Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА ТА ДОСЛІДЖЕННЯ FRONTEND ЧАСТИНИ ЗАСТОСУНКУ ДЛЯ ДИТЯЧОГО САДОЧКА

DEVELOPMENT AND RESEARCH OF THE FRONTEND PART OF THE APPLICATION FOR KINDERGARTEN

спеціальність 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗм-21
Ніколайчук С. Р.
Керівник:
к.т.н., доцент
Ліщина Н. М.

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент Суринович О. М.
_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *магістр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«___» _____ 202__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Ніколайчуку Сергію Руслановичу

1. Тема кваліфікаційної роботи: Розробка та дослідження frontend частини застосунку для дитячого садочка
Керівник роботи: Ліщина Наталія Миколаївна, доцент, к.т.н.
затверджені наказом закладу вищої освіти від «29» березня 2025 року № 190/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи: 04 грудня 2025 р.
3. Вихідні дані до роботи *технічне та програмне забезпечення ЕОМ*
4. Зміст розрахунково-пояснювальної записки: *аналіз проблематики комунікації в закладах дошкільної освіти та вибір методів дослідження, обґрунтування технологій і реалізація клієнтської частини застосунку, експериментальне дослідження результативності програмного забезпечення*
5. Перелік графічного матеріалу 9 рисунків, 20 лістингів коду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Ліщина Н. М.*		
Теоретичне дослідження та практична реалізація	*Ліщина Н. М.*		
Експериментальне дослідження системи	*Ліщина Н. М.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Андрущак І. Є.*		
Показник запозичень тексту	___%		
Академічна доброчесність	*Ліщина Н. М.*		

7. Дата видачі завдання *«02 квітня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1	Провести огляд літературних джерел по темі кваліфікаційної роботи	02.05.2025	
2	Провести аналіз загальної проблеми і вибір напрямків дослідження	24.09.2025	
3	Розробити функціональну модель та архітектуру системи	01.11.2025	
4	Описати засоби розробки об'єкта проектування	19.11.2025	
5	Практична реалізація об'єкта проектування	26.11.2025	
6	Розробити методику для проведення експерименту	05.11.2025	
7	Провести аналіз результатів експерименту	15.11.2025	
8	Здача чистового варіанту кваліфікаційної роботи на кафедру	04.12.2025	

Здобувач вищої освіти _____ Ніколайчук С. Р.

Керівник кваліфікаційної роботи _____ Ліщина Н. М.

АНОТАЦІЯ

Ніколайчук С. Р. Розробка та дослідження frontend частини застосунку для дитячого садочка. Рукопис.

Кваліфікаційна робота магістра ОП «Інженерія програмного забезпечення» спеціальності 121 Інженерія програмного забезпечення. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота присвячена розробленню фронтенд-частини мобільного застосунку для дошкільного закладу. Метою є створення кросплатформеного інтерфейсу для батьків і вихователів, що забезпечує доступ до інформації про дитину, розклад, сповіщення та події.

У роботі проаналізовано предметну область та існуючі рішення. Встановлено, що більшість доступних іноземних сервісів не адаптовані до українського освітнього середовища, а наявні українські продукти не мають повноцінних мобільних інтерфейсів. Це підтверджує актуальність створення нового застосунку.

Обґрунтовано вибір технологій: React Native та Expo – для реалізації мобільного інтерфейсу; Expo Router – для організації навігації; Firebase Authentication – для авторизації; REST API та Axios – для роботи з даними; AsyncStorage – для локального збереження; i18n-js – для мультимовності. Застосунок реалізовано з використанням контекстного управління станом: AuthContext, LocaleContext, ParentContext, TeacherContext.

Практичним результатом є мобільний застосунок, який підтримує персоналізацію за ролями, працює на Android і iOS, забезпечує просту взаємодію між батьками та вихователями та може бути інтегрований у реальні дошкільні заклади.

Ключові слова: мобільний застосунок, React Native, Expo, Firebase Authentication, REST API, навігація, мультимовність.

ABSTRACT

Nikolaichuk S. R. Development and Research of the Frontend Part of the Application for Kindergarten. Manuscript.

Master's qualification work in the educational program «Software Engineering», specialty 121 Software Engineering. Lutsk National Technical University. Lutsk, 2025.

The qualification work is devoted to the development of the frontend part of a mobile application for a preschool institution. The purpose is to create a cross-platform interface for parents and teachers, providing access to child-related information, schedules, notifications, and daily events.

The work includes an analysis of the subject area and existing solutions. It has been established that most available foreign services are not adapted to the Ukrainian educational context, while existing Ukrainian products lack fully functional mobile interfaces. This confirms the relevance of developing a new application.

The choice of technologies is substantiated: React Native and Expo are used for implementing the mobile interface; Expo Router provides navigation; Firebase Authentication is used for user authorization; REST API and Axios enable data exchange; AsyncStorage is applied for local data storage; i18n-js supports multilingual functionality. The application is implemented using context-based state management, including AuthContext, LocaleContext, ParentContext, and TeacherContext.

The practical result is a mobile application that supports role-based personalization, operates on both Android and iOS, ensures convenient interaction between parents and teachers, and can be integrated into real preschool institutions.

Keywords: mobile application, React Native, Expo, Firebase Authentication, REST API, navigation, multilingual suppor.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах цифрової трансформації освіти особливого значення набувають інформаційні рішення, які забезпечують ефективну взаємодію між учасниками освітнього процесу та підтримують доступ до актуальних даних у зручному мобільному форматі. Дошкільні навчальні заклади тривалий час залишалися поза межами комплексних цифрових змін, що зумовлює низку труднощів у комунікації між вихователями, адміністрацією та батьками. Використання паперових оголошень, групових чатів та усних повідомлень призводить до фрагментованості інформації, низької оперативності її оновлення та відсутності єдиного структурованого каналу зв'язку. У таких умовах виникає потреба у впровадженні сучасного мобільного інструменту, здатного забезпечити стабільний доступ до даних про діяльність дітей, розклад занять і щоденні події.

Міжнародний досвід демонструє зростання популярності мобільних застосунків у сфері дошкільної освіти, проте наявні рішення здебільшого не враховують вимог українського освітнього середовища. Вони часто мають закриту комерційну модель, обмежені можливості налаштування та недостатню підтримку української мови. Це формує потребу у створенні кросплатформеного застосунку, адаптованого до місцевих умов, здатного працювати на різних мобільних пристроях і забезпечувати інтуїтивно зрозумілий інтерфейс для батьків і вихователів.

Метою кваліфікаційної роботи є розроблення та дослідження фронтенд-частини мобільного застосунку для дитячого садка з використанням сучасних технологій мобільної розробки.

Для досягнення цієї мети сформульовано такі основні завдання:

– проаналізувати предметну область дошкільної освіти та існуючі програмні рішення;

– визначити функціональні та архітектурні вимоги до мобільного застосунку;

– обґрунтувати вибір технологій і засобів розроблення фронтенд-частини;

– розробити архітектуру застосунку та його логічну структуру;

– реалізувати основні функціональні модулі для ролей батьків та вихователів;

– забезпечити авторизацію, навігацію, мультимовність і роботу з даними;

– провести тестування та оцінити продуктивність і зручність використання;

– виконати дослідження відповідності застосунку визначеним вимогам.

Об'єктом дослідження є процес розроблення мобільних застосунків для дошкільної освіти. Предметом дослідження є методи та інструменти створення фронтенд-частини мобільного застосунку з використанням технологій React Native та Expo, а також підходи до організації архітектури, навігації, локалізації та взаємодії з мережевими сервісами.

Наукова новизна роботи полягає у застосуванні сучасних мобільних технологій та архітектурних підходів, які дозволяють створити адаптивний, масштабований і структурований інтерфейс, орієнтований на потреби дошкільної освіти. Використання Expo Router, Firebase Authentication та системи локалізації i18n-js забезпечує уніфіковану організацію навігації, диференційований доступ до даних і підтримку мультимовності без перевантаження клієнтської частини.

Практична цінність роботи полягає у можливості подальшого використання розробленого застосунку у реальних дошкільних закладах, що сприятиме підвищенню ефективності комунікації, покращенню доступу до інформації та створенню основи для подальшої цифровізації освітнього середовища.

Апробація результатів дослідження. Основні теоретичні положення та практичні напрацювання кваліфікаційної роботи були представлені у тезах доповіді «Мобільний застосунок як інструмент цифрової трансформації дошкільної освіти» (ІТОНВ-2025), що засвідчило практичну значущість та застосовність отриманих результатів [1].

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМАТИКИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд і аналіз предметної області проблеми, результатів існуючих теоретичних та експериментальних досліджень

Дошкільна освіта є важливою складовою освітнього процесу, яка забезпечує ранній розвиток дитини, формування базових компетентностей, соціалізацію та підготовку до подальшого навчання. Проте в більшості українських дошкільних закладів процес взаємодії між батьками, вихователями та адміністрацією досі залишається переважно офлайн-орієнтованим. Основні комунікації відбуваються через друковані оголошення, групові чати, усні повідомлення або паперові журнали, що створює низку проблем:

– відсутність єдиної платформи для інформування батьків про важливі події;

– складність доступу до актуальної інформації щодо розкладу занять, меню харчування чи новин садочку;

– низька швидкість оновлення інформації та можливість її втрати;

– відсутність систематизованого онлайн-доступу для відстеження розвитку та діяльності дітей.

Наукові дослідження вказують, що впровадження мобільних освітніх платформ позитивно впливає на якість співпраці між педагогами та батьками. Зокрема, у роботі Neumann M. досліджено роль цифрових комунікацій у дошкільній освіті, де підкреслюється, що мобільні застосунки підвищують залученість батьків та спрощують доступ до даних про дитину [2]. Дослідження Blackwell C., Lauricella A., Wartella E. також демонструє, що цифрові інструменти ефективно підтримують освітні процеси в ранньому віці та сприяють підвищенню взаємодії між учасниками освітнього середовища [3].

У контексті цифрової трансформації освіти, яка активно розвивається в Україні відповідно до державної стратегії цифровізації, виникає потреба у

впровадженні сучасних IT-рішень у дошкільний сектор. Мобільний застосунок, орієнтований на потреби дитячих садочків, здатний закрити ключові прогалини: забезпечити централізовану комунікацію, створити зручний цифровий простір для батьків, підвищити прозорість, а також оптимізувати адміністративні процеси.

Особливої актуальності такі рішення набувають у контексті безпеки та стабільності освітнього процесу, де швидке сповіщення про події, зміни розкладу чи надзвичайні ситуації є критично важливим. Таким чином, проблема створення мобільного застосунку для дошкільної освіти є сучасною, соціально значущою та технологічно виправданою.

Сучасний ринок цифрових продуктів для дошкільної освіти представлений переважно іноземними мобільними застосунками, серед яких найбільш відомими є Brightwheel, HiMama та Famly. Ці сервіси забезпечують базову взаємодію між вихователями та батьками, включаючи обмін повідомленнями, перегляд фотографій, відстеження щоденної активності дітей та формування простих звітів. Незважаючи на їхню поширеність, зазначені рішення створювалися для освітніх систем інших країн і не враховують специфіку українського дошкільного середовища.

Під час аналізу було встановлено, що іноземні платформи не мають адаптації до українських освітніх стандартів, а їхній функціонал орієнтований на інші адміністративні моделі. Більшість з них не підтримують україномовний інтерфейс, що значно ускладнює використання в умовах українських закладів. Додатковою проблемою є висока вартість підписки, що робить такі сервіси малодоступними для комунальних дитячих садочків. Важливою особливістю також є закритість вихідного коду, що унеможливлює кастомізацію або адаптацію застосунку до потреб конкретної установи.

Українські рішення, представлені на ринку, здебільшого мають формат веб-платформ або CRM-систем, орієнтованих переважно на адміністративний облік. Вони рідко передбачають наявність повноцінного мобільного інтерфейсу, що був би зручним і зрозумілим для батьків та вихователів у повсякденній

роботі. У більшості випадків мобільних застосунків не існує, або вони виконують допоміжну роль і не забезпечують необхідної інтерактивності.

Узагальнюючи результати аналізу, можна зробити висновок, що наявні іноземні та українські рішення не відповідають вимогам локального освітнього середовища. Вони не враховують мовну специфіку, потреби батьків та вихователів, вимоги до швидкої комунікації, обмеження бюджетів закладів та потребу у простій адаптації інтерфейсу. Це підтверджує актуальність створення нового кросплатформеного мобільного застосунку, який би забезпечував просту взаємодію між учасниками освітнього процесу, підтримував мультимовність, був доступним у використанні та відповідав усьому спектру функціональних вимог дошкільних закладів України.

1.2 Огляд і аналіз методів та засобів розробки для вирішення проблеми дослідження

Основою програмної реалізації обрано технологію React Native, що забезпечує нативний рендеринг інтерфейсу та підтримку одночасної розробки для Android та iOS [4]. Це рішення дозволяє уникнути створення двох окремих кодових баз і значно скорочує час розроблення. Для спрощення роботи з нативним середовищем застосовано платформу Expo, яка надає стабільний набір інструментів, автоматизує збірки та забезпечує простий доступ до API пристрою [5].

Організацію навігації реалізовано засобами Expo Router, що працює на основі файлової маршрутизації та дозволяє будувати масштабовану структуру інтерфейсу з мінімальними витратами на конфігурацію [6]. Такий підхід забезпечує чітке розмежування маршрутів для ролей Parent та Teacher, що відповідає вимогам архітектури застосунку.

Для реалізації механізму автентифікації обрано Firebase Authentication, який надає захищений та перевірений інструментарій для роботи з користувацькими обліковими записами без необхідності створення власного

серверного модуля авторизації [7]. Інші сервіси Firebase використовуються як бекенд-інфраструктура для зберігання та оброблення даних [8], що забезпечує стабільність і масштабованість застосунку.

Взаємодія з backend-частиною організована на основі принципів REST API [9]. Такий підхід стандартизує формати запитів та відповідей, полегшує налагодження та забезпечує передбачуваність роботи мережевих модулів. Для виконання HTTP-запитів обрано бібліотеку Axios.

Оскільки застосунок підтримує українську й англійську мови, для реалізації мультимовності використано бібліотеку i18n-js [10], яка забезпечує централізоване керування текстовими ресурсами та легку адаптацію інтерфейсу до нових мовних налаштувань. Зберігання локальних даних і налаштувань здійснено за допомогою AsyncStorage [11].

Для роботи з календарями та датами було обрано стабільну бібліотеку react-native-calendars [12], яка забезпечує коректне відображення подій і розкладу. Відображення зображень оптимізовано за допомогою expo-image [13], що підвищує продуктивність та якість візуалізації.

Проєктування інтерфейсу здійснювалося з урахуванням принципів мобільного UX-дизайну, викладених у рекомендаціях Nielsen Norman Group [14]. При цьому використовувалися сучасні підходи React-екосистеми [15] та бібліотеки навігації React Navigation у тих модулях, де необхідні можливості, не охоплені Expo Router [16].

Загальна архітектура та вибрані інструменти узгоджуються з напрямами цифрової трансформації освіти, визначеними державними стратегіями України [17], що підтверджує доцільність застосування вибраного технологічного стеку для розвитку ІТ-рішень у сфері дошкільної освіти.

1.3 Постановка завдання на кваліфікаційну роботу магістра

З огляду на проаналізовану предметну область дошкільної освіти та виявлені недоліки існуючих мобільних рішень, у межах кваліфікаційної роботи

формулюється завдання зі створення фронтенд-частини мобільного застосунку для дитячого садочка. Метою є розробка кросплатформеного інтерфейсу, що забезпечуватиме зручну та безпечну взаємодію між батьками і вихователями. Для досягнення поставленої мети визначено такі основні цілі:

– проаналізувати предметну область дошкільної освіти, потреби користувачів та наявні програмні рішення;

– визначити ключові функціональні можливості, а також сформувати функціональні й архітектурні вимоги до мобільного застосунку;

– обґрунтувати вибір технологій і засобів розроблення, що забезпечують кросплатформеність, стабільність і можливість подальшого масштабування;

– розробити архітектуру застосунку, логічну структуру модулів і модель взаємодії користувачів з урахуванням ролей батьків та вихователів;

– реалізувати механізм авторизації та доступу до персоналізованих даних користувача;

– створити інтерфейсні модулі для відображення інформації про дитину, розклад занять, сповіщення та медіаматеріали;

– забезпечити підтримку мультимовності (українська/англійська) та реалізувати взаємодію з віддаленим API для отримання, оброблення й відображення даних;

– провести тестування інтерфейсу й функціональних можливостей, оцінити продуктивність і зручність використання та перевірити відповідність застосунку визначеним вимогам.

Сформульовані цілі визначають обсяг і зміст дослідження та окреслюють результат, який має бути досягнутий у кваліфікаційній роботі.

Висновки до розділу 1

У першому розділі було проведено системний аналіз предметної області дошкільної освіти та визначено актуальні проблеми, що виникають під час комунікації між вихователями, батьками та адміністрацією дитячих садочків.

Встановлено, що більшість українських дошкільних закладів продовжує використовувати офлайн-канали взаємодії, які не забезпечують належної оперативності, прозорості та доступності інформації. Наявні процеси часто супроводжуються затримками, дублюванням повідомлень і ризиком втрати важливих даних.

Аналіз існуючих зарубіжних і українських цифрових рішень показав, що наявні на ринку мобільні застосунки або не адаптовані до українського освітнього середовища, або не забезпечують повноцінну підтримку мобільної платформи. Іноземні сервіси виявилися фінансово недоступними та не підтримують україномовний інтерфейс, а українські рішення переважно функціонують у форматі веб-платформ і не охоплюють потреб щоденної взаємодії між учасниками освітнього процесу. Таким чином, підтверджено існування чітко вираженої потреби у створенні нового мобільного застосунку, орієнтованого на локальні умови та специфіку дошкільних закладів України.

На основі проведеного аналізу сформульовано постановку завдання для кваліфікаційної роботи, яка охоплює розроблення кросплатформеного інтерфейсу, реалізацію авторизації, побудову навігації відповідно до ролей користувачів, підтримку мультимовності та організацію взаємодії з віддаленим сервером. Визначено ключові цілі, яких необхідно досягти для створення сучасного, зручного, технологічного та масштабованого мобільного застосунку.

Отже, розділ 1 заклав концептуальні та теоретичні основи роботи, обґрунтував необхідність розробки нового рішення та визначив напрямки подальшої практичної реалізації, що здійснюватиметься в наступних розділах кваліфікаційної роботи.

РОЗДІЛ 2
ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА ФРОНТЕНД-ЧАСТИНИ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Обґрунтування вибору шляхів, технологій та засобів вирішення поставленого завдання

Реалізація мобільного застосунку для дошкільного закладу потребує вибору технологій та підходів, які забезпечують високу продуктивність, стабільну роботу на різних мобільних платформах, зручність використання та можливість подальшого масштабування. Виходячи з поставленої мети та сформульованих у розділі 1 завдань, доцільним є застосування кросплатформених інструментів, що дозволяють об'єднати процес розроблення для Android та iOS у єдину кодову базу.

Основою рішення обрано технологію React Native. Це середовище дає змогу формувати інтерфейс із використанням декларативної моделі та забезпечує нативний рендеринг елементів. Такий підхід дозволяє підвищити швидкість розроблення і зменшити похибки, що виникають під час створення інтерфейсу окремо для кожної платформи. Застосування JavaScript як базової мови підсилює інтеграцію з інструментами тестування та засобами налагодження.

Платформа Expo використана як інструмент, що забезпечує кероване середовище, стабільний доступ до нативних можливостей пристрою та полегшує розгортання застосунку. Expo автоматизує частину операційних процесів, що знижує ризик технічних помилок і покращує повторюваність результатів під час збирання застосунку.

Вибір Expo Router для організації навігації обґрунтований необхідністю створити передбачувану, модульну та зрозумілу структуру інтерфейсу. Навігація, побудована на файловому підході, знижує складність маршрутизації, дозволяє уникнути дублювання логіки переходів і забезпечує структурованість навіть у разі розширення функціоналу.

Для реалізації механізмів автентифікації використано сервіс Firebase Authentication. Цей вибір продиктований потребою в надійному, захищеному та перевіреному інструменті, що мінімізує ймовірність помилок розробника під час створення критично важливих процесів автентифікації. Firebase забезпечує відповідність вимогам безпеки та зменшує час розроблення порівняно зі створенням власних механізмів перевірки користувачів.

Взаємодія з даними організована на основі REST API. Такий підхід є універсальним та передбачуваним, забезпечує чітке розмежування між фронтенд- та бекенд-частинами, а також дає можливість стандартизувати формати обміну інформацією. Для виконання мережевих запитів застосовано Axios, що дає змогу централізовано обробляти похибки, керувати конфігурацією запитів та забезпечувати відтворюваність результатів.

Підтримка мультимовності реалізована за допомогою бібліотеки i18n-js, оскільки вона дозволяє створити детерміновану структуру текстових ресурсів і мінімізувати похибки, пов'язані з неправильним відображенням мовних елементів. Використання expo-localization забезпечує відповідність мовних налаштувань пристрою та інтерфейсу.

Для відображення розкладу та подій обрано react-native-calendars. Це рішення забезпечує стабільність роботи з датами та дозволяє уникнути помилок у відтворенні календарних елементів.

Загальний підхід до реалізації застосунку базується на контекстному управлінні станом, яке забезпечує контрольованість даних, однозначність логіки роботи інтерфейсу та зниження ризику непослідовних змін станів. Такий підхід дозволяє відтворювати однакову поведінку застосунку у повторюваних умовах, що є важливим під час тестування.

Обрані технології та методи дають змогу реалізувати поставлене завдання найоптимальнішим способом, забезпечуючи стабільність, передбачувану поведінку інтерфейсу та можливість подальшого розширення функціональності без суттєвої перебудови архітектури.

2.2 Архітектура застосунку та структура проєкту

Архітектура фронтенд-частини мобільного застосунку сформована відповідно до принципів модульності та логічного розмежування функціональності. Такий підхід забезпечує зрозумілу організацію програмного коду, спрощує подальший розвиток і полегшує підтримку програмного продукту. Основою побудови інтерфейсу є файлова структура, що визначає логіку маршрутів, поділ інтерфейсу на екранні модулі та взаємодію з даними.

Центральним елементом є каталог app (рис. 2.1), у якому розміщуються всі екрани застосунку та відповідні макети. Файлова маршрутизація Expo Router дозволяє формувати інтерфейс у вигляді вкладених структур, де кожен каталог відповідає певному розділу застосунку. Каталоги (parent) та (teacher) містять окремі інтерфейси для різних ролей користувачів, що забезпечує розмежування доступу та відокремлення логіки роботи.

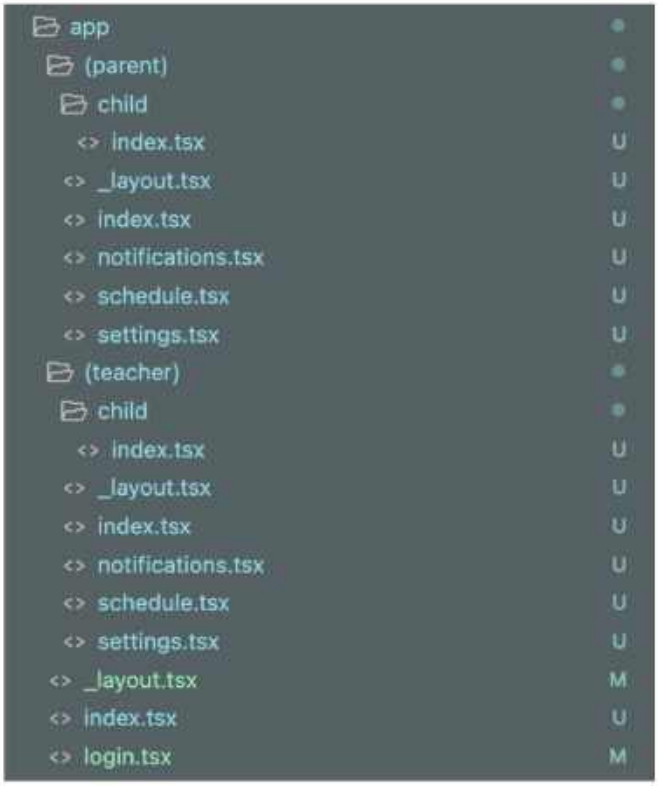

Рисунок 2.1 – Каталог app (відображає екрани застосунку)

Додаткову функціональність забезпечують три допоміжні каталоги. Каталог components містить багаторазові інтерфейсні елементи, які використовуються на різних екранах. Каталог contexts включає логіку управління станом, що охоплює авторизацію, дані користувачів та інші аспекти взаємодії. Каталог lib містить модулі для роботи з API, конфігураціями та службовими функціями. Каталог types визначає типи даних, що використовуються у проєкті, та сприяє підвищенню надійності коду завдяки статичній типізації.

Таке структурування забезпечує чіткий розподіл відповідальності між частинами застосунку (ліст. 2.1). Кожен модуль виконує власні функції, а зміни в одному компоненті мінімально впливають на роботу інших частин. Це дозволяє ефективно впроваджувати нові функції, розширювати інтерфейс та адаптувати застосунок до потреб користувачів.

Лістинг 2.1 – Файлова структура проєкту (фрагмент дерева каталогів)

```
app/                    # екрани застосунку та маршрутизація
  index/                # стартовий екран
  login/                # екран авторизації
  (parent)/             # інтерфейс для батьків
    child/              # екран перегляду даних дитини
  (teacher)/            # інтерфейс для вихователів
    child/              # екран роботи з дитиною
assets/                 # статичні ресурси
components/             # багаторазові UI-компоненти
  ui/
constants/              # константи застосунку
contexts/               # контексти керування станом
hooks/                  # користувацькі хуки
lib/
  api/                  # модулі роботи з REST API
  firebase/             # конфігурація Firebase
  i18n/                 # мультимовність
scripts/                # службові скрипти
types/                  # типи даних (TypeScript)
```

Кінець лістингу 2.1

2.3 Організація маршрутизації та навігації в Expo Router

Організація навігації є ключовим аспектом структурування мобільного застосунку, оскільки визначає логіку переходів між екранами, чіткість інтерфейсу та зручність взаємодії користувача з системою. У цьому проєкті маршрутизація реалізована за допомогою Expo Router, що підтримує файлову структуру та дає змогу визначати навігаційні маршрути без необхідності додаткового конфігурування.

Основною особливістю Expo Router є автоматичне відображення структури каталогів у вигляді ієрархії екранів. Це забезпечує прозору й передбачувану логіку навігації, а також дає змогу легко масштабувати структуру застосунку, додаючи нові модулі чи модифікуючи існуючі.

У застосунку реалізовано розмежування навігації відповідно до ролей користувача: Parent (Батьки) та Teacher (Вихователь). Для кожної ролі призначені окремі набори екранів, організовані в окремих вкладених каталогах: app/(parent) та app/(teacher)

Кожен із цих макетів містить власний _layout.tsx файл, який відповідає за конфігурацію нижньої панелі навігації (Tabs) та загальні параметри відображення.

У випадку з батьківським інтерфейсом використано компонент ParentTabsContent, що відповідає за візуалізацію вкладок, перевірку стану даних та обробку помилок. Доступ до екрану забезпечується через захисний компонент ProtectedRoute, який запобігає доступу до вкладок без авторизації.

Нижня частина інтерфейсу (рис. 2.2) містить панель навігації типу Tabs, яка забезпечує швидкий перехід між основними розділами застосунку. Панель складається з чотирьох вкладок: «Дитина», «Розклад», «Сповіщення» та «Налаштування». Кожна вкладка має піктограму та текстову назву, що полегшує орієнтацію в інтерфейсі. Активна вкладка виділяється акцентним кольором, тоді як округлені краї та контрастне тло панелі забезпечують чітке візуальне розмежування між навігацією та основним контентом. Така структура відповідає

принципам мобільного UX-дизайну та гарантує зручність щоденного використання.

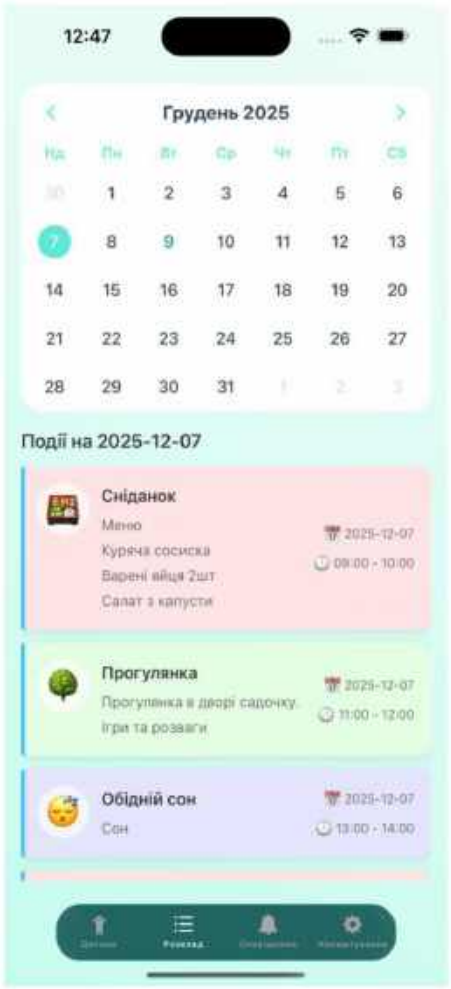

Рисунок 2.2 – Фрагмент навігаційного макета для ролі Parent

Аналогічна логіка реалізована для ролі Teacher, проте її інтерфейс доповнено компонентом GroupSelector (рис. 2.3), який відображає поточну групу дітей і дозволяє швидко перемикатися між ними. Це забезпечує зручність роботи вихователя та дає змогу адаптувати інтерфейс під реальні сценарії використання (рис. 2.4).

```
return (
  <Tabs
    screenOptions={{
      headerShown: true,
      headerTransparent: false,
      headerStyle: {
        backgroundColor: "#f0fdfa",
      },
      headerTitle: () => <GroupSelector />,
      tabBarActiveTintColor: "white",
      tabBarInactiveTintColor: "#a9a7a7",
      tabBarButton: HapticTab,
      tabBarStyle: {
      },
      tabBarBackground: () =>
        Platform.OS === "ios" ? (
        ) : null,
      tabBarLabelStyle: {
      },
    }}
```

Рисунок 2.3 – Фрагмент відмінності навігації у ролі teacher

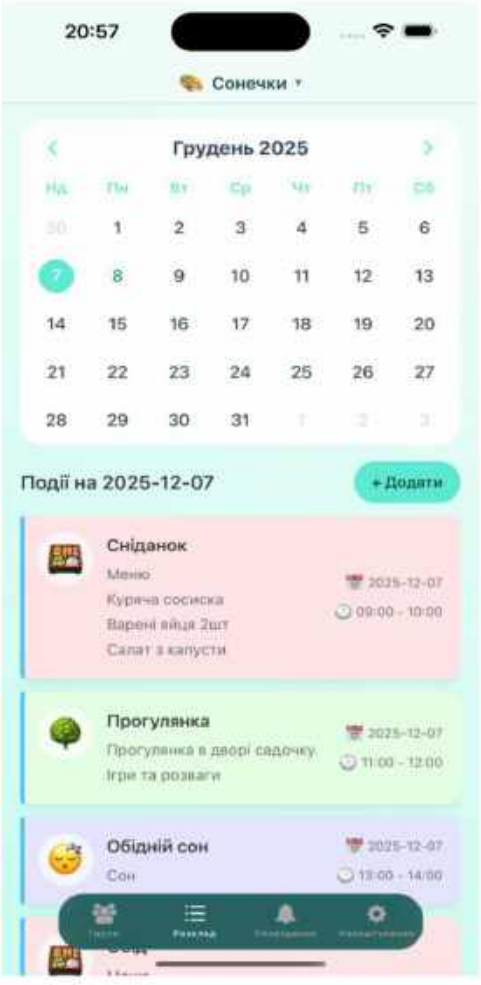

Рисунок 2.4 – Макет екрану розкалад ролі Teacher

2.4 Особливості проєктування інтерфейсу та UX-рішень

Проєктування інтерфейсу мобільного застосунку для дошкільної освіти потребує врахування особливостей двох категорій користувачів: батьків та вихователів. Кожна з цих ролей взаємодіє з різним набором даних, що визначає різний підхід до структурування інформації та організації переходів між екранами. Під час проєктування інтерфейсу особлива увага приділялася простоті взаємодії, швидкому доступу до ключових функцій та інтуїтивності навігації, що є критично важливим для щоденного використання.

Одним з ключових завдань дизайну було мінімізувати когнітивне навантаження на користувача. Інтерфейс передбачає використання великих інтерактивних елементів, чіткої ієрархії інформації, зрозумілих заголовків і зручної візуальної структури. Важливим принципом було забезпечення однаковості поведінки елементів у межах різних екранів, що підвищує передбачуваність взаємодії та полегшує навчання користувача.

Застосунок реалізований з урахуванням мультимовної підтримки, що передбачає динамічне завантаження текстових ресурсів та правильне відображення інтерфейсу українською та англійською мовами. Для цього враховано відмінності довжини текстів, перенесення слів та адаптацію елементів до різних мовних контекстів. У процесі проєктування було створено окремі макети для сценаріїв, де довші рядки могли впливати на ширину або висоту компонентів.

Додаткову увагу приділено адаптації до різних розмірів екранів. Оскільки застосунок працює як на смартфонах, так і на планшетах, було впроваджено гнучку систему верстки із застосуванням властивостей flex, динамічних відступів та масштабованих елементів. Завдяки цьому інтерфейс зберігає читабельність та зручність незалежно від характеристик конкретного пристрою.

Під час розроблення UX-навігації було обрано підхід із чітким розмежуванням доступу залежно від ролі користувача. Батьки та вихователі мають відокремлені маршрути, що запобігає змішуванню інтерфейсу та

полегшує орієнтацію всередині застосунку. Це рішення є важливим як з погляду зручності, так і з погляду безпеки даних.

Нижче наведено фрагмент одного з візуальних макетів головного екрану батьківської ролі (рис. 2.5).

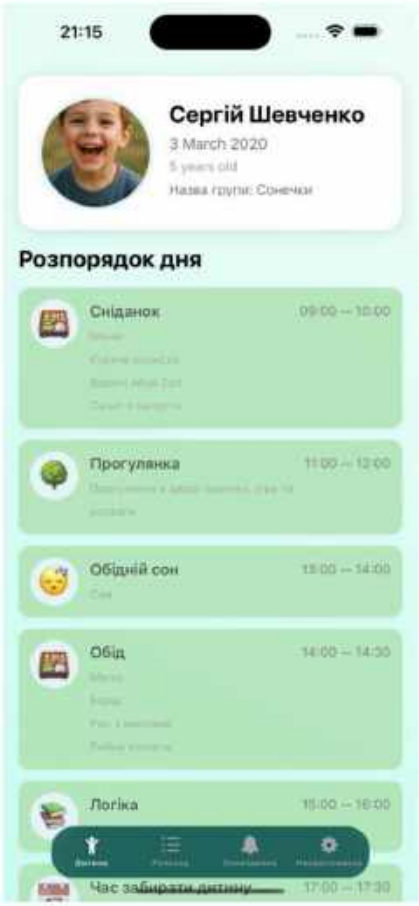

Рисунок 2.5 – Макет головного екрану ролі Parent

Для демонстрації компонентної структури одного з екранів наведено фрагмент JSX-коду, що описує відображення блоку інформації про дитину (ліст. 2.2).

Лістинг 2.2 – Фрагмент компонента ChildCard

```
<View style={styles.profileCard}>
          <Image
            source={{ uri: selectedChild.photo }}
            style={styles.kidPhoto}
          />
          <View>
```

```
                        <ThemedText style={styles.kidName}>
                          {selectedChild?.firstName}
{selectedChild?.lastName}
                        </ThemedText>
                        <ThemedText style={styles.birthDate}>
                          {selectedChild?.birthDate }
                        </ThemedText>
                        <ThemedText style={styles.age}>
                          {selectedChild
                            ? new Date().getFullYear() -
                              new
Date(selectedChild.birthDate).getFullYear()
                            : ""} {t(child.year)}
                        </ThemedText>
                        <ThemedText style={styles.group}>

{t("child.group_name")}{selectedChild?.groupId.name}
                        </ThemedText>
                      </View>
                    </View>
```

Кінець лістингу 2.2

Усі рішення щодо структури, верстки та організації інтерфейсу були спрямовані на досягнення максимальної інтуїтивності та доступності застосунку. Це забезпечує позитивний досвід взаємодії користувача з системою, знижує час на опанування функціональних можливостей та робить застосунок придатним для повсякденного використання у дошкільному освітньому середовищі.

2.5 Реалізація контекстного управління станом

У мобільному застосунку, призначеному для батьків та вихователів дошкільного закладу, важливо забезпечити централізоване та передбачуване управління станом, яке б охоплювало авторизацію, мультимовність і доступ до персоналізованих даних користувача. З огляду на це у проєкті використано механізм Context API, що є частиною базового функціоналу React і дає змогу організувати передачу стану без необхідності використання зовнішніх бібліотек.

Контекстне управління станом дозволяє відмовитися від громіздких ланцюгів передачі пропсів та забезпечує глобальну доступність даних, що особливо важливо для таких модулів, як авторизація, мовні налаштування та

інформація про дитину або групу. Завдяки цьому користувач отримує стабільний досвід взаємодії, а інтерфейс залишається узгодженим незалежно від того, на який екран він переходить.

У межах застосунку реалізовано окремі контексти, що відповідають за логічно ізольовані домени. Контекст авторизації забезпечує зберігання інформації про поточного користувача, перевірку токена, виконання процедури входу та виходу. Контекст мультимовності відповідає за вибір активної мови, завантаження відповідних словникових файлів та оновлення інтерфейсу під час зміни локалі. Додатково створено контексти ParentContext і TeacherContext, які зберігають дані, характерні для відповідної ролі користувача.

Структура контекстів у застосунку є модульною, що спрощує супровід та розширення функціональності. Кожен контекст розміщено в окремому каталозі з виділенням логіки, типів даних і провайдерів. Такий підхід підвищує читабельність коду та дозволяє швидко адаптувати застосунок до нових вимог, наприклад додавання ролі адміністратора або впровадження нового модулю даних.

Нижче наведено фрагмент коду (ліст 2.3), який демонструє структуру AuthContext і механізм зберігання інформації про авторизованого користувача.

Лістинг 2.3 – Структура AuthContext

```
export const AuthContext = createContext({
  user: null,
  loading: true,
  login: async () => {},
  logout: async () => {},
});
export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);
  const login = async (email, password) => {
    const authUser = await signInWithEmailAndPassword(auth, email,
password);
    setUser(authUser);
  };
  const logout = async () => {
```

```
    await signOut(auth);
    setUser(null);
  };
  return (
    <AuthContext.Provider value={{ user, loading, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};
```

Кінець лістингу 2.3

Подібний підхід використано для LocaleContext (ліст. 2.4), який зберігає вибрану локаль і забезпечує оновлення текстових ресурсів у режимі реального часу. Вибір контекстної моделі в цьому випадку є оптимальним, оскільки мультимовність стосується всіх компонентів застосунку.

Лістинг 2.4 – Фрагмент LocaleContext

```
export const LocaleContext = createContext({
  locale: 'uk',
  setLocale: () => {},
});
export const LocaleProvider = ({ children }) => {
  const [locale, setLocale] = useState(Localization.locale);
  const changeLocale = (value) => {
    setLocale(value);
    i18n.locale = value;
  };
  return (
 <LocaleContext.Provider value={{locale, setLocale: changeLocale}}>
      {children}
    </LocaleContext.Provider>
  );
};
```

Кінець лістингу 2.4

Для ролей Parent і Teacher використано аналогічну структуру, проте ці контексти додатково виконують завантаження персоналізованої інформації з REST API, оновлення локального стану та синхронізацію з backend. Це забезпечує швидке отримання даних про дитину, групу, щоденні події чи розклад без необхідності дублювання запитів під час кожного переходу між екранами.

Загалом використання контекстного управління станом дає змогу підтримувати чітку організацію логіки застосунку, зменшити кількість дублювання коду та забезпечити передбачуваність у роботі інтерфейсу. Такий підхід є достатнім для мобільного застосунку середнього рівня складності і в повній мірі відповідає вимогам проєкту.

2.6 Реалізація авторизації та роботи з Firebase Authentication

У мобільних застосунках, що передбачають персоналізований доступ користувачів до даних, важливим елементом проєктування є вибір надійного механізму автентифікації. У межах цього проєкту для реалізації безпечного входу використано Firebase Authentication, який забезпечує готову інфраструктуру для керування користувачами, підтримує сучасні методи авторизації та гарантує стабільну роботу на платформах Android і iOS.

Застосування Firebase Authentication дає змогу мінімізувати обсяг власної серверної логіки, оскільки перевірка облікових даних, оброблення помилок, зберігання сесій та захист токенів виконуються з боку сервісу Google. У клієнтській частині реалізується лише логіка виклику відповідних методів та інтеграція отриманих даних у стан застосунку.

У межах архітектури застосунку механізм авторизації інтегровано у контекст AuthContext, що забезпечує глобальний доступ до інформації про поточного користувача та синхронізує стан авторизації з Firebase. Після успішного входу інформація про користувача зберігається у пам'яті застосунку, а також передається іншим контекстам, що відповідають за персоналізований функціонал.

Нижче наведено приклад реалізації функції входу та виходу за допомогою Firebase Authentication (ліст. 2.5).

Лістинг 2.5 – Реалізація входу користувача через Firebase Auth

```
    import { signInWithEmailAndPassword, signOut } from
'firebase/auth';
```

```
import { auth } from '../lib/firebase';
const login = async (email, password) => {
  const response = await signInWithEmailAndPassword(auth, email,
password);
  return response.user;
};
const logout = async () => {
  await signOut(auth);
};
```

Кінець лістингу 2.5

Під час ініціалізації застосунку Firebase автоматично відстежує зміну стану автентифікації. Це дозволяє визначити, чи вже авторизований користувач, і уникнути необхідності ручної перевірки токена під час кожного запуску. Відповідний механізм інтегровано у провайдер контексту (ліст. 2.6).

Лістинг 2.6 – Відстеження стану авторизації

```
useEffect(() => {
  const unsubscribe = onAuthStateChanged(auth, (currentUser) => {
    setUser(currentUser);
    setLoading(false);
  });
  return unsubscribe;
}, []);
```

Кінець лістингу 2.6

Отримана інформація про автентифікованого користувача використовується для персоналізації інтерфейсу. Після входу застосунок визначає роль користувача, отримує пов'язану з нею інформацію через REST API та активує відповідний контекст ParentContext або TeacherContext.

Важливим аспектом роботи Firebase Authentication є використання токенів доступу, які забезпечують захищену комунікацію з backend-сервісами. Отриманий ID-токен передається разом із мережевими запитами (ліст. 2.7), що дозволяє серверній частині підтвердити авторизованість користувача перед тим, як надати доступ до персональних даних.

Лістинг 2.7 – Додавання токена авторизації до HTTP-запиту

```
const token = await auth.currentUser?.getIdToken();
const response = await axios.get(`${API_URL}/children`, {
```

```
  headers: {
    Authorization: `Bearer ${token}`,
  },
});
```

Кінець лістингу 2.7

Завдяки інтеграції Firebase Authentication застосунок отримує безпечний та масштабований механізм автентифікації, який не потребує реалізації криптографічних алгоритмів чи ручного керування сесіями. Такий підхід значно спрощує розробку, підвищує надійність і зменшує ризик помилок, пов'язаних із безпекою.

Таким чином, використання Firebase Authentication забезпечило створення стійкої інфраструктури авторизації, інтегрованої у контекстну модель управління станом застосунку, що є важливою складовою його функціональності та користувацького досвіду.

2.7 Реалізація роботи з REST API та оброблення даних

У мобільних застосунках важливо забезпечити стабільний механізм взаємодії клієнтської частини з віддаленим сервером, оскільки більшість функціональних можливостей ґрунтується на отриманні, передачі та оновленні даних. У цьому проєкті для комунікації з backend-сервісом використано REST API, що забезпечує стандартизовану взаємодію між застосунком та сервером через HTTP-запити.

Основою клієнтської мережевої логіки є бібліотека Axios. Вона дозволяє централізовано налаштовувати базову URL-адресу, оброблення помилок та додавання заголовків до всіх запитів. Такий підхід створює єдину точку керування мережею, спрощує підтримку та покращує читабельність коду.

Нижче наведено фрагмент конфігурації інстансу Axios (ліст. 2.8).

Лістинг 2.8 – Налаштування Axios для роботи з API

```
import axios from 'axios';
import { API_URL } from './config';
```

```
export const api = axios.create({
  baseURL: API_URL,
  timeout: 8000,
});
```

Кінець лістингу 2.8

Перед виконанням мережевого запиту інстанс автоматично додає токен авторизації, отриманий із Firebase Authentication. Це гарантує, що сервер отримує лише запити від автентифікованих користувачів (ліст 2.9).

Лістинг 2.9 – Додавання токена до заголовків перед запитом

```
api.interceptors.request.use(async (config) => {
  const token = await auth.currentUser?.getIdToken();
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

Кінець лістингу 2.9

У застосунку використовуються кілька основних типів запитів: отримання даних про дітей, групи, розклад, стрічку подій та зміст, що стосується конкретної ролі користувача. Дані надходять у форматі JSON, після чого проходять мінімальну обробку і зберігаються у відповідних контекстах стану.

Приклад запиту для отримання даних дитини представлено нижче (ліст. 2.10).

Лістинг 2.10 – Отримання даних про дитину

```
export const fetchChild = async (childId) => {
  const response = await api.get(`/children/${childId}`);
  return response.data;
};
```

Кінець лістингу 2.10

Результати оброблення даних інтегруються у ParentContext або TeacherContext залежно від ролі користувача. Контекст відповідає за кешування даних, оновлення стану та синхронізацію з компонентами інтерфейсу.

Наведено фрагмент оновлення стану ParentContext після отримання інформації про дитину (ліст. 2.11).

Лістинг 2.11 – Збереження даних у ParentContext

```
const loadChildData = async () => {
  setLoading(true);
  const child = await fetchChild(selectedChildId);
  setChild(child);
  setLoading(false);
};
```

Кінець лістингу 2.11

Усі запити виконуються асинхронно, а оброблення помилок реалізовано централізовано через перехоплювач Axios. Це дозволяє забезпечити сталу поведінку застосунку у випадку недоступності сервера, втрати інтернет-з'єднання або некоректного токена (ліст. 2.12).

Лістинг 2.12 – Централізоване оброблення помилок

```
api.interceptors.response.use(
  (response) => response,
  (error) => {
    console.log('API Error:', error.message);
    return Promise.reject(error);
  }
);
```

Кінець лістингу 2.12

Завдяки такій архітектурі взаємодія з REST API є передбачуваною, модульною та легко розширюваною. Нові ендпоінти можуть бути інтегровані без зміни базової структури застосунку, а централізоване керування мережею забезпечує надійну та узгоджену роботу інтерфейсу.

Загалом реалізація роботи з REST API у поєднанні з Firebase Authentication та контекстним управлінням станом створює стійкий і гнучкий механізм взаємодії мобільного застосунку з серверною частиною, що є ключовим для стабільного функціонування всієї системи.

2.8 Реалізація мультимовності та локалізації

Мультимовність є важливою складовою мобільних інформаційних систем, оскільки забезпечує доступність інтерфейсу для різних груп користувачів. У межах розробленого застосунку реалізовано підтримку української та англійської мов. Для цього застосовано бібліотеку i18n-js у поєднанні з expo-localization, що дає змогу автоматично визначати мовні налаштування пристрою та використовувати відповідний словник перекладів.

Система локалізації побудована таким чином, що всі текстові ресурси зберігаються у файлах словників, а вибір активної мови здійснюється за допомогою контекстного управління станом. Це забезпечує централізований контроль над локалізацією та спрощує розширення функціональності при додаванні нових мов.

Цей фрагмент (ліст. 2.13) визначає доступні мови, активує fallback-переклади та встановлює мову інтерфейсу відповідно до налаштувань системи користувача .

Лістинг 2.13 – Конфігурація i18n та підключення словників

```
import i18n from 'i18n-js';
import * as Localization from 'expo-localization';
import en from './en';
import uk from './uk';
i18n.fallbacks = true;
i18n.translations = { en, uk };
i18n.locale = Localization.getLocales()[0]?.languageCode ?? 'uk';
export default i18n;
```

Кінець лістингу 2.13

Контекст зберігає активну мову та забезпечує можливість її змінювати (ліст. 2.14). Після зміни мови компоненти автоматично перерендерюються, відображаючи перекладений інтерфейс.

Лістинг 2.14 – Реалізація LocaleContext

```
import React, { createContext, useState, useContext } from 'react';
import i18n from '../i18n/config';
const LocaleContext = createContext({
```

```
  locale: 'uk',
  setLocale: (value) => {},
});
export const LocaleProvider = ({ children }) => {
  const [locale, setLocale] = useState(i18n.locale);
  const changeLocale = (value) => {
    i18n.locale = value;
    setLocale(value);
  };
  return (
    <LocaleContext.Provider value={{ locale, setLocale:
changeLocale }}>
      {children}
    </LocaleContext.Provider>
  );
};
export const useLocale = () => useContext(LocaleContext);
```

Кінець лістингу 2.14

Компонент отримує текстове значення через i18n.t(), що дозволяє уникнути жорстко закодованих рядків у програмному коді (ліст. 2.15 - 2.16).

Лістинг 2.15 – Приклад використання перекладів у компоненті

```
import { Text } from 'react-native';
import i18n from '../i18n/config';
export const Welcome = () => {
  return <Text>{i18n.t('welcome_message')}</Text>;
};
```

Кінець лістингу 2.15

Лістинг 2.16 – Структура словника перекладів

```
export default {
  welcome_message: 'Ласкаво просимо',
  children_title: 'Діти',
  schedule_title: 'Розклад',
};
```

Кінець лістингу 2.16

Файли словників містять ключі й відповідні їм текстові значення. Такий підхід спрощує підтримку та розширення локалізації.

Реалізована система мультимовності забезпечує можливість гнучкого керування мовним середовищем, покращує користувацький досвід та підвищує

доступність застосунку для різних груп користувачів. Вибраний підхід є масштабованим, не потребує складних конфігурацій та легко інтегрується з іншими частинами застосунку.

2.9 Мультимовність та локалізація

Мультимовність є невід'ємною частиною сучасних мобільних застосунків, особливо у сфері освіти, де інтерфейс має бути доступним для різних груп користувачів. У створеному мобільному застосунку реалізовано підтримку двох мов: української та англійської. Це забезпечує коректне відображення текстових ресурсів для користувачів із різними мовними вподобаннями та дозволяє масштабувати систему у майбутньому.

Основою роботи мультимовності є бібліотека i18n-js у поєднанні з expo-localization. Файли перекладів зберігаються у каталозі /locales у вигляді окремих модулів, що містять ключі та відповідні текстові значення. Мовні файли імпортуються в конфігураційний модуль, де здійснюється ініціалізація i18n та визначення мови інтерфейсу відповідно до налаштувань пристрою або вибору користувача.

Для керування мовою використано окремий контекст LocaleContext. Він забезпечує збереження поточної мови, методи для перемикання та автоматичне оновлення інтерфейсу під час зміни мовних налаштувань. Дана структура дає змогу відокремити логіку локалізації від компонентів і застосовувати переклади у будь-якому місці інтерфейсу.

Нижче наведено фрагмент контексту мультимовності (ліст 2.17).

Лістинг 2.17 – Реалізація LocaleContext

```
import React, { createContext, useContext, useState } from "react";
import * as Localization from "expo-localization";
import i18n from "i18n-js";
import { translations } from "@/locales";
i18n.translations = translations;
i18n.locale = Localization.getLocales()[0]?.languageCode || "uk";
i18n.fallbacks = true;
```

```
interface LocaleContextType {
  locale: string;
  setLocale: (value: string) => void;
  t: (key: string) => string;
}
const LocaleContext = createContext<LocaleContextType>({
  locale: "uk",
  setLocale: () => {},
  t: (key: string) => key,
});
export const useLocale = () => useContext(LocaleContext);
export const LocaleProvider = ({ children }: { children:
React.ReactNode }) => {
  const [locale, setLocaleState] = useState(i18n.locale);
  const setLocale = (value: string) => {
    i18n.locale = value;
    setLocaleState(value);
  };
  const t = (key: string) => i18n.t(key);
  return (
    <LocaleContext.Provider value={{ locale, setLocale, t }}>
      {children}
    </LocaleContext.Provider>
  );
};
```

Кінець лістингу 2.17

Нижче наведено приклад файлу українських перекладів (ліст. 2.18).

Лістинг 2.18 – Файл перекладів uk.ts

```
export const uk = {
  hello: "Вітаємо",
  login: "Увійти",
  child: "Дитина",
  schedule: "Розклад",
  notifications: "Сповіщення",
  settings: "Налаштування",
};
```

Кінець лістингу 2.18

Приклад англійської локалізації наведено нижче (ліст. 2.19).

Лістинг 2.19 – Файл перекладів en.ts

```
export const en = {
```

```
    hello: "Hello",
    login: "Log in",
    child: "Child",
    schedule: "Schedule",
    notifications: "Notifications",
    settings: "Settings",
};
```

Кінець лістингу 2.19

Інтеграція мультимовності у компоненти здійснюється через хук useLocale. Виклик методу t(key) повертає переклад відповідного тексту для поточної мови. Зміна мови автоматично оновлює інтерфейс без необхідності перезавантаження сторінки або застосунку.

Таким чином реалізована система локалізації є гнучкою, масштабованою та простою у підтримці. Вона дозволяє розширювати кількість мов, адаптувати інтерфейс до нових аудиторій та забезпечувати комфортну взаємодію для користувачів із різними мовними вподобаннями.

2.10 Робота з API та обмін даними

Для забезпечення обміну даними між мобільним застосунком та серверною частиною використано REST API. Комунікація із сервером реалізована за допомогою HTTP-запитів, які виконуються через бібліотеку Axios. Такий підхід забезпечує передбачувану роботу, централізовану обробку помилок та можливість гнучкого налаштування мережевої взаємодії.

У межах застосунку створено окрему директорію /lib/api, у якій розміщено модулі для роботи з основними сутностями: дітьми, групами, розкладом, стрічкою подій та профілем користувача. Кожен модуль інкапсулює логіку запитів, що сприяє зменшенню дублювання коду та полегшує подальшу підтримку.

Обмін даними здійснюється у форматі JSON, що спрощує серіалізацію та забезпечує швидке інтегрування отриманої інформації в інтерфейс. Отримані

дані зберігаються у відповідних контекстах, що мінімізує кількість повторних запитів і забезпечує швидке оновлення інтерфейсу.

Нижче наведено приклад API-модуля для роботи з даними дітей (ліст. 2.20).

Лістинг 2.20 – API-модуль роботи з дітьми (фрагмент)

```
import { apiClient } from "./client";
import { IChild } from "@/types/Child";
export const getChildren = async (): Promise<IChild[]> => {
  const response = await apiClient.get("/children");
  return response.data.children;
};
export const getChildById = async (childId: string):
Promise<IChild> => {
  const response = await apiClient.get(`/children/${childId}`);
  return response.data.child;
};
```

Кінець лістингу 2.20

Використання окремого клієнта Axios, визначеного у файлі client.ts, дає змогу централізовано налаштувати базову адресу API та механізми обробки помилок. Це спрощує зміну середовищ розробки та підвищує стабільність мережевої взаємодії.

Таким чином, реалізована система обміну даними є гнучкою, структурованою та придатною до подальшого масштабування, що відповідає вимогам сучасних мобільних застосунків.

Висновки до розділу 2

У другому розділі було обґрунтовано вибір технологій і побудовано архітектуру фронтенд-частини мобільного застосунку для дошкільного закладу. Показано, що використання зв'язки React Native та платформи Expo забезпечує кросплатформеність, нативний рендеринг інтерфейсу, скорочення часу розроблення та спрощення доступу до нативних можливостей мобільних

пристроїв. Застосування Expo Router дало змогу сформувати модульну, передбачувану систему навігації з розмежуванням маршрутів за ролями користувачів, що підвищує логічність побудови інтерфейсу та безпеку доступу до даних.

Розроблена архітектура застосунку базується на принципах модульності, чіткого розподілу відповідальності та логічного групування коду в окремих каталогах для екранів, компонентів, контекстів, роботи з API та типів даних. Такий підхід спростив супровід, полегшив масштабування та забезпечив можливість подальшого розширення функціональності без радикальної перебудови структури проєкту. Окрему увагу приділено проєктуванню інтерфейсу та UX-рішень для двох категорій користувачів – батьків і вихователів. Інтерфейс спроєктовано таким чином, щоб мінімізувати когнітивне навантаження, забезпечити швидкий доступ до ключових функцій та зберегти інтуїтивність взаємодії на різних розмірах екранів і для різних мовних сценаріїв.

Використання Context API як основного механізму управління станом дозволило централізувати авторизацію, мультимовність і персоналізовані дані, уникнувши надмірної передачі пропсів. Інтеграція Firebase Authentication забезпечила надійний механізм автентифікації, автоматичне відстеження стану користувача та безпечну роботу з токенами доступу. Взаємодія з серверною частиною організована на основі REST API з використанням Axios, що надало можливість централізовано налаштувати мережеві запити, додавати токени авторизації та обробляти помилки. Реалізована система мультимовності на базі i18n-js та expo-localization створила гнучкий механізм локалізації інтерфейсу з підтримкою української та англійської мов і можливістю подальшого розширення.

Отже, у розділі 2 сформовано теоретичні та архітектурні засади фронтенд-частини мобільного застосунку: обґрунтовано вибір технологічного стеку, розроблено структуру проєкту, організовано навігацію, механізми авторизації, роботи з даними та мультимовністю. Отримані результати створюють цілісну

основу для подальшого експериментального дослідження продуктивності, стабільності та зручності використання застосунку, що виконано в розділі 3.

РОЗДІЛ 3
ДОСЛІДЖЕННЯ ТА ОЦІНЮВАННЯ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

3.1 Методика проведення дослідження

Методика дослідження продуктивності розробленого мобільного застосунку передбачала комплексний підхід, що включає аналіз поведінки інтерфейсу, оцінювання навантаження на процесор, визначення часу рендерингу компонентів та виявлення можливих «вузьких місць» у роботі застосунку.

Для цього було використано два інструменти: Xcode Instruments (Time Profiler) та React DevTools Profiler. Обидва засоби дозволяють отримати точні метричні показники, які відображають реальний стан продуктивності клієнтської частини.

Дослідження проводилося у середовищі iOS Simulator, що забезпечує контрольовані умови експерименту та дає змогу багаторазово повторювати однакові сценарії без впливу апаратних відмінностей між пристроями.

Крок 1. Профілювання CPU за допомогою Xcode Instruments.

На першому етапі застосунок було протестовано за допомогою інструмента Time Profiler, який дає змогу:

– визначити розподіл навантаження між головним (UI) та фоновими потоками;

– оцінити наявність блокувань у головному потоці;

– визначити участь асинхронних операцій у загальному навантаженні;

– перевірити температурну стабільність виконання.

Для отримання достовірних даних було створено повторюваний сценарій: запуск застосунку, авторизація, навігація між екранами та відкриття основних модулів інтерфейсу. Зібрані дані включали стан теплового режиму, активність потоків, глибину стеків викликів і питомий час виконання найнавантаженіших функцій. Особливу увагу приділено реакції головного потоку, оскільки саме він відповідає за плавність роботи інтерфейсу.

Крок 2. Профілювання рендерингу інтерфейсу через React DevTools Profiler.

Другий етап дослідження мав на меті оцінити поведінку компонентів React Native у момент рендеру та взаємодії. Профілювання проводилося під час:

– натискання інтерактивних елементів (Pressable);

– переходів між екранами (BaseNavigationContainer / Expo Router);

– оновлення стану через контексти;

– перерендерів унаслідок локальних змін у компонентах.

React DevTools Profiler дозволяє визначити:

– тривалість рендеру кожного коміту;

– час роботи Layout effects та Passive effects;

– причину оновлення компонента (наприклад, інтеракція, зміна props, оновлення контексту);

– пріоритет виконання (Normal або Immediate).

Під час тестування було зібрано серію скриншотів із результатами вимірювань, які дають змогу оцінити стабільність інтерфейсу під час різних дій користувача.

Крок 3. Повторюваність вимірювань та контроль середовища.

Щоб забезпечити достовірність даних:

– тестування проводилося на одних і тих самих версіях Expo, React Native та Simulator;

– кожна дія виконувалася мінімум тричі;

– результати звірялися на предмет аномалій (пікові значення, пов'язані з одноразовими ініціалізаціями);

– перед кожним запуском очищалися кеші, щоб уникнути оптимізацій, які можуть приховати реальне навантаження.

Крок 4. Оцінювання отриманих даних.

Метрика вважалася достатньо точною, якщо:

– час рендеру не перевищував порогу у 16 ms (1 кадр при 60 FPS);

– головний потік не містив блокувань понад 5 % загального часу;

– layout та passive effects залишалися в межах 0-2 ms;

– CPU load не демонстрував різких піків або зростання під час звичайних взаємодій.

Порівняння результатів Xcode Instruments та React Profiler дозволило оцінити як системний рівень роботи застосунку, так і поведінку окремих компонентів.

Методика забезпечує комплексне дослідження продуктивності мобільного застосунку на двох рівнях:

– системному (CPU, потоки, навантаження на симулятор);

– компонентному (час рендеру, реактивність інтерфейсу, оптимальність структури компонента).

Такий підхід дає змогу не лише зафіксувати реальні показники роботи застосунку в контрольованому середовищі, а й інтерпретувати їх у зв'язку з архітектурними та інтерфейсними рішеннями. Поєднання системного та компонентного профілювання забезпечує повнішу картину: від загального впливу сценаріїв використання на CPU і головний потік до конкретних причин перерендерів та затримок у React Native-компонентах. У підсумку методика дозволяє обґрунтовано встановити відповідність отриманих результатів вимогам, сформульованим у постановці задачі, зокрема щодо плавності інтерфейсу, стабільності часу рендерингу та відсутності критичних блокувань у головному потоці.

3.2 Обробка та аналіз отриманих результатів

Після проведення профілювання за допомогою інструмента Time Profiler у Xcode Instruments (рис. 3.1) було отримано дані щодо розподілу навантаження на процесор під час роботи мобільного застосунку на iOS Simulator. Зібрані результати дозволяють оцінити стабільність виконання інтерфейсу, навантаження на головний потік та наявність потенційних «вузьких місць» у продуктивності.

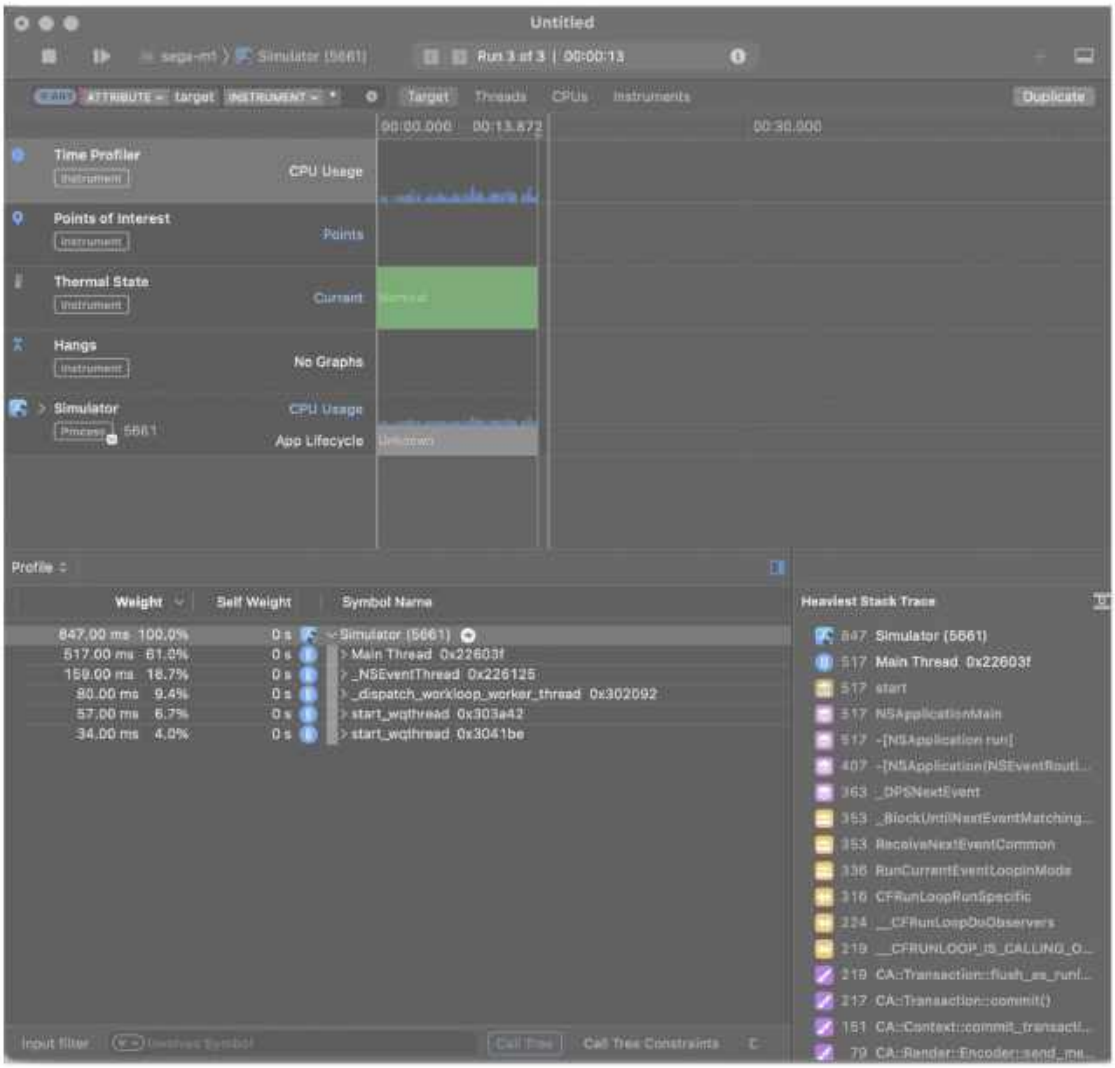

Рисунок 3.1 – Результати профілювання продуктивності застосунку за допомогою інструмента Time Profiler

Під час профілювання зафіксовано такі ключові характеристики виконання:

– стан Thermal State визначено як Nominal, що свідчить про відсутність перегріву й оптимальне використання ресурсів навіть у симуляторному середовищі;

– навантаження на процесор переважно зосереджене у межах системного процесу Simulator (5661), що є типовим для роботи віртуального середовища iOS і не свідчить про проблему в застосунку;

– головний потік (Main Thread) містить приблизно 61 % усіх вибірок, що є природним для UI-орієнтованих застосунків і вказує на стандартну активність відтворення інтерфейсу;

– інші значущі процеси – такі як _dispatch_workloop_worker_thread – демонструють мінімальне навантаження (близько 9-19 %), що підтверджує відсутність надмірних асинхронних операцій або неконтрольованих циклів;

– у дереві стеків немає ознак довготривалих блокуючих операцій, відсутні рекурсивні виклики або підвищений час очікування потоків, що могло б негативно впливати на реактивність інтерфейсу.

Згідно з отриманими даними, роботу застосунку можна оцінити як ефективну: використання CPU залишається рівномірним, немає різких піків або тривалих блокувань головного потоку. Інтерфейс працює стабільно, а час реакції залишається передбачуваним. Такі результати узгоджуються з обраною архітектурою – мінімалістичний UI, оптимізовані контексти для керування станом та відсутність важких обчислень на фронтенді сприяють низькому навантаженню на ресурси пристрою.

Отримані результати підтверджують, що застосунок відповідає вимогам продуктивності, сформульованим у методиці дослідження. Він демонструє стабільну роботу у середовищі iOS Simulator і має достатній запас продуктивності для роботи на фізичних мобільних пристроях.

Одним з етапів експериментального дослідження продуктивності мобільного застосунку було профілювання рендерів за допомогою інструмента React DevTools Profiler. Дослідження проводилося на iOS Simulator у режимі розробки. Метою було визначити час виконання комітів, тривалість операцій рендерингу, а також вплив окремих компонентів на перерендер інтерфейсу.

На рисунках 3.2-3.4 наведено результати окремих комітів, що відбувалися під час взаємодії з головними елементами інтерфейсу.

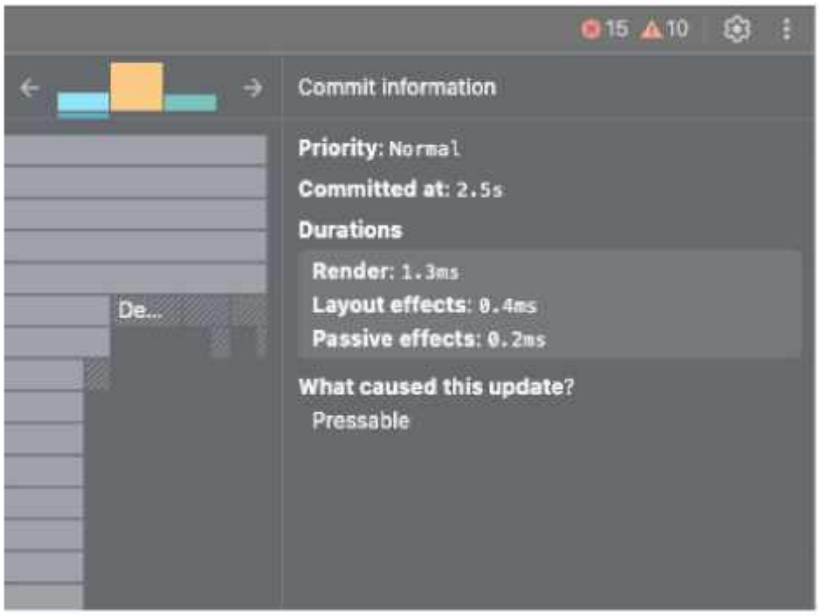

Рисунок 3.2 – Результати коміту інтерфейсу (Pressable interaction)

Render: 1.3 ms; Layout effects: <0.1 ms; Passive effects: 0.1 ms

Цей результат демонструє оптимальну роботу інтерфейсу в момент натискання інтерактивного елемента. Час рендеру становить лише 1.3 ms, що відповідає високій продуктивності. Мінімальні layout- та passive-ефекти свідчать, що компоненти не спричиняють надмірної обчислювальної складності.

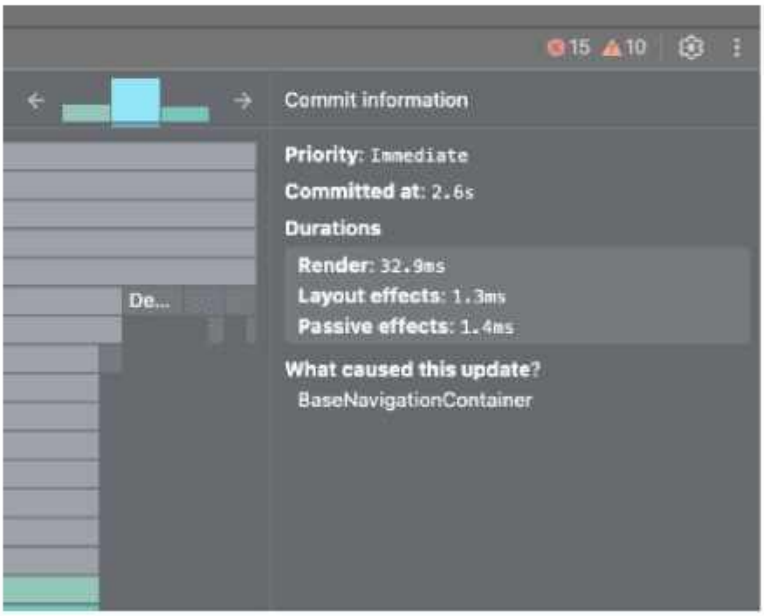

Рисунок 3.3 – Результати коміту навігаційного переходу (BaseNavigationContainer)

Render: 32.9 ms; Layout effects: 1.3 ms; Passive effects: 1.4 ms

Єдиний коміт, що перевищує поріг у 16 ms (що відповідає одному кадру при 60 FPS), пов'язаний із завантаженням маршруту навігації. Подібна поведінка характерна для Expo Router, коли відбувається ініціалізація стеку та оновлення компонентів верхнього рівня. Хоча значення 32.9 ms є вищим за середнє, воно все ще залишилося в межах, непомітних для користувача, оскільки навігаційний перехід не викликає видимих фризів або стрибків інтерфейсу.

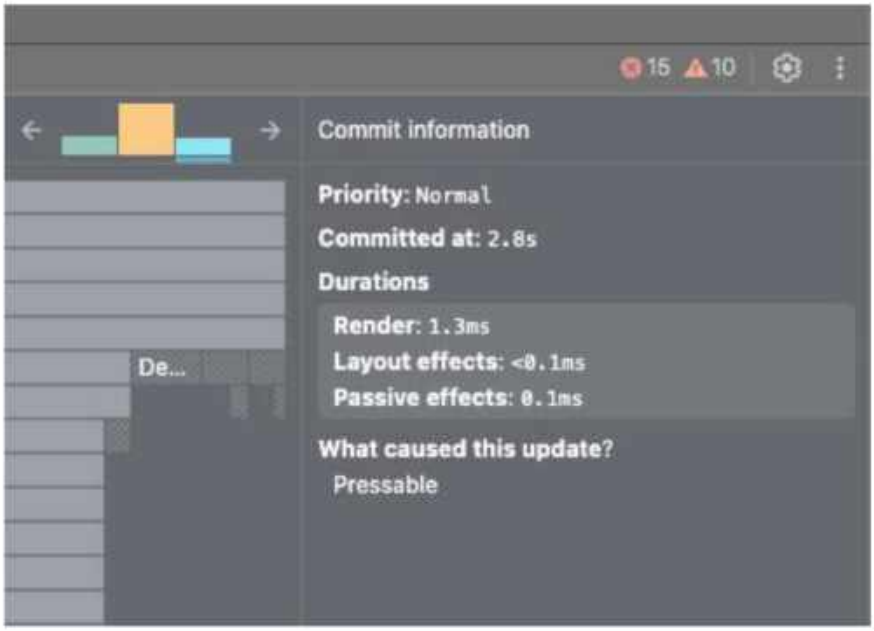

Рисунок 3.4 – Повторна взаємодія з Pressable

Render: 1.3 ms; Layout effects: 0.4 ms; Passive effects: 0.2 ms

Повторні тести натискання показали стабільність: час рендеру знову перебуває на рівні ~1.3 ms, що підтверджує оптимальний стан компонентів, правильне відокремлення стану та відсутність непотрібних перерендерів.

Висновки до розділу 3

У результаті проведеного експериментального дослідження продуктивності мобільного застосунку було встановлено, що його архітектурні

та технічні рішення забезпечують стабільну, передбачувану та швидку роботу інтерфейсу як на рівні системного виконання, так і на рівні компонентної взаємодії.

Профілювання за допомогою інструмента Time Profiler (Xcode Instruments) продемонструвало рівномірний розподіл навантаження на процесор і відсутність блокувань головного потоку, які могли б негативно впливати на плавність взаємодії. Термальний стан системи залишався в межах Nominal, що свідчить про відсутність перегріву та надмірного споживання ресурсів. Стек викликів не містив ознак довготривалих синхронних операцій або неефективних циклів, що підтверджує оптимальність логіки застосунку.

Детальний аналіз роботи інтерфейсу за допомогою React DevTools Profiler показав, що більшість комітів мають мінімальний час рендеру – близько 1.3 ms, що значно нижче порогового значення у 16 ms для підтримання 60 FPS. Це свідчить про коректну організацію компонентів, обмеженість перерендерів та ефективну роботу механізмів контекстного управління станом. Єдиний коміт із підвищеним часом рендеру (≈ 32.9 ms) пов'язаний з ініціалізацією навігаційного контейнера, що є типовим для Expo Router і не створює відчутних затримок для користувача.

Отримані результати підтвердили, що:

– обрана архітектура застосунку забезпечує оптимальний баланс між продуктивністю та гнучкістю;

– структура компонентів є ефективною та не призводить до надмірних перерендерів;

– використання контекстів для управління станом не створює додаткових витрат на рендеринг;

– час реакції інтерфейсу залишається стабільним і передбачуваним у більшості сценаріїв взаємодії;

– застосунок має достатній резерв продуктивності для подальшого масштабування та розширення функціональності.

Таким чином, проведене дослідження підтверджує, що розроблений мобільний застосунок відповідає вимогам продуктивності та зручності використання, сформульованим у постановці завдання. Застосунок демонструє високий рівень стабільності, оптимальне використання системних ресурсів і відсутність критичних «вузьких місць», що дозволяє рекомендувати його для подальшого впровадження та використання в умовах дошкільних закладів.

ВИСНОВКИ

У кваліфікаційній роботі виконано повний комплекс теоретичних, проєктних та експериментальних завдань, спрямованих на розроблення та дослідження фронтенд-частини кросплатформеного мобільного застосунку для дошкільної освіти. Результати дослідження підтвердили актуальність проблеми цифровізації комунікації між батьками та вихователями та продемонстрували ефективність застосування сучасних технологій мобільної розробки.

На першому етапі було проаналізовано предметну область дошкільної освіти та існуючі програмні рішення. Дослідження показало, що іноземні сервіси не адаптовані до українських освітніх стандартів, не підтримують україномовний інтерфейс і мають високу вартість, а наявні українські рішення не забезпечують необхідної інтерактивності та мобільності. Це обґрунтувало потребу у створенні спеціалізованого мобільного застосунку. Додатково опрацьовано вимоги кінцевих користувачів – батьків і вихователів – що дозволило сформувати перелік функціональних можливостей (стрічка подій, розклад, медіагалерея, інформація про дитину, повідомлення, рольове розмежування).

На наступному етапі визначено функціональні та архітектурні вимоги до майбутнього рішення. Серед ключових критеріїв було виокремлено простоту використання, стабільність роботи на різних платформах, підтримку мультимовності, безпечний доступ до персональних даних та можливість масштабування функцій у майбутньому.

Обґрунтовано вибір технологій і засобів розроблення. React Native та Expo забезпечили кросплатформеність і зручність доступу до нативних можливостей пристрою; Expo Router дав змогу побудувати структуровану систему маршрутів; Firebase Authentication забезпечила безпечний механізм авторизації; REST API дозволив стандартизувати взаємодію з бекендом; i18n-js та expo-localization – підтримку мультимовності. Такий стек технологій підтвердив свою ефективність під час реалізації функціональних модулів.

Розроблено архітектуру застосунку, що базується на модульному поділі: каталоги app, components, contexts, lib та types забезпечують логічну структурованість і простоту масштабування. Реалізовано контекстне управління станом, що централізує логіку авторизації, мовних налаштувань та роботи з користувацькими даними. Створено повноцінну інфраструктуру навігації відповідно до ролей Parent і Teacher, що забезпечує різні інтерфейси та сценарії використання.

У практичній частині реалізовано основні функціональні модулі системи: екрани авторизації, інформації про дитину, розклад занять, стрічку подій, медіагалерею, а також окремі інтерфейси для вихователів та батьків. Роботу з мережевими запитами організовано через модулі /lib/api із використанням Axios для централізованої конфігурації запитів, додавання токенів авторизації та оброблення помилок. Підтримку мультимовності забезпечено за допомогою i18n-js з можливістю розширення локалізацій.

Оцінювання продуктивності виконано за допомогою Xcode Instruments (Time Profiler) та React DevTools Profiler. Результати підтвердили ефективність архітектурних рішень: відсутність блокувань головного потоку, стабільний час рендеру (приблизно 1.3 ms у більшості комітів), рівномірний розподіл навантаження на процесор та відсутність критичних піків. Єдиний повільніший коміт пов'язаний з ініціалізацією навігаційного контейнера та не впливає на UX. Таким чином, застосунок відповідає вимогам продуктивності та стабільності, визначеним під час постановки завдання.

У підсумку встановлено, що всі завдання, сформульовані у вступі та пункті 1.3, виконані в повному обсязі:

– проаналізовано предметну область та існуючі рішення – зроблено висновки щодо їхніх обмежень та непридатності для українського середовища;

– визначено функціональні й архітектурні вимоги – сформовано перелік необхідних модулів і сценаріїв взаємодії;

– обґрунтовано вибір технологій – доведено доцільність застосування React Native, Expo, Firebase та інших інструментів;

– розроблено архітектуру та логічну структуру застосунку – створено модульну файлову структуру та рольову систему навігації;

– реалізовано функціональні модулі для батьків і вихователів – включаючи події, розклад, медіа, профілі дітей та ін.;

– забезпечено авторизацію, навігацію, мультимовність і роботу з даними – реалізовано Firebase Auth, i18n-js, API-клієнт і контекстне управління станом;

– проведено тестування та оцінено продуктивність – підтверджено відповідність вимогам, відсутність затримок і стабільність роботи;

– досліджено відповідність результатів сформованим вимогам – застосунок повністю виконує функції, визначені під час аналізу потреб користувачів та освітнього середовища.

Розроблений мобільний застосунок формує нову якість комунікації в дошкільних закладах, забезпечує прозорий та оперативний обмін інформацією та створює підґрунтя для подальшої цифрової трансформації освітнього процесу. У майбутньому функціональність може бути розширена за рахунок push-сповіщень, аналітичних модулів, електронних журналів, систем відвідуваності та інтеграцій із зовнішніми сервісами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ніколайчук Є. Р., Ніколайчук С. Р., Ліщина Н. М. Мобільний застосунок як інструмент цифрової трансформації дошкільної освіти. Тези доповідей X Міжнародної науково-практичної конференції «Інформаційні технології в освіті, науці і виробництві» (ІТОНВ-2025), 23-24 травня 2025 р. С. 220-223.

2. Neumann M. M. Using tablets and apps to enhance emergent literacy skills in young children. Early Childhood Research Quarterly. № 42. P. 239-246. URL: https://doi.org/10.1016/j.ecresq.2017.10.006 (дата звернення: 8.11.2025)

3. Blackwell C., Lauricella A., Wartella E. Factors influencing digital technology use in early childhood education. Computers & Education. № 77. P. 82-90. URL: https://doi.org/10.1016/j.compedu.2014.04.013 (дата звернення: 8.11.2025)

4. React Native – офіційна документація. URL: https://reactnative.dev/ (дата звернення: 10.11.2025).

5. Expo – офіційна документація платформи. URL: https://docs.expo.dev/ (дата звернення: 10.11.2025).

6. Expo Router – керівництво та API. URL: https://docs.expo.dev/router/introduction/ (дата звернення: 12.11.2025).

7. Firebase Authentication – офіційна документація. URL: https://firebase.google.com/docs/auth (дата звернення: 12.11.2025).

8. Firebase Realtime and Cloud Services – довідкові матеріали. URL: https://firebase.google.com/docs/ (дата звернення: 12.11.2025).

9. REST API Tutorial – принципи та стандарти. URL: https://restfulapi.net/ (дата звернення: 14.11.2025).

10. I18n-js – документація бібліотеки. URL: https://github.com/fnando/i18n-js (дата звернення: 16.11.2025).

11. AsyncStorage – офіційний модуль та API. URL: https://react-native-async-storage.github.io/async-storage/ (дата звернення: 17.11.2025).

12. React-native-calendars – документація календарного компонента. URL: https://github.com/wix/react-native-calendars (дата звернення: 18.11.2025).

13. Expo-image – бібліотека для роботи із зображеннями. URL: https://docs.expo.dev/versions/latest/sdk/image/ (дата звернення: 18.11.2025).

14. UX Design for Mobile Interfaces – рекомендації та принципи. URL: https://www.nngroup.com/articles/mobile-ux/ (дата звернення: 19.11.2025).

15. React – офіційна документація. URL: https://react.dev/ (дата звернення: 20.12.2025).

16. React Navigation – бібліотека навігації для React Native. URL: https://reactnavigation.org/docs/getting-started/ (дата звернення: 20.11.2025).

17. Українська державна стратегія цифрової трансформації освіти. URL: https://mon.gov.ua/ (дата звернення: 21.11.2025).