МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра фундаментальної та прикладної математики

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА МАТЕМАТИЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АСИМЕТРИЧНОГО ШИФРУВАННЯ»**

Виконала: студентка 4 курсу, групи 6.1139-пмк

спеціальності 113 прикладна математика
(шифр і назва спеціальності)

освітньої програми Математичне та програмне забезпечення криптології
(назва освітньої програми)

К. М. Шапошник
(ініціали та прізвище)

Керівник завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С. М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент декан математичного факультету, професор, д.т.н. Гоменюк С. І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра фундаментальної та прикладної математики

Рівень вищої освіти бакалавр

Спеціальність 113 прикладна математика
(шифр і назва)

Освітня програма математичне та програмне забезпечення криптології

ЗАТВЕРДЖУЮ
Завідувач кафедри фундаментальної та прикладної математики, д.т.н., професор
_____ Гребенюк С.М.
(підпис)

« ____ » _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Шапошник Катерини Миколаївна
(прізвище, ім'я та по-батькові)

1. Тема роботи (проєкту) Розробка математичного та програмного забезпечення для асиметричного шифрування

керівник роботи (проєкту) Гребенюк Сергій Миколайович, д.т.н., професор
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 02.06.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд проблеми
2. Розробка програмного забезпечення
3. Результати застосування програмного забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	30.01.2023	
2.	Збір вихідних даних.	13.02.2023	
3.	Обробка методичних та теоретичних	15.03.2023	
	джерел.		
4.	Розробка першого розділу.	09.04.2023	
5.	Розробка другого розділу.	17.04.2023	
6.	Оформлення та нормоконтроль	02.05.2023	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	19.06.2023	

Студент _____ К. М. Шапошник
(підпис) (ініціали та прізвище)

Керівник роботи _____ С. М. Гребенюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О. Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка математичного та програмного забезпечення для асиметричного шифрування»: 53 с., 12 рис., 7 джерел.

АСИМЕТРИЧНЕ ШИФРУВАННЯ, ВІДКРИТИЙ ТА ЗАКРИТИЙ КЛЮЧ, МОВА ПРОГРАМУВАННЯ KOTLIN, RSA АЛГОРИТМ, СИСТЕМА КОНТРОЛЮ ДОСТУПУ.

Об’єкт дослідження – процес розробки системи контролю доступу із застосуванням асиметричного шифрування.

Предмет дослідження – розробка програмне забезпечення для системи контролю доступу із застосуванням RSA шифрування.

Мета роботи – розробити програмне забезпечення для системи контролю доступу із застосуванням асиметричного шифрування.

У кваліфікаційній роботі бакалавра поставлено завдання розробити компоненти системи контролю доступу до приміщень. Проаналізовано математичний апарат основних методів шифрування та дешифрування. На основі RSA методу шифрування було створено програмне забезпечення прикладної системи «Ключ і замок». В проєкті забезпечено повну реалізацію та логічний взаємозв’язок усіх основних функцій між додатками та хмарною базою даних.

Для оптимізації ефективності та мінімізації витрат, пов’язаних із системою, була розроблена програма «Дверний замок» на основі мікрочіпа спеціально для реального використання в готельному середовищі. Ця програма слугувала емулятором, надаючи наочну демонстрацію взаємодії програма-користувач у системі контролю доступу. Програмне забезпечення реалізовано на мові програмування Kotlin.

SUMMARY

Bachelor’s qualifying theses "Development of Mathematical and Software for Asymmetric Encryption": 53 pages, 12 illustrations, 7 references.

ACCESS CONTROL SYSTEM, ASYMMETRIC ENCRYPTION, KOTLIN PROGRAMMING LANGUAGE, OPEN AND CLOSED KEY, RSA ALGORITHM.

The object of research is the process of developing an access control system using asymmetric encryption.

The subject of research is the development of software for an access control system using RSA encryption.

The purpose of the work is to develop software for an access control system using asymmetric encryption.

In the qualification work of the bachelor, the task was set to develop the components of the access control system to the premises. The mathematical apparatus of the main methods of encryption and decryption is analyzed. Based on the RSA encryption method, the application system software "Key and Lock" was created. The project provides full implementation and logical interconnection of all basic functions between applications and the cloud database.

To optimize efficiency and minimize the costs associated with the system, a microchip-based Door Lock application was developed specifically for real-world use in a hotel environment. This program served as an emulator, providing a visual demonstration of program-user interaction in an access control system. The software is implemented in the Kotlin programming language.

ЗМІСТ

ВСТУП

Проблема контролю доступу до приміщень, авторизації користувачів є досить актуальною проблемою, пов’язаною із кодуванням та декодуванням даних. Розв’язок цієї проблеми неможливий без використання математичних методів шифрування та дешифрування й створення відповідного програмного забезпечення, що реалізує його.

Використання програм «Дверний замок» і «Ключ» в готелях дає численні переваги. Ці програми служать різним цілям, які сприяють зручності та безпеці як працівників готелю так і самих гостей.

Інтеграція карток у програму «Дверний замок» та «Ключ» забезпечує безпечний і зручний спосіб доступу до номерів готелю. Замість того, щоб носити з собою додаткову картку чи запам’ятовувати коди, працівники та гості можуть зручно використовувати власні мобільні пристрої, які зазвичай у них під рукою.

Крім того, використовуючи технологію Nearby Share, додаток «Дверний замок» і «Ключ» покращує контроль доступу порівняно з традиційними системами на основі карток. Зв’язок у режимі реального часу між програмами DoorLock і Key забезпечує швидкий і безпечний доступ, коли пристрій знаходиться поблизу.

Впровадження за допомогою Firebase Database і DataStore Android для зберігання інформації, дозволяє готелям ефективно контролювати історію доступу. Ця функція є безцінною для цілей відстеження, допомагаючи у виявленні будь-яких порушень безпеки або несанкціонованого використання та полегшуючи відповідні дії.

Програми «Дверний замок» і «Ключ» використовують асинхронну криптографію, забезпечуючи додатковий рівень безпеки. Це гарантує, що лише авторизовані користувачі можуть отримати доступ, зберігаючи конфіденційність і цілісність інформації.

Нарешті, гнучкість віддаленого оновлення та модифікації програми «Дверний замок» і «Ключ» дозволяє зручно керувати доступом до номерів для різних людей. Такої адаптивності було б складніше досягти за допомогою фізичних карток. Ці додатки дозволяють контролювати використання картки, допомагаючи виявляти будь-які порушення безпеки або зловживання.

Одними із основних проблем, з якими стикаємося при виконанні кваліфікаційної роботи, є технічні труднощі. Можуть виникнути технічні проблеми, пов’язані з розробкою додатків, як-от труднощі з впровадженням технології Nearby Share, труднощі з впровадженням асинхронної криптографії або труднощі з інтеграцією бази даних Firebase і DataStore Android.

Зауваження щодо безпеки: можуть виникнути проблеми безпеки, пов’язані зі зберіганням і передачею конфіденційних даних, таких як картки та ключі доступу, які необхідно вирішити, щоб забезпечити безпеку програми.

Проблеми з конфіденційністю: можуть виникнути проблеми з конфіденційністю, пов’язані з використанням біометричних даних, таких як відбитки пальців чи розпізнавання обличчя, або проблеми щодо зберігання та використання даних гостей та працівників, які необхідно вирішити, щоб забезпечити відповідність програми відповідним нормам і найкращим практикам.

Обмежені ресурси: розробка програми може бути обмежена доступними ресурсами, такими як бюджет, персонал або обладнання, що може вплинути на обсяг або функціональність програми.

Прийняття користувачами: може бути важко змусити користувачів прийняти та прийняти програму, особливо якщо вони звикли до традиційних систем на основі карток або якщо вони мають проблеми з безпекою чи конфіденційністю.

Обслуговування: зберігання та оновлення програм може бути проблемою, особливо якщо вам потрібно додати нові функції чи виправити помилки.

Сумісність: програма може працювати не на всіх пристроях і операційних системах, що може обмежити її використання.

Тестування: може бути важко перевірити програму та переконатися, що вона працює правильно в усіх сценаріях.

Однією з головних проблем у впровадженні бездротових технологій у мобільних додатках є відсутність стандартизації та сумісності між різними пристроями та платформами. Через це може бути важко забезпечити належну роботу програми на всіх пристроях і операційних системах.

Іншою проблемою є відсутність документації та вказівок розробників, що може перешкодити правильному та ефективному впровадженню бездротових технологій. Це може призвести до вразливості безпеки та низької продуктивності мобільного додатка.

Дослідження також показали відсутність розуміння різноманітних бездротових технологій та їхніх можливостей серед розробників, що може призвести до впровадження невідповідних або неефективних рішень.

Крім того, існує низка проблем безпеки та конфіденційності, пов’язаних із використанням бездротових технологій у мобільних програмах. Дослідження показали, що багато мобільних додатків не впроваджують належним чином заходи безпеки для захисту конфіденційних даних, що може призвести до витоку даних та інших проблем безпеки.

Щоб вирішити ці проблеми, дослідники запропонували різні рішення, такі як використання стандартизованих протоколів, розробка рекомендацій і найкращих практик для розробників, а також використання технологій підвищення безпеки та конфіденційності.

1 МЕТОДОЛОГІЯ СТВОРЕННЯ СИСТЕМИ ДОСТУПУ ДО ПРИМІЩЕНЬ

1.1 Опис програми «Ключ»

Програма «Ключ» – це передове рішення для керування доступом у готельному середовищі. Це забезпечує безпечний, ефективний і зручний спосіб доступу гостей, персоналу та адміністраторів до зон обмеженого доступу. Кроки роботи програми:

а) автентифікація: після запуску програми користувачам пропонується пройти автентифікацію за допомогою адреси електронної пошти та пароля. Процесом автентифікації керує автентифікація Firebase, яка перевіряє облікові дані користувача в авторизованій базі даних користувачів. Лише адміністратори можуть додавати користувачів до бази даних і не можуть створювати нові облікові записи в програмі;

б) основний інтерфейс: після успішної автентифікації користувачам відкривається головний інтерфейс програми. Для гостей інтерфейс помітно відображає їхні особисті дані (ім’я, номер кімнати) у форматі картки, що нагадує готельну картку-ключ. У верхній частині екрана є кнопка, яка надає спадний список доступних зручностей і послуг;

в) доступ до кімнати: після натискання кнопки зі списком зручностей, розкривний список розширюється посередині екрана, показуючи всі доступні зручності та зони, до яких гість наразі має доступ. У випадках, коли є багато кімнат або зон, список можна прокручувати вгору та вниз для зручності використання. Кожна кімната або зона відповідає фізичній картці-ключу, за допомогою якої можна відкривати двері, а не покладатися на додаток.

Додаток «Ключ» працює безпечно та ефективно з точки зору користувача. На головному екрані програми представлено два поля для введення даних: адреса електронної пошти та пароль, а також кнопка входу. Коли ви натискаєте кнопку входу, програма надсилає запит до бази даних автентифікації Firebase для перевірки облікових даних користувача. Якщо користувача знайдено та пароль збігається, користувач буде перенаправлено на головний екран програми. З іншого боку, якщо користувача не знайдено або пароль не збігається, буде показано сповіщення про помилку входу.

Головний екран формується на основі типу доступу користувача. Ця інформація отримується з бази даних автентифікації Firebase разом із електронною адресою користувача, ім’ям і прізвищем, які зберігаються в сховищі даних програми. Якщо користувач має тип доступу гостю готелю, відображається Guest. Якщо користувач має тип доступу співробітників готелю, відображається Staff. А також, якщо користувач має тип доступу як універсальний ключ, відображається Admin.

У верхній частині екрана відображається список доступних номерів, який оновлюється в режимі реального часу на основі хмари. Після відкриття дверей номеру фон програми засвітиться зеленим світлом на 5 секунд, щоб вказати на успішну взаємодію із замком.

Програма «Ключ» пропонує кілька переваг для готелів і працює ефективно, щоб оптимізувати керування доступом.

Покращена безпека: програма «Ключ» забезпечує безпечну систему керування доступом, гарантуючи, що лише авторизовані особи можуть отримати доступ до зон обмеженого доступу. Використовуючи автентифікацію Firebase і перевірену базу даних користувачів, програма перевіряє облікові дані користувача, зменшуючи ризик неавторизованого доступу.

Зручний користувальницький досвід: програма пропонує зручний інтерфейс, що полегшує навігацію та використання функцій для гостей, персоналу та адміністраторів. Інтуїтивно зрозумілий дизайн дозволяє

користувачам швидко автентифікуватися, переглядати свої особисті дані та отримувати доступ до доступних зручностей або кімнат.

Ефективне керування доступом: програма ефективно керує доступом, надаючи чіткий огляд доступних кімнат. Користувачі можуть легко переглядати та вибирати зі списку номерів або зручностей, до яких вони мають доступ, заощаджуючи час і мінімізуючи плутанину, особливо у випадках, коли є багато варіантів.

Повна інтеграція: програма «Ключ» легко інтегрується з існуючими готельними системами та процесами. Використовуючи автентифікацію Firebase і можливості бази даних, програма може синхронізуватися з іншими системами управління готелем, забезпечуючи точну й актуальну інформацію про гостей, персонал і доступ до номерів.

Покращене адміністрування: програма спрощує адміністративні завдання, пов’язані з керуванням доступом. Адміністратори можуть легко додавати або видаляти користувачів з авторизованої бази даних, зберігаючи контроль над правами доступу. Крім того, додаток дозволяє ефективно контролювати та перевіряти дії доступу, підвищуючи заходи безпеки.

Інтеграція з фізичною карткою-ключем: програма Ключ інтегрується з фізичною карткою-ключем, надаючи користувачам звичну та зручну альтернативу для доступу до своїх кімнат. Додаток діє як цифрове представлення картки-ключа, дозволяючи користувачам автентифікувати та розблоковувати свої кімнати за допомогою своїх смартфонів.

Масштабованість: програма є масштабованою та може задовольнити потреби готелів різного розміру та місткості номерів. У міру того, як готель розширюється або додає нові зручності, додаток можна легко адаптувати для включення додаткових номерів або зручностей і керування ними.

Додаток «Ключ» пропонує індивідуальні інтерфейси для різних ролей користувачів. Давайте розглянемо особливості кожної ролі:

а) GuestUI: GuestUI представляє зображення картки переходу в центрі екрана, що відображає ім’я гостя. У верхній частині екрана

відображається список доступних кімнат, який оновлюється в режимі реального часу з хмарної бази даних про наявність кімнат. Це забезпечує точну та актуальну інформацію для гостей;

б) StaffUI: Подібно до GuestUI, StaffUI також містить зображення картки переходу в центрі екрана, на якому відображається ім’я співробітника. У верхній частині екрана відображається список доступних об’єктів або зон, як-от конференц-зали, фітнес-центри або спа-послуги. Цей список оновлюється в режимі реального часу, що дозволяє співробітникам легко отримувати доступ до доступних зручностей і керувати ними;

в) AdminUI: AdminUI містить зображення картки переходу в центрі екрана, на якому відображається ім’я адміністратора. Адміністратор, як користувач із правами доступу, має головний ключ, який надає доступ до всіх номерів і об’єктів у готелі. Тому немає потреби в кнопці «Доступні кімнати» на верхній панелі головного інтерфейсу користувача. AdminUI дозволяє адміністратору ефективно керувати доступом до кімнат і об’єктів, зокрема надавати та скасовувати доступ для гостей і співробітників.

Крім того, AdminUI дозволяє адміністратору переглядати та затверджувати або відхиляти запити на доступ до кімнат або зручностей поза стандартною доступністю. Ця функція забезпечує ефективне використання ресурсів готелю та допомагає підтримувати безперебійну роботу.

1.2 Додаток «Дверний замок»

Додаток «Дверний замок» розроблено для забезпечення безконтактного доступу до номерів і об’єктів готелю. Додаток використовує такі технології, шифрування та дешифрування RSA, Nearby Share та підключення до Firebase.

У цьому описі ми розглянемо технічні аспекти, дизайн інтерфейсу та функціональність програми.

Технічний опис.

Після активації програма «Дверний замок» генерує пару ключів за алгоритмом RSA. Приватний ключ надійно зберігається в локальному сховищі програми, тоді як відкритий ключ надсилається на сервер Firebase, зокрема на таблицю Rooms. Програма оновлює генерацію пари ключів кожні 5 годин.

Інтерфейс програми «Дверний замок:

Інтерфейс програми розроблено з центральним значком чорної клавіші на білому тлі. Коли двері успішно розблоковано, фон стає зеленим, що вказує на успішну взаємодію. Зелений фон залишається видимим протягом 5 секунд, а потім повертається до стандартного стану.

Наявність номерів для користувачів:

Додаток «Дверний замок» отримує дані про наявність номерів у реальному часі з хмарної бази даних, зокрема таблиці AvailableRooms. Потім ці дані зберігаються в локальному сховищі даних програми в таблиці AvailableRooms. Таблиця AvailableRooms містить інформацію про користувачів, які мають доступ до певних кімнат. Додаток «Дверний замок» використовує збережені дані для розблокування відповідних номерів.

Режим спільного доступу до хосту поблизу:

Після початкового налаштування програма «Дверний замок» переходить у режим Host Nearby Share. У цьому режимі програма очікує зв’язку з іншими пристроями, на яких запущено програму Key, для обміну інформацією. Це дозволяє додатку «Дверний замок» розпізнавати та автентифікувати користувачів з інших пристроїв, на яких запущено додаток «Ключ», забезпечуючи плавний доступ до номерів і об’єктів.

Ці адаптації відображають функціональні можливості та роботу програми «Дверний замок», забезпечуючи безпечне та зручне рішення для керування доступом для гостей і персоналу готелю.

Шифрування в додатку «Ключ»:

Програма «Ключ» включає шифрування як важливий компонент. Він безпечно передає попередньо зашифровані дані по бездротовому зв’язку на коротку відстань за допомогою відкритого ключа, отриманого з хмарної бази даних готелю, зокрема таблиці номерів. Зашифровані дані містять унікальний ідентифікатор користувача, пов’язаний із програмою «Ключ».

Розшифровка в додатку «Дверний замок»:

Після встановлення з’єднання між програмою «Дверний замок» і програмою «Ключ» зашифровані дані приймаються та розшифровуються програмою «Дверний замок» за допомогою її закритого ключа.

Програма «Дверний замок» спеціально розроблена для полегшення безконтактного доступу до різноманітних номерів і об’єктів готелю. Він використовує такі технології, як Kotlin, шифрування та дешифрування RSA, Nearby Share та підключення до Firebase. У цій документації ми надамо вичерпний огляд технічних аспектів, дизайну інтерфейсу та робочих функцій програми.

Технічний опис:

Після запуску програма «Дверний замок» генерує пару ключів за алгоритмом RSA. Приватний ключ надійно зберігається в базі даних програми, а відповідний відкритий ключ передається на сервер Firebase, зокрема в таблицю Rooms. Додаток «Дверний замок» періодично оновлює генерацію пари ключів кожні 5 годин.

Інтерфейс програми «Дверний замок»:

В інтерфейсі користувача програми помітно зображено чорний значок замка, розташований у центрі екрана. Після успішного доступу до кімнати фон інтерфейсу стає яскраво-зеленим. Зелений фон залишається видимим протягом 5 секунд, що свідчить про успішну взаємодію із замком.

Наявність номерів для користувачів:

Додаток «Дверний замок» отримує дані про наявність номерів із хмарної бази даних готелю, зокрема таблиці AvailableRooms. Потім ці дані

зберігаються в локальному сховищі даних програми за допомогою таблиці AvailableRooms. Таблиця AvailableRooms містить відповідну інформацію про дозволи користувача на доступ до певних кімнат, ідентифікованих за ідентифікатором кімнати, пов’язаним із програмою «Ключ».

Режим спільного доступу до хосту поблизу:

Після початкового налаштування програма «Дверний замок» переходить у режим Nearby Host Sharing. У цьому режимі програма очікує зв’язку з іншими пристроями, на яких працює програма “Ключ», щоб полегшити обмін даними. Ця функція гарантує, що програма може розпізнавати та автентифікувати користувачів з інших пристроїв, на яких працює програма «Ключ», таким чином забезпечуючи плавний доступ до номерів і об’єктів.

Шифрування в додатку «Ключ» використовує відкритий ключ, отриманий із хмарної бази даних, забезпечуючи безпечний спосіб передачі даних бездротовим способом.

Розшифровка в додатку "Дверний замок":

Після перевірки програми «Дверний замок» із підтримкою «Ключ» та успішного надсилання символів програма «Дверний замок» розшифрує текст за допомогою закритого ключа.

Ідентифікація користувача:

Після розшифровки додаток «Дверний замок» ідентифікує користувача на основі унікального ідентифікатора, наявного в розшифрованому тексті. Програма «дверний замок» перевіряє наявність унікального ідентифікатора користувача в таблиці AvailableRooms у сховищі даних програми. Якщо користувача буде успішно знайдено, додаток «Дверний замок» перейде в режим «Відкриті двері».

Режим відкриття дверей:

Додаток «Дверний замок» перемикається в режим «Відкриті двері» після ідентифікації користувача. У цьому режимі фон програми стає зеленим, що імітує успішне відкриття дверей готелю. Режим «Відкриті двері»

активується на 5 секунд, після чого програма повертається до нормального стану та активує режим «Host Nearby Share».

Таблиця «Історія відкриття дверей» :

Коли користувач успішно взаємодіє з програмою «Дверний замок», програма зберігає час, дату й унікальний ідентифікатор користувача в таблиці «Історія відкриття дверей» . Програма «Дверний замок» спочатку зберігає ці дані в сховищі даних і оновлює таблицю в однойменній хмарній базі даних «Історія відкриття дверей» кожні 10 годин даними з цієї таблиці, додаючи лише номер готелю цього дверного замка.

Додаток «Дверний замок» забезпечує безпроблемний безконтактний доступ до номерів і кімнат готелю. Завдяки передовим технологіям, таким як Kotlin, шифрування RSA, Nearby Share і Firebase, програма надає студентам і викладачам безпечний і надійний спосіб доступу до визначених місць. Інтерфейс користувача програми простий, інтуїтивно зрозумілий і легкий у використанні з чіткою індикацією стану блокування.

Функції безпеки програми мають вирішальне значення для її успіху. Програма «Дверний замок» використовує алгоритм шифрування RSA, щоб забезпечити безпеку зв’язку між програмою та сервером. Програма генерує унікальну пару ключів, із закритим ключем, який зберігається локально на пристрої, а відкритий ключ надсилається на сервер Firebase. Це гарантує, що лише авторизовані користувачі мають доступ до визначених зон.

Додаток також використовує спільний доступ поблизу для зв’язку з додатком «Ключ». Ця функція дозволяє бездротовим способом обмінюватися інформацією між пристроями на короткій відстані. Програма «Ключ» використовує відкритий ключ, отриманий із сервера Firebase, для шифрування попередньо визначеного рядка, що містить унікальний ідентифікатор користувача. Потім програма «Дверний замок» використовує закритий ключ для розшифровки отриманого рядка, таким чином підтверджуючи особу користувача.

Щоб підтримувати надійність програми, вона використовує функцію Kotlin Coroutines для виконання періодичних оновлень кожні п’ять годин. Ці оновлення регенерують пару ключів, оновлюють таблицю доступних класів і перезапускають функції програми. Крім того, програма записує всі спроби доступу до таблиці «Історія відкриття дверей» у хмарі, оновлюючи її кожні 10 годин новими даними з програми Data Store.

Додаток «Дверний замок» забезпечує надійне та безпечне рішення для доступу до аудиторій та спеціально відведених місць в університеті. Його розширені функції та зручний інтерфейс роблять його практичним рішенням для студентів і викладачів. Періодичні оновлення програми, функції безпеки та можливості журналювання забезпечують її ефективну роботу.

1.3 Kotlin

Kotlin – сучасна мова програмування, яка широко використовується для розробки різноманітних додатків, у тому числі систем електронних ключів для готелів. Він був розроблений JetBrains і офіційно оголошений Google як першокласна мова для розробки додатків Android у 2017 році. Kotlin пропонує стислий, виразний і сумісний синтаксис, що робить його потужним вибором для розробки надійних і ефективних програмних рішень.

Ось ключові функції та переваги Kotlin, завдяки яким він добре підходить для розробки електронних ключів для готелів:

Стислий і зрозумілий синтаксис: Kotlin має стислий синтаксис, який зменшує шаблонний код і покращує його читабельність. Він надає різні мовні функції, такі як визначення типу, захист від нульових значень, лямбда-вирази та функції розширення, які сприяють написанню чистого та виразного коду. Kotlin більш виразний, ніж Java, що полегшує написання та читання коду, а також робить його більш безпечним. Чим виразніша мова, тим

чіткіше вона передає задум творця. Це полегшує іншим розробникам розуміння та підтримку коду, а також ускладнює прослизання помилок.

Взаємодія з Java: Kotlin повністю сумісний з Java, що дозволяє розробникам легко інтегрувати код Kotlin в існуючі проєкти Java. Це робить його практичним вибором для розробки ключових систем готелю, оскільки він може використовувати існуючі бібліотеки та фреймворки Java.

Безпека нульових значень: система типів Kotlin включає вбудовані функції безпеки нульових значень, що знижує ризик виникнення винятків нульового покажчика. Це особливо важливо для ключових систем готелю, де забезпечення безпеки та надійності контролю доступу має вирішальне значення. Крім того, функція нульової безпеки Kotlin гарантує, що ваша програма не викидає винятки нульового покажчика, які можуть призвести до збою вашої програми. З іншого боку, визначення типу автоматично визначає тип змінної, що усуває потребу в явних оголошеннях типу та робить код більш читабельним і лаконічним.

Співпрограми для асинхронного програмування: Kotlin забезпечує вбудовану підтримку співпрограм, що спрощує асинхронне програмування. Співпрограми дозволяють розробникам писати асинхронний код послідовним і лінійним способом, полегшуючи виконання таких завдань, як мережеві запити та фонові операції в системі ключів готелю.

Об’єктно-орієнтоване та функціональне програмування: Kotlin підтримує парадигми об’єктно-орієнтованого та функціонального програмування, надаючи розробникам гнучкість у проєктуванні архітектури ключової системи готелю. Він дозволяє використовувати класи, успадкування, інтерфейси, а також функції вищого порядку, лямбда-вирази та незмінність, сприяючи модульності коду та зручності обслуговування.

Інструменти та підтримка спільноти: Kotlin має чудову підтримку інструментів, включаючи потужне інтегроване середовище розробки (IDE), таке як IntelliJ IDEA, яке забезпечує інтелектуальне завершення коду, інструменти рефакторингу та можливості налагодження. Спільнота Kotlin є

активною та зростаючою, з достатньою кількістю ресурсів, бібліотек і фреймворків, які розробники можуть використовувати у розробці ключових систем для готелів. Kotlin розроблений як кросплатформенний і може використовуватися для створення додатків для багатьох платформ, включаючи Android, iOS, Інтернет і робочий стіл. Kotlin також має чудову взаємодію з Java, що означає, що ви можете легко викликати код Java з Kotlin і навпаки. Це означає, що можна використовувати існуючі бібліотеки Java у коді Kotlin, а також можна використовувати код Kotlin у своєму коді Java.

Покращена обробка помилок: блок try-catch у Kotlin є виразом, а не оператором, тобто його можна використовувати в правій частині призначення. Це дає змогу обробляти помилки більш стислим і зрозумілим способом, а також дає змогу повертати значення чи створювати виняток. Ця функція також дозволяє вам писати більш функціональний код, де функції можуть бути складені разом для створення більш складних операцій. Це полегшує міркування про код, а також робить його більш придатним для перевірки.

Функції розширення: функції розширення Kotlin дозволяють додавати нові функції до існуючих класів без необхідності їх успадковувати. Це може бути корисним для додавання допоміжних функцій і службових методів до існуючих бездротових класів, що може зробити ваш код більш читабельним і придатним для повторного використання. Функції розширення – це потужна функція, яка дозволяє додавати функціональність до існуючих класів без необхідності змінювати сам клас. Це означає, що ви можете додавати функції до класів, яких у вас немає, а також ви можете додавати функції до класів, які є частиною Android SDK.

Класи даних: класи даних Kotlin дозволяють визначати прості класи, які містять дані. Ці класи мають вбудовану підтримку equals, hashCode і toString, що полегшує написання та підтримку вашого коду. Класи даних – це потужна функція, яка дозволяє визначати прості класи лише для даних без необхідності писати стандартний код для методів equals, hashCode і toString.

Це означає, що ви можете зосередитися на самих даних, а не на деталях впровадження. Крім того, класи даних також мають вбудовану підтримку для декларацій деструктуризації, що дозволяє витягувати властивості класу та призначати їх окремим змінним. Це може спростити роботу з даними, а також зробити ваш код більш читабельним.

Ще однією перевагою використання Kotlin для розробки мобільних додатків є підтримка функціонального програмування. Функціональне програмування — це парадигма програмування, яка наголошує на незмінності, декларативному коді та функціях без стану. Це може допомогти зробити ваш код більш передбачуваним, зручним для перевірки та обслуговуванням. Kotlin надає низку функцій для підтримки функціонального програмування, таких як лямбда-вирази, функції вищого порядку та визначення типу. Це дозволяє розробникам писати чистий, стислий і читабельний код, що може покращити загальну якість програми.

Kotlin також має зростаюче співтовариство та екосистему, яка може надати велику кількість ресурсів і підтримки для розробників. Мова програмування Kotlin активно розробляється та підтримується JetBrains, компанією, що стоїть за популярним середовищем розробки IntelliJ IDEA. У JetBrains є спеціальна команда розробників, яка працює над мовою, і компанія також прагне підтримувати спільноту та екосистему Kotlin. Це означає, що розробники можуть покладатися на велику й активну спільноту розробників, а також на широкий спектр бібліотек і інструментів, які допоможуть їм створювати кращі мобільні програми.

Таким чином, Kotlin є універсальною та сучасною мовою програмування, яка пропонує численні переваги для розробки систем електронних ключів для готелів. Його стислий синтаксис, сумісність із Java, нульові функції безпеки, співпрограми для асинхронного програмування, підтримка як об’єктно-орієнтованого, так і функціонального програмування, потужні інструменти та підтримка спільноти роблять його ідеальним вибором для створення надійних і ефективних ключових програм для готелів.

Kotlin є потужною та універсальною мовою програмування, яка чудово підходить для розробки мобільних додатків. Він пропонує низку функцій і переваг перед Java, таких як нульовий рівень безпеки, визначення типу, підтримка між платформами, покращена обробка помилок, співпрограми, функції розширення, класи даних і підтримка функціонального програмування. Він також має зростаючу спільноту та екосистему, яка надає розробникам велику кількість ресурсів і підтримки. Усі ці функції роблять Kotlin чудовим вибором для розробки мобільних додатків і допомагають розробникам створювати більш надійні, безпечні та ефективні мобільні додатки.

1.4 Алгоритм RSA

У період, коли інформаційна безпека має першорядне значення, алгоритм RSA став важливим інструментом для безпечного зв'язку та захисту даних.

Алгоритм RSA заснований на математичних властивостях простих чисел, модульної арифметики та теорії чисел. Він використовує два великі прості числа для генерації відкритого та закритого ключів, які складають основу процесів шифрування та дешифрування. Безпека алгоритму залежить від обчислювальної складності факторизації великих чисел, що формує основу його стійкості до атак.

Генерація ключів. Щоб розпочати процес шифрування, вибираються два простих числа p і q. Їх добуток n служить модулем алгоритму. Крім того, функція Ейлера використовується для обчислення значення експоненти e, яка взаємно проста з (p-1) та (q-1). Відкритий ключ складається з пари (n, e), а закритий ключ виходить із простих чисел p, q та відкритого показника ступеня e.

Шифрування та дешифрування. Коли повідомлення необхідно зашифрувати, воно подається як числове значення з використанням призначеної схеми кодування. Зашифрований текст генерується шляхом зведення числового значення ступінь публічного показника *e* по модулю *n*. Розшифровка виконується із застосуванням закритого ключа, що включає зведення зашифрованого тексту в ступінь приватного показника *d* по модулю *n*. Вихідне повідомлення відновлюється за допомогою процесу зворотного кодування.

Безпека та практичне застосування. Безпека алгоритму RSA залежить від складності розкладання великих чисел на прості множники. Ця фундаментальна математична проблема лежить в основі його стійкості до грубої сили та інших відомих атак. Алгоритм RSA знаходить широке застосування в різних областях, включаючи безпечний зв'язок електронною поштою, цифрові підписи, безпечні онлайн-транзакції та безпечну передачу даних по мережах. Його широке поширення є свідченням його міцності та надійності.

Переваги. Сила алгоритму RSA полягає у його здатності забезпечувати безпечний зв'язок без необхідності безпечного каналу для обміну ключами. Його асиметричний характер дозволяє зручно розподіляти відкриті ключі, одночасно надійно зберігаючи закритий ключ у передбачуваного одержувача. З розвитком обчислювальної потужності алгоритм RSA еволюціонував використовувати ключі більшого розміру для забезпечення своєї безпеки.

Алгоритм RSA зробив революцію в сучасній криптографії, надавши потужні та ефективні засоби безпечного зв'язку та захисту даних. Використовуючи математичні принципи простих чисел та модульну арифметику, він став незамінним інструментом для забезпечення конфіденційності та цілісності конфіденційної інформації. У міру розвитку технологій алгоритм RSA продовжує розвиватися, щоб відповідати вимогам безпечного зв'язку, що постійно зростають, в епоху цифрових технологій.

Алгоритм RSA є свідченням геніальності його творців та сили математики у сфері шифрування. Його значення у забезпеченні безпеки нашого цифрового світу неможливо переоцінити, а його постійна актуальність зміцнює його положення як наріжний камінь у галузі криптографії.

Щоб створити пару ключів RSA, потрібно виконати такі дії:

а) вибрати два різних простих числа p і q;

б) обчислити $n = p * q$;

в) обчислити коефіцієнт $\phi(n) = (p-1) * (q-1)$;

г) вибрати таке ціле число e, щоб $1 < e < \phi(n)$ і e було взаємно простим з $\phi(n)$ (тобто e і $\phi(n)$ не мали спільних множників, крім 1);

д) визначити d, модульну мультиплікативну обернену до e ($mod\ \phi(n)$). Іншими словами, розв'яжіть d за рівнянням: $e * d \equiv 1\ (mod\ \phi(n))$;

е) відкритий ключ – (n, e), закритий – d.

Щоб зашифрувати повідомлення за допомогою відкритого ключа RSA (n, e):

а) представити повідомлення у вигляді цілого числа m, де $0 < m < n$;

б) обчисліть $c \equiv m\text{^}e\ (mod\ n)$;

в) надіслати зашифроване повідомлення c.

Щоб розшифрувати повідомлення за допомогою закритого ключа RSA d:

а) отримати зашифроване повідомлення c;

б) обчислити $m \equiv c\text{^}d\ (mod\ n)$;

в) отримати вихідне повідомлення з m.

Формули шифрування та дешифрування RSA можна записати так:

– шифрування: $c \equiv m\text{^}e\ (mod\ n)$;

– розшифровка: $m \equiv c\text{^}d\ (mod\ n)$

1.5 Nearby Share

Платформа Nearby – це потужний інструмент для зв’язку між пристроями та наближення за допомогою різних технологій, таких як Bluetooth, Wi-Fi, IP та аудіо. Однією з його ключових функцій є Nearby Messages API, який доступний як для Android, так і для iOS і дозволяє спілкуватися між платформами. Цей API дозволяє передавати невеликі бінарні дані між пристроями, підключеними до Інтернету, незалежно від того, чи знаходяться вони в одній мережі.

Крім Nearby Messages API, також доступний протокол Nearby Connections. Цей високорівневий протокол діє як незалежний від медіа сокет, що дозволяє пристроям рекламувати, сканувати та підключатися один до одного через будь-який спільний носій. Після підключення обидва пристрої можуть надати спільний доступ до списку всіх підтримуваних носіїв і переключитися на той із найвищою пропускною здатністю для швидкої та безпечної передачі даних.

Однією з ключових переваг Nearby Connections є те, що він працює в одноранговому режимі повністю в автономному режимі, забезпечуючи повністю зашифровані з’єднання з високою пропускною здатністю та малою затримкою. Крім того, для зручності користувачів Nearby Connections API автоматично активує Bluetooth або Wi-Fi, коли це необхідно, відновлюючи пристрій до початкового стану після завершення програми, забезпечуючи безперебійну роботу користувача.

Коли справа доходить до обміну даними, API стає симетричним, що означає, що обидві сторони можуть обмінюватися даними у формі об’єктів Payload, які включають типи BYTES, FILE і STREAM. Для закриття з’єднання API надає команду disconnectFromEndpoint(), яка від’єднується від зазначеної віддаленої кінцевої точки, і команду stopAllEndpoints(), яка від’єднується від усіх підключених кінцевих точок.

Щоб використовувати Google Play Services Nearby SDK, вам потрібен обліковий запис Google, остання версія Android Studio та два пристрої

Android із встановленими службами Google Play. Крім того, потрібне підключення до Інтернету, на відміну від Nearby Connections API, який цього не робить. Зауважте, що коли ви використовуєте SDK, аналітика використання збирається для покращення взаємодії з користувачем, зокрема показники продуктивності, інформація про пристрій та інші дані.

Щоб використовувати Nearby Sharing у Kotlin, треба спочатку включити відповідні залежності у файл build.gradle вашої програми. А також можливо зробити це, додавши такий рядок до своїх залежностей:

implementation'com.google.android.gms:play-services-nearby:21.0.0'

Після додавання залежностей можна використовувати Nearby Sharing API для обміну файлами та посиланнями з пристроями поблизу. API надає кілька методів, які можна використовувати для обміну файлами та посиланнями, наприклад, методи startAdvertising() і startDiscovery().

Ось приклад того, як використовувати Nearby Share API, щоб поділитися файлом із пристроєм поблизу:

```
val shareOptions = ShareOptions.Builder()
    .setStrategy(Strategy.P2P_CLUSTER)
    .build()
nearby.getConnectionsClient(контекст)
  .startAdvertising(
    "Ім'я пристрою",
    "com.example.app",
    connectionLifecycleCallback,
    shareOptions
  )
```

У цьому прикладі створили об’єкт ShareOptions із політикою P2P_CLUSTER, яка вказує на те, що ми хочемо використовувати однорангові кластери для спільного використання файлів. Потім треба використовувати

метод startAdvertising(), щоб почати рекламувати наш пристрій як ціль спільного доступу.

Треба застосовувати startDiscovery(), щоб виявити пристрої поблизу, які обмінюються файлами.

```
val shareOptions = ShareOptions.Builder()
    .setStrategy(Strategy.P2P_CLUSTER)
    .build()
nearby.getConnectionsClient(контекст)
  .startDiscovery(
    "com.example.app",
    endpointDiscoveryCallback,
    shareOptions
  )
```

А також можна використовувати методи stopAdvertising() і stopDiscovery(), щоб зупинити спільний доступ до файлів або виявлення, відповідно.

Коли користувач хоче поділитися файлом, треба використовувати метод startPayload(), щоб розпочати процес спільного доступу, і метод stopPayload(), щоб зупинити його.

```
val payload = Payload.fromFile(файл)
val id = Nearby.getConnectionsClient(context).startPayload(endpointId, payload)
```

Окрім спільного доступу до файлів, також можна використовувати Nearby Share API для обміну посиланнями з пристроями поблизу, використовуючи метод Payload.fromUri() для створення корисного навантаження з URI.

```
val payload = Payload.fromUri(uri)
val id = Nearby.getConnectionsClient(context).startPayload(endpointId, корисне навантаження)
```

Загалом, Nearby Sharing — це потужна функція, яка дозволяє користувачам обмінюватися файлами та посиланнями з пристроями поблизу без підключення до Wi-Fi або Інтернету. Як мобільний розробник Kotlin потрібно використовувати цю функцію для створення інноваційних і корисних програм, які дозволяють користувачам ділитися файлами та посиланнями з пристроями поблизу.

1.6 Сховище даних. DataStore

DataStore – це нове рішення для зберігання даних із сімейства Jetpack, яке було розроблено, щоб забезпечити безпечнішу та надійнішу альтернативу традиційному методу SharedPreferences для зберігання конфіденційних даних користувача на Android. Ця бібліотека зараз знаходиться на стадії альфа-версії та розроблена за допомогою спільних програм Kotlin і Flow API.

Однією з головних переваг використання DataStore є підвищення безпеки. На відміну від SharedPreferences, який зберігає дані у вигляді звичайного тексту, DataStore шифрує дані за допомогою унікального ключа, створеного для кожного пристрою. Завдяки цьому дані захищені від несанкціонованого доступу та доступні лише програмі, яка їх створила. Крім того, DataStore використовує файлову систему зберігання, що ускладнює доступ зловмисників до ваших даних.

DataStore також пропонує два різні підходи до зберігання даних, а саме Preferences DataStore і Proto DataStore.

DataStore Preferences схожий на SharedPreferences тим, що він зберігає пари ключ-значення і не може вказати схему чи надати доступ до правильного типу ключів. Однак він має більш безпечну та надійну реалізацію, ніж SharedPreferences, оскільки використовує ту саму систему шифрування та зберігання файлів, що й DataStore.

З іншого боку, Proto DataStore дозволяє розробникам створювати схеми за допомогою буферів протоколу. Цей підхід дозволяє зберігати дані зі

строгою типізацією, які є швидшими, меншими та менш неоднозначними, ніж інші формати, такі як XML. Крім того, під час зберігання пар ключ-значення будуть доступні лише ключі, а не сам вміст. Схема, визначена в Proto DataStore, забезпечує сильну безпеку типів, роблячи її менш схильною до помилок і більш ефективною з точки зору зберігання та пошуку.

DataStore також пропонує транзакційний API та механізм звітування про помилки, що підвищує загальну надійність зберігання даних. Транзакційний API дозволяє атомарно оновлювати кілька ключів, а механізм звітування про помилки гарантує, що будь-які помилки, які виникають під час зберігання або отримання даних, повідомляються розробнику.

1.7 Firebase

Firebase на базі Google – це комплексна платформа для розробки мобільних додатків, яка стає все більш популярною серед розробників у всьому світі. Він пропонує широкий спектр функцій, таких як керування базами даних, зберігання файлів, хмарні обчислення, аналітика, масштабований хостинг і машинне навчання, які можна використовувати для створення потужних і ефективних веб-додатків, програм для Android та iOS.

Однією з ключових сильних сторін Firebase є його комплексне середовище розробки, яке дозволяє розробникам швидко та легко створювати, тестувати та запускати програми. Крім того, інфраструктура Firebase має високу масштабованість, що означає, що вона може легко обробляти великі обсяги трафіку та даних.

Firebase поставляється з двома базами даних, Cloud Firestore і Realtime Database, які є корисними інструментами для сучасних потреб розробки програм. Cloud Firestore – це база даних NoSQL на основі документів, яка забезпечує легку масштабованість і роботу в автономному режимі, тоді як Realtime Database – це хмарна база даних у реальному часі, яка дозволяє швидко й легко синхронізувати дані на кількох пристроях.

Серед інших переваг використання Firebase:

а) безкоштовний стартовий план;

б) швидший час розробки;

в) на платформі Google;

г) дозволяє розробникам зосередитися на інтерфейсі користувача;

д) немає необхідності в сервері;

е) вбудовані можливості машинного навчання;

ж) можливість генерувати трафік для вашої програми;

з) контроль помилок;

и) високий рівень безпеки.

Хоча Firebase має багато переваг, він також має деякі недоліки. Наприклад, він не є відкритим кодом і залежить від постачальника для оновлень і обслуговування. Крім того, він недоступний у багатьох країнах і пропонує лише бази даних NoSQL. Запити можуть бути повільними, і не всі послуги є безкоштовними в базовому плані. Це недешева платформа, і ціни важко передбачити. Крім того, він працює лише в Google Cloud, не має виділених серверів і не пропонує корпоративної підтримки. Нарешті, API GraphQL недоступні.

Загалом Firebase – це потужна й універсальна платформа для розробки мобільних додатків, яка пропонує багато функцій і переваг. Однак важливо зважити всі «за» і «проти» і визначити, чи підходить він для вашого конкретного проєкту.

Щоб надіслати дані з мобільного додатка Kotlin до бази даних Firebase Cloud Firestore, треба використовувати Firebase Cloud Firestore API. Приклад того, як можна додати дані до колекції Firestore за допомогою Kotlin:

а) спочатку потрібно додати залежність Firebase Cloud Firestore у файл build.gradle програми:

```
implementation'com.google.firebase:firebase-firestore:23.1.1'
```

б) потім потрібно ініціалізувати Firebase у вашій програмі, додавши такий код у метод onCreate():

```
FirebaseApp.initializeApp(це)
```

в) щоб додати дані до колекції, потрібно створити екземпляр класу FirebaseFirestore і викликати метод collection() для доступу до потрібної колекції:

```
val db = FirebaseFirestore.getInstance()
val collectionRef = db.collection("collectionName")
```

г) після створення посилання на колекцію потрібно створити новий документ, викликавши метод add() і передавши карту даних, які потрібно додати:

```
val data = hashMapOf(
  "field1" to "value1",
  "field2" to "value2"
)
collectionRef.add(data)
```

д) а також потрібно використовувати метод set(), щоб оновити або створити новий документ у колекції:

```
val data = hashMapOf(
  "field1" to "value1",
  "field2" to "value2"
)
collectionRef.document("documentId").set(data)
```

1.8 Авторизація

У сфері готельних додатків авторизація користувачів відіграє ключову роль у забезпеченні безпеки та безперебійної взаємодії з користувачем у системі.

Процес реєстрації гостей:

На стійці реєстрації гості надають свої електронні адреси, які згодом збирає персонал готелю. Потім ці електронні адреси вставляються в програму реєстрації гостей разом із відповідною інформацією про розподіл кімнат. Цей початковий крок встановлює важливий зв’язок між електронною поштою гостя та призначеною кімнатою.

Електронна адреса для підтвердження та код:

Після процесу реєстрації гості отримують електронний лист з унікальним кодом. Гість повинен надати цей код персоналу готелю для перевірки. Цей крок забезпечує достовірність і точність реєстрації гостя.

Генерація пароля:

Після успішної перевірки система генерує пароль для гостя. Комбінація електронної пошти гостя та згенерованого пароля служить основним набором облікових даних для подальшої реєстрації та доступу до функцій системи.

Процес входу користувача:

Під час доступу до програми готелю гостям пропонується ввести адресу електронної пошти та пароль, наданий співробітником готелю. Ця інформація безпечно передається на сервер авторизації Firebase, який діє як центральний центр для перевірки облікових даних користувача.

Безпека з Firebase:

Сервер авторизації Firebase використовує розширені заходи безпеки для захисту даних користувачів. Отримавши електронну пошту та пароль, сервер хешує пароль за допомогою галузевих стандартних алгоритмів. Цей процес перетворює пароль у необоротне криптографічне представлення, забезпечуючи конфіденційність інформації користувача.

Обробка помилок і успішний вхід:

У разі невдалого входу, наприклад неправильних облікових даних, система негайно сповіщає користувача відповідним повідомленням про помилку. І навпаки, після успішного входу гості отримують доступ до функцій програми та інтегрованих систем готелю.

Впровадження надійної системи авторизації користувачів у готельному додатку не тільки підвищує безпеку, але й сприяє безперебійній роботі гостей. Використовуючи сервер авторизації Firebase, готелі можуть впевнено захищати дані гостей за допомогою безпечного хешування паролів і механізмів перевірки. Оскільки готелі прагнуть створити безпечне та зручне середовище, авторизація користувачів залишається критично важливим компонентом у забезпеченні цілісності даних і задоволеності гостей.

1.9 Біометрична аутентифікація

Біометричну автентифікацію можна інтегрувати в програму Android за допомогою API BiometricPrompt у Kotlin. Клас BiometricPrompt надає простий і послідовний API для відображення діалогових вікон біометричної автентифікації.

Спочатку потрібно створити екземпляр класу BiometricPrompt.Builder, передавши контекст і BiometricPrompt.AuthenticationCallback для обробки результатів автентифікації. Потім установити будь-які параметри, як-от заголовок і підзаголовок діалогового вікна.

Потім викликати метод Authenticate() екземпляра BiometricPrompt, щоб відобразити для користувача діалогове вікно біометричної автентифікації.

Коли користувач успішно проходить автентифікацію, викликається метод onAuthenticationSucceeded() методу AuthenticationCallback, і далі можна продовжити дію, яка вимагає автентифікації. Якщо автентифікація не вдається, буде викликано onAuthenticationError() або onAuthenticationFailed().

А також слід перевірити біометричні можливості пристрою, перш ніж отримати запит.

Повинно перевірити це, викликавши метод BiometricManager.canAuthenticate().

Приклад:

```
val executor = ContextCompat.getMainExecutor(this)
val biometricManager = BiometricManager.from(this)
val biometricPrompt = BiometricPrompt(this, executor, object : BiometricPrompt.AuthenticationCallback() {
    override fun onAuthenticationError(errorCode: Int, errString: CharSequence) {
        super.onAuthenticationError(errorCode, errString)
    }

        override fun onAuthenticationSucceeded(result: BiometricPrompt.AuthenticationResult) {
        super.onAuthenticationSucceeded(result)
    }

    override fun onAuthenticationFailed() {
        super.onAuthenticationFailed()
    }
})
val promptInfo = BiometricPrompt.PromptInfo.Builder()
        .setTitle("Title")
        .setSubtitle("Subtitle")
        .setDescription("Description")
        .setNegativeButtonText("Cancel")
        .build()
```

```
    if(biometricManager.canAuthenticate()                    ==
BiometricManager.BIOMETRIC_SUCCESS) {
      biometricPrompt.authenticate(promptInfo)
    }
```

1.10 Контроль доступу

Контроль доступу до програми мобільного ключа у вашому готелі може бути реалізований за такими критеріями: гостьовий, персонал, універсальний ключ. Ось як можна визначити контроль доступу для кожної категорії:

Гостьовий доступ.

Доступ до окремої кімнати: кожному гостю надається унікальний цифровий ключ у мобільному додатку, що надає доступ лише до призначеної їм кімнати. Цифровий ключ надійно зберігається та шифрується в додатку.

Доступ на основі тривалості: права доступу цифрового ключа налаштовано відповідно до періоду бронювання гостя. Ключ залишається активним протягом усього періоду їхнього перебування та автоматично закінчується після виїзду.

Доступ персоналу.

Зони з обмеженим доступом: співробітникам призначаються цифрові ключі в мобільному додатку для доступу до певних зон, необхідних для виконання їхніх обов’язків, наприклад, приміщень, зон лише для персоналу або службових входів.

Доступ за часом: залежно від їхніх змін або графіків роботи, цифрові ключі для персоналу можуть мати обмеження за часом, щоб забезпечити доступ лише в дозволені години.

Універсальний ключ доступу.

Функціональність головного ключа: обмежена кількість цифрових ключів, які часто зарезервовані для авторизованих співробітників або

менеджерів, можуть мати універсальні привілеї доступу в мобільній програмі. Ці ключі надають доступ до всіх номерів на території готелю, забезпечуючи вищий рівень контролю доступу.

Контрольоване використання: Універсальні цифрові ключі суворо контролюються та видаються лише довіреним особам, яким потрібен широкий доступ для управління або в екстрених цілях. Існують суворі протоколи для моніторингу та реєстрації використання універсальних ключів, щоб забезпечити підзвітність і запобігти неправильному використанню.

Впровадження контролю доступу в програмі мобільного ключа для вашого готелю дає кілька переваг і підвищує ефективність різними способами:

Покращена безпека: контроль доступу гарантує, що лише авторизовані особи, наприклад гості та персонал, можуть отримати доступ до певних зон. Завдяки цифровим ключам, які надійно зберігаються в мобільному додатку, ризик дублювання фізичної картки-ключа або несанкціонованого доступу значно знижується.

Зручність і гнучкість: гості можуть зручно отримувати доступ до призначених їм кімнат за допомогою своїх мобільних пристроїв, усуваючи потребу носити фізичні ключ-карти. Вони можуть легко розблокувати свої кімнати простим натисканням на екрани своїх мобільних пристроїв. Це позбавляє від клопоту щодо керування та відстеження фізичних карток-ключів.

Управління доступом у режимі реального часу: додаток для мобільного ключа дозволяє безперебійно та миттєво оновлювати привілеї доступу. Наприклад, якщо гість вирішив продовжити своє перебування, дозволи на доступ можна продовжити в режимі реального часу без необхідності фізичної заміни картки-ключа. Подібним чином дозволи на доступ персоналу можна оновлювати або скасовувати за потреби.

Журнал аудиту та моніторинг: додаток для мобільного ключа може забезпечити повний журнал аудиту, реєструючи всі дії доступу. Це дозволяє

контролювати та відстежувати події доступу, що може бути корисним для цілей безпеки, вирішення суперечок або розслідування будь-яких інцидентів.

Операційна ефективність. Завдяки цифровим ключам і контролю доступу, керованим через мобільну програму, персонал готелю може оптимізувати адміністративні завдання, пов’язані з керуванням ключами. Це включає в себе скорочення часу та зусиль, витрачених на розповсюдження, отримання та заміну карток-ключів.

Масштабованість та інтеграція: програму мобільного ключа можна легко інтегрувати з іншими готельними системами, такими як системи управління нерухомістю (PMS) або платформи бронювання. Це дозволяє безперебійно синхронізувати інформацію про гостей, розподіл кімнат і привілеї доступу, підвищуючи загальну ефективність роботи.

Загалом, реалізація контролю доступу за допомогою програми мобільного ключа пропонує покращену безпеку, зручність, керування доступом у реальному часі та ефективність роботи. Це сучасне та ефективне рішення для керування доступом у вашому готелі, покращення досвіду відвідувачів і спрощення адміністративних процесів для персоналу.

2 ІНТЕРФЕЙС ТА ПРИКЛАДИ РОБОТИ СИСТЕМИ ДОСТУПУ ДО ПРИМІЩЕНЬ

У цьому розділі наведено результати розробки програми системи контролю доступу на мобільний пристрій.

Наведені нижче результати засновані на Samsung S10 Lite, Android 13.0, інтерфейсі користувача 5.0. Результати можуть відрізнятися залежно від розміру пристрою. Додаток зберігається у форматі AAB Android App Bundle – це новий так званий формат пакета для додатків Android.

На рис. 2.1 показаний інтерфейс основного екрана користувача з рівнем доступу універсальний ключ доступу, цей рівень доступу більш докладно описаний методології управління доступом універсальний ключ доступу.

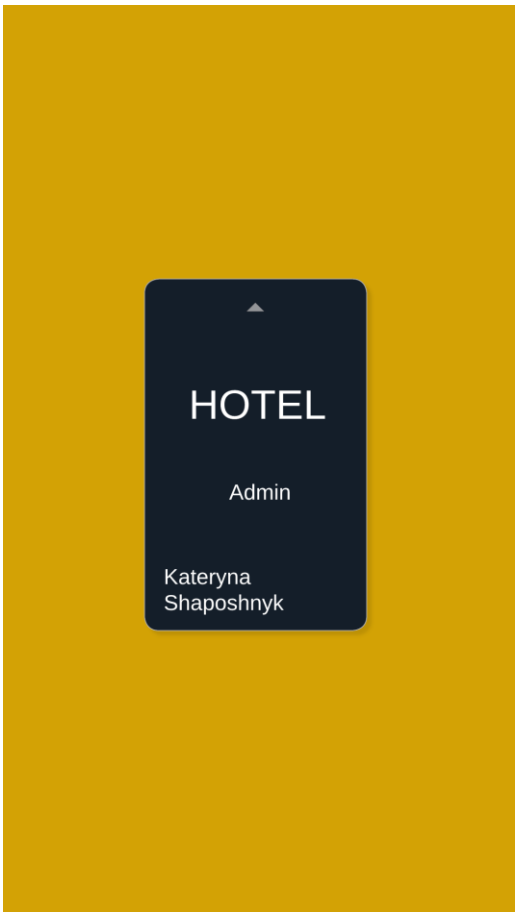

Рисунок 2.1 – Інтерфейс основного екрана користувача з рівнем доступу універсальний ключ доступу

Змінні дані на цій вкладці, лише Ім'я користувача та Прізвище, решта даних статичні і можуть бути змінені тільки кодом програми. Особливість цього рівня доступу в тому, що користувач має універсальний ключ доступу, відправляючи публічний ключ на пристрій "Дверний замок". Він відкриває замок для будь-якої аудиторії, незалежно від того, чи цей користувач знаходиться в базі даних AvailableRooms, тому в сховищі даних програми зберігається лише один головний ключ без певного для нього класу.

На рис. 2.2 показаний інтерфейс основного екрана користувача з рівнем доступу персоналу, цей рівень доступу більш докладно описаний методології управління доступом персоналу. Змінні дані на цій вкладці, лише Ім'я користувача та Прізвище, решта даних статичні і можуть бути змінені тільки кодом програми.

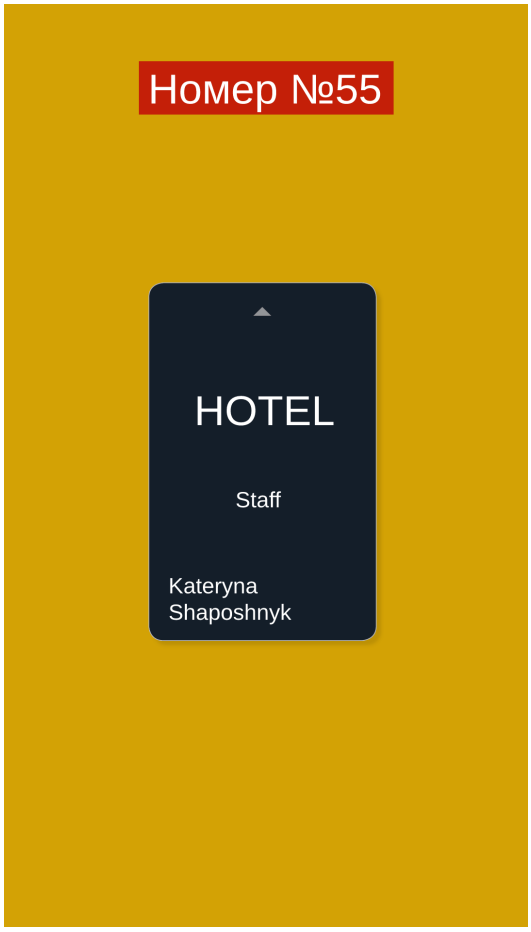

Рисунок 2.2 – Інтерфейс основного екрана користувача з рівнем доступу персоналу

Особливість цього рівня доступу в тому, що користувач має ключ доступу, який після взаємодії пристрою з іншим пристроєм з додатком Дверний замок. Ключ-карта цього користувача може відчиняти лише ті двері, які схвалені для використання адміністраторами готелю, всі доступні номери зберігаються в хмарній базі даних Firebase AvailableRooms, зміну даних можуть робити тільки співробітники готелю, які мають доступ до платформи Firebase.

Користувач може переглянути список доступних номерів при натисканні на текст обраного раніше номера. На рис. 2.3 показаний інтерфейс користувача з відкритим списком доступних номерів, при великій кількості доступних номерів у списку користувач зможе прокручувати цей список вгору і вниз. Цей список завантажується автоматично, коли користувач входить у DataStore і зберігається під ключами «RoomNumber», «RoomKey».

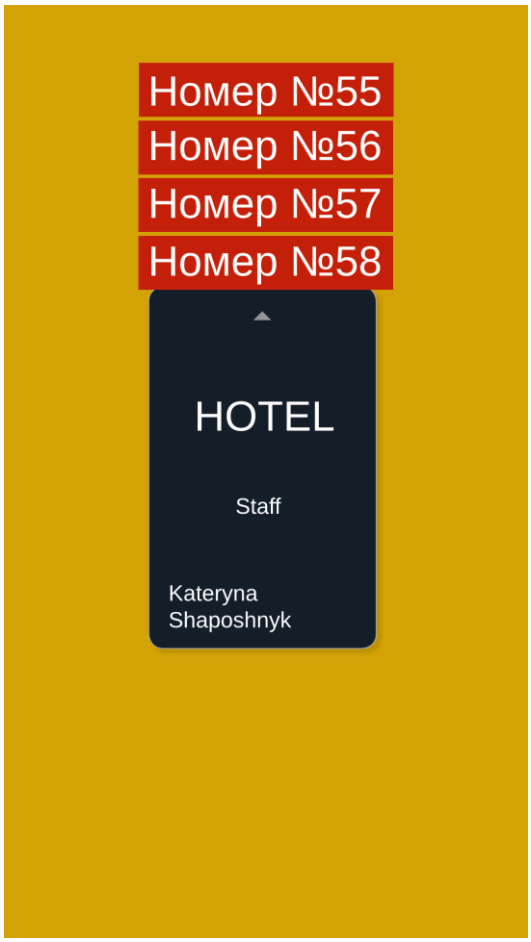

Рисунок 2.3 – Інтерфейс основного екрана користувача з відкритим списком доступних номерів

На рис. 2.4 показаний інтерфейс основного екрана користувача з рівнем доступу гість, цей рівень доступу більш докладно описаний у Методології керування доступом гість. Змінні дані на цій вкладці, тільки Ім'я, Прізвище, решта даних статичні і можуть бути змінені тільки за допомогою коду програми. Особливість цього рівня доступу в тому, що користувач має ключ доступу, який після взаємодії пристрою з іншим пристроєм з додатком Дверний замок. Ключ-карта цього користувача може відчиняти тільки ті двері, які схвалені для використання адміністраторами готелю, всі доступні аудиторії зберігаються у хмарній базі даних Firebase AvailableRoom, зміну даних можуть робити тільки співробітники готелю, які мають доступ до платформи Firebase.

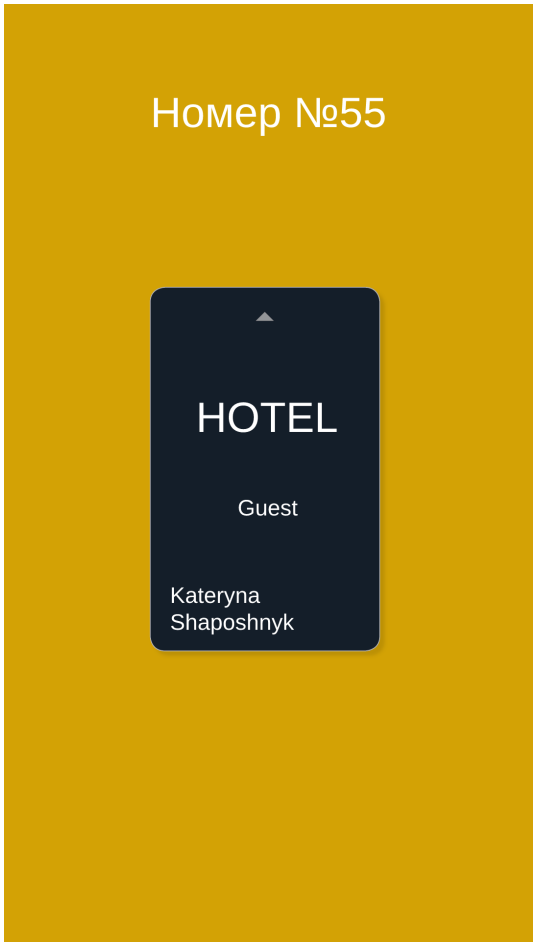

Рисунок 2.4 – Інтерфейс основного екрана користувача з рівнем доступу гість

На рис. 2.5 показаний інтерфейс додатка Дверний замок у стандартному режимі роботи, роботу з цим додатком можна знайти у розділі методології додатка Дверний замок. На цьому екрані у центрі є велике зображення замку. Цей екран призначений для прийому та обробки ключів користувачів, які бажають відчинити двері до певного номера готелю. Приносячи пристрій із додатком «Ключ», вони обмінюються ключами за допомогою технології Nearby Share, ця технологія описана в розділі Методологія Nearby Share, а за допомогою асиметричного шифрування додаток «Ключ» перевіряє ключ користувача, це більш детально описано в додатку «Методологія дверних замків» . Коли користувач зв'язується з програмою «Дверний замок», він може дізнатися про стан дверей з аудиторії на екрані пристрою.

Рисунок 2.5 – Інтерфейс додатка Дверний замок у стандартному режимі роботи

На рис. 2.6 показаний інтерфейс програми, коли користувач успішно відчиняє двері. Фон екрана стане зеленим, а двері залишаться відчиненими протягом п'яти секунд. Через 5 секунд блокування закривається, а програма повертається в стандартний стан і знову відкривається для інших користувачів.

Рисунок 2.6 – Інтерфейс програми, в момент коли користувач успішно відчиняє двері

На рис. 2.7 показано повне представлення таблиці "Користувачі". Категорії цієї таблиці використовуються у додатку «Ключ» не тільки для автентифікації користувача, але й для заповнення інформації про користувача у додатку. Після успішної аутентифікації деякі дані користувача переносяться в локальне сховище на пристрої. Щоб отримати додаткові відомості про сховище даних, див. Методологія сховища даних. Дані UserUID, Email, AccessType, First name, Last name зберігаються в локальному сховищі для подальшого використання в системі додатків, докладніше

обробка та використання даних з таблиці описані в розділі Методологія Ключ програми.

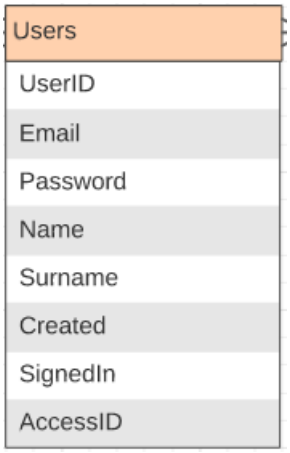

Рисунок 2.7 – Таблиця "Користувачі"

На рис. 2.8 показано схему всіх таблиць хмарної платформи Firebase, які використовуються для зберігання проєктних даних прикладної системи Дверний замок і Ключ.

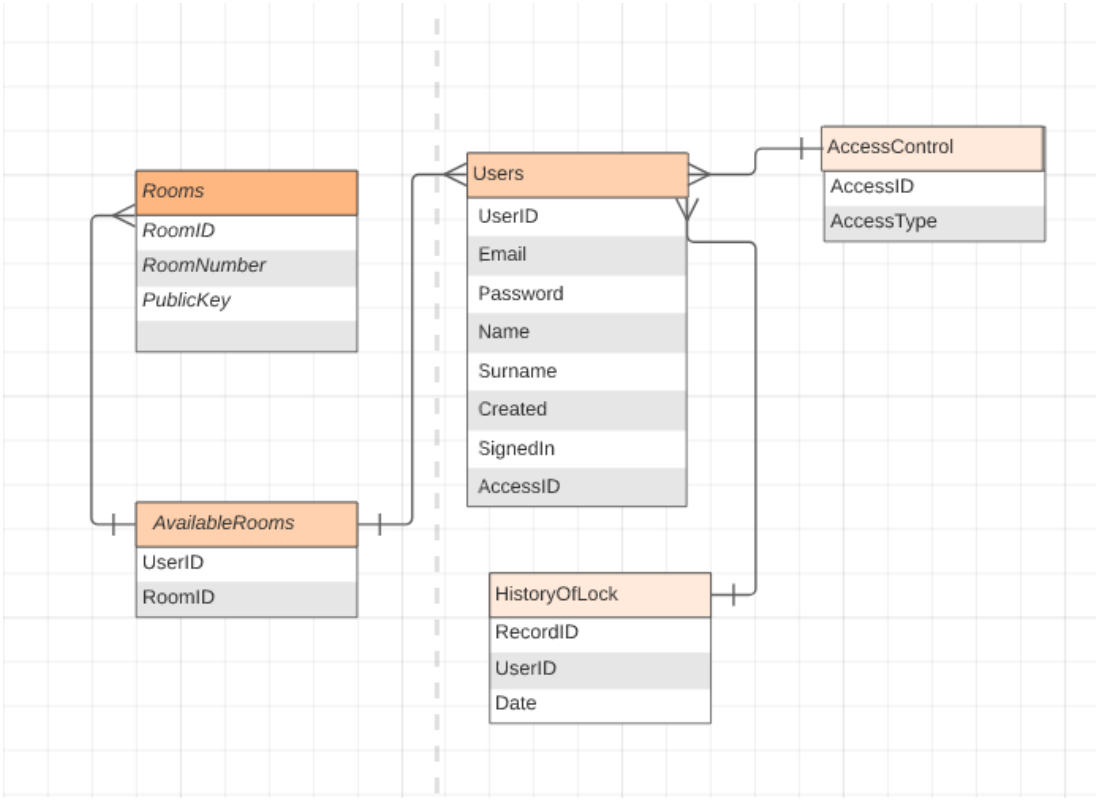

Рисунок 2.8 – Схема таблиць хмарної платформи Firebase для зберігання проєктних даних прикладної системи Дверний замок і Ключ

Додаток «Дверний замок» працює з двома основними базами даних «Доступний клас» та «Історія використання замку», процес зв'язку між хмарними базами даних та додатком більш детально описаний у розділі «Методологія додатка «Дверний замок». Додаток «Ключ» працює з двома основними таблицями баз даних Users і AvailableRoom, процес зв'язку між хмарними базами даних і додатком більш детально описаний в підрозділі методології програми «Ключ».

На рис. 2.9 показано таблицю «Історія використання замку», у якій зберігаються дані, отримані від додатка Дверний замок, всі відкриття конкретної аудиторії користувачами. Ця таблиця містить записи Номер запису, Номер користувача, Дата відкриття аудиторії. Дані таблиці автоматично оновлюються кожні 12 годин, всі нові записи додаються до існуючих, що допомагає стежити за безпекою та контролювати всі інциденти у класі.

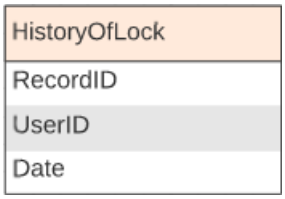

Рисунок 2.9 – Таблиця «Історія використання замку»

На рис. 2.10 показана таблиця Контроль доступу, в якій зберігаються статичні дані для всіх рівнів доступу системи управління доступом організації. Таблиця має лише дві категорії AccessId та AccessType. Рівень доступу в готелі всього три, це гість, співробітник готелю та універсальний, кожен з них має різні характеристики, про які ви можете дізнатися в розділі Методологія контролю доступу. З метою подальшого розвитку системи управління доступом організації співробітника готелю, відповідальні за цю програму, зможуть змінювати та додавати різні категорії користувачів, всі з яких будуть просто додані за допомогою цієї таблиці.

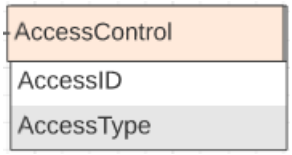

Рисунок 2.10 – Таблиця Контроль доступу

На рис. 2.11 показана таблиця AvailableRoom, в якій зберігаються дані про доступ аудиторії, ця таблиця заснована на розкладі гість та персонал готелю. Таблиця є основною сполучною ланкою між користувачами та одержувачами, вона оголошує, які користувачі мають доступ до якихось кімнат, вона використовує систему «багато до багатьох», один користувач може отримати доступ до багатьох одержувачів, а один одержувач має доступ до багатьох користувачів.

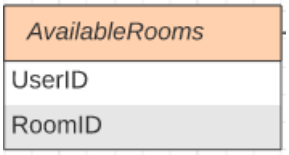

Рисунок 2.11 – Таблиця AvailableRoom

На рис. 2.12 показано таблицю Room, в якій зберігаються дані про номер у готелі. Таблиця містить категорії: RoomId, RoomNumber, PublicKey. Публічний ключ оновлюється кожні 12 годин для кожної шафки з Дверний Замок, більш докладно це описано в розділі методології Дверний замок.

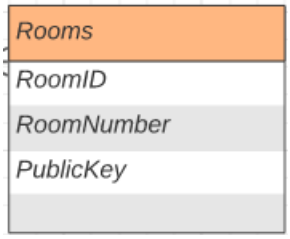

Рисунок 2.12 – Таблиця Room

3 АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ ДОСТУПУ ДО ПРИМІЩЕНЬ

У розробці програми на основі архітектурного шаблону Model-View-ViewModel (MVVM) детально досліджувалась залежності між компонентами програми, структуру та бізнес-логіку прикладних систем, а також взаємодію пристроїв за допомогою сучасних технологій бездротової передачі даних. Протягом усього проєкту я читала кілька книг, щоб отримати уявлення про функціонування Android, потенційні стани пристроїв і програм, обробку помилок і створення винятків. Дослідження бізнес-логіки та структури взаємодії різних сервісів і платформ спиралося на статті з онлайн-ресурсів.

Після ретельного аналізу плану проєкту та конкретних завдань розробки прийнято було рішення використовувати мову програмування Kotlin. У розділі «Методологія Kotlin» окреслено переваги та критерії вибору, за якими я обрала Kotlin як основну мову для розробки програм, а також визнаємо можливість використання Java та C# як альтернативних мов програмування.

Kotlin, створений на основі віртуальної машини Java, забезпечує сумісність із Java, що дає змогу Kotlin повністю розуміти код Java і взаємодіяти з ним. Будучи більш сучасною та стислою мовою, Kotlin добре підходить для написання мобільних і кросплатформних програм. Він також забезпечує покращену ефективність порівняно з Java, фактично слугуючи вдосконаленою версією мови.

Хоча мова програмування C# також підходить для розробки мобільних і кросплатформних програм, Kotlin виділяється як основна мова для розробки програм Android. До 2017 року Java займала позицію основної мови програмування для розробки додатків для Android. Отже, численні бібліотеки та модулі були розроблені спеціально для Java, що зробило Kotlin більш підходящим вибором завдяки його підвищеній безпеці та лаконічності, оскільки він забезпечує сумісність із цими існуючими ресурсами.

Середовищем розробки для нашого додатку слугувала Android Studio версії 2022.1.1.20, що працює на платформі Windows 11. Використання Android Studio спростило процес розробки проєкту завдяки його адаптованій підтримці для розробки пристроїв Android і наявності численних шаблонів для створення компонентів, класів та інших об’єктів, які використовуються в проєкті.

Для розробки інтерфейсу додатків застосовувалась технологію Jetpack Compose, яка наразі є основною технологією для написання коду для розробки додатків. Jetpack Compose спрощує розробку інтерфейсу користувача, дозволяючи працювати з інтерфейсом користувача в одному файлі Kotlin. Це дозволяє створювати окремі елементи інтерфейсу користувача, які можна повторно використовувати в різних частинах програми, полегшуючи написання системного коду та взаємодію між різними елементами інтерфейсу користувача. Крім того, це спрощує доступ до цих об’єктів для отримання та обробки різноманітних обчислень і алгоритмічних результатів, які використовуються в процесі проєктування.

Протягом усього проєкту приділялися значні зусилля вивченню та впровадженню алгоритмів шифрування ключів безпеки, а також на алгоритмі RSA для генерації пари ключів, шифрування та дешифрування. Щоб покращити безпеку та можливості шифрування, використовуються додаткові параметри, надані постачальником Bouncy Castle. Щоб отримати докладнішу інформацію про алгоритми, зверніться до розділу «Методологія асиметричного шифрування».

Щоб забезпечити обмін інформацією між двома пристроями, зокрема програмою «Дверний замок» і програмою «Ключ» досліджувалися різні варіанти бездротової передачі даних між пристроями. Це передбачало аналіз різних підходів, таких як Bluetooth, Nearby Sharing, Wifi-Direct, NFC тощо. Зрештою, наш остаточний вибір складався з трьох технологій: Bluetooth, Nearby Share та Wifi-Direct.

Nearby Share – це бездротова технологія, яка полегшує обмін інформацією між двома пристроями за допомогою Wi-Fi і Bluetooth.

Він був представлений Google у 2019 році і з тих пір став популярним вибором для розробників Android. З іншого боку, Wi-Fi Direct – це однорангова технологія, яка дозволяє двом пристроям спілкуватися один з одним напряму без необхідності використання точки доступу Wi-Fi.

Bluetooth – це широко поширена бездротова технологія, яка дозволяє двом пристроям спілкуватися один з одним на короткій відстані.

Nearby Share є найкращим вибором для надсилання рядкових значень між двома пристроями Kotlin Android на короткій відстані. Він має діапазон 1-2 метри, що є достатнім для цього проєкту та забезпечує швидку передачу даних. Безпека також є головним пріоритетом для Nearby Share, і дані шифруються та перевіряються, щоб забезпечити їх отримання лише призначеним одержувачем. Сумісність також є сильною стороною функції Nearby Sharing, оскільки вона доступна на пристроях Android 6.0 і вище. Додаток Thereforce for Door Lock підтримує пристрої під керуванням Android 6.0 і вище, це дешевше, ніж купувати сучасні пристрої.

Порівнюючи з Wi-Fi Direct і Bluetooth функція Nearby Share є найпростішою у використанні. Завдяки Nearby Share користувачі можуть ділитися даними лише кількома дотиками, не турбуючись про складні процедури налаштування. Навпаки, Wi-Fi Direct і Bluetooth можуть вимагати додаткових кроків для встановлення з’єднання між пристроями, що робить їх менш зручними для користувача.

Щоб розробити програму для Android, яка вимагає передачі даних на короткі відстані, Nearby Share є найкращою технологією для використання. Завдяки швидкій передачі даних, сильним функціям безпеки та простоті використання, він забезпечує надійне та зручне рішення для передачі рядкових значень між двома пристроями Android у Kotlin.

Асиметричне шифрування – це криптографічний метод, який використовує пару математично пов’язаних ключів для шифрування та

дешифрування. Ці два ключі називають відкритим ключем і закритим ключем.

У роботі над готельним додатком для безпечного зв’язку та захисту даних використовувалося асиметричне шифрування. Ось огляд використання асиметричного шифрування:

Генерація ключа: користувач або організація генерує пару ключів – відкритий і закритий ключ. Ключі математично пов’язані, але обчислювально неможливо вивести приватний ключ із відкритого ключа.

Шифрування: щоб надіслати зашифровані дані одержувачу, відправник використовує відкритий ключ одержувача для шифрування даних. Зашифровані дані можна розшифрувати лише за допомогою відповідного закритого ключа одержувача.

Розшифровка: одержувач, який володіє закритим ключем, використовує його для розшифровки отриманих зашифрованих даних. Лише власник закритого ключа може виконати дешифрування, забезпечуючи конфіденційність інформації.

Використання асиметричного шифрування дає кілька переваг, зокрема:

- Безпечний зв'язок: асиметричне шифрування гарантує, що конфіденційна інформація залишається конфіденційною під час передачі.

- Обмін ключами: асиметричне шифрування спрощує обмін симетричними ключами шифрування між сторонами. Відкритими ключами можна вільно ділитися, що забезпечує безпечне спілкування без необхідності попереднього обміну ключами.

Впровадивши асиметричне шифрування в готельну програму, з’явилася можливість встановити безпечні та приватні канали зв’язку, захистити конфіденційні дані та забезпечити автентичність і цілісність інформації, що обмінюється.

ВИСНОВКИ

Підсумовуючи, розробка програми контролю доступу до готелю на мобільних пристроях відкриває величезні можливості для масштабних досліджень. Початкова фаза створення прикладної системи передбачає встановлення конкретних цілей, яких потрібно досягти, а також розгляд потенційних майбутніх удосконалень. Вибір технологій, шаблонів проєктування, бібліотек і мов програмування безпосередньо впливає на продуктивність проєкту, його масштабованість, зручність обслуговування та загальний успіх.

Вибір відповідного архітектурного шаблону має вирішальне значення для ефективної взаємодії між додатками та хмарними даними. Вибраний шаблон визначає структуру та поведінку програми, впливаючи на організацію коду, довгострокову роботу та простоту обслуговування. .

Вибір правильної бездротової технології забезпечує оптимальну продуктивність програми за різних умов, мінімізуючи енергоспоживання та подовжуючи термін служби батареї. Ефективне використання бездротових технологій не тільки покращує роботу користувача, але й приносить значні переваги для бізнесу, такі як оптимізоване функціонування, підвищення продуктивності та зниження витрат.

Використання хмарних баз даних забезпечує гнучкі та масштабовані можливості зберігання та пошуку даних, що дозволяє мобільним програмам обробляти великі обсяги даних у багатьох регіонах.

Таким чином, розробка програми контролю доступу до готелю на мобільних пристроях вимагає прийняття стратегічних рішень щодо архітектурних моделей, технологій бездротового зв’язку та платформ хмарних баз даних. Завдяки ретельному аналізу та вибору найбільш підходящих параметрів для кожного аспекту програма може досягти

оптимальної продуктивності, масштабованості, зручності обслуговування та задоволення користувачів.

Асиметричне шифрування, також відоме як шифрування з відкритим ключем, відіграє вирішальну роль у забезпеченні передачі конфіденційних даних між програмою ключа готелю та сервером.

Застосовуючи асиметричне шифрування, проект досяг покращеної безпеки завдяки використанню пари відкритий-приватний ключ. Цей підхід гарантує, що дані, надіслані з мобільної програми, зашифровані за допомогою відкритого ключа та можуть бути розшифровані лише за допомогою відповідного закритого ключа, який безпечно зберігається на сервері. Цей асиметричний механізм шифрування додає додатковий рівень захисту, зменшуючи ризик несанкціонованого доступу або витоку даних. Шифруючи дані під час передачі, він знижує ризик перехоплення або підробки зловмисниками. Це особливо актуально в контексті програми ключа готелю, оскільки вона передбачає передачу конфіденційної інформації, такої як облікові дані доступу та особисті дані.

Підсумовуючи, інтеграція асиметричного шифрування в програму готельного ключа значно підвищила її безпеку, конфіденційність і загальну надійність.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація Kotlin. URL : https://kotlinlang.org/docs/ (дата звернення: 05.04.2023).

2. Асиметрична криптографія (криптографія з відкритим ключем). URL : https://www.techtarget.com/searchsecurity/definition/asymmetric-cryptography (дата звернення: 25.04.2023)

3. Pointcheval D. Asymmetric Cryptography: Primitives and Protocols. Kentucky, 2022. 304 p.

4. Онацький А. В., Йона Л. Г. Методи асиметричного шифрування. Одеса, 2010. 184 с.

5. Fazio M. Kotlin and Android Development featuring Jetpack. New York, 2021. 549 p.

6. Vivo M. Now in Android№75. URL : https://medium.com (дата звернення: 05.04.2023).

7. Johnson L. Asymmetric Cryptography, Security Controls Evaluation, Testing and Assessment Handbook. Second edition. Upton Pyne, 2019. 788 p.