Міністерство освіти і науки України
«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
Завідувач кафедри кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко

« 18 » червня 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

випускної кваліфікаційної роботи
бакалавра
(назва освітнього рівня)

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

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: Розробка елементів системи захисту інформації мобільного застосунку «Gang! – новий сленг молоді»

Виконавець: студент IV курсу, групи КБ-42

_____ **Поєдинок Олексій Миколайович**
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	**Підпис**
Керівник	Мирутенко Л.В.	

Нормоконтроль	Зюбіна Р. В.	

Київ 2021

Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____ Н. В. Лукова-Чуйко

«11» листопада 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

студенту **КБ-42** (група) **Поєдинку Олексію Миколайовичу** (прізвище ім'я по-батькові)

Тема дипломної роботи Розробка елементів системи захисту інформації Мобільного застосунку «Gang! – новий сленг молоді»

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол № 2 від 08.10.2020 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Специфікації мобільних загроз, технічний опис мобільного додатку Gang, технології захисту мобільного застосунку.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Аналіз основних вразливостей мобільних застосунків, аналіз методів захисту мобільних застосунків, розробка елементів захисту доступу до інформації в Gang!

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність розробка механізму перевірки коректності прав доступу до інформаційних ресурсів в Gang! в процесі повноцінної роботи мобільного застосунку.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 11 листопада 2020 року.

Завдання видала		Л. В. Мирутенко
	(підпис)	(ініціали, прізвище)
Завдання прийняв до виконання		О. М. Поєдинок
	(підпис)	(ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	25.01.2021 – 29.01.2021	*виконано*
2	Аналіз літератури	30.01.2020 – 11.02.2020	*виконано*
3	Обґрунтування вибору рішення	12.02.2020 – 15.02.2020	*виконано*
4	Розробка мобільного застосунку Gang!	16.02.2021 – 25.03.2021	*виконано*
5	Розробка механізмів захисту в мобільному застосунку Gang!	26.03.2021 – 10.04.2021	*виконано*
6	Виконання практичної частини	11.04.2021 – 17.05.2021	*виконано*
7	Оформлення пояснювальної записки	18.05.2021 – 08.06.2021	*виконано*
8	Підготовка до захисту дипломної роботи	09.06.2020 – 21.06.2021	*виконано*

Студент-дипломник		О. М. Поєдинок
	(підпис)	(ініціали, прізвище)
Керівник випускної кваліфікаційної роботи		Л.В. Мирутенко
	(підпис)	(ініціали, прізвище)

Термін подання дипломної роботи до ЕК 08 червня 2021 року

РЕФЕРАТ

Дипломна робота складається з переліку умовних позначень та скорочень, вступу, основної частини, що містить 3 розділи, висновків і списку використаних джерел. Загальний обсяг роботи – 67 сторінок. Список використаних джерел включає 40 наймінувань.

Метою даної роботи є розробка механізмів захисту інформації в мобільному застосунку Gang!

У роботі проаналізована сучасна науково-технічна література з захисту мобільних застосунків, виконано аналіз основних вразвливостей в мобільних застосунків, розроблено безпечний мобільний застосунок з наступними реалізованими механізмами захисту:

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

Ключові слова: мобільний застосунок, розмежування доступу, інформаційна безпека, політика безпеки, розмежування доступу за ролями, автентифікація, авторизація.

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 тим, що на даний момент практично кожен користувач мобільних телефонів стикався у своїй роботі з мобільними застосунками.

Безпека мобільних додатків швидко набула важливості, оскільки мобільні пристрої набули поширення в багатьох країнах і регіонах, але механізми захисту є слабкою ланкою мобільного застосунку. 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, які надають змогу реалізувати кібератаки.

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 несуть відповідальність не тільки розробники застосункій, а й власник мобільного пристрою.

Ризик пов'язаний не тільки з захистом клієнт-серверної архітектури, але й з уразливими місцями, які виникають під час взаємодії клієнт-серверу. Обмін даними з відкритих протоколів може повністю пошкодити переданий трафік. Однак навіть безпечні зв’язки не завжди є надійними, що свідчить про те, що розробники не приділяють достатньої уваги питанням безпеки.

Тому *темою роботи* є розробка механізмів захисту даних до інформації в мобільному застосунку Gang!

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

Предметом дослідження є механізми захисту інформації в мобільному застосунку "Gang! - новий сленг молоді"».

Методи дослідження — аналіз, синтез, методи імітаційного моделювання.

РОЗДІЛ 1 ПРОЕКТУВАННЯ ТА РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

1.1. Обзор аналогов

Kasta – модний маркетплейс

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

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. 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 маркетплейс

«Prom.ua» — український маркетплейс, проект IT-компанії EVO. На його платформі підприємці самостійно створюють інтернет-магазини і / або розміщують свої товари в загальному каталозі. Для покупців на Промі зібрано більше 100 мільйонів товарів.

AliExpress.com

AliExpress.com — інтернет-магазин роздрібної торгівлі малого бізнесу Китаю, що пропонує товари міжнародним покупцям в Інтернеті. Належить групі компаній

Alibaba. Сайт допомагає малим підприємствам продавати свою продукцію клієнтам по всьому світові і, так само як і на сайті Amazon або eBay, на AliExpress можна знайти практично все що завгодно для продажу.

Alibaba Group

Alibaba Group — група компаній, що займаються бізнесом в Інтернеті. Alibaba Group знаходиться у приватній власності й базується в Китаї, в місті Ханчжоу. Основними видами діяльності є торгові операції між компаніями за схемою B2B, роздрібна онлайн-торгівля. У 2012 обсяг торгів — 1,1 трлн юанів ($170 млрд обсяг продажів на двох порталах групи Alibaba). Компанія діє, перш за все, у Китайській Народній Республіці (КНР)

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 продажу 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.

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.

Lamoda

Lamoda – один з найбільших інтернет-магазинів в Росії і СНД . З 2014 року входить до складу Global Fashion Group.1.3. Висновок

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

Огляд аналогів показав, що в першу чергу варто звернути увага 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

Розробляється система для формування заказу повинна задовольняти наступним функціональним вимогам:

- система повинна надавати користувачеві можливість регїістрації і авторизації з кількома пристроями одночасно
- система повинна забезпечувати синхронізацію даних на різних пристроях користувача;
- система повинна дозволяти користувачеві переглядати без підключення до Інтернету заздалегідь завантажену інформацію;
- система повинна надавати користувачеві можливість змінювати свої реєстраційні дані;
- система повинна надавати користувачеві інформацію про магазини;
- система повинна надавати користувачеві можливість поіску товару/магазину за назвою або за тегами;
- система повинна надавати користувачеві можливість фільтрування пошуку;

- система повинна дозволяти користувачеві створювати заявки на закази (із зазначенням дати, часу, товарів, коментарі) і скасовувати їх;
- система повинна надавати менеджеру можливість обрабки призначених для користувача заявок: прийняти або відхилити заказ, або зателефонувати користувачеві на зазначений номер телефону;
- система повинна дозволяти менеджеру змінювати інформацію про магазин, про товари.

1.2.2. Нефункціональні вимоги

В результаті аналізу функціональних вимог і огляду аналогичних проектів, були сформульовані наступні нефункціональні вимоги:

- програма має бути написано на мові Kotlin під платформу Android;
- програма має бути написано на мові Swift під платформу IOS;
- сервер системи повинен мати REST інтерфейс;
- сервер повинен підходити як під IOS так і під Android;
- дизайн додатків повинен бути ідентичним.

1.2.3. Проектування архитектури мобільного клієнт-серверного мобільного застосунку

Діаграма варіантів використання мобільного застосунку передставлена на рисунку процедуру авторизації/регістрації магазину. Авторизований користувач – це людина, яка успішно пройшов процедуру авторизації або реєстрації як покупець.

Рисунок 1 - Діаграма варіантів використання мобільного застосунку

Невідомий користувач може зареєструватися в додатку, вказавши ім'я, номер теле

Користувач може авторизуватися в додатку через номер телефону або електронну адресу.

Авторизований/неавторизований користувач може подивитися інформацію про за

Авторизований/неавторизований користувач може шукати товари в загальному пошуку, тобто отримати список речей, знайдених за назвою або за тегами в меню різних магазинів.

Авторизований користувач може створювати заявки на товари із зазначенням дати, часу, кількості речей, кольору, розміру, довільного коментаря (за бажанням).

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

Авторизований користувач може управляти особистими даними, тобто змінювати номер телефону, ім'я, електронну адресу, пароль, тощо. Схема варіантів використання сервера системи представлена на рис. 2

З сервером взаємодіють два актори: клієнтська програма, додаток для користувача. Клієнтський додаток – додаток, яке звернулося до сервера, але ще не було упізнано їм як додаток для користувача або менеджера.

Рисунок 2 - Схема варіантів використання сервера системи

Клієнтську програму може авторизуватися або зарегіструватися. Якщо клієнтську програму спробує зареєструватися як додаток для менеджера, йому повернеться відповідь з повідомленням про те, що додатки для менеджерів реєструє адміністратор системи.

Додаток для користувача може отримувати інформацію про заклади, тобто загальну інформацію, акції і каталог, тощо.

Додаток для користувача може змінювати особисту інформацію: номер телефону, ім'я та електронну адресу, пароль, тощо.

Додаток для користувача може створювати заявки на бронювання товарів в певному закладі.

Додаток для користувача (для менеджера) може також отримувати інформацію про

Додаток для користувача (для менеджера) може змінювати інформацію про магазин, додавати, змінювати і видаляти акції і фотографії закладу, редугувати пункти каталогу.

Обидві програми можуть змінювати статус заявок на бронювання, додаток для менеджера може встановити один з наступних статусів: «в процесі», «відхилена», «завершена» або «товару не має».

Обидві програми можуть отримувати історію бронювань: додаток користувача отримує історію бронювань користувача, а додаток менеджера отримує історію бронювань закладу.

Диспетчер URL виконує прив'язку URL, за яким до сервера звернулося клієнтську програму, до функції з обробника запитів додатка користувача або менеджера.

Оброблювач запитів додатка користувача і обробник запитів додатка менеджера містять логіку збору, перетворення і компонування даних, наданих (або запитаних) додатком користувача або менеджера відповідно.

Сервіс авторизації – компонент, що надає кошти прийняття рішенн про те, чи має клієнтську програму право на виконання запитаного дії.

Компонент розсилки СМС-повідомлень генерує тексти СМС повідомлень і містить конфігурацію для використання стороннього сервісу СМС-розсилок.

Компонент розсилки електронних листів генерує тексти електронних листів і містить конфігурацію для використання стороннього сервісу розсилки електронних листів.

Компонент відправки звітів про помилки – компонент, який в разі некоректної поведінки системи створює запис у файлі логів, формує текст звіту про помилку та за допомогою компонента розсилки електронних листів відправляє сформований звіт розробнику.

Менеджер взаємодії з базою даних відповідає за взаємодію з базою даних, а також містить конфігураційні дані для з'єднання з базою даних. На рис. 3 представлені деякі структури бази даних сервера системи.

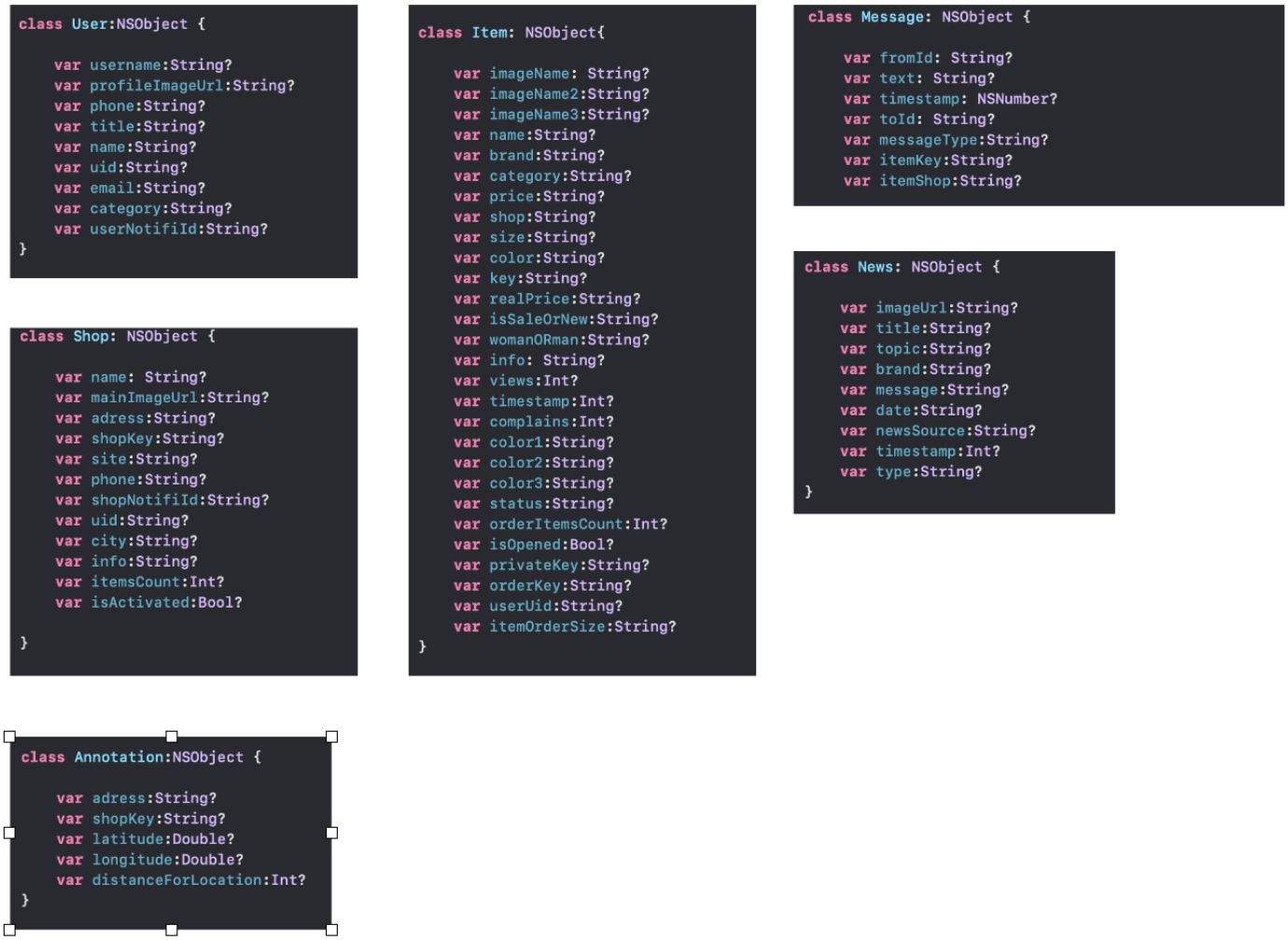


Рисунок 3 - Сутруктура бази даних сервера системи

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 Інтерфейс користувача призначений для взаємодії наслідком з системою.

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.

Компонент інформації про користувача дозволяє переглянути і змінити реєстраційні дані користувача.

Компонент меню надає меню закладу за категоріями, а також відповідає за пошук бажаного товару за назвою або тегу по каталогу всіх закладів.

Компонент авторизації визначає необхідність авторизації і надає можливість реєстрації або авторизації при необхідності.

Компонент поновлення даних оновлює дані в базі даних програми, якщо пристрій авторизовано, сервер системи доступний, і пройшов певний час після попереднього поновлення даних.

HTTP клієнт реалізує взаємодію з сервером системи способом звернення до REST-інтерфейсу сервера.

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

Сервіс прийому повідомлень FCM обробляє приходять спільно від сервісу Firebase Cloud Messaging (FCM) і повідомляє користувача про отримані повідомлення. Firebase Cloud Messaging – це безкоштовний сервіс для відправки повідомлень з серверів в додатки для пристроїв Android, iOS і Chrome . В системі «Gang!» FCM використовується для прискорення обміну інформацією про заявках на бронювання.

1.3. Реалізація системи

Серверна частина системи реалізована мовами програмування Swift/Kotlin з використанням веб-фреймворку Realtime Database. Swift/Kotlin підтримують кілька парадигм програмування, в тому числі структурний, об'єктно-орієнтоване і функціональне. Realtime Database – надає в режимі реального часу базу даних та бекенд як службу. Ця служба надає розробникам додатків API, який дає можливість синхронізувати дані застосунків між користувачами та зберігати їх у хмарі Firebase. REST API використовує протокол подій із сервером, який є інтерфейсом для створення HTTP-з'єднань для отримання push-повідомлень від сервера. Розробники, які використовують Realtime Database, можуть захищати свої дані за допомогою правил безпеки, що застосовуються на сервері. На сервері дані зберігаються в базі даних, створеної за допомогою Firebase Database, тому що вона вільно поширюється і інтегрується .

Клієнтська частина системи реалізована для платформ Android та IOS, являющії найбільш популярними мобільними операційними системами в світі на об'єктно-орієнтованих мовах Kotlin та Swift. Для локального зберігання даних використовується база даних, створена з за допомогою СУБД NoSQL, підтримка якої вбудована в платформу Android та IOS.

Взаємодія клієнтського додатка і сервера реалізовано засобом HTTP-запитів від програми до сервера і FCM-повідомлень від сервера.

Робота з базою даних здійснюється за допомогою Firebase, створення і зміна таблиць відбувається, за якими створюються NoSQL-запити на створення і зміна таблиць бази даних. REST API використовує протокол подій із сервером, який є інтерфейсом для створення HTTP-з'єднань для отримання push-повідомлень від сервера.

В процесі реалізації мобільного додатка було вирішено відкзаться від використання ORM-бібліотек на користь використання класу-помічника СУБД SQLite, SQL- виразів. Рішення обумовлено тим, що додаток підтримує великий спектр версій платформи Android, тоді як робота деяких ORM-бібліотек нестабільна

на платформах з версією API менше, ніж 21. Крім того, збереження і читання даних відбувається швидше без використання ORM, ніж з використанням будь-якого ORM рішення, а в реалізованому додатку однією з найбільш частих операцій з базою даних є збереження великого обсягу даних, отриманих від сервера системи .

Правила безпеки в режимі реального часу Firebase визначають, хто читав та записує доступ до бази даних, як дані структуровані, і які показники існують. Ці правила живуть у серверах Firebase і застосовуються автоматично у будь-який час. Кожен запит на читання та запису буде завершено, якщо б правила дозволяють його. За замовчуванням правила не дозволяють будь-кому доступ до бази даних. Це для захисту бази даних від зловживань, доки не вдасться налаштувати свої правила або налаштувати автентифікацію.

Правила безпеки в режимі реального часу мають синтаксис JavaScript, подібний до JavaScript, приклад програмного коду JavaScript, в якому прописано правило безпеки в режимі реального часу представлено на рис. 4:

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

Рисунок 4 - Правила безпеки в режимі реального часу

Робота в реальному часі

Замість типових HTTP-запитів база даних Firebase Realtime використовує синхронізацію даних – кожен раз, коли дані змінюються, будь підключений

пристрій отримує це оновлення протягом мілісекунд. Що дозволяє забезпечувати спільний і захоплюючий досвід, не думаючи про мережевому коді.

Офлайн робота

Додатки Firebase залишаються чуйними навіть в автономному режимі, оскільки SDK бази даних Firebase Realtime зберігає дані на диск. Після відновлення з'єднання клієнтський пристрій отримує будь-які припущення зміни, синхронізуючи його з поточним станом сервера.

Доступність з пристроїв клієнтів

Доступ до бази даних Firebase Realtime можна отримати безпосередньо з мобільного пристрою або веб-браузера; немає необхідності в сервері додатків. Безпека і перевірка даних доступні через правила безпеки баз даних Firebase Realtime, засновані на виразах правила, які виконуються при читанні або запису даних.

Масштабування по декількох базах даних

За допомогою бази даних Firebase Realtime можна масштабувати потреби додатків в даних, розбиваючи дані на кілька примірників бази даних в одному проекті Firebase. Є можливість оптимізувати аутентифікацію за допомогою Firebase Authentication для даного проекту і перевіряти справжність користувачів у всіх примірниках бази даних. Керувати доступом до даних в кожній базі даних за допомогою призначених для користувача правил бази даних Firebase Realtime для кожного екземпляра бази даних.

Сувора перевірка за замовчуванням включена для операцій запису, що запускають події. Будь-які операції записи, які запускають більше 1000 хмарних функцій або одна подія розміром більше 1 МБ, завершаться невдало і повернуть помилку, що повідомляє про досягнутому межі. Це означає, що деякі хмарні функції взагалі не запускаються, якщо вони не проходять попередню перевірку.

Правила безпеки

Правила бази даних Firebase Realtime визначають, хто має права на читання і запис в базу даних, як структуровані дані і які індекси існують. Ці правила розташовуються на серверах Firebase і застосовуються автоматично в будь-який час. Кожен запит на читання і запис буде виконаний, тільки задовольняє правилам. За замовчуванням правила не дозволяють будь-кому доступ до базі даних. Це необхідно для захисту бази даних від зловживань до тих пір, поки не буде часу налаштувати свої правила або налаштувати аутентифікацію.

Правила бази даних реального часу мають JavaScript-подібний синтаксис і бувають чотирьох типів:

Види правил бази даних реального часу

- .read - описує, чи дозволено читання даних користувачам;
- .write - описує можливість і час запису даних;
- .validate - визначає, як буде виглядати правильно відформатований значення, чи має воно дочірні атрибути, і тип даних;
-.indexOn - визначає дочірній елемент для індексації для підтримки впорядкування і запитів.

База даних Firebase Realtime надає повний набір інструментів для управління безпекою застосування. Ці інструменти полегшують аутентифікацію користувачів, забезпечення прав користувачів і перевірку вхідних даних.

Додатки на базі Firebase виконують більше клієнтського коду, ніж додатки з багатьма іншими технологіями.

Аутентифікація

Поширеним першим кроком в захисті програми є ідентифікація користувачів. Цей процес називається аутентифікацією. Для входу користувачів в програму можно використовувати платформу Firebase Authentication. Аутентифікація в Firebase включає в себе підтримку звичайних методів аутентифікації, таких як Google і Facebook, а також вхід по електронній пошті і паролю, анонімний вхід і багато іншого.

Ідентифікація користувача є важливою концепцією безпеки. У різних користувачів різні дані, а іноді у них різні можливості. Наприклад, в додатку чату кожне повідомлення пов'язано з користувачем, який його створив. Користувачі також можуть видаляти повідомлення у цьому, але не повідомлення інших користувачів.

Авторизація

Ідентифікація користувача – це тільки частина безпеки. Необхідно встановити розмежування доступу до інформації в базі даних. Правила бази даних в реальному часі дозволяють контролювати доступ для кожного користувача. Наприклад на рис. 5 зображено набір правил безпеки, який дозволяє будь-кому читати шлях / foo /, але ніхто не може вносити зміни до нього:

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

Рисунок 5 - Програмний код, який забороняє вносити зміни до шляху /foo/

Правила бази даних реального часу включають в себе вбудовані змінні і функції, які дозволяють посилатися на інші шляхи, тимчасові мітки на стороні сервера, інформацію про аутентифікації і багато іншого. Ось приклад на рис. 6 правила, яке надає доступ на запис для аутентіфіцированний користувачів / users / <uid> /, де <uid> - це ідентифікатор користувача, отриманий за допомогою Firebase Authentication.

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

Рисунок 6 - Приклад надання доступу на запис

Валідація даних

База даних Firebase Realtime є нереляційних. Це дозволяє легко щось міняти в міру розробки, але як тільки застосунок буде готове виходу в маси, важливо, щоб дані залишалися узгодженими. Мова правил включає в себе validate правило, яке

дозволяє застосовувати логіку перевірки з використанням тих же виразів, що .read і для .write правил. Єдина відмінність полягає в тому, що правила перевірки не каскадуються, тому всі відповідні правила перевірки повинні мати значення true для дозволу запису.

Ці правила на рис. 7 забезпечують, що записуються дані / foo / повинні бути рядком довжиною менше 100 символів:

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

Рисунок 7 - Приклад забезпечення правил довжини параметра foo не більше 100 символів

Правила перевірки мають доступ до всіх тих же вбудованим функціям і змінним, що .read і для .write правила. Їх можна використовувати для створення правил перевірки, які будуть знати дані в інших місцях бази даних, особистість користувача, час сервера і багато іншого.

Параметри пароля

Вся взаємодія між клієнтом аутентифікації Firebase і внутрішніми серверами відбувається за HTTPS-з'єднань, які зашифровані наскрізним чином. Таким чином, хоча пароль може відображатися у вигляді простого тексту в коді, він не може бути переглянутий ніким, перевіряючим мережевий трафік (якщо мережа вже не зламана).

Проста утиліта шифрування на основі пароля доступна як демонстрація бібліотеки Script. Він може бути викликаний як Script {key} {salt} {rounds}

{memcost} [-P]. Утиліта попросить простий текстовий пароль і вивести хеш після успіху. Цей хеш повинен бути закодований до BASE64 та порівняно з паролем Hash експортованого облікового запису користувача.

{Key} – ключ підписує з параметрів хеша пароля проекту. Цей ключ повинен бути береться стверджувати з base64 перед тим, як він буде переданий утиліті.

{Сіль} – Об'єднання солі пароля з експортованої облікового запису і роздільник солі з хеш-параметрів пароля проекту. Кожна половина має бути декодована з base64 перед конкатенацией.

{Rounds} – Параметр раундів з хеш-параметрів пароля проекту.

{Memcost} – Параметр mem_cost з хеш-параметрів пароля проекту.

[-P] – Може бути також надано необов'язковий -P, щоб дозволити читання пароля в необробленому тексті з STDIN.

Відправлення повідомлень користувачам

При реєстрації користувач вказує номер телефону та адресу електронної пошти. На вказаний номер телефону надсилається сгенеріроний сервером код авторизації – рядок символів, перебуваючи з цифр. При авторизації в додатку вже зареєстрованого користувача процедура трохи інша: код авторизації не висилається, а користувач заходить в свій аккаунт за допомогою адресу електронної пошти та паролю.

СМС-розсилка і Email-розсилка здійснюються з використанням сторонніх сервісів OneSignal і Firebase.

Коли менеджер змінює статус заявки, користувачеві відправляється FCM-повідомлення, в якому передаються такі дані: ідентифікація по заявці, статус якої змінився, та повідомлення.

У разі помилки при відправці будь-якого типу повідомлення, сервер робить відповідну запис в файлі логів.

Підтримка різних версій платформи IOS та Android

У кожному проекті необхідно позначити мінімальну підтримувану додатком версію платформи Android та IOS. Чим менше версія, тим на більшу кількість

пристроїв буде можлива установка додатка вання, але буде неможлива або обмежена робота з деякими віз можностями API пізніших версій платформи.

Згідно з офіційним сайтом платформи Android , частка Android-пристроїв версії 4.1 і вище становить 99,8% на момент початку 2019 року. Тому в якості мінімальної підтримуваної версії було вирішено вибрати версію 4.1 (Jelly Bean, API 16).

Для коректної роботи програми на пристроях з різними версіями API були розроблені різні версії деяких компонентів інтерфейсу (див пункт 3.3.2).

Реалізація користувальницького інтерфейсу

Дизайн додатка розроблявся з опорою на принципи Material Design. Material Design представляє собою комплексну концепцію з будівлі візуальних, що рухаються і інтерактивних елементів для різних платформ і пристроїв від компанії Google. Скріншоти користувачів інтерфейсу представлені в додатку.

Для навігації по основним екранів додатка використовується паттерен Navigation Drawer – меню, знаходиться знизу екрану і містить основні пункти навігації за додатком (як в IOS).

Деякі можливості Material Design, наприклад на рис. №8, певні види анімації та ряд XML-атрибутів елементів доступні тільки в Android з рівнем API не менш 21. Для забезпечення сумісності додатків з пристроями під керуванням більш ранніх версій платформи Android створюються різні файли ресурсів для версій платформи з рівнем API менше 21 (в каталогах values /, drawable / і т.д.) і з рівнем 21 і вище (в каталогах values-v21 /, drawable-v21 / і т.д.).

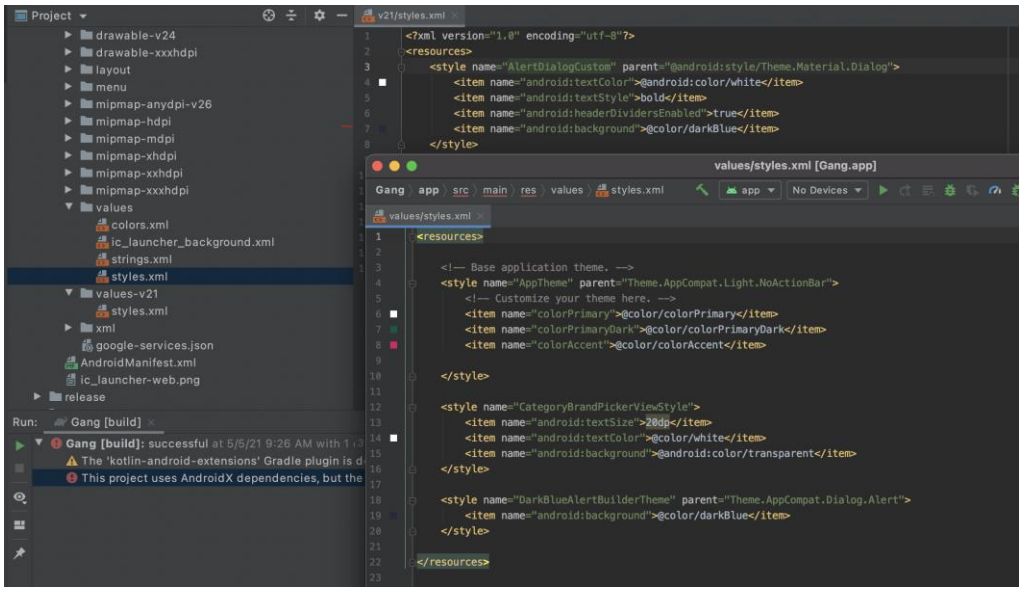


Рисунок 8 - Деякі можливості Material Design

Розглянемо вищеописаний підхід на прикладі визначення конконкретного стилю raisedButton, використовуваного в додатку для стилізації елементів управління. На рис. 9 наведено визначення стилю raisedButton для платформи Android з версією API нижче 21, на рис. 10 – з версією 21 і вище. Ресурси, що використовуються для визначення типу зміни зовнішнього вигляду елемента при взаємодії користувача з ним, наведені на рис. 11 і рис. 12 відповідно для вищеописаних стильових файлів.

Прийом повідомлень сервісу Firebase Cloud Messaging та OneSignal

Коли на сервер системи надходить інформація про те, що менеджер змінив статус заявки клієнта, на всі пристрої клієнта за допомогою сервісу OneSignal чи FCM відправляється повідомлення з інформацією про новий статус заявки, після чого додаток виводить на екран push-повідомлення.

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

- Статус заявки було змінено
- Надійшло повідомлення від магазину
- Повідомлення було відправлено програмою примусово задля сповіщення користувачів про новини чи зміни/оновлення
- Веріфікація номера телефону

Як тільки користувач здійснює будь-яку дію зі своїми заявками (створює нову або скасовує вже існуючу), додаток відправляє дані про створену або зміненої заявці на сервер. Сервер зберігає отримані дані в базу даних і відправляє з додатком менеджера OneSignal-повідомлення з інформацією про заявку. У додатку менеджера відбувається обробка заявки, і, якщо в результаті обробки її статус змінився, додаток відправляє дані про заявку на сервер системи. Сервер зберігає дані і відправляє OneSignal-повідомлення з інформацією про заявку з додатком користувача. Додаток користувача виводить push-повідомлення з інформацією про зміну статусу заявки.

1.4. Постановка задачі

На сьогоднішній день в сфері товарного бізнесу магазини реалізуються за допомогою мобільних застосунків, що має як великий недолік, та і чималий ризик. Недолік полягає в тому, що співробітникам необхідно витрачати частину робочого часу на підтримку актуальності наданої інформації, тобто вчасно вносити зміни через інтерфейс адміністратора магазину або повідомляти про них технічній підтримці проекту. До того ж необхідно стежити за новим джерелом заявок на бронювання товарів. Також недоліком є те, що у магазинів з’являється деяка конкуренція, тому що магазини повинні виставляти всі ціни на товари. У той же час маркетингові дослідження показують, що надання інформації про магазин та товари його офіційними джерелами і тим більше сторонніми сервісами є ефективним впливом на споживача, що підвищує конкурентоспроможність закладу.

Для забезпечення взаємодії бажающого забронювати товар користувача і менеджера відповідного закладу, система Gang! Повинна включати в себе додатки для клієнтів під різні платформи, сервер системи і додаток для менеджерів. В рамках дан ної роботи будуть реалізовані сервер системи і мобільний додаток під платформу Android та IOS.

Невід'ємною частиною процесу розробки інформаційних систем, що взаємодіють з критичними ресурсами, є етап аналізу інформаційної захищеності

таких систем. Доцільність дослідження інформаційних систем в контексті захищеності інформаційного середовища обумовлюється ризиками часткової або повної компрометації системи і конфіденційних даних внаслідок наявності програмних вразливостей в компонентах таких систем.

Безліч компонентів інформаційних систем, що взаємодіють з критичними ресурсами, включає програмні рішення для мобільних платформ. Актуальність таких рішень визначається значним розвитком мобільних екосистем і, в сукупності з затребуваністю аудиту інформаційної безпеки як етапу розробки захищених систем обробки даних, зумовлює нагальність дослідження мобільних додатків в контексті інформаційної безпеки.

Таким чином для побудови захищеного мобільного застосунку потрібно вирішити наступний спектр завдань:

- аналіз вразливостей мобільних застосунків;
- аналіз методів захисту мобільних застосунків;
- розробка безпеки персональних даних користувачів мобільного застосунку Gang!;
- розробка розмежування доступу до інформації в мобільному застосунку Gang!;
- реалізація безпечного мобільного застосунку Gang!

Висновки за розділом 1

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

Відповідно до вимог була спроектована архітектура системи «Gang!». Крім того, були спроектовані бази даних сервера системи і мобільного застосування під платформи Ios/Android.

На основі розробленої архітектури були реалізовані сервер системи «Gang!» і мобільний застосунок під платформи IOS та Android. Взаємодія сервера і

мобільного застосування реалізовано за допомогою HTTP-запитів і FCM-повідомлень та було розроблено розмежування доступа до інформації, а також було проведено захист на рівні програмного коду.

РОЗДІЛ 2 АНАЛІЗ МЕХАНІЗМІВ ЗАХИСТУ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1. Аналіз вразливостей мобільних застосунків

Безпека мобільних додатків – це практика захисту цінних мобільних додатків і цифрової особистості від шахрайських атак в усіх їх формах. Це включає втручання, зворотне проектування, шкідливе ПЗ, реєстратори ключів та інші форми маніпуляції або втручання. Комплексна стратегія безпеки мобільних додатків включає технологічні рішення, такі як захист мобільних додатків, а також передові методи використання і корпоративні процеси.

Безпека мобільних додатків швидко набула важливість, оскільки мобільні пристрої набули поширення в багатьох країнах і регіонах. Тенденція до більш широкого використання мобільних пристроїв для банківських послуг, покупок і інших дій корелює зі зростанням кількості мобільних пристроїв, додатків і користувачів. Банки посилюють свою безпеку, і це хороша новина, яка використовує свої мобільні пристрої для банківських послуг.

Розробники розуміють важливість безпеки мобільних додатків, але не всі розуміють це. Крім зростання швидкості мобільне шахрайство, є кілька інших причин, за якими фінансові установи повинні серйозно ставитися до безпеки мобільних додатків і взяти на себе зобов'язання розробити комплексну стратегію.

Споживачі повинні з обережністю ставитися до інформації, яку вони розкривають, і до даних, які вони завантажують при роботі в Інтернеті, але бізнес-професіонали також повинні проявляти пильність. Мобільні пристрої майже завжди включені, завжди поруч з вами і зберігають величезну кількість особистої інформації, а також конфіденційних даних і документів. Це може зробити їх справжньою знахідкою для зловмисників.

Мобільні додатки можуть бути самовпевненими в запитуваних дозволах. Чому, наприклад, погодного додатком може знадобитися доступ до камери або

мікрофону? І чи може зловмисник знайти в цьому додатку вразливість, яка надає йому доступ до камери або мікрофону для проведення промислового шпигунства?

При аналізі файлу AndroidManifest.xml наші фахівці нерідко виявляють директиву android: allowBackup в значенні true. Це дозволяє створювати резервну копію даних програми при підключенні до комп'ютера. Недоліком може скористатися зловмисник і отримати дані додатки навіть без прав користувача root.

Уразливість небезпечного взаємодії між процесами закладається на етапі проектування інтерфейсів взаємодії компонентів програми та відноситься до помилок в реалізації механізмів захисту. Помилки в механізмах захисту стали причиною 74% вразливостей в додатках для iOS і 57% вразливостей для платформ Android.

Очікується, що в майбутньому частка мобільних пристроїв, що надають функціональні можливості відкритої платформи, буде продовжувати зростати. Відкритість цих платформ дає значні можливості для всіх частин мобільної екосистеми, завдяки можливості гнучкого надання програм і послуг – опцій, які можуть бути встановлені, видалені або оновлені безліч разів згідно з потребами і вимогами користувача. Однак з відкритістю з'являється і необмежений доступ до мобільних ресурсів і API-інтерфейсів додатками невідомого або ненадійного походження, що може призвести до заподіяння шкоди користувачеві, влаштуванню, мережі або всього разом узятого, якщо не використовуються відповідні архітектури безпеки і мережеві запобіжні заходи. Безпека застосування забезпечується в тій чи іншій формі на більшості мобільних пристроїв з відкритою ОС (Symbian OS, Microsoft, BREW, і т.д.). У 2017 році Google розширив свою програму винагород за знайдені вразливості, щоб охопити уразливості, виявлені в додатках, розроблених третіми особами і доступних через Google Play Store. Промислові групи також розробили рекомендації, включаючи Асоціацію GSM і Open Mobile Terminal Platform (OMTP).

Більшість людей, які не думають про безпеку мобільного додатка при практиці свого телефону платити за капучино в Starbucks або спортивні новітню гру під час скорочення, а також під час виконання онлайн-діяльності за допомогою

програми мобільного банкінгу. Тому, як програма будівельника, найбільше питання, яке потрібно вирішити, перш ніж ви вирішите опублікувати додаток, має бути – як я можу захистити свою програму проти будь-якого шкідливого наміру?

Мобільний додаток стає хорошим способом вивчення, щоб зробити його: є сам програмний код, проблема логіка на задній частині та клієнтській стороні, базах даних, даних API, передаючи дані між двома, пристроєм та його операційною системою, і користувач. Кожен відтворює життєву роль у безпеці додатків. Для компаній з мобільними програмами в болоті, конкурентний ринок, збирання надійної безпеки може бути цінним диференціатором.

Помилки безпеки додатків для Android і iPhone, як правило, є менш пріоритетною галуззю для мобільного розробника, в основному тому, що через брак часу. Зазвичай він не отримує того, що заслуговує в планах проектів. Більш того, в разі відсутності власника безпеки в проектних командах ніхто не бере на себе відповідальність. Ось чому безпеку мобільних додатків – це питання уваги, залишеного тільки ініціативи розробника.

Безпека і зручність використання – це дві концепції, які назад пов'язані. Високобезпечні рішення вимагають додаткових процесів і потоків. Тим не менше, більшість бізнес-підрозділів, що працюють безпосередньо зі споживачами, не вважають безпеку додатків головним пріоритетом.

На практиці ніхто не висловить проблему безпеки, якщо тільки щось дійсно не піде не так в разі «злому». Більшість розробників додатків не беруть на себе конкретні тести безпеки Android і безпеки iPhone. (Тест безпеки додатків).

Завжди майте на увазі принцип, що жодна програма не є 100% безпечним!

Наша мета полягає в тому, щоб зробити застосунок більш безпечним, ніж інші, використовуючи швидкі та прості функції, тим самим відразу хакерів до плутанини з мобільним застосунком. Підготуйте свій додаток до мобільної безпеки.

Мобільні додатки, пов'язані з бізнес-брендами, часто стають метою шахраїв, які або використовують своїх клієнтів, дітей своїх клієнтів, або атакують сам бізнес. Якщо атакувати мобільне шкідливе ПО націлене на пристрій користувача, наслідки можуть включати:

- Підміна номера рахунку
- Вкрадені облікові дані для входу
- Вкрадені і перепродані дані кредитної картки
- Несанкціонований доступ до бізнес-мереж
- Крадіжка особистих даних
- Пристрій iOS або Android може поширювати шкідливе ПЗ на інші пристрої.
- SMS-повідомлення можна копіювати і сканувати на предмет особистої інформації

Мобільні додатки генерують величезну кількість даних про нас і нашого життя. Тому забезпечення безпеки додатків для створення і використання цієї інформації має першорядне значення. В іншому випадку небезпечні програми - простий шлях для зловмисного акту крадіжки і продажу особистої інформації. Крім того, існують інші мобільні рішення, які можуть принести значні вигоди.

2.2. Аналіз методів захисту мобільних застосунків

Підхід до зберігання даних: перш за все, конфіденційні дані не повинні зберігатися на пристрої під час виконання якомога більше. Дані можуть бути оброблені в разі потреби і повинні бути негайно вилучені, коли це не потрібно. У разі необхідності зберігання даних на мобільному пристрої дані повинні бути зашифровані і збережені в папці документів. Паролі повинні зберігатися в KeyChain для iOS і KeyStore для безпеки Android. Це також важливо для перевірок безпеки магазину додатків.

Відсутня перевірка зовнішнього інтерфейсу: відсутність перевірки введення даних викликає проблеми як з безпекою, так і з форматуванням. Це можуть бути такі речі, як дозвіл буквено-цифрових значень в числові поля, відсутність маскування на відформатованих полях, які не перевірка значень символів високого ризику, таких як <> `" () | #. Такі відсутні перевірки можуть привести до порушень безпеки, дозволяючи віддалено виконувати код або несподівані відповіді.

Серверні елементи управління: розробка застосунків - це операція на стороні клієнта. Серверна сторона - це місце, де ви повинні зберігати дані і управляти ними. Перевірка боку сервера повинна застосовуватися незалежно від каналу (мобільний, веб- і т.д.) для безпеки і форматування даних. Ми не маємо на увазі в'язку ключів iCloud або аналогічну функцію, будь ласка, будьте обережні. Йдеться про безпеку серверної частини програми. Існують також проблеми безпеки Apple iCloud, проте це робота Apple, щоб зробити це більш безпечно!

SSL: HTTPS повинен використовуватися для безпечної передачі конфіденційної інформації. Якщо можливо, замість сертифікатів з вбудованого сховища сертифікатів пристроїв слід використовувати власний сертифікат. Сертифікат, унікальний для програми та сервера, повинен бути вбудований в додаток.

Обфускація: дуже важливо, особливо для додатків Android, щоб вони проходили обфускацію. Якщо скріптові файли також використовуються в деяких частинах додатка, ці файли також повинні бути взяті через заплутування.

Рядки коментарів: пояснювальні дані, включені в якості рядків коментаря, можуть бути переглянуті іншими, якщо додаток декомпіїровано. Це дуже просто, але поширена помилка як для додатків Android, так і для iOS. Це не означає, що ви не повинні використовувати коментарі під час розробки програми, просто не забудьте видалити їх з виробничої версії програми.

Надмірні дозволу: під час редагування налаштувань дозволів для додатків Android повинні бути включені тільки абсолютно необхідні дозволи. Дозволів на доступ до особистої інформації, такої як «доступ до контактів», слід уникати в максимально можливій мірі для безпеки Android. Якщо щось піде не так, ймовірність витоку даних менше.

Шифрування: ключ, використовуваний в шифруванні, також повинен бути зашифрований і збережений в безпечному сховищі. Установчий файл також повинен бути заплутаний. Ще однією небезпечною практикою, якої слід уникати, є завантаження ключа шифрування з сервера під час виконання.

Пристрої з root / Jailbroken: неможливо зберігати повністю захищені дані в кореневих пристроях, так як дозволу root забезпечують необмежений доступ до файлової системи пристрою і пам'яті. Однак розробники можуть перевірити, чи є пристрій рутом чи ні. Цей ризик слід зазначити і оцінити на основі масштабів проекту для всіх потоків і процесів.

Захист від несанкціонованого доступу додатків: виконавчі файли програми (.apk або .ipa) можуть бути змінені і опубліковані хакерами на різних ринках. Вони можуть вводити шкідливий код в додаток або зламати мобільні застосунки, щоб обійти ліцензійні та платіжні обмеження. В даний час навіть можна зламати покупки в додатку і перевірки ліцензій, просто емулюючи відповіді магазину додатків. Тому цілісність додатків і покупки в додатку повинні бути перевірені на сторонньому довіреному сервері, щоб запобігти таким випадкам. Для цього на ринку є кілька хороших рішень. Знову ж таки, це дуже важливо для безпеки Google Play і iTunes App Store.

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

Для індустрії мобільного безпеки необхідно використовувати крос-платформні вбудовані фреймворки. Проаналізувавши популярні крос-платформені гібридні фреймворки (DOM Payload, Scripting Engine Scope і т.д.) було зроблено висновок про перевагу Smartface Native Framework.

Існує кілька стратегій підвищення безпеки мобільних додатків:

- Складання «білого списку» додатків
- Забезпечення безпеки транспортного рівня
- Сувора аутентифікація і авторизація
- Шифрування даних при записі в пам'ять
- Пісочниця додатків
- Надання доступу додатків на рівні API

- Прив'язка процесів до ідентифікатора користувача
- Доля взаємодії між мобільним додатком та ОС
- Вимога підтвердження користувача для надання привілейованого / підвищеного доступу з додатком
- Правильна обробка сеансу

Розглянемо уразливості загального характеру, без прив'язки до конкретної платформі. Тут і на далі використовується абревіатура КВД - критично важливі дані користувачів. До КВД відносяться будь-які дані, які не повинні бути доступні третій стороні, це стосується як персональних даних користувача (дата народження, адреса проживання, особисте листування), так і його приватних даних (паролі, дані кредитних карт, номери банківських рахунків, номери замовлень і так далі).

Таким чином, можно сформувати наступний перелік основних вразливостей мобільних застосунків.

Використання незахищених локальних сховищ.

Небезпека: дуже висока.

Зустрічається повсюдно, виражається в зберіганні КВД в незахищених або слабо захищених локальних сховищах, специфічних для конкретної платформИ. Розтин третьою стороною - елементарне, і, як правило, не потрібна наявність спеціальних навичок у атакуючого.

Захист: зберігати КВД можна тільки в захищених сховищах платформи.

Зберігання КВД в коді.

Небезпека: висока.

Уразливість стосується зберігання КВД всередині коду (в статичних константних рядках, в ресурсах додатка і т.п.). Яскраві приклади: зберігання солі для пароля (password salt) в константі або макросі, яка застосовується по всьому коду для шифрування паролів; зберігання приватного ключа для асиметричних алгоритмів; зберігання паролів і логінів для серверних вузлів або баз даних. Легко розкривається третьою стороною при наявності базових навиків декомпіляцію.

Захист: не зберігати ніякі КВД в коді або ресурсах додатка.

Застосування алгоритмів зі зберіганням приватного ключа.

Небезпека: висока.

Уразливість актуальна в разі, якщо приватна інформація алгоритму (приватний ключ) вимушено зберігається в коді або ресурсах мобільного застосування (найчастіше так і буває). Легко розкривається методом декомпіляцію.

Захист: у мобільній розробці бажано застосовувати тільки сучасні симетричні алгоритми з генеруються випадковим одноразовим ключем, що володіють високою стійкістю з злому методом грубої сили, або виводити асиметричний приватний ключ за межі програми, або персоналізувати цей ключ (як приклад - приватним ключем може виступати призначений для користувача код входу Вони зберігаються в зашифрованому вигляді в захищеному сховище операційної системи).

Використання асиметричного алгоритму з приватним ключем, відомим сервера.

Небезпека: залежить від ступеня захищеності сервера.

Уразливість носить подвійний характер. Зберігання приватного ключа допускає можливість розшифровки даних користувача на стороні сервера. По-перше, це некоректно з точки зору безпеки (якщо сервер буде зламаний - атакуючий також отримає доступ до приватних даних користувачів), а по-друге, це порушує приватність персональних даних. Користувач завжди повинен бути впевнений, що його персональна інформація не відома нікому, крім нього самого (тільки якщо він явно не дав дозвіл на її публікацію). Часто додатки позиціонують себе як захищені, але на ділі такими не є, оскільки містять в собі засоби для розшифровки персональної 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 (сервера абсолютно нема чого знати пароль користувача; якщо ж користувач його забув - для такої ситуації існує давно налагоджений механізм відновлення пароля, в тому числі з двухфакторной авторизацією клієнта для підвищеної безпеки процедури відновлення).

Використання самописних алгоритмів шифрування і захисту.

Небезпека: середня.

Це пряме порушення принципу Керкгоффса. Виражається в спробі розробника винайти "свій особистий, невідомий нікому, а тому супер-захищений алгоритм шифрування". Будь-яке відхилення від існуючих, багаторазово перевірених і вивчених, математично доведених алгоритмів шифрування в 99% випадків обертається швидким зломом подібної "захисту". Вимагає наявності середньо-високих навичок у атакуючого.

Захист: слід підбирати відповідний алгоритм тільки з налагоджених і актуальних загальновідомих криптографічних алгоритмів.

Передача КВД в зовнішнє середовище у відкритому вигляді.

Небезпека: середня.

Виражається в передачі КВД без застосування шифрування по будь-якого доступного каналу зв'язку із зовнішнім середовищем, будь то передача даних сторонньому додатку або передача в мережу. Може бути розкрите опосередковано шляхом розтину не програми, а його сховища, або цільового програми. Злом вимогливий до наявності навичок у атакуючого, за умови, що сховище є захищеним.

Захист: будь-які КВД перед виходом за межі додатки повинні бути зашифровані. Локальні сховища платформи не є цариною докладання, вони теж повинні отримувати на вхід тільки зашифровані дані.

Ігнорування факту наявності рутованних 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). Щоб встановити програму на рутованних девайс нівелює всі штатні засоби захисту операційної системи.

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 (спасибі за доповнення DjPhoeniX).

Зберігання КВД в захищених сховищах, але у відкритому вигляді.

Небезпека: середня.

Розробники часто схильні зберігати КВД в захищені системні сховища без додаткового захисту, оскільки системні механізми добре пручаються злому. Однак рівень їх стійкості падає до мінімуму в разі, якщо пристрій рутованних.

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.

Найчастіше виглядає як передача КВД у вбудований браузер, де завантажується зовнішня веб-сторінка, яка виконує свою частину функціоналу. Рівень захисту в цьому випадку різко знижується, особливо для рутованних пристроїв.

Захист: чи не використовувати вбудований браузер і вбудований веб-движок в операціях з КВД. На крайній випадок - шифрувати КВД перед передачею.

Реверсивна інженерія алгоритмів, що представляють інтелектуальну цінність.

Небезпека: низька, залежить від цінності алгоритму.

Якщо при розробці програми всередині компанії використовуються якісь власні алгоритми, які можуть представляти високу цінність для потенційних конкурентів або зломщиків, то ці алгоритми повинні бути захищені від стороннього доступу.

Захист: автоматична або ручна обфускація коду.

Неправильне використання платформи

Даний вид вразливостей стоїть на чолі списку. Незалежно від того Android це або iOS, при розробці для будь-якої з цих платформ необхідно дотримуватися конкретних вимог з метою забезпечення належного рівня безпеки підсумкового

продукту. Але буває, що при створенні додатків деякі з запропонованих правил і рекомендацій ненавмисно порушуються або просто реалізуються з помилками.

В результаті виникає загроза, викликана неправильним використанням будь-якої можливості платформи або відсутністю реалізації протоколів безпеки. Тут можна згадати такі випадки:

Помилкове використання функції iOS Touch ID може привести до неавторизованого доступу до пристрою.

Неправильне застосування iOS Keychain, в слідстві чого чутливі дані, наприклад ключі сесії або паролі, зберігаються в локальному сховищі додатки, а не в захищеному.

Запит надмірних або невірних дозволів платформи.

Наміри (intents) в Android (використовувані для запиту дії з іншого компонента додатка), помічені як відкриті, можуть розкривати чутливу інформацію або дозволяти неавторизоване виконання.

Як запобігти: подібні уразливості потрібно усувати на серверній стороні. Поряд з дотриманням рекомендацій платформи по розробці, ризики можна також мінімізувати за допомогою безпечних підходів до написання коду і застосування правильних налаштувань сервера. Крім цього, зменшити ймовірність невірного використання платформи допоможе:

Заборона додатків на взаємодію один з одним, обмеження доступу, реалізація обмежувальних прав доступу до файлів і ін.

Застосування найсуворішого класу захисту для ланцюжків ключів iOS і проходження найкращим методам розробки з метою уникнення слабкою реалізації елементів управління.

Небезпечне сховище даних

Мобільні пристрої нерідко губляться або крадуться, опиняючись в руках зловмисників. Крім цього, витік особистої інформації користувача може статися через шкідливого ПО, яке дозволяє атакуючому задіяти уразливості пристрою. За допомогою злому або рутінга пристрою можна легко обійти захист шифруванням,

тому розробники ПЗ повинні виходити з припущення, що зловмисники можуть отримати доступ до файлової системи.

І оскільки практично всі програми так чи інакше зберігають інформацію, дуже важливо реалізувати її збереження в такій області, де вона не буде доступна іншому додатку або користувачеві.

Як запобігти: виробляти тестування програми на вразливість загрозам для з'ясування, які інформаційні активи обробляються, і як з цими даними взаємодіють API. Це допоможе:

Визначити, чи ефективно застосовується шифрування, і як захищені ключі шифрування.

Утруднити злом коду за допомогою обфускації, захисту проти переповнення буфера і т.д.

Уникнути збереження / кешування даних там, де це можливо.

Впровадити звукові методи аутентифікації і авторизації.

Небезпечна комунікація

Якщо дані передаються незашифрованими у вигляді чистого тексту, будь-хто, хто відстежує цю мережу, може перехопити і прочитати їх.

Мобільні застосунки, як правило, обмінюються даними по моделі клієнт-сервер. При цьому процес передачі через мережу оператора або інтернет повинен бути реалізований безпечно. 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:

Установку SSL / TLS сертифікатів від перевірених центрів сертифікації (CA).

Попередження користувачів при виявленні недійсного SSL / TLS сертифіката або в разі провалу перевірки ланцюжка сертифікатів.

Небезпечна аутентифікація

Перш ніж надати доступ, додаток повинен перевірити справжність користувача. Обхід аутентифікації зазвичай реалізується через існуючі уразливості, такі як неправильна перевірка сервісних запитів сервером. Мобільні додатки повинні перевіряти і утримувати справжність користувача, особливо в процесі передачі конфіденційних даних, наприклад, фінансової інформації.

Як запобігти: Використання слабких місць механізму аутентифікації дозволяє зловмисникам обходити системи перевірки паролів або отримувати додаткові дозволи, здійснюючи крадіжку даних і інші дії.

Для запобігання подібних ризиків рекомендується:

- Виключити локальну аутентифікацію. Замість цього можна передати її виконання на сторону сервера і завантажувати дані додатки тільки після успішної перевірки автентичності користувача.

- Утриматися від використання вразливих методів аутентифікації (наприклад, посвідчення пристрої), не зберігати паролі локально, реалізувати мультифакторна аутентифікацію (MFA), заборонити використання 4-ціферний PIN-коду в якості пароля і т.д.

Недостатня криптографічний стійкість

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

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 призначених для користувача або 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 різні права, в результаті чого одні отримують стандартний доступ, в той час як інші, наприклад адміністратори, можуть мати додаткові дозволи і привілеї. Слабкі схеми авторизації, незважаючи на успішну перевірку автентичності користувача, можуть не справлятися з перевіркою його прав на доступ до запитуваною ресурсів. Подібний недолік дозволяє хакерам авторізовиваться і виконувати атаки з метою підвищення привілеїв.

Як запобігти: як і у випадку з аутентифікацією, недоліки авторизації можуть вести до крадіжки даних, підриву репутації і навіть штрафів за недотримання вимог. Як протидія цим ризикам варто розглянути наступні варіанти:

Реалізувати перевірку сервером кожного запиту на предмет відповідності вхідних ідентифікаторів тієї особистості користувача, з якої вони асоціюються.

Перевіряти ролі і дозволу аутентифицированного користувача, використовуючи інформацію з бекенд-систем, а не з мобільного пристрою.

Якість коду клієнта

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

Атакуючий може передавати особливі вхідні дані в виклики функцій, провокуючи їх виконання і спостерігаючи за поведінкою програми. У зв'язку з цим падає продуктивність, підвищується споживання пам'яті тощо. 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, які можуть привести до наступних проблем:

- вразливостям форматуючих рядків;
- переповненню буфера;
- впровадженню небезпечних сторонніх бібліотек;

- виконанню віддаленого коду.

При розробці застосунків часто використовуються сторонні бібліотеки, які самі по собі можуть містити баги і бути недостатньо протестовані. Ці нюанси знаходяться поза контролем розробника, оскільки вихідний код йому недоступний. В інших же випадках найчастіше помилки коду виправляються переписуванням його відповідних частин.

Для усунення вразливостей і інших, викликаних неякісним кодом, проблем, рекомендується:

- Використовувати автоматизовані інструменти для тестування буфера на переповнення, визначення витоків пам'яті і ін.
- Спиратися на рев'ю вихідного коду і спочатку створювати його в зрозумілому, добре документованої вигляді.
- Застосовувати в організації узгоджені шаблони написання коду.

Підробка коду

Іноді в магазинах зустрічаються підроблені версії додатків. Від оригіналу їх відрізняє вбудоване в виконуваний файл шкідливий вміст, наприклад закладка, що дозволяє отримувати несанкціонований доступ до системи. Зловмисники можуть повторно підписувати ці підроблені програми та розміщувати їх в сторонніх магазинах або навіть безпосередньо доставляти жертві через фішингові атаки.

Як запобігти: підробка коду програми може вести до втрачених вигод, крадіжці особистих даних, підриву репутації та іншим згубних наслідків. Для запобігання подібних проблем існують такі рекомендації:

Додаток має вміти ідентифікувати будь-яке порушення цілісності коду (тобто зміна або впровадження сторонніх елементів) і відповідним чином реагувати на це в середовищі виконання. Повідомити ж користувачам про легітимних зміни коду можна, наприклад, за допомогою сертифікатів його підпису.

Впровадити техніки, що перешкоджають виконанню підроблених додатків, наприклад, контрольні суми, цифрові підписи, посилення коду та інші методи перевірки.

Реверс-інжиніринг

Зловмисники можуть розібрати і декомпілювати додаток для аналізу коду. Цей спосіб злому особливо небезпечний, так як дозволяє інспектувати, зрозуміти і змінити код, включивши в нього шкідливу функціональність або транслювання небажаної реклами. Розібравшись в принципі роботи програми, хакери можуть змінити його за допомогою таких інструментів, як IDA Pro, Hopper та інших. Після реалізації потрібного їм поведінки, вони можуть повторно скомпілювати додаток і використовувати в своїх намірах.

Як запобігти: перешкодити зловмиснику виконати реверс-інжиніринг програми може тільки неможливість провести деобфускацію коду за допомогою IDA Pro, Hopper і аналогічних інструментів.

Зайва функціональність

Іноді розробники можуть ненавмисно залишати закладки або додаткову функціональність, які не очевидні для кінцевого користувача. В результаті продукт випускаються в продакшен з функцією, яка не повинна бути доступною, що створює додаткові ризики для безпеки програми.

Хакери експлуатують подібні слабкі місця програм безпосередньо зі своїх систем, не потребуючи сприяння з боку регулярних користувачів. Вони вивчають файли конфігурації, виконавчі файли та інші компоненти, розкриваючи функціонал бекенд-частини, який потім використовують для здійснення атак.

*Як запобігти***:** один з найбільш ефективних способів уникнення подібних типів вразливостей - це самостійний аналіз безпеки коду. Таким чином ви зможете:

Перевірити налаштування програми на предмет наявності прихованих перемикачів.

Переконатися, що логи не містять надмірно описових інструкцій про роботу сервера.

Підтвердження особистості

Підтвердження особистості допомагає запобігти крадіжці зловмисником особистих даних користувачів і реєстрацію облікових записів під їх ім'ям. Надійний процес перевірки особистості підтверджує, що користувач є тим, ким він є, і допомагає запобігти шахрайству зловмисником.

Надійна аутентифікація

Захоплення облікового запису - поширена проблема, і паролі швидко застарівають. Через велику кількість витоків даних за останні десять років багато комбінацій імені користувача і пароля вже доступні для продажу в Dark Web. Надійні методи аутентифікації гарантують, що тільки законні користувачі отримують доступ до своїх облікових записів, а зловмисники не можуть увійти в систему в мерзенних цілях.

Біометрія

Біометрія - це безпечний і зручний спосіб входу в мобільні додатки, використовуючи дані, отримані від власного тіла. Немає надійного способу визначити, хто вводить пароль. Розробник програми може тільки визначити, чи відповідає введений пароль ключу пароля в серверній частині системи. Біометрія включає додатковий індикатор довіри, оскільки він підтверджує особистість людини, що пропонує біометричний зразок для перевірки. Тому що відбиток пальця, розпізнавання особи або сканування райдужної оболонки ока відображаються в реальному часі і підключаються до користувача в реальному часі.

Бути безпечним за допомогою заднього кінця

Значна кількість заднього API передбачає, що навряд чи додаток, який було повідомлено, щоб отримати доступ, може вийти з нею. Справа, хоча, це значно далеко від нього. Назад сервери повинні мати системи безпеки, щоб захистити від шкідливих атак. Тому ви повинні захистити, що всі API є автентифікованими на основі мобільної платформи, яку ви очікуєте, до коду, оскільки транспортні процеси та автентифікація API може варіюватися від платформи до платформи.

Аутентифікація високого рівня повинна

Як зазначено вище, багато порушень безпеки стануть до слабкої автентифікації. Отже, це перетворюється на більш вирішальне значення для використання сильнішої автентифікації. Аутентифікація часто відноситься до паролів. Це обов'язок, як заявка, щоб підтримувати своїх користувачів, які будуть стурбовані паролями. Для ілюстрації ви можете створити свою програму, щоб він

приймає лише сильні буквено-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 користувач буде закликано, щоб ввести код, доставлений до своїх текстів або електронної пошти після входу. Якщо ми 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

Для забезпечення надійного шифрування необхідно правильно обрати керування ключами. Зберігайте клавіші в безпечних контейнерах, ніколи не залишати їх на сервері.

Тестування програми

Для розробки якісного коду необхідним є етап тустування. Штально, багато розробників не перевіряють свій код. Це необхідна частина розробки коду якості.

Для створення безпечного застосунку команда повина регулярно оцінюванти код та аналізувати лазівки безпеки, які можуть виникнути в порушеннях даних.

На сьогоднішній день основними стандартами безпеки мобільних застосунків:

- 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

Висновки за розділом 2

Наведений список вразливостей мобільних застосунків може послужити відправною точкою для організацій, розробників, фахівців з безпеки і користувачів, які тільки починають вирішувати відповідні проблеми. Були розглянуті основні вразливості мобільних застосунків, приклади сценаріїв атаки, способи їх запобігання і рекомендовані стратегії по мінімізації шкоди.

Застосунки повинні охоплювати ризики, що виникають в результаті загроз що порушують дані під час роботи мобільних застосунків. Користуючись розглянутими методами безпеки мобільного застосунку необхідно забезпечити захист мобільного застосунку й дані всередині. Ці методи не є складними для виконання. Розробни 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 було реалізовано деякі елементи захисту мобільного застосунку Gang! На рис. 9,10,11 зображені реалізація баз даних для різних рівнів прав доступу.

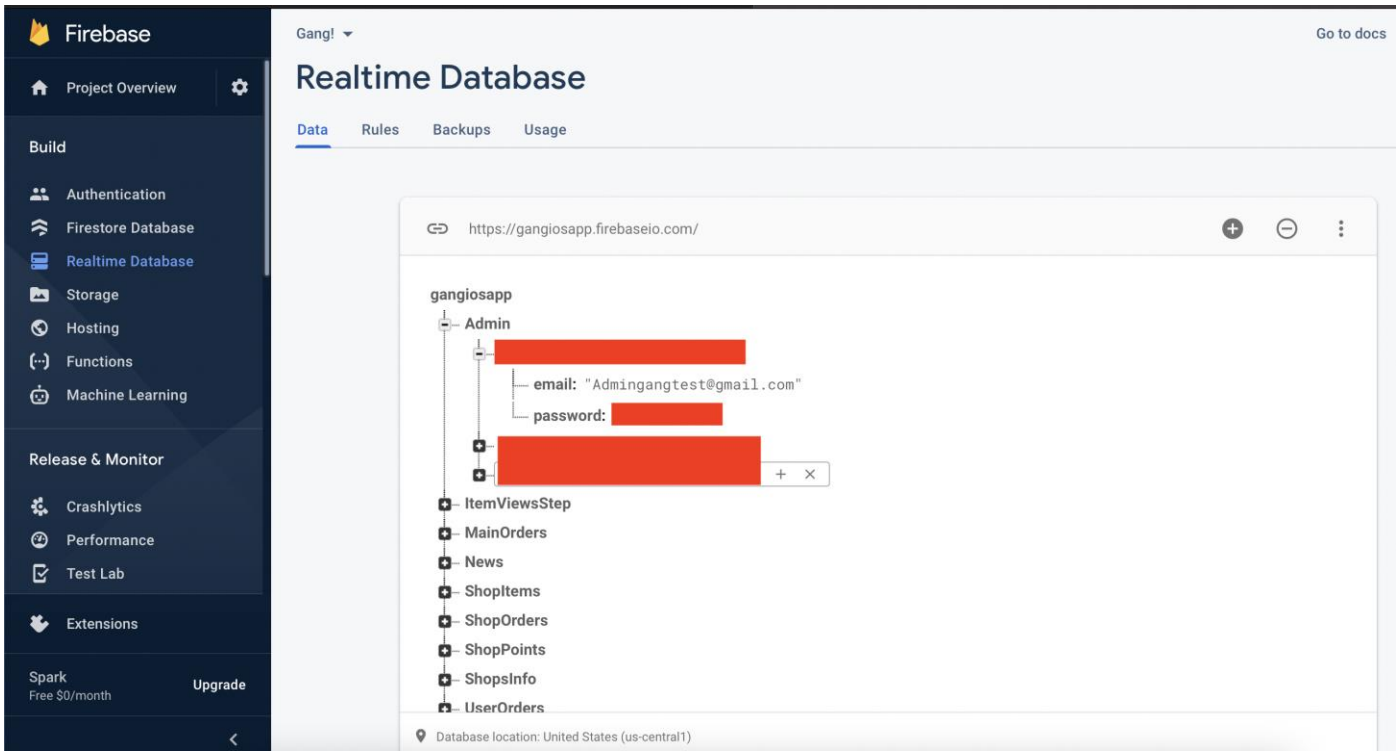


Рисунок 9 - Реалізація доступу до бази даних для адміністратора у мобільному застосунку Gang!

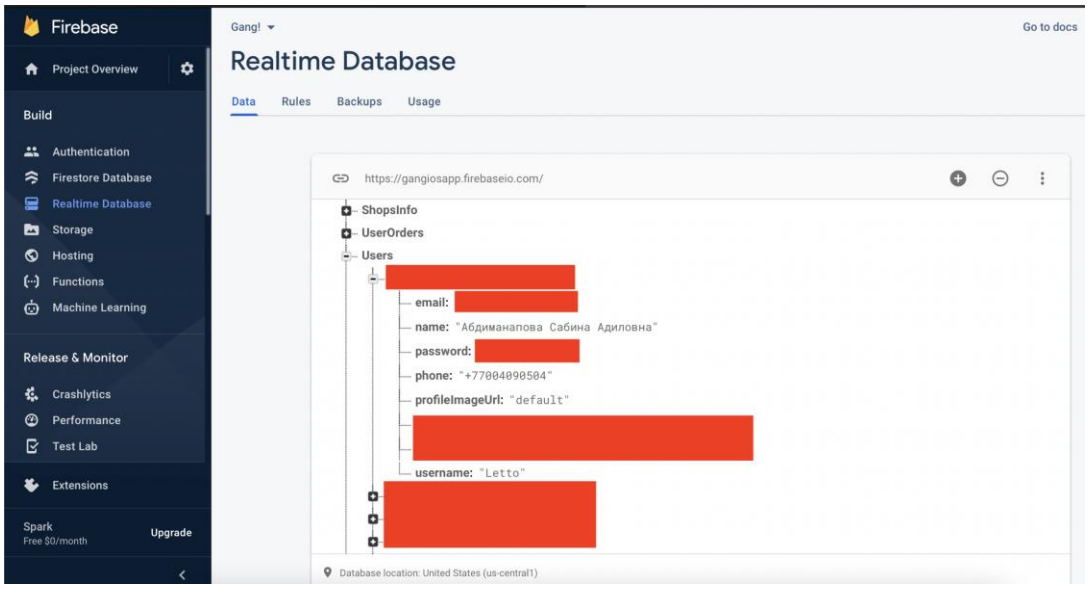


Рисунок 10 - Реалізація доступу до бази даних для користувача у мобільному застосунку Gang!

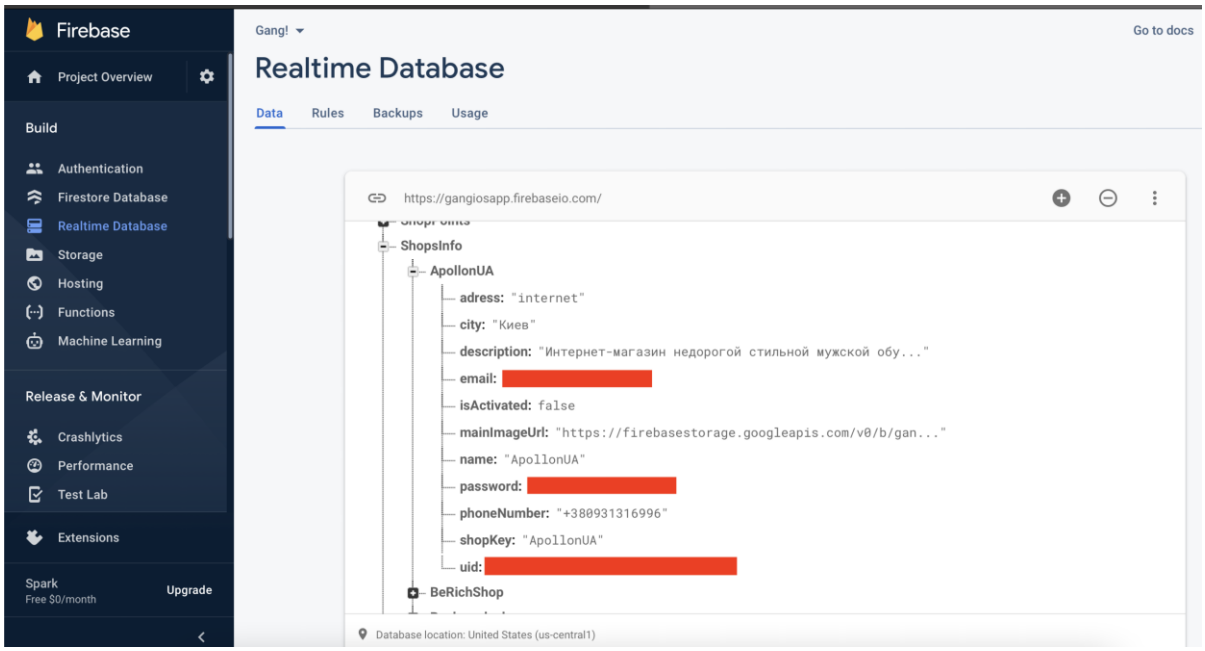


Рисунок 11 - Реалізація доступу до бази даних для менеджера магазину у мобільному застосунку Gang!

Захист пароля та персональних даних користувачів також реалізовані у мобільному застосунку, що є однією з важливим елементом захисту мобільного застосунку Gang! На рис. 12 зображено параметри та спосіб шифрування паролів.

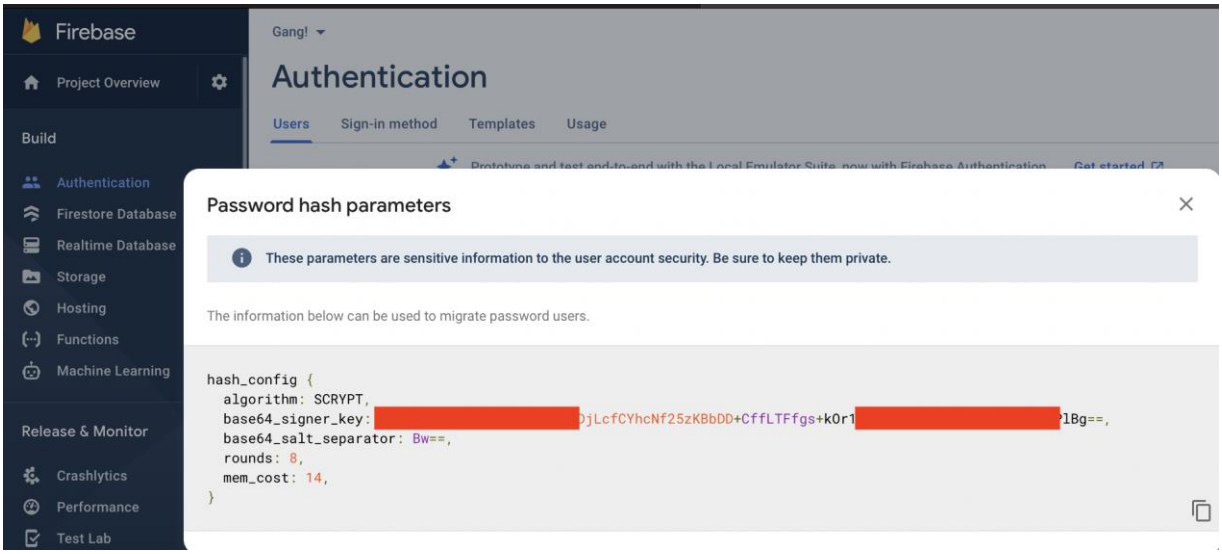


Рисунок 12 - Параметри пароля

В мобільному застосунку Gang! також було реалізовано розмежування прав доступа: є чотири види користувачів (гість, адміністратор, покупець, менеджер).

На рис. 13 зображено приклад розмежування доступу до головної сторінки мобільного застосунку.

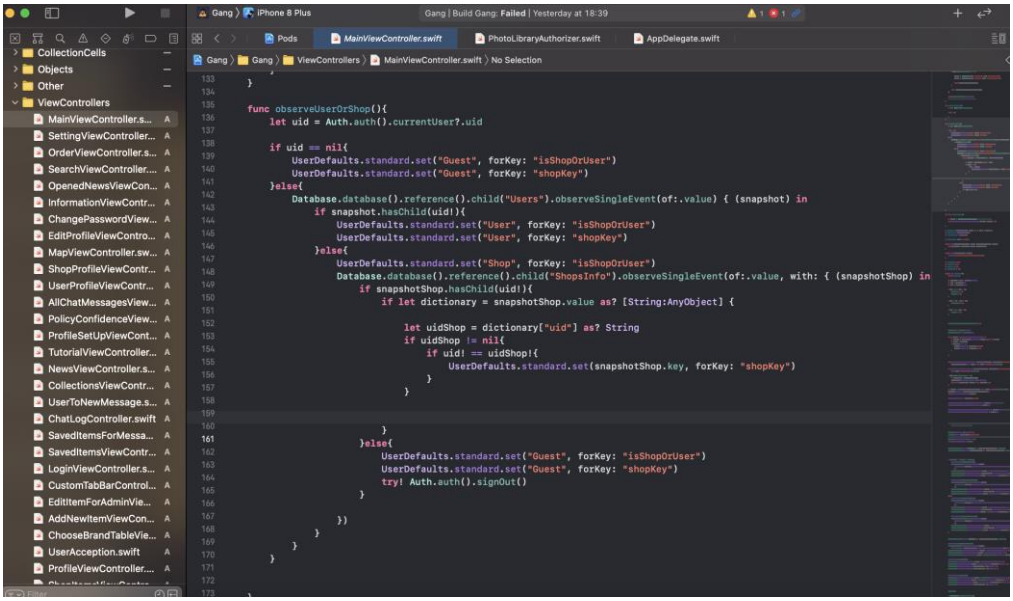


Рисунок 13 - Приклад розмежування доступу у мобільному застосунку Gang!

Процес авторизації в мобільному застосунку Gang! зображено на рис. 14.

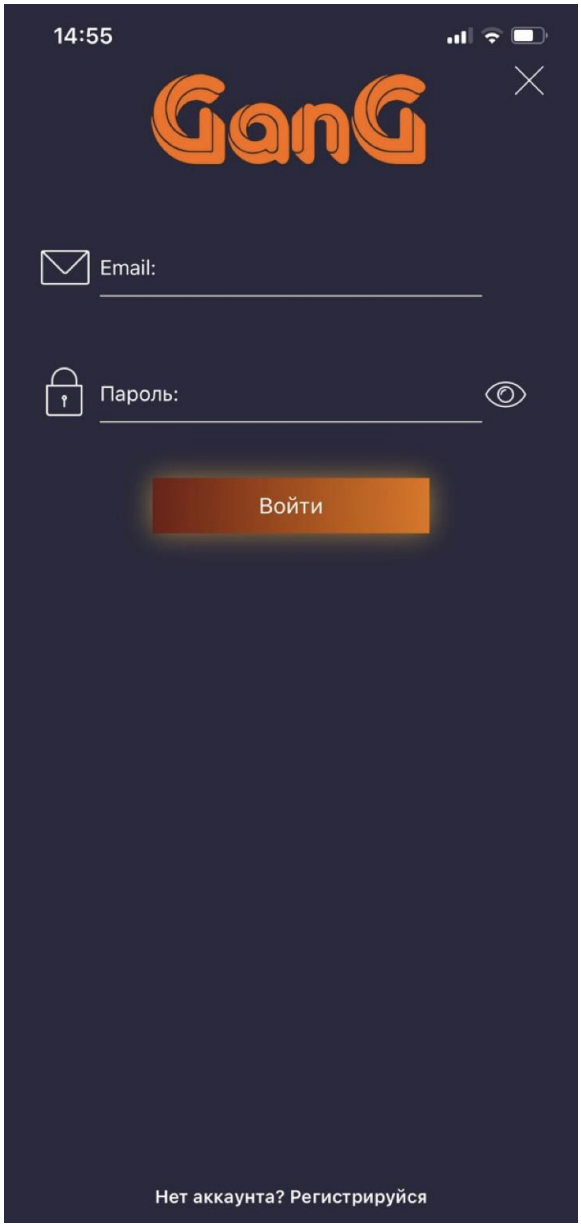


Рисунок 14 - Авторизація у мобільному застосунку Gang!

Додатковий елемент захисту у мобільному застосунку при реєстрації аккаунту це перевірка реального мобільного телефона за кодом. На рис. 15 зображено захист реєстрації користувача за допомогою відправки коду на мобільний номер телефону.

Рисунок 15 - Реєстрація користувача за номером телефона

Більша частина коду мобільного застосунку реалізована таким чином, що будь-які дії з базою даних (читання чи запис) прив’язані до унікального ідентифікатора користувача (UID). На рис. 16 зображен приклад реалізації.

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

Рисунок 16 - Програмний код прив’язки дій до унікального ідентифікатора користувача

3.2 Тестування системи

Для тестування системи застосовувалося функціональне тестування, тобто тестування програмного забезпечення з метою перевірки виконання функціональних вимог.

Тестування Android та IOS - додатків проводилося вручну

Модульні тести в Android можна розділити на два типи:

- локальні модульні тести - тести, для виконання яких використовується тільки віртуальна машина Java (JVM). Вони призначені для тестування бізнес-логіки, яка не взаємодіє з операцион- функціональною системою Android;

- інструментальні модульні тести - тести, за допомогою яких тестується логіка, яка використовує Android API. Їх виконання проходить на фізичному пристрої або емуляторі.

Велика частина логіки застосунку пов'язана зі взаємодією з API системи Android, тому здебільшого були використані інструментальні модульні тести.

Для модульного тестування використовувалася бібліотека JUnit. Бібліотека призначена для тестування програм на мові Java і має велике число можливостей: поділ даних і логіки тесту, угруповання тестів, використання анотацій для опису тестів, обра- лення винятків, що виникли в тесті, і т.д.

Оскільки в процесі реалізації програми було прийнято рішеняня відмовитися від ORM-бібліотек, для роботи з базою даних було написано велику кількість коду, в якому легко допустити помилки, зв’язані з використанням прекомпілірованние SQL-виразів. У зв'язку з цим було вирішено приділити особливу увагу тестуванню компонента, призначеного для роботи з базою даних.

Висновки за розділом 3

На основі аналізу методів захисту були вирішені наступні завдання:

- реалізовано сервер системи «Gang!» за допомогою платформи Firebase;
- реалізовано мобільний застосунок для платформ IOS та Android;

- реалізовано механізм авторизації за поштою та паролем;
- реалізована реєстрація аккаунту за номером телефону;
- реалізовано розмежування доступу до інформації;
- реалізовано механізм прив'язки дій до унікального ідентифікатору користувача (UID);
- реалізовано механізм хешування паролів.

ВИСНОВКИ

У даній дипломній роботі було досліджено механізми захисту мобільних застосунків задля реалізації власного захищенного мобільного застосунку Gang!

На початку проведення дослідження було наведено основні терміни та визначення, а також було з’ясовано, які загрози та вразливості можуть бути у мобільних застосунках. Після дослідження вразливостей булу з’ясовано, які методи захисту існують та як ці методи можна реалізувати.

На основі аналізу механізмів захисту мобільних застосунків було реалізовано власний мобільний застосунок Gang!, в якому було враховано всі основні вразливості та механізми захисту мобільних застосунків.

Було реалізовано механізми захисту як клієнт- так і серверної частини. Була проведена реалізація захисту персональних даних, розмежування доступу до інформації та розроблено механізм захисту паролів.

Код системи склав понад 40000 рядків коду на мові Swift, більше 25000 рядків на мові Kotlin і близько 6000 рядків на мові розмітки XML.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ешіна С.С. Порівняльний аналіз ORM-бібліотек для платформи Android на основі критерію продуктивності. // Інформатика: проблеми, методологія, технології матеріали XV міжнародної науково-методичної конференції .

2. 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.

3. Малишкіна Е.А. Удосконалення маркетингових інструментів в інтернет-бізнесі як фактор найбільш ефективного впливу на споживача. // Соціально-економічні явища і процеси.

4. Lamoda. Додаток в App Store «Lamoda» [Електронний ресурс] / Lamoda. – 2020. – Режим доступу до ресурсу: https://apps.apple.com/ru/app/lamoda-одежда-и-обувь-онлайн/id777645417.

5. Grupa OLX. Додаток в App Store «OLX» [Електронний ресурс] / Grupa OLX. – 2020. – Режим доступу до ресурсу: https://apps.apple.com/ua/app/olx-сервис-объявлений-1/id663217552?l=ru.

6. Farfetch. Додаток в App Store «Farfetch» [Електронний ресурс] / Farfetch. – 2020. – Режим доступу до ресурсу: https://apps.apple.com/ua/app/farfetch-мода-премиум-класса/id906698760?l=ru.

7. Додаток в App Store «Prom».

8. Додаток в App Store «Lipe».

9. Додаток в App Store «Kasta».

10. Додаток в App Store «Alibaba».

11. Додаток в App Store « Wildberries ».

12. Додаток в App Store « AliExpress ».

13. Харді Б., Філліпс Б. Програмування під Android.

14. Шевченко Д. версійна міграція структури бази даних: ос новні підходи.

15. Android Developers Dashboard. [Електронний ресурс] URL: https://developer.android.com/about/dashboards/index.html

16. Android Development Tool. [Електронний ресурс] URL: https://developer.android.com

17. Swift documentation. [Електронний ресурс] URL: https://swiftbook.ru

18. Firebase Cloud Messaging. [Електронний ресурс] URL: https://firebase.google.com/docs/cloud-messaging/

19. Material design guidelines SendGrid. [Електронний ресурс] URL: https://material.io/guidelines/material-design/introduction.html

20. OpenShift: PaaS by Red Hat. [Електронний ресурс] URL: https://www.openshift.com/ (дата звернення: 20.02.2017).

21. Beginning IPhone Development with Swift: Exploring the IOS SDK

22. Swift. Основи розробки додатків під iOS, iPadOS і macOS Василь Усов

23. OneSignal. [Електронний ресурс] URL: https://onesignal.com .

24. SQLite database engine. [Електронний ресурс] URL: https://www.sqlite.org/

25. https://console.firebase.google.com

26. https://habr.com/ru/post/277941/

27. https://firebase.google.com/docs

28. https://datami.ua/bezpeka-i-kiberbezpeka-smartfoniv/

29. https://www.ptsecurity.com/ru-ru/research/analytics/mobile-application-security-threats-and-vulnerabilities-2019/

30. https://ru.wikipedia.org/wiki/Безопасность_приложений

31. Книга Вячеслава Семенчука «Мобильное приложение как инструмент бизнеса»

32. https://play.google.com/store/apps/details?id=alexey.poedinok.gang_

33. http://itzashita.ru/mobilnyie-ustroystva/problemyi-bezopasnosti-mobilnyih-ustroystv-sistem-i-prilozheniy-chast-5.html

34. http://lib.itsec.ru/articles2/25kadr/bezopasnost-mobilnyh-bankovskih-prilozheniy

35.https://cyberleninka.ru/article/n/issledovanie-informatsionnoy-zaschischennosti-mobilnyh-prilozheniy

36.https://studentlib.com/chitat/diplom-203181-razrabotka_mobilnogo_prilozheniya_informacionnoy_podderzhki_deyatelnosti_servis_inzhenera.html

37.https://coderoad.ru/53092640/Лучшая-практика-для-шифрования-Firebase-Realtime-Database-с-помощью-Google

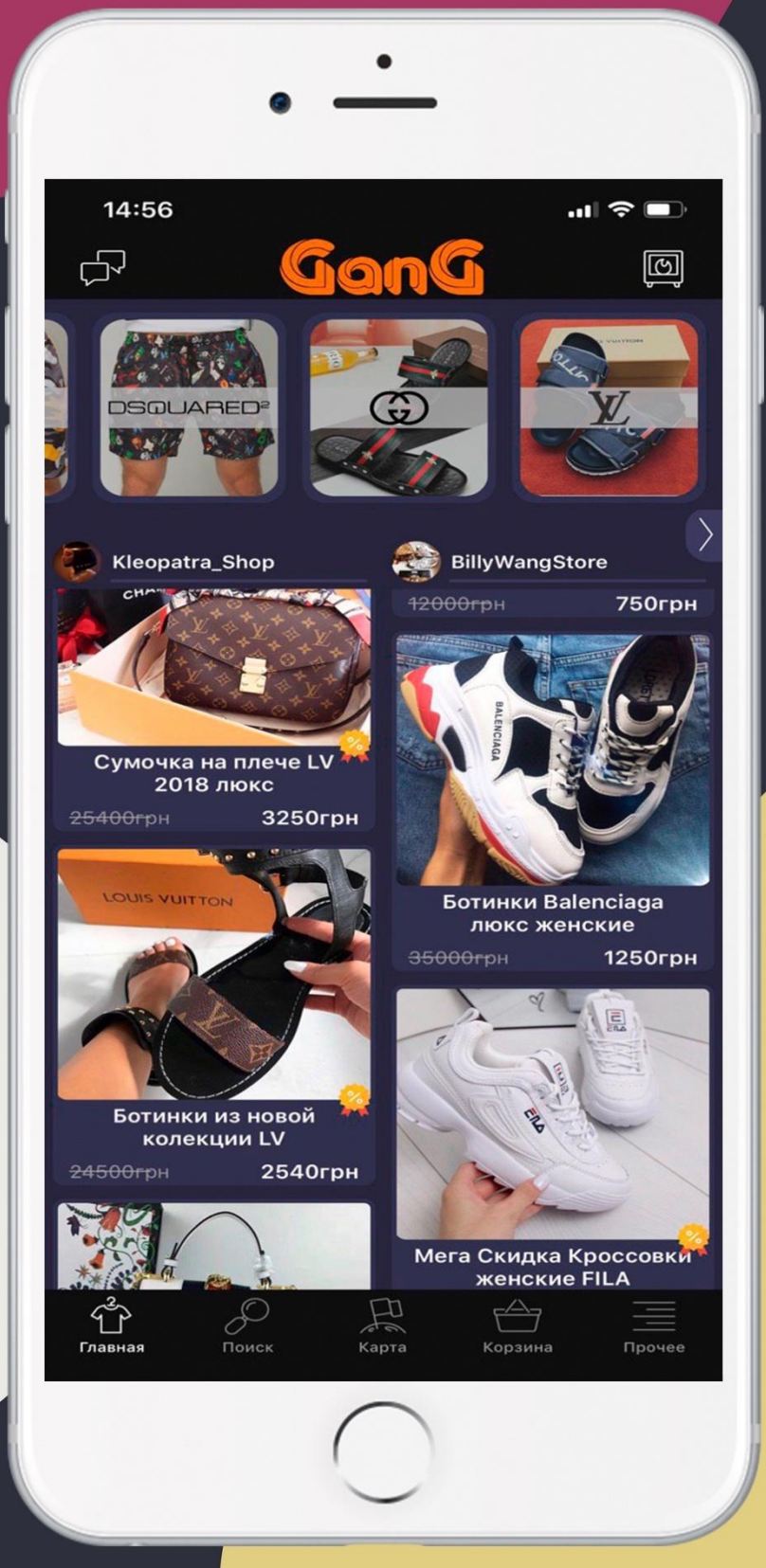
38. https://smartface.io/10-most-common-app-security-mistakes/

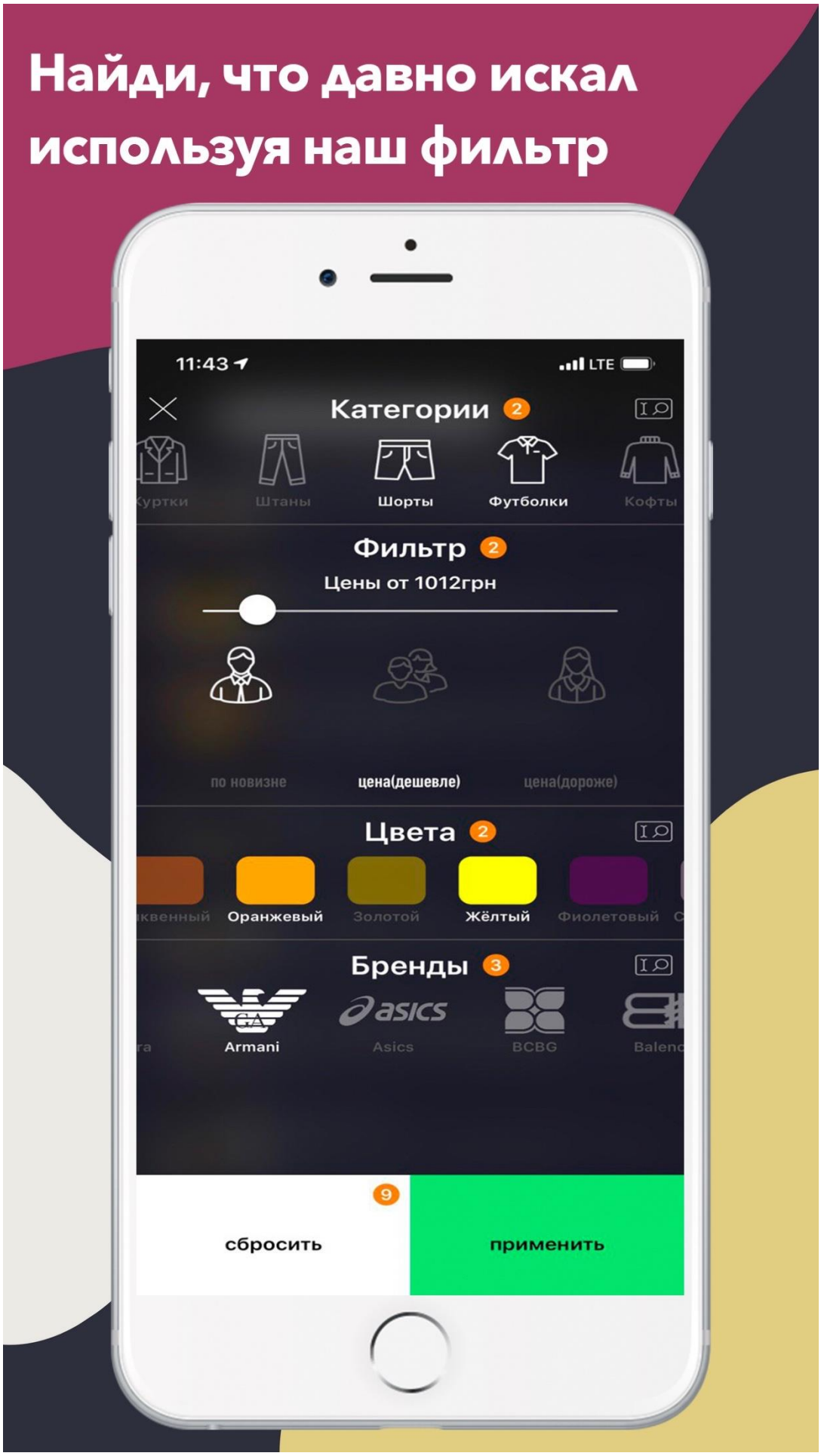
39. https://cryptoworld.su/uyazvimosti-mobilnyx-prilozhenij-i-ix-ustranenie/#4

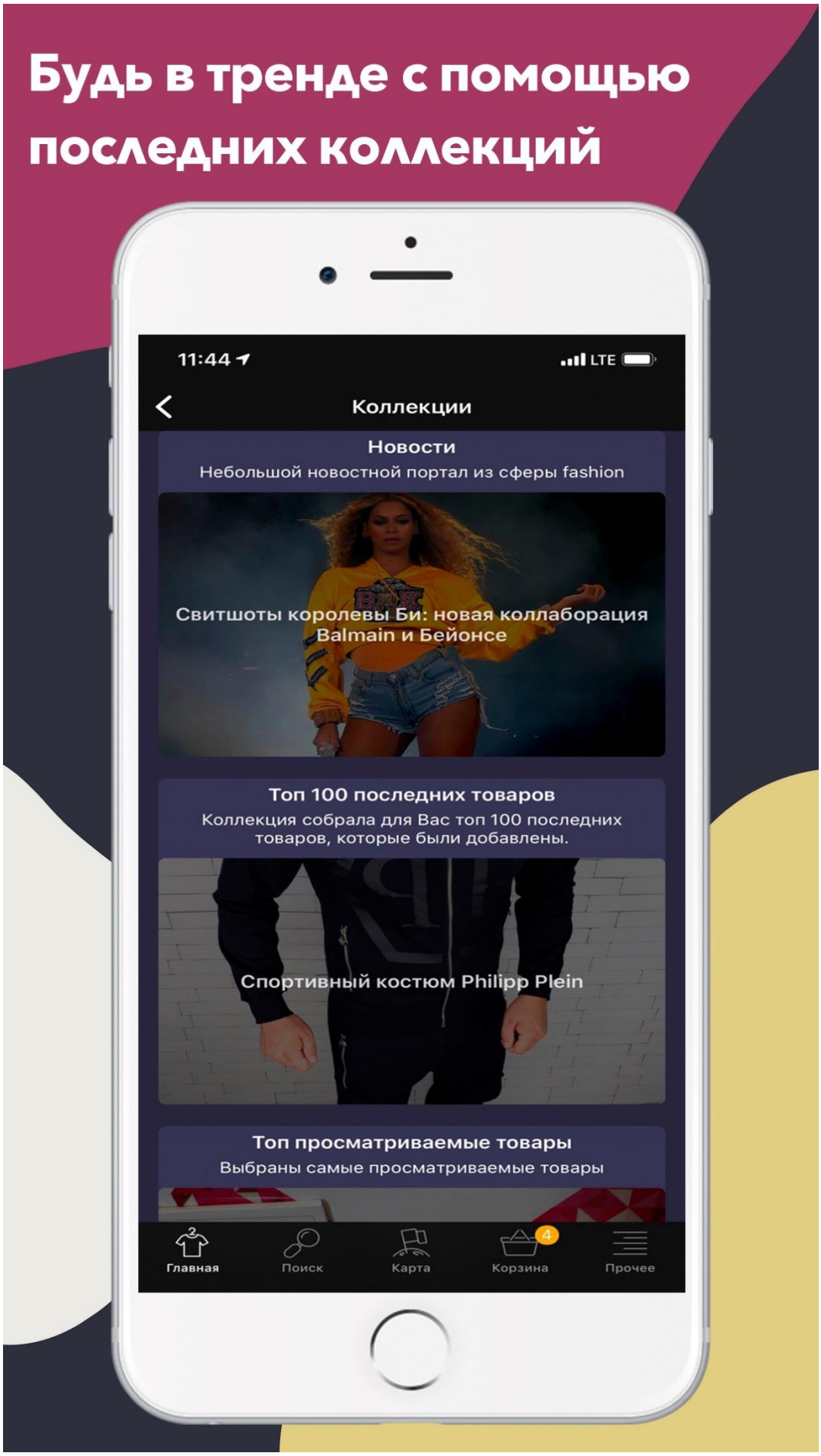
40. https://tproger.ru/articles/application-security/

ДОДАТОК 1

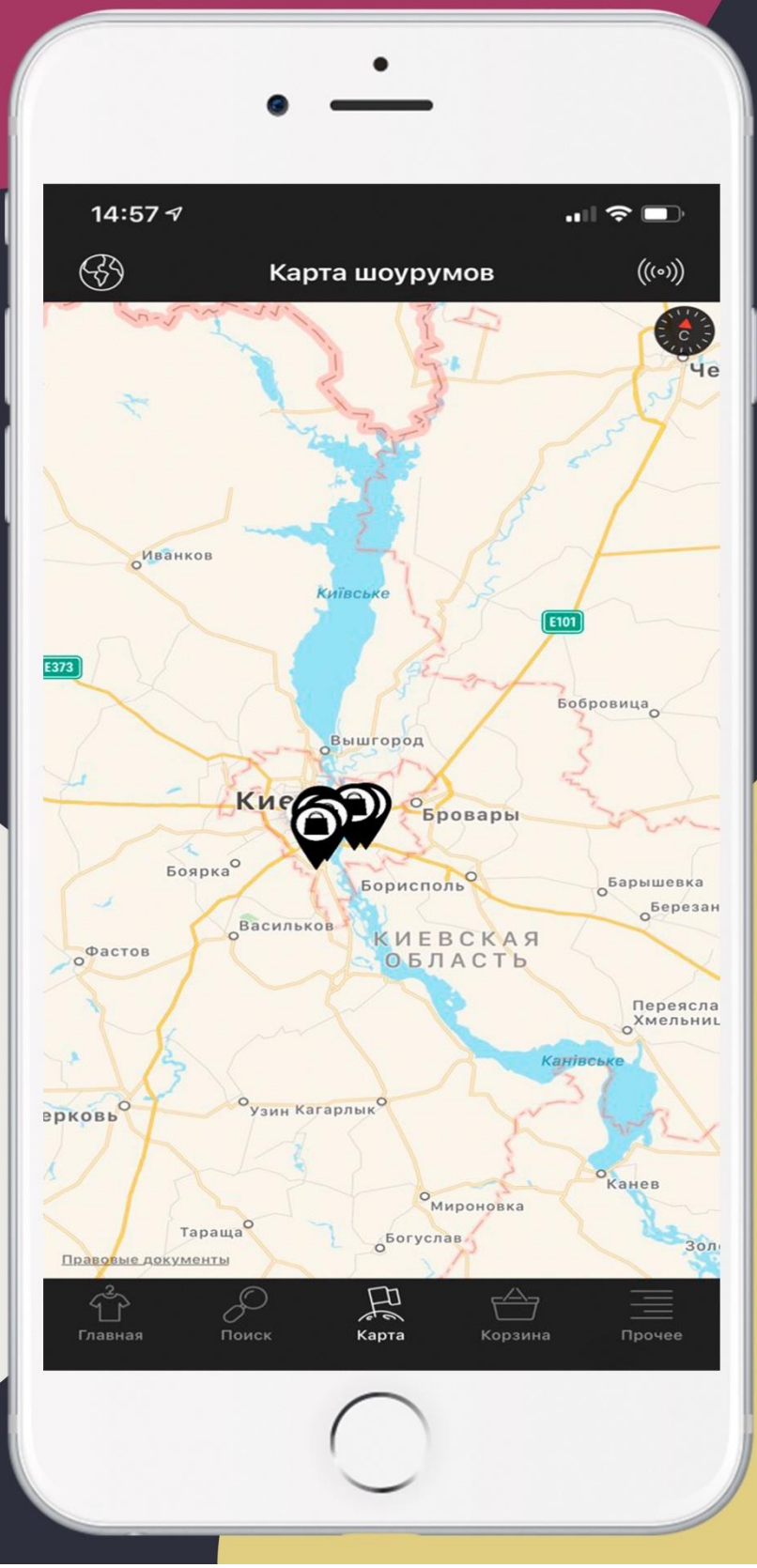
Удобная лента новинок



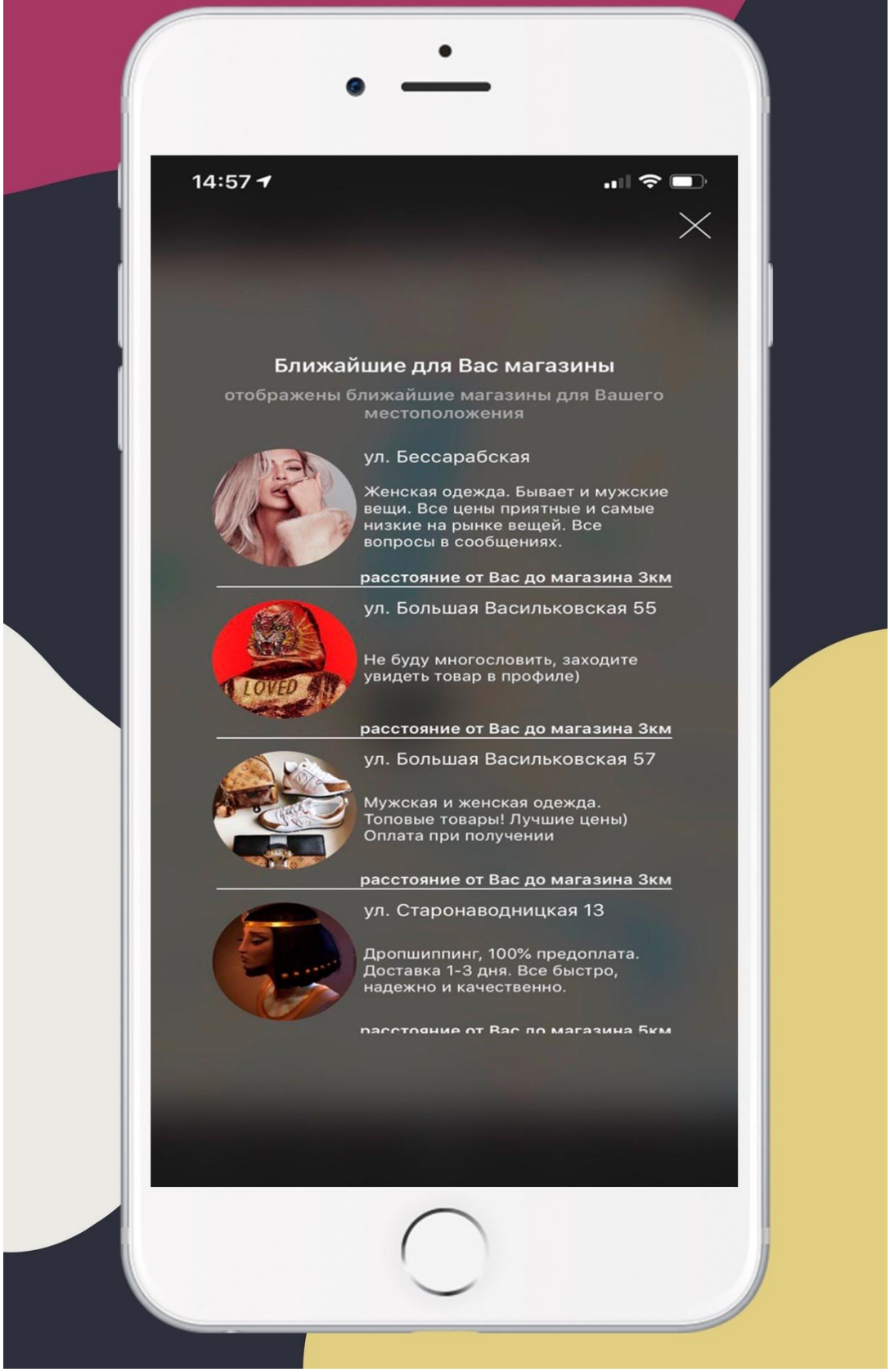

Найди, что давно искал
используя наш фильтр
11:43
LTE
Категории 2
Куртки
Штаны
Шорты
Футболки
Кофты
Фильтр 2
Цены от 1012грн
по новизне
цена(дешевле)
цена(дороже)
Цвета 2
Оранжевый
Золотой
Жёлтый
Фиолетовый
Бренды 3
Armani
Asics
BCBG
9
сбросить
применить


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
Корзина
4
Прочее

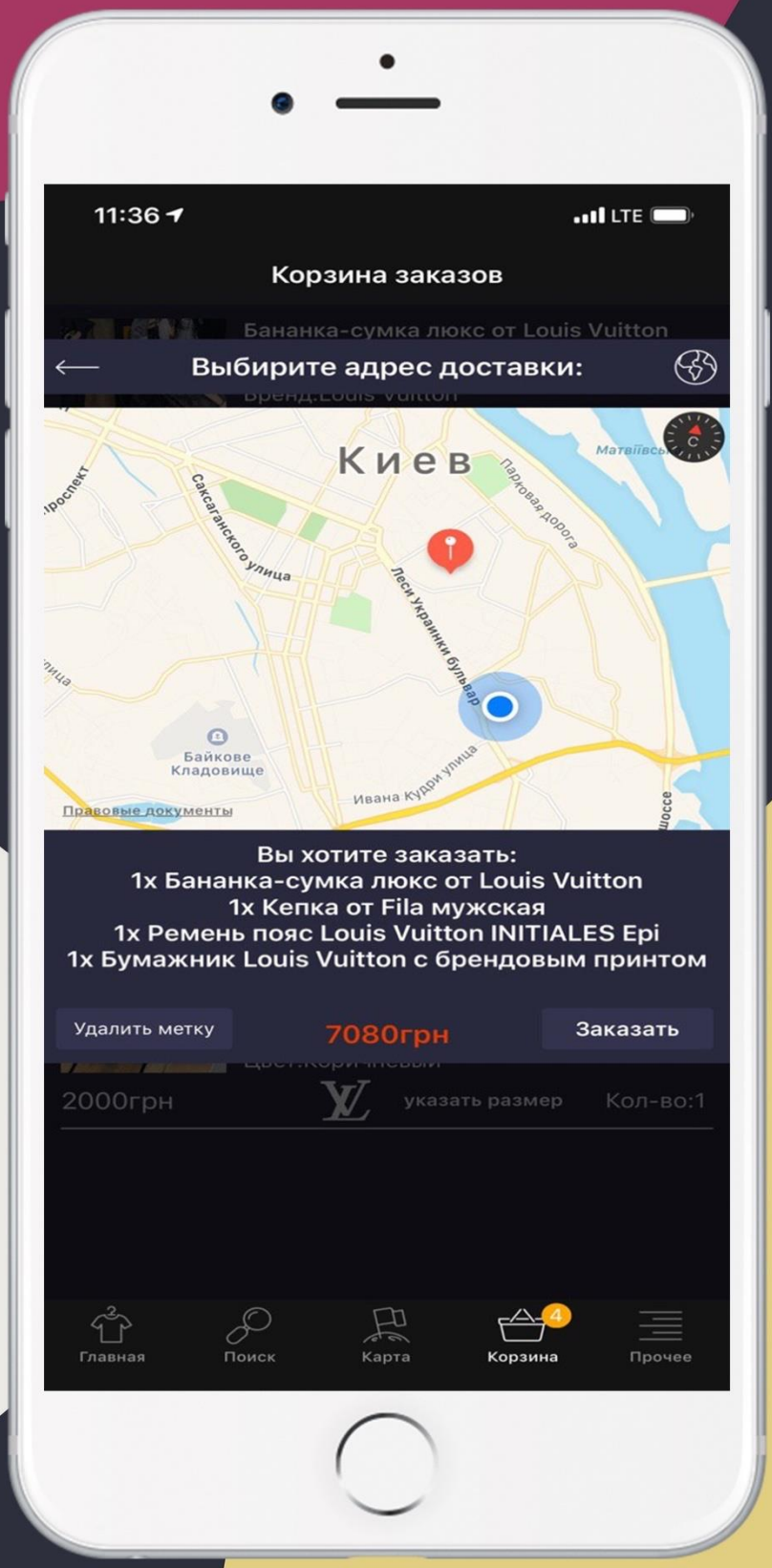
Используй карту для достижения своей цели



Выбери магазин поблизости



Нет времени на шоппинг?
Укажи адрес доставки



Уникальная система статусов заказа