Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи магістра

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Метод захисту мобільних застосунків»

Виконавець: студент ІІ курсу, групи КБм-21

_____ Олексій ПОЄДИНОК
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Володимир НАКОНЕЧНИЙ	
Нормоконтроль	Олена БОГУСЛАВСЬКА	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Сергій Толюпа
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача(ки) КБм-21 (група) Поєдинка Олексія Миколайовича (прізвище ім'я по-батькові)

Тема дипломної роботи Метод захисту мобільних застосунків

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 3 від 20.10.2022

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес захисту інформації в мобільному застосунку «Gang! - новий сленг молоді».
Предмет досліджень	Аналіз інформаційної захищеності мобільних додатків, до яких пред'являються підвищені вимоги інформаційної безпеки.
Мета	Розробка методів захисту мобільного застосунку
Вихідні дані для проведення роботи	Захищений мобільний застосунок Gang.

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна	удосконалення методу розмежування доступу до даних за допомогою постійної суворої перевірки
Практична цінність	розробка безпечного мобільного застосунку та перевірка коректності прав доступу до інформаційних ресурсів в мобільному застосунку Gang! в процесі функціонування.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Розробка плану для досягнення мети роботи	24.10.2022 – 23.01.2023
Аналіз літературних джерел	24.01.2023 – 14.02.2023
Розробка методів захисту мобільного застосунку	15.02.2023 – 24.04.2023
Оформлення і друк пояснювальної записки	25.04.2023 – 19.05.2023

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект	Зниження збитків через викрадення даних користувачів
Соціальний ефект	Покращення технологій забезпечення захисту інформації у мобільному застосунку як особисто так і на підприємствах.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Володимир НАКОНЕЧНИЙ
(підпис) (Ім’я, ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Олексій ПОЄДИНОК
(підпис) (Ім’я, ПРІЗВИЩЕ)

Дата видачі завдання: 24.10.2022 р.
Термін подання кваліфікаційної роботи до ЕК 19.05.2023 р.

РЕФЕРАТ

Дипломна робота складається зі списку скорочень, вступу, основної частини, що містить 3 розділи, висновків і списку літератури та джерел.

Загальний обсяг роботи – 94 сторінок основного тексту. Список використаних джерел включає 117 джерел.

Метою даної роботи є розробка механізмів захисту мобільних застосунків, зокрема мобільного застосунку «Gang! - новий сленг молоді».

У роботі проаналізована сучачна література з захисту мобільних застосунків, виконаний аналіз основних вразвливостей мобільних застосунків, розроблено безпечний мобільний застосунок з наступними реалізованими механізмами захисту:

- авторизація;
- реєстрація за номером телефону;
- хешування пароля;
- розмежвання доступу;
- прив’язка дій до унікального ідентифікатора користувача.

Ключові слова: розмежування доступу, інформаційна безпека Gang!, політика безпеки, розмежування доступу за ролями, автентифікаці, firebase, kotlin, ios, android, swift.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ABAC	–	Attribute Based Access
Android	–	Операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux.
API	–	Application Programming Interface
BAC	–	Broken Access Control
CRI	–	Container Runtime Interface
CVE	–	Common Vulnerabilities and Exposures
DSL	–	Domain-specific language
DNS	–	Domain Name System
Firebase	–	Платформа розробки мобільних та веб-застосунків
Gang!	–	Мобільний додаток «Gang! – новий сленг молоді»
HTTP	–	Hyper Text Transfer Protocol HTTPS Hyper Text Transfer Protocol Secure
IaaS	–	Infrastructure as a Service
IoT	–	Information of Things
IT	–	Information Technology
JDK	–	Java Development Kit
JSON	–	Javascript Object Notation
Kotlin	–	Статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains
MVP	–	Model-View-Presenter
MVVM	–	Model-View-ViewModel
NIST	–	National Institute of Standards and Technology
ORM	–	Object-relational mapping
(O)TD	–	(OWASP) Threat Dragon
OWASP	–	Open Web Application Security Project
PaaS	–	Platform as a Service
PLEG	–	Pod Lifecycle Event Generator
RBAC	–	Role Based Access
SaaS	–	Software as a Service
SDK	–	Software development kit
SQL	–	Structured query language

Swift	–	Багатопарадигмова компільована мова програмування, розроблена компанією Apple для того, щоб співіснувати з Objective C і бути стійкішою до помилкового коду
UI	–	User interface
UID	–	User Identificator
URL	–	Uniform Resource Locator
VM	–	Virtual Machine
YAML	–	Yet Another Markup Language
ОІД	–	Об’єкт інформаційної діяльності
ПЗ	–	Програмне забезпечення
АС	–	Автоматизована Система
ПРД	–	Правила Розмежування Доступу
ІзОД	–	Інформація з Обмеженим Доступом
СУБД	–	Система управління базами данних

ЗМІСТ

ВСТУП

Актуальність даної роботи визначається тією обставиною, що на даний момент практично кожен користувач мобільних телефонів стикався у своїй роботі з мобільними застосунками.

Кількість мобільних програм у світі швидко зростає. Щомісяця тільки в Google Play з'являється близько 100 тис. нових розробок, а станом на I квартал 2022 там було представлено 3,48 млн додатків для Android. Для порівняння: Apple App Store пропонує 2,22 млн програм для iOS. Кількість користувачів цих продуктів стрімко зростає: ще 2016 р. мобільний трафік у світі перевищив веб-трафік і з того часу зріс на 58% [1].

Кожен великий банк, магазин, авіакомпанія мають свій мобільний додаток для взаємодії з клієнтами – мобільний доступ до товарів і послуг вже давно став звичайним явищем. Однак популярність мобільних додатків, їхнє проникнення в процеси компаній провокують зростання загроз кібербезпеці. Хакери все частіше здійснюють атаки з метою крадіжки особистих даних, інформації про транзакції або влаштовують збій у роботі додатків.

У прагненні швидко створити продукт та залучити користувачів розробники нерідко пишуть небезпечний код, таким чином створюючи вразливості та наражаючи на загрозу свою компанію та її клієнтів.

Кібератаки можуть коштувати бізнесу дуже дорого: вони призводять до прямих фінансових втрат, завдають шкоди репутації компанії, провокують штрафи з боку регуляторів та відтік клієнтів.

Дослідження показують, що 50% мобільних програм мають критичні вразливості [2].

Ризики мобільної безпеки постійно зростають. Компанії використовують досить просунуті інструменти та практики для перевірки своїх веб-додатків на захищеність. Але якщо говорити про мобільне програмне забезпечення, то тут все

обмежується періодичними ручними перевірками. Така ситуація склалася через нестачу якісних інструментів для аналізу безпеки та нестачі компетенцій.

Мобільне ПЗ істотно відрізняється від веб-додатків, і потенційно воно більш уразливе. На відміну від веб-розробок, які запускаються в ізольованому браузері, мобільні програми працюють на пристрої, підключеному до сервера хмар, і безпосередньо взаємодіють з ОС та іншими додатками, а також зберігають системну інформацію на пристрої.

Мобільні рішення відкривають хакерам широкі можливості атак. На сьогоднішній день майже третина додатків містить такі вразливості, як зберігання інформації в небезпечному місці, небезпечна передача інформації, небезпечна авторизація, можливість передачі довільних команд на сервер, проблеми безпеки в бібліотеках з відкритим вихідним кодом.

Безпека мобільних додатків швидко набула важливості, оскільки мобільні пристрої набули поширення в багатьох країнах і регіонах, але механізми захисту є слабкою ланкою мобільного додатка. Більшість вразливостей притаманні фазі проєктування та є результатом недостатньої розробки концепції захисту. Тенденція до більш широкого використання мобільних пристроїв для банківських послуг, покупок і інших дій корелює зі зростанням кількості мобільних пристроїв, додатків і користувачів.

Мобільні пристрої є привабливою мішенню для хакерських атак, оскільки вони зберігають та обробляють велику кількість особистої інформації та даних банківських карток. Недостатня увага приділяється розробці мобільних додатків, а основна проблема пов’язана з небезпечним зберіганням даних. Зберігання інформації про користувача на скріншотах вихідного коду, відкриття ключів і паролів, незахищені дані - це лише деякі недоліки відкриття місця для кібератак.

Користувачі можуть допомогти саботувати свої пристрої: розширити стандартні функції смартфонів, позбавити їх можливостей захисту, відстежувати підозрілі посилання в SMS-повідомленнях, завантажувати програми з неофіційних джерел, тому за безпеку даних користувачів несе відповідальність не тільки розробник додатків, а й власник мобільного пристрою.

Ризик пов'язаний не тільки з уразливими місцями, які захищають клієнта або сервер, але й з уразливими місцями, які виникають під час взаємодії клієнт-сервер. Обмін даними з відкритих протоколів може повністю пошкодити переданий трафік. Однак навіть безпечні зв’язки не завжди надійні, що свідчить про те, що розробники не розуміють глибоко важливості питань безпеки.

Тому темою роботи є методи захисту мобільних застосунків.

Об’єктом дослідження є процес захисту інформації в мобільному застосунку «Gang! - новий сленг молоді».

Предметом досліджень є аналіз інформаційної захищеності мобільних додатків, до яких пред'являються підвищені вимоги інформаційної безпеки.

Методи дослідження — аналіз, синтез, методи імітаційного моделювання.

РОЗДІЛ 1
ПРОЕКТУВАННЯ ТА РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

1.1. Огляд аналогів

Kasta – модний маркетплейс

“Kasta” – найбільший модний маркетплейс одягу, взуття та аксесуарів в Україні. Вже 10 років Kasta відкриває українцям найкращі міжнародні та локальні бренди і доводить власним прикладом, що модний онлайн-шопінг це простіше, швидше й дешевше [3].

Olx – дошка оголошень

«OLX» — платформа онлайн-оголошень, яка об'єднує людей для покупки, продажу або обміну товарами та послугами. Станом на 2018 рік на майданчику зареєстровано 1,5 млн продавців, розміщено понад 11 млн оголошень і кожну хвилину додається близько 100 нових [4].

Оголошення класифікуються за такими категоріями, як «Дитячий світ», «Нерухомість», «Транспорт», «Запчастини для транспорту», «Робота», «Тварини», «Дім і сад», «Електроніка», «Бізнес та послуги», «Мода і стиль», «Хобі, відпочинок і спорт», «Віддам безкоштовно» і «Обмін». OLX є найбільш відвідуваним сервісом оголошень в Україні. Кожен другий інтернет-користувач з України відвідує ресурс мінімум один раз на місяць.

Prom.ua – український торговий майданчик

«Prom.ua» — український торговий майданчик, проект IT-компанії EVO. На цій платформі підприємці самостійно створюють свої власні інтернет-магазини або розміщують свої товари в загальному каталозі. У Промі зібрано понад 100 мільйонів товарів для покупців [5].

AliExpress.com

AliExpress.com — інтернет-магазин малого бізнесу Китаю, який пропонує товари міжнародним покупцям в Інтернеті. Цей магазин належить групі компаній Alibaba. Сайт допомагає малим підприємствам продавати свою продукцію клієнтам

по всьому світу, і, так само як на Amazon або eBay, на AliExpress можна знайти майже все, що тільки бажається продати [6].

Alibaba Group

Alibaba Group — група компаній, які займаються бізнесом в Інтернеті. Alibaba Group є приватною компанією зі штаб-квартирою в Китаї, у місті Ханчжоу. Основною діяльністю групи є торгівельні операції між компаніями за схемою B2B та роздрібна онлайн-торгівля. У 2012 році загальний обсяг торгівлі досяг 1,1 трлн юанів ($170 млрд обсягу продажів на двох порталах групи Alibaba). Компанія діє, насамперед, на території Китайської Народної Республіки (КНР) [7].

Wildberries

Wildberries (Вайлдберіз) – міжнародний інтернет-магазин одягу, взуття, електроніки, дитячих товарів, товарів для дому та інших товарів. Крім РФ, працює в Білорусії, Казахстані, Киргизії і Вірменії. У січні 2020 року компанія почала працювати на ринку ЄС, відкривши продаж в Польщі, з травня 2020 року – в Словаччині, з вересня 2020 року – на Україні, з грудня 2020 року – в Ізраїлі, з січня 2021 року – в Німеччині [8].

Farfetch

Farfetch – це британо-португальська мережа розкішних роздрібних платформ моди, яка продає товари понад 700 бутиків та брендів з усього світу. Компанія була заснована в 2007 році португальським підприємцем Хосе Невесом зі штаб-квартирою в Лондоні та головними філіями в Порто, Гімарайнш, Бразі, Лісабоні, Нью-Йорку, Лос-Анджелесі, Токіо, Шанхаї, Гонконгу, Сан-Паулу та Дубаї [9].

Lamoda

Lamoda – один з найбільших інтернет-магазинів в Росії і СНД . З 2014 року входить до складу Global Fashion Group [10].

З розглянутих додатків для країни Україна актуальними є додатки “Lamoda”, “Kasta” “Olx” та “Prom.ua”.

Огляд аналогів показав, що в першу чергу варто звернути увагу на продавців товарів, які розташовуються на маркетплейсах для продажу покупацям.

Користувачеві має бути зрозуміло, чому в додатку немає жодного доступного закладу, якщо список закладів виявився порожнім, тобто потрібно явно диференціювати поведінку програми при дійсно відсутності доступних закладів, при недоступному сервері додатка і при випадковому одиничному збої в роботі системи.

В додатку, що розробляється повинна бути дозволена робота без підключення до Інтернету з заздалегідь завантаженою інформацією.

Крім того, програма має володіти зручним і інтуїтивно зрозумілим інтерфейсом і дизайном, що забезпечує комфортну роботу з додатком.

1.2. Проектування мобільного застосунку Gang!

1.2.1. Функціональні вимоги

Система повинна задовольняти наступним функціональним вимогам [11]:

- надавати користувачеві можливість регіїстрації і авторизації з кількома пристроями одночасно;
- забезпечувати синхронізацію даних на різних пристроях користувача;
- дозволяти користувачеві переглядати без підключення до Інтернету заздалегідь завантажену інформацію;
- надавати користувачеві можливість змінювати свої реєстраційні дані;
- надавати користувачеві інформацію про магазини;
- забезпечувати користувачу можливість здійснювати пошук товару/магазину за назвою або за тегами.
- надавати користувачу можливість використовувати фільтри для пошуку.
- дозволяти користувачу створювати замовлення (з вказанням дати, часу, товарів, коментарів) і скасовувати їх.
- надавати менеджеру можливість обробки замовлень, що призначені для користувача: приймати або відхиляти замовлення або зв'язуватися з користувачем за вказаним номером телефону.
- дозволяти менеджеру змінювати інформацію про магазин, про товари;

- перевіряти користувача на реального (підтвердження мобільного номеру користувача) ;
- давати можливість змінювати пароль задля безпеки з підтвердженням прав на цю функцію.

1.2.2. Нефункціональні вимоги

В результаті аналізу вимог щодо функціональності та огляду аналогічних проектів, були сформульовані такі наступні вимоги, що не стосуються функціональності:

- програма повинна бути розроблена з використанням мови Kotlin для платформи Android.
- для платформи iOS програма має бути розроблена з використанням мови Swift.
- сервер системи має мати REST-інтерфейс.
- сервер повинен бути сумісним як з платформою iOS, так і з Android.
- сервер повинен бути захищений;
- сервер повинен шифрувати дані користувачів;
- дизайн додатків повинен бути ідентичним.

1.2.3. Проектування архитектури мобільного клієнт-серверного мобільного застосунку

На рисунку 1.1 зображено діаграму варіантів використання мобільного додатку, в якому взаємодіють два основних учасника - менеджер магазину і авторизований користувач. Менеджер магазину - це особа, яка використовує мобільний додаток після пройденої процедури авторизації або реєстрації в якості менеджера магазину. Авторизований користувач - це особа, яка успішно пройшла процес авторизації або реєстрації як покупець.

Рисунок 1.1 - Діаграма варіантів використання мобільного застосунку

Невідома особа може зареєструватися в додатку, вказавши своє ім'я, номер телефону, електронну пошту та іншу відповідну інформацію.

Користувач має можливість авторизуватися в додатку за допомогою номера телефону або електронної пошти. Авторизований або неавторизований користувач може переглядати дані про заклад, такі як адресу, телефон, режим роботи, каталог товарів, геолокацію, фотографії, акції і т.д.

Авторизований або неавторизований користувач може здійснювати пошук товарів у загальному пошуку, отримуючи список речей, знайдених за назвою або тегами у меню різних магазинів. Авторизований користувач може створювати заявки на товари, вказуючи дату, час, кількість, колір, розмір і додаткові коментарі (за бажанням).

Користувач має можливість переглядати свої заявки на бронювання та відстежувати їх статус. Якщо заявка ще не оброблена або вже прийнята, але не завершена, користувач може скасувати заявку.

Авторизований користувач має можливість керувати своїми особистими даними, такими як зміна номера телефону, імені, електронної адреси, пароля і т.д. Схема варіантів використання сервера системи подана на рис. 1.2.

З сервером взаємодіють два учасники: клієнтська програма - додаток, який звернувся до сервера, але ще не ідентифікувався як додаток для користувача або менеджера.

Рисунок 1.2 - Схема варіантів використання сервера системи

Клієнтська програма має можливість авторизуватися або зареєструватися. Якщо клієнт спробує зареєструвати програму як додаток для менеджера, то він отримає повідомлення, що реєстрацію додатків для менеджерів здійснює адміністратор системи. Додаток для користувача може отримувати інформацію про заклади, таку як загальна інформація, акції і каталог товарів і т.д. Додаток для користувача може змінювати особисту інформацію, таку як номер телефону, ім'я, електронну адресу, пароль і т.д. Додаток для користувача може створювати заявки на бронювання товарів в певному закладі.

Додаток для користувача (для менеджера) може також отримувати інформацію про заклади, тобто загальну інформацію, акції і товари закладу, в якому працює менеджер та про інше.

Додаток для користувача (для менеджера) також може отримувати інформацію про заклади, включаючи загальну інформацію, акції та товари, що пропонуються в закладі, де працює менеджер, а також іншу важливу інформацію.

Додаток для користувача (для менеджера) має можливість змінювати дані про магазин, додавати, редагувати та видаляти акції та фотографії закладу, а також редагувати пункти в каталозі.

Обидва додатки можуть змінювати статуси заявок на бронювання. Додаток для менеджера може встановити один з таких статусів: "в процесі", "відхилена", "завершена" або "товару не має".

Обидва додатки мають можливість переглядати історію бронювань. Додаток для користувача отримує історію бронювань користувача, тоді як додаток для менеджера отримує історію бронювань закладу. Диспетчер URL виконує прив'язку URL-адреси, за якою клієнтська програма звернулася до сервера, до відповідного обробника запитів додатка для користувача або менеджера.

Обробники запитів додатка для користувача та додатка для менеджера містять логіку збору, перетворення та компонування даних, які надаються або запитуються відповідними додатками. Сервіс авторизації - компонент, що здійснює процес прийняття рішень про те, чи має клієнтська програма право на виконання запитаних дій.

Компонент, що забезпечує відправку СМС-повідомлень, формує текст СМС-повідомлень та містить налаштування для використання стороннього сервісу СМС-розсилок. Компонент, відповідальний за розсилку електронних листів, генерує текст електронних листів і містить конфігурацію для використання зовнішнього сервісу електронної пошти.

Компонент, що відповідає за відправку звітів про помилки, створює запис у файлі логів у випадку некоректної поведінки системи, формує текст звіту про помилку і за допомогою компонента для розсилки електронних листів надсилає сформований звіт розробникам.

Менеджер взаємодії з базою даних відповідає за взаємодію з базою даних, а також містить конфігураційні дані для з'єднання з базою даних. На рис. 1.3 представлені деякі схеми бази даних сервера системи.

База даних складається з наступних таблиць:

- Item – таблиця, що зберігає дані про товари закладів.
- User – таблиця, що зберігає дані про користувачів додатків.
- Shop – таблиця, в якій містяться дані про магазини.
- Annotation – таблиця-перерахування, що зберігає можливі точки розташування магазинів на мапі.
- Message – таблиця, що зберігає пункти повідомлень між магазином й користувачем.
- news – таблиця для зберігання даних про новини.

Компонент закладів надає інформацію про заклади і відповідає за пошук закладів за назвою або типом.

Компонент бронювань дозволяє отримувати історію бронювань, створювати нові заявки на бронювання, змінювати статус незавершених заявок.

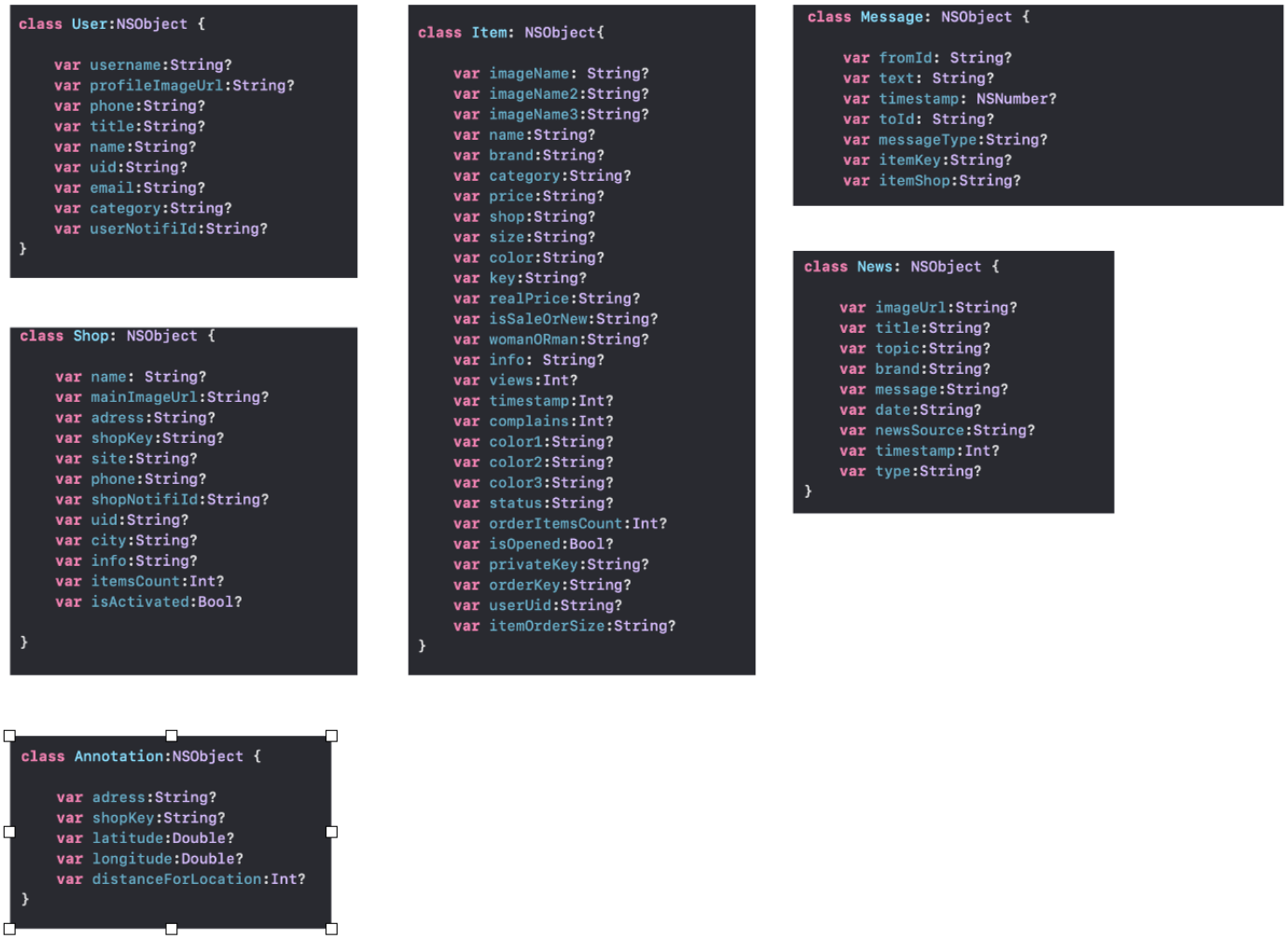


Рисунок 1.3 - Схеми бази даних сервера системи

Компонент, що відповідає за інформацію про користувача, дозволяє переглядати і змінювати реєстраційні дані користувача.Компонент меню надає меню закладу з розподілом за категоріями і також забезпечує можливість пошуку бажаного товару за його назвою або тегом в каталозі всіх закладів.

Компонент автентифікації визначає необхідність проведення процедури автентифікації і надає можливість реєстрації або авторизації, якщо це необхідно.

Компонент оновлення даних оновлює дані в базі даних програми, якщо пристрій користувача авторизовано, сервер системи доступний і пройшов певний час з моменту останнього оновлення даних.

Клієнт HTTP забезпечує взаємодію з сервером системи шляхом використання REST-інтерфейсу сервера.

Менеджер з'єднання з базою даних містить конфігураційні дані для підключення до бази даних, скрипт ініціалізації бази даних, а також скрипт створення схеми бази даних.

Сервіс отримання повідомлень FCM обробляє та повідомляє користувача про отримані повідомлення. Firebase Cloud Messaging – це безкоштовний сервіс для надсилання повідомлень з серверів до додатків для пристроїв Android, iOS і Chrome. У системі «Gang!» FCM використовується для прискорення обміну інформацією щодо заявок на бронювання.

1.3. Реалізація системи

Серверна частина системи реалізована мовами програмування Swift/Kotlin з використанням веб-фреймворку Realtime Database. Swift/Kotlin підтримують кілька підходів до програмування, зокрема структурні, об'єктно-орієнтовані та функціональні. Realtime Database надає базу даних та бекенд у режимі реального часу як сервіс. Цей сервіс надає розробникам додатків API, який дозволяє синхронізувати дані між користувачами та зберігати їх у хмарному середовищі Firebase.

REST API використовує подійний протокол для забезпечення HTTP-з'єднань та отримання push-повідомлень від сервера. Розробники, що використовують Realtime Database, можуть захищати свої дані за допомогою правил безпеки, які застосовуються на сервері. Дані зберігаються у базі даних, створеній за допомогою Firebase Database, оскільки вона поширюється та інтегрується безкоштовно [12].

Клієнтська складова системи розроблена для операційних систем Android та IOS, які є найпопулярнішими мобільними платформами у світі, з використанням мов Kotlin та Swift, що належать до об'єктно-орієнтованих мов програмування. Для зберігання даних локально використовується база даних, створена з використанням Системи управління базами даних без схеми (NoSQL), підтримка якої вбудована в платформи Android та IOS.

Взаємодія між клієнтським додатком і сервером здійснюється за допомогою виконання HTTP-запитів з програми до сервера та отримання FCM-повідомлень від сервера. Робота з базою даних виконується за допомогою платформи Firebase, де використовуються запити без схеми бази даних типу NoSQL для створення і зміни таблиць.

REST API використовує протокол подій із сервером, який є інтерфейсом для створення HTTP-з'єднань з метою отримання push-повідомлень від сервера.

Під час розробки мобільного додатка було прийнято рішення відмовитися від використання ORM-бібліотек на користь використання класу-помічника СУБД SQLite та SQL-виразів. Це рішення було обумовлено тим, що додаток має підтримувати широкий спектр версій платформи Android, а деякі ORM-бібліотеки нестабільно працюють на платформах з версією API, що менше за 21.

Крім цього, зберігання та читання даних відбуваються швидше без використання об'єктно-реляційного відображення даних (ORM) порівняно з будь-яким ORM рішенням.

У реалізованому додатку одним із найчастіших операцій з базою даних є збереження значного обсягу даних, отриманих від сервера системи. Правила безпеки в режимі реального часу Firebase контролюють, хто має доступ для читання та запису в базу даних, як дані структуровані і які обмеження існують. Ці правила існують на серверах Firebase та автоматично застосовуються у будь-який момент.

Кожен запит на читання та запис буде виконаний, якщо правила цього дозволяють. За замовчуванням правила не надають доступ до бази даних нікому.

Це зроблено для захисту бази даних від неправомірного використання, доки ви не налаштуєте власні правила або аутентифікацію. Правила безпеки в режимі реального часу використовують синтаксис, схожий на JavaScript. Приклад такого синтаксису наведено на рис. 1.4:

```
{
 “rules”: {
  “users”: {
   “$user_id”: {
    “.write”: “$user_id === auth.uid”,
    «.read»: true
   }
  }
 }
}
```

Рисунок 1.4 - Правила безпеки в режимі реального часу

Робота у реальному часі

Замість стандартних HTTP-запитів база даних Firebase Realtime використовує механізм синхронізації даних – кожного разу, коли дані змінюються, будь-який підключений пристрій отримує це оновлення майже миттєво.

Офлайн режим

Додатки Firebase залишаються активними навіть у відсутності з'єднання з мережею, оскільки SDK бази даних Firebase Realtime зберігає дані на пристрої. Після відновлення з'єднання клієнтський пристрій автоматично синхронізує будь-які зміни з поточним станом сервера.

Доступність зі засобів клієнта

Доступ до бази даних Firebase Realtime можна отримати безпосередньо з мобільного пристрою або веб-браузера, без необхідності використання сервера додатків. Безпека та перевірка даних забезпечуються за допомогою правил безпеки баз даних Firebase Realtime, що базуються на правилах виразів, які виконуються під час читання або запису даних.

Масштабування в кількох базах даних

За допомогою Firebase Realtime Database ви можете масштабувати потреби свого додатка в даних, обмінюючись даними між кількома екземплярами бази даних в одному проекті Firebase. Є можливість оптимізувати автентифікацію для проекту за допомогою Firebase Authentication і перевіряти автентичність користувача у всіх екземплярах бази даних. Використовуйте власні правила Firebase Realtime Database для кожного екземпляра бази даних, щоб контролювати доступ до даних у кожній базі даних.

За замовчуванням для операцій запису, які викликають події, увімкнено сувору перевірку. Будь-яка операція запису, яка запускає понад 1000 хмарних функцій, або одна подія, розмір якої перевищує 1 МБ, завершиться помилкою з повідомленням про те, що обмеження досягнуто. Це означає, що деякі хмарні функції взагалі не працюватимуть, якщо не пройдуть попередню перевірку.

Правила безпеки

Правила бази даних у реальному часі Firebase визначають, хто має доступ на читання та запис до бази даних, як структуровані дані та які існують індекси. Ці правила розміщені на серверах Firebase і автоматично застосовуються в будь-який час. Кожен запит на читання та запис виконується, якщо це дозволено правилами.

За замовчуванням правила забороняють будь-кому отримати доступ до бази даних. Це потрібно для захисту бази даних від зловживань, поки не буде часу для налаштування правил або автентифікації.

Правила бази даних реального часу мають синтаксис, схожий на JavaScript, із чотирма типами [13].

Види правил бази даних реального часу

- Опція "Read" вказує, чи мають користувачі дозвіл на читання даних.
- Опція "Write" визначає можливість та момент запису даних.
- Опція "Validate" встановлює правила для правильного форматування значення, наявності дочірніх атрибутів та типу даних.
- Опція "IndexOn" визначає дочірній елемент для створення індексу або підтримки сортування та запитів.

База даних Firebase Realtime надає широкий набір інструментів для управління безпекою додатка. Ці інструменти спрощують процес аутентифікації користувачів, забезпечують контроль над правами користувачів та перевірку вхідних даних. Додатки, побудовані на основі Firebase, виконують більше клієнтського коду порівняно з додатками, розробленими за допомогою інших технологій [14].

Аутентифікація

Поширеним першим кроком у захисті програми є перевірка ідентичності користувачів. Цей процес відомий як аутентифікація. Для цього можна використовувати Firebase Authentication, щоб користувачі могли ввійти до програми. Аутентифікація в Firebase включає підтримку різних методів перевірки, таких як Google і Facebook, а також вхід за допомогою електронної пошти та паролю, анонімний вхід та багато інших.

Ідентифікація користувача

Визначення особи користувача є важливою складовою безпеки. Різні користувачі мають різні дані і часом різні можливості. Наприклад, у чат-додатку кожне повідомлення пов'язане з користувачем, який його створив. Крім того, користувачі можуть видаляти свої повідомлення, але не повідомлення інших користувачів.

Авторизація

Ідентифікація користувача – це тільки частина безпеки. Як тільки буде відомо, хто вони, знадобиться спосіб контролювати їх доступ до даних у базі даних.

Правила бази даних в реальному часі дозволяють контролювати доступ для кожного користувача. Такий приклад наведено на рис. 1.5, тобто набір правил безпеки, який дозволяє будь-якому читати шлях / foo /, але ніхто не може писати в нього:

```
{ “rules” : { “foo” : { “.read” : true , “.write” : false } } }
```

Рисунок 1.5 - Набір правил безпеки

Правила бази даних в режимі реального часу містять вбудовані змінні та функції, що дозволяють використовувати посилання на інші шляхи, тимчасові мітки на стороні сервера, інформацію про автентифікацію та інші корисні можливості. На малюнку 1.6 представлено приклад правила, яке надає можливість запису для автентифікованих користувачів за шляхом /users/<uid>/ , де <uid> - це ідентифікатор користувача, отриманий за допомогою Firebase Authentication.

```
{ «rules» : { «users» : { «$ uid» : { «.write» : «$ uid === auth.uid» } } } }
```

Рисунок 1.6 - Приклад надання доступу на запис

Перевірка даних

База даних Firebase Realtime не є реляційною. Це дозволяє легко щось змінити

але коли програма буде готова до широкого розповсюдження, важливо, щоб дані залишалися послідовними. Мова правил містить правило .validate, яке дозволяє застосовувати

Логіка перевірки з використанням тих самих виразів, що й правила .read і .write. Єдина відмінність полягає в тому, що правила перевірки не є каскадними, тому всі відповідні правила перевірки мають бути істинними для надання дозволу

Задокументуйте ці правила на схемі. 1.7 Доказ того, що записувати дані в /foo/ має бути

Рядки коротші за 100 символів:

Правила перевірки мають доступ до тих самих вбудованих функцій і змінних, що й правила .read і .write. Їх можна використовувати для створення правил

Перевірки будуть знати про дані в інших місцях бази даних, особу користувача, час сервера тощо.

```
{
  “rules”: {
```

```
  “foo”: {
    “.validate”: “newData.isString () && newData.val (). Length <100”
   }
 }
}
```

Рисунок 1.7 - Приклад правила, яке гарантує, що довжина параметра foo не перевищує 100 символів

Встановлення пароля

Усі взаємодії між клієнтом автентифікації Firebase і внутрішнім сервером

Відбувається через наскрізне зашифроване з’єднання HTTPS. Таким чином, хоча пароль може відображатися у вигляді звичайного тексту в коді, він може ні

Будь-хто, хто перевіряє мережевий трафік, може його переглянути. Для демонстрації доступна проста утиліта шифрування на основі пароля

бібліотека сценаріїв. Можна назвати script {key} {salt} {rounds} {memcost} [-P]. Утиліта запитає простий текстовий пароль і відобразить хеш у разі успіху. це

Хеш має бути закодований BASE64 і порівнюватися з хешем пароля

Експортовані облікові записи користувачів. {Key} – ключ для підпису хеш-параметрів пароля проекту. Цей ключ повинен

Стверджуйте за допомогою base64 перед передачею в утиліту. {Salt} – комбінація пароля та роздільника для експортованого облікового запису

Сіль із хеш-параметрів пароля проекту. Кожна половина має бути декодована за базою 64 перед конкатенацією.

{Rounds} – параметр раундів із хеш-параметра пароля проекту. {Memcost} – параметр mem_cost із хеш-параметра пароля проекту.

[-P] - необов'язковий -P також може бути наданий, щоб дозволити читання

Простий текстовий пароль із STDIN.

Надіслати повідомлення користувачеві

При реєстрації користувач вказує номер телефону та електронну адресу. На вказаний номер телефону надсилається згенерований сервером код авторизації – рядок символів, переважно цифр. При авторизації вже зареєстрованого користувача в додатку процедура дещо інша: замість надсилання коду авторизації користувач входить у свій обліковий запис за допомогою адреси електронної пошти та пароля.

Розсилка СМС та електронною поштою здійснюється за допомогою сторонніх сервісів OneSignal і Firebase.

Коли менеджер змінює стан програми, користувачеві надсилається FCM-повідомлення, яке передає такі дані: id програми, стан якої змінилося, та повідомлення.

Якщо під час надсилання будь-якого типу повідомлення виникає помилка, сервер створює відповідний запис у файлі журналу.

Підтримка різних версій платформ IOS і Android

У кожному проекті необхідно вказати мінімальні версії платформ Android та IOS, які підтримує додаток. Менші версії дозволяють установлювати програму на більшій кількості пристроїв, але не можуть або обмежують доступ до певних функцій API пізніших версій платформи.

Офіційний сайт платформи Android показує, що на початку 2019 року пристрої з Android 4.1 і вище становили 99,8%. Тому було вирішено вибрати версію 4.1 (Jelly Bean, API 16) як мінімальну підтримувану версію.

Щоб програма коректно працювала на пристроях з різними версіями API, були розроблені різні версії компонентів інтерфейсу.

Реалізація інтерфейсу користувача

Дизайн програми розроблено відповідно до принципів Material Design. Принципи матеріального дизайну представляють комплексну концепцію створення візуальних, мобільних та інтерактивних елементів для різних платформ і пристроїв Google Inc. [17].

Скріншоти користувачів інтерфейсу представлені в Додатку А.

Для переходу між основними екранами програми використовується шаблон Navigation Drawer — меню розташоване в нижній частині екрана та містить основні точки навігації програми (як в IOS).

Деякі елементи матеріального дизайну, як-от приклад на рис. 1.8, певні анімації та ряд атрибутів XML доступні лише на Android із рівнем API 21 або вище.

Для забезпечення сумісності додатків з пристроями, що працюють під управлінням більш ранніх версій платформи Android, створюються різні файли ресурсів для платформ з різними рівнями API (в каталогах values /, drawable / і т.д.).

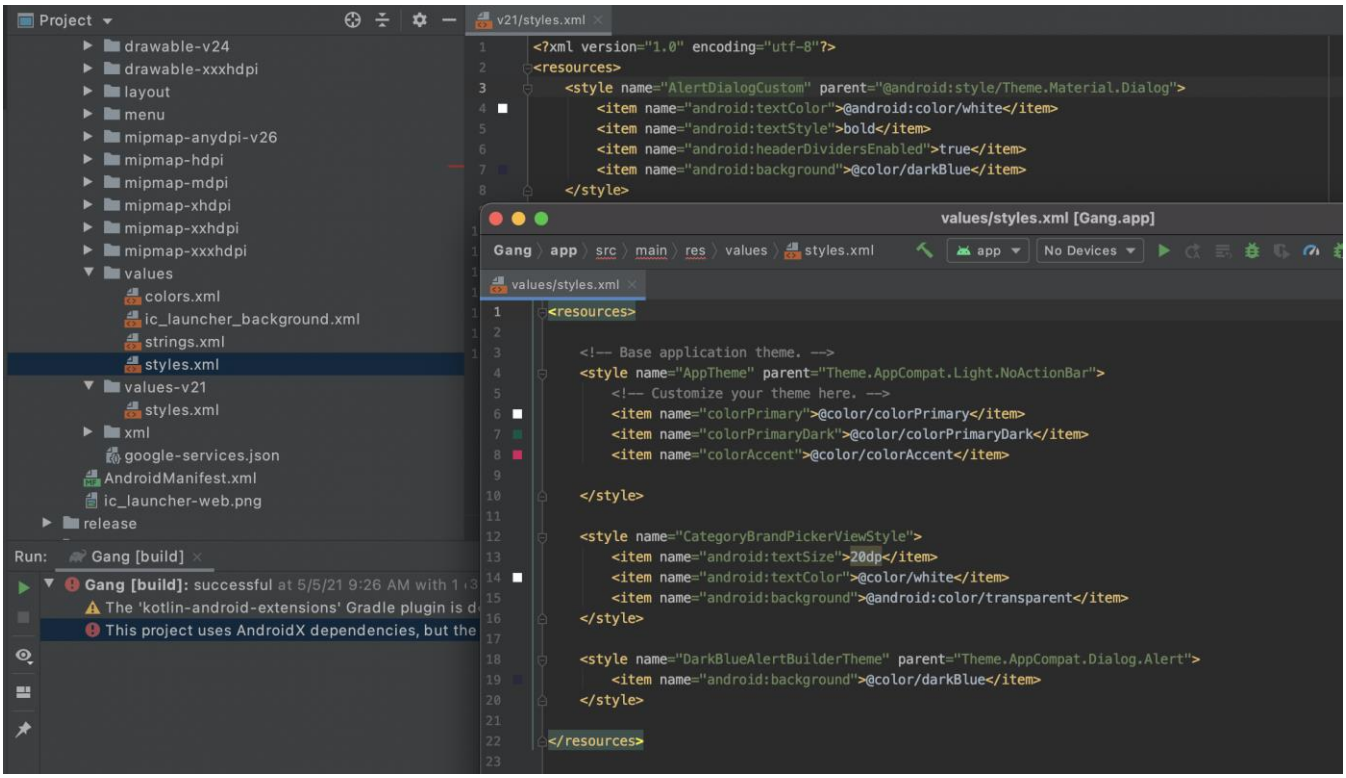


Рисунок 1.8 - Деякі можливості Material Design

Давайте розглянемо раніше згаданий підхід на прикладі визначення конкретного стилю raisedButton, який використовується у додатку для оформлення елементів керування. На малюнку 1.9 показано визначення стилю для платформи Android з версією API вище 21.

Для визначення типу зміни зовнішнього вигляду елемента при взаємодії користувача з ним використовуються ресурси, які представлені на малюнках 1.10 і 1.11 відповідно для вищеописаних стильових файлів.

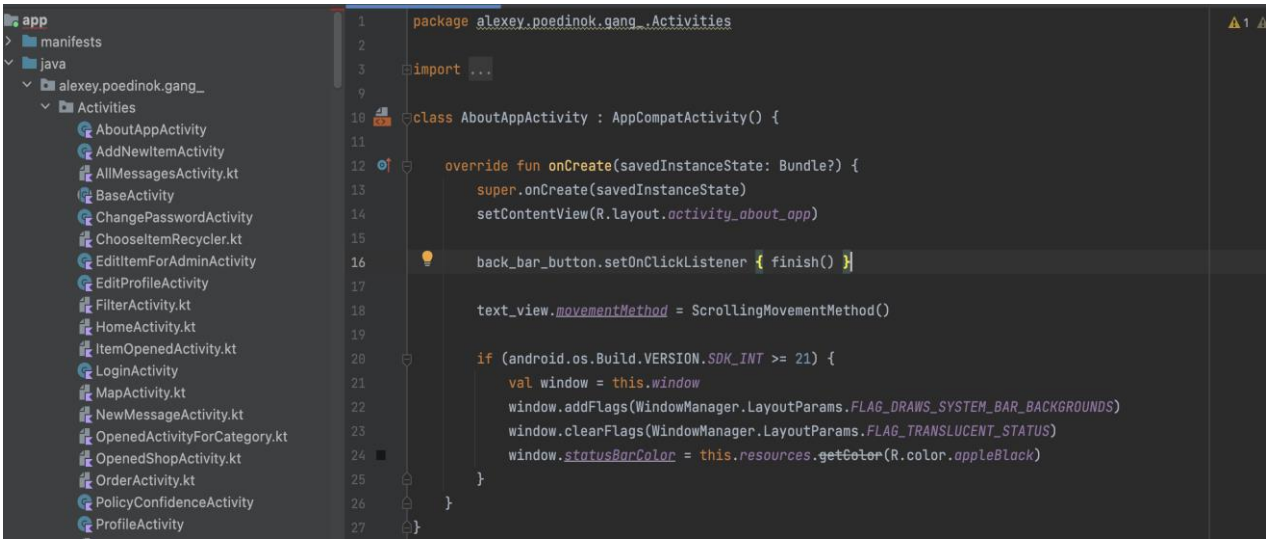


Рисунок 1.9 - Визначення стилю для API вище 21

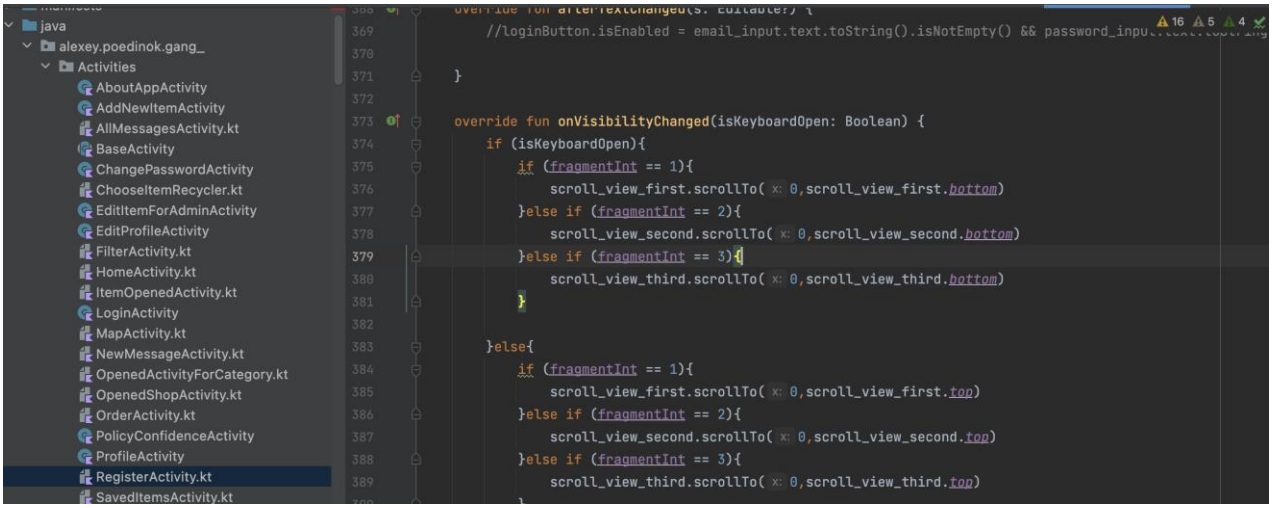


Рисунок 1.10 - Зміна зовнішнього вигляду

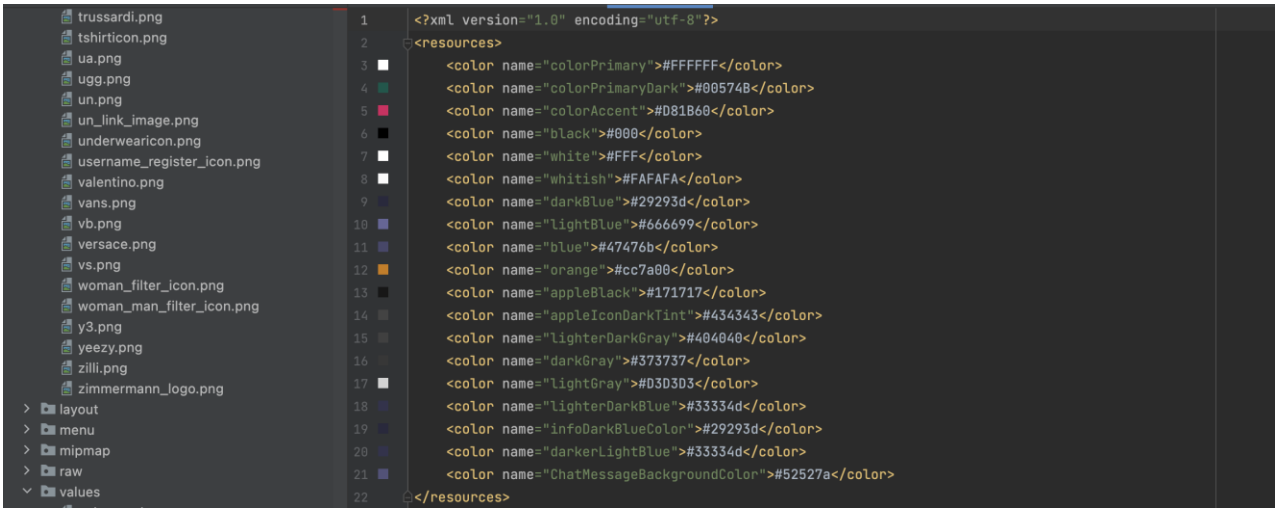


Рисунок 1.11 - Зміна зовнішнього вигляду Colors

Прийом повідомлень від Firebase Cloud Messaging (FCM) та OneSignal

Коли на сервер системи надходить інформація про зміну статусу заявки менеджером, з використанням сервісу OneSignal або FCM надсилається повідомлення на всі пристрої клієнта, що містить інформацію про новий статус заявки. Додаток відображає отримане push-повідомлення на екрані.

Повідомлення можуть бути надіслані у різних випадках. Повідомлення буде відправлено, якщо:

Змінено статус заявки.

Отримано повідомлення від магазину.

Повідомлення відправлено програмно задля сповіщення користувачів про новини, зміни або оновлення.

Виконується верифікація номера телефону.

Коли користувач здійснює дію зі своїми заявками (створює нову або скасовує існуючу), додаток надсилає дані про створену або змінену заявку на сервер. Сервер зберігає отримані дані у базі даних і надсилає OneSignal-повідомлення з інформацією про заявку до додатка менеджера [18].

У додатку менеджера відбувається обробка заявки, і якщо під час обробки статус заявки зазнав змін, додаток передає дані про заявку на сервер системи. Сервер зберігає ці дані і надсилає OneSignal-повідомлення з інформацією про заявку до

додатка користувача. Додаток користувача відображає push-повідомлення, яке містить інформацію про зміну статусу заявки.

1.4. Постановка задачі

Впровадження інформаційних технологій у сферу товарного бізнесу є необхідним для розв'язання наступного спектру завдань [22]:

- аналіз уразливостей мобільних додатків;

- аналіз методів захисту мобільних додатків;

- розробка системи безпеки для особистих даних користувачів мобільного додатку Gang!;

- розробка механізму контролю доступу до інформації в мобільному додатку Gang!;

- реалізація безпечного мобільного додатку Gang!

Участь магазинів у таких проектах має як великий недолік, так і значний ризик. Недолік полягає у необхідності витрачати частину робочого часу співробітників на підтримку актуальності наданої інформації, тобто своєчасне внесення змін через інтерфейс адміністратора магазину або повідомлення про них відділу технічної підтримки проекту. Крім того, потрібно слідкувати за новими джерелами заявок на бронювання товарів. Також недоліком є те, що в магазинах з'являється певний рівень конкуренції, оскільки всі магазини повинні публікувати однакові ціни на товари. У той же час маркетингові дослідження свідчать, що надання інформації про магазин та його товари з офіційних джерел, а також зовнішніх сервісів, має значний вплив на споживача, що підвищує конкурентоспроможність закладу.

Для забезпечення взаємодії між користувачем, який має намір зарезервувати товар, та менеджером відповідного закладу, система Gang! має включати додатки для клієнтів на різних платформах, сервер системи та додаток для менеджерів. У рамках даної роботи будуть розроблені сервер системи та мобільний додаток для платформ Android та iOS.

Неодмінною складовою процесу розробки інформаційних систем, що взаємодіють з критичними ресурсами, є етап аналізу захищеності таких систем. Обгрунтованість дослідження інформаційних систем з погляду безпеки інформаційного середовища обумовлена ризиками часткового або повного порушення цілісності системи та конфіденційності даних через наявність програмних вразливостей у компонентах таких систем.

Безліч компонентів інформаційних систем, що взаємодіють з критичними ресурсами, включають програмні рішення для мобільних платформ. Актуальність таких рішень обумовлена значним розвитком мобільних екосистем і, разом з потребою в аудиті інформаційної безпеки як етапу розробки захищених систем обробки даних, підкреслює необхідність дослідження мобільних додатків у контексті інформаційної безпеки.

Предметом дослідження даної роботи є аналіз інформаційної захищеності мобільних додатків, до яких пред'являються підвищені вимоги інформаційної безпеки.

Висновки за розділом 1

Була створена архітектура системи під назвою "Gang!". Крім цього, були розроблені бази даних для сервера системи і мобільного додатка під платформи iOS/Android.

На основі розробленої архітектури були реалізовані сервер системи "Gang!" і мобільний додаток для платформ iOS та Android. Взаємодію між сервером і мобільним застосуванням забезпечено шляхом використання HTTP-запитів і повідомлень FCM.

Крім того, було розроблено механізм розмежування доступу до інформації, а також здійснено захист на рівні програмного коду.

РОЗДІЛ 2
АНАЛІЗ МЕХАНІЗМІВ ЗАХИСТУ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1. Практики аналізу мобільних застосунків на захищенність

В IT-індустрії було розроблено підходи до забезпечення захищеності мобільних додатків – практики MAST (Mobile Application Security Testing), які стали вирішенням багатьох проблем з безпекою додатків [23].

SAST - статичний аналіз вихідного коду програми. Виявляє небезпечну конфігурацію: шукає токени, ключі шифрування та інші конфіденційні дані, перевіряє коректність конфігурації мережевої взаємодії та ін.

DAST – динамічний аналіз програми. Виявляє незахищений мережевий трафік, точки входу, які можуть бути викликані сторонніми програмами.

API ST – аналіз програми API. Аналіз повідомлень, що пересилаються між додатком і його сервером на предмет наявності чутливої інформації.

IAST – інтерактивний аналіз програми. Моніторинг потоків даних програми, відстеження руху даних від точок входу до потрапляння до потенційно небезпечних функцій.

Регулярне використання практик MAST для аналізу захищеності допоможе забезпечити максимальне охоплення вразливостей мобільних програм.

П'ять базових засад, які допоможуть підвищити безпеку мобільної екосистеми компанії [24].

1. Регулярний автоматизований аналіз мобільних програм на вразливості відповідно до практик MAST. Для цього можна використовувати спеціальні рішення, що дозволяють вбудувати автоматичні перевірки у цикл розробки DevOps.

2. Регулярна перевірка мобільних програм на відповідність індустріальним стандартам інформаційної безпеки ОУД4, OWASP Mobile Top 10, PCI DSS, CWE/SANS Top 25.

3. Періодичні тести на проникнення (penetration tests) для зовнішньої ручної перевірки програм.

4. Регулярні перевірки випущених мобільних додатків для виявлення нещодавно описаних уразливостей, у тому числі у сторонніх компонентах.

5. Розвиток компетенцій команди для створення безпечного коду на ранніх стадіях розробки мобільних додатків. На це спрямовані спеціальні програми навчання розробників.

2.2. Аналіз вразливостей мобільних застосунків

Забезпечення безпеки мобільних додатків – це практика захисту цінних мобільних застосунків і цифрової особистості від шахрайських атак у всіх їх проявах. Це включає втручання, зворотне інжиніринг, шкідливе програмне забезпечення, реєстратори ключів та інші форми маніпуляцій або втручання. Комплексна стратегія забезпечення безпеки мобільних додатків включає технологічні рішення, такі як захист мобільних застосунків, а також передові методи використання і корпоративні процеси.

Забезпечення безпеки мобільних додатків швидко набуло важливості, оскільки мобільні пристрої стали поширеними в багатьох країнах і регіонах. Тенденція до все більшого використання мобільних пристроїв для банківських послуг, покупок і інших дій взаємозв'язана зі зростанням кількості мобільних пристроїв, застосунків і користувачів. Банки зміцнюють свою безпеку, оскільки багато людей почали використовувати мобільні додатки для банківських послуг.

Розробники усвідомлюють важливість безпеки мобільних додатків, але не всі користувачі розуміють це. Крім зростання швидкості поширення мобільного шахрайства, існують й інші причини, з яких фінансові установи мають серйозно ставитися до безпеки мобільних додатків і зобов'язатися розробити комплексну стратегію їх захисту.

Користувачам необхідно бути обережними щодо інформації, яку вони надають і даних, які завантажують під час роботи в Інтернеті. Мобільні пристрої майже завжди

увімкнені, постійно поруч і зберігають значну кількість особистої інформації, конфіденційних даних і документів. Це може стати справжньою золотою жилою для зловмисників.

Мобільні додатки можуть вимагати дозволів, що викликає сумнів. Наприклад, навіщо додатку потрібний доступ до камери або мікрофона? І чи може зловмисник знайти вразливість в цьому додатку, що дозволить йому здобути доступ до камери або мікрофона з метою промислового шпигунства?

Під час аналізу файлу AndroidManifest.xml експерти часто виявляють директиву android: allowBackup зі значенням true. Це дозволяє створювати резервні копії даних програми при підключенні до комп'ютера. Це може стати проблемою, якою зловмисник може скористатися, щоб отримати дані додатка навіть без прав користувача root.

Уразливість небезпечних взаємодій між процесами залежить від проектування інтерфейсів взаємодії компонентів програми і відноситься до помилок у реалізації механізмів захисту. Помилки в механізмах захисту становлять 74% вразливостей додатків для iOS і 57% вразливостей для платформи Android [39].

Очікується, що в майбутньому зростатиме частка мобільних пристроїв, які мають функціональні можливості відкритої платформи. Відкритість цих платформ надає значні можливості для всіх складових мобільної екосистеми, завдяки гнучкості по встановленню, видаленню або оновленню програм і послуг - опцій, які можуть бути змінені відповідно до потреб і вимог користувача. Однак разом з відкритістю появляється необмежений доступ до ресурсів і API-інтерфейсів мобільних пристроїв для додатків невідомого або ненадійного походження, що може завдати шкоди користувачеві, пристрою, мережі або всьому цьому, якщо не використовувати відповідні механізми безпеки і мережеві заходи. Безпека додатків забезпечується у різних форматах на більшості мобільних пристроїв з відкритою операційною системою (Symbian OS, Microsoft, BREW і т.д.).

У 2017 році Google розширив свою програму винагород за виявлення вразливостей, щоб охопити вразливості, виявлені в додатках, розроблених сторонніми розробниками і доступних через Google Play Store. Промислові групи

також розробили рекомендації, включаючи Асоціацію GSM і Open Mobile Terminal Platform (OMTP) [40].

Більшість людей не звертають уваги на безпеку свого мобільного телефону під час оплати за капучино в Starbucks або під час здійснення онлайн-операцій за допомогою мобільного банкінгу. Отже, перед тим як опублікувати додаток, основне питання, яке потрібно вирішити, це як забезпечити захист програми від будь-яких шкідливих намірів?

Мобільний додаток є чудовим засобом для вивчення, оскільки він має програмний код, логіку на серверній та клієнтській сторонах, бази даних, API для передачі даних між пристроями та їх операційними системами, а також з користувачем.

У більшості випадків безпека додатків для Android та iPhone не є пріоритетом для мобільних розробників, головним чином через обмежений час. Зазвичай вона не отримує достатньо уваги в планах проекту. Більше того, якщо в проектній команді немає власника безпеки, ніхто не приймає на себе відповідальність за це. Тому безпека мобільних додатків є питанням, яке залишається на увазі лише за ініціативою розробника.

Безпека та зручність використання є двома тісно пов'язаними концепціями. Високий рівень безпеки вимагає додаткових процесів і потоків. Однак, більшість бізнес-підрозділів, які працюють безпосередньо зі споживачами, не вважають безпеку додатків головним пріоритетом.

На практиці мало хто звертає увагу на проблему безпеки, якщо тільки щось дійсно не піде не так внаслідок "злому". Більшість розробників додатків не проводять конкретні тести безпеки для Android та iPhone. (Тести на безпеку додатків).

Завжди треба мати на увазі, що жодна програма не може бути на 100% безпечною!

Мета розробників полягає в тому, щоб зробити мобільний додаток більш безпечним за інші, використовуючи швидкі та прості функції. Підготуйте свій додаток до мобільної безпеки.

Мобільні додатки, що пов'язані з бізнес-брендами, часто стають ціллю шахраїв, які використовують або клієнтів самого бренду, або їхніх дітей, або ж нападають на сам бізнес.

Якщо спрямоване на пристрій користувача шкідливе програмне забезпечення на мобільних пристроях було атаковане, наслідки можуть включати [46]:

- Заміна номера рахунку.
- Викрадення облікових даних для входу.
- Крадіжку та подальшу продажу кредитних карток.
- Несанкціонований доступ до бізнес-мереж.
- Крадіжку особистих даних.
- Можливість поширення шкідливого програмного забезпечення з пристрою iOS або Android на інші пристрої.
- Копіювання та сканування SMS-повідомлень для отримання особистої інформації.

Мобільні додатки генерують значну кількість даних про нас та наше життя. Тому безпека додатків для створення та використання цих даних має першочергове значення. В іншому випадку небезпечні програми стають простим способом злочинців для крадіжки та продажу особистої інформації. Крім того, існують інші мобільні рішення, які можуть принести значну користь.

2.3. Аналіз методів захисту мобільних застосунків

Цілісний підхід до безпеки включає [49]:

• Захист у спокої.

• Обфускування коду.

• Запобігання переупаковці.

• Криптографія White-box.

• Безпечне локальне сховище.

• Безпечне прикладне ПЗ (SAROM).

• Захист під час виконання.

• Джейлбрейк та корінь.

• Ін'єкція антикоду.

• Реєстрація антиклавіш.

• Анти-екранний рідер.

• Антисистемні скріншоти.

• Антиекранне віддзеркалення та зовнішні монітори.

• Відладчик та емулятор.

• Перевірка цілісності та видимість для подальшого аналізу.

• Різноманітні відповіді у режимі реального часу.

• Можливості порталу самообслуговування.

Більшість сучасних користувачів цифрового банкінгу використовують мобільні пристрої для проведення банківських операцій. На жаль, це відносно недавнє явище корелює із рекордно високим зростанням мобільного шахрайства. У той же час мобільні програми стають все більш критично важливими для бізнесу або будь-якого підприємства. Тим не менш, покращення та оновлення функцій часто мають пріоритет над безпекою, що, як правило, розглядається як тягар, який лише перешкоджає швидкості розробки та випуску.

Ось чому так важливо забезпечити повний захист додатків, у стані спокою та під час виконання – завдяки простому у використанні рішенню, яке не потребує додаткового часу чи зусиль від команди розробників.

Захист у спокої. Надійна безпека необхідна для будь-якої мобільної програми, яка містить особисту інформацію або іншу конфіденційну інформацію, пов'язану з платежами, смарт-контрактами, метаданими чи бізнес-операціями. Це особливо складно для програм, що працюють на джейлбрейкованих або кореневих пристроях.

Щоб підвищити стійкість до інжинірингу, також заплутуємо код мобільного додатка передовими технологіями. Застосовувати обфускацію коду після компіляції, забезпечуючи неінвазивний підхід, не впливаючи на продуктивність програми. Цей додатковий рівень унеможливлює видалення або обхід екранування програм.

Поряд з цими методами ефективно захищаємо додаток від злому та перепакування. Захист програм визначає, чи зловмисник дублював вихідний код програми і ввів шкідливу функціональність.

Підхід до зберігання даних можна унікалізувати таким чином: в першу чергу, конфіденційні дані не повинні залишатися на пристрої довше, ніж це необхідно. Дані можуть бути оброблені за потребою і мають бути негайно видалені, якщо вони більше не потрібні. Якщо необхідно зберегти дані на мобільному пристрої, вони повинні бути зашифровані та збережені в папці документів. Паролі повинні зберігатися в KeyChain для iOS та KeyStore для безпеки Android. Це також важливо для перевірки безпеки магазину додатків.

Відсутня перевірка зовнішнього інтерфейсу може бути перефразована таким чином: відсутність перевірки введених даних призводить до проблем як у сфері безпеки, так і у форматуванні. Це можуть бути такі ситуації, як дозвіл на введення літерно-цифрових значень у числові поля, відсутність маскування для поля, яке не має перевірки на наявність символів високого ризику, таких як <> `" () | #. Відсутня перевірка може призвести до порушень безпеки, дозволяючи виконувати код здалеку або отримувати неочікувані відповіді.

Про серверні елементи управління можна сказати наступне: розробка додатків зазвичай відбувається на стороні клієнта, але серверна частина відповідає за зберігання і управління даними. Перевірка серверної сторони має бути застосована незалежно від каналу передачі (мобільний, веб-тощо) для забезпечення безпеки та форматування даних. Це стосується безпеки серверної частини програми. Існують також проблеми безпеки Apple iCloud, проте це робота Apple, щоб зробити це більш безпечно!

SSL. HTTPS має застосовуватися для безпечної передачі конфіденційної інформації. У разі можливості рекомендується використовувати власний сертифікат замість сертифікатів, що знаходяться відразу в сховищі сертифікатів пристрою. Сертифікат, унікальний для програми та сервера, повинен бути вбудований у додаток.

Обфускація є надзвичайно важливою, особливо для додатків на платформі Android. Скриптові файли також повинні пройти процес заплутування, якщо вони використовуються в деяких частинах додатка.

Рядки коментарів, що містять пояснювальні дані, можуть бути доступними для перегляду іншими. Це поширена помилка, яка стосується як додатків Android, так і iOS. Це не означає, що коментарі не повинні використовуватися під час розробки програми, просто потрібно видалити їх з виробничої версії програми.

Під час налаштування дозволів для додатків Android слід включати лише необхідні дозволи. Дозволи на доступ до особистої інформації, наприклад "доступ до контактів", слід уникати настільки, наскільки це можливо, для забезпечення безпеки Android. Це зменшить ймовірність витоку даних у разі виникнення проблем.

Шифрування. Ключ, який використовується для шифрування, також повинен бути кодовим і зберігатися у безпечному сховищі. Крім того, необхідно заплутувати установчий файл. Ще однією небезпечною практикою, яку слід уникати, є завантаження ключа шифрування з сервера під час виконання [60].

Пристрої з root / Jailbroken. Повністю захищені дані неможливо зберігати на пристроях з отриманим рут-доступом, оскільки кореневі привілеї надають необмежений доступ до файлової системи пристрою та його пам'яті. Однак розробники можуть перевірити, чи пристрій має рут-доступ чи ні. Цей ризик слід відмітити і оцінити з урахуванням масштабів проекту для всіх потоків і процесів.

Захист від несанкціонованого доступу додатків. Виконавчі файли програми (.apk або .ipa) можуть бути змінені і розміщені хакерами на різних майданчиках. Вони можуть впроваджувати шкідливий код у додаток або взламати мобільні додатки, щоб обійти ліцензійні та платіжні обмеження. В даний час навіть можна взламати покупки в додатку і перевірки ліцензій, просто емулюючи відповіді магазину додатків. Тому необхідно перевіряти цілісність додатків і покупки в додатку на сторонньому довіреному сервері, щоб запобігти таким випадкам. На ринку існує кілька надійних рішень для цього. Знову ж таки, це дуже важливо для забезпечення безпеки Google Play і iTunes App Store.

Якщо розробляється додаток з платформним підходом (Obj-C, Swift, Java), вищевказані пункти повинні управлятися повністю розробником або командою. Однак, якщо використовуєте крос-платформний підхід до створення фреймворка, більшість перерахованих вище пунктів охоплені платформою мобільної розробки для безпеки мобільних додатків.

Особливо рекомендую використовувати крос-платформні вбудовані фреймворки для індустрії мобільної безпеки.

Існує кілька стратегій підвищення безпеки мобільних додатків [62]*:*

- Складання "білого списку" додатків.
- Забезпечення безпеки на рівні передачі даних.
- Сувора перевірка автентифікації та авторизації.
- Шифрування даних при їх записі в пам'ять.
- Використання пісочниці для додатків.
- Керування доступом додатків на рівні API.
- Прив'язка процесів до ідентифікатора користувача.
- Взаємодія між мобільним додатком та операційною системою.
- Вимога підтвердження користувача для отримання привілеїв або підвищеного рівня доступу в додатку.
- Правильне управління сеансами.

Розглянемо загальні уразливості, без прив'язки до конкретної платформи. Тут і надалі використовується абревіатура КВД - критично важливі дані користувачів. Під КВД розуміються дані, які не повинні бути доступні третім особам. Це можуть бути персональні дані користувача (дата народження, адреса проживання, особисті повідомлення) або його приватні дані (паролі, інформація про кредитні картки, банківські рахунки, номери замовлень та інше).

Таким чином, ми можемо виділити наступні основні вразливості мобільних додатків:

- Використання незахищених локальних сховищ.

Небезпека: Дуже висока.

Зустрічається повсюди, проявляється у зберіганні КВД у незахищених або слабо захищених локальних сховищах, специфічних для певної платформи.

Розкриття третьою стороною - проста процедура і, як правило, не вимагає особливих навичок з боку зловмисника.

Захист: Необхідно зберігати КВД лише в надійних сховищах, що надаються платформою.

Зберігання КВД у вихідному коді.

Небезпека: Висока.

Вразливість полягає у зберіганні КВД безпосередньо в коді (у статичних рядках, ресурсах додатка та ін.). Яскраві приклади цього - зберігання солі для паролю у вигляді константи або макросу, що використовується у всьому коді для шифрування паролів; зберігання приватного ключа для асиметричних алгоритмів; зберігання паролів і логінів для серверних вузлів або баз даних. Це легко розкривається третьою стороною за наявності базових навичок декомпіляції.

Захист: Необхідно уникати зберігання будь-яких КВД у вихідному коді або ресурсах додатка.

Використання алгоритмів із зберіганням приватного ключа.

Небезпека: Висока.

Уразливість виникає, коли конфіденційна інформація алгоритму (приватний ключ) примушується зберігатися в коді або ресурсах мобільного додатка (що найчастіше має місце). Це легко розкривається за допомогою декомпіляції.

Захист: У розробці мобільних додатків рекомендовано використовувати лише сучасні симетричні алгоритми з генеруванням випадкового одноразового ключа, які мають високу стійкість до перебору методом "грубої сили". Також можна виводити асиметричний приватний ключ за межі програми або персоналізувати його (наприклад, використовувати код входу, призначений для користувача, який зберігається у зашифрованому вигляді в захищеному сховищі операційної системи).

Використання асиметричного алгоритму з приватним ключем, відомим серверу.

Небезпека: Залежить від рівня захищеності сервера.

Уразливість має дві сторони. Зберігання приватного ключа дозволяє розшифрувати дані користувача на стороні сервера. По-перше, це небезпечно з точки

зору безпеки (якщо сервер буде скомпрометований, зловмисник також отримає доступ до приватних даних користувачів). По-друге, це порушує приватність особистих даних. Користувач завжди повинен мати впевненість, що його персональна інформація відома тільки йому (за винятком випадків, коли він явно надав дозвіл на її розкриття). Часто додатки претендують на статус безпечних, але насправді вони містять засоби для розшифрування особистої інформації.

Захист: Без явної необхідності і явного дозволу користувача (найчастіше через ліцензійну угоду) ні додаток, ні сервер не повинні мати ніякої можливості розшифрувати приватні дані користувача. Найпростіший приклад - пароль користувача повинен йти на сервер вже у вигляді хешу, і перевірятися повинен хеш, а не вихідний пароль (серверу абсолютно нема чого знати пароль користувача; якщо ж користувач його забув - для такої ситуації існує давно налагоджений механізм відновлення пароля, в тому числі з двухфакторною авторизацією клієнта для підвищеної безпеки процедури відновлення).

Використання самописних алгоритмів шифрування і захисту.

Небезпека: Середня.

Захист: Це пряме порушення принципу Керкгоффса, коли розробник намагається створити власний, унікальний і невідомий нікому супер-захищений алгоритм шифрування. Будь-яке відхилення від встановлених, перевірених і математично обґрунтованих алгоритмів шифрування у 99% випадків призводить до швидкого злому такого "захисту". Це вимагає високих навичок у зловмисника.

Захист: Для забезпечення безпеки необхідно використовувати відповідний алгоритм з надійних і визнаних криптографічних алгоритмів.

Передача КВД в незахищеному вигляді до зовнішнього середовища.

Небезпека: Середня.

Це означає передачу КВД без застосування шифрування через будь-який доступний канал зв'язку з зовнішнім середовищем, незалежно від того, чи це передача даних до стороннього додатку або мережі. Злом може відбутися через розтин не самої програми, а його сховища або цільової програми. Цей тип атаки вимагає володіння певними навичками у зловмисника, за умови, що сховище є захищеним.

Захист: Будь-які КВД, які виходять за межі додатка, повинні бути обов'язково зашифровані. Локальні сховища платформи не є надійними місцями зберігання, тому вони також повинні отримувати лише зашифровані дані.

Ігнорування факту наявності пристроїв з рут-правами або заражених пристроїв.

Небезпека: Середня.

Рутовані пристрої - це девайси, де виконано модифікацію для отримання прав суперкористувача на будь-які операції, спочатку заборонені виробником операційної системи. Виконується користувачем на своєму пристрої самостійно, і не обов'язково добровільно (клієнт може бути не в курсі, що пристрій зламано).

Захист: Якщо це технічно можливо для платформи - то бажано забороняти роботу додатка, якщо вдалося зрозуміти, що запуск проводиться на рутованном пристрої, або хоча б попереджати про це користувача

Зберігання КВД в захищених сховищах, але у відкритому вигляді.

Небезпека: Середня.

Розробники часто схильні зберігати КВД в захищені системні сховища без додаткового захисту. Однак рівень їх стійкості падає до мінімуму в разі, якщо пристрій рутований.

Захист: КВД не повинні використовуватися в додатку без додаткового шифрування. Як тільки потреба в "відкритих" КВД відпала - вони негайно повинні бути або зашифровані, або знищені.

Переклад частини функціоналу під вбудовані веб-движки.

Небезпека: Середня.

Захист: Найчастіше це виявляється у використанні вбудованого браузера, де завантажується зовнішня веб-сторінка, що виконує свої функції. У такому випадку рівень захисту значно знижується, особливо для рутованих пристроїв.

Захист: Рекомендується уникати використання вбудованого браузера та вбудованого веб-движка для операцій з КВД. У крайньому випадку можна шифрувати КВД перед передачею.

Реверсний інжиніринг алгоритмів з великою інтелектуальною цінністю.

Небезпека: Низька, залежить від цінності алгоритму.

Якщо під час розробки програми всередині компанії використовуються власні алгоритми, які можуть мати велику цінність для потенційних конкурентів або зловмисників, то ці алгоритми повинні бути захищені від зовнішнього доступу.

Захист: Автоматична або ручна обфускація коду.

Неправильне використання платформи.

Цей тип вразливості займає провідне місце у списку. Незалежно від того, чи це Android чи iOS, при розробці для будь-якої з цих платформ необхідно дотримуватися конкретних вимог для забезпечення належного рівня безпеки кінцевого продукту. Однак іноді при створенні додатків несвідомо порушуються деякі з запропонованих правил або виникають помилки у їх реалізації.

В результаті виникає загроза, спричинена неправильним використанням будь-якої можливості платформи або відсутністю реалізації протоколів безпеки. Тут можуть мати місце такі ситуації [64]:

Неправильне використання функції Touch ID в iOS може призвести до несанкціонованого доступу до пристрою.

Некоректне використання iOS Keychain, що призводить до зберігання чутливих даних, таких як ключі сесії або паролі, у локальному сховищі додатка замість захищеного сховища.

Запит надмірних або неправильних дозволів платформи.

Intents в Android (використовуються для запиту дій з іншого компонента додатка), які відкриті для доступу, можуть розкривати чутливу інформацію або дозволяти несанкціоноване виконання.

Як запобігти: Схожі вразливості слід виправляти на серверній стороні. Крім дотримання рекомендацій платформи щодо розробки, ризики можна зменшити за допомогою безпечних підходів до написання коду та правильних налаштувань сервера. Для мінімізації неправильного використання платформи також можна:

Забороняти взаємодію між додатками, обмежувати доступ та застосовувати обмежувальні права доступу до файлів та інших ресурсів.

Використовувати найсуворіший рівень захисту для ключових ланцюжків iOS та використовувати найкращі методи розробки, щоб уникнути слабких реалізацій елементів управління.

Небезпечне сховище даних

Мобільні пристрої часто загублюються або потрапляють у руки зловмисників. Крім того, витік особистої інформації користувача може статися через шкідливе програмне забезпечення, яке використовує вразливості пристрою. Злом або рутінг пристрою дозволяє обійти шифрування, тому розробники програмного забезпечення повинні припускати, що зловмисники можуть отримати доступ до файлової системи.

Оскільки практично всі програми зберігають інформацію, дуже важливо забезпечити її збереження в області, недоступній для інших додатків або користувачів.

Як запобігти: Проводити тестування програми на вразливості, щоб визначити, які інформаційні активи обробляються і як взаємодіють з цими даними API. Це допоможе:

Перевірити ефективність застосування шифрування та захисту ключів шифрування.Ускладнити злам коду шляхом обфускації, захисту від переповнення буфера і т.д.

Уникнути збереження/кешування даних там, де це можливо.

Впровадити надійні методи аутентифікації і авторизації.

Небезпечна комунікація

Якщо дані передаються у вигляді звичайного тексту без шифрування, будь-хто, хто відстежує цю мережу, може перехопити і прочитати їх.

Мобільні додатки, як правило, обмінюються даними за моделлю клієнт-сервер. При цьому процес передачі через мережу оператора або Інтернет має бути забезпечений безпекою. Трафік може перехоплюватися проксі-серверами, базовими станціями, а також за допомогою злому WiFi або шляхом установки на пристрій шкідливого ПЗ.

Як запобігти: Для уникнення крадіжки даних в процесі їх передачі по мережі слід покладатися на затверджені індустрією протоколи шифрування і інші практики, включаючи:

Встановлення сертифікатів SSL/TLS від авторитетних центрів сертифікації (ЦС).

Сигналізування користувачам при виявленні недійсного SSL/TLS сертифіката або невдачі перевірки ланцюжка сертифікатів.

Небезпечна перевірка автентичності

Перед тим, як надати доступ, додаток повинен перевірити, чи є користувач справжнім. Обхід аутентифікації зазвичай виконується через використання вразливостей, таких як некоректна перевірка запитів до сервісів сервером. Мобільні додатки повинні перевіряти та забезпечувати справжність користувача, особливо під час передачі конфіденційної інформації, наприклад, фінансових даних.

Як запобігти: Використання слабких методів аутентифікації дозволяє зловмисникам обходити системи перевірки паролів або отримувати додаткові привілеї, виконуючи крадіжку даних та інші зловживання.

Для попередження таких ризиків рекомендується:

Відмовитися від локальної аутентифікації. Замість цього, виконання аутентифікації можна передати на серверну сторону, а дані додатка завантажувати лише після успішної перевірки автентичності користувача.

Уникати використання вразливих методів аутентифікації (наприклад, пристроєвих ідентифікаторів), не зберігати паролі локально, використовувати багатофакторну аутентифікацію (МФА), заборонити використання 4-значного PIN-коду як пароля та інші заходи.

Недостатня криптографічний стійкість

Існує два випадки, в яких криптографія системи може бути скомпрометована для розкриття чутливих даних [65]:

- Слабкий внутрішній алгоритм шифрування / дешифрування.

- Прогалини в реалізації самого процесу криптографії.

Успішний злом в таких випадках може бути наслідком ряду факторів, включаючи:

- Обхід вбудованих алгоритмів шифрування коду.
- Неправильне керування цифровими ключами.
- Використання застарілих протоколів шифрування.

*Як запобігти***:** Недоліки системи криптографічного управління доступом можуть вести до витоку чутливих даних з пристрою. У зв'язку з цим слід:

- Застосовувати суворі стандарти криптографії, рекомендовані Національним інститутом стандартів і технологій (NIST).
- Уникати зберігання на пристроях важливої інформації.

Небезпечна авторизація

Для різних користувачів передбачаються різноманітні привілеї, що призводить до того, що деякі мають стандартний доступ, тоді як інші, такі як адміністратори, можуть мати додаткові дозволи та привілеї. Вразливі схеми авторизації, незалежно від успішної перевірки автентичності користувача, можуть не забезпечити перевірку його прав на доступ до потрібного ресурсу. Ця недолік дозволяє хакерам отримати авторизацію та виконувати атаки з метою підвищення привілеїв.

Як запобігти: Аналогічно до аутентифікації, недоліки авторизації можуть призвести до крадіжки даних, підриву репутації та навіть санкцій за невиконання вимог. Щоб запобігти цим ризикам, розгляньте такі варіанти:

Реалізуйте перевірку кожного запиту сервером на відповідність вхідних ідентифікаторів особі користувача, з якої вони асоційовані.

Перевіряйте ролі та дозволи автентифікованого користувача, використовуючи інформацію з бекенд-системи, а не з мобільного пристрою.

Якість коду клієнта

Ця категорія в певному сенсі є загальною причиною проблем мобільного клієнта, пов'язаних з неправильною реалізацією коду.

Атакуючий може передавати спеціальні вхідні дані в виклики функцій, що призводить до їх виконання та спостереження за поведінкою програми. Це може

спричинити погіршення продуктивності, збільшення використання пам'яті та інші проблеми.

У цьому випадку варто мати на увазі, що помилки в коді потрібно виправляти локальним способом, оскільки вони виникають в мобільному клієнті і відрізняються від помилок серверної сторони. Привести ж вони можуть до:

- вразливості форматуючих рядків;
- переповнення буфера;
- впровадженню небезпечних сторонніх бібліотек;
- віддаленого виконання коду.

При розробці додатків часто використовуються зовнішні бібліотеки, які самі можуть містити помилки і не бути достатньо протестованими. Ці недоліки знаходяться поза контролем розробника, оскільки він не має доступу до вихідного коду. У інших випадках частіше за все помилки коду виправляються переписуванням відповідних його частин. Проте, що ще можна зробити?

Як запобігти: Для усунення вразливостей та інших проблем, спричинених некоректним кодом, рекомендується [70]:

Використовувати автоматизовані інструменти для тестування буфера на переповнення, виявлення витоків пам'яті тощо.

Покладатися на рецензування вихідного коду та спочатку створювати його в зрозумілому, добре задокументованому вигляді.

Використовувати в організації узгоджені шаблони написання коду.

Підробка коду

Іноді у магазинах можна зустріти підроблені версії додатків. Вони відрізняються від оригіналу тим, що містять шкідливий вміст, наприклад, закладку, що дозволяє незаконний доступ до системи. Зловмисники можуть повторно підписувати ці підроблені програми та розміщувати їх у сторонніх магазинах або навіть безпосередньо доставляти жертві через фішингові атаки.

Як запобігти: Підробка коду програми може призвести до втрати користі, крадіжки особистих даних, підриву репутації та інших негативних наслідків. Для запобігання подібним проблемам рекомендується:

Додаток повинен мати можливість виявляти будь-яке порушення цілісності свого коду (тобто зміну або введення сторонніх елементів) і адекватно реагувати на це у середовищі виконання. Інформування користувачів про законні зміни в коді можна, наприклад, здійснювати за допомогою сертифікатів підпису.

Необхідно впровадити техніки, які перешкоджають виконанню підроблених додатків, такі як контрольні суми, цифрові підписи, зміцнення коду та інші методи перевірки.

Реверс-інжиніринг

Зловмисники можуть розбирати та декомпілювати додаток, щоб проаналізувати його код. Цей метод зламу особливо небезпечний, оскільки дозволяє перевірити, зрозуміти та змінити код, включаючи шкідливі функції або небажану рекламу. Після розуміння роботи програми, хакери можуть змінювати її за допомогою таких інструментів, як IDA Pro, Hopper та інші. Після досягнення потрібної поведінки вони можуть повторно компілювати додаток і використовувати його для своїх цілей.

Як запобігти: Зловмиснику можна ускладнити процес реверс-інжинірингу програми, забезпечивши неможливість деобфускації коду за допомогою IDA Pro, Hopper та інших подібних інструментів.

Зайвий функціонал

Іноді розробники можуть ненавмисно залишати закладки або додатковий функціонал, який не є очевидним для кінцевого користувача. В результаті продукт випускають в продакшен з функцією, яка не повинна бути доступною, що створює додаткові ризики для безпеки програми.

Хакери використовують подібні вразливості програм безпосередньо зі своїх систем, без необхідності взаємодії з регулярними користувачами. Вони досліджують файли конфігурації, виконувані файли та інші компоненти, розкриваючи можливості бекенд-частини, які в подальшому використовують для здійснення атак.

Як запобігти: Один з найбільш ефективних способів уникнення таких вразливостей - це проведення самостійного аналізу безпеки коду. Це дозволить:

Перевірити налаштування програми на наявність прихованих параметрів.

Переконатися, що логи не містять занадто детальних інструкцій щодо роботи сервера.

Підтвердження особистості

Підтвердження особистості допомагає запобігти крадіжці особистих даних користувачів та реєстрації облікових записів під їхнім ім'ям. Надійний процес підтвердження особистості гарантує, що користувач є тим, ким він стверджує, і допомагає запобігти шахрайству зловмисників.

Надійна аутентифікація

Зловмисники часто намагаються заволодіти обліковими записами, і паролі швидко стають застарілими. З огляду на значну кількість витоків даних протягом останніх десяти років, багато комбінацій імен користувачів та паролів вже доступні для продажу в темному вебі. Надійні методи аутентифікації гарантують, що тільки законні користувачі отримують доступ до своїх облікових записів.

Біометрія

Біометрія - це безпечний і зручний спосіб входу в мобільні додатки, використовуючи дані, отримані від вашого власного організму. Немає надійного способу встановити, хто саме вводить пароль. Розробник програми може лише перевірити, чи співпадає введений пароль зі збереженим паролем у серверній частині системи. Біометрія надає додатковий рівень довіри, оскільки підтверджує особистість людини, що надає біометричний зразок для перевірки. Відбиток пальця, розпізнавання обличчя або сканування райдужної оболонки ока передаються в реальному часі та пов'язані з користувачем в режимі реального часу.

Забезпечення безпеки через бекенд

Велика кількість бекенд-API показує, що неправдоподібно, що додаток, якому було надано доступ, зможе вийти з-під контролю. Бекенд-сервери повинні мати вбудовані системи безпеки для захисту від шкідливих атак. Усі API підлягають аутентифікації на основі мобільної платформи, з якою вони працюють, оскільки процеси передачі даних та аутентифікації можуть відрізнятися залежно від платформи.

Високий рівень аутентифікації

Тому стає важливим використання сильної аутентифікації. Аутентифікація часто пов'язується з використанням паролів. Це завдання, що ставиться перед розробником, щоб підтримувати користувачів, які хвилюються про безпеку своїх паролів. Для ілюстрації можете створити свою програму, щоб він приймав лише сильні літеро-цифрові паролі, які можуть бути відновлені кожні три місяці.

Аутентифікація подвійного фактора також є прекрасною ідеєю для забезпечення мобільного додатка. Якщо програма дозволяє аутентифікації подвійного фактора, колись користувача буде закликано, щоб ввести код, доставлений до електронної пошти після входу. Якщо більше говоримо про сучасні методи автентифікації, то вона включає в себе біометричні засоби, як сканування сітківки і відбитки пальців.

Запустіть найкращі інструменти та методи шифрування

Робота номер один, щоб зробити для сильного шифрування, полягає у виборі керування ключами. Зберігайте клавіші в безпечних контейнерах. Ніколи не кладіть їх на сервері.

Накласти політику доступу

Щоб скоротити поверхню атаки, зробити його сильним - використовувати лише безпечні бібліотеки та рамки. Додаток, який робите, повинен призвести до спільної політики, реалізованої інформацією Організації IT-менеджерів або програмою Google Play та Apple App Store.

Тестування програми

Багато розробників не перевіряють свій код. Це необхідна частина розробки коду якості. Ось чому лише частина процесу застосування керується створенням чудового мобільного додатка.

Щоб мати безпечний додаток, команда повинна регулярно оцінювати код та аналізувати лазівки безпеки, які можуть виникнути в порушеннях даних.

Стандарти та вимоги безпеки:

- Стандарт написання коду CERT.
- CWE.
- Технічне керівництво з безпеки (STIG).

- ISO / IEC 27034-1: 2011 Information technology - Security techniques - Application security - Part 1: Overview and concepts.
- ISO / IEC TR 24772 діє до: 2013 Information technology - Programming languages -Guidance to avoiding vulnerabilities in programming languages through language selection and use.
- NIST Special Publication 800-53.
- OWASP.
- Стандарт безпеки даних індустрії платіжних карт PCI DSS.

Проблеми з перевіркою автентичності роблять мобільні додатки вразливими до порушень безпеки. У сфері розробки мобільних додатків вивчають можливості безпарольних рішень, таких як біометрія та двофакторна автентифікація, як альтернативу перевірці облікових даних. Організації та розробники, які ще не ознайомлені з безпарольним підходом, повинні переконатися, що їх мобільні додатки приймають тільки надійні складні паролі з літер і цифр.

Якщо додатки мають велику конфіденційність, надійні практики автентифікації, які відповідають бізнес-вимогам, повинні бути ключовими елементами у розробці мобільних додатків. Оскільки проблеми з безпекою додатків все частіше привертають увагу, використання високорівневої автентифікації стає необхідністю.

У майбутньому розробники мобільних додатків повинні приділяти увагу кібербезпеці. Порушення безпеки даних може завдати фінансової шкоди організаціям, незалежно від їх типу чи причини. Завданням організацій є зрозуміння необхідності впровадження найкращих практик кібербезпеки та їх врахування на кожному етапі розробки.

Атаки на рівні програми. На відміну від веб-додатків, мобільні додатки піддаються атакам на рівні програми, оскільки їх вихідний код повинен бути загальнодоступним. Зловмисник може завантажити програму, скомпрометувати вихідний код і використати його. Способи зробити це: Реверс-інжиніринг.

Деякі хакери використовують спеціальні засоби для реверс-інжинірингу вихідного коду програми. Це може розкрити основну бізнес-логіку компанії, яка потім може бути використана конкурентами для крадіжки ідей і тактик.

Витягування конфіденційної інформації. Існують доступні інструменти, що дозволяють видобути рядкові константи з двійкового файлу. Це може призвести до розкриття критичної інформації, такої як облікові дані адміністратора чи конфіденційні URL-адреси.

Впровадження шкідливого коду та нелегальне поширення програми. Деякі хакери можуть зламати сам двійковий файл програми та вставити у нього свій шкідливий код. Після цього вони поширюють програму через неофіційні канали та встановлюють її на пристроях користувачів, які нічого не підозрюють.

Це дозволяє їм здійснювати такі дії, як розкрадання користувацьких даних через "фішинг", автоматичне перенаправлення користувачів на їхні веб-сайти або продукти або показ речей, які можуть завдати шкоди репутації та довірі до компанії.

Атаки на рівні мобільних пристроїв. Атаки на рівні пристрою полягають у використанні вразливого пристрою для отримання доступу до мережі. Ці атаки можуть бути здійснені на будь-яких підключених пристроях і набувати різних форм, наприклад:

Шкідливі програми, які викрадають дані. Хакери розповсюджують власні програми під виглядом ігор, утиліт тощо, які негласно спостерігають за діями та введеннями користувачів. Таким чином, вони зможуть викрасти багато деталей, наприклад, які інші програми встановлено, усі введення користувача з клавіатури, усю мережеву активність тощо.

Встановлення вашого додатка на пристрої з рут-правами/прошивкою. Хакери модифікують операційну систему, яка встановлена на їх телефоні, і потім запускають програму. Це дозволяє їм спостерігати за внутрішньою активністю програми, наприклад, збереженими в ній даними, мережевими викликами тощо, про що звичайний користувач не має уявлення. Збираючи цю інформацію, вони отримують більше знань про роботу продукту чи послуги та можуть зловживати цими даними.

Модифікація даних програми. Хакери перевіряють файлову систему та виявляють, як програма зберігає файли та дані локально. Часом зміна файлів даних може змінити поведінку програми залежно від намірів хакера. Наприклад, шляхом зміни файлу, хакер може виглядати так, ніби він увійшов до програми без використання облікових даних.

Спостереження за журналами. Іноді розробники програми залишають журнали для налагодження програми і забувають їх видалити перед випуском робочої версії. Будь-хто може просто переглянути ці журнали і отримати уявлення про роботу додатків.

Спостереження за незашифрованим мережевим трафіком. Якщо зв'язок програми з сервером не належним чином зашифрований, спостерігач може прочитати всі передані дані мовою простою. Це включає облікові дані, які передаються на сервер, конфіденційну інформацію, яку сервер повертає та інше.

Атаки на рівні сервера. Зламавши мобільну програму, як описано на попередніх двох рівнях, хакер міг отримати інформацію про те, як програма взаємодіє з веб-службою, і спробувати використати веб-службу.

Атаки «людина посередині». Хакер розуміє виклики API, зроблені програмою, використовує автентифікацію, надіслану програмою, і видає себе за законного користувача. Нічого не підозрюючи сервер може надати хакеру конфіденційні дані.

DDoS-атаки: знаючи кінцеві точки API, які використовує програма, хакер може використовувати автоматизовані інструменти для передачі великого трафіку на ці кінцеві точки, що призведе до виходу з ладу сервера. По суті, продукт або послуга стануть непридатними для реальних користувачів.

Щоб переконатися, що програма витримає ці атаки та забезпечити найкращий захист для продуктів і користувачів є певні стандарти та найкращі практики, яких потрібно дотримуютьсь з самого початку написання першого рядка коду. Це окрім стандартних заходів, як ніколи не додавати конфіденційну інформацію в код, відключати всі журнали робочих збірок, очищати всі введені користувачем дані перед їх обробкою, додавати звіти про збої та аналітику для виявлення будь-якої незвичної поведінки програми тощо [73].

Ніколи не зберігайте облікові дані на пристрої. Ніколи не зберігайте конфіденційну інформацію, як ім’я користувача та пароль, на пристрої, навіть у приватному просторі програми. Натомість використовуйте OAuth або іншу автентифікацію на основі маркерів, щоб надсилати запити від імені користувача. Крім того, в ідеалі оновлюйте та закінчуйте сеанси користувача через попередньо встановлений проміжок часу, щоб запобігти неправильному використанню будь-якої скомпрометованої інформації автентифікації протягом тривалого часу.

Використовуйте приватний простір програми та використовуйте шифрування для зберігання будь-яких даних користувача. У всіх операційних системах для кожної програми існує окремий простір для зберігання своїх даних, який недоступний для звичайних користувачів. Щоб запобігти читанню цих даних навіть користувачем із кореневим доступом, зашифруйте всі дані. Ніколи не використовуйте загальнодоступну область даних, як SD-карту, для зберігання даних будь-якої програми.

Підтримуйте свою програму та її компоненти в актуальному стані. Щодня в мобільному та веб-просторі виявляються сотні вразливостей, і регулярно випускаються виправлення. Розробники повинні переконатися, що вони включають ці виправлення у свої програми, і заохочують користувачів регулярно оновлювати свої програми. Це гарантує, що хакери, які спробують виявити ці відомі вразливості, не матимуть успіху.

Використовуйте обфускацію на робочих збірках. Щоб перешкодити спробам перепроектувати двійковий файл програми для вилучення основної бізнес-логіки та іншої конфіденційної інформації в коді, використовуйте обфускацію (навмисне приховування даних) і мінімізацію. Це не тільки ускладнить читання коду зворотного проектування, але й зробить двійковий файл програми більш компактним. Прикладом є Proguard, доступний для Android.

Використовуйте HTTPS для зв’язку через API. HTTPS — це безпечний протокол для зашифрованого зв’язку у відкритій мережі. Під час використання HTTPS особа, яка спостерігає за мережевою активністю, не зможе отримати жодної інформації про запити, такої як URL-адреса, інформація для входу/автентифікації або

будь-які дані, що передаються через веб-службу. Окрім цього, залежно від високих вимог безпеки клієнта, також включаємо такі функції для запобігання будь-яким проблемам безпеки:

Вбудована можливість видаляти дані програми віддалено. Якщо дізнаємося, що пристрій було зламано або облікові дані для входу зламано, надамо віддалену веб-панель, яка дасть можливість легко змінити облікові дані для входу або виконати видалення даних пристрою через API або навіть просте SMS.

Виявляти рутовані/зламані пристрої. Програма буде створена з можливістю визначати, чи працює вона на зламаному пристрої, і може навмисно припинити роботу.

Виявлення підробленого двійкового файлу додатка. Додаток можна створити з можливістю перевіряти, чи його двійковий файл було змінено, вставивши або видаливши частину коду шляхом перевірки контрольної суми файлів. Після виявлення він завчасно припинить роботу, щоб запобігти подальшому пошкодженню.

2.4. Аналіз атак на мобільні застосункі

За 2022 рік експерти Positive Technologies виявили 216 уразливостей у 25 парах досліджених програм для Android та iOS. Найбільша частка вразливостей (14%) припала на зберігання даних у відкритому вигляді. На думку фахівців Positive Technologies, незважаючи на зусилля з боку розробників операційних систем та співтовариств із безпечної розробки додатків, цей клас уразливостей продовжує впевнено зберігати лідерство кілька років поспіль і зберігатиме актуальність і у 2023 році [74].

Друге місце поділили вразливості щодо контролю цілісності додатків та зберігання конфіденційної інформації в коді (по 9% на кожен клас).

Замикає трійку лідерів клас уразливостей, пов'язаних із перевірками на недовірене оточення (8%). Наявність у додатках перерахованих вище вразливостей свідчить, що розробники недостатньо суворо контролюють цілісність додатків і їх виконання.

Також за відсутності хорошої обфускації коду (таку комбінацію експерти виявили у 36% додатків, досліджених у 2022 році), складається сприятлива ситуація для зловмисників: стає дуже просто проводити якісний аналіз додатків, що, у свою чергу, спрощує створення ботів, клонів та троянів, націлених на конкретні програми.

Найцікавішим трендом 2022 року в Positive Technologies назвали відсутність у додатках деяких класів уразливостей. Наприклад, розробники тепер не зберігають криптографічні ключі у файловій системі і не припускають помилок, що відкривають можливість обходу директорій (path traversal). Вразливість, пов'язана з небезпечною відправкою неявних міжпроцесних повідомлень, зустрілася у досліджених додатках у 2022 році лише раз (у попередньому році — шість випадків). Експерти очікують, що ця позитивна тенденція зростатиме 2023 року [76].

Нові версії операційних систем теж допомагають розробникам додатків: вводяться більш гранулярні дозволи виконання системних операцій, а низки дозволів з'явилася можливість запитувати їх щоразу.

На думку групи досліджень безпеки мобільних додатків Positive Technologies, санкції та блокування вивели проблему клонованих та підроблених додатків на новий рівень. Мобільні програми багатьох компаній були видалені з офіційних магазинів, через що користувачам довелося шукати їх на інших майданчиках. Зловмисники не забули цим скористатися і почали активно розміщувати фальшиві програми відомих компаній.

Вимушеним трендом 2022 став запуск магазинів додатків, покликаних замістити Google Play і App Store. У Positive Technologies зазначили, що на шляху до серця користувача цим магазинам доведеться пройти непростий шлях та скоротити його допоможуть участь у програмах bug bounty та співпраця з спільнотаи фахівців з ІБ [80].

У 2023 році не втратить актуальності проблема нестачі фахівців із аналізу захищеності мобільних додатків. У той же час розвиток тематичних спільнот, програм bug bounty, і поява більш просунутого інструментарію дадуть поштовх до збільшення числа фахівців цього профілю на ринку, а отже, і підвищення рівня безпеки мобільних додатків.

Проблема 1. Атака DDoS на веб-ресурси та сервери компанії.

Для вирішення даної проблеми доступний сервіс AntiDDoS, який допомагає захистити ресурси підприємства від атак DDoS. AntiDDoS працює в мережі Інтернет, що належить Київстар, і відстежує трафік, що надходить з Інтернету до підмереж (ресурсів) клієнта. Ця програма виявляє відхилення від нормального трафіку і автоматично перенаправляє його для очищення, а лише потім надсилає клієнту.

AntiDDoS працює з трафіком не лише за допомогою сигнатурного аналізу, а й з використанням машинного навчання. Цей сервіс захищає системи від відомих і невідомих атак нового типу, бореться з будь-якими DDoS-атаками на різних рівнях моделі OSI (3, 4, 7), об'ємними атаками і атаками SSL Renegotiation/HTTP. Крім того, завдяки вбудованим засобам комплексного звітності, AntiDDoS надсилає централізовану аналітику щодо подій.

Захист від атак DDoS. AntiDDoS забезпечує багаторівневий захист IT-інфраструктури компанії від відомих і невідомих атак. Це рішення легко розгортати та використовувати, оскільки воно включає комплексні інструменти для аналізу та автоматичного формування звітів. AntiDDoS базується на рішенні FortiDDoS від Fortinet.

Проблема 2. Потік нелегітимного трафіку.

Для того, щоб запобігти проникненню шкідливих програм через зашифрований трафік у онлайн-мережу, необхідно налаштувати ефективну та надійну систему перевірки. Цю задачу впорається виконати сервіс Next Generation Firewall (NGF). Він допоможе захищати ресурси від атак на різних рівнях, постійно аналізуючи корпоративний трафік і оперативно блокуючи загрози після їх виявлення.

Сервіс Network Access Control (NAC) забезпечує захист корпоративної ритейл-мережі від шахрайства. Він обмежує доступ несанкціонованих користувачів та пристроїв до корпоративної мережі. Це гарантує, що у мережу компанії можуть входити лише автентифіковані користувачі та пристрої, що відповідають політикам безпеки. Вручну аналізувати всі пристрої, що підключаються до мережі, є неможливим. Автоматизовані функції NAC допомагають зекономити час і знизити

витрати, пов'язані з автентифікацією та авторизацією користувачів, а також визначенням відповідності їхніх пристроїв вимогам безпеки.

Проблема 3. Крадіжка та втрата важливих даних компанії.

Забезпечення безпеки будь-якої корпоративної інформації здійснюється за допомогою програми Complete Data Protection. Вона контролює доступ до ресурсів, аналізує поведінку користувачів і шифрує диски на рівні підприємства. Таким чином, у випадку втрати даних їх буде неможливо прочитати або використати. Проте сервіс Backup and Recovery забезпечує можливість резервного копіювання та відновлення інформації, яка зберігається у хмарних, віртуальних та локальних середовищах.

У той же час, File Storage Optimization безпечно оптимізує сховища даних компанії, що підвищує ефективність зберігання та зменшує витрати. Оскільки надлишкові або застарілі дані, що перевищують рекомендований період зберігання і не мають корисності для бізнесу, займають приблизно 80% сховища.

Проблема 4. Завантаження шкідливого програмного забезпечення через електронну пошту.

Для захисту корпоративної електронної пошти від шкідливих програмних зловмисників використовується сервіс Email Security. Він забезпечує шифрування та архівацію електронної пошти, запобігає втраті важливої інформації та захищає від фішингових атак та спаму. Сервіс eDiscovery&Compliance допомагає автоматизовано збирати дані для пошуку конкретних електронних листів, файлів та інформації у електронному форматі. Програма дозволяє забезпечити надійність даних та встановити ланцюг зберігання. У такий спосіб мінімізується можливість втрати або зміни вихідної інформації, адже зберігаються резервні копії файлів.

Рішення для кібербезпеки . Інструменти, що комплексно захистять мережеву інфраструктуру, сервіси та хмарні рішення компанії від загроз різних типів. Розроблено лідерами у сфері інформаційного захисту Fortinet, Barracuda і Commvault.

Проблема 5. Надходження шкідливого трафіку до вебдодатків.

Із сервісом Web Application Firewall (WAF) можливо захистити вебдодатки від нелегітимного трафіку з мережі інтернет. Він відстежує та блокує будь-який

зловмисний HTTP/S-трафік, що надходить до веб-програми, а також запобігає виходу будь-яких несанкціонованих даних з ресурсу компанії.

Проблема 6. Порушення правил використання мережі.

Програма User and Entity Behavior Analytics (UEBA) використовує алгоритми та машинне навчання для виявлення відхилень у поведінці не тільки користувачів корпоративної мережі, але й маршрутизаторів, серверів та кінцевих пристроїв у цій мережі. UEBA не лише спостерігає за поведінкою людей - вона також контролює машини. Наприклад, сервер в одному з відділень може раптово отримати надзвичайно велику кількість запитів. Це може бути початком DDoS-атаки, яка може спричинити порушення нормальної роботи інтернет-магазину, у результаті чого сайт не зможе обробляти замовлення. UEBA здатна розпізнати цей тип атаки та припинити загрозу.

Безпека кінцевих пристроїв є важливим стратегічним завданням, оскільки кожен смартфон або POS-термінал може бути точкою входу для атаки. Сучасні системи захисту кінцевих пристроїв розроблені для швидкого виявлення, аналізу, блокування та стримування загроз. Саме тому сервіс Endpoint Security допомагає захищати кінцеві пристрої IT-інфраструктури торгової компанії (сервери, персональні комп'ютери, ноутбуки, смартфони, термінали) від відомих і невідомих атак, виявляє та блокує загрози.

Захист від сучасних мобільних загроз та надання максимального комфорту користувачам є також важливими аспектами.

Фінансові організації не можуть контролювати безпеку на мобільних пристроях клієнтів. Однак тепер існує спосіб захистити банківську програму "зсередини" — за допомогою технології шилдування. Ця технологія забезпечує безпечну роботу програми навіть у потенційно скомпрометованому середовищі, наприклад, на зламаному пристрої. Робота програми припиняється лише у найнебезпечніших ситуаціях [87].

Впровадьте багаторівневий захист програм від шкідливого програмного забезпечення та інших загроз. Це рішення забезпечує надійний захист банківських систем від "троянів", оверлей-атак, клавіатурних "шпигунів", ін'єкцій коду тощо. Виявляйте та блокуйте інструменти зловмисників:

• налагоджувачі;

• емулятори;

• перехоплювачі;

• запобігайте загрозам, що створюються зламаними пристроями, та реалізуйте безпечне середовище виконання відповідно до стандартів PSD2;

• інтегруйте інструменти для захисту мобільних фінансових сервісів від шкідливого коду відповідно до вимог FFIEC;

• використовуйте інструменти моніторингу цілісності програм та захисту даних користувачів, щоб забезпечити відповідність вимогам Загального регламенту захисту даних та інших норм.

2.5. Тестування безпеки мобільних застосунків

Тестування безпеки мобільного додатка передбачає тестування мобільного додатка таким чином, щоб зловмисний користувач спробував його атакувати. Ефективне тестування безпеки починається з розуміння бізнес-цілей програми та типів даних, які вона обробляє. Звідти поєднання статичного аналізу, динамічного аналізу та тестування на проникнення призводить до ефективної цілісної оцінки для пошуку вразливостей, які були б упущені, якби ці методи не використовувалися разом ефективно [114].

Процес тестування включає:

- Взаємодію з програмою та розуміння того, як вона зберігає, отримує та передає дані.
- Розшифровка зашифрованих частин програми.
- Декомпіляція програми та аналіз отриманого коду.
- Використання статичного аналізу для визначення слабких місць безпеки в декомпільованому коді.
- Застосування знань, отриманих під час зворотного проектування та статичного аналізу, для динамічного аналізу та тестування на проникнення.

- Використання динамічного аналізу та тестування на проникнення для оцінки ефективності засобів контролю безпеки (наприклад, елементів керування автентифікацією та авторизацією), які використовуються в програмі.

Існує низка безкоштовних і комерційних інструментів безпеки мобільних додатків, які оцінюють додатки за допомогою методологій статичного або динамічного тестування з різним ступенем ефективності. Однак жоден інструмент не забезпечує комплексної оцінки програми. Щоб забезпечити найкраще покриття, потрібна комбінація як статичного, так і динамічного тестування з переглядом вручну.

Тестування безпеки мобільних додатків можна розглядати як перевірку перед виробництвом, щоб переконатися, що елементи керування безпекою в додатку працюють належним чином, одночасно захищаючи від помилок впровадження. Це може допомогти виявити крайні випадки (які перетворюються на помилки безпеки), яких команда розробників, можливо, не передбачала. У процесі тестування враховуються проблеми як з кодом, так і з конфігурацією в середовищі, подібному до робочого, щоб переконатися, що проблеми виявлено перед запуском [116].

Використання мобільних пристроїв є повсюдним і затьмарило традиційні обчислення. Ми часто чуємо, як в обіг випускають різні шкідливі мобільні додатки. З цих причин розробка мобільних додатків повинна зосереджуватися на кібербезпеці так само, як і на функціональності та гнучкості, якщо не більше. Це неминучий аспект розробки додатків, до якого слід ставитися більш серйозно, оскільки дуже реальні загрози для бізнесу поширюються. Занадто багато компаній применшують безпеку під час розробки та обслуговування додатків передового досвіду через потребу задовольнити постійно зростаючий попит на мобільні бізнес-програми. Це ризик, який вони не можуть собі дозволити.

Організації повинні підвищити свою обізнаність про багато відомих і добре зрозумілих загроз безпеці для мобільних програм під час розробки та виробництва. Підприємства будь-якого розміру та типу повинні підвищувати свою обізнаність про безпеку та пов’язані з нею загрози під час і після розробки програми. Середовища розробки можуть бути скомпрометовані різними способами, включно з

неправильною конфігурацією безпеки, незахищеними методами зберігання даних, розголошенням конфіденційних даних, невиправленим програмним і апаратним забезпеченням і неналежним джерелом засобів розробки. Зловмисники все частіше переорієнтовують свою увагу з традиційних мережевих середовищ на мобільні програми з більш складними атаками, часто спрямованими на «фішинг» або використання слабких місць безпеки в програмі чи інструментах розробки. Невпинна увага до кібербезпеки під час розробки мобільних додатків може тримати організацію на крок попереду цих загроз, щоб забезпечити більшу гнучкість і зручність у додатках. Така увага до безпеки також підвищить рентабельність інвестицій у мобільні програми. Кібербезпека покращує можливості мобільних додатків

Мобільні програми потребують кількох ключових можливостей, щоб забезпечити організаціям операційну ефективність і продуктивність, забезпечуючи стабільну продуктивність за будь-яких сценаріїв загроз. Апаратне забезпечення мобільних пристроїв, наприклад камери та сканери відбитків пальців, слід використовувати для підвищення кібербезпеки в світі, що постійно працює. Деякі приклади включають біометричні засоби контролю доступу, як розпізнавання обличчя, сканування відбитків пальців і двофакторну автентифікацію. Програми мають працювати без сигналів Wi-Fi або стільникового зв’язку, щоб підтримувати продуктивність користувача навіть у разі збою звичного з’єднання. І, звісно, мобільні додатки мають успішно працювати на будь-якій операційній системі чи мобільному пристрої, зберігаючи при цьому постійну взаємодію та безпеку.

Кібербезпека є критично важливою для бізнесу, щоб запобігти витоку даних і несанкціонованому доступу до активів конфіденційних даних. Зламана мобільна програма цілком може надати зловмисникам доступ до цих активів або можливість перевести користувачів в автономний режим. Уразливі місця безпеки в мобільних програмах можуть дозволити зловмисникам використовувати або платформу програми, або операційну систему платформи мобільної програми, щоб отримати доступ і викрасти конфіденційну інформацію. Уразливості безпеки в базовому середовищі Wi-Fi, особливо в середовищах «роботи з дому», також можуть використовуватися кіберзлочинцями, щоб отримати доступ до конфіденційної бізнес-

інформації. Слабкі місця безпеки базових мобільних програм також можуть використовуватися для викрадення інформації автентифікації для наступних атак на програми та бізнес-системи.

Розробники можуть уникнути цих проблем, враховуючи кібербезпеку на кожному етапі розробки мобільних додатків. Методи, які слід розглянути, включають зашифровані бази даних (із суворим керуванням ключами шифрування/дешифрування) і шифрування всіх даних під час передачі через загальнодоступні мережі. Ці методи гарантують, що навіть якщо хакер проникне в додаток або мережу, будь-які вкрадені дані будуть нечитабельними. Крім того, відповідні методи шифрування також можна використовувати для підписання та позначення часу всіх змін корпоративних даних, що може бути корисним для юридичних цілей або у випадку відновлення втрачених або пошкоджених баз даних.

Незахищений код є основною проблемою кібербезпеки при розробці мобільних додатків. Злочинці зазвичай використовують погано розроблений або запрограмований код, щоб заразити базові мобільні додатки та використовувати їх у підлих цілях, зокрема для викрадення конфіденційних даних або вимагання непомірних викупів (зараз у мільйони доларів за успішну атаку).

Під час розробки мобільних додатків підприємствам слід завжди застосовувати найкращі методи безпеки, зокрема ручне або автоматичне сканування коду для виявлення загальних слабких місць безпеки, як незахищені бібліотеки, інструменти розробки без виправлень, порушення стандартів розробки, незахищений сторонній код і суворі стандарти кодування, тестування та оновлення виробничих бібліотек.

Програмне забезпечення для розробки мобільних додатків із низьким або без коду може допомогти, особливо під час створення додатків на основі завдань для транзакційних систем малого бізнесу, веб-додатків і аналітичних додатків. Ці програмні рішення є надійними, оскільки не вимагають значної участі IT для створення основної програми, і часто мають потужні вбудовані можливості та стандарти безпеки. Однак для керування цими типами рішень для розробки

мобільних додатків потрібен певний рівень технічної експертизи, включаючи кібербезпеку та інтеграцію з іншими критично важливими системами.

Програми з низьким кодом/без коду оптимізують процеси перевірки безпеки, забезпечуючи інтеграцію коду безпеки з системою на ранніх стадіях циклу розробки з частими оновленнями. Наявність конвеєрів автоматизації з перевіркою коду безпеки та вбудованим тестуванням допомагає оптимізувати процес перевірки. Це гарантує кращу плавність розробки додатків і те, що найкращі методи кібербезпеки завжди дотримуються та бездоганно вбудовуються в код.

Проблеми з автентифікацією роблять мобільні програми вразливими до порушень безпеки. Індустрія розробки мобільних додатків досліджує потенціал безпарольних рішень, досліджуючи біометрію та двофакторну автентифікацію як альтернативи перевірці облікових даних. Організаціям і розробникам, які ще не знайомі з безпарольним маршрутом, слід переконатися, що мобільний додаток призначений лише для прийому надійних літерно-цифрових паролів.

Якщо програми дуже конфіденційні, найнадійніші практики автентифікації, які відповідають бізнес-практикам, повинні бути ключовими моментами розробки мобільних програм. У зв’язку з тим, що проблеми з додатками все більше привертають увагу, високорівнева автентифікація має збільшуватися через необхідність.

У майбутньому розробники мобільних додатків повинні займатися кібербезпекою. Порушення даних може завдати фінансової шкоди організаціям, незалежно від типу чи причини. Все більше організацій розуміють потребу в найкращих практиках кібербезпеки та повинні включати ці практики в кожен елемент процесу розробки. Однак для керування цими типами рішень для розробки мобільних додатків потрібен певний рівень технічної експертизи, включаючи кібербезпеку та інтеграцію з іншими критично важливими системами.

Висновки за розділом 2

Наведений перелік вразливостей мобільних додатків може стати вихідною точкою для організацій, розробників, експертів з безпеки та користувачів, які тільки починають вирішувати відповідні проблеми. У цьому розділі були надані короткі описи для кожного типу вразливості, приклади сценаріїв атак, методи їх запобігання та рекомендовані стратегії для мінімізації шкоди.

Використання мобільного зв'язку збільшується, де хакери намагаються викрасти чутливу інформацію та компромісувати безпеку програми. З надійною стратегією безпеки мобільного зв'язку та найпопулярнішим мобільним розробником за можливістю підтримувати й негайно реагувати на загрози та помилки, програма буде захищеною, більш надійним місцем для користувачів.

Додатки повинні охоплювати ризики, надані загрозами Cybersecurity та порушення даних під час здійснення своїх мобільних додатків. Проводячи вищезгадані способи безпеки мобільного додатка, вони будуть здатні забезпечити як програми, так і дані всередині. Ці методи не є складними для виконання. Додатки та розробники вимагають взяти глобальний підхід до розробки додатків, і повинні мати справу з усіма обставинами, які впливають на безпеку програми.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ЕЛЕМЕНТІВ ЗАХИСТУ МОБІЛЬНОГО ЗАСТОСУНКУ GANG!

3.1 Реалізовані методи захисту

На основі проведеного аналізу автором цієї работи було реалізовано деякі елементи захисту мобільного застосунку Gang! На рис. 3.1,3.2,3.3 зображені реалізація баз даних для різних рівнів прав доступу.

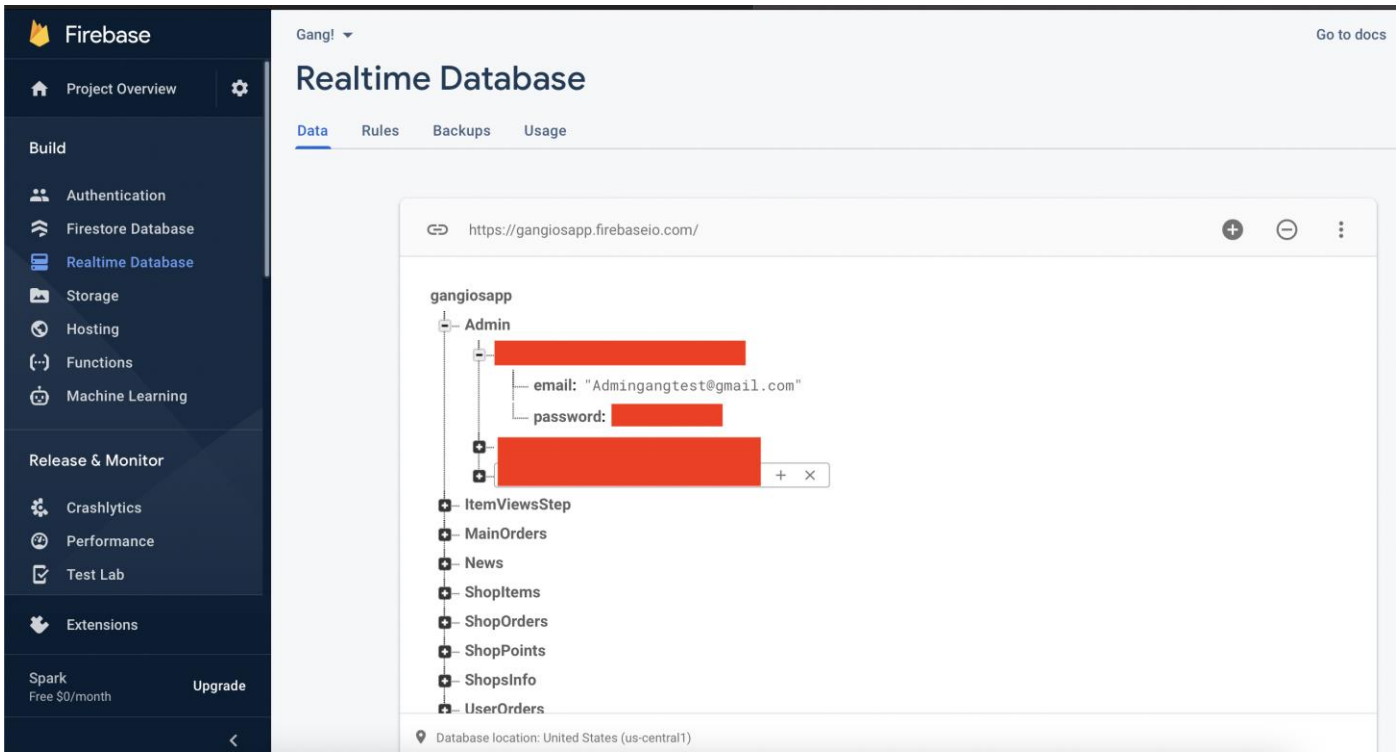


Рисунок 3.1 - Реалізація доступу до бази даних для адміністратора у мобільному застосунку Gang!

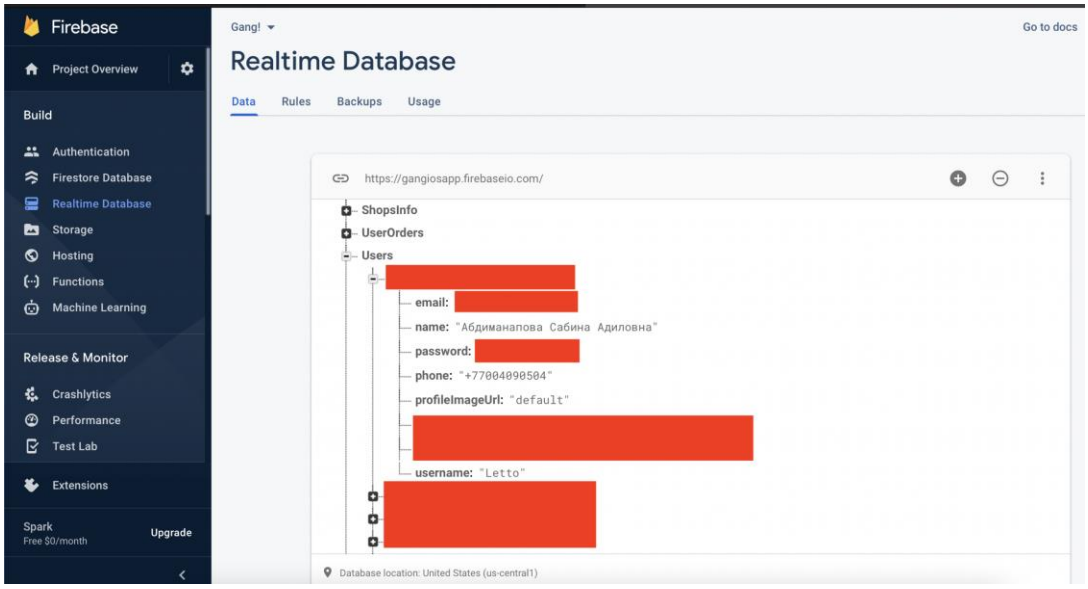


Рисунок 3.2 - Реалізація доступу до бази даних для користувача у мобільному застосунку Gang!

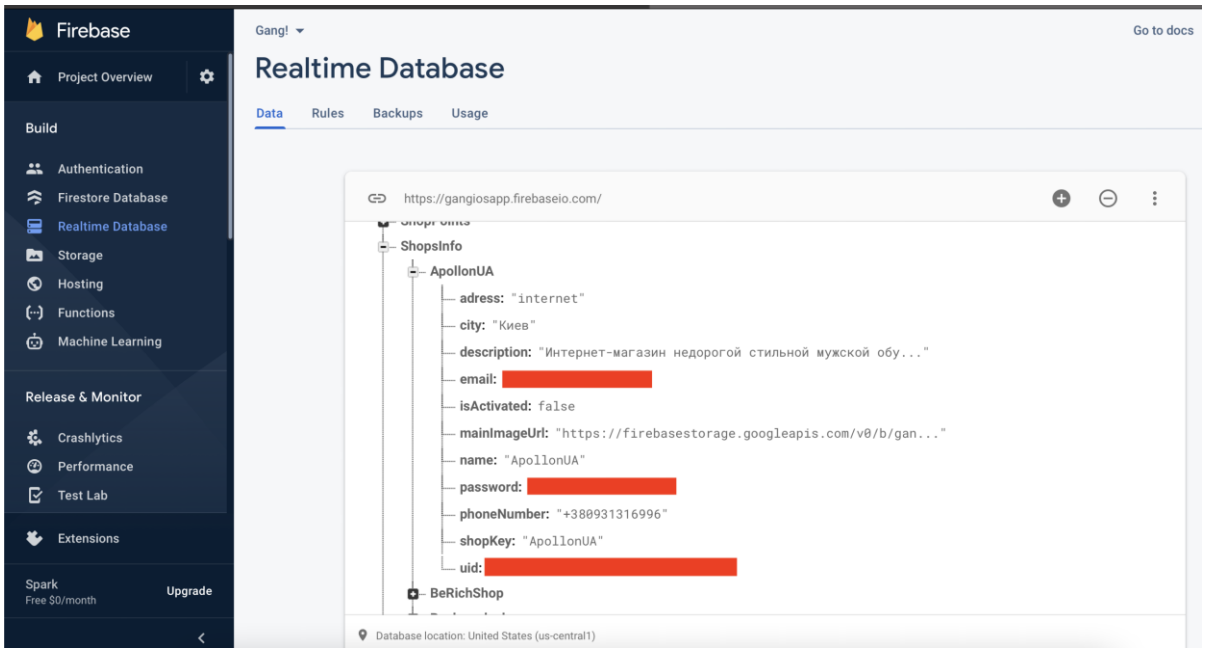


Рисунок 3.3 - Реалізація доступу до бази даних для менеджера магазину у мобільному застосунку Gang!

Ззахист пароля й персональних даних користувачів також реалізовані у мобільному застосунку та це є однією з головних елементів захисту мобільних застосунків. На рис. 3.4 зображено параметри та способи шифрування паролів.

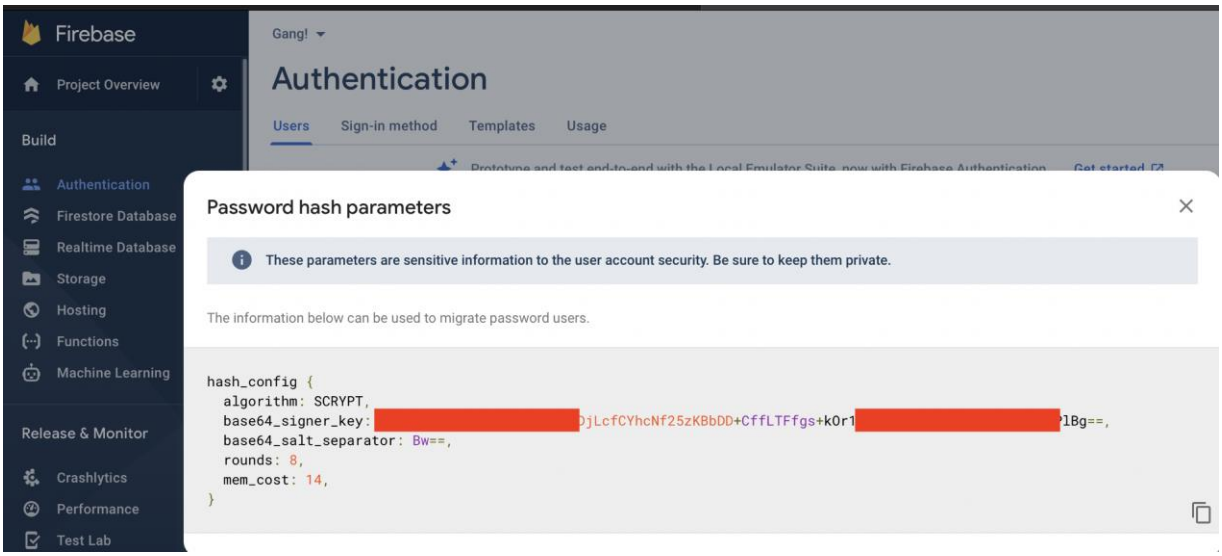


Рисунок 3.4 - Параметри пароля

Розмежуванню доступу було приділено особливу увагу, бо на цьому елементі захисту реалізується велика частина мобільного застосунку, тому що в Gang! є чотири види користувачів (гість, адміністратор, покупець, менеджер). На рис. 3.5 зображено приклад розмежування доступу до головної сторінки мобільного застосунку.

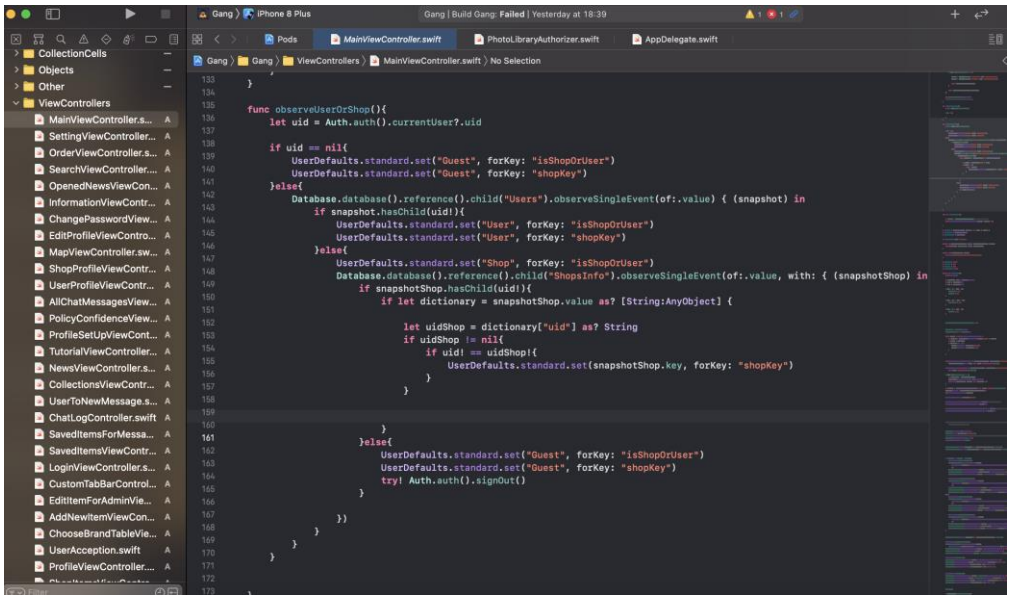


Рисунок 3.5 - Приклад розмежування доступу у мобільному застосунку Gang!

Будь-який мобільний застосунок повинен мати засіб захисту такий як авторизація На рис. 3.6 зображено авторизаію у мобільному застосунку.

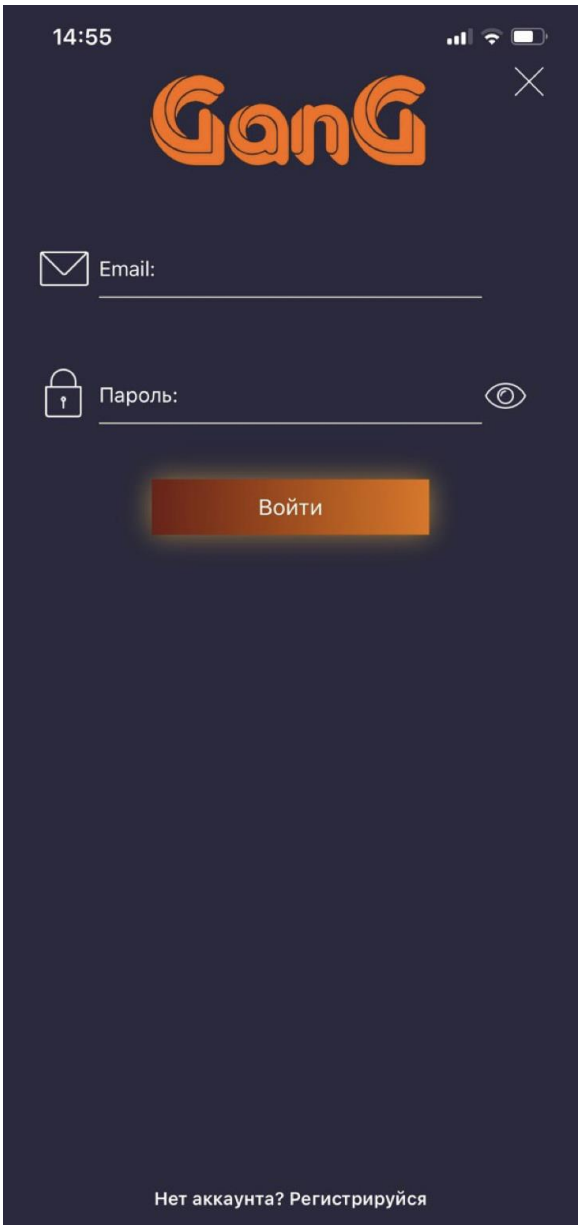


Рисунок 3.6 - Авторизація у мобільному застосунку Gang!

Додатковий механізм захисту у мобільному додатку під час реєстрації облікового запису полягає у верифікації реального мобільного телефону за допомогою коду. На зображенні 3.7 показано захист процесу реєстрації користувача, де відбувається відправка коду на мобільний номер телефону.

Рисунок 3.7 - Реєстрація користувача за номером телефона

Найбільша частина програмного коду мобільного додатка реалізована у такий спосіб, що будь-які операції з базою даних, такі як читання або запис, пов'язані з унікальним ідентифікатором користувача (UID). На рис. 3.8 зображен приклад реалізації.

```
if Auth.auth().currentUser?.uid == nil{
    let tutorialView = LoginViewController()
    tutorialView.modalTransitionStyle = .flipHorizontal
    self.present(tutorialView, animated: true, completion: nil)
}
```

Рисунок 3.8 - Прив’язка дій до унікального ідентифікатора користувача

Якщо користувач має намір змінити пароль від аккаунту, було реалізовано таку можливість. Користувач повине перейти у налаштування аккаунту, зміна паролю та обовязково треба вписати поточний пароль та два рази новий. Система перевіряє чи є у користувача така можливість, дає запит до бази даних та перезаписує у випадку якщо все добре.

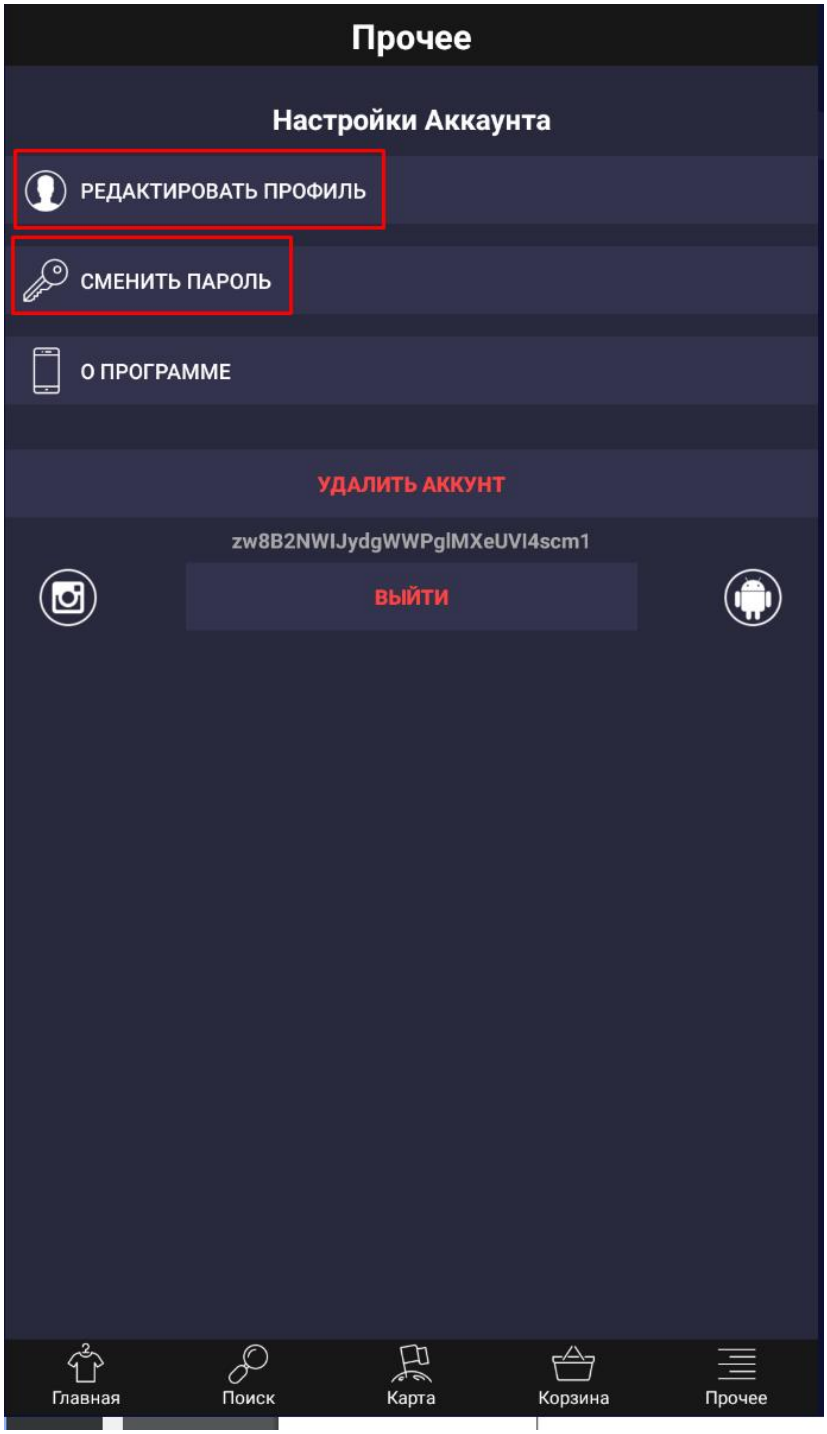


Рисунок 3.9 – Зміна конфіденційних даних

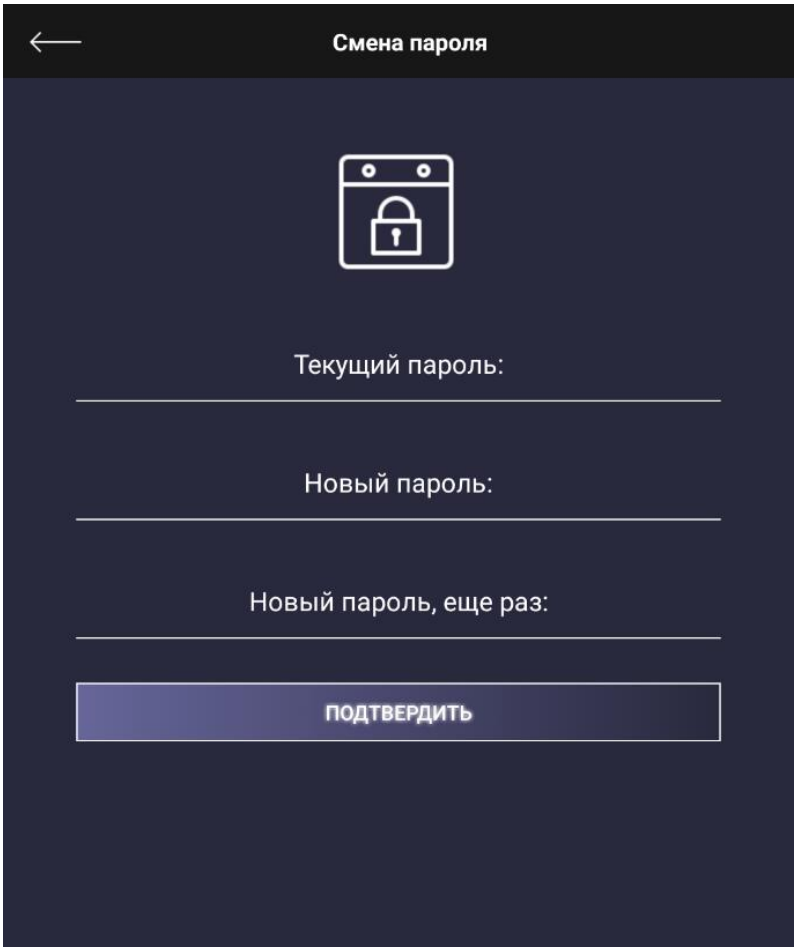
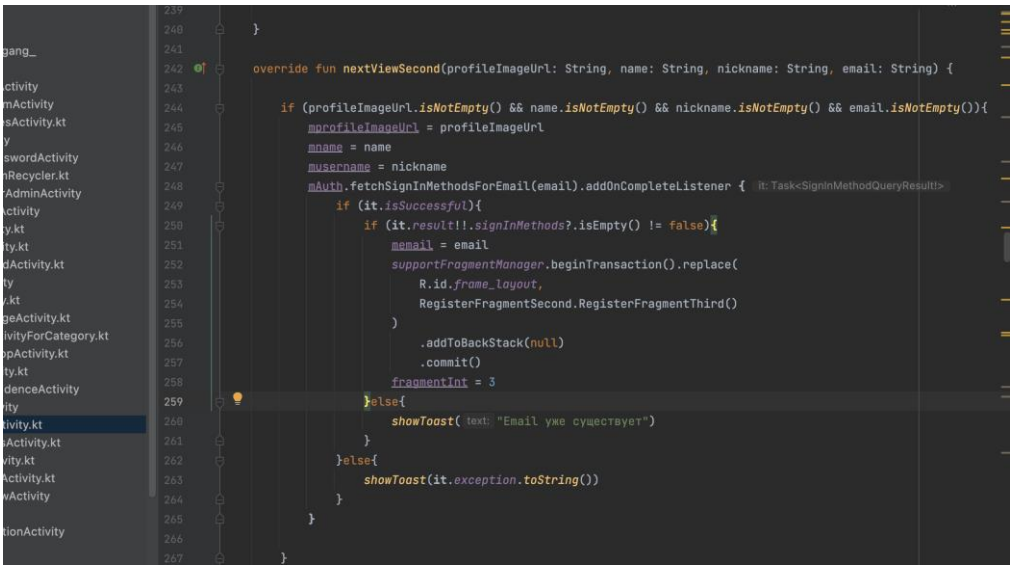


Рисунок 3.10 – Зміна паролю



```
}

override fun nextViewSecond(profileImageUrl: String, name: String, nickname: String, email: String) {

    if (profileImageUrl.isNotEmpty() && name.isNotEmpty() && nickname.isNotEmpty() && email.isNotEmpty()){
        mprofileImageUrl = profileImageUrl
        mname = name
        musername = nickname
        mAuth.fetchSignInMethodsForEmail(email).addOnCompleteListener {  it: Task<SignInMethodQueryResult>
            if (it.isSuccessful){
                if (it.result!!.signInMethods?.isEmpty() != false){
                    memail = email
                    supportFragmentManager.beginTransaction().replace(
                        R.id.frame_layout,
                        RegisterFragmentSecond.RegisterFragmentThird()
                    )
                        .addToBackStack(null)
                        .commit()
                    fragmentInt = 3
                }else{
                    showToast( text: "Email уже существует")
                }
            }else{
                showToast(it.exception.toString())
            }
        }

    }
```

Рисунок 3.11 - Програма має легку перевірку на існуючий емейл, задля того щоб не було дублікатів

3.2 Тестування системи

Для випробування системи використовувалось функціональне тестування, що полягає у перевірці реалізованості функціональних вимог програмного забезпечення.

Тестування мобільних додатків для Android та iOS проводилося вручну.

Модульні тести в Android можна розділити на два типи:

локальні модульні тести - це тести, що виконуються лише на віртуальній машині Java (JVM). Вони призначені для перевірки бізнес-логіки, яка не взаємодіє з операційною системою Android;

інструментальні модульні тести - це тести, які використовують Android API для перевірки логіки. Вони виконуються на фізичному пристрої або емуляторі.

Багата частина логіки додатка пов'язана з взаємодією з API системи Android, тому головним чином використовувалися інструментальні модульні тести.

Для модульного тестування використовувалася бібліотека JUnit. Ця бібліотека призначена для тестування Java-програм та має широкі можливості, такі як розподіл даних і логіки тесту, групування тестів, використання анотацій для опису тестів, обробка винятків, що виникли під час тестування, тощо.

Оскільки в процесі реалізації програми було прийнято рішення відмовитися від ORM-бібліотек, було написано багато коду для роботи з базою даних. Це може призвести до помилок, пов'язаних з використанням прекомпільованих SQL-виразів. Тому було приділено особливу увагу тестуванню компонента, що відповідає за роботу з базою даних.

Висновки за розділом 3

- На основі аналізу методів захисту було розроблено наступне:
- створено сервер системи "Gang!" з використанням платформи Firebase;
- розроблено мобільний додаток для платформи iOS та Android;
- реалізовано механізм автентифікації за допомогою електронної пошти та пароля;
- реалізовано процес реєстрації облікового запису за номером телефону;
- впроваджено систему обмеження доступу до інформації;
- реалізовано механізм прив'язки дій до унікального ідентифікатора користувача (UID);
- застосовано механізм хешування паролів.

ВИСНОВКИ

У даній дипломній роботі було проведено дослідження механізмів захисту мобільних додатків з метою реалізації власного захищеного мобільного застосунку "Gang!". На етапі початкового дослідження було визначено основні терміни та поняття, а також виявлено потенційні загрози та вразливості, які можуть існувати в мобільних додатках. Після аналізу вразливостей було досліджено методи захисту та розглянуто їх можливі реалізації.

Також спроектовано архітектуру системи мобільного додатку, бази даних серверу під платформи IOS/Android. Додано FCM-повідомленя та реалізона взаємодію сервера і мобільного застосунку за допомогою HTTP-запитів. Крім того розроблено розмежування доступу до інформації, а також преведений аналіх і захист програмного коду.

На основі аналізу механізмів захисту мобільних застосунків реалізовано власний мобільний застосунок Gang!, де враховано всі основні вразливості та механізми захисту мобільних застосунків. Реалізовано механізми захисту як клієнта так і серверної частини.

Проведена реалізація захисту персональних даних, розмежування доступів до інформації та розроблено хешування паролів. Сервер мобільного додатку реалізован за допомогою платформи Google Firebase, розроблено механізми авторизації та прив’язка дій до унікального ідентификатору користувача.

Системний код складається з понад 40000 рядків коду на мові Swift, понад 30000 рядків на мові Kotlin і приблизно 6000 рядків мови розмітки XML.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. F. Marturana, G. Me, R. Berte, and S. Tacconi, “A Quantitative Approach to Triaging in Mobile Forensics,” in Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on, Nov., pp. 582–588.
2. Книга Вячеслава Семенчука «Мобильное приложение как инструмент бизнеса»
3. Додаток в App Store «Kasta».
4. Додаток в App Store «OLX».
5. Додаток в App Store «Prom».
6. Додаток в App Store «AliExpress».
7. Додаток в App Store «Alibaba».
8. Додаток в App Store «Wildberries».
9. Додаток в App Store «Farfetch».
10. Додаток в App Store «Lamoda».
11. Малишкіна Е.А. Удосконалення маркетингових інструментів в інтернет-бізнесі як фактор найбільш ефективного впливу на споживача. // Соціально-економічні явища і процеси.
12. Firebase. [Електронний ресурс] URL: https://firebase.google.com/docs
13. Ведення в Firebase. [Електронний ресурс] URL: https://habr.com/ru/post/277941/
14. Console Firebase. [Електронний ресурс] URL: https://console.firebase.google.com
15. Ешіна С.С. Порівняльний аналіз ORM-бібліотек для платформи Android на основі критерію продуктивності. // Інформатика: проблеми, методологія, технології матеріали XV міжнародної науково-методичної конференції .
16. Цеглярів А.П., Ляшева С.А., Шлеймовіч М.П., Єремєєв Д. Є. Автоматизована система взаємодії користувачів з базою даних за допомогою додатків для мобільних пристроїв.
17. Як захиститись від кіберзагроз. [Електронний ресурс] URL: https://hub.kyivstar.ua/news/yak-rytejleram-zahystyty-biznes-vid-kiberzagroz/
18. OneSignal. [Електронний ресурс] URL: https://onesignal.com .

19. OneSpan App Shielding. [Електронний ресурс] URL: https://www.onespan.com/ru/products/application-shielding/features
20. OneSpan Mobile App Shielding. [Електронний ресурс] URL: https://www.onespan.com/resources/mobile-app-shielding/datasheet
21. OneSpan App Shielding Datasheet. [Електронний ресурс] URL: https://www.onespan.com/sites/default/files/2022-10/app-shielding-datasheet.pdf
22. Розробка мобільного додатку. [Електронний ресурс] URL: https://studentlib.com/chitat/diplom-203181-razrabotka_mobilnogo_prilozheniya_informacionnoy_podderzhki_deyatelnosti_servis_inzhenera.html
23. Безпека даних в розробці мобільних додатків. [Електронний ресурс] URL: https://habr.com/ru/post/327760/
24. Безпека застосунків. [Електронний ресурс] URL: https://wikipedia.org/wiki/Безопасность_приложений
25. Харді Б., Філліпс Б. Програмування під Android.
26. Шевченко Д. версійна міграція структури бази даних: ос новні підходи.
27. Android Developers Dashboard. [Електронний ресурс] URL: https://developer.android.com/about/dashboards/index.html
28. Android Development Tool. [Електронний ресурс] URL: https://developer.android.com
29. Firebase Cloud Messaging. [Електронний ресурс] URL: https://firebase.google.com/docs/cloud-messaging/
30. Material design guidelines SendGrid. [Електронний ресурс] URL: https://material.io/guidelines/material-design/introduction.html
31. OpenShift: PaaS by Red Hat. [Електронний ресурс] URL: https://www.openshift.com
32. Beginning IPhone Development with Swift: Exploring the IOS SDK
33. Swift. Основи розробки додатків під iOS, iPadOS і macOS Василь Усов
34. SQLite database engine. [Електронний ресурс] URL: https://www.sqlite.org/

35. Безпека і кібербезпека смартфонів. [Електронний ресурс] URL: https://datami.ua/bezpeka-i-kiberbezpeka-smartfoniv/
36. Mobile App Security. [Електронний ресурс] URL: https://www.ptsecurity.com/ru-ru/research/analytics/mobile-application-security-threats-and-vulnerabilities-2019/
37. Мобільний застосунок Gang. [Електронний ресурс] URL: https://play.google.com/store/apps/details?id=alexey.poedinok.gang_
38. 10 Most common app security mistakes. [Електронний ресурс] URL: https://smartface.io/10-most-common-app-security-mistakes/
39. Вразливості мобільних додатків. [Електронний ресурс] URL: https://cryptoworld.su/uyazvimosti-mobilnyx-prilozhenij-i-ix-ustranenie/#4
40. Microsoft: what is cybersecurity. [Електронний ресурс] URL: https://www.microsoft.com/ru-ru/security/business/security-101/what-is-cybersecurity
41. Dissercat Методика захисту інформації. [Електронний ресурс] URL: https://www.dissercat.com/content/metodika-zashchity-informatsii-v-besprovodnykh-setyakh-na-osnove-dinamicheskoi-marshrutizats
42. Lookout, “Mobile Lost and Found.” [Електронний ресурс] URL: https://www.mylookout.com/resources/reports/mobile-lost-and-found.
43. Lookout, “Phone Theft in America.” [Електронний ресурс] URL: https://www.lookout.com/resources/reports/phone-theft-in-america.
44. C.-T. Li, “Source camera identification using enhanced sensor pattern noise,” Information Forensics and Security, IEEE Transactions on, vol. 5, no. 2, pp. 280–287, 2010.
45. Lookout, “Lookout Mobile Threat Report.” [Електронний ресурс] URL: https://www.lookout.com/resources/reports/mobile-threat-report.
46. Wikipedia, “iOS version history.” [Електронний ресурс] URL: http://en.wikipedia.org/wiki/History_of_iOS#Version_history:_iPhone.2C_iPad.2C_and_iPod_Touch
47. Wikipedia, “Android version history.” [Електронний ресурс] URL: http://en.wikipedia.org/wiki/Android_version_history.

48. P. Thomas, P. Owen, and D. McPhee, “An Analysis of the Digital Forensic Examination of Mobile Phones,” in Next Generation Mobile Applications, Services and Technologies (NGMAST), 2010 Fourth International Conference on, July, pp. 25–29.

A. Hoog, Android Forensics: Investigation, Analysis and Mobile Security for Google Android. Syngress, 2011.

49. J. Sylve, A. Case, L. Marziale, and G. G. Richard, “Acquisition and analysis of volatile memory from android devices,” Digital Investigation, 2011.

50. V. L. L. Thing, K. Y. Ng, and E. C. Chang, “Live memory forensics of mobile phones,” digital investigation, vol. 7, pp. S74–S82, 2010.

51. C. Miller and C. Mulliner, “Fuzzing the Phone in Your Phone,” presented at the Black Hat USA, 2009.

52. M. Sutton, A. Greene, and P. Amini, Fuzzing: Brute Force Vulnerability Discovery. Addison-Wesley Professional, 2007.

53. DFRWS, “A Road Map for Digital Forensic Research,” Report from the 1st Digital Forensic Research Workshop (DFRWS), 2001, p. 16.

54. F. Adelstein, “Live forensics: diagnosing your system without killing it first,” Communications of the ACM, vol. 49, no. 2, pp. 63–66, 2006.

55. J. M. Urrea, “An analysis of Linux RAM forensics,” Monterey, California. Naval Postgraduate School, 2006.

56. IEEE, “IEEE Standard Glossary of Software Engineering Terminology,” IEEE Std 610.12-1990, p. 1, 1990.

57. J. Seitz, Gray Hat Python: Python Programming for Hackers and Reverse Engineers. San Francisco, CA, USA: No Starch Press, 2009.

58. P. Amini, “Sulley Fuzzing Framework.” [Електронний ресурс] URL:https://github.com/OpenRCE/sulley.

59. P. Amini, “Fuzzing Software Collection.” [Електронний ресурс] URL: http://fuzzing.org/.

60. Aitel, The Advantages of Block-Based Protocol Analysis for Security Testing. 2002.

61. S. Schrittwieser, P. Fruehwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. Weippl, “Guess Who’s Texting You? Evaluating the Security of Smartphone Messaging Applications,” in Proceedings of the 19th Annual Network Distributed System Security Symposium (NDSS’12), San Diego, California, USA, 2012.
62. M. Egele, C. Kruegel, E. Kirda, and G. Vigna, “PiOS: Detecting Privacy Leaks in iOS Applications,” in Proceedings of the 18th Annual Network Distributed System Security Symposium (NDSS’11), 2011.
63. W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, “TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones,” in Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI’10), 2010.
64. Hoog, “Forensics Security Analysis of Google Wallet,” Dec-2011. [Електронний ресурс] URL:https://www.nowsecure.com/blog/2011/12/12/forensics- security-analysis-google-wallet/.
65. Google, “Google Wallet.” [Електронний ресурс] URL:https://www.google.com/wallet/.
66. Gartner, “Gartner Says Smartphone Sales Surpassed One Billion Units in 2014,” 03-Mar-2015. [Електронний ресурс] URL:http://www.gartner.com/newsroom/id/2996817.
67. Google, “Google Play.” [Електронний ресурс] URL:https://play.google.com.
68. “Android Debug Bridge.” [Електронний ресурс] URL:http://developer.android.com/guide/developing/tools/adb.html.
69. “JSON Format home page.” [Електронний ресурс] URL:http://www.json.org/.
70. CIPA, “Exchangeable Image File format (EXIF).” [Електронний ресурс] URL:http://www.cipa.jp/std/documents/e/DC-008-2012_E.pdf.
71. Ponemon Institute, “Study finds lack of investment in mobile app security.” [Електронний ресурс] URL:http://searchsecurity.techtarget.com/news/4500243095/Study-finds-lack-of-investment-in-mobile-app-security.

72. Cellebrite UFED. [Електронний ресурс] URL:http://www.cellebrite.com/mobile-forensics.
73. Oxygen Forensics. [Електронний ресурс] URL:http://www.oxygen-forensic.com/en/.
74. ENISA and OW ASP , “Smartphone Secure Development Guidelines,” Nov-
75. Smartphone secure development. [Електронний ресурс] URL:http://www.enisa.europa.eu/activities/Resilience-and- CIIP/critical-applications/smartphone-security-1/smartphone-secure- development-guidelines.
76. OWASP, “Top Ten Mobile Controls.” [Електронний ресурс] URL:https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#Top_Ten_ Mobile_Controls.
77. International Data Corporation (IDC), “Android and iOS Squeeze the Competition, Swelling to 96.3% of the Smartphone Operating System Market for Both 4Q14 and CY14.” [Електронний ресурс] URL:http://www.idc.com/getdoc.jsp?containerId=prUS25450615.
78. Kaspersky, “Financial cyberthreats in 2014,” Feb. 2015.
79. P. Stirparo and I. Kounelis, “The mobileak project: Forensics methodology for
80. mobile application privacy assessment,” in Internet Technology And Secured
81. Transactions, 2012 International Conferece For, Dec., pp. 297–303.
I. Kollár, “Forensic RAM dump image analyser,” Department of Software
82. Engineering, Charles University, Prague, 2010.
A. Case, A. Cristina, L. Marziale, G. G. Richard, and V. Roussev, “FACE:
83. Automated digital evidence discovery and correlation,” Digital Investigation, vol.
84. 5, Supplement, no. 0, pp. S65 – S75, 2008.
A. Case, L. Marziale, C. Neckar, and G. G. Richard III, “Treasure and tragedy
85. in kmem_cache mining for live forensics investigation,” Digital Investigation,
86. vol. 7, Supplement, no. 0, pp. S41 – S47, 2010.
A. Case, L. Marziale, and G. G. Richard III, “Dynamic recreation of kernel
87. data structures for live forensics,” Digital Investigation, vol. 7, Supplement, no. 0,

88. pp. S32 – S40, 2010.
J. Butler and J. Murdock, “Physical Memory Forensics for Files and Cache,”
89. BlackHat USA, 2011.
W. Mauerer, Professional Linux kernel architecture. John Wiley & Sons,
90. 2010.
The Volatility Framework. [Online]. Available:
91. Volatility [Електронний ресурс]
URL:http://www.volatilityfoundation.org/#!releases/component_71401.
92. S. Leppert, “Android Memory Dump Analysis,” Student Research Paper,
93. Chair of Computer Science, vol. 1, 2012.
Google, “Using DDMS.” [Електронний ресурс] URL:
http://developer.android.com/tools/debugging/ddms.html.
94. H. Macht, “Live Memory Forensics on Android with Volatility,” Friedrich-
95. Alexander University Erlangen-Nuremberg, 2013.
EmbeddedLinux, “Android Memory Usage.” [Електронний ресурс] URL:
http://elinux.org/Android_Memory_Usage.
96. Google, “Processes and Threads.”
97. Android Developer [Електронний ресурс]
URL:http://developer.android.com/guide/components/processes-and-threads.html.
98. “Android Emulator.” [Електронний ресурс] URL:
http://developer.android.com/tools/help/emulator.html. [Accessed: 29-Apr-2015].
99. “Android SDK.” . [Електронний ресурс] URL:
http://developer.android.com/sdk/index.html.
100. “Android NDK.” [Електронний ресурс] URL:
http://developer.android.com/tools/sdk/ndk/index.html.
101. J. Sylve, “Android Mind Reading,” in ShmooCon Security Conference, 2012.
J. Sylve, “LiME - Linux Memory Extractor.” [Електронний ресурс] URL:
https://github.com/504ensicslabs/lime.
102. Bluetooth SIG, “Bluetooth Specification.” [Електронний ресурс] URL:
https://www.bluetooth.org/en-us/specification/adopted-specifications.

103. Bluetooth SIG, “Bluetooth Special Interest Group.” [Електронний ресурс] URL: http://www.bluetooth.com.

104. Bluetooth SIG, “Bluetooth SIG Analyst Digest 2H 2014 - A summary of recent reports by industry analysts covering wireless technology,” Dec. 2014.

105. H. Dwivedi, C. _Clarck, and D. Thiel, Mobile Application Security. McGraw Hill, 2010.

106. Bluetooth SIG, “Bluetooth Specification: Core Versione 2.0 + EDR,” Nov- 2004. [Електронний ресурс] URL:https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=40560.

107. Bluetooth SIG, “Bluetooth Specification: Core Versione 2.1 + EDR,” Jul-2007. [Електронний ресурс] URL:https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=241363.

108. Bluetooth SIG, “Bluetooth Specification: Core Versione 3.0 + HS,” Apr-2009. [Електронний ресурс] URL:https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=174214 .

109. Bluetooth SIG, “Bluetooth Specification: Core Versione 4.0,” Jun-2010. [Електронний ресурс] URL:https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737.

110. W. Stallings, Wireless Communications and Networks, 2nd ed. Prentice Hall, 2004.

111. NIST, “Guide to Bluetooth Security (Draft), Special Pubblication 800-121, Rev. 1,” NIST, 2011.

A. Gehrmann, J. Persson, and B. Smeets, Bluetooth Security. Artech House, Inc., 2004.

S. Hay and R. Harle, “Bluetooth tracking without discoverability,” in 4th International Symposium on Location and Context Awareness, 2009, pp. 120– 137.

112. L. Carettoni, C. Merloni, and S. Zanero, “Studying bluetooth malware propagation: The bluebag project,” IEEE Security & Privacy, vol. 5, no. 2, pp. 17–25, 2007.

113. “Trifinite Group.” [Електронний ресурс] URL:http://trifinite.org/trifinite_group.html.

114. M. Herfurt and C. Mulliner, “Remote Device Identification based on Bluetooth Fingerprinting Techniques,” Trifinite Group, White Paper, 2004.

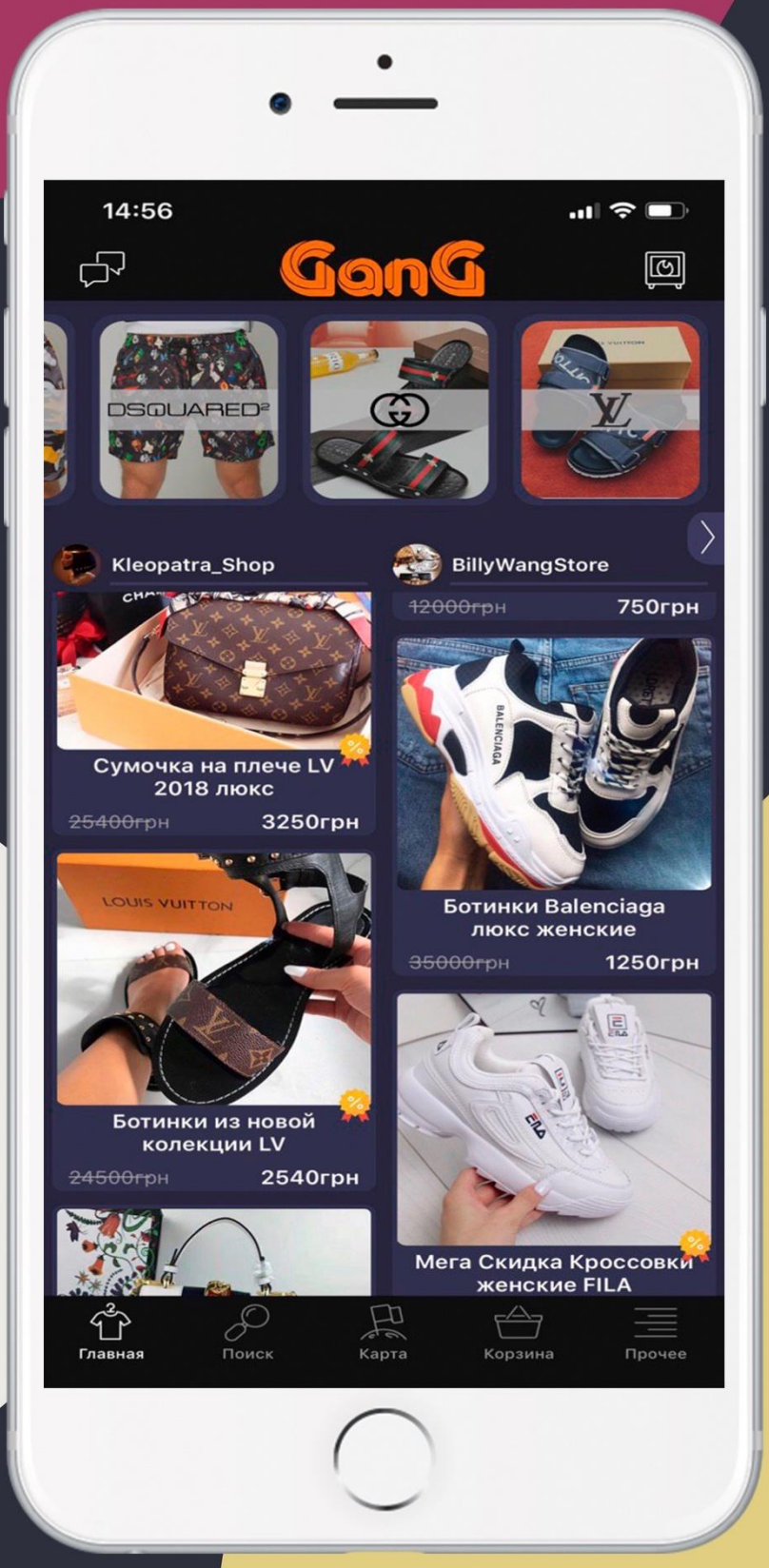
115. ISO/IEC 14443 - Identification cards - Contactless integrated circuit cards - Proximity cards. 2001.

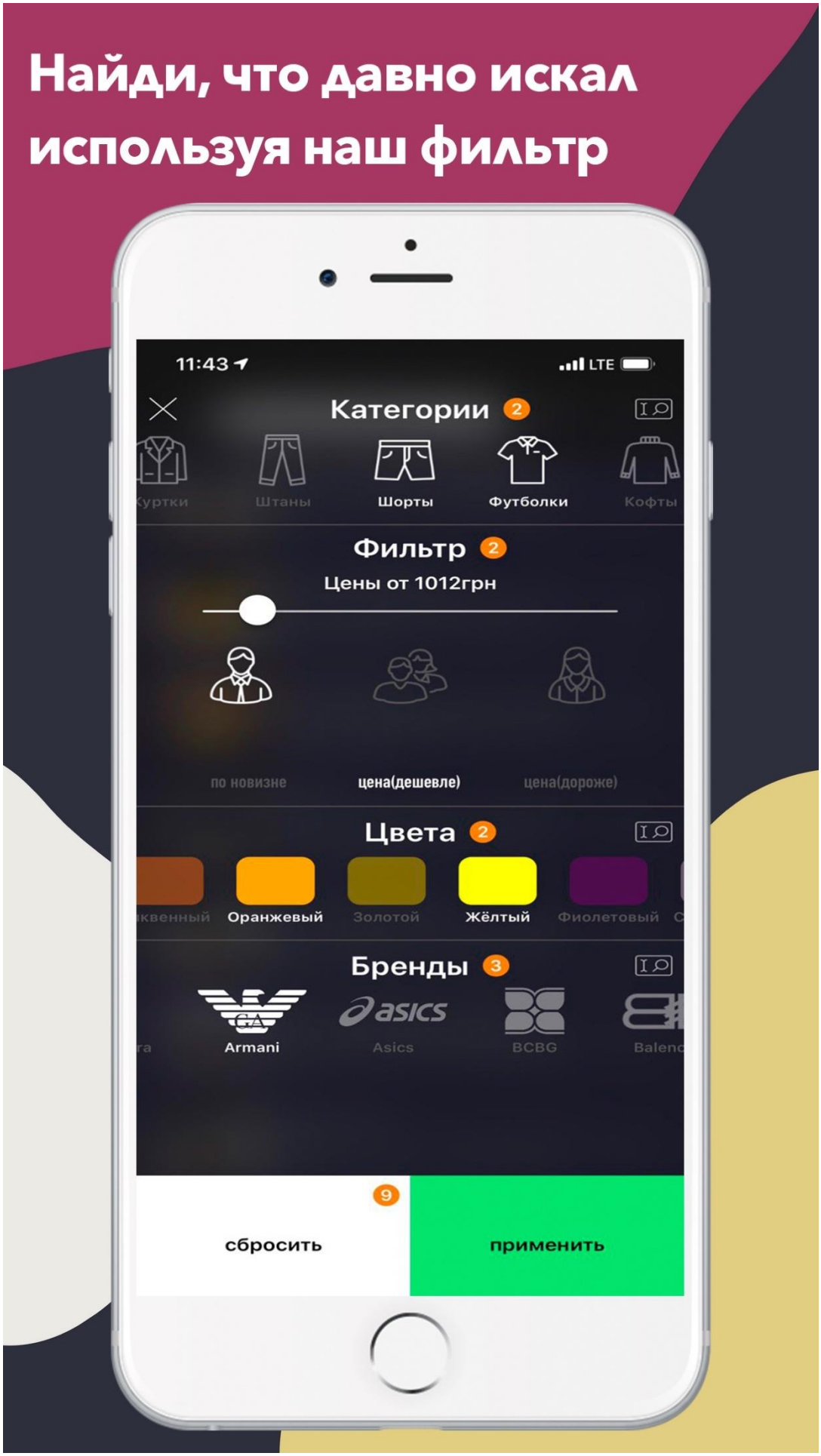
116. Miller, “Fuzz Testing of Application Reliability.” [Електронний ресурс] URL: http://pages.cs.wisc.edu/bart/fuzz/

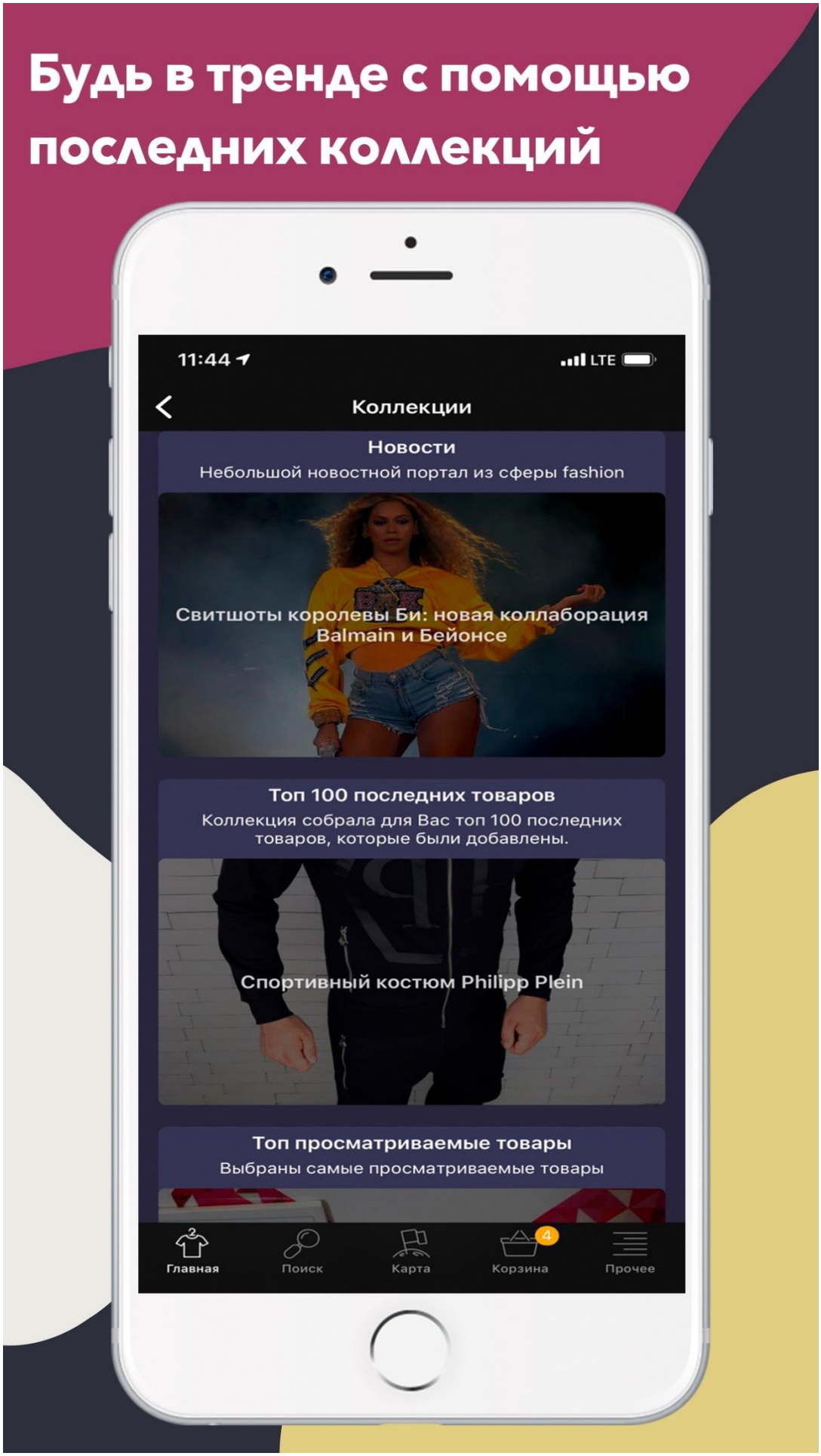
117. “Android API.” [Електронний ресурс] URL:http://developer.android.com/reference/packages.html

ДОДАТОК А

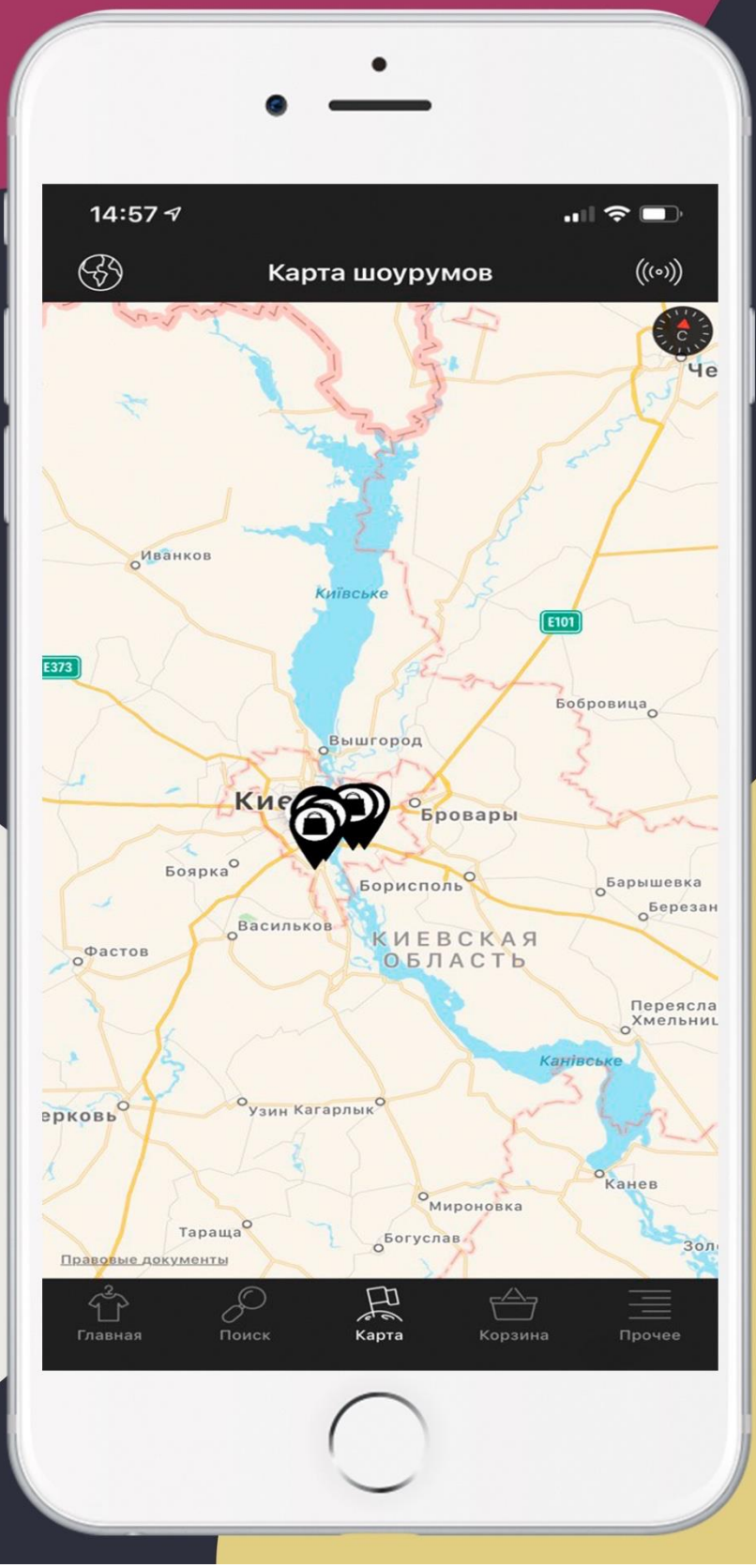
Удобная лента новинок



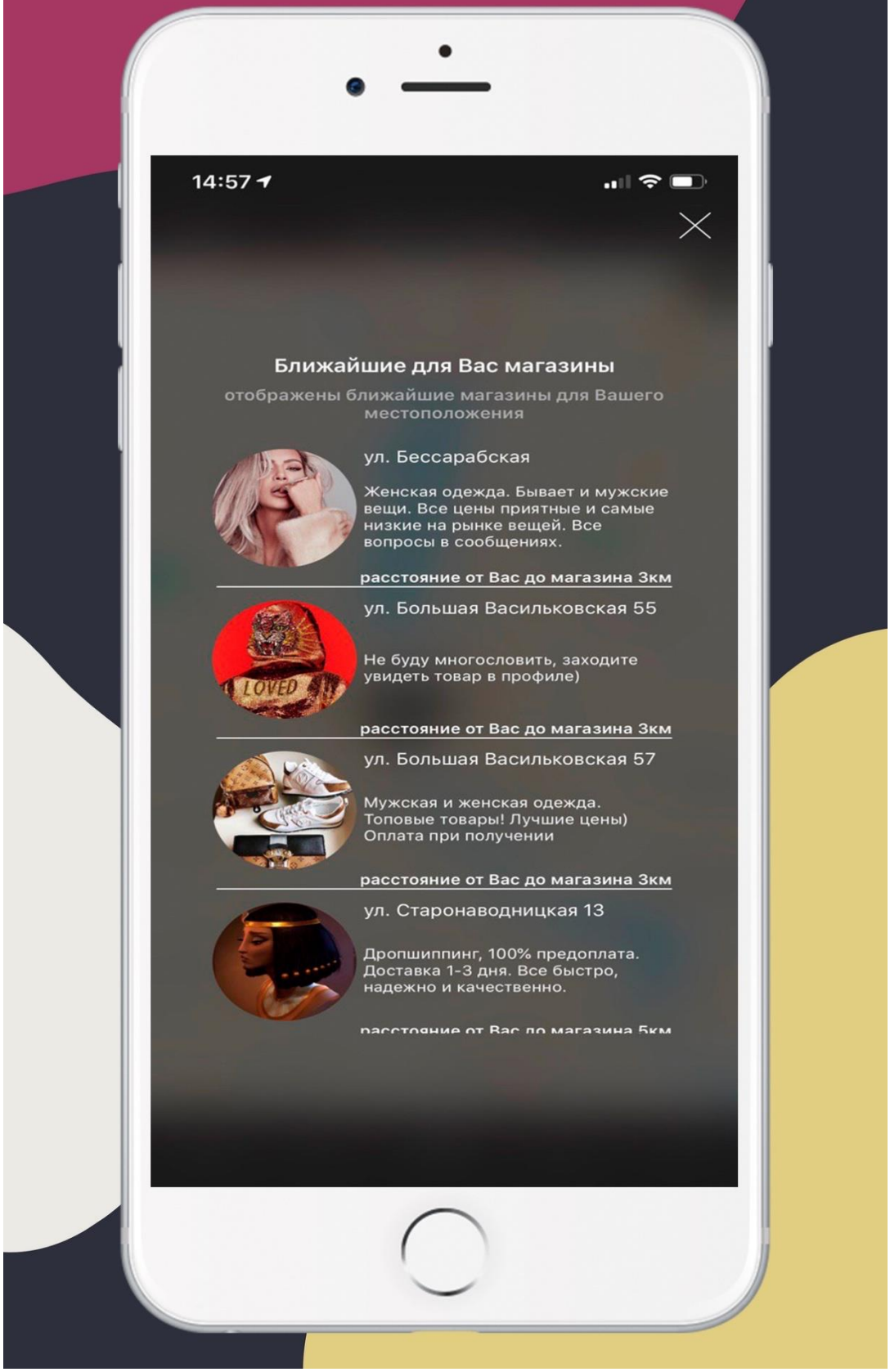

Найди, что давно искал
используя наш фильтр
11:43
LTE
Категории 2
Куртки
Штаны
Шорты
Футболки
Кофты
Фильтр 2
Цены от 1012грн
по новизне
цена(дешевле)
цена(дороже)
Цвета 2
Оранжевый
Золотой
Жёлтый
Фиолетовый
Бренды 3
Armani
Asics
BCBG
сбросить
9
применить


Будь в тренде с помощью последних коллекций
11:44
LTE
Коллекции
Новости
Небольшой новостной портал из сферы fashion
Свитшоты королевы Би: новая коллаборация Balmain и Бейонсе
Топ 100 последних товаров
Коллекция собрала для Вас топ 100 последних товаров, которые были добавлены.
Спортивный костюм Philipp Plein
Топ просматриваемые товары
Выбраны самые просматриваемые товары
Главная
Поиск
Карта
4
Корзина
Прочее

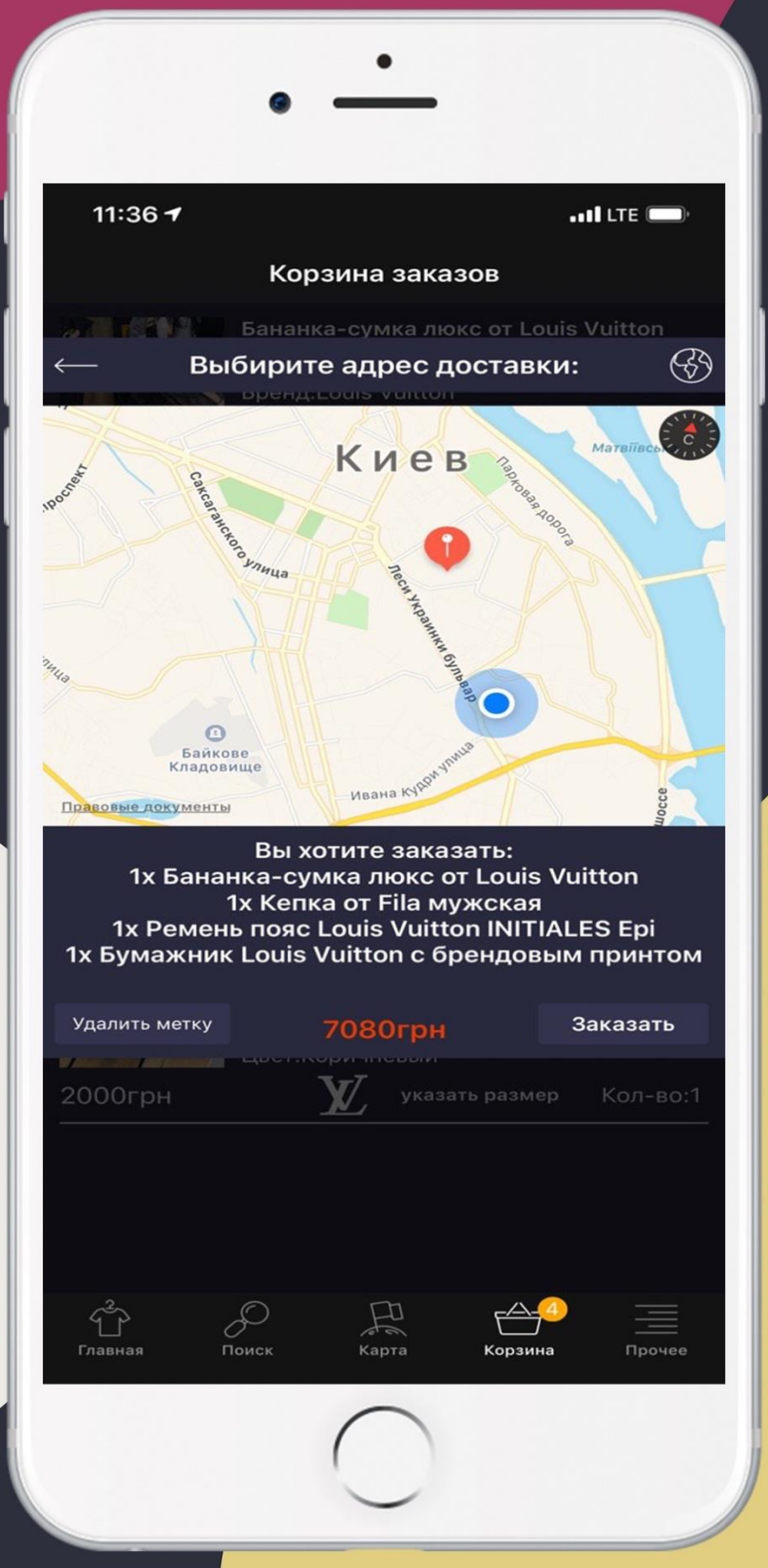
Используй карту для достижения своей цели



Выбери магазин поблизости



Нет времени на шоппинг?
Укажи адрес доставки



Уникальная система статусов заказа