

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет _____ комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
(повна назва)

Кафедра _____ програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти _____ перший (бакалаврський)

Програмна система для контролю логістичного забезпечення війська.

Мобільний застосунок.
(тема)

Виконав:

Студент 4 курсу, групи ПЗП-21-5

Нікіта ІВАКІН
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. каф. ПІ Олексій НАЗАРОВ
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри

Кирило СМЕЛЯКОВ
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет _____ комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)

Кафедра _____ програмної інженерії

Рівень вищої освіти _____ перший (бакалаврський)

Спеціальність _____ 121 – Інженерія програмного забезпечення

Тип програми _____ Освітньо-професійна

Освітня програма _____ Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:

Зав. Кафедри

_____ (підпис)

«_____» _____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту _____ Івакіну Нікіті Максимовичу
(прізвище, ім'я по батькові)

1. Тема роботи _____ Програмна система для контролю логістичного забезпечення війська. Бекенд.

Затверджена наказом по університету від _____ 19.05.2025р. № 397 Ст

2. Термін подання студентом роботи до екзаменаційної комісії _____ 12.06.2025

3. Вихідні дані до роботи _____ Розробити серверну частину програмної системи для контролю логістичного забезпечення війська з урахуванням ролей користувачів, захисту даних, обробки логістичних запитів і можливості масштабування системи. Реалізація має базуватись на сучасних підходах до створення розподілених веб-сервісів.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	04.04.2025	виконано
2	Створення специфікації ПЗ	10.04.2025	виконано
3	Проектування ПЗ	15.04.2025	виконано
4	Розробка ПЗ	20.04.2025	виконано
5	Тестування ПЗ	20.05.2025	виконано
6	Оформлення пояснювальної записки	27.05.2025	виконано
7	Підготовка презентації та доповіді	30.05.2025	виконано
8	Попередній захист	09.06.2025	виконано
9	Нормоконтроль, рецензування	07.06.2025	виконано
10	Здача роботи у електронний архів	07.06.2025	виконано
11	Допуск до захисту у зав. кафедри	09.06.2025	виконано

Дата видачі завдання « 04 » « квітня » 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. кафедри ІІ Олексій НАЗАРОВ
(підпис) (посада, Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра: 57 с., 5 джерел.

ЛОГІСТИКА, РЕСУРС, ІНФОРМАЦІЙНА СИСТЕМА, ANDROID, KOTLIN, MVVM, МОБІЛЬНИЙ ЗАСТОСУНОК, БОЙОВА ГРУПА, ЛОГІСТИЧНИЙ ЗАПИТ, ІНТЕРФЕЙС, БЕЗПЕКА, REST API,.

Об'єкт розробки – мобільний застосунок для системи військового логістичного управління «UA Logistics».

Мета розробки – створити Android-додаток, що дозволяє військовим підрозділам ефективно керувати ресурсами, формувати запити на постачання та взаємодіяти з логістичною системою в реальному часі.

Метод реалізації – Android Studio, мова Kotlin, архітектурний шаблон MVVM, Jetpack-бібліотеки (ViewModel, StateFlow, DataStore), REST API для обміну з сервером.

Результат розробки – мобільний застосунок, який забезпечує:

- авторизацію користувачів за ролями.
- перегляд і оновлення ресурсів.
- додавання нових ресурсів.
- перегляд місій.
- створення логістичних запитів та місій.
- відправку критичних звернень.
- швидкий доступ до даних навіть у польових умовах завдяки адаптивному інтерфейсу.середовищі.

ABSTRACT

LOGISTICS, RESOURCE, INFORMATION SYSTEM, ANDROID, KOTLIN, MVVM, MOBILE APPLICATION, BATTLE GROUP, LOGISTICS REQUEST, INTERFACE, SECURITY, REST API,.

The object of development is a mobile application for the military logistics management system "UA Logistics".

The purpose of the development is to create an Android application that allows military units to effectively manage resources, form supply requests and interact with the logistics system in real time.

Implementation method - Android Studio, Kotlin language, MVVM architectural pattern, Jetpack libraries (ViewModel, StateFlow, DataStore), REST API for exchange with the server.

The result of the development is a mobile application that provides:

- user authorization by roles.
- viewing and updating resources.
- adding new resources.
- viewing missions.
- creation of logistics requests and missions.
- sending critical requests.
- quick access to data even in field conditions thanks to the adaptive interface.environment.

ЗМІСТ

ВСТУП	8
ПЕРЕЛІК СКОРОЧЕНЬ.....	11
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	12
1.1 Аналіз предметної галузі та постановка задачі.....	12
1.2 Цільова аудиторія.....	13
1.3 Огляд систем аналогів	15
1.4 Монетизація	18
1.5 Постановка задачі.....	19
2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ.....	21
2.1 Функціональні вимоги.....	22
2.2 Нефункціональні вимоги.....	24
2.3 Організація командної роботи та методологія розробки	28
3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ..	30
3.1 UML проєктування ПЗ.....	30
3.2 Архітектура системи.....	32
3.3 Проєктування структури зберігання даних	33
3.4 Обмеження та вимоги	36
3.5 Забезпечення якості	39
3.6 Керування проектом.....	41
3.7 Тестування	42
4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ	45
5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	52
6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	54
ВИСНОВКИ.....	56
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	58
ДОДАТОК А	59
ДОДАТОК Б.....	60

ДОДАТОК В	61
ДОДАТОК Г	62
ДОДАТОК Ґ	63
ДОДАТОК Д	70

ВСТУП

У сучасному світі програмні системи відіграють ключову роль у різних сферах життя – вони сприяють автоматизації процесів, зменшують витрати часу, підвищують точність і загальну ефективність виконання завдань. Логістичні операції не є винятком: впровадження спеціалізованих цифрових рішень значно спрощує управління процесами та мінімізує кількість помилок, які можуть виникати під час ручного виконання робіт. В умовах повномасштабної війни, що триває в Україні, виявлено низку критичних проблем у сфері логістичного забезпечення військ. Серед них – недостатній рівень автоматизації, що призводить до надмірної ручної роботи та супутніх помилок, обмежена прозорість процесів, а також складність контролю за розподілом і використанням ресурсів. Це створює передумови для зловживань та неефективного використання наявних запасів. Крім того, актуальною є реалізація оборонної реформи, орієнтованої на посилення обороноздатності країни та інтеграцію з силами НАТО й ЄС [3]. Усе це зумовлює нагальну потребу у створенні інноваційної програмної системи для ефективного та прозорого логістичного забезпечення Збройних сил України.

Розробка та впровадження програмної системи для контролю логістичного забезпечення військ є надзвичайно актуальною задачею [4]. Україна гостро потребує сучасних цифрових рішень, технічного оснащення, смарт-пристроїв та інновацій для зміцнення обороноздатності та ефективної протидії агресору. Проведення такої системи дозволить значно покращити контроль за логістичними процесами, забезпечити прозорість у звітності щодо використання ресурсів, а також організувати електронне зберігання даних про їхній розподіл і постачання між бойовими підрозділами. Крім того, система забезпечить швидкий та надійний обмін інформацією, що критично важливо для координації дій на

полі бою. Завдяки цьому воєнне керівництво Збройних сил України зможе більш ефективно управляти логістикою, уникати помилок і плутанини, а також оперативно реагувати на змінені потреби бойових частин.

Метою даної роботи є проєктування та розробка програмної системи для контролю логістичного забезпечення військ. Для досягнення цієї мети планується створення комплексної системи, що включатиме чотири основні компоненти: серверний застосунок, веб-клієнт, мобільний застосунок.

Розробка такої системи забезпечить низку важливих переваг: прозорість у розподілі ресурсів, точний облік, зниження ризиків розкрадань, прискорення виконання наказів на постачання та загальне підвищення ефективності логістичних процесів.

У фокусі даної роботи перебуває мобільний застосунок, який виконує роль повноцінного клієнтського інтерфейсу до серверної частини логістичної системи. Саме через нього кінцеві користувачі – представники бойових підрозділів – можуть здійснювати щоденну взаємодію з платформою: реєструвати ресурси, створювати запити на постачання, переглядати призначені місії та інформувати про критичні ситуації в режимі реального часу.

Мобільна частина проєкту була реалізована для Android-платформи з урахуванням вимог до надійності, адаптивності та зручності користування. Архітектура клієнта побудована з використанням сучасного стеку Android-розробки (MVVM, LiveData, Retrofit, ViewBinding), що дозволяє досягти високої модульності, масштабованості та тестованості коду. Взаємодія з сервером здійснюється через REST API, що дає змогу легко синхронізувати дії користувача з даними, збереженими на бекенді.

Ключові функції мобільного застосунку включають: авторизацію користувача, роботу з ресурсами підрозділу (додавання, редагування, видалення), створення логістичних запитів, а також перегляд активних місій.

Окрема увага приділена реалізації механізмів валідації форм, UI/UX-деталей, таких як анімації, відображення помилок, індикація дій та перехід між екранами.

Одним з викликів у розробці стало забезпечення стабільної роботи інтерфейсу в умовах обмеженого з'єднання, що реалізовано через належну обробку помилок, валідацію вводу та візуальне інформування користувача про статус виконання операцій. Також враховано особливості військового контексту, де критичною є швидкість взаємодії з інтерфейсом, мінімізація кількості дій для досягнення результату, та підтримка зрозумілої навігації.

Розроблений мобільний застосунок є важливою складовою частиною комплексного рішення, націленого на покращення логістичного забезпечення бойових підрозділів.

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

HTTP – HyperText Transfer Protocol

JWT – JSON Web Token

REST – Representational State Transfer

MVVM – Model-View-ViewModel

HTTP – HyperText Transfer Protocol

CI/CD – Continuous Integration/Continuous Delivery

AES – Advanced Encryption Standard

UI/UX – User Interface/User Experience

APK – Android Application Package

DTO – Data Transfer Object

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної галузі та постановка задачі

Логістичне забезпечення є однією з ключових складових бойової ефективності армії. Успішне виконання завдань безпосередньо залежить від швидкого й точного постачання необхідних ресурсів – від амуніції до технічних засобів. Традиційні методи управління (ручний облік, застарілі системи) втрачають свою ефективність в умовах сучасних бойових дій, де потрібна висока швидкість обробки запитів і мінімізація помилок [1].

Автоматизація логістичних процесів дозволяє підвищити точність обліку, пришвидшити реагування на запити та зменшити навантаження на командний склад. Сучасні цифрові рішення дають змогу інтегрувати різні джерела інформації, здійснювати оперативний моніторинг стану ресурсів та ухвалювати рішення в реальному часі.

У межах теми проекту – розробки мобільного застосунку для контролю логістичного забезпечення військових підрозділів – формуються специфічні вимоги до мобільного доступу, взаємодії з серверною частиною, обробки чутливих даних та безпечної взаємодії з користувачем. Мобільний застосунок є одним із ключових інтерфейсів системи, який використовується командирами підрозділів безпосередньо в польових умовах, де часто обмежено доступ до стаціонарної техніки.

На відміну від традиційних ERP- або логістичних систем, розробка мобільного додатку у військовому контексті вимагає підвищеної надійності, офлайн-доступу до певних функцій, а також інтуїтивно зрозумілого інтерфейсу, оскільки кінцевими користувачами виступають командири та оперативний персонал з різним рівнем цифрової грамотності.

Особливістю предметної області є те, що мобільний застосунок повинен підтримувати:

- створення та перегляд логістичних запитів.
- підтвердження виконання доставки ресурсів.
- актуалізацію інформації про стан підрозділу.
- перегляд актуальних місій та можливості створення та перегляду їх логістичних запитів.

Важливо також забезпечити можливість динамічного оновлення UI, з урахуванням статусу запитів (очікується, відхилено, виконано) та швидкого реагування на зміни у внутрішній структурі бойової групи.

Окремий виклик становить безпека: усі операції мають бути авторизованими, паролі зашифрованими, а дані передаватися лише через захищені канали зв'язку. Приділено увагу локалізації (українська й англійська мови) та адаптації під мобільні пристрої з Android 6.0 і вище.

Таким чином, предметна область вимагає від мобільного застосунку стабільної роботи в умовах обмеженого зв'язку, підтримки багаторівневої ролі користувачів, швидкої взаємодії з сервером, а також високої продуктивності й безпеки – що й визначає вимоги до його архітектури, інтерфейсу та логіки [5].

1.2 Цільова аудиторія

Основна цільова аудиторія мобільного застосунку «UA Logistics» охоплює наступні групи користувачів:

а) Командири бойових підрозділів Збройних Сил України та Національної Гвардії. Це особи, що приймають рішення безпосередньо в польових умовах. У їх обов'язки входить формування запитів на постачання, контроль логістики та моніторинг забезпечення підрозділів. Для них застосунок забезпечує

оперативний доступ до ресурсної інформації, створення запитів та керування статусами місій. Вік користувачів цієї категорії переважно варіюється від 20 до 50 років.

б) Логісти та відповідальні за постачання. Ця категорія користувачів відповідає за виконання логістичних запитів, актуалізацію складів та звітність щодо виданих ресурсів. Застосунок надає їм зручні інструменти для підтвердження доставок, перегляду запитів і швидкої взаємодії з командирами.

в) Оперативний персонал та молодший командний склад. До цієї групи належать військовослужбовці, які беруть участь у місіях, перевіряють стан ресурсів на місцях та забезпечують виконання оперативних завдань. Вони використовують застосунок для ознайомлення з місіями перегляду наданих ресурсів та взаємодії зі старшим командуванням.

г) Адміністративний персонал і розробники військової інфраструктури. Представники цієї групи використовують застосунок для тестування, аналізу логістичних процесів, а також контролю доступів і ролей користувачів. Для них важливими є безпечність, масштабованість і підтримка централізованого адміністрування.

Необхідно розробити програмну систему, яка буде задовольняти потреби усіх вище перерахованих, матиме зрозумілий та адаптивний інтерфейс. Також потрібно зробити систему адаптивною до різних операційних систем мобільних застосунків та персональних комп'ютерів, що дозволить зробити систему більш захищеною та полегшить користування нею, зменшуючи час на навчання особового складу.

1.3 Огляд систем аналогів

Аналіз сучасного ринку логістичних інформаційних систем показує нестачу рішень, адаптованих до потреб військової сфери. Більшість доступних продуктів орієнтовані на цивільну або комерційну логістику й не враховують особливості військових умов – таких як інтеграція з суміжними системами, підвищені вимоги до захисту даних, а також здатність працювати в умовах обмеженого доступу та високої динаміки бойових дій.

Серед відомих військових логістичних рішень варто відзначити системи, які використовуються в арміях країн НАТО та США – зокрема «LOGFAS», «GCSS-Army» та «SAMS-E (Standard Army Maintenance System-Enhanced)».

Система «LOGFAS» є застосунком для повного планування, ведення обліку та стратегічного корегування, зображена на рисунку 1.1.

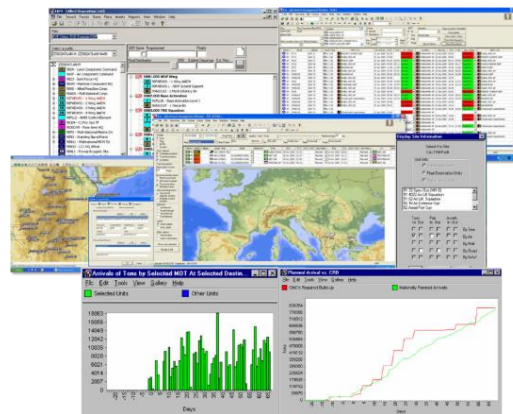


Рисунок 1.1 – Приклад інтерфейсу системи «LOGFAS»

Переваги:

— охоплює широкий спектр сервісів, зокрема для обліку логістичних операцій, аналітики та підтримки прийняття рішень, управління інженерною інфраструктурою, відстеження ресурсів і медичного забезпечення.

- вона об'єднує різні функції управління й координації логістичних даних, що сприяє підвищенню ефективності процесів постачання.

Недоліки:

- складність інтерфейсу.
- громіздкість системи та її невідповідність організаційній структурі ЗСУ.
- складність інтеграції з іншими системами, через не надання доступу до кодової бази з сторони НАТО.

Система «GCSS-Army» працює на базі централізованої БД і забезпечує управління постачанням, технічним обслуговуванням та логістичними процесами у військах США.

Переваги:

- має електронну звітність, що надає прозорість.
- має групу підсистем, які надають можливість не лише контролювати розподіл ресурсів, але й планувати маршрути постачання.

Недоліки:

- жорстко інтегрована з внутрішніми структурами американської армії, що унеможлиблює її адаптацію до структури та вимог українського війська.
- Довгий час навчання особового складу.
- Проблеми з збереження секретних даних оскільки не має можливості підключення до системи власної бази даних.

«SAMS-E (Standard Army Maintenance System-Enhanced)» – це інформаційна система, розроблена для автоматизації процесів технічного обслуговування військової техніки, управління запасами та оформлення замовлень на постачання.

Переваги:

- забезпечує комплексну підтримку логістичних процесів та формування детальних звітів.

Недоліки:

- має складний інтерфейс, вимагає тривалого навчання персоналу та регулярного оновлення програмного забезпечення.

На меті є створення програмної системи, яка буде основана на базі взаємодії з ресурсами підрозділів, з максимальною гнучкістю, а саме додавання нових типів ресурсів, бойових груп ролей командирів та їх звань. Основні аспекти розробки мобільного застосунку будуть зосереджені на простоті, швидкодії та зручності використання в умовах обмеженого зв'язку. Інтерфейс розроблено з урахуванням потреб військових підрозділів, зокрема командирів, логістів та оперативного персоналу, які працюють у польових умовах.

Ключова функціональність включає створення логістичних запитів, підтвердження отримання ресурсів, перегляд місій та управління ресурсами бойової групи. Інтерактивність реалізовано через адаптивні екрани, змінні кнопки залежно від статусу запитів і ролі користувача, а також спрощену навігацію між основними модулями системи.

Особливу увагу приділено безпеці та авторизації – кожна дія користувача перевіряється через токени доступу, дані передаються виключно по захищених каналах, а ролі (логіст, командир, адміністратор) мають чітко розмежовані права.

Застосунок підтримує інтеграцію з серверною частиною через REST API, що дозволяє синхронізувати інформацію в режимі реального часу. Також реалізовано локалізацію українською та англійською мовами, адаптацію до різних розмірів екранів та підтримку роботи на Android 6.0+.

Таким чином, застосунок забезпечує оперативне управління логістикою війська в умовах бойової обстановки, зберігаючи при цьому високу продуктивність, безпеку та зручність користування.

1.4 Монетизація

Мобільний застосунок системи «UA Logistics» розробляється як некомерційний продукт, орієнтований на підвищення ефективності логістичного забезпечення Збройних Сил України. Основна модель фінансування базується не на прямому продажу або підписці користувачів, а на підтримці з боку державних та недержавних структур.

Основними джерелами монетизації можуть виступати:

- Гранти та державні програми підтримки цифрової трансформації сектору безпеки та оборони України;
- Фінансування з боку Міністерства оборони України, що зацікавлене у впровадженні технологічних рішень для підвищення бойової готовності та логістичного контролю;
- Волонтерські фонди та благодійні організації, які інвестують у цифрову модернізацію ЗСУ та забезпечення армії сучасними інформаційними інструментами.

Таким чином, замість класичної платної моделі, застосунок «UA Logistics» орієнтований на цільове фінансування в рамках державних і волонтерських ініціатив, що дозволяє забезпечити його безкоштовну доступність для військовослужбовців і гарантує постійну підтримку, оновлення та масштабування функціоналу.

1.5 Постановка задачі

У сучасних умовах ведення бойових дій ефективне управління ресурсами залежить не лише від потужних централізованих систем, а й від зручних інструментів для роботи в реальному часі безпосередньо в польових умовах. Це зумовлює потребу у розробці мобільного застосунку, який забезпечить безпечний, інтуїтивний і функціональний доступ до логістичної системи прямо на місці.

Основна мета мобільної частини системи «UA Logistics» – створити інструмент, що дозволить військовослужбовцям (командирам, логістам, відповідальним за ресурси) оперативно переглядати, створювати та редагувати запити, керувати ресурсами підрозділу та відстежувати їх статус.

Основні функціональні можливості:

- Облік ресурсів – перегляд наявних ресурсів у підрозділі, оновлення кількості, додавання нових або видалення непотрібних. Усі зміни синхронізуються із сервером у режимі реального часу.

- Створення та редагування запитів – формування нових логістичних запитів із зазначенням місії, кількості ресурсів, їх редагування та контроль статусу.

- Безпечна робота з даними – реалізовано авторизацію, зберігання токена доступу та ідентифікатора підрозділу за допомогою DataStore. Уся взаємодія з сервером відбувається через авторизовані REST-запити.

- Оптимізований інтерфейс – інтерфейс адаптовано до польових умов: швидке введення, валідація форм, зрозуміла навігація, анімації помилок, підтримка шифрування паролів.

- Стабільність та масштабованість – асинхронне завантаження даних забезпечує плавну роботу застосунку навіть при нестабільному з'єднанні.

Таким чином, мобільний клієнт «UA Logistics» є надійним та ефективним інструментом для оперативного управління логістикою на місцях, що сприяє підвищенню гнучкості, точності та швидкості реагування бойових підрозділів.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Головна ідея проєкту полягає у створенні мобільного застосунку в рамках програмної системи «UA Logistics», який дозволяє військовим підрозділам ефективно взаємодіяти з логістичним забезпеченням у режимі реального часу. Цей застосунок слугує зручним інструментом для керування ресурсами, створення запитів, перегляду наявних запасів і моніторингу логістичних процесів безпосередньо в польових умовах.

Основна мета кваліфікаційної роботи – забезпечити повний цикл розробки мобільного компонента системи: від аналізу потреб цільових користувачів до побудови функціональних і нефункціональних вимог, реалізації інтерфейсу, перевірки стабільності, безпеки та масштабованості рішення.

Ключові завдання, які ставилися у межах проєкту:

- аналіз потреб кінцевих користувачів (командирів, логістів, відповідальних за ресурси).
- визначення функціоналу, який має бути реалізований у мобільному застосунку.
- обґрунтування вибору інструментів і технологій для реалізації.
- побудова чіткого плану розробки та етапів впровадження.
- визначення підходів до тестування та підтримки проєкту.
- забезпечення адаптивності, безпеки та інтуїтивності інтерфейсу.

Перед формуванням вимог важливо чітко окреслити потреби основної цільової аудиторії – військовослужбовців, що діють в умовах обмеженого часу, зв'язку та ресурсу. Для таких користувачів особливо критичними є стабільна робота, простота навігації, швидкий доступ до ключових дій (створення запиту, перегляд ресурсів, обробка звернень), а також безпечний захист даних.

Успішне формування вимог передбачає врахування таких категорій:

- вимоги до основних функцій застосунку (авторизація, перегляд та редагування ресурсів, створення запитів).
- вимоги до безпеки обробки даних (токени, автентифікація, локальне зберігання).
- вимоги до підтримки локалізації (українська та англійська мови).
- вимоги до масштабованості (можливість розширення функціоналу без зміни архітектури).

Таким чином, у процесі формування вимог до мобільного клієнта «UA Logistics» було враховано як функціональні потреби користувачів, так і особливості середовища, у якому він буде застосовуватись. Це дозволяє забезпечити надійний, гнучкий та ефективний засіб логістичного управління для сучасної армії.

2.1 Функціональні вимоги

Функціональні вимоги до мобільного застосунку системи «UA Logistics» окреслюють ключовий функціонал, який повинен бути реалізований у клієнтській частині, орієнтованій на військових користувачів – командирів, логістів і відповідальних за облік ресурсів. Основна мета додатку – забезпечити швидкий та зручний доступ до можливостей логістичної системи у режимі реального часу, безпосередньо в польових умовах.

До основних функціональних можливостей належать:

- Авторизація користувача – вхід до системи за допомогою електронної пошти або токена, з подальшим локальним збереженням токена доступу, `userId` та `unitId` для коректної взаємодії з сервером.

- Перегляд ресурсів підрозділу – можливість переглядати перелік наявних ресурсів із виведенням їх кількості, назв та категорій.
- Редагування та видалення ресурсів – функціонал для коригування обсягу ресурсів або їх повного видалення у разі витрати, пошкодження чи передачі.
- Додавання нових ресурсів – створення нових записів із валідацією (наприклад, заборона на введення від’ємних значень).
- Робота з логістичними запитами, а саме формування нового запиту з вибором місії, додавання декількох ресурсів і зазначення кількості, надсилання запиту до логістичного центру, редагування або скасування запиту до моменту його підтвердження.
- Перегляд статусів запитів – інформування користувача про поточний стан усіх запитів (наприклад: Pending, Approved, Declined, Delivered).
- Фільтрація за місіями – відображення лише запитів, пов’язаних із конкретною місією.
- Рольова логіка інтерфейсу – обмеження або надання доступу до окремих функцій залежно від ролі користувача або статусу запиту.
- Захист даних – усі запити до серверу виконуються з використанням авторизаційного токена, а локальні дані зберігаються у захищеному середовищі DataStore.
- Валідація форм і обробка помилок – додаток інформує користувача про помилки через Snack-повідомлення або візуальні ефекти (наприклад, анімація shake при невірному введенні).
- Зручна навігація – реалізація нижньої навігаційної панелі (Bottom Navigation) для швидкого перемикання між основними екранами.

- Анімації переходів – плавні переходи між екранами для покращення користувацького досвіду.

- Підтримка офлайн-режиму – інформування про втрату з’єднання, блокування повторної відправки критичних запитів без підтвердження.

У сукупності ці вимоги формують основу мобільного клієнта як надійного інструменту оперативного логістичного управління в умовах бойових дій.

2.2 Нефункціональні вимоги

Нефункціональні вимоги до мобільного застосунку «UA Logistics» визначають характеристики, що забезпечують стабільну, захищену та ефективну роботу застосунку в реальному середовищі, зокрема – в умовах обмеженого зв’язку, високої відповідальності та стресових ситуацій.

Основні нефункціональні характеристики включають:

- Висока продуктивність – час реакції на базові дії користувача (наприклад, перехід між екранами, збереження запиту) не повинен перевищувати 1–2 секунд.

- Стійкість до нестабільного інтернету – додаток має коректно функціонувати навіть при повільному або нестабільному мобільному з’єднанні, із відповідними повідомленнями про втрати зв’язку.

- Надійність – стабільна робота застосунку без збоїв, з автоматичним збереженням введених даних у разі аварійного закриття або втрати мережі.

- Адаптивність інтерфейсу – підтримка різних типів пристроїв і орієнтацій екрану (портретна/ландшафтна), включаючи планшети.

- Захист даних, а саме використання HTTPS для шифрування мережеских запитів, локальне збереження чутливих даних у зашифрованому

вигляді, автоматичне завершення сесії та можливість примусового скидання токена.

- Інтерфейсна доступність – дотримання принципів контрастності, читабельності, мінімалізму та інтуїтивності для зручності роботи у складних умовах.

- Гнучкість архітектури – можливість розширення функціоналу (наприклад, додавання чату, карт або сканера) без необхідності кардинальної перебудови застосунку.

- Рольовий контроль доступу – інтерфейс має динамічно адаптуватися до прав користувача відповідно до його ролі.

- Мультимовність – можливість реалізації локалізації щонайменше українською та англійською мовами.

Для забезпечення стабільної, надійної та безпечної роботи мобільного застосунку «UA Logistics», у процесі розробки передбачено комплексний підхід до тестування. Тестування є критично важливим етапом, що дозволяє виявляти помилки на ранніх стадіях і забезпечити високу якість кінцевого продукту.

У процесі тестування застосовуються два основні підходи юніт-тести, інструментальні тести та мануальне тестування.

Юніт-тести використовуються для перевірки окремих модулів, функцій та класів застосунку. Це дозволяє локалізувати помилки та гарантувати коректність логіки на рівні ViewModel, UseCase або окремих утиліт.

Для реалізації юніт-тестів застосовуються наступні інструменти:

- JUnit – основний фреймворк для створення та запуску юніт-тестів у середовищі Kotlin/Java;

- Mockito – бібліотека для створення моків (заглушок) залежностей, що дозволяє тестувати компоненти ізольовано від зовнішніх сервісів або API.

Юніт-тестування дає змогу перевірити бізнес-логіку без необхідності запуску повного додатку, що прискорює процес розробки та знижує ризик помилок у майбутньому.

Інструментальне тестування проводиться для перевірки взаємодії користувача з інтерфейсом застосунку. Основна мета – переконатися, що ключові сценарії (авторизація, створення запиту, редагування ресурсу тощо) працюють коректно на різних пристроях.

Для цього використовується фреймворк Espresso, який дозволяє автоматизовано виконувати дії користувача (натискання, введення тексту, скролінг) та перевіряти очікувані результати прямо на екрані.

Окрім автоматизованих перевірок, виконується ручне (мануальне) тестування, яке моделює дії реального користувача. Воно дозволяє виявити неочевидні помилки, пов'язані з UX, візуальними багами, неправильним відображенням даних або поведінкою інтерфейсу при нестабільному інтернет-з'єднанні.

Також перед початком реалізації мобільного застосунку було проведено аналіз вимог до функціональності, надійності та зручності використання, що дозволило визначити оптимальний стек технологій та інструментів розробки. Вибрані інструменти забезпечують сучасний підхід до створення Android-застосунків, спрощують процес розробки, полегшують тестування та гарантують масштабованість проєкту в майбутньому.

Мова програмування та платформа:

- Kotlin – основна мова розробки мобільного застосунку, яка рекомендована Google для Android і дозволяє писати менш об'ємний та більш безпечний код порівняно з Java;

- Android SDK – офіційний інструментарій для розробки Android-застосунків.

Архітектура та бібліотеки:

- MVVM (Model–View–ViewModel) – архітектурний підхід, що забезпечує розділення відповідальностей та полегшує тестування компонентів.
- Jetpack Libraries – набір сучасних бібліотек від Google, включно з:
- ViewModel – для збереження даних при зміні конфігурації.
- LiveData / StateFlow – для реактивного оновлення інтерфейсу.
- Navigation Component – для безпечної та зручної навігації між екранами.

Зберігання даних та взаємодія з сервером:

- DataStore – для збереження токенів доступу та параметрів користувача у локальному зашифрованому сховищі.
- Retrofit + OkHttp – для виконання REST-запитів до серверної частини.

Тестування:

- JUnit та Mockito – для написання юніт-тестів логіки.
- Espresso – для автоматизованого UI-тестування.

Середовище розробки:

- Android Studio – офіційне IDE для Android, яке забезпечує зручні інструменти для кодування, налагодження, запуску емуляторів та інтеграції з системами контролю версій.

Контроль версій:

- Git – система контролю версій, яка забезпечує ефективну командну роботу.
- GitHub – платформа для зберігання коду, керування задачами, обговорення змін і версіонування проєкту.

Ці вимоги забезпечують високу якість, надійність і безпечність мобільного застосунку, роблячи його придатним для інтенсивної експлуатації у військових умовах.

2.3 Організація командної роботи та методологія розробки

Для ефективного управління розробкою мобільного застосунку «UA Logistics» було використано гнучку методологію Scrum, що дозволила організувати процес у вигляді послідовних ітерацій (спринтів) із чітко визначеними завданнями та регулярним зворотним зв'язком.

Основний акцент було зроблено на колаборації в команді, прозорості виконання завдань і постійній перевірці прогресу. Щоб досягти цього, команда використовувала такі інструменти:

- Taiga.io – система управління проєктами з відкритим кодом, яка забезпечила створення беклогу завдань, планування спринтів, візуалізацію прогресу та контроль дедлайнів.

- Mural – платформа для спільної роботи, яка використовувалась для мозкових штурмів, моделювання архітектури додатку, фіксації ідей та обговорень на дошках у візуальному форматі.

Основні етапи організації командної взаємодії включали:

- Формування списку завдань: розробка початкового беклогу з описом функціональних і нефункціональних вимог до застосунку.

- Планування спринтів: визначення переліку задач на кожен ітераційний період із пріоритезацією за важливістю та складністю.

- Щоденні стендапи: короткі синхронізації команди для обговорення прогресу, виявлення блокерів і координації.

— Огляд результатів: підбиття підсумків кожного спринту, демонстрація реалізованого функціоналу та обговорення можливих покращень.

Такий підхід дозволив підтримувати прозорість розробки, швидко адаптуватися до змін у вимогах, а також покращувати якість мобільного додатку на кожному етапі його створення.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Важливою складовою процесу проєктування програмного забезпечення є створення UML-діаграм, зокрема діаграми варіантів використання (Use Case diagram). Такий тип діаграми дає змогу наочно відобразити взаємодію користувачів із системою, визначити основні ролі та окреслити ключові функціональні можливості застосунку.

Для мобільного застосунку системи «UA Logistics» було побудовано діаграму Use Case, яка демонструє основні сценарії використання, доступні дії та відповідні ролі користувачів під час роботи з додатком (див. Додаток А).

Діаграма варіантів використання (Use Case) є важливим інструментом на етапі проєктування, який дозволяє наочно відобразити функціональні можливості системи, ролі користувачів та характер їхньої взаємодії з програмним забезпеченням. У межах системи «UA Logistics» визначено кілька основних категорій користувачів, кожна з яких має доступ до специфічного функціоналу.

Основні ролі користувачів:

- Неавторизований користувач – гість або новий користувач, який має доступ лише до екрана входу та функції запиту допомоги.
- Командир бойової групи – створює місії, керує підрозділом, формує запити на ресурси.
- Логіст бойової групи – обробляє запити на ресурси та відстежує їх виконання.
- Адміністратор – має доступ до перегляду логів, запитів і взаємодії з даними користувачів.

— Головний адміністратор – відповідає за керування адміністративними обліковими записами, створення бригад і призначення командирів.

Функціональність мобільного застосунку:

- Авторизація користувачів із безпечним зберіганням токенів.
- Перегляд ресурсів підрозділу.
- Додавання, редагування та видалення ресурсів.
- Створення та надсилання логістичних запитів.
- Перегляд поточних статусів запитів.
- Робота з призначеними місіями.
- Надсилання критичних запитів типу «MAYDAY» (з передачею координат і опису ситуації).
- Інтерактивні діалоги підтвердження чи редагування.
- Зручна навігація між основними розділами: ресурси, запити, місії, підтримка.
- Валідація форм, анімація помилок, кастомний інтерфейс зі шрифтами.

Функціональність веб-інтерфейсу:

- Реєстрація нових облікових записів та керування ними.
- Призначення ролей, створення нових підрозділів і бригад.
- Модерація запитів на доступ до системи.
- Перегляд і контроль логістичних запитів у межах бригади або штабу.
- Адміністративні функції: налаштування системи, перегляд звітів, аналіз інцидентів.
- Доступ до системних журналів і повідомлень.

Діаграма варіантів використання виконує роль візуального відображення логіки функціонування системи та взаємодії різних типів користувачів. Вона

зкладає основу для побудови інших UML-діаграм (діаграм активностей, класів, послідовностей) і сприяє формуванню цілісної архітектури програмного продукту «UA Logistics».

3.2 Архітектура системи

Архітектура мобільного застосунку «UA Logistics» побудована за принципами клієнт-серверної моделі з чітким розділенням відповідальностей між клієнтською частиною (Android-додаток) і серверною частиною (REST API-сервіс на Java). Такий підхід дозволяє досягти гнучкості, масштабованості та надійності системи в умовах активної експлуатації у військових підрозділах.

Мобільний застосунок реалізовано з використанням архітектури MVVM (Model–View–ViewModel), що сприяє покращенню структури коду, спрощенню тестування та забезпеченню реактивної взаємодії з інтерфейсом користувача. Дані у ViewModel зберігаються за допомогою StateFlow, що забезпечує актуалізацію інтерфейсу при будь-яких змінах стану.

Зв'язок між клієнтською частиною та сервером здійснюється через HTTPS-запити з використанням бібліотек Retrofit і OkHttp, що дозволяє безпечно та ефективно обмінюватися даними. Для авторизації та доступу до API використовується токенизація (JWT), а збереження облікових даних (токен, userId, unitId) реалізовано через DataStore, з дотриманням принципів безпеки [2].

Серверна частина обробляє запити мобільного клієнта, забезпечує валідацію, зберігання та обробку логістичних даних, а також взаємодію між користувачами різних ролей. Завдяки централізованій обробці інформації досягається цілісність даних і спрощене адміністрування системи.

Такий підхід дозволяє:

- Швидко масштабувати серверну частину незалежно від мобільного клієнта.
- Вносити зміни до інтерфейсу без впливу на бізнес-логіку.
- Підтримувати автономну роботу клієнта в разі втрати мережі, із подальшою синхронізацією.
- Підвищити безпеку даних завдяки централізованому доступу і зберіганню інформації.

3.3 Проєктування структури зберігання даних

У програмній системі для контролю логістичного забезпечення війська використовується комбінований підхід до зберігання даних, що поєднує реляційну базу даних MySQL та документно-орієнтовану базу MongoDB. Такий підхід забезпечує ефективну обробку як чітко структурованої, так і динамічно змінної інформації.

MySQL відповідає за збереження ключових структурованих даних: облікових записів користувачів, ролей, підрозділів, місій та логістичних запитів. Підтримка транзакцій і складних зв'язків між таблицями гарантує цілісність та надійність цих даних.

MongoDB, зі свого боку, використовується для зберігання даних, які мають високу частоту змін і гнучку структуру. До таких даних належать, наприклад, списки ресурсів підрозділів, інформація про їхнє споживання чи наявність. Завдяки своїй масштабованості MongoDB дозволяє швидко й ефективно працювати з цими записами.

Мобільний додаток UA Logistics взаємодіє з обома базами даних через REST API, реалізований на серверній стороні. Зокрема, такі функції мобільного клієнта, як:

- авторизація користувача.
- перегляд ресурсів підрозділу.
- створення та редагування запитів на постачання.
- відображення місій та їхніх статусів – опосередковано отримують дані з відповідної бази.

Це дозволяє мобільному застосунку працювати з актуальною інформацією в режимі реального часу: оновлювати ресурси, створювати запити на постачання, а також оперативно реагувати на зміну ситуації, що є критично важливим у бойових умовах.

Для візуалізації структури даних створено ER-діаграму, яка демонструє сутності, їхні атрибути та взаємозв'язки (див. Додаток Б). Вона слугує важливим орієнтиром під час інтеграції клієнтської частини з API, дозволяючи точно розуміти, які саме дані потрібні інтерфейсу, як вони мають виглядати та коли синхронізуються із сервером.

Основні сутності та їх призначення:

- Users – зберігає базову інформацію про всіх користувачів системи: ім'я, електронну пошту, роль (солдат, логіст, адміністратор) тощо.
- Soldiers – деталізує облікові записи користувачів із роллю "солдат", доповнюючи їх службовими даними: підрозділ, посада тощо. Пов'язана з таблицею Users через user_id.
- Logisticians – зберігає інформацію про користувачів з логістичними обов'язками.
- Admins – містить додаткові поля для адміністраторів, зокрема статус акаунта, дату підтвердження тощо.
- Units – описує бойові підрозділи, включаючи назву, тип і структуру підпорядкування через поле parent_unit_id.

- Missions – зберігає інформацію про військові або логістичні місії: опис, складність, період дії та підрозділ-ініціатор.

- Logistics_requests – відображає запити на забезпечення підрозділів ресурсами в межах місій; містить статус, дату створення та зв'язки з місіями й підрозділами.

- Request_resource – пов'язує логістичний запит із конкретними ресурсами, вказуючи їх кількість. Реалізує зв'язок "багато до багатьох".

- Resource_stocks – фіксує фактичну наявність ресурсів у підрозділах.

- Critical_support – таблиця, що зберігає звернення щодо критичних ситуацій (може подаватися навіть неавторизованим користувачем): позиція, опис, потреби.

- Feedbacks – містить відгуки користувачів про функціонування системи, із зазначенням тематики, статусу та змісту.

У документно-орієнтованій базі даних MongoDB:

- Resource – сутність, що зберігає назву ресурсу, тип, категорію, одиницю виміру та вартість. У перспективі передбачається додавання вкладених полів (наприклад, технічна інструкція або мета-дані), що забезпечує гнучке представлення даних.

Основні взаємозв'язки:

- Один користувач може бути або солдатом, або логістом, або адміністратором.

- Один підрозділ може мати багато солдатів, місій, запитів, ресурсів і звернень.

- Один логістичний запит може містити кілька ресурсів (багато до багатьох через Request_resource).

- Один ресурс може входити до багатьох запитів і бути присутнім у запасах кількох підрозділів.

- Структура підрозділів є ієрархічною: підрозділи можуть бути вкладеними один в один через `parent_unit_id`.

Ця модель забезпечує чітке розмежування обов'язків між таблицями та дозволяє реалізовувати складні запити до даних, не втрачаючи гнучкості, яку надає використання MongoDB для змінних за структурою ресурсів.

3.4 Обмеження та вимоги

Під час розробки мобільного застосунку «UA Logistics» було враховано низку технологічних обмежень і вимог, що визначають можливості масштабування, підтримки, безпеки та зручності використання системи в реальних бойових умовах [7].

Обмеження технологій:

- Android-платформа. Оскільки застосунок орієнтовано на Android, усі користувачі з iOS наразі залишаються поза охопленням системи. У майбутньому передбачено створення мультиплатформного рішення або окремої iOS-версії.

- Kotlin + Jetpack-бібліотеки (MVVM, ViewModel, LiveData, Navigation). Вибір цих інструментів обумовлений їх сучасністю та гнучкістю, проте вони вимагають від розробників достатнього рівня володіння архітектурними підходами та асинхронними API.

- DataStore для локального зберігання має обмежену місткість для складних структур даних, тому його застосування переважно зводиться до зберігання токенів, `userId` та `unitId`.

— REST API із захищеним обміном даними через JWT обмежує функціональність офлайн-режиму – користувачам доступні лише базові функції без оновлення даних.

— Espresso та JUnit/Mockito – основні інструменти для тестування. Вони покривають потреби в UI-тестах і юніт-тестах, однак не замінюють повноцінне інтеграційне тестування з бекапами чи аварійними сценаріями.

Функціональні та інтерфейсні обмеження:

— Інтерфейс оптимізовано під телефони. Підтримка планшетів не тестувалась і не гарантується на поточному етапі.

— Мультимовність реалізована лише частково. Інтерфейс наразі українською та англійською, у майбутньому передбачається підтримка інших мов (через використання strings.xml з перекладами).

— Авторизація виконується через email + пароль. Інтеграція з біометрією або сторонніми постачальниками (OAuth2, SSO) не реалізована.

— Критичні запити («Mayday») потребують активного інтернет-з'єднання. У разі його втрати користувач інформується, але запит не кешується для майбутнього надсилання.

Адміністративні вимоги:

— Підтримка та оновлення має виконуватись через пряме оновлення APK-файлу. CI/CD-процес наразі відсутній, що може ускладнити автоматичне розгортання нових версій.

— Інструменти управління проєктом: для командної координації використовуються Taiga.io (дошка задач, спринти, беклог) і Mural (мозкові штурми, UX-макети). Їх використання дозволяє підтримувати ефективний Scrum-процес розробки, однак потребує стабільного інтернету для всієї команди.

— Сумісність: застосунок протестовано на Android 9–14. Для старіших версій підтримка не гарантується.

Час відгуку інтерфейсу в мобільному додатку має бути мінімальним – перехід між екранами (ресурси, місії, запити, критичні звернення) повинен відбуватись не більше ніж за 500–800 мс. Це критично важливо в бойових умовах, коли користувачеві потрібно швидко орієнтуватися в інформації. У випадку помилок або втрати з'єднання, система повинна надавати зрозумілі повідомлення про проблему та пропонувати можливі дії (повторити запит, перевірити мережу тощо) без використання термінів, незрозумілих звичайному користувачу.

Застосунок має бути спроектований із урахуванням можливості масштабування та підтримки. Архітектура повинна дозволяти швидко впроваджувати нові функції, не порушуючи стабільності вже реалізованих модулів. Для цього важливо дотримуватись принципів модульності, розділення відповідальностей (наприклад, ViewModel не залежить від UI), а також мати стандартизовану систему перекладів для подальшого впровадження мультимовності (зокрема, англійської, української та потенційно інших мов НАТО).

Також безпека є одним із ключових пріоритетів мобільного застосунку «UA Logistics», враховуючи його використання у військовому середовищі. Усі запити до серверу виконуються через захищене HTTPS-з'єднання, що захищає передані дані від перехоплення [7]. Для автентифікації користувачів застосовується механізм JWT-токенів, які зберігаються локально у DataStore із захищеним доступом. Додатково, всі чутливі дані, включно з токенами, userId та іншою критичною інформацією, шифруються з використанням симетричного алгоритму шифрування AES (Advanced Encryption Standard) [6]. Також передбачено шифрування полів у запитах, що забезпечує додатковий захист у разі перехоплення трафіку. Варто зазначити, що використання шифрування може

незначно знизити швидкодію системи, проте цей компроміс є виправданим задля забезпечення високого рівня конфіденційності та безпеки даних.

На сервері реалізовано багаторівневий захист даних. Для зберігання паролів застосовується надійний алгоритм хешування BCrypt, який виключає можливість зворотного відновлення паролів навіть у разі компрометації бази даних. Для захисту від атак перебором реалізовано механізм обмеження кількості спроб входу на основі Bucket4j.

Крім того, чутливі поля в базі даних – зокрема службова інформація, метадані та персональні дані – шифруються з використанням AES-256 через Java Cryptography Extension (JCE). Для централізованого й безпечного зберігання ключів, паролів і токенів використовується система HashiCorp Vault, яка забезпечує гнучкі політики доступу, шифрування даних у стані спокою та можливість автоматичної ротації ключів. Це значно знижує ризики витоку критичної інформації в разі компрометації інфраструктури й підвищує загальну стійкість системи до атак. Такий комплексний підхід до безпеки гарантує захист конфіденційної інформації як на стороні клієнта, так і на стороні сервера, відповідаючи високим стандартам захищених інформаційних систем.

У разі втрати пристрою або виходу користувача із системи токен авторизації автоматично анулюється, що унеможлиблює доступ до функціоналу без повторного входу. Також реалізовано автоматичне перенаправлення на екран входу при недійсному токені, що виключає несанкціонований доступ до захищених екранів застосунку.

3.5 Забезпечення якості

Для забезпечення високої якості мобільного застосунку «UA Logistics» команда дотримувалася принципів прозорості командної роботи, технічного

контролю та регулярної комунікації. Усі рішення щодо змін у проєкті приймалися колегіально під час обговорень, де кожен учасник мав змогу висловити свою позицію, запропонувати альтернативні підходи або заперечити конкретне рішення. Такий підхід дозволив досягти консенсусу та приймати збалансовані рішення, що враховують технічні, часові та користувацькі обмеження.

Процес забезпечення якості включав щотижневі мітинги в Google Meet, які проводилися у двох форматах:

- Командні мітинги розробників, на яких обговорювалися технічні оновлення, блокери, рефакторинг коду, вибір архітектурних рішень та тестування.
- Окремі зустрічі з керівником, де презентувалися досягнення за тиждень, уточнювалися вимоги та визначався фокус на наступний етап розробки.

Для підвищення креативності та швидшого вирішення складних задач ми проводили брейншторми, що дозволяло генерувати нові функціональні ідеї, покращувати UI/UX та знаходити компромісні технічні рішення.

Крім регулярної комунікації, в проєкті практикувалися огляди коду (code reviews), що давало змогу перевірити якість реалізації, відповідність стандартам та уникати повторюваних помилок. Особливу увагу приділяли технічним деталям: правильному зберіганню даних, перевірці валідації форм, обробці помилок та відповідності очікуваній логіці користувача.

Для оцінки стабільності та функціональності застосунку застосовувалися:

- Unit-тестування з використанням бібліотеки JUnit та Mockito, що дозволяло протестувати окремі компоненти бізнес-логіки.
- UI-тестування з використанням Espresso, що забезпечувало перевірку цілісного користувацького досвіду, перевірку переходів між екранами та взаємодію з інтерфейсом.

— Мануальне тестування, яке імітувало поведінку кінцевого користувача в реальних сценаріях, у тому числі в умовах нестабільного інтернет-з'єднання.

Якість коду також підтримувалась через впровадження розмежування логіки відповідно до архітектури MVVM, що підвищувало тестованість, спрощувало масштабування та підтримку застосунку.

Завдяки поєднанню автоматизованого тестування, командної синергії та регулярного контролю було досягнуто високої надійності та відповідності мобільного застосунку «UA Logistics» функціональним і нефункціональним вимогам.

3.6 Керування проектом

Керування проектом є невід'ємною складовою ефективної розробки мобільного застосунку «UA Logistics». Воно охоплює планування, організацію, контроль виконання завдань, взаємодію в команді, облік прогресу та своєчасну адаптацію до змін. Для цього було впроваджено гнучку методологію Scrum, яка дозволила структурувати роботу за ітеративним підходом із регулярною оцінкою результатів.

На етапі ініціації проекту ми провели аналіз проблематики та сформулювали ключову мету: створення мобільного інструменту для логістичного забезпечення бойових підрозділів у польових умовах. На основі обговорень з куратором і в команді було визначено склад користувачів, їх ролі та функціональні потреби. Ці дані стали основою для формування Backlog-списку – переліку функцій і задач, необхідних для реалізації системи.

Для управління завданнями та комунікацією ми використовували Taiga.io – сучасну платформу проєктного менеджменту з підтримкою Kanban-дошки, Story

Mapping і задач по спринтах. У Taiga були створені категорії завдань: To Do, In Progress, Ready for Review, Done, що дозволяло зручно відслідковувати поточний стан розробки, призначати відповідальних і контролювати дедлайни.

Візуалізація ідей, планування архітектурних рішень та брейншторми проводилися на платформі Mural, що дозволяло команді спільно працювати над діаграмами, функціональними блоками та дизайном інтерфейсу. За допомогою Mural ми також структурували логіку взаємодії між ролями користувачів, екранами застосунку та запитам до API.

Окрім цього, ми активно використовували Google Таблиці для складання графіків, логів функцій, трекінгу прогресу і тестування, а також Google Документи для ведення проєктної документації, фіксації обговорень і збору технічних рішень. Це забезпечило прозорість процесу розробки та доступ до інформації для кожного учасника команди.

Таким чином, завдяки використанню гнучких інструментів керування, регулярному плануванню спринтів і візуалізації робочого процесу, команда досягла чіткої координації, високого рівня прозорості та ефективності на всіх етапах розробки мобільного застосунку «UA Logistics».

3.7 Тестування

Для забезпечення високої якості, стабільності та надійності мобільного застосунку «UA Logistics», у процесі розробки було передбачено використання двох основних підходів до тестування: мануального тестування та автоматизованого юніт-тестування.

Мануальне тестування передбачає перевірку функціональності мобільного додатку вручну, без використання спеціалізованих тестових фреймворків. Такий тип тестування дозволяє імітувати поведінку реального користувача, оцінити

UX/UI, візуальну коректність екранів, логіку переходів між розділами, а також перевірити реакцію системи на нетипові сценарії. Перевагою мануального тестування є його гнучкість, здатність швидко адаптуватись до змін у застосунку, а також можливість виявлення помилок, які важко виявити автоматизовано – наприклад, неузгодженості в дизайні, некоректна робота навігації або локалізації.

Юніт-тестування використовується для перевірки окремих одиниць програмного коду – функцій, методів, класів або логіки ViewModel. У проєкті активно застосовуються JUnit для тестування логіки бізнес-рівня та ViewModel, а також Mockito – для створення ізольованих тестів із використанням мок-об'єктів. Завдяки цьому ми можемо тестувати компоненти незалежно від зовнішніх залежностей, таких як репозиторії, API чи бази даних. Це дозволяє виявляти критичні помилки ще до інтеграції окремих частин застосунку, що значно зменшує вартість їх усунення.

Для перевірки інтерфейсу та інтеграцій між елементами UI застосовується Espresso – фреймворк для автоматизованого UI-тестування Android-додатків. За допомогою Espresso тестуються сценарії авторизації, навігації, відправки запитів, відображення діалогів, а також перевіряється коректна обробка помилок. Такі тести дозволяють гарантувати, що додаток функціонує очікувано на реальному пристрої або емуляторі.

Тестування виконувалося в кілька етапів:

- Початкове юніт- і мануальне тестування – для перевірки базової функціональності кожного модуля.
- Фіксація знайдених помилок – усі помилки документувались у спільному Google Spreadsheet та трекались у системі керування проєктом (Taiga).
- Усунення помилок – розробники виправляли виявлені дефекти, після чого тестувальники повторно перевіряли відповідні ділянки.

— Регресійне тестування — проводилось після кожного великого оновлення для перевірки, чи не виникли нові помилки в уже протестованих частинах застосунку.

— Ітеративне доповнення юніт-тестів — після додавання нової функціональності автоматизовані тести оновлювались відповідно до змін.

Усі тести проводились на максимально наближених до реальності тестових даних, щоб гарантувати коректність роботи додатку при взаємодії з бойовими підрозділами та реальними логістичними запитами.

Таким чином, поєднання мануального тестування, автоматизованих модульних тестів на основі JUnit і Mockito, а також UI-тестування через Espresso забезпечило надійну перевірку всіх функцій мобільного застосунку. Це дозволило мінімізувати кількість помилок, покращити користувацький досвід та забезпечити відповідність вимогам безпеки, продуктивності та масштабованості проєкту.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Для реалізації клієнтської частини мобільного додатку «UA Logistics», яка працює на операційній системі Android, було обрано мову програмування Kotlin разом із фреймворками та бібліотеками з екосистеми Jetpack Android. Такий вибір обґрунтовано сучасністю технологій, широкою підтримкою спільноти та офіційною рекомендацією від Google для розробки Android-додатків.

Kotlin – це сучасна, статично типізована мова, яка повністю сумісна з Java, але має простіший і компактніший синтаксис. Основні переваги Kotlin у розробці мобільного додатку:

- Сучасний синтаксис. Kotlin має лаконічніший та більш виразний синтаксис у порівнянні з Java, що зменшує кількість шаблонного коду та підвищує читабельність.

- Безпечна робота з null-посиланнями. Kotlin надає вбудовану підтримку null-безпеки, що дозволяє уникати помилок типу NullPointerException на етапі компіляції.

- Розширення функціоналу. Мова підтримує extension-функції, data-класи, корутини для асинхронної роботи та інші можливості, які або відсутні в Java, або реалізовані складніше.

- Підтримка Google. Kotlin є офіційною мовою розробки Android-додатків, що забезпечує її глибоку інтеграцію з Android SDK та стабільну підтримку з боку Google.

У порівнянні з Java, Kotlin дозволив нам значно пришвидшити розробку, зробити код менш громіздким і безпечнішим у підтримці, що було критично важливо під час швидких спринтів і частих оновлень логіки.

Але не зважаючи на всі переваги Kotlin в нього також присутні недоліки, як і в будь-якій мові програмування. Зазначимо головні з них:

- Більший розмір APK, а саме у порівнянні з чистим Java-кодом, Kotlin додає до застосунку додаткові залежності (Kotlin стандартна бібліотека, корутини тощо), що може збільшити розмір APK. Це особливо критично для проєктів, які мають суворі вимоги до розміру застосунку.

- Процес компіляції Kotlin трохи повільніший, особливо при використанні складних синтаксичних конструкцій або великої кількості extension-функцій. Це може впливати на час збирання проєкту, особливо у великих застосунках.

- Менше прикладів і рішень у спільноті, ніж у Java. Хоча Kotlin швидко набирає популярність, кількість прикладів, бібліотек і відповідей на форумах типу StackOverflow все ще менша, ніж у Java, яка існує десятиліттями.

- Наявність прихованої магії. Через свою лаконічність Kotlin іноді приховує складні процеси, як-от генерацію анонімних класів або неочевидні виклики функцій. Це ускладнює налагодження (debugging), особливо у великих командах із різним рівнем досвіду.

Попри наявні недоліки, переваги Kotlin у контексті сучасної Android-розробки переважають. Знання про можливі труднощі дозволяє уникнути типових помилок, оптимізувати архітектуру застосунку та краще планувати розвиток проєкту. У нашому випадку Kotlin виправдав себе як ефективний, зручний та сучасний інструмент для розробки мобільного застосунку «UA Logistics».

Основу архітектури додатку становить MVVM (Model-View-ViewModel) – архітектурний патерн, який дозволяє розділити логіку бізнес-процесів, управління станом і відображення даних. Така архітектура обрана завдяки своїй масштабованості, підтримці повторного використання коду та легкій підтримці

ViewModel, особливо в поєднанні з StateFlow, який дозволяє спостерігати за станом даних у реактивному стилі.

Model відповідає за бізнес-логіку, зберігання і обробку даних, а також за взаємодію з зовнішніми API (через Retrofit) та локальними сховищами (через Room, DataStore або SharedPreferences).

View – це UI-компоненти, які відповідають виключно за відображення інформації та взаємодію з користувачем. Вони не містять логіки, що спрощує їх повторне використання.

ViewModel виконує роль посередника між View і Model. Вона керує станом UI, отримує дані з моделі та надає їх View у зручному вигляді. Для керування асинхронними потоками та станом застосовується StateFlow, що дозволяє реалізувати реактивний підхід та оновлювати UI у реальному часі при зміні стану даних.

Використання StateFlow (та іноді LiveData) у ViewModel дозволяє уникати зайвих перерисовок, ефективно управляти життєвим циклом компонентів і забезпечити односпрямований потік даних (unidirectional data flow), що суттєво підвищує передбачуваність логіки.

Ключовими компонентами на рівні UI є Activity та Fragment. Вони виконують роль контейнерів для View і взаємодіють із ViewModel. У застосунку кожна Activity відповідає за окрему логічну частину функціональності (наприклад, логін, перегляд ресурсів, створення запитів). Activity ініціалізує ViewModel, спостерігає за її станом і підписується на зміни через StateFlow, щоб оновлювати UI. Вся бізнес-логіка винесена з Activity, що забезпечує її мінімальну відповідальність, чистоту та простоту. Такий підхід полегшує тестування та повторне використання коду, а також забезпечує стабільну поведінку під час змін конфігурації пристрою (наприклад, обертання екрана).

Для масштабованості проєкту реалізовано розділення на окремі шари:

- Data layer (API, DTO, моделі, DataSource).
- Domain layer (use cases, бізнес-логіка).
- Presentation layer (ViewModels, UI-компоненти, Activity/Fragment).

Такий підхід дозволяє незалежно тестувати логіку, легко замінювати частини системи, а також швидко адаптуватися до змін у вимогах.

Крім того, для зменшення зв'язності між компонентами та зручного впровадження залежностей використовується Hilt – офіційна бібліотека від Google для Dependency Injection у Android. Вона спрощує створення та передачу залежностей між шарами та підвищує контрольованість структури проєкту.

Завдяки такій архітектурі мобільний застосунок «UA Logistics» є:

- масштабованим – додавання нових функцій не впливає на вже реалізовані модулі.
- тестованим – окремі компоненти можуть бути протестовані незалежно.
- підтримуваним – логіка чітко структурована та зрозуміла для будь-якого нового розробника.
- реактивним – зміни даних автоматично оновлюють інтерфейс
- стабільним – завдяки правильному управлінню життєвим циклом Activity та ViewModel.

Для зберігання токенів авторизації та користувацьких даних було використано Jetpack DataStore, як заміну застарілому SharedPreferences. DataStore гарантує асинхронну роботу, потокову обробку даних та збереження навіть у разі раптового завершення роботи застосунку.

Для відображення інтерфейсу користувача обрано класичний підхід з використанням XML-макетів, що дає змогу створювати чітко структуровані екрани, а також Jetpack ViewBinding – сучасний інструмент, який автоматично

генерує безпечні доступи до елементів інтерфейсу, спрощуючи роботу з UI та зменшуючи кількість потенційних помилок. Водночас ми свідомо відмовилися від Flutter та мови Dart через такі причини:

- Природна інтеграція з Android. XML та Jetpack Compose є нативними інструментами для Android-розробки, що дозволяє повністю використовувати можливості платформи без необхідності додаткових обгорток або bridge-технологій.

- Краща продуктивність. Оскільки наш застосунок взаємодіє з мережею, локальними базами даних і має складну логіку, важливо було отримати максимальну нативну продуктивність, якої важче досягти у кросплатформенних рішеннях.

- Гнучкість у виборі підходу. XML дозволяє чітко контролювати структуру layout'ів, зокрема для складних адаптивних форм, а Jetpack Compose дає змогу швидко створювати реактивні елементи UI, використовуючи декларативний підхід, подібний до React.

Переваги XML та Jetpack Compose

XML:

- Забезпечує стабільну розмітку з детальним контролем розташування елементів.

- Підтримується всіма Android Studio інструментами та має розвинене прев'ю.

- Ідеально підходить для екранів із фіксованою структурою, як-от форми створення або перегляду запиту.

Jetpack Compose:

- Підтримує декларативний підхід, що дозволяє динамічно змінювати інтерфейс відповідно до стану програми.

- Дає змогу уникати XML-роздування в багатьох простих компонентах.
- Зменшує кількість коду за рахунок вбудованої логіки UI, а також чудово інтегрується з StateFlow та ViewModel.

У поєднанні ці два підходи дозволили створити гнучкий, стабільний та зручний інтерфейс, який відповідає високим вимогам до військової логістики, зокрема в аспектах простоти, швидкості взаємодії та стабільної роботи навіть за обмежених ресурсів.

Але попри численні переваги XML та Jetpack Compose у створенні інтерфейсів для Android-застосунків, ці інструменти також мають свої недоліки. Як і будь-які технології, вони не є універсальним рішенням для всіх випадків. Нижче наведено основні з них:

- XML-підхід є імперативним, що не дозволяє створювати повністю декларативний інтерфейс. Зміни в UI доводиться реалізовувати вручну через ViewBinding або знайдення елементів за findViewById()
- Для реалізації адаптивних або анімованих інтерфейсів потрібно писати багато додаткового коду, або використовувати сторонні бібліотеки. Це може ускладнювати підтримку проєкту.
- Великі ієрархії XML-макетів можуть знижувати продуктивність через перевантаження ViewGroup, що призводить до збільшення часу рендерингу екранів.
- Jetpack Compose хоч і є офіційним фреймворком, він ще досить молодий. У деяких версіях бібліотек можуть з'являтися непередбачувані баги, що ускладнює роботу на продакшн-проєктах.
- Хоч документація Compose активно зростає, для багатьох складних або кастомних випадків ще не вистачає прикладів та пояснень у спільноті, особливо у порівнянні з традиційною XML-розробкою.

Обидва підходи мають свої недоліки, тому в нашому проєкті «UA Logistics» ми обрали комбіновану стратегію. Такий гібрид дозволив нам поєднати перевірену стабільність XML із новітніми перевагами Jetpack Compose, поступово рухаючись до повної декларативної розробки.

Збірка та тестування застосунку автоматизовані за допомогою Gradle, що дозволяє легко інтегрувати мобільний клієнт у загальний пайплайн CI/CD [8]. Для забезпечення передбачуваності середовища розробки та тестування використовуються емулятори Android з фіксованими конфігураціями, які можуть бути розгорнуті як локально, так і в хмарі.

Мобільний застосунок орієнтований на роботу з серверною частиною, яка контейнеризована у Docker і розгортається через Docker Compose або Kubernetes. Це дозволяє розробникам протестувати застосунок на локальному стенді або в хмарному середовищі з реальними залежностями (бази даних, бекенд-сервіси) у кілька команд. Такий підхід дозволяє уникнути проблем з несумісністю середовищ і забезпечує передбачувану поведінку під час інтеграційного тестування.

Таким чином, клієнтська частина «UA Logistics» – це не лише інтуїтивний і надійний мобільний застосунок, а й повноцінний елемент сучасної мікросервісної архітектури, інтегрований у спільну інфраструктуру з можливістю швидкого масштабування, автоматизованого тестування та безперебійного оновлення. Це забезпечує стабільну роботу застосунку навіть в умовах бойових дій або обмеженого підключення до мережі.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування є ключовим етапом у розробці мобільного застосунку «UA Logistics», адже від якості реалізації залежить надійність, стабільність та зручність взаємодії користувачів із системою в умовах обмеженого зв'язку або підвищеної відповідальності. У процесі розробки застосунку було реалізовано комплексне тестування з використанням JUnit, Mockito та Espresso, що дозволило забезпечити як перевірку логіки окремих компонентів, так і взаємодії між ними в рамках повноцінних сценаріїв.

Модульне тестування реалізовувалося за допомогою бібліотеки JUnit, яка дозволяла ізолювати перевірку поведінки ViewModel, репозиторіїв і use-case логіки. Для емуляції залежностей, зокрема API, локальних сховищ або сервісів авторизації, використовувався Mockito, що дозволяло створювати гнучкі мок-об'єкти без потреби в реальному бекенді. Це дозволило протестувати, як ViewModel реагує на різні типи відповідей – як успішні, так і з помилками – та переконатися в коректності формування стану інтерфейсу на основі StateFlow.

Інтеграційне тестування було зосереджене на перевірці зв'язку клієнта з сервером у контрольованому середовищі. Для цього застосунок тестувався у зв'язці з локальним backend, запущеним через Docker, що дозволило перевірити повні користувацькі сценарії: автентифікація, отримання та оновлення ресурсів, створення логістичних запитів. Такий підхід гарантував, що застосунок коректно обробляє як типові відповіді API, так і нестандартні ситуації – наприклад, недійсний токен, відсутність прав доступу або помилки сервера.

Окрему увагу було приділено автоматизованому UI-тестуванню, яке виконувалося за допомогою фреймворку Espresso. Завдяки йому перевірялася поведінка застосунку при взаємодії з ключовими елементами інтерфейсу:

натискання кнопок, введення даних у поля форм, переходи між активностями, а також валідація введених даних. Особливий акцент робився на перевірку сторінки входу, списку ресурсів, інтерфейсу створення та редагування запитів. Це дало змогу забезпечити цілісність UX-досвіду та відстежити можливі UI-помилки до їхнього потрапляння в продакшн.

Регулярне проведення регресійного тестування після кожного релізного спринту забезпечувало стабільність основного функціоналу. Автоматичні тести запускалися в середовищі CI з інтеграцією в GitHub Actions, що гарантувало оперативне виявлення збоїв у логіці чи інтерфейсі.

У результаті реалізованого підходу до тестування було підтверджено, що мобільний застосунок демонструє високу стабільність, обробляє помилки відповідно до очікувань і витримує сценарії роботи в умовах бойового застосування. Застосування JUnit, Mockito та Espresso дозволило забезпечити впевненість у якості продукту, а також створити надійний фундамент для подальшої підтримки та масштабування системи.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В Розгортання серверної частини програмної системи UALogistics відбувалося поетапно, із суворим дотриманням принципів безперервної інтеграції (CI) та безперервного постачання (CD). Основним завданням цього процесу було забезпечити стабільну роботу програмного забезпечення в умовах командної співпраці, гарантувати якісне тестування функціональності, підготувати систему до подальшого використання у реальних умовах та продемонструвати її можливості зацікавленим сторонам.

Важливою подією в історії розвитку проєкту стала його презентація на виставці технічної творчості молоді, яка відбулася в межах 29-го Міжнародного молодіжного форуму «Радіoeлектроніка та молодь у XXI столітті». Команда, яка включала розробника серверної частини, а також відповідальних за мобільну й веб-версії, успішно представила UALogistics і отримала високу оцінку журі. Система посіла перше місце в номінації «Програмне забезпечення. Програми», що стало визнанням високого технічного рівня рішень, ефективності реалізації функціональних вимог і актуальності проєкту в умовах сучасного військового середовища.

Проєкт демонструє значний потенціал до масштабування. Серверна частина повністю готова до подальшого застосування як у навчальному процесі, так і в умовах бойових полігонів або в інтеграції з іншими інформаційними системами військового призначення. Завдяки високому ступеню функціональної завершеності та реалізованим заходам безпеки, UALogistics розглядається як перспективне рішення для впровадження у підрозділах Збройних Сил України. Система може бути використана не лише для тилового забезпечення, а й

безпосередньо на передовій, де вона здатна автоматизувати процеси обліку, планування й розподілу ресурсів.

У подальшому планується розгортання системи в хмарному середовищі з використанням технологій на кшталт Kubernetes, що дозволить забезпечити динамічне масштабування, автоматичне балансування навантаження та високу доступність сервісів. Це відкриває широкі перспективи для застосування UALogistics у різноманітних сценаріях військової логістики – від тренувальних операцій до реального бойового застосування.

ВИСНОВКИ

У результаті виконаної роботи було розроблено мобільний застосунок для інформаційної системи «UA Logistics», що слугує зручним інструментом для оперативної взаємодії військових підрозділів у процесі логістичного забезпечення. Головною метою стала розробка функціонального, інтуїтивного та захищеного мобільного інтерфейсу, який дозволяє користувачам – насамперед командирам і логістам – керувати запитами на ресурси, переглядати наявні запаси, створювати місії, залишати звернення та отримувати важливу інформацію безпосередньо в польових умовах.

Застосунок реалізовано на платформі Android із використанням мови Kotlin, архітектурного шаблону MVVM та бібліотек Jetpack (ViewModel, LiveData, Navigation). Для локального зберігання даних застосовано DataStore, а для взаємодії із сервером – REST API. Реалізована система авторизації з підтримкою JWT-токенів забезпечує надійний та безперервний доступ користувача без повторної аутентифікації.

Інтерфейс адаптовано до умов польового використання – він має лаконічний, контрастний дизайн і забезпечує швидкий доступ до основних функцій. Реалізовано валідацію форм, анімації помилок, візуальні підказки та плавні переходи, що покращують взаємодію користувача з системою.

Особливу увагу приділено управлінню ресурсами: користувач може переглядати запаси підрозділу, створювати запити на поповнення, контролювати статус постачання та оперативно змінювати кількість необхідних матеріалів. Також передбачено можливість подати запит на критичну підтримку у разі екстреної ситуації.

Таким чином, мобільний застосунок «UA Logistics» є ключовим елементом цифрової екосистеми військового забезпечення, що підвищує швидкість

реагування, точність обліку та знижує ризик помилок у складних оперативних умовах. Його використання сприяє загальному підвищенню ефективності логістичного управління у Збройних Силах України.

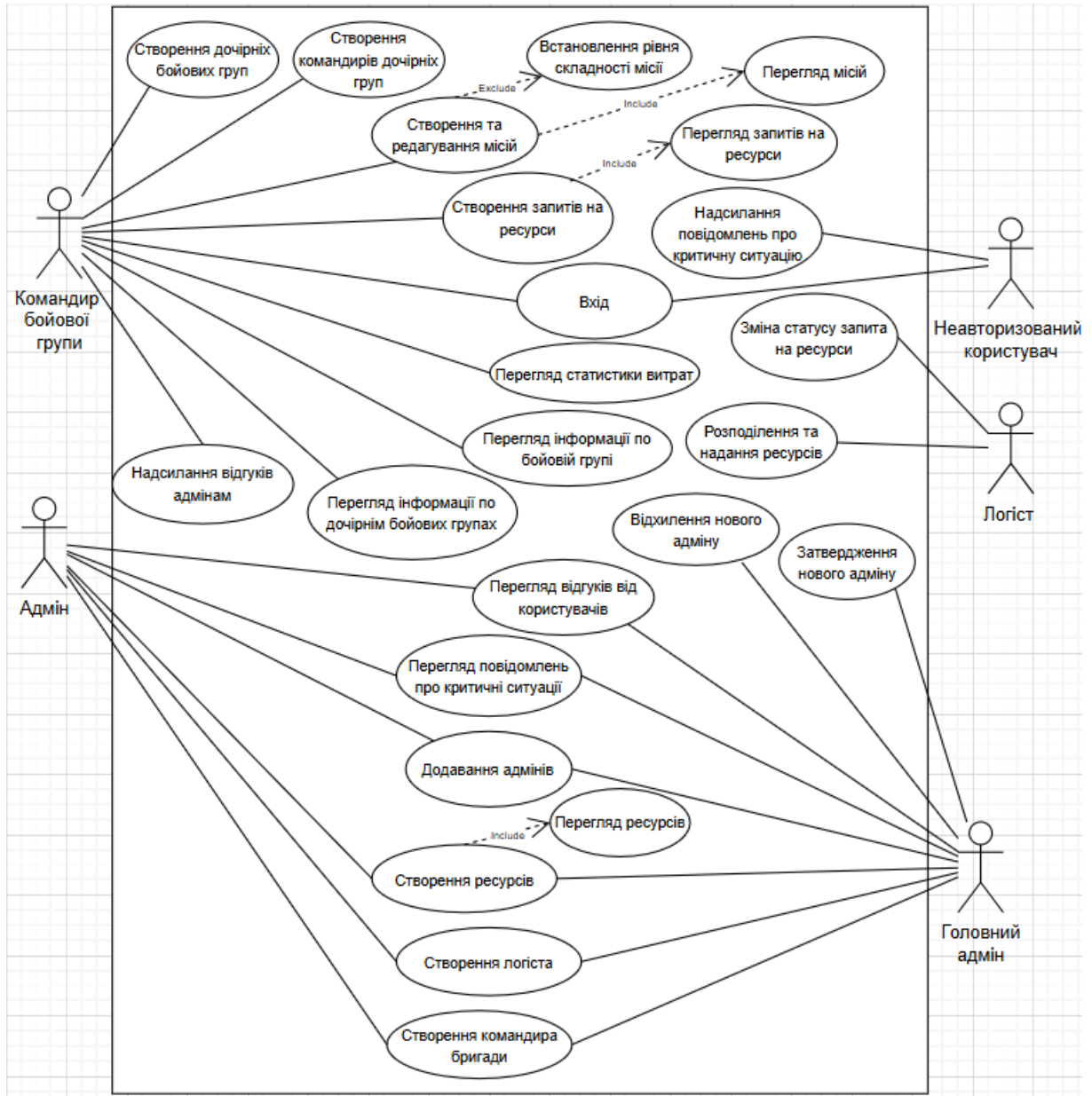
Для організації роботи над проєктом використовувалась система управління завданнями Taiga.io та інструмент для візуального планування Mural, що дозволило ефективно реалізувати підхід за методологією Scrum. Процес тестування включав написання юніт-тестів, а також проведення ручного (мануального) тестування з метою перевірки стабільності та коректності роботи застосунку на різних етапах розробки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Проектування інформаційних систем в умовах військових технологій / Ю. Шевченко. – Київ: Видавництво Національного університету оборони України, 2020. – 250 с.
2. JWT Authentication: Огляд концепцій та реалізація / К. Сидоренко. – К.: КНУ, 2021. – 150 с.
3. Проблема створення інформаційної системи логістики в ЗСУ [Електронний ресурс] – Режим доступу до ресурсу: <http://dspace.nbu.gov.ua/handle/123456789/150912>
4. Планування логістичних процесів у повсякденній діяльності військових підрозділів Національної гвардії України – Режим доступу до ресурсу: http://pev.kpu.zp.ua/journals/2021/1_24_ukr/24.pdf
5. Порядок логістичного забезпечення сил [Електронний ресурс] – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/laws/show/1208-2018-%D0%BF#Text>
6. Kryvoruchko M., Shirokopetleva M., Vechur O. Trip Planning Software System Design // International Conference on Software Engineering : proceedings of the conference (April 10–12, 2022). – 2022. – P. 97–100.
7. Голян В. В., Міхневич І. Д. Програма реалізації безпеки застосувань для обміну інформацією // Зб. наук. пр. за матеріалами 22-го Міжнародного молодіжного форуму «Радіоелектроніка і молодь у ХХІ столітті». – 2019. – С. 119–120.
8. Вечур О. В., Назаров О. Ю. Використання технологій та методів CI/CD для розгортання коду в хмарному середовищі // Збірник наукових праць «ЛОГОΣ». – 2021
9. GitHub – Режим доступу до ресурсу: https://github.com/NureIvakinNikita/PZPI-21-5_Ivakin

ДОДАТОК А

Діаграма прецендентів



ДОДАТОК Б

ER-діаграма



ДОДАТОК В
Сертифікат участі у виставці



ДОДАТОК Г

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

**StrikePlagiarism.com**



Дата звіту **6/3/2025**
Дата редагування ---



 Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ-21-5_Івакін_Н_М_скорочений
Автор
Івакін Нікіта Максимович / **Олена Олійник**
Науковий керівник / Експерт
підрозділ
каф. ПІ




Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.





Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		0

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Копір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз			Копір тексту
ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	https://openarchive.nure.ua/bitstreams/ece77560-01ff-451d-b29e-e016f4fc93db/download	10	0.12 %
з бази даних RefBooks (0.00 %)			
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
з домашньої бази даних (0.00 %)			

ДОДАТОК Г

Слайди презентації



Програмна система
для контролю
логістичного
забезпечення війська.
Мобільний
застосунок.



Івакін Нікіта Максимович, ПЗПІ-21-5
Керівник: доц. кафедри ПІ Назаров Олексій
Сергійович

12 червня 2025

Мета роботи

Мета роботи:

Розробити мобільний застосунок UALogistics, призначений для підтримки логістичного забезпечення військових підрозділів. Застосунок має надавати:

- перегляду та обліку ресурсів;
- створення запитів на постачання;
- перегляд бойових місій та логістичних запитів до них;
- відмічання прийняття поставок, відміна та їх редагування.

Актуальність теми:

У контексті повномасштабної війни в Україні виявлено критичні проблеми у військовій логістиці: низький рівень автоматизації, обмежена прозорість та складність контролю за ресурсами, що призводить до помилок і зловживань. У зв'язку з оборонною реформою та інтеграцією з НАТО постала потреба у сучасній програмній системі для прозорого й ефективного забезпечення ЗСУ. Така система покращить облік і розподіл ресурсів, забезпечить швидкий обмін даними між підрозділами та підвищить оперативність і точність управлінських рішень у бойових умовах.



Аналіз проблеми (аналіз існуючих рішень)

Досліджені конкурентні рішення:

Logfas (АТО)

- Високий рівень автоматизації
- Відсутність доступу до коду системи, що унеможливує її адаптацію
- Складний інтерфейс та довгий процес навчання персоналу

SAP ERP для військових структур

- Потужний функціонал
- Висока вартість впровадження та складність налаштування
- Проблеми зі збереженням секретної інформації та впровадження цього у процесу на законодавчому рівні України

Системи обліку ресурсів радянського зразка

- На разі підходить, однак з майбутніми змінами, які відбуваються зараз є не гнучким
- Повністю паперовий або напівавтоматизований облік
- Високий ризик втрати/спотворення інформації, через людський фактор



Основні недоліки наявних аналогів:

- Недостатня адаптованість до потреб сучасного українського війська
- Відсутність зменшених варіантів систем, які могли б використовуватися з планшетів або мобільних застосунків для командирів у бойових зонах
- Надмірна складність та вартість впровадження для локального використання
- Низька гнучкість

Висновок:

Створення системи з можливістю підключення мобільних застосунків, яке є безпечним, гнучким, легким у використанні та масштабованим орієнтованим на потреби Збройних Сил України

3

Постановка задачі та опис системи

Формулювання проблеми:

У польових умовах військові підрозділи стикаються з труднощами у швидкому доступі до логістичної інформації, оформленні запитів на ресурси та контролі їх постачання.

Відсутність мобільного інструменту призводить до:

- затримок у реагуванні та постачанні ресурсів
- підвищеного ризику помилок через ручний облік
- ускладнення оперативного планування та координації дій між підрозділами
- неможливості ефективного планування

Мета розробки:

Розробити зручний, безпечний та масштабований мобільний застосунок, що є частиною модульної логістичної системи для військових підрозділів. Застосунок має забезпечити:

швидкий доступ до інформації про наявні ресурси
можливість створення та управління бойовими місіями
оформлення та обробку логістичних запитів у реальному часі
контроль статусів постачання та отримання сповіщень про зміни

Очікувані результати:

Інтуїтивно зрозумілий мобільний інтерфейс для командирів

Повноцінний мобільний застосунок з MVVM архітектурою з можливістю впровадження додаткових сервісів

Можливість подальшого впровадження в ЗСУ



4

Вибір технологій розробки

Мова програмування:

Kotlin – сучасна, лаконічна мова для Android-розробки, що забезпечує високий рівень безпеки та сумісності з Jetpack-компонентами.

Фреймворки та бібліотеки:

Jetpack ViewModel, LiveData – для реактивної обробки стану UI.

Hilt – DI-фреймворк від Google для автоматичного впровадження залежностей, включно з підтримкою інструментальних тестів.

Retrofit + Moshi – для зручної роботи з REST API та JSON-серіалізацією.

OkHttp – мережевий клієнт з підтримкою логування HTTP-запитів.

AndroidX (Activity, RecyclerView, Lifecycle, AppCompat) – для забезпечення зворотної сумісності та стабільної роботи UI-компонентів.

Аргументи вибору:

Всі компоненти є сучасними, активно підтримуються Google та спільнотою.

Висока сумісність між бібліотеками дозволяє масштабувати застосунок без втрати стабільності.

Забезпечено повноцінну підтримку модульного тестування, що критично важливо для військових рішень з підвищеними вимогами до надійності.



5

Архітектура створеного програмного забезпечення

Тип архітектури системи:

Клієнт-серверна архітектура, оскільки мобільний застосунок функціонує як клієнт, що взаємодіє з серверною частиною через REST API. Це дозволяє зберігати дані централізовано, зменшує навантаження на пристрій користувача та забезпечує єдину точку контролю.

Взаємодія із сервером:

Протокол – REST API

Формат даних – JSON

Безпека – авторизація за допомогою JWT токенів, передача токена в заголовках HTTP-запитів. мережевої взаємодії з backend-сервісами.



Ключові компоненти системи:

Model та Api layer – класи та інтерфейси, що відповідають за взаємодію з сервером та обробку запитів і отриманих даних.

UI layer – це шар мобільного застосунку, який містить такі компоненти як Activity та ViewModel. Перші відповідають за відображення компонентів інтерфейсу та їх зміну. ViewModel зберігає та керує даними для UI, робить коли використовуючи API з data layer.

Також головним елементом системи є файли ресурсів, а саме layouts, що відповідають за початкову розмітку сторінок застосунку та їх стилі.



6

Опис програмного забезпечення, що було використано у дослідженні

Середовища розробки використані у ході дослідження:

- IntelliJ IDEA Ultimate
- Android Studio Ladybug
- Visual Studio Code

Мови програмування:

- Java – для написання серверної логіки.
- Kotlin – для розробки мобільного застосунку під Android.
- TypeScript – для фронтенд застосунку.

Інструменти для тестування:

- JUnit + MockK – для юніт-тестування логіки застосунку.
- Espresso + Hilt Testing – для інструментального тестування взаємодії користувача з UI та перевірки логіки з DI.

Інші інструменти:

- Docker – контейнеризація бекенд-сервісів для розгортання.
- Postman – ручне тестування REST API.
- Gradle – інструмент автоматизації збірки для розробки багатомовного програмного забезпечення.
- SwaggerUI – інтерактивний веб-інтерфейс, який дозволяє розробникам візуалізувати, досліджувати та тестувати API (інтерфейси прикладного програмування), створені за допомогою специфікації OpenAPI.



Дизайн системи

Мобільний застосунок UALogistics реалізовано з використанням архітектурного шаблону MVVM (Model–View–ViewModel), що забезпечує чітке розділення відповідальностей, покращує масштабованість та полегшує тестування застосунку.

Послідовність обробки:

- Користувач натискає кнопку (наприклад, "Видалити ресурс").
- View передає запит у ViewModel. ViewModel викликає відповідний метод у Model (через ResourceApi або TokenManager).
- Дані обробляються асинхронно за допомогою Kotlin Coroutines.
- Отриманий результат оновлює StateFlow, що автоматично оновлює View.
- У разі помилки – ViewModel генерує повідомлення, яке View відображає у вигляді Snackbar.

Захист та безпека:

JWT (JWE) – мобільний застосунок зберігає та передає зашифрований токен авторизації, виданий сервером після успішного входу

TokenManager + DataStore – токен зберігається локально у зашифрованому вигляді через Jetpack DataStore, що гарантує стійкість до втрати при перезапуску застосунку.



Менеджер токенів та структура системи

```
private val Context.dataStore by preferencesDataStore(name = "auth_prefs")

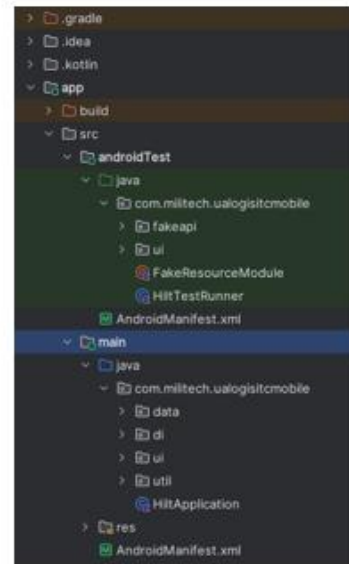
@Singleton
open class TokenManager @Inject constructor(
    @ApplicationContext private val context: Context
) {

    companion object {
        private val TOKEN_KEY = stringPreferencesKey(name = "jet_token")
        private val USER_ID_KEY = longPreferencesKey(name = "user_id")
        private val UNIT_ID_KEY = longPreferencesKey(name = "unit_id")
    }

    open val token: Flow<String?> = context.dataStore.data.map { it[TOKEN_KEY] }
    open val userId: Flow<Long?> = context.dataStore.data.map { it[USER_ID_KEY] }
    open val unitId: Flow<Long?> = context.dataStore.data.map { it[UNIT_ID_KEY] }

    open suspend fun saveToken(token: String) {
        context.dataStore.edit { it[TOKEN_KEY] = token }
    }

    open suspend fun saveUserId(id: Long) {
        context.dataStore.edit { it[USER_ID_KEY] = id }
    }
}
```



9

Тестування

Тип тестування:

Мобільний застосунок було протестовано з використанням інструментального (UI) тестування.

Тести реалізовані з використанням Espresso, ці тести перевіряють правильність відображення елементів інтерфейсу, роботу діалогових вікон, кнопок, RecyclerView та зміну стану після взаємодії. Також використовуються Espresso Intents для перевірки навігації між екранами.

Технології тестування:

JUnit 4 – основний фреймворк для unit-тестів.

MockK – мокання залежностей (API, TokenManager).

Espresso – перевірка UI-логіки, роботи діалогів, введення тексту, натискання кнопок.

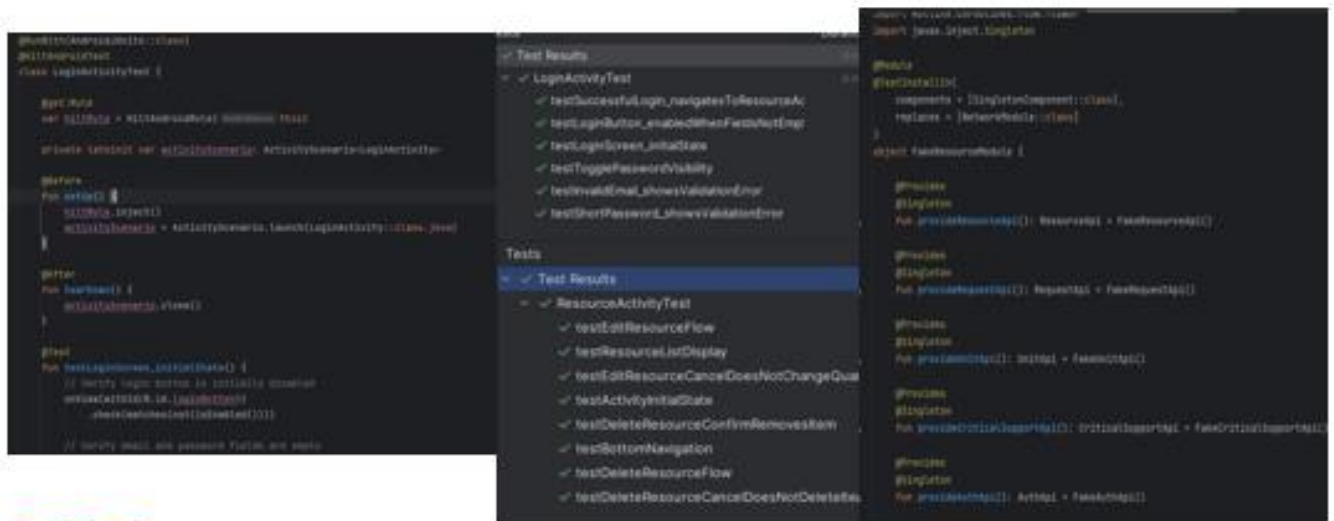
Hilt Testing – забезпечує впровадження тестових залежностей (@TestInstallIn) у середовищі Android.

Coroutine Test Library – симуляція асинхронного виконання корутин.



10

Тестування



Публікація результатів

Проект UALogistics було представлено на 29-му Міжнародному молодіжному форумі "Радіоелектроніка та молодь у XXI столітті".

Отримано 1 місце в номінації: "Програмне забезпечення. Програми".



Підсумки

Результати роботи:

- Розроблено мобільний застосунок системи UALogistics для Android, що забезпечує зручний доступ командирів та логістів до ключових функцій логістичного забезпечення.
- Реалізовано функціонал для перегляду та управління ресурсами, створення логістичних запитів, взаємодії з місіями та контролю статусу постачання.
- Забезпечено безпечну авторизацію з використанням JWT-токенів, а також збереження ключових даних за допомогою DataStore.

Практичне значення:

- Мобільний застосунок дає змогу швидко реагувати на потреби підрозділів у польових умовах, зменшуючи час на обробку запитів.
- Сприяє прозорості обліку ресурсів, покращує взаємодію між підрозділами та підвищує ефективність управління.

Можливості подальшого розвитку:

- Розширення застосунку для роботи в офлайн-режимі з подальшою синхронізацією.
- Додання сервісу з передбаченням розтррати ресурсів.
- Додання додаткових функцій в залежності від типу бойової групи та її специфікації.

ДОДАТОК Д

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЗМІСТ

1 ВСТУП	Ошибка! Закладка не определена.
1.1 Огляд продукту	Ошибка! Закладка не определена.
1.2 Мета	Ошибка! Закладка не определена.
1.3 Межі.....	Ошибка! Закладка не определена.
1.4 Посилання	Ошибка! Закладка не определена.
1.5 Означення та аббревіатури.....	Ошибка! Закладка не определена.
2 ЗАГАЛЬНИЙ ОПИС	Ошибка! Закладка не определена.
2.2 Функції продукту	Ошибка! Закладка не определена.
2.3 Характеристики користувачів	Ошибка! Закладка не определена.
2.4 Загальні обмеження	Ошибка! Закладка не определена.
2.5 Припущення й залежності.....	Ошибка! Закладка не определена.
3 КОНКРЕТНІ ВИМОГИ	Ошибка! Закладка не определена.
3.1 Вимоги до зовнішнього інтерфейсу	Ошибка! Закладка не определена.
3.1.1 Інтерфейс користувача	Ошибка! Закладка не определена.
3.1.2 Комунікаційний протокол.....	Ошибка! Закладка не определена.
3.1.3 Функції продукту	Ошибка! Закладка не определена.
3.1.4 Припущення й залежності	Ошибка! Закладка не определена.
3.2 Властивості програмного продукту	Ошибка! Закладка не определена.
3.3 Атрибути програмного продукту	Ошибка! Закладка не определена.
3.3.1 Надійність	Ошибка! Закладка не определена.
3.3.2 Доступність.....	Ошибка! Закладка не определена.
3.3.3 Безпека	Ошибка! Закладка не определена.
3.4 Вимоги до бази даних	Ошибка! Закладка не определена.

1 ВСТУП

1.1 Огляд продукту

Мобільний застосунок UA Logistics – це клієнтська частина інформаційної системи, призначеної для автоматизації логістичного забезпечення військових підрозділів у реальному часі. Застосунок реалізовано на платформі Android із фокусом на використання в бойових умовах – безпосередньо командирами, логістами та оперативним персоналом у зоні дій.

Мобільний клієнт забезпечує доступ до ключових функцій системи: перегляд наявних ресурсів, створення та редагування логістичних запитів, підтвердження поставок і взаємодію з бойовими місіями. Він створений з урахуванням специфіки польових умов, що включає нестабільний зв'язок, обмежений доступ до інфраструктури та потребу в простому й швидкому інтерфейсі.

Продукт відповідає високим вимогам до адаптивності та масштабованості, а також є частиною єдиної цифрової екосистеми логістичного забезпечення Збройних Сил України. Завдяки модульності архітектури застосунок легко підтримується й готовий до подальшого розвитку – зокрема офлайн-режиму, прогнозування потреб та розширення функцій згідно з новими бойовими сценаріями.

1.2 Мета

Метою цього документа зі специфікації вимог до програмного забезпечення (Software Requirements Specification – SRS) є формалізація функціональних і нефункціональних вимог до мобільного застосунку UA

Logistics, який є частиною інформаційної системи для підтримки логістичного забезпечення військових підрозділів.

Документ призначений для команди розробників, тестувальників, проєктних менеджерів, а також військових замовників, щоб забезпечити спільне бачення архітектури, функціоналу й обмежень мобільного клієнта. Він виступає базою для технічного проєктування, реалізації та перевірки якості розробленого застосунку.

UA Logistics спрямований на автоматизацію та спрощення ключових логістичних процесів безпосередньо в польових умовах. Основними завданнями системи є: перегляд та управління ресурсами бойового підрозділу, створення і контроль логістичних запитів, взаємодія з місіями та запит критичної підтримки у разі надзвичайної ситуації. Застосунок також забезпечує авторизацію користувачів, збереження конфіденційних даних на пристрої, та обмін інформацією з сервером у режимі реального часу.

Таким чином, мета даного проєкту – розробити мобільну-частину критично важливої інформаційної системи, яка не лише відповідає поточним потребам військової логістики, а й створює фундамент для її подальшої цифрової трансформації.

1.3 Межі

Мобільний застосунок UA Logistics розроблено як частину єдиної системи логістичного забезпечення Збройних Сил України. Його основна функція – забезпечити командирів, логістів і відповідальних осіб інструментом для оперативного управління запитами на постачання, контролю за ресурсами підрозділу та взаємодії з поточними місіями у реальному часі. Застосунок дозволяє здійснювати перегляд, створення, редагування логістичних запитів, а

також підтвердження отримання поставань, що є критично важливим у польових умовах.

Програмний продукт не призначений для управління всією логістичною інфраструктурою в ізоляції, а виступає фронт-інтерфейсом до вже розгорнутої серверної частини. Він не виконує централізоване зберігання даних і не обробляє фінансові, контрактні або стратегічні аспекти логістики. Вся логіка обробки запитів і збереження інформації делегується серверу через захищені REST API.

Застосунок орієнтований на пристрої з Android 6.0+ та оптимізований для використання у середовищі з обмеженим або нестабільним інтернет-з'єднанням. Водночас він не передбачає функціоналу автономного зберігання великих обсягів даних або офлайн-обробки запитів без попередньої синхронізації.

Таким чином, межі функціоналу чітко окреслені: застосунок реалізує інтерфейс для оперативного управління логістичними процесами на рівні бойової одиниці, а також виступає інструментом для швидкої взаємодії з серверною частиною системи в рамках визначених повноважень користувача.

1.4 Посилання

— ISO/IEC/IEEE 29148:2018 – Systems and software engineering – Life cycle processes – Requirements engineering.

— ISO/IEC 25010:2011 – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.

1.5 Означення та аббревіатури

UALogistics – назва розробленої системи: Ukrainian Army Logistics.

JWT (JSON Web Token) – формат токена для передачі даних автентифікації у захищеному вигляді.

JWE (JSON Web Encryption) – розширення JWT, що забезпечує шифрування вмісту токена.

REST (Representational State Transfer) – архітектурний стиль побудови веб-сервісів з використанням HTTP.

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертекстових даних у мережі Інтернет.

CI/CD (Continuous Integration / Continuous Deployment) – підходи до безперервної інтеграції і доставки оновлень у систему.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Мобільний застосунок UA Logistics є складовою частиною комплексної системи логістичного забезпечення військових підрозділів, яка включає також серверну частину з REST API та веб-клієнт для адміністраторів. Застосунок безпосередньо залежить від наявності стабільного мережевого з'єднання та коректної роботи серверної інфраструктури, з якою він взаємодіє через авторизовані HTTP-запити.

Для повноцінного користування системою необхідно пройти авторизацію, яка реалізується за допомогою JWT-токена. Усі функції – перегляд ресурсів, створення місій, подання запитів та підтвердження постачання – доступні лише авторизованим користувачам із роллю командира або логіста.

У майбутньому передбачено розширення функціоналу за рахунок інтеграції з аналітичним модулем, системами моніторингу бойових дій та IoT-пристроями. Також планується реалізація офлайн-режиму з локальним кешуванням критичних даних та подальшою синхронізацією, коли з'єднання буде відновлено. Це дозволить підвищити стійкість системи до нестабільного інтернет-зв'язку у бойових умовах.

2.2 Функції продукту

Мобільний застосунок UA Logistics призначений для підтримки процесів логістичного забезпечення військових підрозділів у реальному часі. Він дозволяє користувачам – командирам, логістам та оперативному персоналу – взаємодіяти з інформаційною системою для перегляду, створення та підтвердження логістичних запитів, керування ресурсами, місіями та запитами на критичну

підтримку. Доступ до функцій надається згідно з роллю користувача, визначеною під час авторизації.

Основні функції системи:

FR-1: Авторизація користувача – перевірка токена доступу та отримання повноважень (роль, ідентифікатор користувача, підрозділу).

FR-2: Перегляд ресурсів підрозділу – відображення списку доступних ресурсів з можливістю фільтрації та сортування.

FR-3: Додавання нового ресурсу – заповнення форми з даними ресурсу та відправка на сервер.

FR-4: Редагування або видалення ресурсу – зміна кількості або повне видалення ресурсу з підтвердженням через діалог.

FR-5: Створення логістичного запиту – вибір місії, додавання необхідних ресурсів, відправка запиту на сервер.

FR-6: Перегляд місій – відображення актуальних місій підрозділу з переходом до їхніх логістичних запитів.

FR-7: Перегляд логістичних запитів – фільтрація за статусом (очікується, схвалено, відхилено, виконано), доступ до змісту запиту.

FR-8: Підтвердження або скасування постачання – комунікація між логістами та командирами, відображення змін у статусі.

FR-9: Запит на критичну підтримку – швидке звернення до командування у надзвичайній ситуації з описом та геопозицією.

FR-10: Виведення повідомлень (Snackbar) – інформування користувача про успішні чи неуспішні дії.

Уся взаємодія реалізована через REST API із захищеною авторизацією. Інтерфейс адаптовано для швидкого використання у польових умовах з

урахуванням обмежень мобільного зв'язку та обмеженого часу прийняття рішень.

2.3 Характеристики користувачів

Мобільний застосунок UA Logistics розрахований на використання у військовому середовищі, зокрема тими, хто безпосередньо залучений до логістичного забезпечення підрозділів у зоні бойових дій або в тилу. Основними користувачами є командири бойових груп, військові логісти, оперативний персонал та адміністративні особи.

Ролі користувачів:

Командири підрозділів – здійснюють контроль за логістикою, формують запити на постачання ресурсів, приймають або відхиляють поставки, переглядають місії та стан ресурсів.

Логісти – відповідають за обробку запитів, підтвердження доставок, оновлення складу та взаємодію з командирами.

Оперативний персонал – взаємодіє із застосунком для ознайомлення з місіями, перегляду статусу ресурсів та виконання службових завдань.

Адміністратори – мають доступ до налаштувань системи, тестування та моніторингу логістичних процесів.

Особливості користувачів:

Рівень цифрової грамотності може варіюватися: від базового у новобранців до високого у персоналу штабу. Тому інтерфейс має бути інтуїтивно зрозумілим і мінімалістичним.

Фізичні умови використання – застосунок використовують в польових умовах, під час бойових дій або переміщення, де можливі перебої зв'язку та обмежена увага користувача.

Вікові категорії – від 20 до 50 років, переважно чоловіки, які проходять військову службу або залучені до логістичних функцій.

Із урахуванням зазначених характеристик мобільний застосунок має бути швидким, зручним, надійним, із мінімальною кількістю кроків до виконання основних дій, а також адаптованим до роботи без постійного інтернету та на пристроях з невисокими технічними характеристиками.

2.4 Загальні обмеження

Розробка мобільного застосунку UA Logistics супроводжувалась низкою обмежень, які необхідно враховувати під час впровадження, використання та подальшого розвитку системи:

Обмежений час на розробку не дозволив реалізувати весь запланований функціонал, зокрема частину механізмів безпеки, розширену статистику використання, офлайн-доступ до запитів та багатомовну підтримку інтерфейсу.

Залежність від серверної частини: застосунок тісно інтегрований із сервером через REST API. Усі критично важливі дії – авторизація, створення запитів, синхронізація даних – вимагають стабільного з'єднання з бекендом. У разі збою або відсутності доступу до сервера функціональність обмежується.

Мобільна платформа: наразі підтримується лише Android 6.0 і вище. Версія для iOS або кросплатформенне рішення поки не розроблені.

Обмеження апаратних ресурсів: застосунок орієнтовано на роботу навіть на пристроях із середніми технічними характеристиками, що частково впливає на використання складних візуальних компонентів чи важких анімацій.

Безпека в процесі доопрацювання: хоча застосунок використовує авторизацію через JWT та передачу даних через HTTPS, в майбутньому планується впровадження додаткових заходів захисту (наприклад, автоматичне

вигирання даних після тривалого простою або спроби несанкціонованого доступу).

Незважаючи на ці обмеження, система залишається функціональною, масштабованою та придатною для використання в реальних умовах, з перспективою подальшого розвитку та вдосконалення.

2.5 Припущення й залежності

Повноцінне використання мобільного застосунку UALogistics можливе за умови дотримання низки припущень і зовнішніх залежностей, які впливають на стабільність та функціональність системи:

- Передбачається, що користувачі (командири, логісти, адміністративний персонал) володіють базовими навичками роботи з мобільними пристроями на Android та здатні самостійно орієнтуватись у навігації застосунку.

- Необхідною умовою для роботи є стабільне підключення до мережі Internet, оскільки значна частина функціоналу (авторизація, синхронізація запитів, отримання місій) реалізована через клієнт-серверну взаємодію з REST API.

- Робота системи залежить від доступності серверної частини, яка повинна бути постійно підтримуваною та захищеною з боку розробників.

- Для забезпечення авторизації та зберігання токенів передбачається використання сервісу DataStore, а також коректне функціонування механізму обробки JWT.

- Для роботи з мережевими запитами застосовуються бібліотеки Retrofit, Moshi і OkHttp, тому система залежить від їхньої підтримки, сумісності з поточною версією Android SDK та відповідного оновлення у випадку зміни API.

— Безпечне зберігання конфіденційних даних на сервері реалізовано із використанням Vault від HashiCorp та шифруванням чутливих полів через AES-256, тому важливою залежністю є підтримка цих механізмів у бекенді [6].

— У майбутньому можливе підключення зовнішніх сервісів. Їхня доступність і тарифи можуть вплинути на функціональність системи або її витрати в продакшн-середовищі.

— Масштабне впровадження в межах ЗСУ або інших оборонних структур передбачає організаційну, юридичну та інфраструктурну підтримку з боку відповідальних органів. Без такого супроводу система не зможе функціонувати на національному рівні.

Застосунок також передбачає, що користувачі мають зареєстрований обліковий запис у системі, і всі дії здійснюються у рамках визначених ролей (логіст, командир, адміністратор), що задаються з боку серверної інфраструктури.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішнього інтерфейсу

3.1.1 Інтерфейс користувача

Інтерфейс мобільного застосунку UA Logistics розроблено з урахуванням вимог до зручності, швидкодії та адаптивності в польових умовах (див. Рисунок 3.1-3.2).

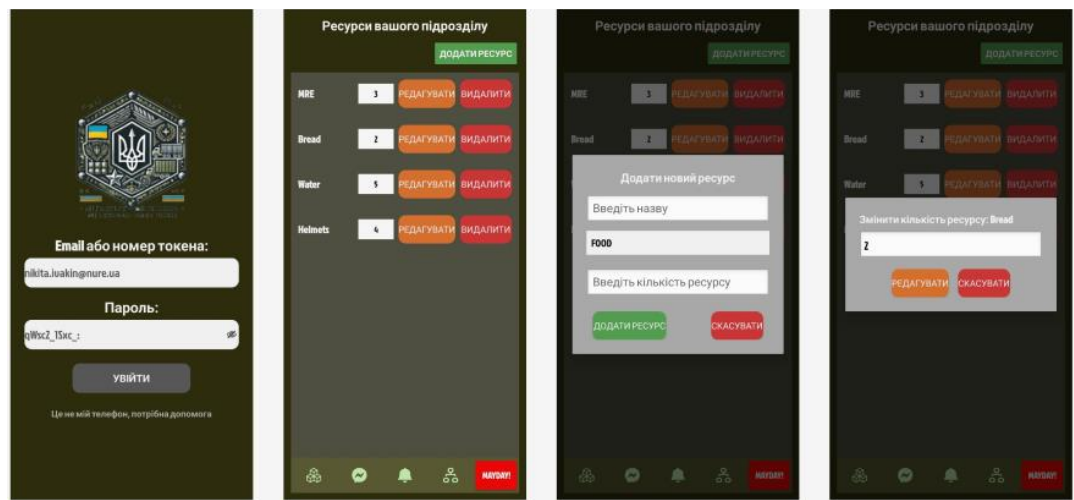


Рисунок 3.1 – скріншоти інтерфейсу мобільного застосунку

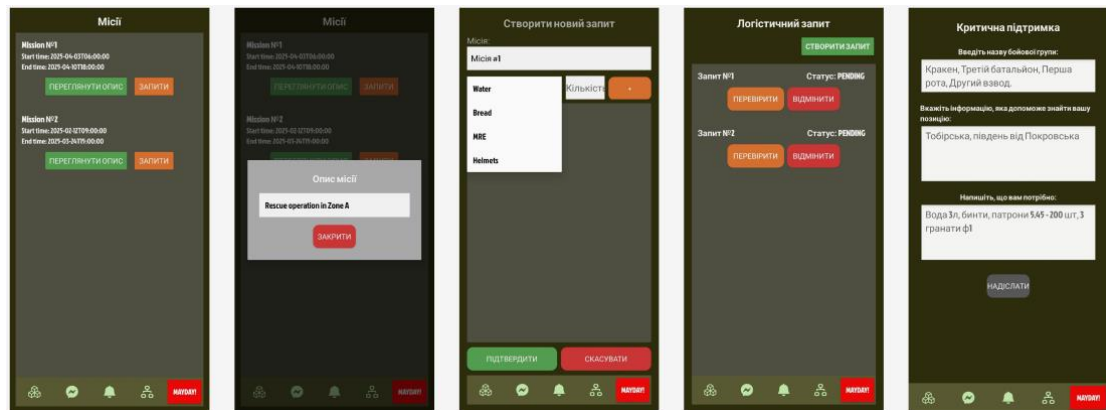


Рисунок 3.2 – скріншоти інтерфейсу мобільного застосунку друга частина

Інтерфейс забезпечує інтуїтивний доступ до основних функцій, таких як створення логістичних запитів, перегляд місій, керування ресурсами та підтвердження поставок.

3.1.2 Комунікаційний протокол

Мобільний застосунок UA Logistics взаємодіє з серверною частиною системи через HTTP-протокол, дотримуючись принципів REST-архітектури. Такий підхід дозволяє досягти високої сумісності, простоти реалізації та передбачуваності обміну даними між клієнтом і сервером, що є критично важливим для застосунку, який функціонує в умовах обмеженого зв'язку.

Уся передача даних здійснюється у форматі JSON – універсальному та легкому форматі обміну, який забезпечує швидке парсингування на мобільному пристрої. Запити до REST API ініціюються з мобільного клієнта через бібліотеку Retrofit у зв'язці з Moshi для перетворення JSON-об'єктів у Kotlin-моделі. Для мережевого рівня застосовується OkHttp, що забезпечує обробку HTTP-запитів з урахуванням таймаутів, ретраїв і логування.

Також вже передбачає можливість безшовного переходу на HTTPS у продуктивному середовищі. Для автентифікації використовується механізм передачі JWT-токенів у заголовках запитів, що дозволяє зберігати сесію користувача без повторної авторизації.

Варто зазначити, що мобільний застосунок виконує весь обмін даними відбувається через централізовану точку доступу (API Gateway або відповідні ендпоінти), що спрощує логіку клієнта та зменшує кількість залежностей.

3.1.3 Функції продукту

Мобільний застосунок UA Logistics реалізує ключові функції, необхідні для оперативного управління логістичними процесами в польових умовах. Його функціональність орієнтована на зручну взаємодію командирів, логістів та інших представників військових підрозділів із цифровою інфраструктурою постачання. Увесь доступ до функцій здійснюється згідно з роллю користувача, яка визначається серверною частиною після авторизації.

Для командирів бойових частин мобільний застосунок дозволяє:

- авторизація у застосунку
 - переглядати ресурси власної бойової групи, їх редагування та видалення.
- можливість додання ресурсів отриманих під час виконання бойового завдання.
- контролювати статус запитів і отриманих поставань.
 - переглядати активні місії та деталізацію щодо ресурсного забезпечення.
 - можливість створення логістичних запитів для певних місій.
 - у критичних ситуаціях – ініціювати запит на екстрену підтримку.

— зміна статусу запитів (наприклад, підтвердження постачання або відмова).

3.1.4 Припущення й залежності

Функціонування мобільного застосунку UA Logistics передбачає низку технічних, інфраструктурних та організаційних умов, дотримання яких є критичним для його стабільної роботи. Визначення цих припущень і залежностей дозволяє чітко окреслити межі використання застосунку в реальних умовах та вчасно виявляти потенційні ризики.

Технологічні припущення полягають у тому, що застосунок працює на сучасних Android-пристроях (версія 6.0 і вище), що підтримують стабільне інтернет-з'єднання. Передбачається, що користувач має базові навички користування мобільними інтерфейсами та здатен взаємодіяти з функціями логістичного обліку в умовах обмеженого часу або ресурсу.

Інфраструктурні залежності охоплюють взаємодію з серверною частиною через REST API. Уся логіка авторизації базується на JWT-токенах, які зберігаються на клієнті у зашифрованому вигляді через DataStore. Для обміну даними використовуються бібліотеки Retrofit, Moshi та OkHttp, стабільна робота яких залежить від сумісності з версією Android SDK і налаштувань серверного оточення. У продакшн-середовищі всі з'єднання мають здійснюватися по HTTPS.

Крім того, передбачається, що користувачі вже мають створений обліковий запис у системі з визначеною роллю (логіст, командир, адміністратор), оскільки мобільний застосунок не реалізує повний функціонал створення чи управління користувачами.

Наразі система не підтримує офлайн-режим, тому для виконання більшості операцій потрібен стабільний доступ до мережі. У подальших релізах планується

впровадження можливостей локального кешування та синхронізації після відновлення з'єднання.

Також розглядається перспективна інтеграція мобільного застосунку з захищеними системами зберігання (наприклад, Vault від HashiCorp) для автоматизованого управління ключами доступу, а також реалізація шифрування даних на клієнті з використанням алгоритму AES-256, як це вже запроваджено для конфіденційної інформації на сервері. Це дозволить значно посилити захист чутливих даних у разі втрати пристрою або спроби його компрометації.

3.2 Властивості програмного продукту

Мобільний застосунок UA Logistics розроблений відповідно до сучасних вимог щодо якості, продуктивності та інформаційної безпеки. Його функціональність орієнтована на використання в складних умовах польового середовища, що накладає особливі вимоги до стабільності, інтуїтивності інтерфейсу та захищеності даних.

Застосунок забезпечує стійку роботу на Android-пристроях 6.0+, з оптимізацією під пристрої з обмеженими апаратними ресурсами. Завдяки застосуванню архітектурного підходу MVVM та використанню бібліотек Android Jetpack (ViewModel, Lifecycle), система підтримує плавне оновлення даних без перезапуску інтерфейсу та без збоїв.

Інформаційна безпека реалізована через авторизацію з використанням JWT-токенів та захищене збереження даних у локальному сховищі DataStore. Чутлива інформація в мережевих запитах буде додатково шифруватися із застосуванням алгоритму AES-256, що вже використовується в серверній частині для захисту бази даних. У перспективі передбачено механізми витирання даних у разі спроб несанкціонованого доступу або тривалого простою.

Мобільність та зручність використання забезпечуються інтуїтивним дизайном, адаптованим до роботи в польових умовах: контрастні кольори, спрощена навігація, адаптивні кнопки та підказки. Використання бібліотек Retrofit, Moshi та OkHttp дозволяє мінімізувати затримки при обміні даними з сервером навіть у умовах нестабільного зв'язку.

Масштабованість застосунку досягається завдяки модульному підходу: окремі функціональні блоки легко доповнюються або замінюються без необхідності повної перебудови програми. Вся логіка доступу до API реалізована централізовано у NetworkModule, що спрощує підтримку та зміну конфігурацій.

Також застосунок легко оновлюється завдяки централізованій структурі лейаутів і ресурсів, які зберігаються у відповідних XML-файлах (res/layout, res/values) з підтримкою локалізації для кількох мов.

Таким чином, мобільний застосунок UA Logistics поєднує в собі продуктивність, безпеку, зручність та гнучкість, що робить його ключовим компонентом цифрової логістичної платформи.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Надійність мобільного застосунку UA Logistics є ключовим аспектом, оскільки він використовується в умовах нестабільного зв'язку та підвищеного навантаження на місцях. Застосунок забезпечує стабільну роботу навіть при тимчасовій втраті мережі – усі критичні дані зберігаються локально через DataStore і синхронізуються із сервером при відновленні підключення.

Архітектура MVVM із розділенням логіки, UI та джерел даних підвищує стійкість до збоїв, а валідація введених даних і обробка винятків запобігають аварійним завершенням. Компоненти перевіряються через unit- і UI-тести, що

дозволяє швидко виявляти і виправляти помилки до виходу в продуктивне середовище.

Таким чином, застосунок забезпечує надійну роботу в польових умовах, гарантуючи доступ до логістичної інформації без збоїв і втрати даних.

3.3.2 Доступність

Доступність мобільного застосунку UA Logistics має вирішальне значення, з огляду на його використання в умовах бойових дій або обмеженого зв'язку. Застосунок розроблено з урахуванням можливості роботи в нестабільному середовищі, де постійне підключення до мережі не завжди доступне.

Завдяки використанню локального сховища DataStore, основна інформація (наприклад, токени авторизації, `userId`, `unitId`) зберігається на пристрої, що дозволяє частково працювати із застосунком навіть при відсутності інтернету. Мережеві запити виконуються асинхронно через Retrofit з підтримкою OkHttp, що дозволяє реалізувати повторні спроби з'єднання або обробку помилок без втрати контексту.

Інтерфейс побудований на основі AndroidX компонентів – зокрема, Lifecycle, ViewModel і Navigation, що забезпечує стабільну та адаптивну роботу навіть при зміні стану пристрою (наприклад, поворот екрана або перезапуск активності).

Доступ до функцій застосунку регулюється через систему ролей (логіст, командир, адміністратор), що дозволяє обмежити навантаження на систему та підвищити безпеку. Крім того, інтерфейс спроектовано з урахуванням принципів доступності – він лаконічний, контрастний та оптимізований для швидкого доступу до критичних функцій.

У майбутньому планується реалізація механізму кешування основних даних для перегляду історії запитів, ресурсів і місій у повністю офлайн-режимі з подальшою синхронізацією після відновлення зв'язку. Це дозволить користувачам зберігати контроль над логістичними процесами навіть у зоні бойових дій без постійного інтернету.

3.3.3 Безпека

Безпека мобільного застосунку UA Logistics є пріоритетною, з огляду на його використання у військовій сфері та взаємодію з критично важливою інформацією. Застосунок передбачає аутентифікацію через JWT-токени, які зберігаються у зашифрованому локальному сховищі DataStore, що унеможливорює несанкціонований доступ до персональних даних навіть у разі доступу до пристрою.

Рольова модель доступу обмежує можливості користувача відповідно до його повноважень (командир, логіст, адміністратор), що знижує ризики помилкових або зловмисних дій.

Уся взаємодія із сервером відбувається через безпечні HTTP-запити з токенами авторизації, а критичні запити додатково перевіряються на рівні ролей. У майбутньому планується повний перехід на HTTPS-протокол для шифрування всього трафіку.

Для підвищення безпеки даних під час передачі, в наступних версіях застосунку планується впровадження шифрування вмісту запитів за допомогою алгоритму AES-256, що вже використовується на сервері для захисту чутливих полів у базі. AES-256 був обраний завдяки своїй криптостійкості, перевірності у промислових рішеннях та підтримці на платформі Android.

Також передбачається реалізація механізму самознищення локальних даних після тривалого періоду неактивності або виявлення потенційної спроби зламу (наприклад, багаторазових невдалих спроб входу).

Таким чином, мобільний застосунок UA Logistics реалізовує багаторівневий підхід до безпеки, включаючи шифрування, обмеження доступу, захист сесій і запобігання компрометації локальних даних, що відповідає сучасним вимогам до мобільних рішень військового призначення.

3.4 Вимоги до бази даних

На поточному етапі мобільний застосунок UA Logistics працює як клієнт до серверної частини, взаємодіючи з базами даних виключно через REST API. Основні дані – ресурси, місії, логістичні запити, користувачі – зберігаються на сервері в реляційній (MySQL) та документній (MongoDB) базах даних, що дозволяє забезпечити централізовану цілісність і масштабованість.

Однак з огляду на важливість безперервного доступу до логістичних функцій навіть при відсутності інтернету, у майбутніх версіях мобільного застосунку планується впровадження власної локальної бази даних, побудованої за аналогічною структурою до серверної. Це дозволить зберігати критичні дані (наприклад, список запитів, місії, ресурси підрозділу) на пристрої користувача з подальшою синхронізацією після відновлення з'єднання.

До основних вимог до локальної БД належать:

- Швидкий доступ до кешованих даних у польових умовах.
- Безпечне зберігання (із можливим шифруванням чутливих полів).
- Сумісність структури з моделями, що використовуються на сервері.
- Контрольований механізм синхронізації для уникнення конфліктів.

Як потенційне рішення розглядається використання Room (SQLite) як основного механізму зберігання, що добре інтегрується з Android Jetpack-компонентами.

Таким чином, розвиток бази даних у мобільному застосунку орієнтований на забезпечення офлайн-доступності, швидкодії та захищеного зберігання, що критично важливо в умовах бойових дій або нестабільного зв'язку.