збереження даних і маршрут до потрібного об'єкта в одному мобільному середовищі. Такий підхід не замінює всі університетські сервіси. Він усуває конкретний розрив між навчальною інформацією та просторовою орієнтацією студента.

Користувач відкриває додаток, переглядає розклад, бачить потрібний корпус і за потреби переходить до маршруту. Без окремого пошуку файлу. Без ручного перенесення адреси в карту.

1.3 Визначення цілей та завдань дослідження

Зважаючи на результати проведеного аналізу проблеми доступу здобувачів освіти до актуальних навчальних відомостей та обмеження існуючих інформаційних систем, сформульовано основну мету дослідження. Вона полягає в обґрунтуванні, проєктуванні та програмній реалізації мобільного додатка для студентів Національного університету біоресурсів і природокористування України, який у межах єдиного клієнтського інтерфейсу поєднає функції динамічного перегляду розкладу занять, інтерактивної карти кампусу з переліком основних об'єктів та побудови маршрутів до вибраних локацій.

Зазначена мета має прикладний інженерний характер, оскільки розроблюваний програмний продукт орієнтований на щоденне використання в реальних умовах освітнього процесу. Зокрема, система має допомагати студенту швидко отримувати відомості про заняття під час перерв між парами, у разі зміни аудиторії або корпусу, а також за умов нестабільного мережевого з'єднання. Впровадження такого рішення зменшує час, який студент витрачає на пошук розкладу, аудиторії або потрібного корпусу.

Проєктна архітектура розробки охоплює кілька взаємопов'язаних функціональних напрямів. Логіка опрацювання навчальних даних передбачає

фільтрацію розкладу за факультетами, академічними групами та днями тижня. Для забезпечення стійкої роботи системи використано механізми локального кешування, які дають змогу відображати раніше збережені дані за відсутності стабільного інтернет-з'єднання. Навігаційний модуль, у свою чергу, забезпечує візуалізацію географічних координат навчальних корпусів і гуртожитків та підтримує побудову маршрутів до вибраних об'єктів кампусу.

Успішна реалізація визначеної мети передбачає послідовне вирішення комплексу науково-практичних завдань кваліфікаційної роботи. На початковому етапі здійснюється аналіз потреб студентської спільноти НУБіП України та дослідження існуючих картографічних й інформаційних рішень, що дозволяє деталізувати функціональні та нефункціональні вимоги до майбутнього додатка. Далі розробляється загальна архітектурна модель системи, будуються UML-діаграми взаємодії компонентів і проєктується реляційна структура бази даних у середовищі Supabase/PostgreSQL для зберігання навчальної та навігаційної інформації.

Практичний етап роботи зосереджується на програмній реалізації клієнтської логіки модуля розкладу занять, впровадженні механізмів кешування даних для підтримки підходу Offline-First, а також розробленні навігаційного компонента на основі мобільної карти з відображенням інфраструктури кампусу. Окремо реалізується підсистема автентифікації та профілю користувача із налаштуванням політик безпеки на основі рольової моделі Row Level Security. На завершальному етапі виконується ETL синхронізація даних із хмарним бекендом та проводиться тестування основних програмних модулів мобільної системи.

1.4 Формулювання вимог до підсистеми навчального процесу мобільної інформаційної системи

Проектування підсистеми навчального процесу в межах мобільної інформаційної системи спрямоване на забезпечення швидкого доступу студентів

до актуального розкладу занять та супутніх навчальних даних. У межах цієї роботи зазначений модуль не розглядається як статичний довідковий екран. Він виконує роль основного інформаційного ядра додатка, з яким користувач взаємодіє найчастіше протягом навчального дня.

Основна вимога до підсистеми полягає в оптимізації процесу отримання даних. Студент має бачити свій графік без тривалого пошуку на вебсайтах, відкриття громіздких таблиць чи ручного зіставлення відомостей із різних джерел. Логіка інтерфейсу побудована так, що після первинного вибору факультету та академічної групи система відображає розклад у зручному для мобільного пристрою форматі, без зайвих переходів між сторінками.

Функціональні вимоги до підсистеми безпосередньо впливають із сценаріїв її повсякденного використання. Після вказання факультету, групи та потрібного дня тижня додаток має сформувати структурований перелік занять. Для кожної пари обов'язково відображаються часові межі проведення, назва дисципліни, аудиторія, номер корпусу та відомості про викладача. Саме ці поля мають практичну цінність, оскільки дозволяють здобувачеві освіти не лише планувати навчальний час, а й оперативно орієнтуватися на території університетського кампусу.

Подання розкладу у вигляді традиційних матричних таблиць є недостатньо зручним для мобільних екранів через потребу в масштабуванні та горизонтальному прокручуванні. Для цього проєкту обрано лінійний формат відображення. Кожне заняття подається окремим адаптивним блоком, що зменшує візуальне перевантаження й фокусує увагу користувача на конкретних навчальних даних.

Окремий блок архітектурних вимог стосується забезпечення актуальності та узгодженості даних. Підсистема повинна взаємодіяти із зовнішнім API розкладу університету, отримуючи оновлення у структурованому форматі JSON. Проте отримані дані не залишаються у “сирому” вигляді. Архітектура системи передбачає їх обробку, приведення до внутрішньої реляційної структури, пов'язування з сутностями факультетів, груп, викладачів, аудиторій і корпусів та

подальше збереження у хмарній базі даних Supabase на базі PostgreSQL. Лише після такої обробки мобільний клієнт використовує інформацію для відображення розкладу.

Водночас розроблюваний модуль не може повністю залежати від постійного підключення до інтернету. У реальних умовах кампусу НУБіП України студент часто перевіряє розклад у приміщеннях із послабленим мобільним сигналом, під час переходів між корпусами або в зонах із нестабільним покриттям. Через це обов'язковою вимогою є підтримка підходу Offline-First за допомогою локального кешування даних через AsyncStorage. За наявності стабільного інтернет-з'єднання додаток оновлює локальне сховище, а в разі втрати сигналу переходить до відображення останньої збереженої версії розкладу. Таке архітектурне рішення зменшує ризик появи порожніх екранів і критичних помилок, що підвищує відмовостійкість системи.

Для покращення ергономіки інтерфейсу в підсистемі реалізовано функцію персоналізації, завдяки якій параметри факультету та групи можуть зберігатися у внутрішній пам'яті смартфона. Це усуває потребу в повторному ручному виборі при кожному новому сеансі роботи. Для щоденного використання така деталь має суттєве значення.

До нефункціональних вимог підсистеми належать швидкодія, низька затримка відгуку інтерфейсу користувача та коректна обробка виняткових ситуацій. Інтерфейс повинен залишатися зрозумілим навіть для першокурсників, які ще не орієнтуються впевнено в інфраструктурі закладу. Якщо дані не вдалося оновити, система має повідомити про це користувача або використати локально збережену версію розкладу.

З погляду безпеки та контролю доступу підсистема враховує рольову модель користувачів. Звичайний студент має права на перегляд розкладу, тоді як адміністративні дії, пов'язані із синхронізацією бази або коригуванням даних, повинні бути обмежені на рівні серверної частини. Розмежування доступу реалізується за допомогою Supabase Auth та політик безпеки Row Level Security (RLS), що забезпечує додатковий рівень захисту на рівні бази даних.

Сформульовані вимоги визначають підсистему навчання як центральний компонент мобільної інформаційної системи. Вона поєднує отримання актуального розкладу, його обробку, локальне кешування, персоналізацію користувацьких параметрів і розмежування прав доступу. Саме ці характеристики роблять модуль придатним для використання в умовах освітнього процесу НУБіП України.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вибір архітектурного підходу та технологічного стеку

Специфіка функціонування мобільного додатка для студентської навігації в межах університетського кампусу потребує використання гібридної архітектурної моделі. Традиційна клієнт-серверна модель у чистому вигляді не повністю відповідає умовам використання такого додатка, оскільки стабільність його роботи значною мірою залежить від якості мережевого з'єднання. У реальних умовах студенти можуть втрачати доступ до інтернету в цокольних або підвальних приміщеннях навчальних корпусів НУБіП України, а також під час переміщення між віддаленими об'єктами кампусу. Якщо додаток повністю залежить від синхронних запитів до зовнішнього сервера, будь-який технічний збій або послаблення сигналу може ускладнити доступ до основних функцій.

З огляду на це базовим архітектурним підходом у розробці обрано концепцію Offline-First. Відповідно до цієї моделі клієнтська частина додатка може спочатку звертатися до локально збережених даних на мобільному пристрої, а вже після цього виконувати асинхронний запит до віддаленого сервера для перевірки наявності оновлень у розкладі занять або даних про об'єкти кампусу. Якщо зв'язок із сервером тимчасово відсутній, додаток продовжує працювати з локальним кешем. Це зменшує залежність користувача від якості інтернет-покриття та підтримує доступ до основних сервісів навіть у несприятливих умовах.

Для програмної реалізації клієнтської частини системи використано фреймворк React Native у поєднанні з інструментарієм Expo SDK. Такий вибір зумовлений фрагментацією мобільних платформ серед цільової аудиторії, оскільки в межах однієї академічної групи можуть бути користувачі пристроїв під керуванням як iOS, так і Android. Застосування кросплатформеної технології React Native дає змогу створювати єдину кодову базу JavaScript/TypeScript для

обох екосистем, скорочуючи трудовитрати на розробку та подальше супроводження програмного продукту. Платформа Expo додатково спрощує цей процес, оскільки надає доступ до нативних можливостей смартфона, зокрема геолокації, локального сховища та інших системних функцій через готові програмні інтерфейси.

Організація збереження даних, автентифікація користувачів і підтримка реляційної структури навчальної інформації потребують надійного серверного середовища. Замість проєктування власного бекенду з нуля у цій роботі використано хмарну платформу Supabase, яка реалізує концепцію Backend-as-a-Service. Вбудована в Supabase система керування базами даних PostgreSQL є доцільним інструментом для структурування навчальної інформації завдяки підтримці реляційних зв'язків між сутностями факультетів, академічних груп, дисциплін, викладачів, аудиторій і корпусів. Крім того, механізм Row Level Security дає змогу обмежувати права доступу користувачів на рівні бази даних: студенти отримують доступ до операцій читання, а адміністративні дії з редагування або синхронізації даних доступні лише користувачам із відповідними правами.

Окрему увагу під час проєктування приділено роботі з геопросторовими даними та збереженню інформації на мобільному пристрої. Використання WebView для відображення карти є менш доцільним, оскільки такий підхід може мати нижчу продуктивність і гіршу інтеграцію з нативними можливостями пристрою. У клієнтській частині задіяно бібліотеку react-native-maps, яка забезпечує нативне відображення карти та підтримку власних маркерів навчальних корпусів і гуртожитків. Для практичного втілення підходу Offline-First використано AsyncStorage. Кешований розклад зберігається у форматі JSON у локальному сховищі пристрою, тому додаток може відображати останні отримані дані без помітної затримки для користувача навіть за відсутності активного інтернет-з'єднання.

2.2 Логічна і фізична модель бази даних

База даних мобільної інформаційної системи має реляційну структуру та побудована навколо десяти взаємопов'язаних сутностей: User, Profile, Faculty, Group, Schedule, Lesson, Subject, Teacher, Auditorium та Building. Така модель дає змогу зберігати користувацькі, навчальні та навігаційні дані в узгодженому вигляді, а також забезпечує швидке отримання інформації, необхідної для роботи мобільного додатка.

Перший логічний блок відповідає за ідентифікацію та профілювання користувачів. Сутність User зберігає базові облікові дані, зокрема унікальний ідентифікатор UserID, повне ім'я користувача fullname, електронну пошту email та роль role. Поле role використовується для розмежування прав доступу між студентом і адміністратором. Деталізована інформація про користувача винесена в таблицю Profile, яка містить поля avatar, bio, last_login та id_group. Зв'язок між User і Profile реалізовано як відношення «один-до-одного» через поле UserID, що в таблиці Profile одночасно є первинним і зовнішнім ключем. Такий поділ дозволяє відокремити облікові дані від додаткових персональних відомостей і спростити перевірку ролі користувача під час авторизації.

Навчальна ієрархія системи формується через сутності Faculty та Group. Таблиця Faculty містить поля id_faculty і faculty_name та описує факультети університету. Таблиця Group містить id_group, group_name і зовнішній ключ id_faculty, який пов'язує академічну групу з відповідним факультетом. У такій структурі один факультет може об'єднувати кілька груп, а кожна група належить лише одному факультету. Це відповідає реальній організаційній структурі закладу освіти та використовується для фільтрації розкладу.

Окрему роль виконує сутність Schedule. Вона містить поля id_schedule, year, semester, start_date, end_date та id_group. Таблиця Schedule пов'язує розклад із конкретною академічною групою і визначає навчальний період, у межах якого діють записи занять. Завдяки цьому система може розділяти розклади за навчальними роками, семестрами та групами.

Центральною сутністю навчального блоку є таблиця Lesson. Саме вона формує основу розкладу, який відображається в мобільному додатку. Запис у Lesson містить дані про час початку та завершення заняття (timeStart, timeEnd), день тижня (day_of_week), порядковий номер пари (lesson_number), тип заняття (type) і статус (status). Через зовнішні ключі id_schedule, id_auditorium, id_subject та id_teacher таблиця Lesson пов'язується з розкладом, аудиторією, дисципліною та викладачем.

Довідкова інформація про навчальні дисципліни зберігається в таблиці Subject, яка містить поля id_subject і name. Відомості про викладачів винесені в таблицю Teacher, де передбачено поля id_teacher, name, department та contactInfo. Такий підхід зменшує дублювання даних, оскільки одна дисципліна або один викладач можуть бути пов'язані з кількома заняттями. Під час відображення розкладу мобільний додаток отримує не окремі технічні ідентифікатори, а сформовану інформаційну картку заняття з назвою дисципліни, часом проведення, аудиторією та даними викладача.

Просторову навігацію забезпечує зв'язка сутностей Building та Auditorium. Таблиця Building містить поля id_building, name, address, latitude та longitude. Вона описує корпуси й інші об'єкти кампусу, які можуть відображатися на інтерактивній карті. Таблиця Auditorium містить id_auditorium, number та id_building, тобто зберігає інформацію про аудиторії й пов'язує кожну з них із конкретним корпусом.

Практична інтеграція розкладу з навігаційним модулем реалізується через реляційний зв'язок Lesson - Auditorium - Building. Коли користувач переглядає заняття, система може визначити аудиторію, у якій воно проводиться, а через неї – відповідний корпус, його адресу та географічні координати. Це дає змогу пов'язати навчальну інформацію з картою кампусу та забезпечити перехід від перегляду розкладу до побудови маршруту до потрібної будівлі.

Фізична реалізація моделі здійснюється в середовищі PostgreSQL на платформі Supabase. Первинні ключі таблиць використовуються як унікальні ідентифікатори записів, а зовнішні ключі забезпечують зв'язки між залежними

сутностями. Така структура підтримує цілісність даних і зменшує ризик появи неузгоджених записів, наприклад заняття без викладача, аудиторії без корпусу або розкладу без прив'язки до академічної групи.

Контроль доступу до даних реалізується за допомогою Supabase Auth та механізму Row Level Security (RLS). Відповідно до рольової моделі, звичайний студент має доступ переважно до читання навчальних і навігаційних даних, зокрема розкладу, аудиторій та корпусів. Адміністратор отримує розширені права для додавання, редагування й оновлення записів. Це дозволяє обмежити несанкціоновані зміни та підтримувати актуальність інформації в базі даних.

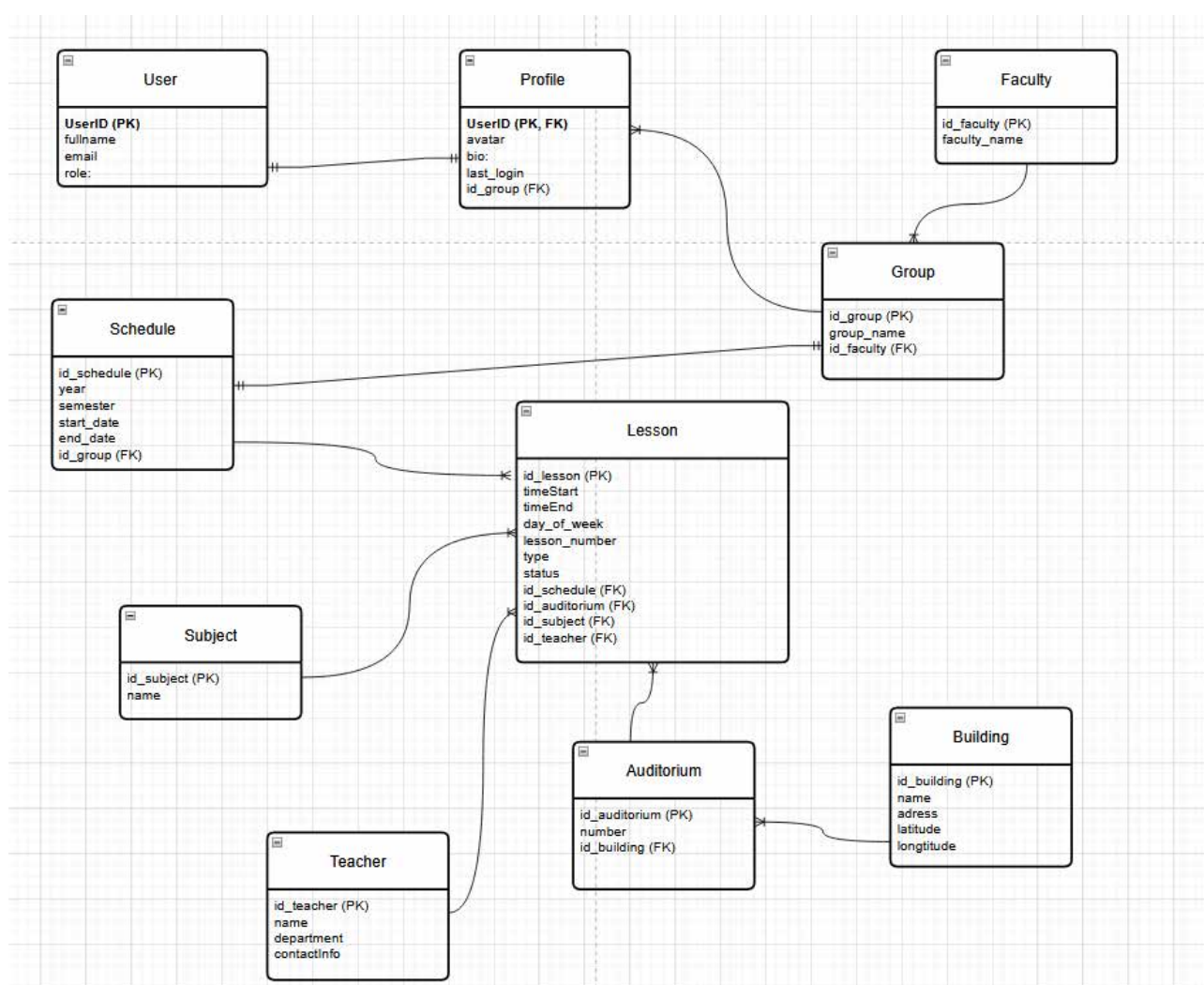


Рис. 2.1 – ER-діаграма підсистеми

2.3 Моделювання бізнес-процесів за допомогою UML-діаграм

На етапі проєктування підсистеми навчального процесу та просторової навігації важливо не обмежуватися лише вибором технологій. Потрібно заздалегідь описати поведінку додатка в типових ситуаціях: студент відкриває розклад перед парою, уточнює корпус після зміни аудиторії, шукає будівлю на карті або будує маршрут між об'єктами кампусу. Саме ці сценарії визначають практичну цінність системи.

Для формалізації логіки роботи додатка використано уніфіковану мову моделювання UML. Вона дає змогу подати систему не як набір окремих екранів, а як послідовну модель взаємодії користувача, мобільного інтерфейсу, локального кешу, хмарної бази даних і зовнішніх сервісів. Це особливо важливо для мобільного середовища, де розклад, карта кампусу, авторизація, профіль користувача та синхронізація даних мають працювати узгоджено.

У межах проєктування було підготовлено кілька типів UML-діаграм. Діаграма прецедентів визначає функціональні межі системи та ролі користувачів. Студент взаємодіє з модулями перегляду розкладу, профілю, списку корпусів, інтерактивної мапи й побудови маршруту. Адміністратор отримує доступ до функцій керування даними, зокрема оновлення навчальної інформації та підтримки довідників у базі даних.

Діаграми діяльності описують логіку виконання окремих процесів. Для модуля розкладу така модель показує вибір факультету й групи, перевірку доступності мережі, звернення до сервера або завантаження даних із локального кешу. Для навігаційного модуля вона фіксує послідовність дій від вибору будівлі до побудови маршруту або виведення повідомлення про відсутність доступу до геолокації.

Діаграма послідовностей деталізує часовий обмін повідомленнями між основними компонентами системи. У ній відображено взаємодію студента з мобільним інтерфейсом, контролером розкладу, локальним сховищем і Supabase API. Такий підхід дозволяє перевірити, чи коректно працює логіка Offline-First:

додаток спочатку використовує доступні локальні дані, а мережеве з'єднання застосовує для оновлення інформації.

Фізичну структуру системи подано за допомогою діаграми розгортання. Вона демонструє розміщення React Native-додатка на мобільному пристрої, взаємодію з хмарною платформою Supabase, базою PostgreSQL, зовнішнім сервісом авторизації, API розкладу НУБіП і картографічним провайдером. Завдяки цьому можна простежити, де саме зберігаються дані, через які канали вони передаються та які компоненти відповідають за обробку запитів.

Комплексне використання UML-діаграм допомогло узгодити поведінкову, структурну й фізичну моделі системи. У результаті навчальний модуль, карта кампусу, профіль користувача, механізм кешування та серверна синхронізація розглядаються як частини єдиного інформаційного середовища. Це забезпечує основу для подальшої реалізації мобільного додатка, орієнтованого на швидкий доступ студентів НУБіП України до навчальної та навігаційної інформації.

2.3.1 Діаграма прецедентів

На етапі проектування логіки додатка потрібно чітко визначити, які дії доступні кожній категорії користувачів. Без цього система швидко втрачає керованість: студент може бачити зайві адміністративні екрани, а адміністратор працювати з даними без зрозумілої межі відповідальності. Для візуалізації цих вимог побудовано діаграму прецедентів (Рис. 2.2).

У системі передбачено два основні актори – студент і адміністратор. Студент виступає кінцевим користувачем сервісів. Його сценарій максимально прикладний: відкрити додаток, подивитися, коли починається заняття, знайти потрібний корпус або гуртожиток і за потреби побудувати маршрут. Адміністратор працює з тими самими базовими екранами, але його головна роль інша – підтримувати актуальність даних у системі.

Точкою входу для обох акторів є прецедент «Вхід в систему». Це не формальність, а базовий механізм контролю доступу. Після авторизації додаток

визначає роль користувача й відкриває відповідний набір функцій. Студент отримує навчальні та навігаційні сервіси. Адміністратор – інструменти для роботи з базою даних.

Робота з розкладом має чітку послідовність. Прецедент «Перегляд навчального розкладу» пов'язано відношенням «include» з дією «Обрати групу / факультет». Це означає, що система не може коректно сформувати графік занять без вибору академічного підрозділу. Спочатку користувач вказує факультет і групу. Потім додаток показує розклад.

Для стабільної роботи в умовах слабкого інтернету до перегляду розкладу додано прецедент «Кешування даних». У діаграмі він виступає як розширення через відношення «extend». Ця дія не вимагає окремого натискання кнопки з боку користувача. Вона виконується у фоновому режимі: додаток зберігає локальну копію розкладу в пам'яті смартфона, щоб студент міг відкрити його навіть без доступу до мережі. Це дрібна дія для системи. Але важлива для користувача.

Навігаційна частина побудована як набір окремих, але пов'язаних сценаріїв. Користувач може переглянути список корпусів і гуртожитків, відкрити мапу кампусу або перейти до побудови маршруту до вибраного об'єкта. Такий поділ робить інтерфейс гнучкішим: студент не зобов'язаний одразу працювати з картою, якщо йому достатньо списку. А коли потрібно знайти шлях до корпусу, система відкриває маршрут.

Окремий прецедент – «Перегляд профілю користувача». Він забезпечує доступ до персональних даних, ролі, контактної інформації або інших параметрів облікового запису. У межах цієї моделі профіль виконує не лише довідкову функцію. Він також пов'язаний із персоналізацією доступу та перевіркою ролі користувача.

Адміністративний блок відокремлено від студентських сценаріїв. Адміністратор має доступ до прецедентів «Керування будівлями», «Внесення змін в БД» та «Синхронізація з БД». Через ці дії він може оновлювати інформацію про корпуси й гуртожитки, коригувати геолокаційні дані кампусу, запускати синхронізацію розкладу та підтримувати узгодженість записів у

Supabase. Тут помилка впливає не на одного користувача. Вона впливає на всю систему.

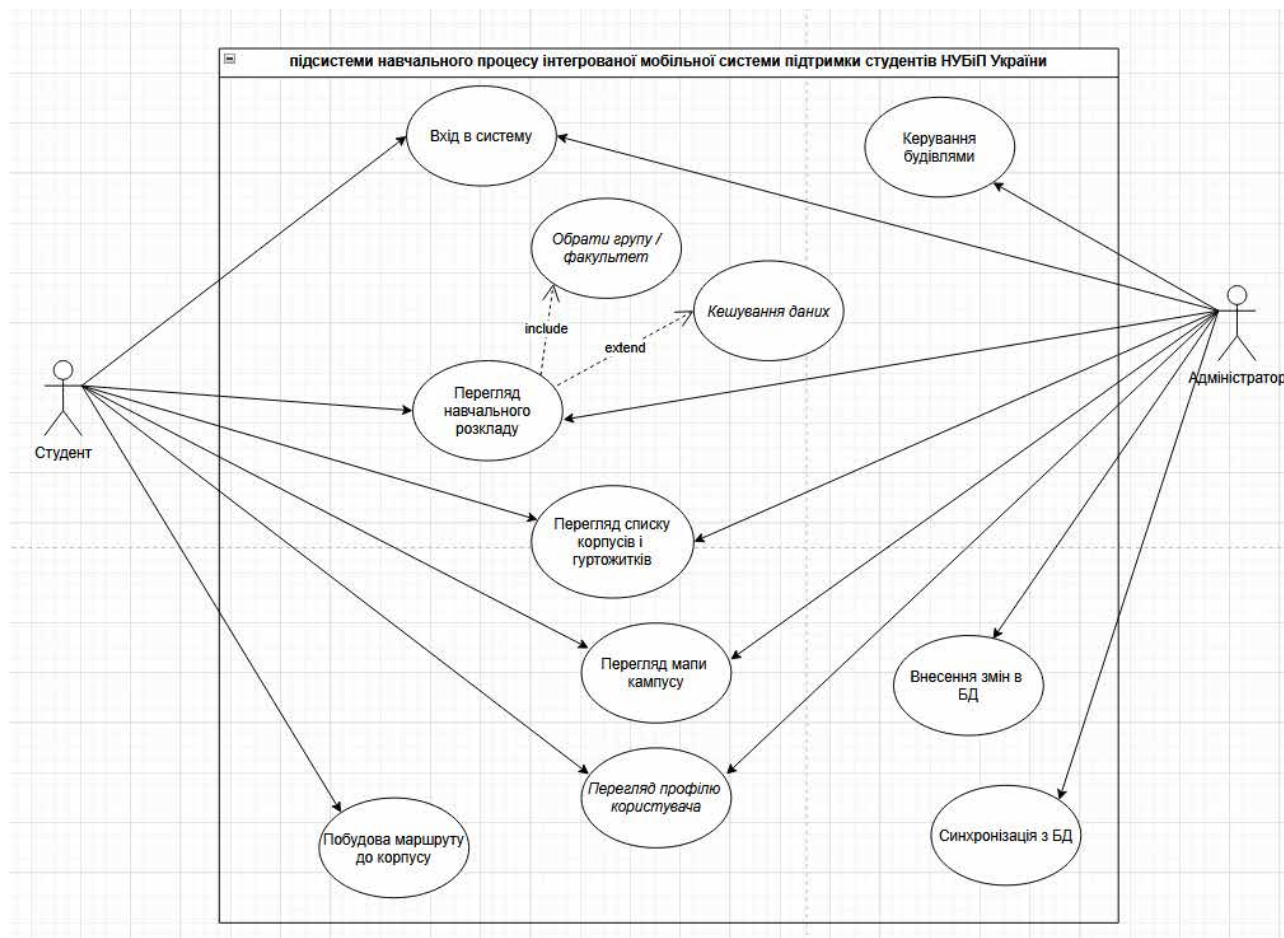


Рис. 2.2 – Діаграма прецедентів підсистеми навчання

Створена структура діаграми чітко розділяє два типи процесів: щоденне використання додатка студентом і технічне адміністрування даних. Студент працює з розкладом, профілем, мапою та маршрутами. Адміністратор забезпечує актуальність інформації, з якою ці модулі працюють.

2.3.2 Діаграма діяльності

Діаграма діяльності (Рис. 2.3, Рис. 2.4) використовується для детального опису послідовності дій, які виконує користувач і система під час роботи з основними функціями мобільного додатка. Якщо діаграма прецедентів показує загальні можливості студентів та адміністратора, то діаграма діяльності розкриває сам перебіг процесу: з якої дії починається сценарій, які умови

перевіряє система, коли відбувається розгалуження та який результат отримує користувач.

У межах проєкту побудовано дві діаграми діяльності, які описують найбільш важливі сценарії студентської частини мобільної інформаційної системи: перегляд навчального розкладу та побудову маршруту до об'єкта кампусу. Обидва процеси мають прикладний характер. Студент відкриває додаток не заради самої системи, а щоб отримати відповідь: яка пара наступна або як дійти до потрібного корпусу.

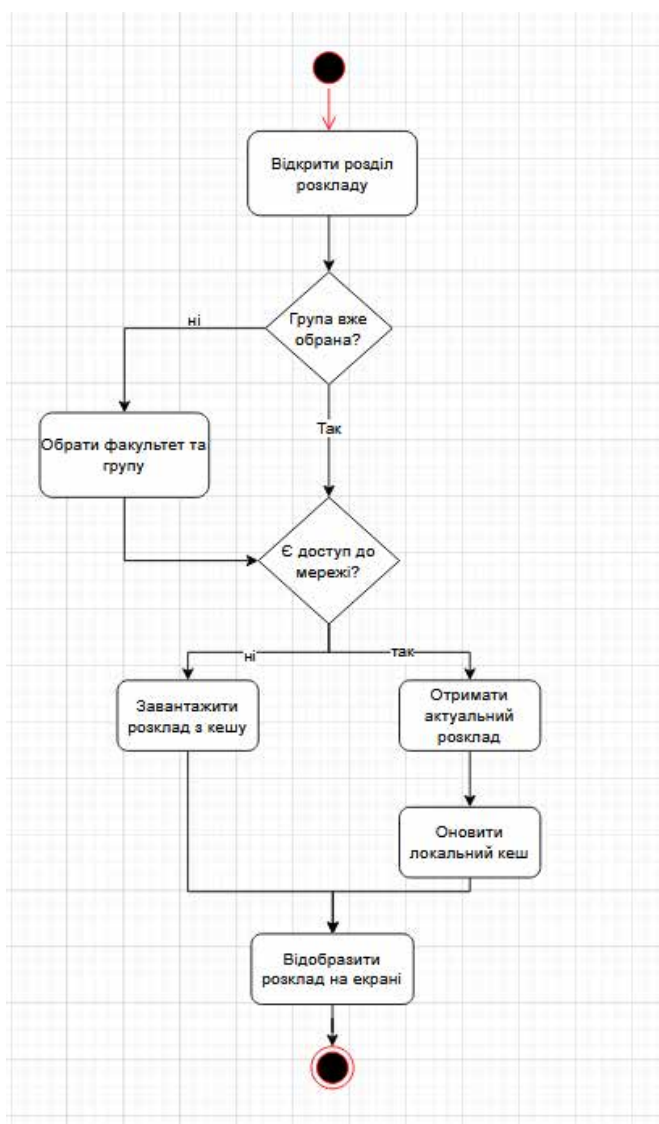


Рис. 2.3 – Діаграма діяльності перегляду розкладу

Перша діаграма описує процес перегляду навчального розкладу. Сценарій починається з відкриття розділу розкладу. Після цього система перевіряє, чи вже обрана академічна група. Якщо групу не вибрано, користувач переходить до

вибору факультету та групи. Це необхідний крок, оскільки без прив'язки до конкретного навчального підрозділу додаток не може коректно сформувати розклад занять.

Після вибору групи система переходить до наступної перевірки – наявності доступу до мережі. Саме тут проявляється логіка Offline-First. Якщо інтернет-з'єднання доступне, додаток отримує актуальний розклад із віддаленого джерела, після чого оновлює локальний кеш. Якщо мережа відсутня, система завантажує дані з кешу.

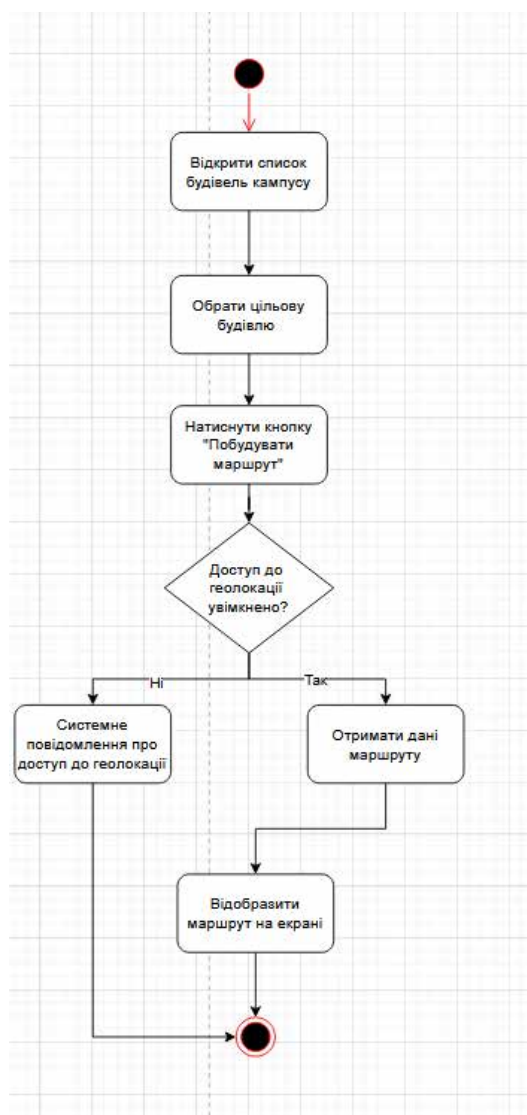


Рис. 2.4 – Діаграма діяльності побудови маршруту

Друга діаграма діяльності описує процес побудови маршруту до об'єкта кампусу. Сценарій починається з відкриття списку будівель кампусу. Користувач

обирає цільову будівлю: навчальний корпус, гуртожиток або інший об'єкт, доступний у системі. Після цього він натискає кнопку «Побудувати маршрут».

На наступному етапі система перевіряє, чи увімкнено доступ до геолокації. Якщо дозвіл на використання геолокації надано, додаток отримує дані маршруту та відображає його на екрані. Для студента це означає простий сценарій: вибрав корпус – отримав шлях. Без ручного введення адреси.

Якщо доступ до геолокації не увімкнено, система не завершує роботу помилкою. Вона показує повідомлення про необхідність надати доступ до геолокації. Такий варіант поведінки важливий, оскільки користувач повинен розуміти, чому маршрут не будується. Не просто порожній екран. А зрозуміле пояснення.

Обидві діаграми демонструють, що логіка мобільного додатка побудована навколо реальних умов використання. Для розкладу враховано ситуацію з відсутністю інтернету та необхідністю кешування. Для навігації передбачено перевірку геолокації перед побудовою маршруту. Це дає змогу уникнути критичних збоїв у типових студентських сценаріях.

Розроблені діаграми діяльності також показують, що система не виконує зайвих дій. Кожен процес має чіткий початок, логічні перевірки, альтернативні гілки та завершення. У випадку розкладу результатом є відображення навчального графіка на екрані. У випадку навігації – побудований маршрут або повідомлення про необхідність увімкнути геолокацію. Така структура спрощує подальшу реалізацію програмної логіки й робить поведінку додатка передбачуваною для користувача.

2.3.3 Діаграма послідовностей

Діаграма послідовностей (рис. 2.5) описує не загальний набір можливостей системи, а порядок взаємодії між її компонентами в конкретному сценарії. У цьому випадку модель показує, як мобільний додаток отримує розклад занять після вибору академічної групи. Тут важлива не лише сама відповідь від сервера.

Важливо, що робить система, якщо мережа доступна, і як вона поводить себе, коли зв'язок зникає.

У діаграмі виділено п'ять учасників взаємодії: студент, мобільний інтерфейс на React Native, контролер розкладу, локальний кеш і Supabase API. Студент ініціює процес через дію `selectGroup(id)`, тобто обирає потрібну групу в інтерфейсі додатка. Після цього мобільний UI передає запит до контролера розкладу через метод `fetchSchedule(groupId)`. Саме контролер бере на себе основну логіку. Інтерфейс не повинен сам вирішувати, звідки брати дані.

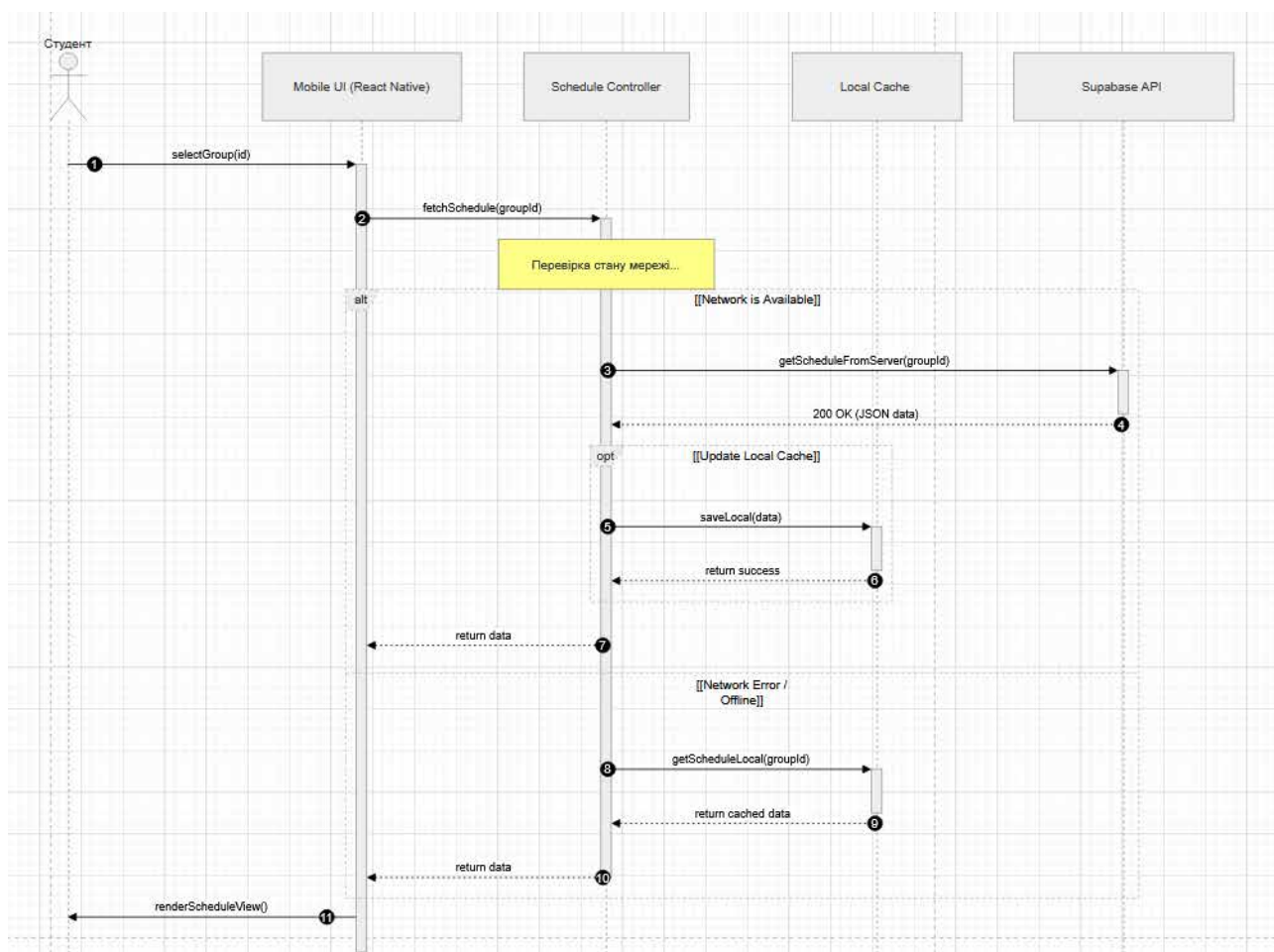


Рис. 2.5 – Діаграма послідовності отримання розкладу занять

Першим кроком після отримання запиту контролер перевіряє стан мережі. Це принциповий момент для Offline-First підходу. Якщо інтернет доступний, система переходить до гілки `[Network is Available]` і надсилає запит `getScheduleFromServer(groupId)` до Supabase API. Сервер повертає відповідь 200

OK у форматі JSON. Після цього контролер отримує актуальний розклад і може передати його назад в інтерфейс.

Але на цьому сценарій не завершується. Якщо система отримала нові дані з сервера, вона має зберегти їх локально. Для цього на діаграмі використано опціональний блок [Update Local Cache]. Контролер викликає операцію `saveLocal(data)` у локальному кеші, після чого отримує підтвердження успішного збереження. Це невелика дія. Але саме вона забезпечує роботу додатка наступного разу без інтернету.

Після оновлення кешу контролер повертає дані мобільному інтерфейсу. Інтерфейс, у свою чергу, викликає `renderScheduleView()` і відображає розклад на екрані смартфона. Для студента цей процес виглядає просто: він обрав групу – і побачив заняття. Усередині системи в цей момент відбувається перевірка мережі, звернення до сервера, оновлення локального сховища та підготовка даних до показу.

Друга гілка діаграми описує ситуацію [Network Error / Offline]. Вона спрацьовує тоді, коли система не може отримати дані з Supabase API або мережа повністю відсутня. У такому випадку контролер не завершує роботу помилкою. Він звертається до локального кешу через метод `getScheduleLocal(groupId)` і отримує збережений раніше розклад. Це практичний сценарій: студент може стояти в коридорі корпусу, де зв'язок нестабільний, але все одно має побачити свій навчальний день.

Після отримання кешованих даних контролер повертає їх мобільному інтерфейсу. Далі інтерфейс так само виконує `renderScheduleView()` і показує розклад користувачу. Різниця лише в джерелі даних: за наявності інтернету система бере актуальний розклад із Supabase, а за його відсутності – останню локально збережену копію. Для користувача логіка залишається однаковою. Він бачить результат.

Така структура взаємодії зменшує залежність додатка від якості мережі. Мобільний інтерфейс відповідає за введення та відображення даних, контролер

розкладу – за вибір джерела інформації, Supabase API – за актуальні серверні дані, а локальний кеш – за резервний доступ.

Розроблена діаграма послідовностей показує, що отримання розкладу в системі не є простим одноразовим запитом до сервера. Це керований процес із двома сценаріями: онлайн-отриманням актуальних даних і офлайн-відновленням із кешу. Саме така логіка відповідає реальним умовам використання мобільного додатка студентами НУБіП України, коли доступ до навчальної інформації має зберігатися навіть у разі тимчасових проблем зі зв'язком.

2.3.4 Діаграма розгортання

Моделювання фізичної топології мобільної інформаційної системи дає змогу визначити, де розміщуються основні програмні компоненти та через які канали вони взаємодіють між собою. На відміну від логічного проєктування, яке описує сценарії користувача, діаграма розгортання фіксує інфраструктурний рівень системи: мобільний пристрій студента, хмарну платформу Supabase, сервер бази даних PostgreSQL, зовнішній сервіс авторизації, API розкладу та картографічний провайдер.

На стороні користувача основним вузлом є мобільний пристрій студента під керуванням Android або iOS. На ньому функціонує клієнтський додаток, розроблений за допомогою React Native. Цей компонент відповідає за відображення розкладу занять, списку корпусів, інтерактивної карти кампусу та навігаційних елементів. У середині клієнтської частини також використовується локальне сховище AsyncStorage/Cache, яке зберігає копію навчальних даних і підтримує роботу додатка в умовах нестабільного інтернет-з'єднання.

Така побудова має практичне значення для студентів НУБіП України. Користувач може перевіряти розклад у коридорі корпусу, під час переходу між будівлями або в зоні зі слабким мобільним сигналом. У таких ситуаціях додаток не повинен втрачати працездатність. Якщо з'єднання із сервером тимчасово

відсутнє, клієнтська частина звертається до локально збереженої копії розкладу у форматі JSON і відображає останні доступні дані.

Центральним серверним середовищем системи виступає Supabase Cloud Platform. Вона виконує роль хмарного бекенду й об'єднує кілька функціональних компонентів: Auth Service, Supabase API Layer та Realtime Service. Взаємодія між мобільним застосунком і платформою Supabase здійснюється через захищений протокол HTTPS із використанням Supabase SDK. Завдяки цьому мобільний клієнт не підключається напряду до бази даних, а працює через контрольований API-рівень.

Компонент Auth Service відповідає за автентифікацію користувачів і керування сесіями. У межах розробленої системи вхід реалізовано через зовнішнього OAuth-провайдера, налаштованого для роботи з корпоративними поштовими обліковими записами домену @nubip.edu.ua. Після успішної автентифікації система отримує токен доступу, перевіряє обліковий запис користувача та визначає його роль. Саме на цьому етапі відбувається розмежування між звичайним студентом і користувачем з адміністративними правами.

Окремий шар хмарної платформи становить Supabase API Layer. Він містить доступ до Database API, таблиць PostgreSQL, механізмів Auth, політик Row Level Security та контрольованого доступу до реляційних таблиць. Усі запити від мобільного додатка проходять через цей проміжний рівень. Це знижує ризик несанкціонованого доступу до даних і дозволяє централізовано керувати правами користувачів.

Безпека даних забезпечується політиками Row Level Security. Вони визначають, які операції дозволені користувачу відповідно до його ролі. Студентські облікові записи отримують доступ переважно до читання навчальних і навігаційних даних. Операції створення, редагування та видалення записів залишаються доступними лише адміністративним користувачам. Такий підхід захищає таблиці розкладу, профілів, корпусів та інших довідників від випадкового або несанкціонованого втручання.

Фізичне збереження структурованих даних виконується на сервері бази даних PostgreSQL. У межах DB Schema розміщуються таблиці, процедури та тригери, необхідні для роботи системи. База зберігає інформацію про користувачів, профілі, факультети, академічні групи, розклад занять, дисципліни, викладачів, аудиторії та корпуси. Зв'язок між Supabase API Layer і PostgreSQL підтримується через внутрішнє SQL-з'єднання, що відображено на діаграмі як Internal SQL Connection.

Компонент Realtime Service передбачений у структурі Supabase Cloud Platform для оперативної реакції клієнтської частини на зміни в базі даних. Його можна використовувати для оновлення інтерфейсу без ручного перезавантаження екрана, наприклад після зміни розкладу або коригування даних про корпус. У цій архітектурі він виступає додатковим засобом підтримки актуальності даних, тоді як базова стабільність роботи забезпечується кешуванням на пристрої.

Картографічний функціонал системи винесено до зовнішнього сервісу Map Tiles Provider. Як джерело картографічної підкладки можуть використовуватися Google Maps або OpenStreetMap. Мобільний додаток звертається до цього сервісу через HTTPS/SDK і отримує необхідні тайли карти, не зберігаючи великих обсягів географічних даних на власному сервері. Водночас інформацію про конкретні об'єкти кампусу – корпуси, гуртожитки та їх координати – додаток отримує з бази PostgreSQL через Supabase.

Ще одним зовнішнім компонентом є NULES Schedule API. Через нього система отримує навчальні дані у форматі JSON. Ці відомості не використовуються безпосередньо в сирому вигляді: вони проходять обробку, узгоджуються з внутрішньою структурою бази даних і записуються до відповідних таблиць PostgreSQL. Завдяки цьому клієнтський додаток працює не з хаотичним зовнішнім потоком, а з уже нормалізованими даними.

Адміністративний контур представлений сервісом Supabase Dashboard / Admin Sync. Він використовується для керування службовими даними, запуску синхронізації, перевірки записів і внесення коригувань у довідники. Через цей

інструментарій адміністратор може оновлювати інформацію про розклад, корпуси, аудиторії та інші структурні елементи системи. Доступ до такого функціоналу обмежується ролями та політиками безпеки.

Розроблена фізична модель чітко розподіляє відповідальність між компонентами. Мобільний додаток забезпечує взаємодію зі студентом і локальне кешування даних. Supabase Cloud Platform виконує функції автентифікації, API-доступу та контролю прав. PostgreSQL зберігає основну структуру навчальної й навігаційної інформації. Зовнішні сервіси забезпечують авторизацію, імпорту розкладу та картографічну основу. Така топологія відповідає вимогам мобільної системи, яка має працювати стабільно навіть за умов нестабільного мережевого з'єднання.

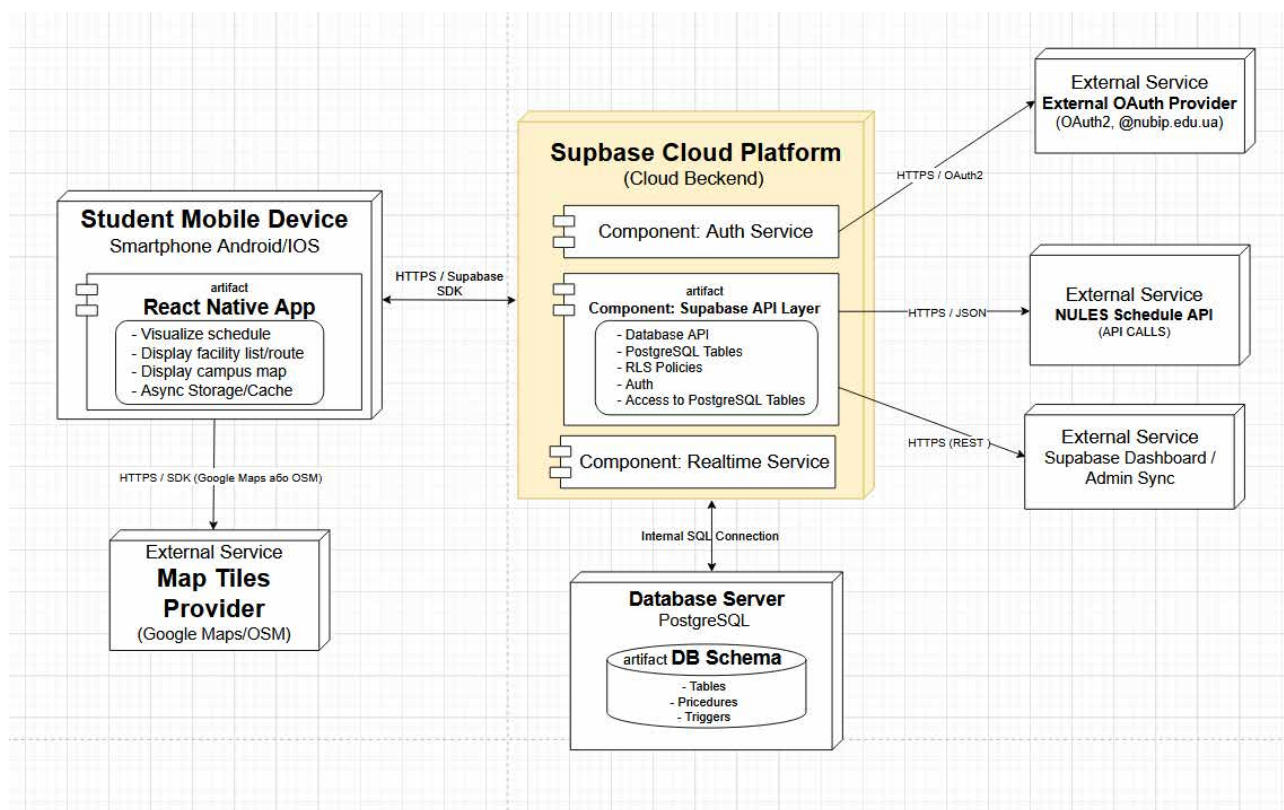


Рис. 2.6 – Діаграма розгортання системи

2.4 Ролі користувачів і політика доступу

У мобільній інформаційній системі доступ до функцій не може бути однаковим для всіх користувачів. Студенту потрібен швидкий перегляд

розкладу, мапи кампусу, маршруту й профілю. Адміністратору потрібні інші інструменти: оновлення даних, керування будівлями, контроль синхронізації та робота з записами в базі. Якщо ці рівні змішати, система стане вразливою.

У межах проєкту передбачено дві основні ролі: **студент** і **адміністратор**. Роль студента є базовою для більшості користувачів додатка. Після авторизації студент отримує доступ до перегляду навчального розкладу, мапи кампусу, списку корпусів і гуртожитків, маршруту до вибраного об'єкта та власного профілю. Дії здобувача освіти не змінюють критичні дані системи.

userid uuid	fullname varchar	email varchar	role user_role
2a06aa5a-df71-4dc6-91db-736e8de84b3b	Владислав Русланович Глуценко	kn22-v.hluschenko@nubip.edu.ua	student
b3d37154-7043-46d6-b433-88d6b31e02c	Микита Олексійович Скрипник	kn22-m.skrypnyk@nubip.edu.ua	admin
f9f16081-3d02-4430-865b-447f8cdaa029	Дмитро Олександрович Сафончик	kn22-d.safonchyk@nubip.edu.ua	student

Рис. 2.7 – Відображення таблиці користувачів із визначенням ролей у Supabase

Адміністратор має розширені права. Він може працювати з даними, від яких залежить коректність усього додатка: будівлями кампусу, навчальним розкладом, синхронізацією та записами в базі даних. Це вже не звичайний перегляд. Будь-яка помилка в адміністративному блоці може призвести до того, що студенти побачать неправильний корпус, застарілий розклад або некоректний маршрут.

Розмежування ролей починається з авторизації. Користувач входить у систему через обліковий запис, пов'язаний з університетською поштою домену @nubip.edu.ua. Після успішного входу система отримує дані профілю та перевіряє роль користувача. Саме на цьому етапі додаток визначає, які екрани й дії потрібно відкрити. Студент бачить навчальні та навігаційні сервіси. Адміністратор додатково отримує доступ до керування даними.

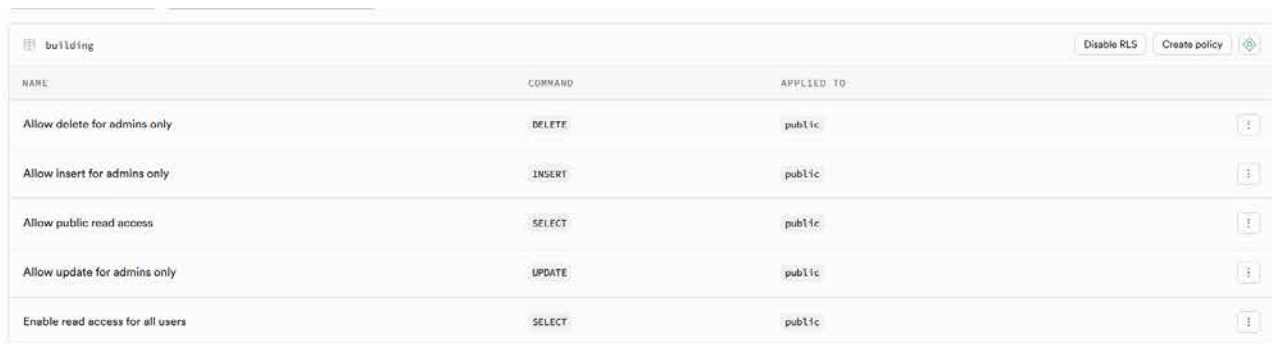
Для контролю доступу використовується зв'язка Supabase Auth і політик Row Level Security (RLS). Supabase Auth відповідає за ідентифікацію користувача, а RLS обмежує доступ до рядків таблиць у PostgreSQL відповідно до ролі. Такий підхід дозволяє переносити частину перевірок не лише на рівень

інтерфейсу, а й на рівень бази даних. Це важливо. Приховати кнопку в додатку недостатньо.

Наприклад, студент може отримувати записи розкладу, переглядати список будівель і читати власний профіль. Але він не повинен змінювати координати корпусів, запускати синхронізацію або редагувати навчальні дані. Адміністратор, навпаки, має доступ до операцій оновлення, додавання та коригування службових записів. Ролі не перетинаються там, де йдеться про цілісність даних.

На рівні інтерфейсу політика доступу проявляється у відображенні різних екранів і дій для різних ролей. Адміністратор, після перевірки ролі, отримує додаткові інструменти для керування даними.

Такий підхід формує подвійний рівень захисту. Перший рівень працює в мобільному додатку: система показує користувачу лише дозволені екрани. Другий рівень працює в Supabase і PostgreSQL: RLS-політики не дозволяють виконати заборонені операції навіть у випадку прямого звернення до API.



The screenshot shows the Supabase RLS configuration page for a table named 'building'. It features a table with three columns: NAME, COMMAND, and APPLIED TO. There are five policies listed, each with a name, a command (DELETE, INSERT, SELECT, UPDATE, SELECT), and the role it applies to (public). Each policy has a 'Disable RLS' button and a 'Create policy' button. The policies are: 'Allow delete for admins only' (DELETE, public), 'Allow insert for admins only' (INSERT, public), 'Allow public read access' (SELECT, public), 'Allow update for admins only' (UPDATE, public), and 'Enable read access for all users' (SELECT, public).

NAME	COMMAND	APPLIED TO
Allow delete for admins only	DELETE	public
Allow insert for admins only	INSERT	public
Allow public read access	SELECT	public
Allow update for admins only	UPDATE	public
Enable read access for all users	SELECT	public

Рис. 2.8 – Налаштування політик RLS для обмеження доступу до даних

Розподіл ролей і політика доступу забезпечують керованість мобільної інформаційної системи. Студент отримує швидкий і безпечний доступ до навчальних та навігаційних сервісів. Адміністратор підтримує актуальність даних і контролює технічні процеси. Система не змішує ці сценарії, тому зберігає стабільність, зрозумілу структуру й захист від несанкціонованих змін.

2.5 Визначення ключових сценаріїв використання системи

Моделювання ключових сценаріїв використання дає змогу описати функціонування мобільної інформаційної системи в умовах реальної експлуатації, а не лише у вигляді формального переліку функцій. Для студентського додатка цей етап має практичне значення, оскільки користувач взаємодіє з інтерфейсом переважно в умовах обмеженого часу. Йому потрібно швидко отримати відповідь щодо часу заняття, місця його проведення та способу переходу до відповідного корпусу.

Першим базовим сценарієм є авторизація користувача. Вхід до системи виконується через університетську пошту домену @nubip.edu.ua із використанням Google OAuth 2.0. Після успішної автентифікації Supabase Auth обробляє результат входу, а система отримує дані профілю та роль користувача. Студент переходить до навчальних і навігаційних модулів. Адміністратор отримує доступ до інструментів керування даними. Попри коротку тривалість, цей етап визначає межі доступу користувача протягом усього сеансу роботи.

Найчастіший сценарій для студента – перегляд розкладу занять. Користувач обирає факультет і академічну групу, після чого додаток формує лінійний список пар із зазначенням часу, дисципліни, типу заняття, аудиторії та корпусу. Такий формат краще відповідає мобільному екрану, ніж велика таблична сітка. Студент може перевірити розклад безпосередньо перед заняттям, під час переходу між корпусами або в коридорі навчального приміщення.

Паралельно працює сценарій автономного доступу до розкладу. Якщо інтернет-з'єднання доступне, клієнтська частина отримує актуальні дані через Supabase API та оновлює локальний кеш AsyncStorage. Якщо зв'язок зникає, додаток відкриває останню збережену версію розкладу. Такий механізм зменшує ризик появи порожніх екранів і забезпечує безперервність роботи додатка. Це відповідає архітектурній логіці Offline-First та враховує реальні умови використання застосунку в корпусах університету.

Навігаційний контур системи реалізується через роботу з мапою кампусу та побудову маршруту. Сценарій взаємодії з інтерактивною мапою дає змогу побачити розташування навчальних корпусів і гуртожитків НУБіП України на картографічній основі. Особливо актуальною ця функція є для студентів перших курсів, які ще не мають достатнього досвіду орієнтування на території університетського кампусу.

Побудова маршруту продовжує навігаційний сценарій. Користувач обирає корпус або гуртожиток, після чого система звертається до геолокації смартфона й формує маршрут до вибраного об'єкта. Якщо дозвіл на доступ до місця розташування не надано, додаток не завершує роботу помилкою, а показує повідомлення з поясненням причини. Такий підхід забезпечує коректну обробку виняткових ситуацій у роботі навігаційного модуля.

Допоміжну роль виконує сценарій роботи з профілем. У ньому зберігаються дані користувача, зокрема роль, електронна пошта та інші службові відомості, необхідні для персоналізації роботи додатка. Якщо студент уже обрав факультет або групу, система може використовувати ці дані як фільтри для швидкого відображення розкладу. Це зменшує кількість повторюваних дій з боку користувача та підвищує зручність щоденного використання додатка.

Адміністративні сценарії відокремлені від інтерфейсу студентів і обмежуються політиками доступу на рівні Supabase та PostgreSQL. Адміністратор може запускати синхронізацію навчальних даних, перевіряти коректність записів розкладу та оновлювати інформацію про об'єкти кампусу. Ці дії мають підвищений рівень відповідальності, оскільки впливають на коректність даних для всіх користувачів системи.

Окремий адміністративний сценарій стосується керування будівлями. Адміністратор додає або редагує назви корпусів, гуртожитків і їхні географічні координати. Після внесення змін мобільний додаток отримує оновлені дані через Supabase API, тому навігаційна частина залишається узгодженою з актуальним станом бази. Це важливо для підтримки карти без зміни коду застосунку.

Розглянуті сценарії формують цілісну модель функціонування системи. Авторизація визначає права доступу, модуль навчання працює з розкладом і локальним кешем, просторова навігація спирається на координати корпусів і гуртожитків, а адміністративна частина підтримує ці дані в актуальному стані. У результаті мобільна інформаційна система працює не як набір окремих екранів, а як узгоджене середовище для щоденних потреб студентів НУБіП України.

2.6 Інтеграція зовнішніх сервісів

Проектування мобільної інформаційної системи для студентської навігації та роботи з розкладом не передбачає повністю ізольованої роботи додатка. Застосунок має авторизувати користувачів через університетську пошту, отримувати актуальні навчальні дані та відображати мапу кампусу. Для цього система взаємодіє із зовнішніми сервісами. Такий підхід є обґрунтованим інженерним рішенням, оскільки частина функцій реалізується на рівні мобільного клієнта, а частина передається сервісам із готовою інфраструктурою.

У межах розробленого проєкту інтеграція охоплює три основні напрями: автентифікацію користувачів через Google OAuth 2.0, отримання розкладу через університетський API та використання картографічної основи для мапи кампусу. Кожен із цих напрямів виконує окрему функцію. Авторизація підтверджує особу користувача. API надає навчальні дані. Картографічний сервіс забезпечує просторове відображення корпусів і гуртожитків.

Перший напрям пов'язаний із входом користувача до системи. Для автентифікації використовується Google OAuth 2.0, налаштований через Google Cloud Console. Студент входить у додаток за допомогою поштового облікового запису домену @nubip.edu.ua, що дає змогу пов'язати користувача з університетським середовищем. За такої схеми відсутня потреба у створенні окремої локальної бази паролів.

Після успішної OAuth-автентифікації Supabase Auth обробляє результат входу та передає системі дані, необхідні для подальшої роботи з профілем користувача. Додаток отримує можливість визначити роль облікового запису й сформувати відповідний інтерфейс. Студент бачить розклад, мапу, маршрут і профіль. Адміністратор отримує доступ до дій, пов'язаних із керуванням даними. Такий підхід забезпечує чітке розмежування прав доступу.

Другий напрям інтеграції стосується університетського API розкладу NULES Schedule API. Розклад не є статичною інформацією. Пари можуть переноситися, аудиторії змінюватися, групи оновлюватися, а окремі записи коригуватися протягом семестру. Через це зберігати навчальні дані лише у вигляді локальних файлів недоцільно. Система виконує HTTPS-запити до зовнішнього API та отримує дані у форматі JSON.

Отримані записи не використовуються в додатку без попередньої обробки. Їх потрібно привести до внутрішньої реляційної структури бази даних: зіставити з факультетами, академічними групами, викладачами, аудиторіями та корпусами. Після цього дані зберігаються в таблицях PostgreSQL у середовищі Supabase. Такий підхід зменшує кількість прямих звернень мобільного клієнта до зовнішнього API й дозволяє працювати з уже підготовленими даними.

Для користувача цей процес залишається майже непомітним. Студент відкриває розклад і бачить структурований перелік занять із зазначенням часу, дисципліни, аудиторії та корпусу. Усередині система вже виконала синхронізацію, обробку й збереження даних. Якщо інтернет-з'єднання відсутнє, додаток використовує останню збережену версію розкладу з локального сховища. Саме в цьому проявляється логіка Offline-First.

Третій напрям інтеграції пов'язаний із мапою кампусу. Для відображення просторової основи використовується картографічний сервіс, зокрема OpenStreetMap. Мобільний додаток не зберігає власні картографічні плиточки й не формує мапу з нуля, оскільки це потребувало б зайвих ресурсів для студентської інформаційної системи. Натомість бібліотека react-native-maps забезпечує відображення картографічної основи на екрані смартфона.

Поверх карти система показує власні маркери з координатами навчальних корпусів і гуртожитків НУБіП України. Ці координати зберігаються в базі даних, тому їх можна оновлювати без зміни коду мобільного додатка. Якщо в системі уточнюється назва або розташування об'єкта, адміністратор може внести зміни централізовано. Це спрощує подальше супроводження системи.

Навігаційний модуль також використовує API геолокації мобільного пристрою. Якщо користувач надає дозвіл, додаток визначає його поточне місце розташування та будує маршрут до вибраного корпусу або гуртожитку. Якщо доступ до геолокації відсутній, система не припиняє роботу аварійно, а виводить повідомлення про необхідність надати відповідний дозвіл. Така обробка підвищує зрозумілість взаємодії з додатком і зменшує ризик некоректного завершення навігаційного процесу.

Supabase у цій архітектурі виконує роль центрального бекенд-компонента. Він зберігає навчальні та навігаційні дані, обробляє автентифікацію, застосовує політики Row Level Security і надає мобільному додатку контрольований доступ до таблиць PostgreSQL. Завдяки цьому зовнішні сервіси інтегруються через визначені архітектурні точки взаємодії, а не підключаються безпосередньо в різних частинах інтерфейсу.

Інтеграція зовнішніх сервісів потребує врахування можливих збоїв. API розкладу може тимчасово не відповідати. Картографічні дані можуть завантажуватися повільніше за слабкого інтернет-з'єднання. OAuth-автентифікація залежить від доступності зовнішнього провайдера. Через це в системі використовується локальне кешування розкладу в AsyncStorage. Якщо оновлення недоступне, користувач усе одно може переглянути останні збережені дані.

У результаті зовнішні сервіси доповнюють власну логіку мобільної інформаційної системи, але не замінюють її повністю. Google OAuth 2.0 забезпечує вхід через університетську пошту, NULES Schedule API надає навчальні дані, OpenStreetMap формує картографічну основу, а Supabase поєднує ці компоненти з базою даних, ролями й політиками доступу.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Реалізація серверної частини мобільної інформаційної системи

Серверна частина мобільної інформаційної системи побудована на базі платформи Supabase, яка поєднує засоби автентифікації, керування доступом, роботу з реляційною базою даних PostgreSQL та API-рівень для взаємодії з клієнтським додатком. Такий підхід є доцільним архітектурним рішенням, оскільки дає змогу не розгортати окремий сервер із нуля, але водночас зберегти повноцінну серверну логіку для збереження, перевірки та передавання даних. Мобільний клієнт у цій моделі не містить усю бізнес-логіку всередині себе, а звертається до централізованого середовища, де зберігаються навчальні, користувацькі та навігаційні відомості.

Основою серверної частини є база даних PostgreSQL, розгорнута в середовищі Supabase. У ній зберігаються профілі користувачів, ролі, дані про факультети, академічні групи, розклад занять, навчальні корпуси, гуртожитки та координати об'єктів кампусу. Така структура відповідає предметній області, оскільки більшість сутностей мають чіткі реляційні зв'язки: факультет об'єднує групи, група пов'язана з розкладом, аудиторія належить до певного корпусу, а корпус має географічні координати. Саме ці зв'язки забезпечують коректну роботу як навчального, так і навігаційного модулів.

Серверну логіку системи доцільно поділити на кілька функціональних модулів. Перший модуль відповідає за автентифікацію користувачів. Вхід до системи здійснюється через університетську пошту домену @nubip.edu.ua із використанням Google OAuth 2.0, налаштованого через Google Cloud Console. Після успішної автентифікації Supabase Auth створює або підтверджує обліковий запис користувача, а система отримує можливість перевірити його роль. Цей етап формує первинний рівень контролю доступу.

Другий модуль пов'язаний із профілями користувачів. Він зберігає основні відомості про студента або адміністратора: ідентифікатор, електронну пошту, роль та службові параметри профілю. На основі цих даних додаток визначає, який інтерфейс потрібно відобразити після входу. Студент отримує доступ до розкладу, мапи кампусу, переліку корпусів і гуртожитків, маршрутизації та власного профілю. Адміністратор додатково отримує інструменти керування даними й синхронізацією.

Окремий модуль відповідає за навчальні дані. До нього належать таблиці та запити, пов'язані з факультетами, групами, дисциплінами, заняттями, аудиторіями, викладачами й часом проведення пар. Саме цей блок забезпечує роботу екрана розкладу. Коли мобільний додаток надсилає запит на отримання розкладу для конкретної академічної групи, серверна частина повертає структурований набір даних, придатний для безпосереднього відображення на екрані смартфона. Користувач бачить не технічні ідентифікатори, а зрозумілу картку заняття з назвою дисципліни, часом, аудиторією та корпусом.

Навігаційний модуль працює з даними про будівлі кампусу. У базі зберігаються назви корпусів і гуртожитків, їхні типи, адреси та географічні координати. Після запиту з боку клієнтського додатка серверна частина передає ці дані через Supabase API, а мобільний інтерфейс відображає відповідні маркери на мапі. Коли користувач обирає конкретний об'єкт, координати будівлі використовуються для побудови маршруту. Такий підхід дає змогу централізовано оновлювати інформацію про кампус без внесення змін до програмного коду мобільного додатка.

Важливим елементом серверної частини є синхронізація навчальних даних із зовнішнім API розкладу. Дані, отримані у форматі JSON, не можуть одразу використовуватися в інтерфейсі без попередньої обробки, оскільки їх потрібно привести до внутрішньої структури бази даних. На цьому етапі записи зіставляються з факультетами, групами, дисциплінами, викладачами, аудиторіями та корпусами. Після нормалізації вони зберігаються в PostgreSQL і стають доступними для клієнтського додатка через контрольований API-рівень.

Це зменшує кількість прямих звернень мобільного клієнта до зовнішнього джерела та підвищує стабільність роботи системи.

Адміністративний модуль забезпечує керування службовими даними. Через нього адміністратор може запускати синхронізацію, перевіряти коректність записів, оновлювати інформацію про будівлі та контролювати актуальність відомостей у базі даних. Доступ до цього модуля обмежується роллю користувача. Звичайний студент не має прав змінювати координати корпусів, редагувати розклад або впливати на службові записи системи.

Під маршрутами серверної частини в межах цієї роботи розуміються не окремі вручну реалізовані REST-endpoints власного сервера, а логічні маршрути доступу до даних через Supabase API. Мобільний додаток звертається до таблиць і представлень бази даних не напряму, а через контрольований API-рівень та Supabase SDK. Наприклад, для отримання профілю використовується запит до таблиці профілів, для розкладу – запит із фільтрацією за академічною групою, для мапи – запит до таблиці будівель. На рівні програмної реалізації це оформлюється як звернення через SDK, а на рівні архітектури – як набір логічних маршрутів доступу до відповідних даних.

Типовий маршрут отримання розкладу починається на мобільному клієнті. Користувач обирає академічну групу, після чого додаток формує запит до Supabase. Серверна частина перевіряє авторизацію, застосовує політики доступу та повертає відповідні записи розкладу. Якщо дані отримано успішно, клієнтська частина оновлює локальний кеш. У разі втрати інтернет-з'єднання додаток використовує останню збережену копію. Сервер у цьому сценарії відповідає за актуальне джерело даних, а локальний кеш забезпечує стійкість роботи клієнтської частини.

Маршрут роботи з профілем має іншу логіку. Після авторизації додаток звертається до профілю користувача, отримує роль і службові дані, а потім формує доступний інтерфейс. Саме на цьому етапі серверна частина впливає на поведінку клієнта. Відображення адміністративних функцій визначається не

лише станом кнопок або екранів у мобільному додатку, а роллю користувача, яка зберігається в базі даних.

Для навігаційного блоку маршрут доступу є простішим. Додаток запитує список будівель, отримує назви, типи об'єктів і координати, після чого відображає їх на мапі. Коли користувач обирає конкретний корпус або гуртожиток, ці координати використовуються для побудови маршруту через картографічний сервіс. Оскільки дані про будівлі зберігаються централізовано, адміністратор може оновлювати їх без зміни коду мобільного застосунку.

Політика доступу реалізується через Supabase Auth і Row Level Security. Supabase Auth підтверджує особу користувача, а RLS визначає, які записи та операції йому дозволені. Такий підхід забезпечує захист не лише на рівні інтерфейсу, а й безпосередньо на рівні бази даних. Операції читання доступні студентам у межах необхідного функціоналу, тоді як дії зі створення, редагування або видалення записів обмежуються адміністративними обліковими записами.

У результаті серверна частина виконує роль центрального ядра мобільної інформаційної системи. Вона забезпечує збереження даних, автентифікацію користувачів, розмежування прав доступу, приймання оновлень із зовнішнього API розкладу та передавання мобільному клієнту підготовлених структурованих записів. React Native відповідає за клієнтський інтерфейс і взаємодію з користувачем, тоді як Supabase та PostgreSQL забезпечують порядок, безпеку й актуальність даних. Саме такий розподіл функцій робить систему керованою, масштабованою та придатною для щоденного використання студентами НУБіП України.

3.2 Реалізація клієнтської частини мобільного додатка

Клієнтська частина мобільної інформаційної системи відповідає за той рівень, з яким студент працює безпосередньо.

Бекенд зберігає інформацію. Додаток робить її зручною для щоденного використання.

Після входу в систему користувач потрапляє на головний екран додатка (Рис. 3.1), де розміщено привітання, блок швидкого доступу та нижню навігаційну панель. Функції згруповано за двома напрямками: «Безпека» та «Навчання». У межах цієї роботи основна увага приділяється саме навчальному блоку, оскільки він містить розклад, кампусні об'єкти та навігацію. Такий поділ скорочує шлях користувача до потрібної функції.

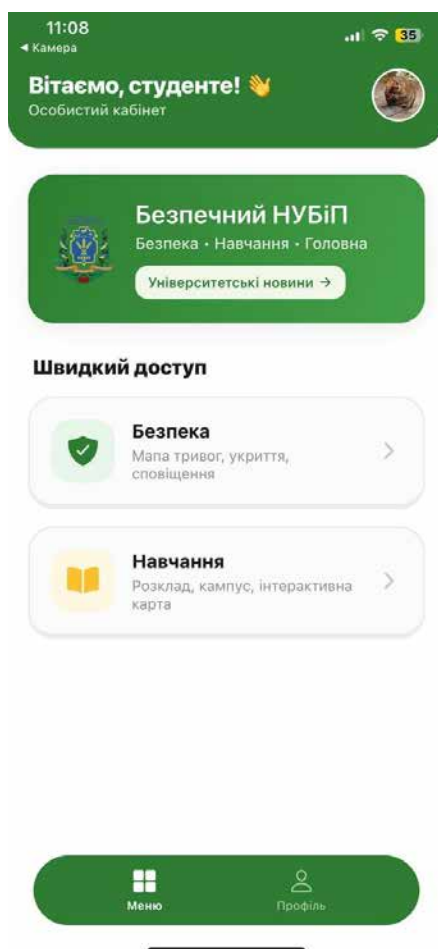


Рис. 3.1 – Головний екран інформаційної системи

Розділ «Навчання» об'єднує основні студентські сервіси, зокрема перегляд розкладу, список корпусів і гуртожитків та інтерактивну мапу (Рис. 3.2). Кожен пункт має окрему картку з піктограмою, назвою та коротким поясненням. Це спрощує сприйняття інтерфейсу, особливо для користувача, який відкриває додаток між парами й не має часу на довгу навігацію.

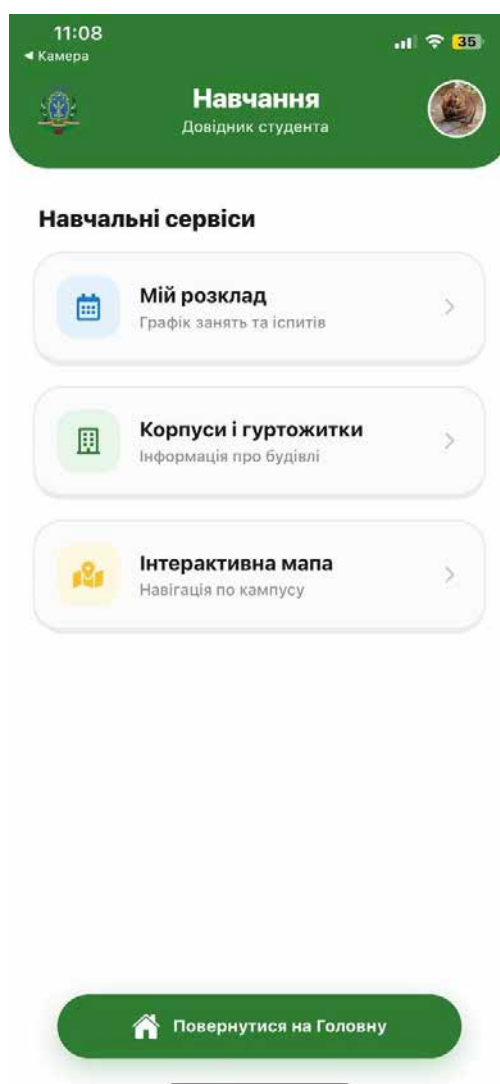


Рис. 3.2 – Екран розділу «Навчання»

Модуль розкладу реалізовано як окремий екран із вибором факультету, академічної групи та дня тижня (Рис. 3.3). Після вибору потрібних параметрів додаток відображає заняття у вигляді карток. Такий формат краще підходить для смартфона, ніж широка таблиця з багатьма колонками. Студент одразу бачить час пари, номер заняття, назву дисципліни, корпус та періодичність. На екрані також виведено джерело даних LIVE API, що показує використання актуального джерела розкладу.

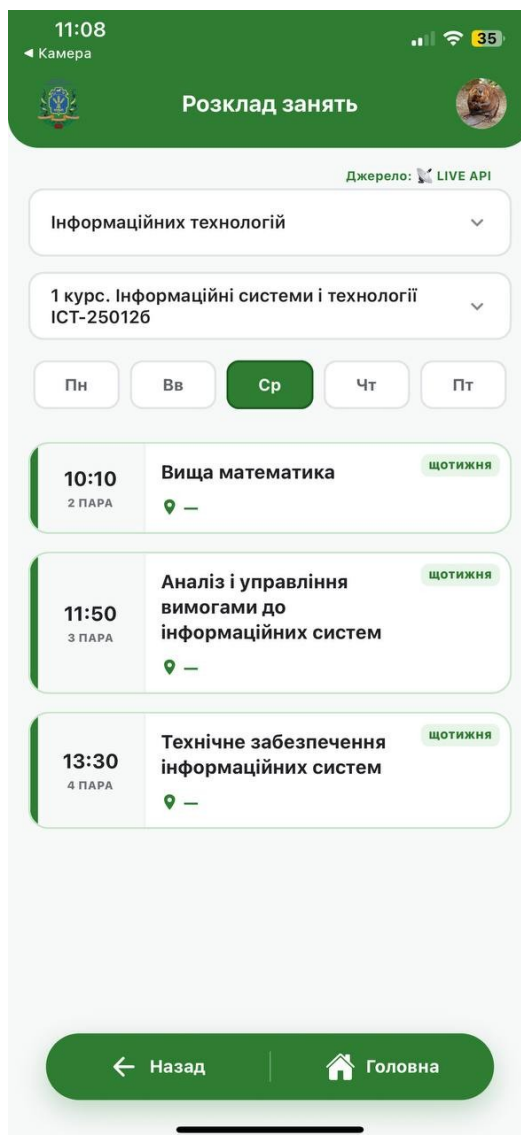


Рис. 3.3 – Екран перегляду розкладу занять

Навігаційний блок починається зі списку корпусів і гуртожитків НУБіП (Рис. 3.4). На цьому екрані користувач може перемикатися між типами об'єктів, виконувати пошук і застосовувати фільтри за територіальними зонами. Кожен корпус подано окремою карткою з назвою, адресою та кнопкою «Маршрут». Це практичний сценарій: студент бачить не лише номер корпусу, а й адресу, після чого може одразу перейти до побудови шляху.

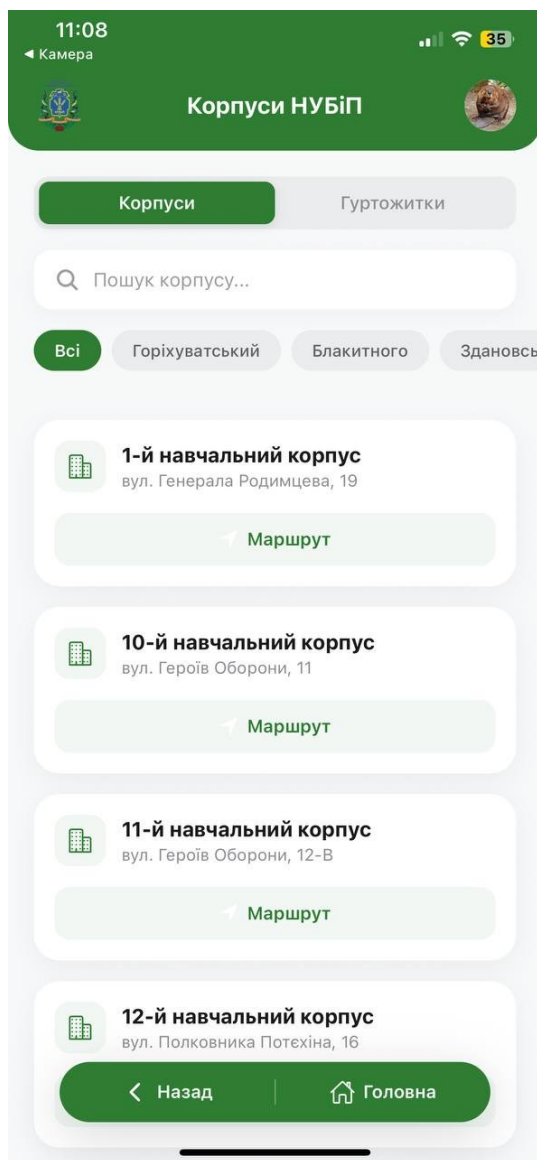


Рис. 3.4 – Екран перегляду корпусів і гуртожитків НУБіП

Інтерактивна мапа (Рис. 3.5) доповнює список будівель візуальним орієнтуванням. На ній відображаються маркери навчальних корпусів і гуртожитків, а після натискання на маркер система показує назву та адресу вибраного об'єкта. У нижній частині екрана передбачено легенду, яка розрізняє типи об'єктів. Це важливо для кампусу з великою кількістю будівель. Користувач швидко розуміє, де розташований потрібний корпус.

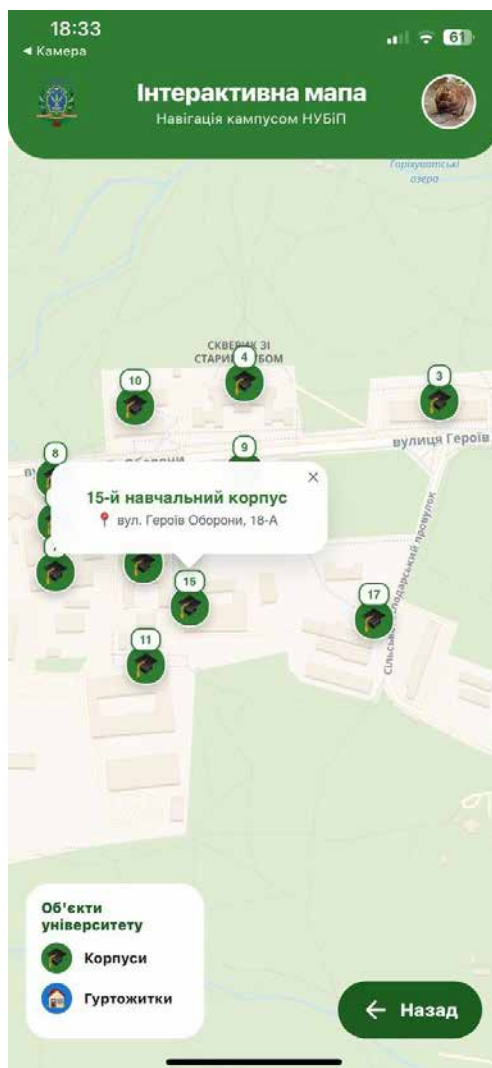


Рис. 3.5 – Екран інтерактивної мапи кампусу

Профіль користувача (Рис. 3.6) реалізує персональний розділ додатка, де відображаються прізвище та ім'я, короткий опис, корпоративна пошта та статус у системі. У цьому випадку статус «Студент НУБіП» підтверджує роль користувача й узгоджується з політикою доступу. Через цей екран також доступна дія виходу із системи. Це потрібна функція для мобільного застосунку з авторизацією через університетську пошту.

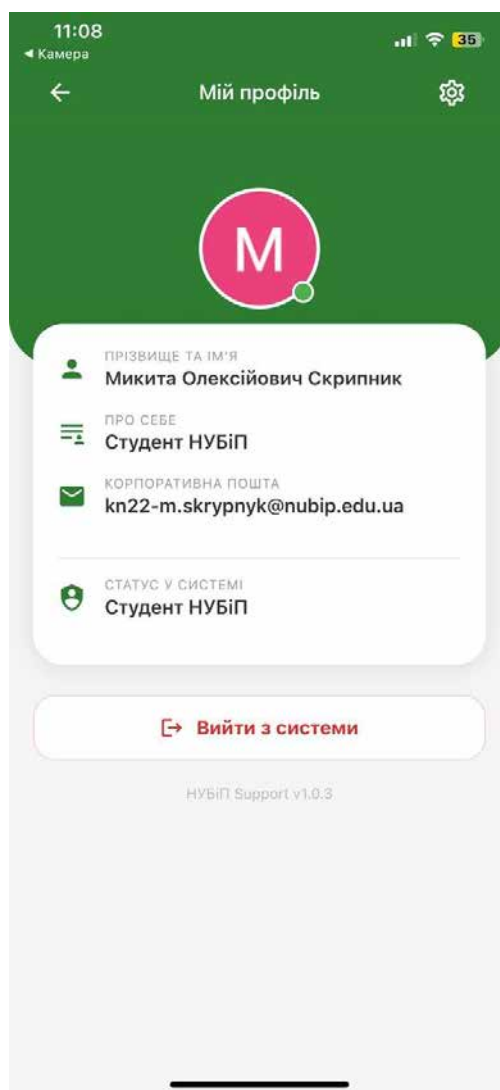


Рис. 3.6 – Екран профілю користувача

Для розробки використано React Native, що дало змогу реалізувати єдину клієнтську логіку для Android та iOS. Це важливо для студентської аудиторії, де користувачі мають різні типи пристроїв. Інтерфейс витримано в єдиній зеленій кольоровій схемі, яка відповідає візуальному стилю НУБіП.

У результаті клієнтська частина мобільного додатка поєднує основні навчальні та навігаційні функції в єдиному інтерфейсі. Студент може перейти від головного екрана до розкладу, знайти потрібний корпус, переглянути його на мапі, побудувати маршрут і перевірити власний профіль.

3.3 Реалізація модуля перегляду розкладу з використанням каскадної логіки отримання даних

Модуль перегляду розкладу занять є одним із основних компонентів навчальної частини мобільного додатка. Він забезпечує доступ студента до інформації про час проведення занять, дисципліни, аудиторії та особливості тижневого графіка. З огляду на те, що розклад належить до найбільш часто використовуваних даних у студентському середовищі, під час реалізації цього модуля особливу увагу приділено стабільності отримання інформації та можливості її резервного відображення.

У межах цього компонента об'єднано завантаження списку факультетів, вибір академічної групи, отримання розкладу, локальне кешування, резервне звернення до Supabase та відображення результату в мобільному інтерфейсі. Для побудови екрана використано компоненти React Native, зокрема View, Text, FlatList, Image, Pressable, TouchableOpacity, а також випадаючі списки з бібліотеки react-native-element-dropdown. Навігація між екранами виконується через expo-router, а для візуальних позначок застосовано іконки Ionicons.

Після відкриття модуля система перевіряє, чи були раніше збережені значення факультету та академічної групи. Для цього використовується локальне сховище AsyncStorage. Якщо такі дані наявні, додаток автоматично підставляє їх у відповідні поля інтерфейсу та ініціює завантаження розкладу. Це зменшує кількість повторних дій користувача та скорочує час доступу до навчальної інформації.

Завантаження факультетів і груп організовано з урахуванням можливих перебоїв мережевого з'єднання. Спочатку додаток звертається до відкритого API НУБіП та отримує актуальний перелік факультетів і відповідних академічних груп. У разі успішного виконання запиту ці дані форматуються для відображення у випадаючих списках і зберігаються в локальний кеш. Якщо API тимчасово недоступне, система використовує раніше збережену копію даних із пам'яті пристрою.

Основну логіку отримання розкладу реалізовано у функції `fetchSchedule()`. Вона працює за каскадною схемою, у якій передбачено кілька джерел даних. Першим джерелом виступає офіційний API розкладу НУБіП. Якщо запит успішний, отримані записи перетворюються у внутрішній формат додатка, відображаються на екрані та зберігаються в локальний кеш за ключем, прив'язаним до конкретного факультету й групи. Такий підхід унеможливорює змішування розкладів різних академічних груп.

У разі недоступності API система переходить до другого джерела – локального кешу. У цьому сценарії студент отримує останню збережену копію розкладу. Хоча такі дані можуть не відображати останні зміни, вони забезпечують базову інформаційну підтримку користувача за відсутності стабільного інтернет-з'єднання.

Якщо локальні дані також відсутні, додаток виконує резервне звернення до Supabase. У межах цього сценарію система шукає академічну групу за ідентифікатором API або за назвою, після чого намагається отримати пов'язані записи занять із таблиці Lesson. Цей механізм виконує роль додаткового рівня надійності та дозволяє уникнути повної відмови модуля за умови недоступності основного джерела.

Окремий етап становить перетворення даних, отриманих із зовнішнього API. Функція `transformApiData()` обробляє записи за днями тижня, визначає числовий ідентифікатор дня, зіставляє номер пари з часом початку та формує масив занять у форматі, придатному для подальшого відображення. Крім того, система виконує спробу відокремити назву дисципліни від аудиторії, якщо ці дані надходять в одному текстовому рядку. Це забезпечує більш структуроване подання інформації в інтерфейсі.

Відображення розкладу реалізовано у вигляді списку карток. Кожна картка містить час початку заняття, номер пари, назву дисципліни, аудиторію та позначку періодичності. Перед виведенням на екран записи фільтруються за вибраним днем тижня. Завдяки цьому користувач бачить не повний тижневий масив, а лише релевантну частину розкладу.

Якщо всі передбачені джерела даних недоступні, додаток переходить до стану повідомлення про помилку. У цьому випадку на екрані відображається інформація про неможливість завантаження розкладу та пропонується перейти до автономного бекапу, як показано на рис. 3.7.

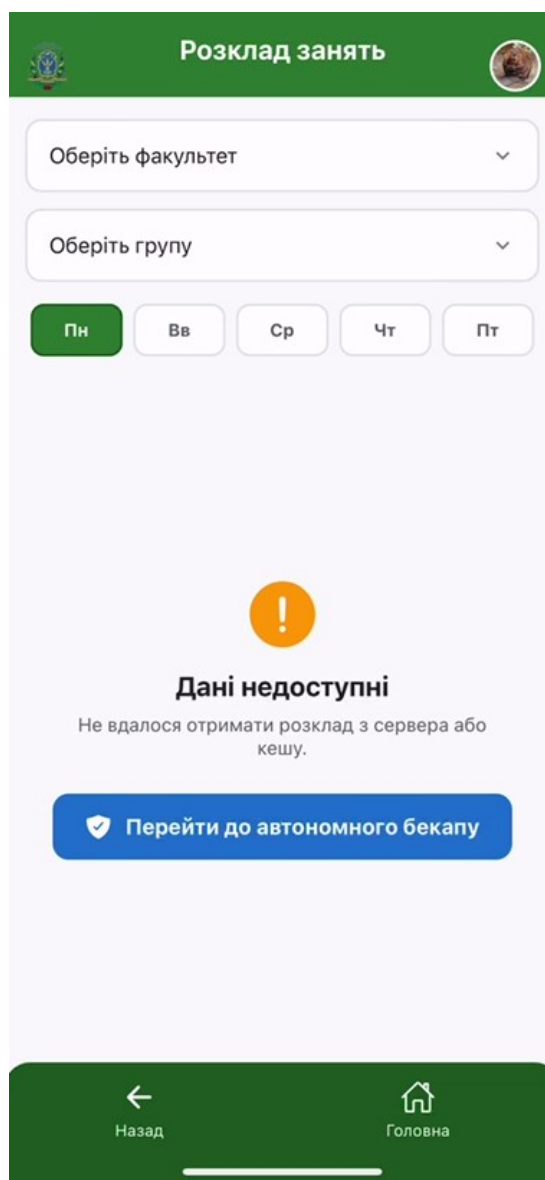


Рис. 3.7 – Повідомлення про недоступність розкладу та перехід до автономного бекапу

Відображення кнопки переходу до автономного бекапу керується станом `showBackupBtn`. Якщо основний API, локальний кеш і резервне джерело Supabase не повертають необхідних даних, система встановлює це значення в `true`. Після цього інтерфейс замість списку занять показує блок із повідомленням

і кнопкою переходу на екран `scheduleBackup`. Такий сценарій забезпечує користувачу альтернативний спосіб доступу до навчальної інформації.

У нижній частині екрана розкладу передбачено навігаційну панель із кнопками повернення назад і переходу на головний екран. Вона залишається доступною незалежно від поточного стану модуля: успішного відображення занять, завантаження даних або повідомлення про помилку. Це забезпечує передбачуваність навігації та не обмежує користувача межами одного екрана.

Реалізація модуля перегляду розкладу занять поєднує роботу з офіційним API НУБіП, локальне кешування через `AsyncStorage`, резервне звернення до `Supabase`, обробку отриманих даних і адаптоване мобільне відображення. Така структура підвищує стійкість модуля до збоїв зовнішніх джерел і забезпечує доступ до навчального розкладу в умовах нестабільного мережевого з'єднання.

3.4 Побудова API-взаємодії між frontend і backend

API-взаємодія між клієнтською частиною на `React Native` та серверною інфраструктурою `Supabase` визначає, як мобільний додаток отримує дані, перевіряє користувача та застосовує правила доступу. Frontend відповідає за інтерфейс, запуск запитів і відображення результатів. Backend зберігає дані, обробляє автентифікацію, перевіряє ролі та контролює доступ до таблиць `PostgreSQL`. Поділ цих функцій зменшує зв'язаність системи й полегшує її подальший супровід.

Для обміну даними використано `Supabase Client SDK`. Він виступає проміжним шаром між мобільним додатком і backend-рівнем, надаючи засоби для роботи з авторизацією, сесіями користувачів, таблицями бази даних і політиками доступу. Завдяки цьому клієнтська частина не формує складні HTTP-запити вручну, а працює з даними через стандартизований API-рівень.

Серверна частина побудована на `Supabase API Layer`, який забезпечує доступ до `PostgreSQL` у форматі `JSON`. Такий формат зручний для мобільного застосунку, оскільки легко перетворюється на внутрішні об'єкти інтерфейсу.

Через цей рівень додаток отримує профіль користувача, роль, навігаційні об'єкти кампусу, навчальні дані та службову інформацію для адміністративних сценаріїв.

Автентифікація користувачів реалізована через OAuth 2.0 із використанням Google Cloud Console. Вхід виконується через університетську пошту домену @nubir.edu.ua, що дозволяє не створювати окрему локальну систему паролів. Після успішного входу Supabase Auth формує сесію користувача, а мобільний додаток отримує дані, потрібні для подальшої перевірки ролі.

Після автентифікації система переходить до авторизації. У межах проєкту використано рольову модель доступу, де основними ролями є студент і адміністратор. Студент отримує доступ до профілю, розкладу, мапи кампусу, списку корпусів і гуртожитків та побудови маршруту. Адміністратор має розширені права для керування даними, синхронізації та оновлення службових записів.

Розмежування доступу реалізовано на двох рівнях. На рівні frontend додаток показує користувачу лише ті екрани й дії, які відповідають його ролі. На рівні backend застосовано політики Row Level Security у PostgreSQL. Same RLS забезпечує реальний контроль доступу до таблиць, оскільки перевіряє права користувача під час виконання запиту, а не лише приховує елементи інтерфейсу.

Навігаційні дані також передаються через API-рівень Supabase. У базі зберігаються корпуси, гуртожитки, адреси та координати, а frontend використовує ці записи для формування списку об'єктів і відображення маркерів на мапі. Картографічний сервіс відповідає за візуальну основу та маршрутну логіку, тоді як власна база містить саме університетські дані.

Для зменшення залежності від мережевого з'єднання клієнтська частина використовує локальне збереження частини даних. Уже отримані записи можуть повторно використовуватися без нового звернення до backend. Це підвищує

стабільність роботи додатка в умовах нестабільного інтернету та скорочує час доступу до часто використовуваної інформації.

У результаті API-взаємодія між frontend і backend побудована як керована багаторівнева модель. Вона поєднує Supabase Client SDK, PostgreSQL, OAuth-автентифікацію, RLS-політики, зовнішній API НУБІП, картографічний сервіс і локальне збереження даних. Така структура забезпечує контрольований обмін інформацією, чітке розмежування ролей і можливість подальшого розширення системи без прямого доступу клієнта до бази даних.

3.5 Розгортання та підготовка мобільного додатка до експлуатації

Розгортання мобільного додатка охоплює перехід від вихідного коду до інсталяційного пакета, придатного для запуску на пристроях користувачів. На відміну від вебзастосунків, мобільна система потребує компіляції у формат, який підтримує операційна система Android. Тому процес збирання має враховувати залежності, сертифікати, змінні середовища та вимоги до подальшого оновлення застосунку.

Для збирання додатка доцільно використати інфраструктуру Expo Application Services. Вона забезпечує хмарну компіляцію React Native-проєкту, керування профілями збірки та формування інсталяційних пакетів. Такий підхід зменшує залежність від локального середовища розробника, оскільки збірка виконується в контрольованому середовищі з визначеними версіями Android SDK, Java Development Kit і Gradle.

Керування параметрами збирання здійснюється через конфігураційний файл eas.json. У ньому задаються профілі для тестової, попередньої та релізної версій застосунку. Для ручного встановлення на пристрої формується файл APK, а для публікації в Google Play може використовуватися формат AAB. Це дозволяє адаптувати процес розгортання до різних сценаріїв використання.

Рис. 3.8 – Процес збірки мобільного додатку за допомогою Expo
Application Services

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ

4.1 Методика тестування функціональних модулів

Тестування мобільної інформаційної системи спрямоване на перевірку коректності роботи основних модулів, стабільності взаємодії з backend-рівнем та придатності додатка до використання в умовах навчального процесу. Основу методики становить функціональне тестування, під час якого перевіряються не внутрішні фрагменти коду, а результат дій користувача в інтерфейсі.

Методика охоплює кілька груп перевірок ключових функцій системи. Такий поділ дозволяє оцінити додаток як послідовність практичних сценаріїв: вхід у систему, відкриття навчального розділу, перегляд розкладу, пошук корпусу та перехід до маршруту.

На першому етапі перевіряється автентифікація через університетську пошту домену @nubir.edu.ua. Тест має підтвердити створення користувацької сесії, завантаження профілю та відхилення сторонніх облікових записів. Після цього аналізується рольова модель: студент повинен мати доступ до навчальних і навігаційних сервісів, а адміністратор – до додаткових функцій керування даними.

Окремо тестується модуль розкладу. Перевіряється вибір факультету, академічної групи та дня тижня, а також коректність відображення занять. За нестабільного з'єднання система має використовувати кеш, резервне джерело або показувати повідомлення про недоступність даних. Це дозволяє оцінити не лише звичайний, а й аварійний сценарій роботи.

Навігаційний модуль перевіряється через список корпусів і гуртожитків, пошук, фільтрацію та відображення об'єктів на інтерактивній мапі. Для кожного об'єкта мають коректно показуватися назва, адреса, тип будівлі та координати. Якщо корпус присутній у списку, він повинен відображатися і на мапі. Інакше користувацький маршрут втрачає цілісність.

Під час тестування побудови маршруту перевіряється вибір об'єкта кампусу, запуск маршрутної логіки та реакція додатка на відсутність дозволу до геолокації. У такому випадку система має показати зрозуміле повідомлення, а не завершити роботу з помилкою.

Результати тестування доцільно фіксувати у таблиці з такими полями: модуль, початкові умови, дії користувача, очікуваний результат і фактичний результат. Такий формат спрощує аналіз роботи системи та дає змогу швидко визначити модулі, які потребують доопрацювання.

Фінальна перевірка має виконуватися на реальному мобільному пристрої, оскільки емулятор не завжди точно відтворює роботу геолокації, мережі та продуктивності. У результаті методика дозволяє оцінити готовність додатка до практичного використання студентами НУБіП України, зокрема стабільність розкладу, коректність навігації та надійність рольового доступу.

4.2 Тест-кейси та результати перевірки

Для перевірки працездатності мобільної інформаційної системи сформовано набір тест-кейсів, які охоплюють основні користувацькі сценарії. Перевірка виконувалася з позиції кінцевого користувача.

Проведена перевірка показала, що основні функціональні модулі працюють відповідно до визначених вимог.

Таблиця 4.1

Тест-кейси функціональних модулів мобільного додатка

№	Модуль	Тестовий сценарій	Очікуваний результат	Фактичний результат	Статус
1	Автентифікація	Вхід через університетськ у пошту домену @nubip.edu.ua	Користувач успішно входить у систему, створюється активна сесія	Вхід виконано, користувач перенаправлений на головний екран	Пройдено
2	Профіль користувача	Відкриття екрана профілю після входу	Відображаються ім'я користувача, email і статус	Дані профілю завантажено та відображено коректно	Пройдено
3	Рольова модель	Вхід користувача з роллю студента	Доступні навчальні та навігаційні функції	Студент бачить лише дозволені розділи	Пройдено
4	Рольова модель	Вхід користувача з роллю адміністратора	Доступні додаткові функції керування даними	Адміністративні функції відкриваються після перевірки ролі	Пройдено
5	Розклад	Відсутність інтернету після попереднього завантаження даних	Система використовує кешовану копію розкладу	Відображено останні збережені дані	Пройдено
6	Розклад	Недоступність основного джерела даних і відсутність кешу	Система показує повідомлення про недоступність даних	Показано повідомлення та кнопку переходу до автономного бекапу	Пройдено
7	Корпуси і гуртожитки	Відкриття списку корпусів	Відображаються назви, адреси та кнопки маршруту	Список корпусів завантажено коректно	Пройдено
8	Інтерактивна мапа	Відкриття мапи кампусу	На мапі відображаються маркери корпусів і гуртожитків	Маркери показано, об'єкти відповідають списку	Пройдено
9	Побудова маршруту	Натискання кнопки «Маршрут» для вибраного корпусу	Система відкриває маршрут до об'єкта	Маршрут формується через картографічний сервіс	Пройдено

4.3 Виявлення недоліків і заходи з їх усунення

Під час тестування скомпільованої APK-версії мобільного додатка було виявлено кілька недоліків, які проявилися саме на реальному пристрої. Основну увагу приділено роботі розкладу без інтернету та коректності відображення навігаційних елементів на Android.

Таблиця 4.2

Виявлені недоліки та заходи з їх усунення

№	Виявлений недолік	Причина виникнення	Заходи щодо усунення
1	Розклад не відкривався в офлайн-режимі у скомпільованому APK	Мережевий запит до зовнішнього джерела виконувався раніше, ніж читання локального кешу. У разі відсутності інтернету виконання переходило до блоку обробки помилки, і система не доходила до збережених даних.	Читання локального кешу перенесено на початок сценарію. Додаток спочатку відображає збережений розклад, а мережеві запити використовує лише для подальшого оновлення даних.
2	Випадаючі списки факультетів і груп були порожніми без інтернету	Структура університету формувалася переважно з мережевого джерела та не зберігалася локально. Через це користувач не міг вибрати факультет або групу в офлайн-режимі.	Додано кешування структури університету у <code>faculties_cache</code> . Після першого успішного завантаження списки факультетів і груп залишаються доступними без підключення до мережі.
3	Маркери на мапі некоректно відображалися на Android-пристроях	Кастомні графічні елементи маркерів по-різному рендерилися на Android через щільність пікселів і особливості картографічного компонента.	Оптимізовано розміри маркерів і скориговано їх позиціонування. Після цього об'єкти кампусу стали стабільніше відображатися на Android-пристроях.

4.4 Оцінка результатів тестування та готовності системи до використання

Після усунення виявлених недоліків було виконано повторну перевірку основних функціональних модулів мобільного додатка. Особливу увагу

приділено роботі розкладу в офлайн-режимі, коректності запитів у APK-версії та відображенню маркерів на Android-пристроях.

Повторне тестування підтвердило стабільну роботу ключових сценаріїв: авторизації через університетську пошту, перегляду профілю, вибору факультету й групи, відображення розкладу, роботи зі списком корпусів та інтерактивною мапою. Після зміни логіки на Offline-First система спочатку використовує локальні дані, а мережеві запити виконує лише для оновлення інформації.

За результатами перевірки встановлено, що додаток відповідає основним функціональним вимогам і може бути використаний у режимі дослідної експлуатації студентами НУБіП України. Подальше вдосконалення доцільно спрямувати на розширення тестування на різних пристроях, покращення пошуку корпусів і розвиток адміністративних інструментів.

4.5 Оцінка користі, продуктивності й зручності системи

Оцінка розробленої мобільної інформаційної системи проводилася за трьома критеріями: практична користь для студента, продуктивність роботи додатка та зручність інтерфейсу. Такий підхід дає змогу оцінити не лише технічну працездатність системи, а й придатність до щоденного використання.

Практична користь системи полягає в об'єднанні навчальної та навігаційної інформації в одному мобільному додатку. Студент отримує доступ до розкладу, списку корпусів і гуртожитків, мапи кампусу та побудови маршруту без потреби звертатися до різних вебресурсів.

Продуктивність додатка забезпечується локальним кешуванням. Система використовує збережені дані та виконує мережеві запити для оновлення інформації, завдяки чому основні екрани відкриваються швидше, а розклад залишається доступним за нестабільного інтернет-з'єднання.

Зручність системи визначається простою структурою інтерфейсу: основні функції винесені в окремі розділи – розклад, мапа, список об'єктів кампусу та

профіль користувача. Додаток також враховує можливу відсутність мережі, помилки зовнішнього API та відмову в доступі до геолокації, використовуючи повідомлення, кеш або резервні сценарії.

У результаті система має достатній рівень користі, продуктивності й зручності для дослідного впровадження. Подальше покращення може бути спрямоване на розширення функцій пошуку, оптимізацію мапи та збір відгуків від реальних користувачів.

4.6 Перспективи масштабування

Перспективи масштабування мобільної інформаційної системи пов'язані з розширенням функцій, збільшенням кількості користувачів і поступовим впровадженням на рівні всього університету. Поточна версія вже охоплює базові потреби студента: доступ до розкладу, перегляд профілю, роботу з мапою кампусу та навігацію до корпусів і гуртожитків.

Подальший розвиток системи може передбачати додавання детальніших навчальних даних: підгруп, індивідуальних подій, консультацій, змін аудиторій і дедлайнів. Це зробить додаток не лише довідковим інструментом, а частиною щоденної організації навчального процесу.

Навігаційний модуль також можна розширити. Доцільно додати аудиторії, кафедри, деканати, бібліотеки, їдальні, спортивні об'єкти та інші точки інтересу. Для кампусу НУБіП це має практичне значення, оскільки студенту часто потрібно знайти не лише корпус, а конкретне приміщення або службу.

Технічне масштабування може здійснюватися на базі вже обраної архітектури: React Native, Supabase, PostgreSQL, кешування та зовнішніх API. За збільшення кількості користувачів потрібно оптимізувати запити, розширити кешування, налаштувати індекси в базі даних і перевірити стабільність роботи системи під навантаженням.

Окремим напрямом розвитку є деталізація ролей користувачів. Крім студента й адміністратора, у майбутньому можна передбачити ролі викладача,

працівника деканату або модератора даних. Це дозволить гнучкіше керувати доступом і розподіляти відповідальність за оновлення інформації.

У перспективі система може бути інтегрована з іншими університетськими сервісами. Такий розвиток дозволить перетворити додаток на єдиний мобільний інформаційний канал для студентів НУБіП України.

ВИСНОВКИ

Робота над розробкою мобільної інформаційної системи для студентів НУБіП України дала змогу розглянути проблему доступу до навчальної та навігаційної інформації не лише з технічного, а й з організаційного погляду. Початковий аналіз показав, що студенти часто користуються кількома розрізненими джерелами даних: окремими сторінками з розкладом, картографічними сервісами, повідомленнями від одногрупників або власними нотатками. Такий підхід ускладнює планування навчального дня, особливо тоді, коли потрібно швидко знайти актуальний розклад, потрібний корпус або маршрут між об'єктами кампусу.

У межах кваліфікаційної роботи було спроектовано та реалізовано мобільний додаток, орієнтований на підтримку навчального процесу й просторової навігації студентів. Система об'єднує в одному середовищі основні функції, необхідні для щоденного використання: перегляд розкладу, роботу з профілем користувача, доступ до інформації про корпуси й гуртожитки, інтерактивну мапу та побудову маршруту. Це дозволяє зменшити залежність студента від різних інформаційних каналів і зробити доступ до потрібних даних більш зручним.

Під час виконання роботи було визначено основні вимоги до системи, ролі користувачів і ключові сценарії взаємодії. На етапі проєктування сформовано загальну структуру додатка, описано логіку роботи його функціональних модулів і взаємозв'язки між ними. Це дало змогу розглядати систему як цілісний програмний продукт, у якому навчальна інформація, навігація, профіль користувача та механізми доступу працюють узгоджено.

Окрему увагу приділено стабільності роботи системи в реальних умовах використання. Для мобільного додатка важливо, щоб базова інформація залишалася доступною навіть за нестабільного інтернет-з'єднання, адже така ситуація часто виникає в навчальних корпусах або під час переміщення

територією кампусу. Саме тому в роботі передбачено механізми локального збереження даних, які підвищують надійність доступу до розкладу та зменшують ризик інформаційних збоїв у повсякденному користуванні.

Проведене тестування підтвердило працездатність основних функціональних модулів системи. Було перевірено сценарії входу користувача, перегляду розкладу, роботи з профілем, пошуку корпусів і гуртожитків, використання мапи та побудови маршруту. У процесі перевірки виявлено окремі недоліки, після чого внесено необхідні коригування. Це дозволило підвищити стабільність роботи додатка та покращити його відповідність практичним потребам студентів.

З урахуванням виконаного аналізу, проєктування, реалізації та тестування можна зробити висновок, що поставлена мета роботи досягнута. Розроблена мобільна інформаційна система забезпечує студентам зручний доступ до навчальної та навігаційної інформації, підтримує автономність роботи й може використовуватися як основа для подальшого розвитку цифрових сервісів університету. Її впровадження здатне зменшити кількість рутинних дій, пов'язаних із пошуком розкладу та корпусів, і зробити щоденну взаємодію студентів з університетською інфраструктурою більш впорядкованою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. НУБіП України. Офіційний сайт Національного університету біоресурсів і природокористування України. URL: <https://nubip.edu.ua/> (дата звернення: 12.01.2026).
2. Факультет інформаційних технологій. НУБіП України. URL: <https://nubip.edu.ua/faculty/it> (дата звернення: 14.01.2026).
3. Розклад занять НУБіП України. НУБіП України. URL: <https://rozklad.nubip.edu.ua/> (дата звернення: 16.01.2026).
4. Моє студмістечко. НУБіП України. URL: <https://campus.nubip.edu.ua/> (дата звернення: 18.01.2026).
5. React Native Documentation. React Native. URL: <https://reactnative.dev/docs/getting-started> (дата звернення: 22.01.2026).
6. Expo Documentation. Expo. URL: <https://docs.expo.dev/> (дата звернення: 25.01.2026).
7. Expo Application Services. Expo. URL: <https://docs.expo.dev/eas/> (дата звернення: 27.01.2026).
8. TypeScript Documentation. TypeScript. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 30.01.2026).
9. Supabase Documentation. Supabase. URL: <https://supabase.com/docs> (дата звернення: 03.02.2026).
10. Supabase Auth. Supabase. URL: <https://supabase.com/docs/guides/auth> (дата звернення: 06.02.2026).
11. Row Level Security. Supabase. URL: <https://supabase.com/docs/guides/database/postgres/row-level-security> (дата звернення: 09.02.2026).
12. PostgreSQL Documentation. PostgreSQL Global Development Group. URL: <https://www.postgresql.org/docs/current/> (дата звернення: 13.02.2026).

13. Using OAuth 2.0 to Access Google APIs. Google for Developers. URL: <https://developers.google.com/identity/protocols/oauth2> (дата звернення: 17.02.2026).
14. Google Maps Platform Documentation. Google for Developers. URL: <https://developers.google.com/maps/documentation> (дата звернення: 21.02.2026).
15. React Native Maps Documentation. GitHub. URL: <https://github.com/react-native-maps/react-native-maps> (дата звернення: 26.02.2026).
16. Unified Modeling Language Specification. Object Management Group. URL: <https://www.omg.org/spec/UML/> (дата звернення: 03.03.2026).
17. Moodle App Features. MoodleDocs. URL: https://docs.moodle.org/en/Moodle_app_features (дата звернення: 09.03.2026).
18. Navigine Indoor Positioning and Indoor Navigation SDK. Navigine. URL: <https://navigine.com/platform/sdk/> (дата звернення: 14.03.2026).

Додаток А**Декларація про академічну доброчесність та використання ШІ**

Я, Скрипник Микита Олексійович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Розробка підсистеми навчального процесу інтегрованої мобільної системи підтримки студентів» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що:

інструменти ШІ: ChatGPT та Gemini були використані для:

- перекладу іншомовних джерел;
- стилістичного редагування та перевірки граматики;
- допомоги у написанні/відлагодженні програмного коду;
- стилістичного редагування авторського тексту та його приведення до норм академічного письма;
- структурування інформації на основі наданих технічних даних та результатів самостійного проектування системи;
- верифікації оформлення пояснювальної записки на відповідність методичним вказівкам університету;
- підготовки перекладу анотації англійською мовою

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України. «__» травня 2026 р. _____ Микита СКРИПНИК