МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Розробка Android-орієнтованого додатку для реалізації мікро сервісів обслуговування замовлень»**

Виконав: студент 4 курсу групи Іт-41
Спеціальності 126 «Інформаційні системи та технології»

Білик Богдан Олегович

Керівник: Боярчук Олег Віталійович

Рецензент:

ДУБЛЯНИ-2024

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій значно змінив підходи до ведення бізнесу, зокрема в сфері управління замовленнями. Сучасні підприємства потребують ефективних інструментів для обробки замовлень, що дозволяють автоматизувати процеси, зменшити затрати часу та підвищити рівень обслуговування клієнтів. Актуальність теми дослідження полягає в необхідності створення мобільного додатку, який забезпечить зручне управління замовленнями в реальному часі, що є важливим аспектом успішного ведення бізнесу.

Метою цієї роботи є розробка Android-орієнтованого додатку для управління замовленнями з використанням архітектури мікросервісів. Основним завданням є створення інтуїтивно зрозумілого інтерфейсу, який дозволить користувачам легко додавати, редагувати та видаляти замовлення, а також переглядати список існуючих замовлень.

Об'єктом дослідження є процес управління замовленнями в бізнес-середовищі. Предметом дослідження виступають методи та засоби розробки мобільних додатків для управління замовленнями на платформі Android.

Методи дослідження включають аналіз існуючих рішень у сфері мобільних додатків для управління замовленнями, проектування архітектури додатку, програмування на Java та XML, тестування функціональності та продуктивності додатку.

Очікуваними результатами роботи є створення мобільного додатку, який буде відповідати вимогам сучасного бізнесу, забезпечить високу продуктивність та надійність, а також задовольнить потреби користувачів у зручному та ефективному інструменті для управління замовленнями.

Практичне значення роботи полягає у можливості використання розробленого додатку в реальних бізнес-процесах, що сприятиме автоматизації

управління замовленнями, покращенню обслуговування клієнтів та підвищенню ефективності діяльності підприємств.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Розробка мобільних додатків

Розробка мобільних додатків — це дія або процес, за допомогою якого розробляється мобільний додаток для одного або кількох мобільних пристроїв, які можуть включати персональні цифрові помічники (PDA), корпоративні цифрові помічники (EDA) або мобільні телефони.[1] Такі програмні додатки спеціально розроблені для роботи на мобільних пристроях з урахуванням численних апаратних обмежень. Загальні обмеження включають архітектуру та швидкість центрального процесора, доступну пам’ять (RAM), обмежену ємність для зберігання даних і значні варіації дисплеїв (технології, розміри, розміри, роздільна здатність) і методів введення (кнопки, клавіатури, сенсорні екрани зі стилусами чи без них).[2] Ці програми (або «програми») можна попередньо інсталювати на телефонах під час виробництва або постачати як веб-програми, використовуючи обробку на стороні сервера або клієнта (наприклад, JavaScript), щоб забезпечити «подібний до програми» досвід у межах веб-браузер.[3]

Розробка мобільних додатків стабільно зростає з точки зору доходів і створених робочих місць. Згідно зі звітом аналітиків за 2013 рік, у ЄС є 529 000 прямих робочих місць у сфері економіки додатків, 28 членів якого (включаючи Великобританію), 60 відсотків із яких є розробниками мобільних додатків[4].

Щоб полегшити розробку додатків для мобільних пристроїв і їх узгодженість, були використані різні підходи.

Більшість компаній, які постачають продукт (наприклад, Apple, iPod/iPhone/iPad), надають офіційний комплект розробки програмного забезпечення (SDK). Вони також можуть вибрати певну форму тестування та/або гарантії якості (QA). В обмін на надання SDK або інших інструментів

потенційному розробнику може знадобитися підписати певну форму угоди про нерозголошення, або NDA, яка обмежує передачу конфіденційної інформації.

Як частина процесу розробки, проектування інтерфейсу мобільного користувача (UI) є важливим кроком у створенні мобільних програм. Дизайнери мобільного інтерфейсу користувача розглядають обмеження, контексти, простір на екрані, методи введення та мобільність як контури дизайну. Обмеження в дизайні мобільного інтерфейсу користувача, які включають обмежену концентрацію уваги користувача та такі фактори форми, як розмір екрана мобільного пристрою для руки користувача. Контекст інтерфейсу мобільного інтерфейсу включає в себе сигнальні підказки від активності користувача, такі як місцезнаходження або час, коли пристрій використовується, які можна спостерігати за взаємодією користувача в мобільній програмі. Такі контекстні підказки можна використовувати для надання автоматичних пропозицій під час планування зустрічі чи діяльності або для фільтрації списку різноманітних послуг для користувача.

Користувач часто є центром взаємодії зі своїм пристроєм, а інтерфейс включає в себе компоненти апаратного та програмного забезпечення. Введення користувача дозволяє користувачам маніпулювати системою, а вихід пристрою дозволяє системі вказувати наслідки маніпулювання користувачами.

Загалом, метою дизайну мобільного інтерфейсу користувача є зрозумілий, зручний інтерфейс. Функціональність підтримується платформами мобільних корпоративних додатків або інтегрованими середовищами розробки (IDE).

Розробники мобільних додатків також повинні розглядати велику кількість пристроїв із різними розмірами екранів, специфікаціями обладнання та конфігураціями через інтенсивну конкуренцію в мобільному обладнанні та зміни всередині кожної з платформ.

Сьогодні мобільні додатки зазвичай розповсюджуються через офіційний онлайн-магазин або ринок (наприклад, Apple — App Store, Google — Google

Play), і існує формалізований процес, за яким розробники подають свої додатки на затвердження та включення до цих ринків. Однак історично це було не завжди так.

Мобільні інтерфейси користувача або інтерфейси покладаються на мобільні сервери для підтримки доступу до корпоративних систем. Мобільний бек-енд полегшує маршрутизацію даних, безпеку, автентифікацію, авторизацію, роботу в автономному режимі та координацію послуг. Ця функція підтримується набором компонентів проміжного програмного забезпечення, включаючи сервери мобільних додатків, мобільний сервер як послугу (MBaaS) та інфраструктуру сервісно-орієнтованої архітектури (SOA).

1.2 CRM

Управління взаємовідносинами з клієнтами (CRM) — це процес, у якому компанія або інша організація керує своєю взаємодією з клієнтами, зазвичай використовуючи аналіз даних для вивчення великих обсягів інформації.[1]

Системи CRM збирають дані з низки різних каналів зв'язку, включаючи веб-сайт компанії, телефон (багато програмне забезпечення постачається разом із програмним телефоном), електронну пошту, живий чат, маркетингові матеріали та останнім часом соціальні медіа.[2] Вони дозволяють компаніям дізнатися більше про свою цільову аудиторію та про те, як краще задовольнити їхні потреби, таким чином утримуючи клієнтів і стимулюючи зростання продажів.[3] CRM можна використовувати з минулими, теперішніми або потенційними клієнтами. Концепції, процедури та правила, яких дотримується корпорація під час спілкування зі своїми споживачами, називаються CRM. Цей повний зв'язок охоплює прямі контакти з клієнтами, такі як продажі та операції, пов'язані з обслуговуванням, прогнозування та аналіз моделей і поведінки споживачів з точки зору компанії.[4] За даними Gartner, розмір світового ринку CRM оцінюється в 69 мільярдів доларів у 2020 році.[5][6]

Концепція управління взаємовідносинами з клієнтами виникла на початку 1970-х років, коли задоволеність споживачів оцінювалася за допомогою щорічних опитувань або опитування на першій лінії.[7] У той час компаніям доводилося покладатися на автономні мейнфрейм-системи для автоматизації продажів, але рівень технологій дозволяв їм класифікувати клієнтів за категоріями в електронних таблицях і списках. Одним із найвідоміших попередників сучасної CRM є файл Farley. Розроблений керівником передвиборчої кампанії Франкліна Рузвельта Джеймсом Фарлі, Файл Фарлі являв собою повний набір записів із детальним описом політичних і особистих фактів про людей, з якими ФДР і Фарлі зустрічалися або мали зустрітися. Використовуючи його, люди, з якими зустрічався ФДР, були вражені його

«пригадуванням» фактів про їхню родину та про те, чим вони займалися професійно та політично.[8] У 1982 році Кейт і Роберт Д. Кестенбаум представили концепцію маркетингу баз даних, а саме застосування статистичних методів для аналізу та збору даних про клієнтів. До 1986 року Пет Салліван і Майк Мані випустили систему оцінки клієнтів під назвою ACT! заснований на принципі цифрового Rolodex, який вперше запропонував послугу управління контактами.

Цю тенденцію наслідували численні компанії та незалежні розробники, які намагалися максимізувати потенційний потенціал, у тому числі Том Сібель із Siebel Systems, який розробив перший продукт CRM Siebel Customer Relationship Management у 1993 році.[9] Для того, щоб конкурувати з цими новими автономними рішеннями CRM, які швидко розвиваються, створили такі компанії, які розробляють програмне забезпечення для планування ресурсів підприємства (ERP), як Oracle, Zoho Corporation [10], [10] SAP, [11] Peoplesoft (дочірня компанія Oracle з 2005 року) [12].] і Navision[13] почали розширювати свої можливості продажу, дистрибуції та обслуговування клієнтів за допомогою вбудованих модулів CRM. Це включало вбудовування автоматизації відділу продажів або розширеного обслуговування клієнтів (наприклад, запити, керування діяльністю) як функції CRM у їхній ERP.

Управління взаємовідносинами з клієнтами було популяризовано в 1997 році завдяки роботі Siebel, Gartner і IBM. Між 1997 і 2000 роками провідні продукти CRM були збагачені можливостями доставки та маркетингу.[14] У 1999 році компанія Siebel представила першу мобільну програму CRM під назвою Siebel Sales Handheld. Ідею автономної клієнтської бази, розміщеної в хмарі, незабаром прийняли інші провідні постачальники того часу, зокрема PeopleSoft (придбана Oracle),[12] Oracle , SAP і Salesforce.com.[15]

Перша система CRM з відкритим вихідним кодом була розроблена компанією SugarCRM в 2004 році. У цей період CRM стрімко мігрувала в хмару,

в результаті чого стала доступною для індивідуальних підприємців і невеликих команд. Це збільшення доступності викликало величезну хвилю зниження цін.[14] Приблизно в 2009 році розробники почали розглядати варіанти отримання прибутку від розвитку соціальних мереж і розробили інструменти, які допоможуть компаніям стати доступними в улюблених мережах усіх користувачів. Багато стартапів того часу виграли від цієї тенденції, щоб надавати виключно соціальні CRM-рішення, включаючи Base і Nutshell.[14] Того ж року компанія Gartner організувала та провела перший саміт з управління взаємовідносинами з клієнтами, на якому підсумувала функції, які системи повинні пропонувати, щоб класифікувати як рішення CRM.[16] У 2013 і 2014 роках більшість популярних продуктів CRM були пов'язані з системами бізнес-аналітики та комунікаційним програмним забезпеченням для покращення корпоративного спілкування та досвіду кінцевих користувачів. Провідною тенденцією є заміна стандартизованих CRM-рішень галузевими або зробити їх достатньо адаптованими для задоволення потреб кожного бізнесу.[17] У листопаді 2016 року Forrester опублікував звіт, у якому «визначив дев'ять найбільш значущих пакетів CRM від восьми відомих постачальників».[18]

Стратегічний

Стратегічний CRM зосереджується на розвитку бізнес-культури, орієнтованої на клієнта.[19]

Орієнтація бізнесу на клієнтоорієнтованість (у розробці та впровадженні стратегії CRM) призведе до вдосконалення CLV.[20]

Оперативний

Основне завдання CRM-систем – інтеграція та автоматизація продажів, маркетингу та підтримки клієнтів. Тому ці системи зазвичай мають інформаційну панель, яка дає загальне уявлення про три функції в одному поданні клієнта, окрему сторінку для кожного клієнта, який може мати компанія. Інформаційна панель може надавати інформацію про клієнта, минулі продажі, попередні

маркетингові зусилля тощо, узагальнюючи всі стосунки між клієнтом і фірмою. Операційна CRM складається з 3 основних компонентів: автоматизації відділу продажів, автоматизації маркетингу та автоматизації обслуговування.[21]

Автоматизація відділу продажів працює на всіх етапах циклу продажів, від початкового введення контактної інформації до перетворення потенційного клієнта на реального клієнта.[22] Він реалізує аналіз стимулювання збуту, автоматизує відстеження історії облікового запису клієнта для повторних продажів або майбутніх продажів і координує продажі, маркетинг, кол-центри та роздрібні точки. Це запобігає дублюванню зусиль між продавцем і клієнтом, а також автоматично відстежує всі контакти та подальші дії між обома сторонами.[22][23]

Автоматизація маркетингу спрямована на спрощення загального маркетингового процесу, щоб зробити його ефективнішим і ефективнішим. Інструменти CRM із можливостями автоматизації маркетингу можуть автоматизувати повторювані завдання, наприклад розсилку клієнтам автоматизованих маркетингових електронних листів у певний час або публікацію маркетингової інформації в соціальних мережах. Мета автоматизації маркетингу — перетворити потенційного клієнта на повноцінного клієнта. CRM-системи сьогодні також працюють над залученням клієнтів через соціальні мережі.[24]

Автоматизація обслуговування — це частина CRM-системи, яка фокусується на технології прямого обслуговування клієнтів. Завдяки автоматизації послуг клієнти отримують підтримку через кілька каналів, таких як телефон, електронна пошта, бази знань, портали продажу квитків, поширені запитання тощо.[21]

Аналітичний

Роль аналітичних систем CRM полягає в аналізі даних про клієнтів, зібраних із багатьох джерел, і представленні їх, щоб бізнес-менеджери могли приймати більш обґрунтовані рішення.[25] Аналітичні системи CRM

використовують такі методи, як аналіз даних, кореляція та розпізнавання шаблонів для аналізу даних клієнтів. Ця аналітика допомагає покращити обслуговування клієнтів, знаходячи невеликі проблеми, які можна вирішити, можливо, шляхом маркетингу для різних частин споживчої аудиторії по-різному.[21] Наприклад, за допомогою аналізу купівельної поведінки клієнтської бази компанія може побачити, що ця клієнтська база останнім часом не купувала багато продуктів. Після сканування цих даних компанія може подумати про те, щоб рекламувати цю підмножину споживачів по-іншому, щоб найкраще повідомити, як саме продукти цієї компанії можуть принести користь цій групі.

Спільний

Третя головна мета систем CRM – об’єднати зовнішніх зацікавлених сторін, таких як постачальники, продавці та дистриб’ютори, і обмінюватися інформацією про клієнтів між групами/відділами та організаціями. Наприклад, можна зібрати відгуки про дзвінки в службу технічної підтримки, які можуть допомогти надати напрямок маркетингу продуктів і послуг цьому конкретному клієнту в майбутньому.[26]

РОЗДІЛ 2
ПОСТАНОВКА ЗАДАЧІ

2.1 Вибір мови програмування

Java є однією з найпопулярніших мов програмування у світі, яка має багату історію та величезну кількість практичних застосувань. Її універсальність та потужність роблять її вибором номер один для багатьох розробників та компаній по всьому світу.

Java була розроблена в 1995 році компанією Sun Microsystems (нині належить Oracle) з метою створення мови програмування, яка була б незалежною від платформи та могла б використовуватись на різноманітних пристроях. Однією з ключових особливостей Java є її здатність працювати на будь-якій платформі, що має Java Virtual Machine (JVM). Це досягається завдяки тому, що Java-код компілюється не у машинний код, як це відбувається в мовах типу C або C++, а в байт-код, який виконується JVM. Це забезпечує надзвичайну портативність Java-додатків, дозволяючи їм працювати на різних операційних системах без змін у коді.

Порівнюючи Java з іншими мовами програмування, такими як Python, C++ або C#, варто виділити кілька ключових переваг. По-перше, Java має добре розвинену екосистему бібліотек та інструментів, що робить її дуже потужною та універсальною мовою. Наприклад, Java має розвинені бібліотеки для роботи з базами даних, мережами, обробки XML, а також безліч фреймворків для веб-розробки, таких як Spring, Hibernate та багато інших. Це дозволяє швидко та ефективно розробляти складні системи, використовуючи готові рішення.

Друга важлива перевага Java полягає у її продуктивності та ефективності. Хоча інтерпретовані мови, такі як Python, часто хваляться за їхню простоту та швидкість розробки, вони зазвичай програють Java у питанні продуктивності виконання коду. Java поєднує в собі відносну простоту та високу продуктивність,

що дозволяє їй бути конкурентоспроможною у багатьох сферах, включаючи високонавантажені серверні додатки, системи обробки великих даних та реального часу.

Третя ключова перевага Java - це її масштабованість та надійність. Завдяки статичній типізації та строгій системі типів, Java забезпечує високу надійність та безпеку коду. Це знижує ймовірність помилок, пов'язаних з некоректною роботою з типами даних, що є важливим аспектом при розробці великих та складних програмних систем. Крім того, Java має вбудовану підтримку багатопоточності, що дозволяє створювати масштабовані та ефективні додатки, здатні обробляти велику кількість паралельних задач.

Якщо порівнювати Java з C++, можна відзначити, що Java забезпечує вищий рівень абстракції та безпеки, оскільки автоматично керує пам'яттю за допомогою механізму garbage collection. У той час як C++ дає розробникам більше контролю над ресурсами системи, це також підвищує ризик помилок, таких як витоки пам'яті та інші проблеми, пов'язані з ручним керуванням пам'яттю. Java, натомість, зменшує ці ризики, дозволяючи розробникам зосередитись на логіці додатку.

Порівняно з Python, Java має переваги у швидкодії та масштабованості. Хоча Python вважається однією з найбільш легких для вивчення мов, Java, завдяки своїй статичній типізації та компіляції у байт-код, забезпечує вищу швидкодію додатків. Це особливо важливо у проектах, що потребують високої продуктивності та ефективності використання ресурсів. Крім того, Java має розвинену систему управління пакетами та залежностями через Maven або Gradle, що значно спрощує управління великими проектами.

В порівнянні з C#, Java має більш універсальну платформу. Хоча C# активно використовується в екосистемі Microsoft та має багато схожих з Java рис, остання забезпечує більшу незалежність від платформи та ширший спектр

застосувань. Це робить Java більш привабливою для компаній, що прагнуть створювати крос-платформенні рішення.

Ще одна важлива перевага Java - це її активна спільнота та постійний розвиток. Величезна кількість розробників по всьому світу активно використовують Java, що забезпечує наявність великої кількості ресурсів, таких як документація, курси, форуми та бібліотеки. Крім того, Oracle та спільнота розробників постійно вдосконалюють мову, додаючи нові можливості та оптимізації, що дозволяє Java залишатися сучасною та конкурентоспроможною.

Окрім того, Java є однією з провідних мов програмування для мобільних додатків завдяки Android, операційній системі від Google. Android додатки пишуться на Java або Kotlin, що робить Java важливою мовою для мобільних розробників. Це додає ще один аспект універсальності Java, дозволяючи розробникам використовувати свої навички для створення додатків на різних платформах, включаючи десктоп, сервер та мобільні пристрої.

Таким чином, Java є потужною, універсальною та надійною мовою програмування, яка поєднує в собі продуктивність, масштабованість та крос-платформенність. Її багата екосистема, активна спільнота та постійний розвиток роблять її одним з найкращих виборів для розробників у всьому світі. Незалежно від того, чи йдеться про розробку корпоративних систем, мобільних додатків чи високонавантажених серверних рішень, Java забезпечує всі необхідні інструменти та можливості для успішної реалізації проектів будь-якої складності.

2.2 Вибір середовища розробки

Android Studio є офіційним інтегрованим середовищем розробки (IDE) для створення додатків під операційну систему Android, розроблене компанією Google. Це потужне та багатофункціональне середовище, яке надає розробникам всі необхідні інструменти для створення, тестування, налагодження та розгортання Android-додатків.

Android Studio був анонсований у 2013 році та базується на популярному IDE IntelliJ IDEA від JetBrains. Відтоді він став основним інструментом для розробки Android-додатків, значно полегшуючи роботу розробників завдяки своїм широким можливостям та інтеграції з екосистемою Android.

Однією з ключових особливостей Android Studio є його потужний редактор коду. Цей редактор підтримує розширене автозаповнення коду, інтелектуальні підказки, швидку навігацію та інші функції, які значно спрощують процес написання коду. Редактор також підтримує кілька мов програмування, включаючи Java, Kotlin і C++, що робить його універсальним інструментом для розробників з різним рівнем знань та уподобань. Важливою перевагою є також підтримка автоматичного рефакторингу коду, що дозволяє легко вносити зміни в код та зберігати його якість.

Візуальний редактор макетів у Android Studio дозволяє розробникам створювати складні інтерфейси користувача шляхом перетягування компонентів. Це значно спрощує процес дизайну та дозволяє швидко створювати прототипи додатків. Редактор макетів також підтримує попередній перегляд інтерфейсу на різних пристроях та з різними версіями Android, що дозволяє забезпечити сумісність додатка на широкому спектрі пристроїв. Це особливо важливо для Android, оскільки існує безліч пристроїв з різними розмірами екранів, роздільною здатністю та версіями операційної системи.

Інтеграція з Gradle, потужною системою збірки, є ще однією важливою особливістю Android Studio. Gradle автоматизує процес компіляції та управління

залежностями, дозволяючи легко налаштовувати проекти та створювати різні версії додатків для різних конфігурацій пристроїв. Це значно спрощує процес розробки та розгортання додатків, дозволяючи розробникам швидко реагувати на зміни вимог та оновлювати свої проекти.

Android Studio також пропонує інтеграцію з системами контролю версій, такими як Git. Це дозволяє розробникам ефективно керувати версіями коду, працювати в команді та відслідковувати зміни в проектах. Інтеграція з Git забезпечує зручні інструменти для злиття змін, вирішення конфліктів та відкату до попередніх версій, що значно підвищує продуктивність та надійність роботи.

Однією з найбільш важливих особливостей Android Studio є його потужні інструменти для тестування додатків. Інтегрований емулятор Android дозволяє розробникам тестувати додатки на віртуальних пристроях з різними конфігураціями, що значно скорочує час на тестування та забезпечує високу якість додатків. Крім того, Android Studio підтримує інструменти для автоматизованого тестування, такі як Espresso та JUnit, що дозволяє розробникам створювати комплексні тестові сценарії та забезпечувати високу якість коду.

Порівнюючи Android Studio з іншими IDE, такими як Eclipse або Visual Studio, можна відзначити кілька ключових переваг. По-перше, Android Studio спеціально розроблене для створення Android-додатків, що забезпечує тісну інтеграцію з Android екосистемою та оптимізацію процесу розробки. По-друге, Android Studio пропонує більш сучасний та інтуїтивно зрозумілий інтерфейс, що полегшує навчання та використання. Крім того, Android Studio регулярно оновлюється компанією Google, що гарантує підтримку останніх технологій та API, а також забезпечує високий рівень безпеки та надійності.

На відміну від Eclipse, яке вимагає встановлення численних плагінів для роботи з Android, Android Studio є готовим рішенням, що включає всі необхідні інструменти для розробки Android-додатків. Це значно спрощує процес

налаштування та використання, дозволяючи розробникам зосередитися на написанні коду, а не на конфігурації середовища розробки.

Visual Studio, хоча і є потужним інструментом для розробки, більше орієнтоване на екосистему Microsoft та створення додатків для Windows. Android Studio ж спеціалізується на Android, що робить його кращим вибором для розробників, які працюють з цією платформою. Крім того, Android Studio забезпечує вищу інтеграцію з інструментами та сервісами Google, такими як Firebase, що надає додаткові можливості для розробки, аналітики та монетизації додатків.

Ще однією важливою перевагою Android Studio є його активна спільнота та постійний розвиток. Величезна кількість розробників по всьому світу активно використовують Android Studio, що забезпечує наявність великої кількості ресурсів, таких як документація, курси, форуми та бібліотеки. Крім того, Google та спільнота розробників постійно вдосконалюють IDE, додаючи нові можливості та оптимізації, що дозволяє Android Studio залишатися сучасним та конкурентоспроможним.

Завдяки своїм розширеним можливостям, Android Studio дозволяє розробникам створювати додатки будь-якої складності. Незалежно від того, чи йдеться про простий додаток для запису нотаток або складну систему для електронної комерції, Android Studio надає всі необхідні інструменти для успішної реалізації проекту. Інтеграція з інструментами для аналізу продуктивності, такими як Android Profiler, дозволяє оптимізувати додатки та забезпечити їх стабільну роботу на різних пристроях.

Крім того, Android Studio підтримує розробку для різних типів пристроїв, включаючи смартфони, планшети, телевізори, автомобілі та носимі пристрої. Це дозволяє розробникам створювати додатки для широкого спектру пристроїв, використовуючи єдине середовище розробки. Інтеграція з Android Things також

дозволяє створювати додатки для Інтернету речей (IoT), що відкриває нові можливості для інновацій та розвитку.

Важливою складовою успішної розробки є підтримка різних форматів розгортання додатків. Android Studio дозволяє легко створювати APK-файли для розгортання додатків на пристроях, а також підтримує створення AAB-файлів для оптимізації розміру додатків та їх розгортання через Google Play. Це забезпечує гнучкість та зручність у розгортанні додатків, дозволяючи розробникам ефективно доставляти свої продукти користувачам.

Android Studio також забезпечує підтримку різних мов програмування. Хоча Java залишається однією з основних мов для розробки Android-додатків, Kotlin стає все більш популярною завдяки своїй сучасній та зручній синтаксичній структурі. Android Studio повністю підтримує Kotlin, надаючи інструменти для написання, налагодження та тестування коду на цій мові. Це дозволяє розробникам вибирати ту мову, яка найкраще відповідає їхнім потребам та вподобанням.

Надійність та безпека є ще однією важливою перевагою Android Studio. Інструменти для аналізу коду та виявлення потенційних проблем дозволяють розробникам забезпечити високу якість та безпеку своїх додатків. Це включає виявлення можливих вразливостей, проблем з продуктивністю та інших помилок, що можуть вплинути на роботу додатка. Це особливо важливо у сучасному світі, де безпека додатків є пріоритетом для розробників та користувачів.

Постійні оновлення Android Studio гарантують, що розробники завжди мають доступ до найновіших інструментів та технологій. Google регулярно випускає оновлення, що включають нові функції, виправлення помилок та оптимізації. Це дозволяє розробникам завжди бути в курсі останніх змін в екосистемі Android та використовувати найновіші можливості для створення своїх додатків.

Загалом, Android Studio є незамінним інструментом для розробників Android-додатків. Завдяки своїм розширеним можливостям, інтеграції з екосистемою Android та підтримці сучасних технологій, Android Studio забезпечує високу продуктивність, надійність та зручність використання. Незалежно від рівня досвіду чи складності проекту, Android Studio надає всі необхідні інструменти для успішної реалізації будь-якої ідеї.

2.3 Вибір СКБД

SQLite є популярною вбудованою системою управління базами даних (СУБД), яка використовується в різних додатках та пристроях. Відома своєю легкістю, продуктивністю та повною відповідністю стандартам SQL, SQLite забезпечує надійне зберігання даних без необхідності встановлення окремого серверного програмного забезпечення. Її головними перевагами є простота використання, відсутність необхідності в налаштуванні та висока продуктивність при роботі з невеликими та середніми об'ємами даних.

SQLite була розроблена у 2000 році Річардом Гіпом з метою створення легкої та самодостатньої СУБД, яка могла б використовуватись вбудовано в програми. Відтоді вона стала однією з найпоширеніших баз даних у світі, використовуючись у мільйонах додатків, включаючи веб-браузери, мобільні додатки, операційні системи та вбудовані системи.

Однією з ключових особливостей SQLite є її вбудованість. На відміну від традиційних СУБД, таких як MySQL або PostgreSQL, які потребують окремого серверного середовища для роботи, SQLite є бібліотекою, яка компілюється та вбудовується безпосередньо у додаток. Це значно спрощує розгортання та використання бази даних, оскільки немає потреби в адмініструванні сервера, налаштуванні мережевих з'єднань та інших супутніх задачах.

Ще однією важливою перевагою SQLite є її портативність. Дані зберігаються у звичайному файлі на диску, що дозволяє легко переносити базу даних між різними системами та пристроями. Це робить SQLite ідеальним вибором для мобільних додатків, де необхідно забезпечити локальне зберігання даних без підключення до мережі. Крім того, файли баз даних SQLite можуть бути легко збережені, передані та інтегровані у резервні копії, що забезпечує додаткову гнучкість та надійність.

Продуктивність SQLite також є одним з її сильних аспектів. Завдяки оптимізації для роботи з файлами та ефективному використанню ресурсів

системи, SQLite забезпечує високу швидкість виконання запитів навіть на пристроях з обмеженими ресурсами. Хоча вона не призначена для роботи з великими об'ємами даних або високонавантажених додатків, її продуктивність є більш ніж достатньою для більшості типових сценаріїв використання, включаючи зберігання налаштувань додатків, історії користувачів, тимчасових даних та інше.

Серед інших переваг SQLite можна виділити її відповідність стандартам SQL. Вона підтримує більшість стандартних SQL команд та синтаксису, що дозволяє розробникам використовувати знайомі інструменти та підходи для роботи з базою даних. Це також полегшує міграцію даних та кодів між SQLite та іншими СУБД, якщо виникає така необхідність. Завдяки цьому SQLite є чудовим інструментом для розробки, тестування та прототипування додатків, які згодом можуть бути перенесені на більш потужні серверні СУБД.

Порівнюючи SQLite з іншими СУБД, такими як MySQL, PostgreSQL або Oracle, можна виділити кілька ключових переваг. По-перше, SQLite не потребує налаштування сервера та адміністрування, що значно спрощує розгортання та знижує витрати на підтримку. По-друге, вона забезпечує високу продуктивність та ефективність роботи з даними завдяки оптимізації для вбудованих додатків та роботи з файлами. По-третє, її портативність та можливість зберігання даних у звичайному файлі на диску робить її ідеальним вибором для мобільних додатків та вбудованих систем.

Однак варто зазначити, що SQLite має свої обмеження. Вона не призначена для роботи з великими об'ємами даних або високонавантажених додатків, де необхідна підтримка одночасного доступу великої кількості користувачів та складних транзакцій. У таких випадках доцільно використовувати більш потужні серверні СУБД, такі як PostgreSQL або MySQL, які забезпечують кращу масштабованість та продуктивність у таких сценаріях.

SQLite також має потужні можливості для роботи з транзакціями, що забезпечує надійність та цілісність даних. Вона підтримує ACID (Atomicity, Consistency, Isolation, Durability) властивості, що гарантує коректне виконання транзакцій навіть у випадку збоїв або аварійних завершень роботи системи. Це забезпечує високу надійність та стабільність роботи додатків, які використовують SQLite для зберігання важливих даних.

Ще однією важливою особливістю SQLite є її розширюваність. Вона підтримує створення користувацьких функцій, агрегацій та тригерів, що дозволяє розробникам розширювати функціональність бази даних відповідно до своїх потреб. Це забезпечує додаткову гнучкість та можливість адаптації SQLite до специфічних вимог додатка.

Однією з цікавих можливостей SQLite є підтримка віртуальних таблиць через інтерфейс SQL/MED (Management of External Data). Це дозволяє розробникам створювати віртуальні таблиці, які взаємодіють з зовнішніми джерелами даних, такими як файли, веб-сервіси або інші бази даних. Це відкриває нові можливості для інтеграції та обробки даних з різних джерел у межах одного додатка.

SQLite також широко використовується у вбудованих системах та Інтернеті речей (IoT) завдяки своїй легкості та ефективності. Вона забезпечує надійне зберігання даних на пристроях з обмеженими ресурсами, таких як сенсори, контролери та інші вбудовані пристрої. Це робить SQLite ідеальним вибором для розробки додатків для IoT, де необхідно забезпечити ефективне зберігання та обробку даних у реальному часі.

Постійний розвиток та підтримка з боку спільноти розробників є ще однією важливою перевагою SQLite. Вона має велику та активну спільноту, яка постійно працює над вдосконаленням бази даних, виправленням помилок та додаванням нових можливостей. Це забезпечує високу якість та надійність продукту, а також

доступ до великої кількості ресурсів, таких як документація, приклади коду та інструменти для розробки.

Загалом, SQLite є потужною та гнучкою системою управління базами даних, яка забезпечує високу продуктивність, надійність та зручність використання. Її легкість, портативність та відповідність стандартам SQL роблять її ідеальним вибором для широкого спектру додатків, включаючи мобільні додатки, вбудовані системи, веб-додатки та багато інших. Незалежно від рівня складності чи обсягу даних, SQLite надає всі необхідні інструменти для ефективного зберігання та обробки даних, забезпечуючи високу якість та надійність роботи додатків.

2.4 Додаткові інструменти

XML (eXtensible Markup Language) є ключовою технологією для створення користувацьких інтерфейсів у розробці додатків під Android. Використовуючи XML для верстки, розробники можуть створювати структуровані, зручні та легко підтримувані інтерфейси, що відповідають сучасним стандартам дизайну та функціональності.

XML став основним інструментом для опису структури та зовнішнього вигляду інтерфейсів користувача в Android завдяки своїй простоті, гнучкості та широкій підтримці. Однією з головних переваг XML є його здатність чітко визначати ієрархію елементів, що робить процес створення складних інтерфейсів прозорим та зрозумілим.

Використання XML для верстки в Android дозволяє розділити логіку додатка та його зовнішній вигляд. Це відповідає принципам Model-View-Controller (MVC) архітектури, де XML-файли виступають у ролі View, а код на Java або Kotlin реалізує Model та Controller. Такий підхід значно спрощує розробку, тестування та підтримку додатків, оскільки зміни в інтерфейсі не впливають на логіку програми і навпаки.

Однією з ключових особливостей XML у Android є його інтеграція з Android Studio, офіційним середовищем розробки для Android. Android Studio пропонує потужний візуальний редактор макетів, який дозволяє розробникам створювати інтерфейси шляхом перетягування елементів на екран. Це значно спрощує процес дизайну, дозволяючи швидко створювати прототипи та візуалізувати кінцевий результат без необхідності написання великої кількості коду.

XML-файли в Android зазвичай зберігаються в каталозі res/layout і мають розширення .xml. Кожен XML-файл визначає структуру одного макета інтерфейсу, включаючи такі елементи, як кнопки, текстові поля, зображення та інші компоненти. Ці елементи можуть бути налаштовані за допомогою атрибутів,

що визначають їхні властивості, такі як розмір, колір, положення на екрані та інші параметри. Це дозволяє створювати багаті та інтерактивні інтерфейси, які можуть адаптуватися до різних розмірів екранів та орієнтацій пристроїв.

Однією з важливих переваг використання XML для верстки в Android є підтримка адаптивного дизайну. Використовуючи такі механізми, як ConstraintLayout, розробники можуть створювати гнучкі макети, які автоматично підлаштовуються під різні розміри та пропорції екранів. Це забезпечує високу якість користувацького досвіду на різних пристроях, від смартфонів до планшетів та інших гаджетів.

ConstraintLayout є потужним інструментом для створення адаптивних макетів. Він дозволяє розробникам визначати відносні позиції елементів інтерфейсу один відносно одного або відносно меж контейнера, що забезпечує високу гнучкість та точність у розміщенні елементів. Це значно спрощує створення складних макетів, зменшуючи необхідність у вкладених макетах, що покращує продуктивність додатка.

Крім того, XML дозволяє легко створювати та використовувати стилі та теми, що сприяє забезпеченню консистентності дизайну додатка. Стилі в XML визначають спільні властивості для групи елементів, що дозволяє централізовано керувати зовнішнім виглядом додатка. Теми, у свою чергу, дозволяють визначати загальний вигляд додатка, включаючи кольорові схеми, шрифти та інші візуальні параметри. Це забезпечує єдиний стиль для всього додатка та спрощує підтримку та оновлення дизайну.

Ще однією важливою перевагою XML у розробці Android-додатків є підтримка ресурсів. Ресурси, такі як рядки, зображення, кольори та інші дані, зберігаються у XML-файлах, що дозволяє легко керувати ними та локалізувати додаток для різних мов та регіонів. Це робить додаток більш доступним для широкої аудиторії та підвищує його конкурентоспроможність на глобальному ринку.

Порівнюючи XML з іншими технологіями для створення інтерфейсів користувача, такими як HTML/CSS або XAML, можна виділити кілька ключових переваг. По-перше, XML у Android є нативною технологією, що забезпечує тісну інтеграцію з платформою та оптимізацію продуктивності. По-друге, XML пропонує високу гнучкість та розширюваність, дозволяючи створювати складні та адаптивні інтерфейси з мінімальними зусиллями. По-третє, використання XML спрощує розділення логіки та дизайну, що підвищує зручність розробки та підтримки додатків.

Однак варто зазначити, що XML у Android має свої обмеження. Створення дуже складних макетів може бути викликом, і у таких випадках розробники можуть стикатися з проблемами продуктивності або необхідністю оптимізації коду. Проте, завдяки постійному розвитку інструментів та технологій, таких як ConstraintLayout, більшість цих проблем можуть бути ефективно вирішені.

Ще однією важливою особливістю XML у Android є підтримка анімацій та трансформацій. Використовуючи XML, розробники можуть створювати плавні та привабливі анімації для покращення користувацького досвіду. Це включає анімацію змін розміру, положення, прозорості та інших властивостей елементів інтерфейсу. Анімації можуть бути визначені у XML-файлах та застосовані до елементів у відповідь на взаємодії користувача або події додатка.

Завдяки своїй простоті, гнучкості та потужності, XML залишається основним інструментом для створення користувацьких інтерфейсів у розробці Android-додатків. Використання XML дозволяє розробникам створювати якісні, адаптивні та легко підтримувані інтерфейси, які відповідають сучасним стандартам дизайну та забезпечують відмінний користувацький досвід на різних пристроях. Незалежно від рівня складності проекту, XML надає всі необхідні інструменти для успішної реалізації будь-якої ідеї.

2.5 Постановка задачі

Метою цієї роботи є розробка Android-орієнтованого додатку для управління замовленнями з використанням архітектури мікросервісів. У сучасному світі бізнес-процеси стають все більш автоматизованими, і управління замовленнями не є винятком. Важливим аспектом успішного ведення бізнесу є ефективне управління замовленнями, яке дозволяє контролювати весь процес від прийняття замовлення до його виконання. У зв'язку з цим постає необхідність у розробці зручного та функціонального інструменту для обслуговування замовлень, який би дозволяв керувати замовленнями в реальному часі з будь-якого місця.

Завдання полягає у створенні додатку, який забезпечить можливість додавання, редагування та видалення замовлень, а також відображення списку існуючих замовлень. Додаток має бути розроблений на основі Java та XML для Android-платформи і включати такі функції, як введення даних через діалогові вікна, інтеграція з серверною частиною для збереження даних та використання сучасних підходів до верстки з використанням ConstraintLayout.

Основна мета полягає в тому, щоб створити інтуїтивно зрозумілий інтерфейс користувача, який дозволить швидко і зручно обробляти замовлення, забезпечуючи високу продуктивність та надійність додатку. Це рішення повинно стати ефективним інструментом для бізнесу, що дозволить підвищити ефективність управління замовленнями та покращити якість обслуговування клієнтів.

РОЗДІЛ 3
ПРОЕКТУВАННЯ І РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Архітектура системи

Монолітна архітектура є традиційним підходом до розробки програмного забезпечення, який передбачає створення єдиного, нероздільного додатка. В такому додатку всі компоненти, включаючи інтерфейси користувача, бізнес-логіку та доступ до даних, інтегровані в одну цілісну систему. Цей підхід має свої переваги та недоліки і залишається популярним у багатьох проектах завдяки своїй простоті та ефективності.

Однією з головних переваг монолітної архітектури є її простота. Розробники можуть працювати з єдиною кодовою базою, що спрощує процеси розробки, тестування та розгортання. Усі компоненти додатка інтегровані та взаємодіють один з одним без необхідності налаштування складних міжпроцесних взаємодій або мережевих протоколів. Це забезпечує швидший розвиток та легшу підтримку, особливо для невеликих команд або проектів з обмеженими ресурсами.

Монолітна архітектура також забезпечує високу продуктивність завдяки тісній інтеграції всіх компонентів у межах одного процесу. Відсутність необхідності комунікації через мережу або інші механізми між різними частинами системи дозволяє досягти низької затримки та швидкого виконання операцій. Це особливо важливо для додатків, які вимагають високої швидкодії та обробки великих обсягів даних у реальному часі.

Ще однією перевагою монолітної архітектури є зручність тестування та відлагодження. Завдяки єдиній кодовій базі, розробники можуть легко запускати тести та відлагоджувати додаток у локальному середовищі. Це значно спрощує процес виявлення та виправлення помилок, а також забезпечує високу якість

коду. Крім того, інтеграційне тестування стає простішим, оскільки всі компоненти системи знаходяться в одному середовищі.

Однак монолітна архітектура має свої недоліки, особливо коли йдеться про великі та складні системи. Одним з основних викликів є зростання складності коду з часом. У міру розширення функціональності додатка кодова база стає все більш складною та важкою для розуміння, що може призвести до проблем з підтримкою та розвитком. Це може ускладнити внесення змін, оскільки кожна зміна може вплинути на велику кількість взаємопов'язаних компонентів.

Ще одним важливим недоліком є обмежена масштабованість монолітних додатків. Оскільки всі компоненти системи інтегровані в один процес, важко масштабувати окремі частини додатка незалежно одна від одної. Наприклад, якщо один компонент потребує більшої обчислювальної потужності або ресурсів, ніж інші, неможливо виділити йому додаткові ресурси без масштабування всього додатка. Це може призвести до неефективного використання ресурсів та зниження продуктивності системи в цілому.

Монолітна архітектура також ускладнює процес розгортання та оновлення додатків. Оскільки всі компоненти системи зібрані в одну єдину одиницю, будь-які зміни в одному компоненті вимагають повторного розгортання всього додатка. Це може призвести до простоїв та перерв у роботі системи, що особливо критично для додатків з високими вимогами до доступності та надійності. Крім того, процес розгортання може стати складним та ризикованим, оскільки зміни в одному компоненті можуть вплинути на роботу інших компонентів системи.

Ще одним важливим аспектом є залежність між компонентами у монолітній архітектурі. Оскільки всі компоненти інтегровані в одну систему, вони можуть мати тісні залежності один від одного. Це може ускладнити процес модернізації або заміни окремих компонентів, оскільки будь-які зміни в одному компоненті можуть вимагати внесення змін в інші частини системи. Це знижує

гнучкість та адаптивність додатка, що може бути проблематичним у динамічних середовищах, де вимоги та умови можуть швидко змінюватись.

Порівнюючи монолітну архітектуру з іншими підходами, такими як мікросервісна архітектура, можна виділити кілька ключових відмінностей. Мікросервісна архітектура пропонує більш гнучкий та масштабований підхід до розробки програмного забезпечення, розділяючи систему на незалежні сервіси, які можуть розвиватися, тестуватися та розгортатися окремо. Це дозволяє досягти вищої гнучкості, адаптивності та ефективності використання ресурсів. Однак, мікросервісна архітектура також вимагає більш складного управління та координації між сервісами, що може підвищити складність розробки та експлуатації системи.

Монолітна архітектура залишається актуальною у багатьох сценаріях, особливо для невеликих та середніх проектів, де простота та швидкість розробки мають пріоритет. Вона забезпечує високу продуктивність, зручність тестування та розгортання, що робить її оптимальним вибором для багатьох додатків. Проте, для великих та складних систем з високими вимогами до масштабованості та гнучкості, мікросервісна архітектура може бути більш підходящим варіантом.

У підсумку, монолітна архітектура є важливим підходом у розробці програмного забезпечення, який має свої переваги та недоліки. Вона забезпечує простоту, високу продуктивність та зручність використання, що робить її популярним вибором для багатьох проектів. Однак, у великих та складних системах, де важлива масштабованість та гнучкість, можуть виникати проблеми з підтримкою та розвитком, що вимагає розгляду альтернативних архітектурних підходів. Незалежно від вибору архітектури, важливо ретельно аналізувати вимоги та умови проекту, щоб забезпечити оптимальне рішення для конкретного випадку.

На рисунку 3.1 представлено схему архітектури системи.

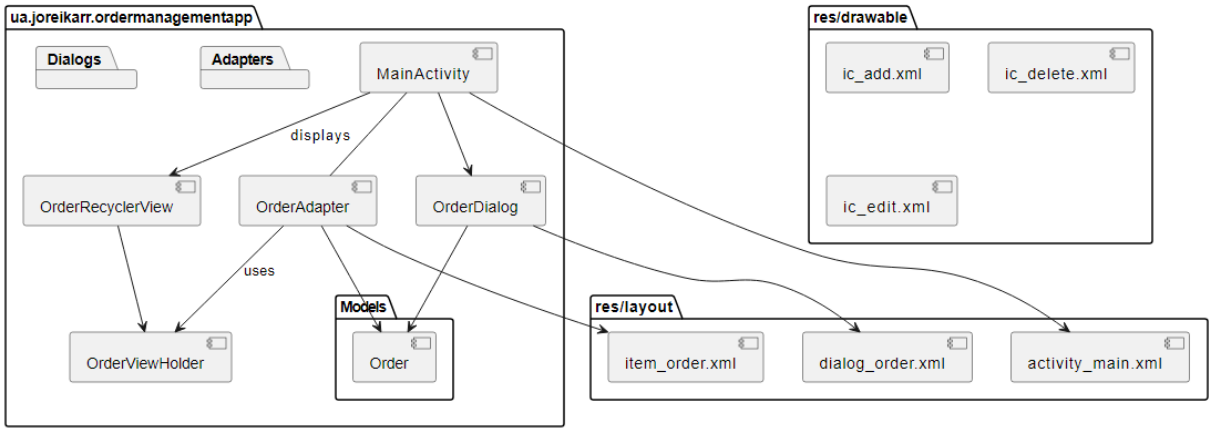

Рисунок 3.1 — Схема архітектури системи

3.2 Проектування функціоналу

Діаграми варіантів використання (use case diagrams) є одним з ключових інструментів в арсеналі системного аналітика та розробника програмного забезпечення. Вони надають наочне уявлення про функціональні можливості системи, дозволяють зрозуміти взаємодію користувачів із системою та сприяють ефективному визначенню вимог до проекту.

Основні компоненти діаграми варіантів використання

Актори (Actors):

Актор представляє зовнішню сутність, яка взаємодіє із системою. Це може бути як людина, так і інша система чи пристрій. Актори можуть бути як основними користувачами, так і допоміжними ролями.

Наприклад, в системі онлайн-банкінгу актором може бути "Клієнт", який виконує операції з рахунком, та "Адміністратор", який підтримує систему.

Варіанти використання (Use Cases):

Варіант використання описує послідовність дій, які система виконує для досягнення певної мети користувача. Кожен варіант використання відповідає на запитання: "Що користувач хоче зробити за допомогою системи?".

Наприклад, для системи онлайн-банкінгу варіантами використання можуть бути "Перегляд балансу", "Переклад коштів", "Оплата рахунків".

Зв'язки (Relationships):

Асоціація (Association): Це прямий зв'язок між актором і варіантом використання, який показує, що актор використовує даний варіант.

Включення (Include): Вказує, що один варіант використання завжди включає поведінку іншого варіанту. Це корисно для повторюваних дій, які використовуються у кількох варіантах.

Розширення (Extend): Показує, що поведінка варіанту використання може бути розширена додатковою поведінкою за певних умов.

Узагальнення (Generalization): Зв'язок між акторами або варіантами використання, який показує спадкування властивостей і поведінки.

Використання діаграм варіантів використання

Визначення вимог:

Діаграми варіантів використання є інструментом для визначення функціональних вимог до системи. Вони допомагають зібрати та структуризувати інформацію про те, що повинна робити система, відображаючи взаємодію між користувачами та системою.

Це особливо корисно на ранніх етапах проектування, коли необхідно отримати загальне уявлення про систему і її можливості.

Комунікація з зацікавленими сторонами:

Діаграми варіантів використання є зрозумілим інструментом для комунікації між розробниками та іншими зацікавленими сторонами, такими як замовники, кінцеві користувачі та менеджери проектів.

Вони забезпечують наочне представлення функціональних можливостей системи, що полегшує обговорення, уточнення та узгодження вимог.

Планування та розробка:

Діаграми варіантів використання допомагають розробникам планувати архітектуру та функціональність системи. Вони служать основою для створення детальних специфікацій, технічної документації та тестових сценаріїв.

Вони також допомагають визначити пріоритети розробки, розподілити завдання та оцінити необхідні ресурси.

Тестування та верифікація:

Варіанти використання можуть бути використані для створення тестових сценаріїв, що дозволяє перевірити, чи виконує система всі необхідні функції.

Вони допомагають виявити можливі помилки та недоліки на ранніх етапах розробки, забезпечуючи якість та відповідність системи вимогам.

Переваги діаграм варіантів використання

Наочність та зрозумілість:

Діаграми варіантів використання є простими для розуміння навіть для людей без технічної освіти. Вони забезпечують зрозуміле та наочне уявлення про функціональні можливості системи.

Гнучкість та адаптивність:

Діаграми варіантів використання можуть бути легко адаптовані до змін вимог та умов проекту. Вони дозволяють швидко вносити зміни та оновлення.

Підтримка модульного підходу:

Використання діаграм варіантів використання сприяє модульному підходу до розробки системи, що полегшує її масштабування та підтримку.

Діаграми варіантів використання є важливим інструментом для моделювання та визначення функціональних вимог до системи. Вони допомагають зрозуміти взаємодію користувачів із системою, спрощують комунікацію з зацікавленими сторонами, полегшують планування та розробку, а також сприяють ефективному тестуванню системи. Завдяки своїй наочності, гнучкості та підтримці модульного підходу, діаграми варіантів використання є незамінним інструментом в арсеналі системного аналітика та розробника.

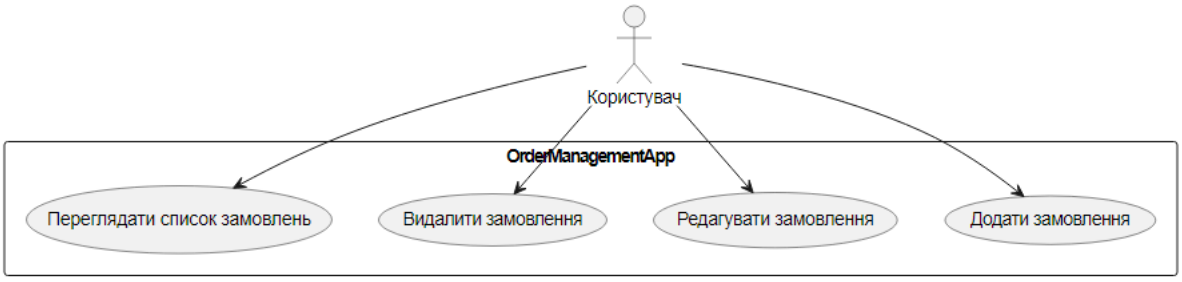

Рисунок 3.2 — Діаграма варіантів використання

Специфікація варіантів використання

1. Додати замовлення

Актор: Користувач

Опис: Користувач може додати нове замовлення до системи.

Передумови: Користувач має доступ до додатку і авторизований для додавання замовлень.

Основний потік:

1. Користувач натискає кнопку "Додати замовлення".
2. Відкривається форма для введення інформації про замовлення.
3. Користувач вводить дані замовлення (ID замовлення, ім'я клієнта, деталі замовлення, статус).
4. Користувач підтверджує додавання замовлення.
5. Система зберігає нове замовлення та відображає його в списку замовлень.

Альтернативні потоки:

- 3а. Користувач вводить неповні або некоректні дані.
    - Система відображає повідомлення про помилку і просить користувача виправити дані.

2. Редагувати замовлення

Актор: Користувач

Опис: Користувач може редагувати існуюче замовлення в системі.

Передумови: Користувач має доступ до додатку і авторизований для редагування замовлень. Існують замовлення, які можна редагувати.

Основний потік:

1. Користувач вибирає замовлення зі списку для редагування.
2. Відкривається форма для редагування інформації про замовлення.
3. Користувач вносить зміни до даних замовлення.
4. Користувач підтверджує зміни.
5. Система зберігає зміни і оновлює інформацію про замовлення в списку.

Альтернативні потоки:

- 3а. Користувач вводить некоректні дані.
    - Система відображає повідомлення про помилку і просить користувача виправити дані.

3. Видалити замовлення

Актор: Користувач

Опис: Користувач може видалити існуюче замовлення із системи.

Передумови: Користувач має доступ до додатку і авторизований для видалення замовлень. Існують замовлення, які можна видалити.

Основний потік:

1. Користувач вибирає замовлення зі списку для видалення.
2. Система відображає підтвердження видалення.
3. Користувач підтверджує видалення замовлення.
4. Система видаляє замовлення з бази даних і оновлює список замовлень.

Альтернативні потоки:

- 3а. Користувач скасовує видалення.
    - Система не видаляє замовлення і повертає користувача до списку замовлень.

4. Переглядати список замовлень

Актор: Користувач

Опис: Користувач може переглядати список усіх замовлень у системі.

Передумови: Користувач має доступ до додатку і авторизований для перегляду замовлень.

Основний потік:

1. Користувач відкриває додаток.
2. Система відображає список усіх існуючих замовлень.
3. Користувач може вибрати конкретне замовлення для перегляду деталей або виконання інших дій (редагування, видалення).

Ці специфікації варіантів використання описують основні функціональні можливості додатку управління замовленнями і включають основні та альтернативні потоки для кожного випадку використання.

3.3 Проектування внутрішньої будови

Діаграми класів є одним з основних інструментів в об'єктно-орієнтованому аналізі і проектуванні. Вони надають візуальне представлення структури системи, відображаючи класи, їх атрибути, методи та взаємозв'язки між ними. Діаграми класів допомагають розробникам зрозуміти і спроектувати внутрішню структуру системи, а також забезпечують основу для написання коду.

Основні компоненти діаграм класів

Класи (Classes):

Клас представляє собою шаблон або прототип об'єктів, які мають спільні атрибути та методи. Клас визначає структуру та поведінку об'єктів, що створюються на його основі.

Наприклад, у системі для управління бібліотекою класом може бути Book, який має атрибути title, author, ISBN та методи borrow, return.

Атрибути (Attributes):

Атрибути описують властивості класу, що характеризують його стан. Вони визначають дані, які зберігаються у кожному екземплярі класу.

Наприклад, для класу Book атрибутами можуть бути title (назва книги), author (автор книги), ISBN (міжнародний стандартний книжковий номер).

Методи (Methods):

Методи визначають поведінку класу, тобто функції, які може виконувати об'єкт цього класу. Методи можуть маніпулювати атрибутами класу або виконувати певні дії.

Наприклад, для класу Book методами можуть бути borrow() (взяти книгу в бібліотеці) та return() (повернути книгу до бібліотеки).

Зв'язки (Relationships):

Асоціація (Association): Вказує на зв'язок між двома класами, де об'єкти одного класу можуть бути пов'язані з об'єктами іншого класу. Асоціації можуть бути односпрямованими або двоспрямованими.

Агрегація (Aggregation): Особливий вид асоціації, який вказує на відносини "частина-ціле" між класами, де один клас складається з об'єктів іншого класу.

Композиція (Composition): Більш сильна форма агрегації, яка вказує на відносини "жорстка частина-ціле", де частини не можуть існувати без цілого.

Узагальнення (Generalization): Відносини між базовим класом та його підкласами, які успадковують атрибути та методи базового класу.

Використання діаграм класів

Аналіз та проектування системи:

Діаграми класів використовуються для моделювання структури системи на етапі аналізу та проектування. Вони допомагають визначити, які класи будуть існувати у системі, які атрибути та методи вони матимуть, а також які зв'язки існуватимуть між класами.

Комунікація між учасниками проекту:

Діаграми класів забезпечують наочне представлення структури системи, що полегшує комунікацію між розробниками, аналітиками, замовниками та іншими зацікавленими сторонами. Вони допомагають усім учасникам проекту отримати спільне розуміння системи.

Документування системи:

Діаграми класів використовуються як частина технічної документації системи. Вони допомагають новим членам команди швидко зрозуміти структуру системи та її компоненти.

Генерація коду:

Деякі інструменти для моделювання UML (Unified Modeling Language) дозволяють автоматично генерувати початковий код класів на основі діаграм класів. Це зменшує кількість ручної роботи та забезпечує відповідність між моделлю та кодом.

Приклад діаграми класів

Для ілюстрації, розглянемо приклад діаграми класів для системи управління бібліотекою. Основні класи можуть включати Book, Member, Library та Loan.

Клас Book має атрибути title, author, ISBN та методи borrow(), return().

Клас Member має атрибути name, member_id та методи register(), updateProfile().

Клас Library має атрибути name, location та методи addBook(), removeBook().

Клас Loan має атрибути loan_date, return_date та методи extendLoan(), closeLoan().

Зв'язки між цими класами можуть включати асоціації між Member та Loan (член бібліотеки бере книги), а також між Book та Loan (книги видаються на позики).

На рисунку 3.3 представлено діаграму класів системи.

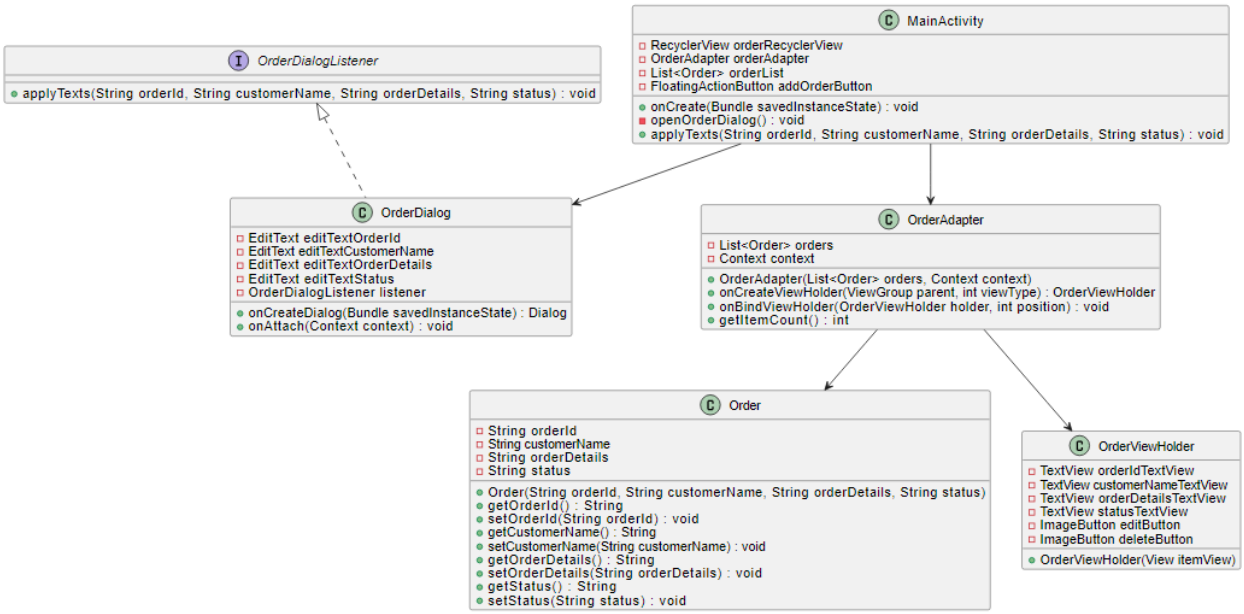

Рисунок 3.3 — Діаграма класів

Діаграма класів описує основні класи додатку управління замовленнями та їх взаємозв'язки. Нижче наведено опис кожного класу та їх взаємодії:

Класи

1. MainActivity
    - Відповідає за головну активність додатку.
    - Поля:
        - RecyclerView orderRecyclerView: відображає список замовлень.
        - OrderAdapter orderAdapter: адаптер для керування даними у RecyclerView.
        - List<Order> orderList: список замовлень.
        - FloatingActionButton addOrderButton: кнопка для додавання нового замовлення.
    - Методи:
        - onCreate(Bundle savedInstanceState): ініціалізує активність.
        - openOrderDialog(): відкриває діалогове вікно для додавання замовлення.
        - applyTexts(String orderId, String customerName, String orderDetails, String status): застосовує введені дані з діалогового вікна для додавання нового замовлення.
2. Order
    - Модель замовлення.
    - Поля:
        - String orderId: ідентифікатор замовлення.
        - String customerName: ім'я клієнта.
        - String orderDetails: деталі замовлення.
        - String status: статус замовлення.

- Методи:
  - Конструктор: Order(String orderId, String customerName, String orderDetails, String status).
  - Геттери та сеттери для кожного поля: getOrderId(), setOrderId(String orderId), getCustomerName(), setCustomerName(String customerName), getOrderDetails(), setOrderDetails(String orderDetails), getStatus(), setStatus(String status).

3. OrderAdapter
   - Адаптер для відображення списку замовлень у RecyclerView.
   - Поля:
     - List<Order> orders: список замовлень.
     - Context context: контекст.
   - Методи:
     - Конструктор: OrderAdapter(List<Order> orders, Context context).
     - onCreateViewHolder(ViewGroup parent, int viewType): створює новий ViewHolder.
     - onBindViewHolder(OrderViewHolder holder, int position): прив'язує дані до ViewHolder.
     - getItemCount(): повертає кількість елементів у списку.
4. OrderViewHolder
   - Відповідає за відображення елементів замовлення у RecyclerView.
   - Поля:
     - TextView orderIdTextView: текстове поле для ідентифікатора замовлення.
     - TextView customerNameTextView: текстове поле для імені клієнта.

        - TextView orderDetailsTextView: текстове поле для деталей замовлення.
        - TextView statusTextView: текстове поле для статусу замовлення.
        - ImageButton editButton: кнопка для редагування замовлення.
        - ImageButton deleteButton: кнопка для видалення замовлення.
    - Конструктор:
        - OrderViewHolder(View itemView): ініціалізує поля.
5. OrderDialog
    - Діалогове вікно для введення даних нового замовлення.
    - Поля:
        - EditText editTextOrderId: поле введення ідентифікатора замовлення.
        - EditText editTextCustomerName: поле введення імені клієнта.
        - EditText editTextOrderDetails: поле введення деталей замовлення.
        - EditText editTextStatus: поле введення статусу замовлення.
        - OrderDialogListener listener: слухач для обробки введених даних.
    - Методи:
        - onCreateDialog(Bundle savedInstanceState): створює діалогове вікно.
        - onAttach(Context context): приєднує слухача до діалогового вікна.
6. OrderDialogListener
    - Інтерфейс для обробки даних з діалогового вікна.
    - Метод:

- applyTexts(String orderId, String customerName, String orderDetails, String status): обробляє введені дані.

Взаємозв'язки

- MainActivity використовує OrderAdapter для керування списком замовлень у RecyclerView.
- MainActivity відкриває OrderDialog для додавання нового замовлення.
- OrderAdapter використовує Order як модель даних.
- OrderAdapter використовує OrderViewHolder для відображення елементів замовлення.
- OrderDialog реалізує інтерфейс OrderDialogListener, який використовується для передачі введених даних з діалогового вікна до MainActivity.

Ця діаграма класів і опис взаємозв'язків показують, як компоненти додатку працюють разом для реалізації функціональності управління замовленнями.

3.4 Розробка графічного інтерфейсу

Інтерфейс нашого додатку управління замовленнями складається з головного екрану, на якому відображається список замовлень, і діалогового вікна для додавання та редагування замовлень.

Головний екран

Головний екран додатку містить такі елементи:

1. Список замовлень: Кожне замовлення відображається у вигляді окремого елементу списку. У кожному елементі списку міститься:
    - Ідентифікатор замовлення (Order ID): Унікальний номер замовлення, відображений у верхньому лівому куті елемента.
    - Деталі замовлення: Опис товарів у замовленні та їх кількість, розташований під ідентифікатором замовлення.
    - Статус замовлення: Поточний статус замовлення (наприклад, "Pending", "Shipped", "Delivered"), розташований під деталями замовлення.
    - Ім'я клієнта: Ім'я клієнта, розташоване праворуч від деталей замовлення.
    - Кнопка редагування: Кнопка у вигляді олівця, яка дозволяє редагувати замовлення.
    - Кнопка видалення: Кнопка у вигляді сміттєвого відра, яка дозволяє видалити замовлення.

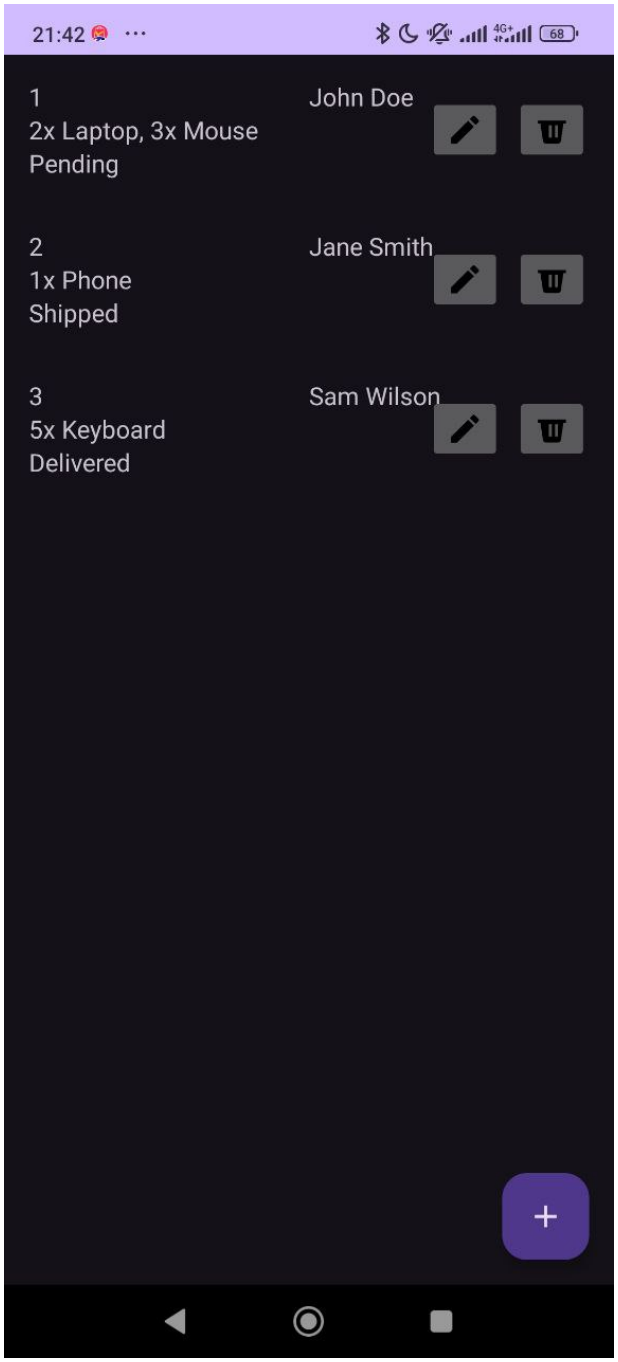

Рисунок 3.4 — Головний екран

2. Плаваюча кнопка дій: Розташована у нижньому правому куті екрана кнопка з символом "+", яка відкриває діалогове вікно для додавання нового замовлення.

   Діалогове вікно додавання/редагування замовлення

Діалогове вікно для додавання або редагування замовлення містить такі елементи:

1. Заголовок: Напис "Add Order" або "Edit Order" залежно від дії, яку виконує користувач.
2. Поле введення ідентифікатора замовлення: Текстове поле для введення унікального номера замовлення.
3. Поле введення імені клієнта: Текстове поле для введення імені клієнта.
4. Поле введення деталей замовлення: Текстове поле для введення опису товарів та їх кількості.
5. Поле введення статусу замовлення: Текстове поле для введення поточного статусу замовлення.
6. Кнопка скасування: Кнопка "cancel" для закриття діалогового вікна без збереження змін.
7. Кнопка підтвердження: Кнопка "add" для збереження нового або відредагованого замовлення.

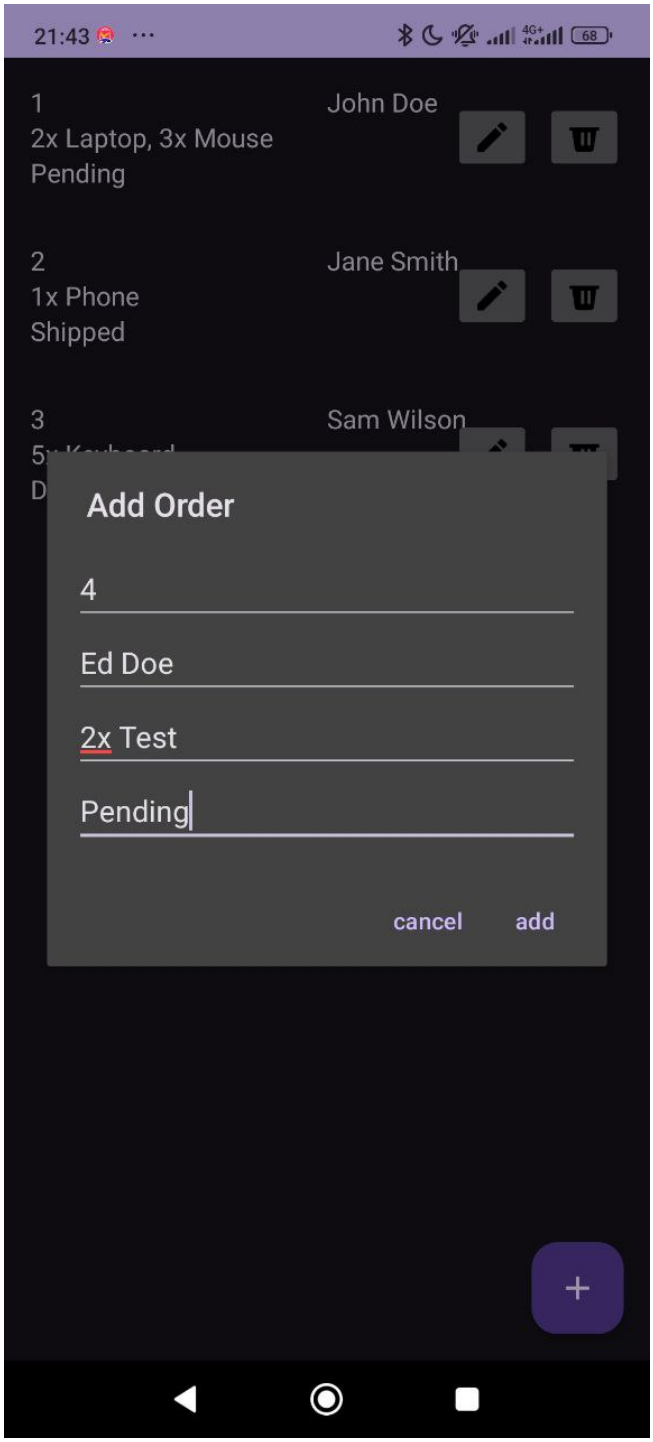

Рисунок 3.5 — Вікно додавання замовлення

Інтерактивність

1. Натискання на плаваючу кнопку дій відкриває діалогове вікно для додавання нового замовлення.

2. Натискання на кнопку редагування у конкретному замовленні відкриває діалогове вікно для редагування цього замовлення з попередньо заповненими полями.
3. Натискання на кнопку видалення відкриває діалогове вікно підтвердження видалення, після чого замовлення видаляється зі списку.

Цей інтерфейс забезпечує інтуїтивно зрозуміле управління замовленнями, дозволяючи користувачам легко додавати, редагувати та видаляти замовлення, а також переглядати детальну інформацію про кожне замовлення.

3.5 Тестування системи

Тестування програмного забезпечення – це процес перевірки та оцінки програмного продукту для забезпечення його відповідності вимогам та очікуванням користувачів. Метою тестування є виявлення дефектів, помилок та недоліків у програмі, а також перевірка її функціональності, продуктивності, безпеки та сумісності. Тестування може бути ручним або автоматизованим і включає різні методи та техніки, такі як модульне тестування, інтеграційне тестування, системне тестування та приймальне тестування.

Таблиця 3.1

Таблиця тестування додатку управління замовленнями

Номер тесту	Тип тесту	Опис тесту	Вхідні дані	Очікуваний результат	Результат тестування
1	Функціональне тестування	Тестування додавання нового замовлення	ID: 4, Ім'я клієнта: Ed Doe, Деталі: 2x Test, Статус: Pending	Нове замовлення успішно додається до списку замовлень	Успішно
2	Функціональне тестування	Тестування редагування існуючого замовлення	ID: 1, Ім'я клієнта: John Doe, Деталі: 3x Laptop, Статус: Shipped	Замовлення успішно оновлюється з новими даними	Успішно

3	Функціональне тестування	Тестування видалення замовлення	ID: 2	Замовлення успішно видаляється зі списку	Успішно
4	Юзабіліті тестування	Тестування інтерфейсу користувача для зручності використання	Використання додатку різними користувачами	Користувачі можуть легко додавати, редагувати та видаляти замовлення	Успішно
5	Тестування навантаження	Тестування продуктивності при великій кількості замовлень	1000+ замовлень у списку	Додаток працює без затримок та крахів	Успішно
6	Тестування сумісності	Тестування додатку на різних пристроях та версіях Android	Пристрої з Android 6.0, 8.0, 10.0, 12.0	Додаток коректно працює на всіх перевірених версіях	Успішно
7	Тестування безпеки	Перевірка на наявність вразливостей, таких як SQL-	Введення спеціальних символів у поля введення	Додаток безпечно обробляє введені дані,	Успішно

		ін'єкції та XSS-атаки		запобігаючи атакам	
8	Тестування стабільності	Тестування тривалого використання додатку без перезавантаження	Тривале використання додатку (4+ години)	Додаток залишається стабільним та не падає	Успішно
9	Інтеграційне тестування	Перевірка інтеграції діалогового вікна з головною активністю	Відкриття та закриття діалогового вікна	Діалогове вікно коректно відкривається та закривається	Успішно
10	Приймальне тестування	Перевірка відповідності додатку вимогам замовника	Виконання сценаріїв використання, зазначених замовником	Додаток відповідає всім вимогам та очікуванням замовника	Успішно

Результати тестування показали, що додаток управління замовленнями успішно пройшов всі етапи перевірки. Додаток коректно виконує свої основні функції: додавання, редагування та видалення замовлень. Інтерфейс користувача є інтуїтивно зрозумілим та зручним у використанні. Під час тестування навантаження додаток продемонстрував високу продуктивність і стабільність

навіть при великій кількості замовлень. Тестування сумісності підтвердило, що додаток працює на різних версіях Android. Безпека додатку також була перевірена, і він успішно захищає від потенційних атак. Таким чином, можна зробити висновок, що додаток відповідає всім вимогам і готовий до використання.

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ

4.1. Аналіз умов праці та шкідливих виробничих чинників

В цьому розділі розглядається аналіз умов праці та пожежна безпека на робочому місці інженера-проектувальника систем «Розумний будинок». Згідно ГОСТу 12.0.003-74 у якому визначені фізичні небезпечні та шкідливі виробничі чинники було виділені ті чинники які присутні на робочому місці інженера-проектувальника систем «Розумний будинок»:

- підвищена чи занижена температура повітря робочої зони;
- підвищений рівень шуму на робочому місці;
- підвищений рівень електромагнітних випромінювань;
- підвищена напруженість електричного поля;
- підвищена напруженість магнітного поля;
- відсутність або нестача природного світла;
- недостатня освітленість робочої зони;
- підвищена яскравість світла;
- знижена контрастність.

В наш час технології безперервно і стрімко розвиваються і професії пов’язані з ІТ стали дуже поширеними. Серед них багато сидячої роботи за комп’ютером. Тому важливо забезпечити безпечні умови праці для робітників, що працюють увесь, або майже увесь час сидячи.

Безпечні та комфортні умови праці забезпечують продуктивну роботу та якісне виконання працівниками конкретної задачі. Безпека на робочому місці потрібна для того щоб запобігти травмам та хворобам працівників на робочих місцях, що в свою чергу також дає змогу підприємству працювати якісно.

4.2. Організація робочого місця під час створення мобільного додатку

Під час створення мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings важливим елементом є ергономіка робочого місця, що являє собою науку про зручність та організацію робочого простору для ефективної та комфортної праці, опираючись на психофізичні особливості організму людини (рис. 4.1).

Дотримання норм, які відзначені на рис. 4.1 знижує втомленість працівника, збільшує ефективність бізнес-процесу та зберігає здоров’я людей. Ергономіка забезпечує зниження навантаження на тіло людини, а нам відомо, що чим більше людина втомлюється, тим гіршою буде її продуктивність, що не вигідно ні працівнику, ні компанії в якій вона працює. Робоче місце повинно бути організоване відповідно стандартів, методичних вказівок та технічних вимог.

Не дивлячись на те, що робота за комп’ютером може здатися цілком безпечною, на відміну від підприємств з більш підвищеною небезпекою, в ній теж є свої нюанси, яких потрібно дотримуватись.

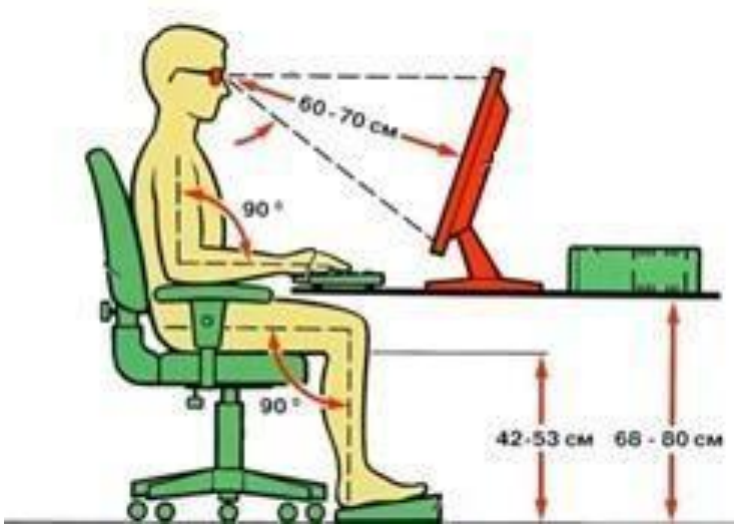

Рис. 4.1. Нормовані значення організації робочого місця під час створення мобільного додатку

Стандарт висоти письмового та комп’ютерного стола 68-80 см, при такій висоті у людини є достатньо міста для ніг. Монітор розміщується на відстані 60-70 см, клавіатура на 10-15 см і вона повинна дозволяти повністю розмістити лікті на столі. Відстань від підлоги до верхнього краю клавіатури 0,7-0,8 м. Відстань від підлоги до центру екрана 0,8-1,9 м.

Якщо недостатньо простору для розміщення усіх необхідних для роботи інструментів, це можна компенсувати розміщенням поряд тумб, стелажу та/або полиць. Усе це повинно бути розташовано по принципу «усе під рукою», що дозволить витрачати менше енергії та направити її на виконання робочого плану.

Якщо людина проводить за ПК більш ніж 6 годин на добу, є ризик розвитку захворювання опорної системи. Для того щоб мінімізувати ризики потрібно обладнати робоче місце спеціальним ортопедичним кріслом для роботи за комп’ютером. Воно оснащене спеціальним валиком у нижній частині спинки, який забезпечує підтримку попереку та повторює анатомічну будову тіла. Спинка крісла у робочому положенні фіксується під прямим кутом 90-95°. Кут між спиною та спинкою крісла 10°-30°. Відстань від підлоги до сидіння крісла 0,3750,5 м. Кут зору 15°-25°. Інформація про ергономіку описана у ГОСТі 12.2.032-78.

4.3. Створення мікроклімату на робочому місці

Для приміщення з ПК існують певні вимоги до вологості, температури та рівню пилу. Температура повинна бути 21…25 °С, відносна вологість – 40…60%, рівень аероіонів – от 400…600 до 50 000 (оптимальний – 1500…5000). Це є оптимальними умовами для комфортного теплового балансу температури тіла людини. На терморегуляцію організму людини також впливає вологість повітря.

Занадто низька вологість, яка менша 20%, викликає пересихання слизових оболонок, а саме дихальних шляхів та очей, а вища 85% ускладнює терморегуляцію. Також дуже важливою є оптимальна вологість, якщо вона вища за норму, то слабкішим стає електростатичне та електромагнітне поле, рівень випромінювання яких в приміщеннях з комп'ютером завжди високий.

Що стосується пилу в приміщеннях з ПК, він є не менш важливим, тому що організм людини погано реагує на велику запиленість. Пил в офісі відрізняється від природнього, він містить частки шкіри людини, будівельних матеріалів, клею, тканей меблів, а також бактерії та віруси. Такий пил може визвати як алергічну реакцію, так і захворювання дихальних шляхів.

Проблемою офісів з комп'ютерами полягає в тому, що через електромагнітне випромінювання пил не осідає на поверхні, він електризується від монітору та висить у повітрі, тому потрапляє на слизові оболонки людини та в легені. Через це вологе прибирання в офісі з ПК повинно проводитися від 3х разів на тиждень.

Також приміщення повинно провітрюватися. Усі заходи безпеки стосовно робочих місць з ПК описані у обов'язкових санітарно-епідеміологічних правилах та нормах – СанПіН 2.2.2/2.4.1340-30 «Гігієнічні вимоги до персональних електронно-обчислювальних машин та організації роботи».

Чи не найбільш важливим є освітлення приміщення та безпосередньо робочого місця, бо більшу частину інформації людина отримує через органи зору, від ступеня втоми очей залежить настрій та самопочуття людини.

Насамперед в приміщенні повинно бути штучне та природне освітлення. Для працівника робоче місце за комп'ютером повинно бути не менше 6 м², а об'єм – більше 20 м³. Має значення й обробка приміщення, а саме її коефіцієнт відображення. Нормою для стін є 0,5-0,6, для стелі 0,7-0,8, для підлоги 0,3-0,5. Для цього застосовують дифузно-видбивні комплектуючі.

Орієнтуватися тільки на природнє освітлення забороняється, але воно є оптимальним, бо більш сприятливе для зору людини. Робоче місце необхідно розмішувати біля вікна. Штучне освітлення використовують, коли природнього

недостатньо. Воно поділяється на загальне, яке використовує систему освітлення стелі, робоче – освітлення на робочому місці здійснюється за допомогою настінних, настільних світильників, та тих, що ставляться на підлогу. Існує документація ДБН В.2.528:20018 , в якій прописані норми та нормативи, які враховуються при організації освітлення при роботі з ПК. Для офісів спільного призначення з використанням комп'ютеру норма освітленості згідно з ДБН 300-500 лк.

Щоб отримати оптимальне освітлення робочого місця, а саме коефіцієнта освітленості потрібно потужність потоку світла розділити на площу. Яскравість освітлення поверхонь, які знаходяться у полі зору повинна бути до 200 кд/м². Яскравість відблисків на екрані монітора не повинна перевищувати 40 кд/м².

4.4. Зниження рівня шуму та забезпечення електробезпеки на робочому місці

Рівень шуму на робочому місці з комп'ютером неповинен перевищувати норм зазначених у СанПіН 2.2.4/2.1.8.562-96. Він складає не більше ніж 50 дБА. Знизити рівень шуму в приміщенні можна за допомогою звукопоглинаючих матеріалів з максимальним коефіцієнтом поглинання звуку в області частот 638000 Гц для обробки стін та стелі робочого приміщення. Джерелами шуму виступають:

- звуки, які доносяться з сусідніх приміщень або вулиці;
- технічні звуки. виникають у процесі функціонування обладнання. щоб мінімізувати шум від нього потрібно використовувати більш якісні пристрої.
- шум джерелом якого є людина. для зменшення шуму в приміщенні існують правила, які встановлює підприємство, порушуючи їх співробітник отримує попередження чи штраф.

На робочому місці працівника розміщуються монітор, клавіатура та системний блок. Коли дисплей включений створюється висока напруга на електронно-променевій трубці в декілька кіловат. Забороняється працювати за комп’ютером, якщо одяг або руки вологі, а також протирати його увімкненому стані.

Потрібно завжди слідкувати за цілісністю проводки, відсутності пошкоджень та наявність заземлення приекранного фільтра. В процесі роботи ПК на корпусах моніторів наведені токи статичної електрики, які при доторканні можуть призвести до розрядів. Вони хоч і не становлять небезпеки для людини, але можуть призвести до поломки комп’ютера.

4.5. Пожежна безпека на робочому місці

Пожежна безпека – комплекс заходів направлених на попередження виникнення випадкової або навмисної пожежі, обмеження та усунення його, якщо він виник та мінімізація наслідків цього явища. Для досягнення потрібного рівня безпеки про роботі з комп’ютером, у виробничому приміщенні повинні бути аптечки першої медичної допомоги, системи автоматичної пожежної сигналізації і вогнегасники. Якщо в приміщені працюють багато комп’ютерів, там повинен бути службовий вимикач, що дозволяє в разі необхідності вимкнути усе живлення кімнати. Пожежна безпека забезпечується пожежною профілактикою та активним пожежним захистом.

Переважна більшість людей гине через токсичність продуктів горіння, а саме отруєнням чадним газом, він більш інтенсивне реагує з гемоглобіном ніж кисень і у людини виникає кисневе голодування та порушення координації рухів. Оксид вуглецю має велику концентрацію в продуктах горіння, тому й створює підвищену небезпеку. Основним токсичними продуктами горіння є оксид сірки та вуглецю, газоподібні кислоти, а саме синильна та соляна, аміак,

альдегіди альфатичні. Чадний газ при концентрації 8-10% приводить до смерті через декілька хвилин.

Температура, яка перевищує 100 °С під час пожежі призводить до втрати свідомості людини і подальше загибелі через декілька хвилин. Така температура може викликати опіки шкіри. Небезпечною температурою вважається від 55 °С. До того ж вона викликає опіки другого ступеня при тривалості впливу 20 с, температура 70 °С завдає шкоди за 1 с.

Для забезпечення пожежної безпеки потрібно проводити бесіди з працівниками стосовно правил пожежної безпеки та не допускати дій, які можуть стати причиною пожежі. Також потрібне встановлення планів евакуації персоналу, технічне обслуговування вогнегасників. Зазвичай причинами пожежі на підприємствах з ПК стають електроприлади, куріння в невстановлених місцях, використання легкозаймистих речовин, порушення технологій, порушення правил використання електроприладів, закрите вентиляційний отвір в електроапаратурі та інше.

Потрібно слідкувати за чистотою приміщення. Сміття та горючі відходи потрібно регулярно утилізувати у спеціально виділене для цього місце.

Евакуаційні виходи, коридори, двері, сходини повинні бути порожніми, нічим не заставлені. Мебель та дроти не повинна бути перешкодою для евакуації людей в разі пожежі. Розташування електричних дротів повинно бути таким щоб вони не пошкоджувались і виключити ризик ураження робітників електричним струмом. По закінченню роботи потрібно вимкнути усі електроприлади та перевірити приміщення.

Для гасіння пожежі у приміщеннях використовують вогнегасники, які призначені для початкової стадії розвитку пожежі. Безпосередньо у приміщеннях з комп’ютерами використовують вогнегасник вуглекислотний ОУ-5, який призначений для гасіння різноманітних матеріалів, електричних установок, які знаходяться під напругою, ПК та оргтехніки. Хід дій такий, що при пожежі потрібно піднести вогнегасник якомога ближче до вогню , направити розтруб у

вогнище, зірвати пломбу, далі відкрити вентиль,натиснути на пусковий важіль, направити газ на вогонь. При цьому розтруб не можна тримати рукою під час його роботи, так як він має дуже низку температуру. Також використовують порошкові вогнегасники ОП-5.

У ДСТУ 3675-98 йдеться про пожежну техніку, вогнегасники переносні, загальні технічні вимоги та методи випробовування. Вогнегасники потрібно грамото розташовувати на підприємстві згідно норм та правил, встановити потрібну кількість та їх положення. Вогнегасники повинні бути на кожному поверсі у кількості не менше ніж 2, у кожного повинен бути сертифікат. Вони повинні бути легкодоступними та розташовуватися у виділених місцях близько до передбачуваного місця пожежі, а також біля евакуаційних шляхів та виходів з приміщення. Вогнегасник повинен бути у робочому стані з запломбованим запірно-пусковим пристроєм. Маса вогнегасника не повинна перевищувати 20 кг. Розташування від підлоги не більше ніж півтора метри до верхньої точки, якщо маса вогнегасника менша ніж 15 кг та один метри, якщо більша. Також можна установити вогнегасник на підставці та на підлогу з надійною фіксацією від падіння. При цьому вогнегасник не повинен заважати пересуванню працівників.

Щоб забезпечити якісний пожежний захист необхідно знати принцип припинення горіння.

Однією з важливих задач пожежної безпеки є забезпечення достатньої міцності будівельної конструкції та захисту приміщень від руйнувань в умовах дії високої температури при пожежі. Приміщення з ПК повинні бути першого та другого ступеня вогнестійкості, через свою велику вартість та категорію пожежної небезпеки

ВИСНОВКИ

У процесі виконання роботи було розроблено Android-орієнтований додаток для управління замовленнями з використанням архітектури мікросервісів. Додаток забезпечує інтуїтивно зрозумілий інтерфейс користувача, що дозволяє легко додавати, редагувати та видаляти замовлення, а також переглядати список існуючих замовлень. В результаті виконаного дослідження та розробки було досягнуто поставлених цілей та завдань.

Важливим аспектом роботи було забезпечення високої продуктивності та стабільності додатку. Проведене тестування підтвердило, що додаток працює коректно навіть при великій кількості замовлень, зберігаючи свою продуктивність та стабільність. Додаток також продемонстрував сумісність з різними версіями Android, що розширює його можливості використання на різних пристроях.

Розробка мобільного додатку для управління замовленнями має значне практичне значення. Він може бути використаний у різних бізнес-середовищах для автоматизації процесів управління замовленнями, що сприятиме зменшенню затрат часу та підвищенню якості обслуговування клієнтів. Завдяки зручному інтерфейсу користувачі можуть швидко і легко взаємодіяти з додатком, що робить його ефективним інструментом для бізнесу.

Таким чином, результати роботи підтверджують актуальність та важливість розробки подібних додатків для сучасних бізнес-процесів. Додаток успішно пройшов усі етапи тестування, відповідає вимогам замовника та готовий до впровадження у практичну діяльність. Розроблені методи та підходи можуть бути використані для подальшого вдосконалення додатку та створення нових рішень у сфері мобільних додатків для бізнесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Anshari, Muhammad; Almunawar, Mohammad Nabil; Lim, Syamimi Ariff; Al-Mudimigh, Abdullah (1 July 2019). "Customer relationship management and big data enabled: Personalization & customization of services". Applied Computing and Informatics. 15 (2): 94–101. doi:10.1016/j.aci.2018.05.004. ISSN 2210-8327. S2CID 67296369.
2. Shaw, Robert (1991). Computer-Aided Marketing & Selling. Butterworth Heinemann. ISBN 978-0-7506-1707-9.
3. "Management Tools – Customer Relationship Management – Bain & Company". www.bain.com. Retrieved 23 November 2015.
4. Hargrave, Marshall. "Customer Relationship Management – CRM goes beyond just software". Investopedia. Retrieved 5 June 2021.
5. "CRM software revenue worldwide 2010-2020".
6. "Market Share: Customer Experience and Relationship Management, Worldwide, 2020".
7. "CRM History: The Evolution Of Better Customer Service". www.streetdirectory.com. Retrieved 24 May 2020.
8. "Software survey: CRM systems in 2021". 27 May 2021. Retrieved 2 June 2021.
9. "How Context Sits at Intersection of CRM, ACD". Retrieved 8 June 2017.
10. Mukherjee, Sharmistha (2 February 2017). "How to build a global company from a small town: The Zoho story". TECHSEEN. Retrieved 17 May 2022.
11. "SAP R/3 SD Wiki". Retrieved 7 January 2019.
12. "It's official: Oracle closes on PeopleSoft acquisition". Computerworld. 10 January 2005. Retrieved 18 August 2021.
13. "Navision 3.0". Archived from the original on 3 June 2021. Retrieved 7 January 2019.

14."History of CRM Software". comparecamp.com. Retrieved 8 February 2017.
15.Jha, Lakshman (2008). Customer Relationship Management: A Strategic Approach. Global India Publications. ISBN 9788190721127. Retrieved 8 June 2017.
16."Gartner Announces Customer Relationship Management Summit 2009". gartner.com. 5 August 2009. Archived from the original on 22 January 2014. Retrieved 8 February 2017.
17."Industry Specific/Vertical Market CRM Solutions". smallbizcrm.com. Retrieved 8 February 2017.
18.The Forrester Wave: CRM Suites For Enterprise Organizations, Q4 2016, Forrester, 21 November 2016, retrieved 13 September 2017
19.Buttle, Francis; Maklan, Stan (11 February 2015). Customer Relationship Management: Concepts and Technologies. Routledge. ISBN 9781317654766.
20.Feiz, Ghotbabadi, Khalifah, (2016-01) Customer Lifetime Value in Organisations
21."Types of CRM and Examples | CRM Software". www.crmsoftware.com. Retrieved 22 November 2015.
22."What is sales force automation (SFA)? - Definition from WhatIs.com". WhatIs.com. Retrieved 26 November 2015.
23.Buttle, Francis (2003). Customer relationship management. London: Routledge. ISBN 9781136412578.
24."What is customer relationship management (CRM) ? - Definition from WhatIs.com". SearchCRM. Retrieved 22 November 2015.
25.Josiah, Ahaiwe; Ikenna, Oluigbo (February 2015). "Role of Technology in Accounting and E-accounting". International Journal of Computer Science and Mobile Computing. 4 (2): 208–215. Retrieved 27 October 2018.
26.Tavana, Ali Feizbakhsh.; Fili, Saeed.; Tohidy, Alireza.; Vaghari, Reza. & Kakouie, Saed. (November 2013). "Theoretical Models of Customer

Relationship Management in Organizations". International Journal of Business and Behavioral Sciences. 3 (11).

27. Greenberg, Paul (13 February 2017). "How customer data platforms can benefit your business". ZDNet.
28. Reinartz, Werner; Krafft, Manfred; Hoyer, Wayne D. (August 2004). "The Customer Relationship Management Process: Its Measurement and Impact on Performance". Journal of Marketing Research. 41 (3): 293–305. doi:10.1509/jmkr.41.3.293.35991. S2CID 167683988.
29. "What's Your Relational Intelligence?". strategy+business. Retrieved 23 November 2015.