Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій

УДК

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема «Генератор мобільних додатків для інтернет-магазинів»
Спеціальність 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА
БР.ІПЗ- .00.00.000

Студент
ІПЗ-41, Олексій ШЕЙКО .
(шифр групи) (підпис) (дата) (розшифровка підпису)

Науковий керівник
к.т.н, доцент, Сергій ДОЦЕНКО .
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту
з питань нормоконтролю
Завідувач кафедри
д.т.н., професор, Олексій БИЧКОВ .
(посада) (підпис) (дата) (розшифровка підпису)

2021 .
(рік)

Рішенням Екзаменаційної комісії

випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова Екзаменаційної комісії

професор, доктор техн. наук Бондарчук А.П.

АНОТАЦІЯ

Шейко Олексій Анатолійович виконав дипломну роботу – проект на тему «Генератор мобільних додатків для інтернет-магазинів» за спеціальністю 121 “Інженерія програмного забезпечення”.

В дипломній роботі проведено аналіз сучасних методів розробки мобільних додатків. Була поставлена мета - розробити програму для автоматичного складання за запитом мобільного додатку, який може змінюватися таким чином, що підсумковий файл APK буде підходити для переважної більшості інтернет-магазинів.

Ключові слова: мобільний додаток, інтернет магазин, конструктор.

ABSTRACT

The degree project: “Generator of mobile applications for online stores” was completed by Sheiko Oleksii Anatoliyovych, specialty 121 «Software engineering».

In the thesis the analysis of modern methods of development of mobile applications is carried out. The goal was to develop a program for automatic compilation on request of a mobile application, which can be changed in such a way that the final APK file will be suitable for the vast majority of online stores.

Keywords: mobile application, online store, builder

ЗМІСТ

ВСТУП

В даний час частка інтернет-магазинів роздрібної торгівлі зростає. Основні гравці ринку вже розробили офіційні мобільні додатки для збільшення кількості покупок, лояльності клієнтів, а також для того, щоб покупці могли робити покупки швидше і в дорозі. Але є значна кількість невеликих інтернет-магазинів, які не можуть дозволити собі не тільки наймати розробників, але навіть замовити розробку мобільного додатків через аутсорсингові компанії через високу вартість.

Змінний проект мобільного застосування для інтернет-магазину, яке змінює зовнішній вигляд для покупця в залежності від переданих значень при складанні - основна частина розробки, пов'язана зі збирачем. В результаті збірки, складальник виробляє файл APK для мобільної операційної системи Android, який готовий для завантаження в Google Play і повертає посилання на цей файл назад в запитуючу систему.

Причин автоматизувати розробку додатків для інтернет-магазинів багато:

Більшість вже існуючих додатків інтернет-магазинів мають схожі інтерфейси і підходи - категорії товарів, опис, кошик покупок, оформлення замовлення і т. Д. І відрізняються лише зовнішнім виглядом і іноді навігацією.

Розробка мобільного додатку вручну розробниками дозволяє створити більш персональне додаток, але і тягне величезні витрати на розробку, які не можуть собі дозволити невеликі інтернет-магазини. Також розробка займає тривалий час, і є можливість зіткнутися з величезною кількістю дефектів в програмному забезпеченні, що призведе до невдоволення користувачів.

Додаток підвищує довіру до інтернет-магазину, збільшує швидкість оформлення замовлень.

В рамках даної випускної кваліфікаційної роботи була поставлена мета - розробити програму для автоматичного складання за запитом мобільного додатку, який може змінюватися таким чином, що підсумковий файл APK буде підходити для

переважної більшості інтернет-магазинів. Для досягнення цієї мети необхідно вирішити такі завдання:

Вивчити поширені способи розробки програми таким чином, що воно буде легко змінним під різні інтернет-магазини;

Провести аналіз підходів до розробки додатків під операційну систему Android і визначитися з набором бібліотек та інструментів, мовою програмування;

Розглянути підходи до автоматизації тестування програми і вибрати відповідний підхід;

Розглянути підходи збірки додатку через інтерфейс командного рядка в APK файл, готовий до поширення через магазин додатків Google Play і вибрати відповідний підхід;

Розглянути підходи розміщення збирача в хмарної віртуальній машині в стійкою до збоїв середовищі;

Розробити структуру взаємодії мобільного додатка з даними інтернет-магазину (продукти, категорії і т.д.), архітектуру мобільного застосування, необхідні умови для збирача;

Розробити програму відповідно до розробленої архітектурою;

Провести тестування і налагодження розробленої програми;

Розробити технічну документацію.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

Progressive Web App (PWA) – вебовий застосунок, який є гібридом звичайної вебової сторінки та мобільного застосунку

Android application package (APK) – формат архівних файлів-додатків для Android

Model–view–presenter (MVP) – шаблон проектування, що відділяє візуальне відображення та поведінку обробки подій у різні класи

Java virtual machine (JVM) – набір комп'ютерних програм та структур даних, що використовують модель віртуальної машини для виконання інших комп'ютерних програм чи скриптів.

1. ОБҐРУНТУВАННЯ НЕОБХІДНОСТІ РОЗРОБКИ ГЕНЕРАТОРА МОБІЛЬНИХ ДОДАТКІВ

1.1 Поняття генератора мобільного додатку

Платформи для створення мобільних додатків розрізняються між собою набором функцій, цінами, а також тим, як з їх допомогою можна зробити додаток. За останньою ознакою вони діляться на дві основні категорії:

Генератори. Це платформи, які створюють мобільний додаток на основі існуючої веб-сторінки. Генератору вказуємо URL сайту, і він автоматично створює мобільний додаток з тими ж розділами і контентом, що і на сайті.

Конструктори. Це платформи, які дозволяють зібрати додаток самостійно з готових елементів, а контент для нього створять майбутні користувачі. Конструктори містять готові шаблони і елементи інтерфейсу, а також фрагменти функціональності, наприклад, геопозиціонування, відправка повідомлень, робота з банківськими картами і багато іншого.

Є два типи програм, які вміють створювати ці платформи:

Гібридні (PWA). Це, фактично, додатки під веб, адаптовані під екран мобільного пристрою. Вони відкриваються на смартфоні за допомогою браузера.

Нативні. Це, власне, додатки, які встановлюються в операційну систему мобільного пристрою. Нативні додатки найбільш зручні для користувача і вигідні для підприємця.

Створення програми саме по собі може бути безкоштовним, потім є два шляхи. По-перше, можна купити у сервісу його вихідні коди і самостійно підтримувати їх і поширювати додаток. Крім того, можна купити платну підписку, і тоді команда сайту сама опублікує додаток в App Store / Google Play і буде підтримувати його.

Крім плати за підтримку, також доведеться купити аккаунт в App Store або Google Play, який коштує $ 99 і $ 25 відповідно. Щоб окупити витрати, у багатьох

платформ є програми лояльності, які дозволяють не тільки зробити додаток, а й заробляти на ньому - наприклад, підключивши рекламу.

1.2 Огляд наявних генераторів мобільних додатків

На даний час є велика кількість генераторів мобільних додатків, що різняться між собою функціональністю, можливістю, інструментами, вбудованими опціями. Розглянемо детальніше найбільш використовувані.

Appy Pie

Найдинамічніша платформа для створення додатків в світі. Ця платформа особливо корисна для тих, хто вперше береться за додатки та є новачком в цій галузі. Одна з причин, чому ця платформа так швидко завоювала популярність, полягає у великій кількості пропонованих унікальних функцій. Наприклад, за допомогою Appy Pie можна додати в додаток вбудовані покупки, рекламу, завантажити електронні книги або інший контент, підключити бази даних, інтегрувати соціальні мережі, створити додаток для обміну миттєвими повідомленнями і так далі.

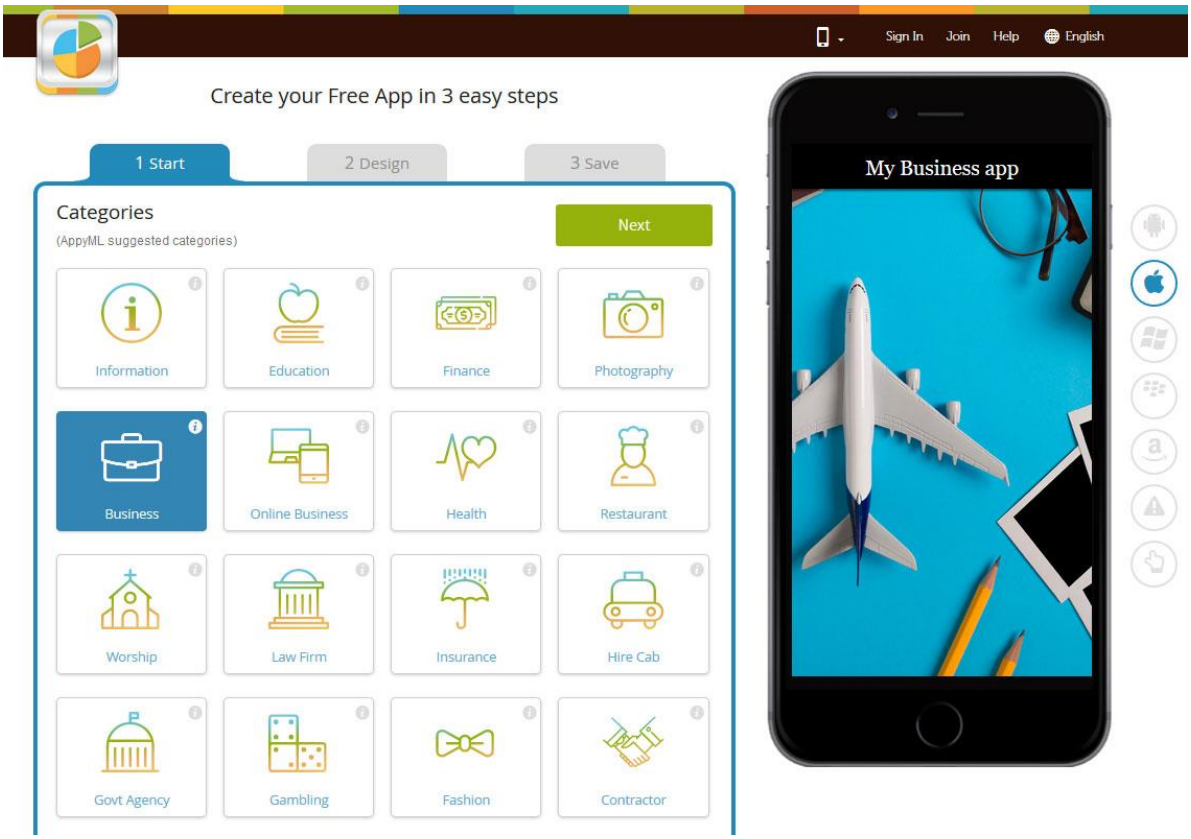


Рисунок 1.1 – Платформа Appy Pie

Shoutem

Shoutem - один з кращих продуктів на ринку, і він постійно зростає з моменту відкриття в 2011 році. У своїй останній версії V5 оновлено платформу, значно поліпшено призначений для користувача інтерфейс. Платформа містить шаблони з великою кількістю варіантів настройки і кожен додаток може отримати унікальний зовнішній вигляд і дизайн. Додатки, зроблені в цьому конструкторі, будуть не тільки красивими, але і функціональними. Ця платформа для створення додатків особливо хороша для додатків, пов'язаних з заходами, а крім того відмінно підходить для спільнот, так як завдяки функції соціальної стіни (Social Wall) користувачі можуть ділитися коментарями та фотографіями.

Shoutem робить свій код відкритим і доступним для розробників. Це дозволяє їм залучати більше програмістів для створення додаткових розширень або розробки більшої кількості функцій для конструктора, так як це в цій галузі вони поки відстають. В цілому, дизайн, користувальницький досвід і шаблони Shoutem гарні, але платформа і раніше обмежена в кількості функцій, пропонованих для створення додатків.

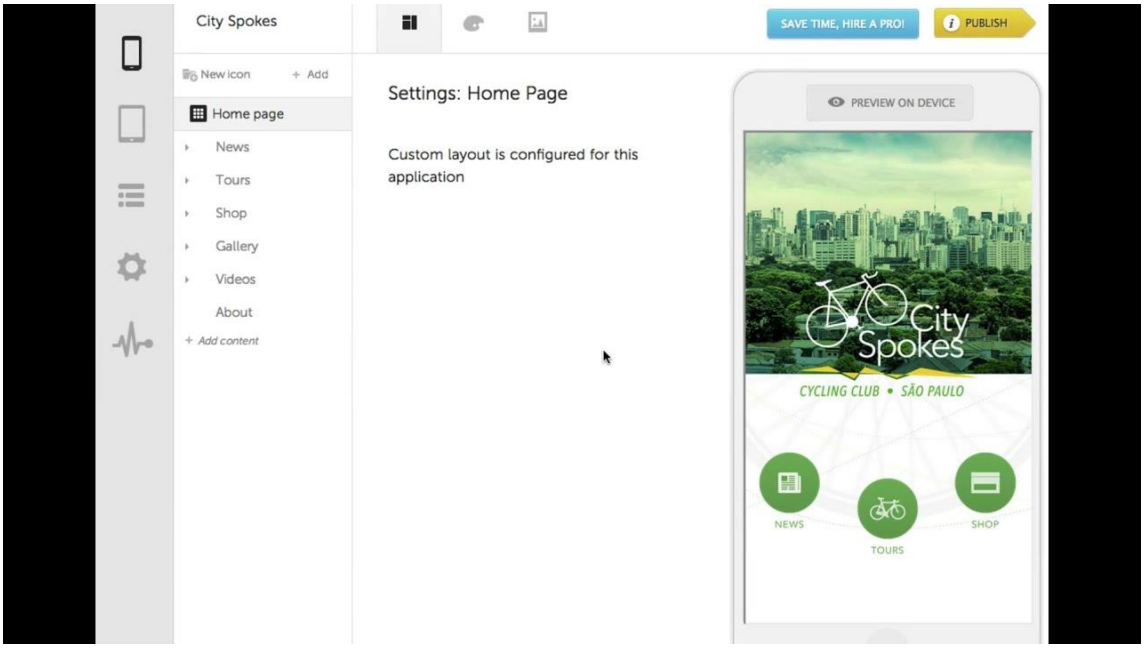


Рисунок 1.2 – Платформа Shoutem

Swiftic

Заснований в 2010 році, Swiftic починався як ізраїльський Como, і з тих пір на ньому по всьому світу зроблено більше мільйона додатків.

Компоненти або будівельні блоки, що надаються цією платформою створення додатків, різноманітні і з їх допомогою можна зробити карту лояльності, планувальник зустрічей, e-commerce магазин, можете збирати огляди і оцінки від користувачів або реалізувати додаток для заходу. Більшість додатків, створених на їхній платформі, належать ресторанам, музичним групам і іншим організаціям, які проводять заходи.

Платформа пропонує сім різних шаблонів, які можна комбінувати з шістьма різними стилями навігації. Що робить додаток по-справжньому персональним, так це кольори додатки, фонові зображення і іконки, які можна змінити за своїм бажанням. Редактор конструктора продуманий і простий, а кількість функцій і варіантів дизайну, які вони надають, велика.

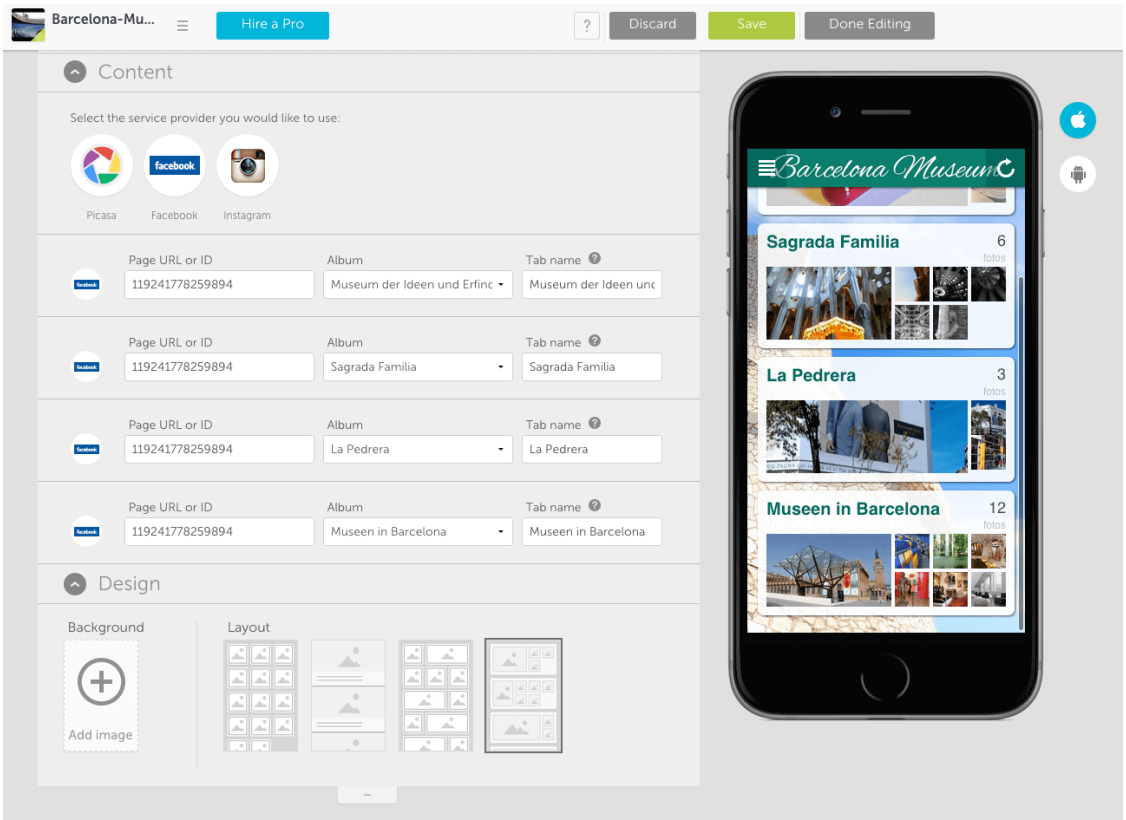


Рисунок 1.3 – Платформа Swiftic

GoodBarber

Все в цьому конструкторі додатків, починаючи з його назви, вражає. Базована на французькому острові Корсика, платформа для створення додатків пропонує одні з найбільш вражаючих шаблонів. Крім зовнішнього вигляду, конструктор також реалізує деякі самі передові функції - соціальні мережі, чати, геофенсінг, iBeacons і багато іншого.

Платформа з гордістю демонструє додатки, створені на їхній платформі, щоб можна було зрозуміти якість і можливості, які вона пропонує.

Мало того, що їх шаблони красиві, але вони також пропонують досить конкурентоспроможну ціну для нативних додатків. Відмінні додаткові функції, такі як push-повідомлення, чати і т.д дають велику гнучкість при створенні додатків. Однак одним з недоліків є відсутність власного компонента інтернет-магазину, але є можливість інтегрувати сторонні магазини, на кшталт Amazon, Etsy, Shopify і т.д.

Рисунок 1.4 – Платформа GoodBarber

BuildFire

BuildFire - одна з найбільш надійних платформ, за допомогою якої вже більше 30 000 компаній створили свої додатки. Більшість їхніх клієнтів - підприємства

бренди. BuildFire називає себе однією з провідних платформ на ринку прискореної розробки додатків.

Зручні панелі управління і адміністрування полегшують процес випуску оновлень. Платформа популярна серед клієнтів завдяки простоті використання, можливостям швидкого перенастроювання, а також широким можливостям зміни програми. Можна вносити зміни на льоту і навіть тестувати ці зміни в режимі реального часу.

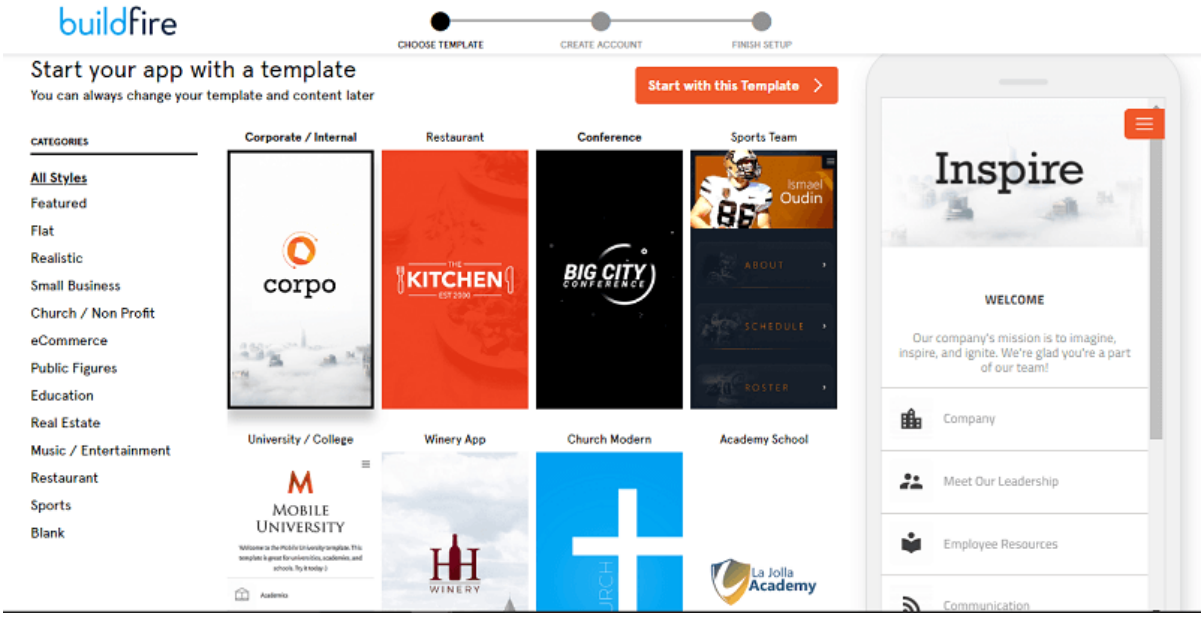


Рисунок 1.5 – Платформа BuildFire

Незважаючи на великий перелік можливостей у даних платформ є серйозний недолік, а саме, платна основа. Саме цей фактор робить неможливим використання даних платформ для власних потреб чи в навчальних цілях.

Саме тому актуальною є розробка власного генератора мобільних додатків.

1.3. Постановка задачі дослідження

Рішення, що розробляється в рамках цього проекту покликане вирішити проблеми пов'язані з розробкою мобільних додатків, так як воно буде:

1) Веб-сайтом, на якому може зареєструватися і створити свій додаток;

2) Безкоштовним для студентів, звичайних користувачів та невеликих інтернет-магазинів, які мріють створити свій додаток для збільшення продажів;

3) Зручним в кастомізації і отриманні готового додатка - за допомогою вибору квітів, шрифтів, модулів і різноманітних параметрів;

4) Простим і доступним в користуванні - для створення програми не потрібно буде знати особливостей мобільної системи або знати мови програмування - система згенерує всі автоматично;

5) Надійним - для додатків написано велику кількість тестів для впевненості в тому, що у різних користувачів буде мінімальна кількість помилок.

Проект являє собою систему з двох частин - веб-сайту - бібліотеки модулів, де і відбувається вся взаємодія користувачів з системою і вибір зовнішнього вигляду програми та інших параметрів і збирача проекту мобільного додатка.

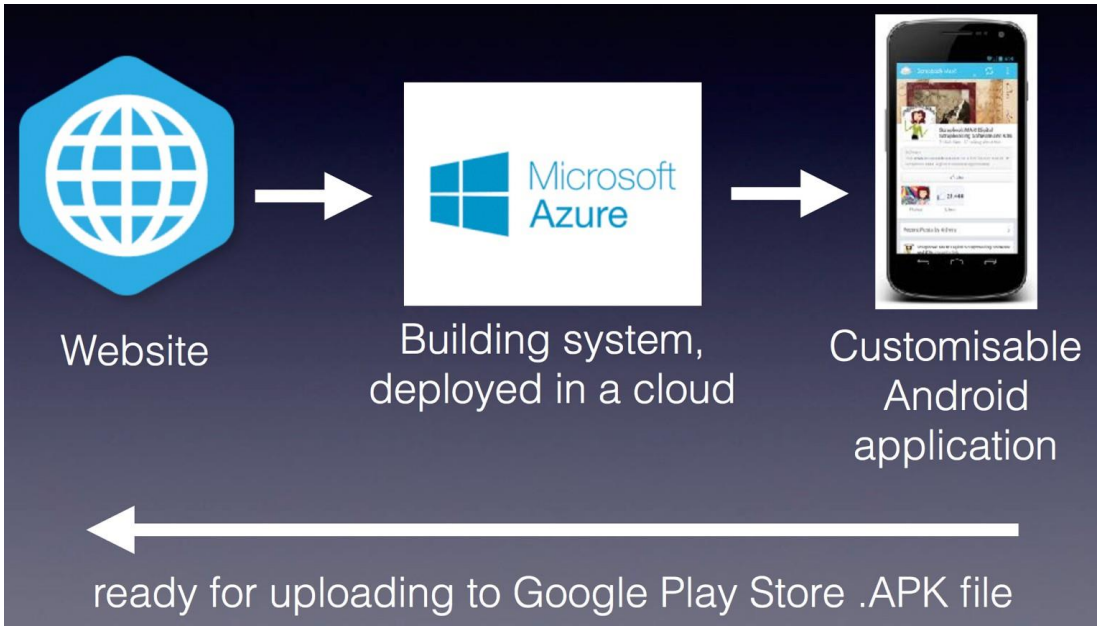


Рисунок 1.6 – Загальна схема роботи системи

Розглянемо збирач і мобільний додаток для операційної системи Android і взаємодія з бібліотекою модулів.

"Website" - бібліотека модулів, "Building system" - складальник в хмарі, пов'язаний з "Customisable Android application", проектом мобільного додатку.

Процес роботи виглядає наступним чином. В системі присутні два зовнішніх інтерфейсу (Front-end) - для розробників і кінцевих користувачів.

Роботу з системою починає розробник, перейшовши на вебсайт розробленої платформи. В даному інтерфейсі розробник реєструється, вибирає параметри програми (такі як кольори, можливість або неможливість оплати замовлення через додаток, назва програми) і через даний інтерфейс викликає збірку. Зовнішній інтерфейс передає запит по HTTP-протоколу збирачеві, який розташований на віддаленому сервері, в запиті передає лише необхідні на даному етапі параметри - ідентифікатор додатку в цій системі, ім'я пакета додатка, назва програми.

Складальник отримує всі параметри від зовнішнього інтерфейсу і, модифікуючи відповідним чином проект мобільного додатка, збирає його з системою Gradle.

Після завершення збирання, у зовнішній інтерфейс для розробника передається посилання для завантаження готового файлу мобільного додатка APK.

Другим зовнішнім інтерфейсом (Front-end) є мобільний додаток. Мобільний додаток може бути розміщено в магазині додатків Google Play, і кінцеві користувачі – відвідувачі додатку.

2. АРХІТЕКТУРА ГЕНЕРАТОРА МОБІЛЬНИХ ПРОГРАМ ТА ТЕХНОЛОГІЇ ЙОГО РОЗРОБКИ

2.1 Патерни побудови архітектури додатків Android

Розробка для ОС Android ведеться на мові Java. Ніяких обмежень на архітектуру, при цьому, від компанії Google в Android SDK немає. Єдине, що варто брати до уваги

- обмеження в використовуваних інтерфейсів SDK - наприклад, Activity, Fragment, Content Resolver, Service і т.д. [6]

В даний час в Android розробці існує три патерну при організації архітектури додатку - MVP (Model-View-Presenter), MVC (Model-View-Controller), MVVM (Model-View-ViewModel). Перший (MVP) є оптимальним і найпопулярнішим на момент написання даного проекту. Другий (MVC) є застарілим, а MVVM недостатньо адаптований для Android-розробки. [3]

У таблиці 2.1 розглянуто ці три підходи.

Таблиця 2.1. Порівняння MVP і MVC

MVP	MVC
Удосконалена форма MVC	Один зі старих патернів, використовуваних для підтримки поділу відповідальності класів
View управляється користувачем, і View викликає методи Presenter при події, викликаних користувачем	Controller управляється користувачем, і цей же Controller управляє моделлю (Model)
View абсолютно пасивна, все взаємодії з моделлю відбуваються через Presenter	У View є велика кількість логіки. І View може безпосередньо взаємодіяти з моделлю (Model)
Відмінно підтримує Unit-тестування	Сильно обмежений в Unit-тестування

Як можна помітити, MVP патерн (рис. 2.1) полегшує тестування Presenter і Model, тому що вони більше не пов'язані з View щільно, і логіка сильно відокремлена від відображення в моделі MVP, що робить модель дуже популярною в Android розробці в останні роки. [16]

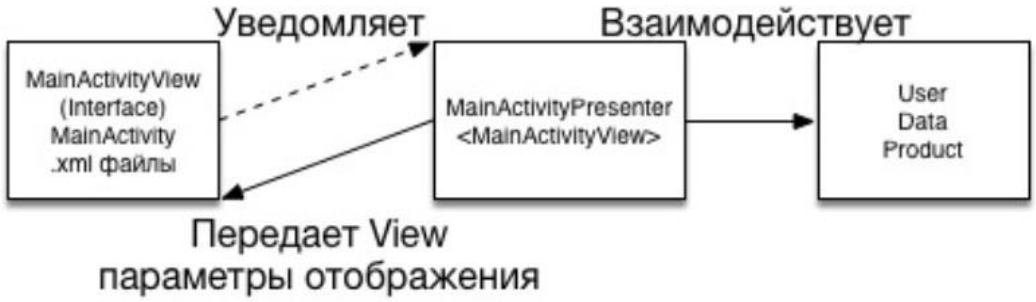


Рисунок 2.1 – Схематичне зображення патерну MVP

2.2 Використання підходу Clean Architecture при розробці мобільних додатків

З кожним роком зростає популярність системи Android і зростає кількість проектів під Android, але з ростом проектів розширюється спектр завдань і часто закладених спочатку способів для гнучкої розробки нових функцій не вистачає. З огляду на те, що в даному проекті мається на увазі велику кількість модулів і очікується зростання їх кількості в майбутньому, до архітектури в цілому слід підходити так, щоб було легко тестувати окремі модулі, не прив'язуючись до класів в Android SDK, а також щоб у майбутньому через змін не доводилося переписувати все додаток.

Тому було вирішено взяти в основу підхід Роберта Мартіна під назвою Clean Architecture. [24] Його структура зображена на рис. 2.2.

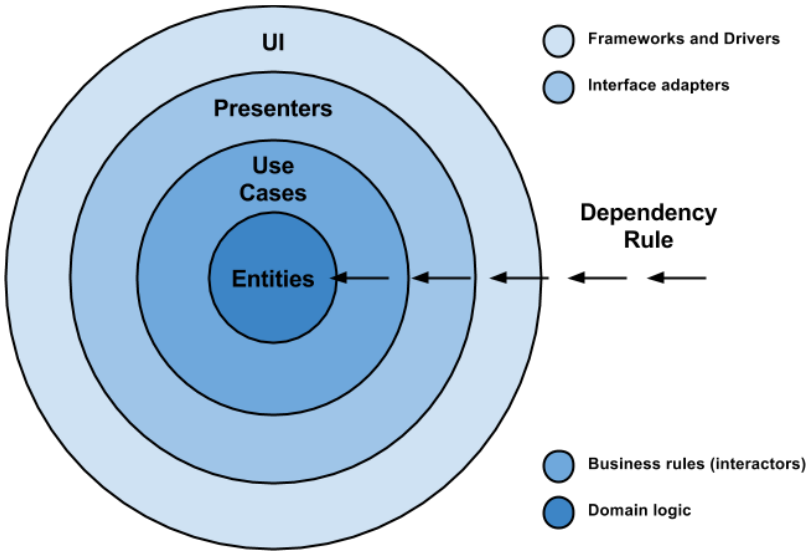


Рисунок 2.2 – Схематичне зображення підходу Clean Architecture

Основні принципи Clean Architecture перераховані тут, і пов'язані зі схемою на малюнку 2.2.

Незалежність від структури. Архітектура не залежить від бібліотек або SDK, програма не перебуває в рамках підходів, запропонованих бібліотеками.

Нестування Бізнес-логіку можна протестувати без користувальницького інтерфейсу, бази даних, веб-сервера або будь-яких інших зовнішніх елементів.

Незалежність від призначеного для користувача інтерфейсу. Інтерфейс може легко змінюватися без зміни іншої системи. Наприклад, веб-інтерфейс можна замінити на призначений для користувача інтерфейс консолі, не змінюючи бізнес-правила.

Незалежність від бази даних. Важливо, щоб можна було легко поміняти Oracle або SQL Server на Mongo, BigTable або CouchDB легко і без зміни бізнес-правил.

Розробляючи додаток так, слід розділити його на 3 основних модуля - Presentation модуль, де буде розташовуватися весь патерн MVP і відображення; Domain модуль, де будуть чисті Java об'єкти; Data модуль, де буде описано методи доступу до даних (з мережі і з збережених на пристрої файлів). Це зображено на рис. 2.3.

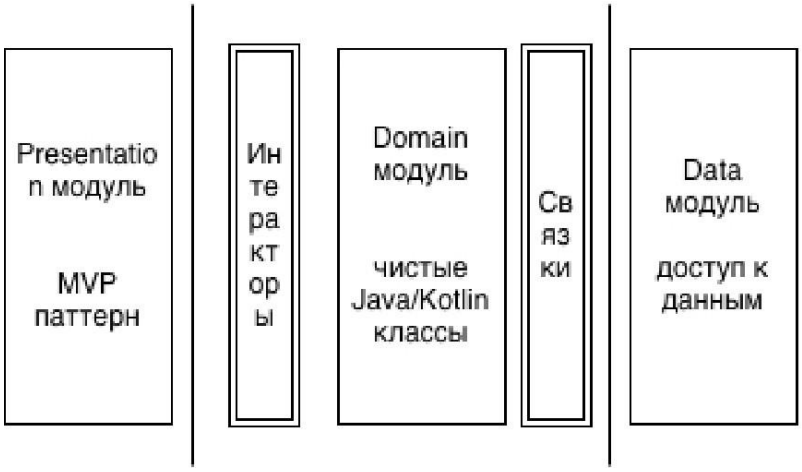


Рисунок 2.3 – Поділ Android програми на модулі по Clean Architecture

2.3 Аналіз архітектури, MVP моделі і вибраних бібліотек

Як було сказано раніше, в проекті є поділ на три модуля - data (з методами отримання даних), domain (з чистими сутностями), presentation (со всім, що стосується відображення і зв'язку архітектури з Android SDK), і на рис. 2.4 можна побачити, створені три модуля [25].

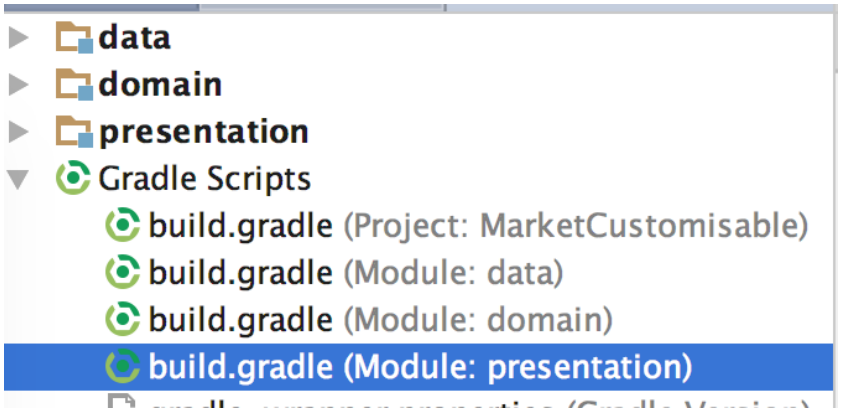


Рисунок 2.4 – Архітектурні модулі в проекті

Розглянемо по порядку кожен з архітектурних модулів.

Модуль додатку data

Перший модуль data - всі дані, необхідні для застосування, надходять з цього рівня через реалізацію паттерна Repository (його інтерфейс знаходиться в рівні domain), який вибирає різні джерела даних в залежності від певних умов.

Наприклад, при отриманні користувача за ідентифікатором, джерелом даних буде обраний кеш, якщо користувач вже існує в кеші, а якщо немає - data-модуль відправить запит на сервер для отримання користувача, а потім збереже його в кеші.

Основна ідея тут – зробити так, щоб джерело даних був абсолютно абстраговано і просто для використання.

У реалізації цього модуля в додатку розглянемо підпакети модуля, зображені на рис. 2.5.

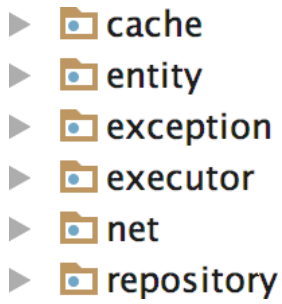


Рисунок 2.5 – Підпакети модуля data

Варто розписати підпакети і пояснити, за що відповідає кожен з них.

- cache - тут розташовані класи, які пов'язані з читанням і записом даних в кеш - робота з базою даних і ін.
- entity - тут суті, використовувані в додатку (як User, Product і т.д.), що використовуються в data модулі - для отримання з сервера / запису в кеш (базу даних).
- exception- виключення, які можуть виникати в data модулі.
- executor - класи, що відповідають за виконання процесів отримання і запису даних.

- net - тут розташовані класи, які пов'язані з отриманням даних з інтернет-сервера.
- repository- класи-репозиторії, розділені по одержуваних сутностей.

Модуль додатки domain

Другий модуль domain - містить чисті Java класи, без залежностей від бібліотек або від Android SDK. Всі зовнішні об'єкти використовують інтерфейси при зверненні до класів domain.

Тут розташовані всі об'єкти бізнес-логіки і бізнес-правила. Все реалізації інтеракторов також знаходяться тут.

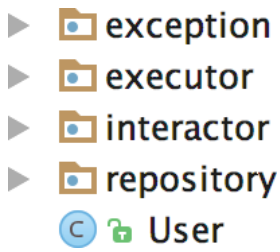


Рисунок 2.6 – Підпакети модуля domain

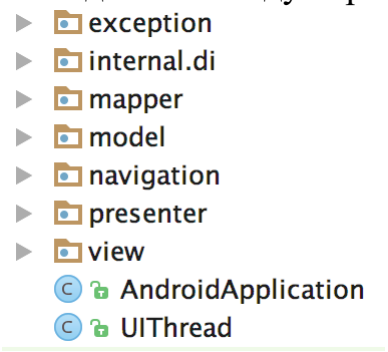
- exception- виключення, які виникають в domain модулі.
- executor - інтерфейси класів, що відповідають за виконання процесів отримання і запису даних.
- interactor- класи, що представляють собою use-case для сутностей.
- repository - інтерфейси класів-репозиторіїв, розділених по одержуваних сутностей.

Модуль додатку presentation

Останній модуль - presentation. У ньому відбувається логіка, пов'язана з відображенням. Можна використовувати будь-який шаблон, MVVM, MVC, але в даному проекті використовується MVP і це пояснено в розділі «1.4.3 MVP». У цьому модулі знаходяться елементи Android для відображення - Fragment, Activity, і немає ніякої логіки, крім логіки призначеного для користувача інтерфейсу. Керують усім класи Presenter - презентери, які звертаються до Interactor - інтеракторам за даними, які будуть відображатися на екрані.

Підпакети цього модуля такі (рис. 2.7).

Рисунок 2.7 – Підпакети модуля presentation



- exception- виключення, що виникають в модулі presentation.
- internal і підпакету di - всі компоненти і модулі, які використовуються в паттерне Dependency Injection (ін'єкція залежностей). Про використання цього патерну буде розписано далі в розділі «3.3.5 Dagger 2 і ін'єкція залежностей».
- mapper- маппера, тобто класи, які переводять суті з сутностей data і domain по суті, зручні для відображення на екрані.
- model- самі сутності, зручні для відображення на екрані.
- navigation- класи для навігації, для переходу між екранами.
- presenter- презентери, класи, в яких розташована основна логіка відображення даних. У кожного view-класу є свій презентер. Презентер отримує дані і відображає їх на екрані через view-класи.
- view- елементи Android SDK, що відображають всі на екрані. Тут Activity і

Підсумкова схема додатка виглядає так, як на рис. 2.8.

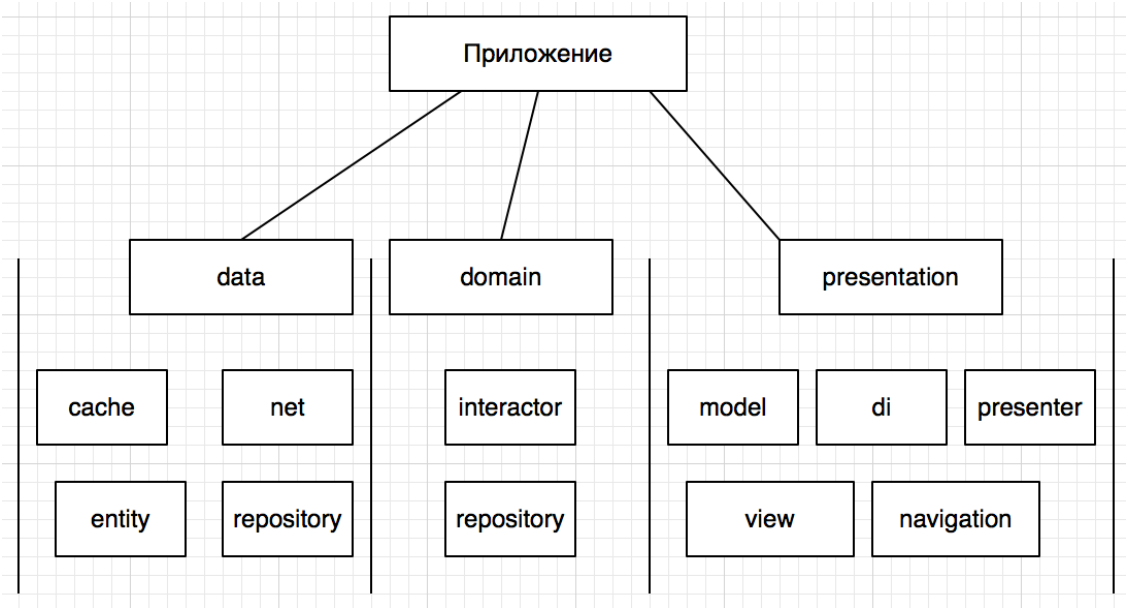


Рисунок 2.8 – Архітектура додатку

2.4 База даних Realm

Realm - база даних для мобільних пристроїв, наприклад, під керуванням ОС Android.

Це заміна стандартної базі даних SQLite, використовуваної в Android.

Однак, на відміну від SQLite, Realm містить кілька особливостей.

1) Простота використання - для створення таблиці слід всього лише позначити клас об'єктом Realm, і всі операції виробляє сам Realm. Простота використання веде за собою читаність коду, а отже - менша кількість помилок.

2) Швидкість роботи - на практиці було відзначено, що швидкість Realm більше швидкості SQLite в більш ніж 2 рази. [4] Це завдяки використанню нативних методів і класів на мові C ++ всередині бібліотеки.

3) Добре документована бібліотека- документації по Realm в рази більше, ніж за SQLite, наведені приклади використання і розписані всі методи, що дає великий простір для використання даної бази даних.

На рис. 2.9 зображена схема бази даних для кешування і зберігання даних в пам'яті пристрою.

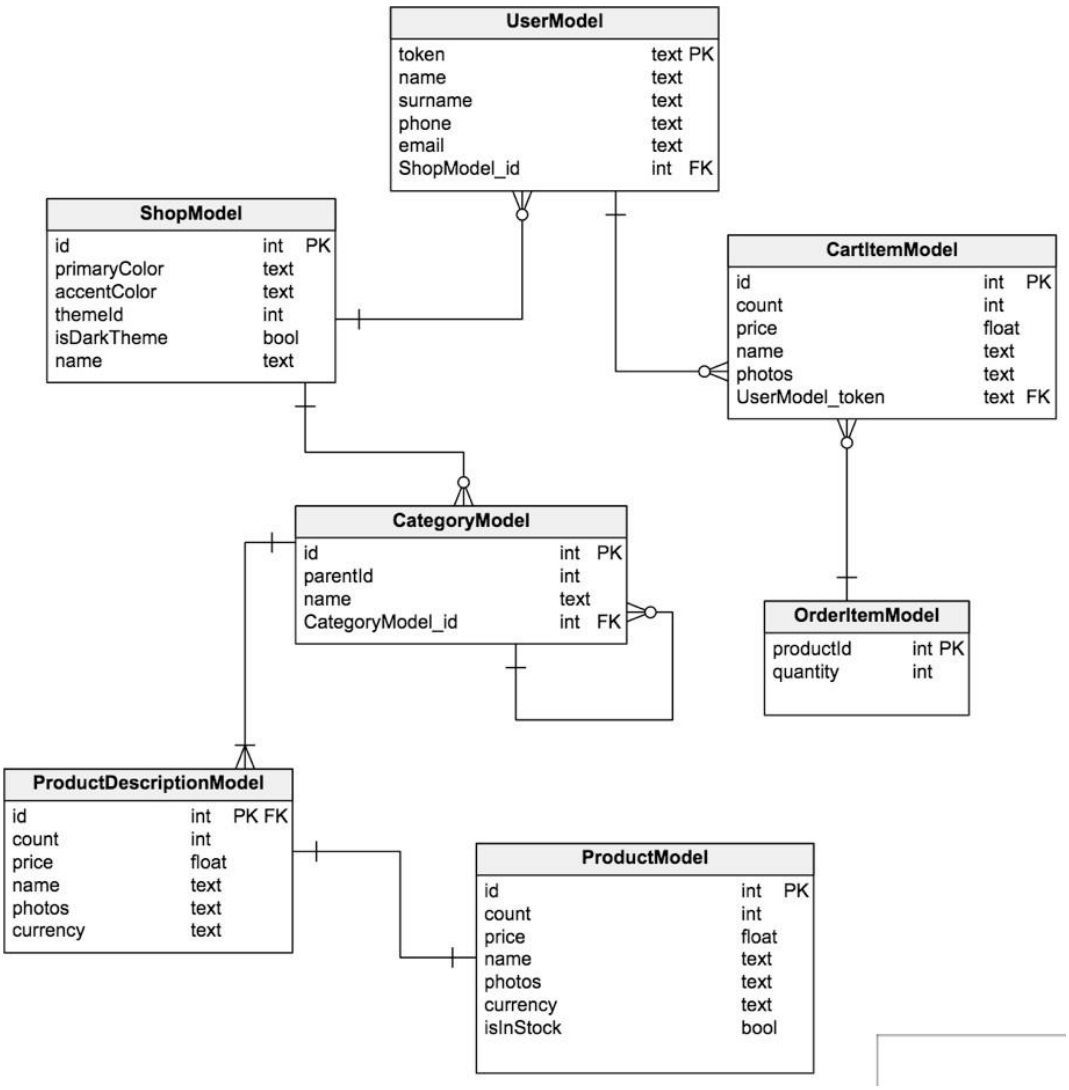


Рисунок 2.9 – Схема бази даних Realm в проекті

3. РЕАЛІЗАЦІЯ ГЕНЕРАТОРА МОБІЛЬНИХ ДОДАТКІВ

3.1 Інструменти розробки мобільного застосування

Як середовище розробки була використана Android Studio останньої на момент розробки версії. Ця IDE (integrated development environment - інтегроване середовище розробки), спеціально налаштована для розробки під Android. Android Studio доступна безкоштовна кожному під ліцензією Apache License 2.0. [13]

Android Studio заснована на IntelliJ IDEA від компанії Jetbrains і доступна на операційних системах Windows, Mac OS X, Linux.

переваги використаннясаме Android Studio складаються в наявності в середовищі розробки підтримки Gradle-збірки, особливих для Android-розробки підказок при написанні коду, емулятор Android та інших функцій.

Основна мова розробки - Java.

В якості системи контролю версій використовується Git система Github. С цією ж системи складальник отримує останню версію проекту для збірки.

3.2 Взаємодія мобільного додатка з джерелом даних

У цьому розділі розглянуто дві головні бібліотеки, які використовуються для отримання даних. Це Retrofit і rxJava.

Що таке rxJava? [14] Це reactive Java, бібліотека, яка використовує патерн Observer (спостерігач). У паттерне Observer об'єкт, званий Subject (суб'єкт), має список своїх Observer (спостерігачів), і автоматично повідомляє їх про будь-які зміни стану, як правило, викликаючи один з їх методів у спостерігачів. Ця бібліотека в основному використовується для компонування асинхронних запитів і подій. В основі вже закладені такі поняття, як низькорівневі потоки, синхронізація, потокобезпечна і паралельні структури даних,

тому розробнику, котрі використовують цю бібліотеку, не треба замислюватися про ці концепції.

Додавання в проект rxJava відбувається шляхом написання в Gradle скрипті рядки:

compile "io.reactivex.rxjava2: rxjava: 2.xy"

Де x і y - версія rxJava.

Розглянемо основні і найпростіші принципи роботи rxJava на прикладі коду на рис. 13.

```
public static void hello(String... names) {
    Observable.from(names).subscribe(new Action1<String>() {

        @Override
        public void call(String s) {
            System.out.println("Hello " + s + "!");
        }

    });
}
```

Рис. 3.1. Приклад коду з використанням rxJava

В даному коді використовується клас Observable - «спостережуваний», це об'єкт, за яким можна спостерігати, тому що він буде повідомляти своїх абонентів про будь-які зміни та події. З масиву names створюється «спостережуваний» об'єкт, тобто кожен рядок в names - це, по суті, подія.

Далі, з методом subscribe відбувається "підписка" на даний "спостережуваний" об'єкт. В якості спостерігача ми використовуємо клас з бібліотеки rxJava Action1 - він отримує якийсь значення (одне, тому що в імені класу цифра «один») і виконує з ним якась агресивна дія - в даному випадку, висновок на екран з форматуванням .

Але це найпростіший приклад. З інших можливостей, які надає rxJava і

«Реактивне» програмування в цілому:

- Оператори, які залежать від умови - оператори, які дозволяють змінювати спостерігаються потоки по певній умові. Це, наприклад, defaultIfEmpty (), skipWhile ().
- Оператори комбінування - вони об'єднують кілька спостережуваних потоків в один за допомогою певних правил. Наприклад, zip (), merge (), join ().
- Оператори фільтрування - оператори, які залишають тільки певні елементи з потоку, наприклад, filter (), skipLast (), firstOrDefault ().
- Оператори трансформації - перетворюють одні елементи в інші. це методи map (), flatMap (), switchMap ().

Всі ці можливості роблять бібліотеку rxJava неймовірно зручною для програмування з підтримкою потоків і різних джерел даних, що дуже важливо в даній роботі.

Ця бібліотека використовується для роботи з потоками. Важливо відзначити, що для роботи з інтернет-з'єднанням використовується інша бібліотека, rxJava можливостей таких не надає. Бібліотека Retrofit перетворює HTTP API в інтерфейси Java, спрощуючи процес роботи з одним API.

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}
```

Рис. 3.2. Код для створення інтерфейсу роботи з API в бібліотеці Retrofit

Це інтерфейс (рис. 3.2), який ми самі використовуємо в додатку. Він являє собою список методів, кожен з яких представляє собою запит до API. Методи можуть бути різними - GET, POST, PUT і т.д., все це прописується в анотації до методу, як і шлях запиту. За допомогою анотацій параметрів @Path, @Header можна додати в запит параметри шляху і заголовки запиту.

Імплементація інтерфейсу генерується бібліотекою, для генерації потрібно викликати метод create у класу Retrofit, створеного із зазначенням baseUrl - базового URL для API (рис. 3.3).

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com/")
    .build();

GitHubService service = retrofit.create(GitHubService.class);
```

Рис. 3.3. Код для генерації імпелементациі інтерфейсу в коді Retrofit

Комбінуючи Retrofit і RxJava можна побудувати додаток, яке дуже гнучко в плані отримання даних з різних джерел.

3.3 Взаємодія з віддаленим сервером (back-end)

Віддалений сервер використовується для отримання даних про зовнішній вигляд програми, про доступні в даному магазині функціях і дані про товари, категоріях, замовленнях, покупців і т. Д.

У кожен запит додається заголовок запиту ("Header"), в якому міститься інформація про ідентифікатор інтернет-магазину (ідентифікатор вбудовується в додаток при складанні). За допомогою такого заголовка віддалений сервер розуміє, дані якого інтернет-магазину потрібно повернути у відповіді.

Взаємодія з сервером back-end здійснюється за допомогою протоколу HTTP і GET і

POST запитів.

Список запитів наведено в розділі 2.6.

Взаємодія здійснюється за допомогою передачі даних в форматі JSON.

Дані зберігаються в кеші з метою подальшого використання та економії часу і трафіку мобільного пристрою.

3.4 Вибір використовуваних бібліотек

У реалізації вихідного Android-додатку використовуються чотири основні технології: мова програмування Java, бібліотека RxJava, бібліотека обробки HTTP-запитів Retrofit, база даних Realm і Dagger 2 для реалізації паттерна Ін'єкція Залежностей (Dependency Injection). Для кожного з обраних інструментів необхідно пояснити деякі конкретні моменти і пояснити їх вибір.

Всі залежності системи прописуються у файлі dependencies.gradle і підключаються окремо до кожного з трьох (Presentation, Data, Domain) модулів.

Мова програмування Java

Мова Java- основна мова програмування, що використовується для розробки проекту програми інтернет-магазину для ОС Android в цьому проекті. Java - основна мова, що використовується Google при розробці Android SDK, і переважна більшість проектів мобільних додатків написано саме на мові Java (в даний момент набирає популярність також мову Kotlin [15]).

RxJava

RxJava- це реактивне (для реактивного програмування) розширення для JVM (віртуальної машини Java), яке забезпечує доступ до низькорівневих потокам, синхронізації і безпеки потоків в проекті.

Retrofit

Retrofit- це бібліотека, яка використовується для HTTP-запитів і зручного перетворення форматованих рядків JSON (JavaScript Object Notation) в DTO (Data Transfer Object, Об'єкти передачі даних), що використовуються в додатку для роботи і зберігання даних в пам'яті.

База даних Realm

База даних Realm- це база даних NoSQL з відкритим вихідним кодом, розроблена для мобільних пристроїв. Вона підтримує двосторонню синхронізацію даних, швидкий доступ до даних, що зберігаються в пам'яті, і способи зв'язки з RxJava і мовою Java.

Dagger 2 і ін'єкція залежностей

Dagger 2 - повністю статична інфраструктура для створення залежностей під час компіляції для Java і Android. Тут варто ознайомитися з двома важливими концепціями -

«Ін'єкція залежностей» («Dependency Injection») і «інверсія контролю» («Inversion of Control»).

Суть інверсії контролю в тому, що, кажучи простими словами, жоден клас не повинен створювати екземпляри іншого класу, а повинен отримувати екземпляри з класу-конфігурації. Адже якщо java-клас створює екземпляр іншого класу за допомогою оператора new, він не може використовуватися і тестуватися незалежно від класу, в якому він створений, і таке називається «Жорсткої залежністю» («Hard Dependency»).

Ін'єкція залежностей вбудовується в концепцію інверсії управління. Ін'єкція залежностей - шаблон, при якому один об'єкт надає залежності іншого об'єкта. Залежність-об'єкт, який можна використовувати ("послуга"). Ін'єкція - передача залежності залежному об'єкту ("клієнту"), який буде його використовувати. "Послуга" стає частиною стану клієнта. Використовуючи ін'єкцію залежностей, ми можемо легко перевикористати об'єкти класів і тестувати класи в Java.

Dagger 2 - ще одна важлива бібліотека для Android-проектів, крім RxJava, тому що Dagger 2 надає можливість реалізувати патерн «ін'єкція залежностей» в Android-проектах.

Версія Android SDK і мінімально підтримувана версія Android OS

Оскільки Android - сильно фрагментована ОС з користувачами, які використовують різні версії Android, розробник повинен вибрати мінімальну версію SDK для програми. Мінімальний Android SDK, який використовується в цій системі - SDK 19. Додаток буде підтримувати всі версії операційної системи Android від Android 4.4 (під кодовою назвою «KitKat»). Згідно Android Dashboards, розподіл всіх нижчих версій ОС Android становить менше 10%.

3.5 Збірка програм для операційної системи Android за допомогою Gradle

Після отримання команди на збірку, складальник повинен виконати своє основне завдання - зібрати проект. Збирати залежності, завантажувати бібліотеки, генерувати підсумковий файл повинен складальник. Для збірки Android-проектів існує два загальновідомих збирача проектів - Gradle і Maven. [2]

Gradle - це система автоматизації побудови з відкритим вихідним кодом, яка ґрунтується на концепціях Apache Ant і Apache Maven, але використовує Groovy-орієнтований доменний мову (DSL) замість XML-форми, використовуваної Apache Maven для оголошення конфігурації проекту.

У таблиці 3.1 коротко порівняні ці системи.

Таблиця 3.1 Коротке порівняння Gradle і Maven

функція	Gradle 3.5	Maven 3.3.3
Інкрементальна компіляція Java	незалежно від того, змінюється вихідний код або шлях до класів, Gradle виявляє всі класи, на які вплинула зміна.	Не підтримується
об'єднання окремих збірок	Можна, можливо комбінувати окремі збірки для роботи з кодом в декількох репозиторіях.	Не підтримується
Конфігурація середовища збірки, контрольована версією	важливі параметри для настройки середовища збірки можуть бути збережені в версії як частина проекту. Розробникам не потрібно налаштовувати їх вручну.	Не підтримується

Якщо проводити повне і комплексне порівняння, перевага Gradle стає ще більш очевидним, але це порівняння не є частиною проекту, тому можна лише відзначити, що Gradle заслужено є стандартом для збірки Android додатків і в даному проекті використовується Gradle складальник.

Система Gradle дозволяє також оголошувати глобальні змінні, які пізніше будуть доступні в коді програми, тому в них зручно записувати таке незмінне значення, як ідентифікатор додатку:

defCUSTOMISABLE_APPLICATION_ID ="\ "replace\ ""

Оголосивши таку змінну в головному файлі збирача build.gradle, можна в процесі складання замінювати її на потрібне значення і покладатися на цю змінну.

Також система Gradle зручна запуском збірки з командного рядка без графічного інтерфейсу, що необхідно для проекту. У таблиці 3.2 наведені команди, які викликаються з папки проекту і їх призначення.

Таблиця 3.2. Короткий список команд для Gradle, що викликається з командного рядка

команда	**значення**
./gradlew tasks	Привести список доступних команд
./gradlew assembleRelease	Зібрати додаток в режимі "Release" для публікації і отримати APK файл
./gradlew assembleDebug	Зібрати додаток в режимі "Debug" для налагодження і отримати APK файл

Так, використовуючи глобальні змінні і команди збірки з командного Сторк, можна домогтися збірки мобільного додатка без графічного інтерфейсу і без середовища розробки на будь-якій системі (OS X, Linux, Windows). І отже, процес складання за допомогою Gradle просто автоматизувати і перенести на віддалену віртуальну машину, наприклад, обраний Digital Ocean с Linux Ubuntu.

Можливості кастомізації додатків під Android

Якщо створити віддалену віртуальну машину на сервісі Digital Ocean, і використовувати збирач Gradle, то можна буде збирати програми віддалено і з глобальними змінними, які можуть бути змінені прямо перед складанням.

Схема роботи системи така, що з бібліотеки модулів в складальник з усіх параметрів передається тільки ідентифікатор додатки в системі, і цей же ідентифікатор записується в файл build.gradle програми:

defCUSTOMISABLE_APPLICATION_ID ="Fds32k4"

В даному випадку ідентифікатором додатка буде"Fds32k4".

Далі в додатку можна використовувати даний ідентифікатор для кастомізації

(Зміни зовнішнього вигляду програми).

За ідентифікатором додатки робляться всі запити на сервер бібліотеки модулів, що розробляється Марікяном А. А. і за допомогою ідентифікатора виходять важливі параметри програми:

- Основний використовуваний колір додатки
- Другорядний колір додатки
- Мова додатка
- Чи включена оплата через додаток
- Використовуваний в додатку шрифт

Відбувається отримання цих параметрів відбувається наступним чином: в додаток через Gradle вбудований параметр ідентифікатор додатку, при першому запуску програма звертається до сервера і отримує дані про ці параметри. Пізніше ці дані зберігаються в пам'яті пристрою і запитуються повторно дуже рідко.

Це рішення дає необмежений контроль над додатком після публікації без повторної його завантаження в Google Play і зручніше в плані передачі параметрів при складанні - передається лише один параметр - ідентифікатор.

У додатку при відображенні екранів дана інформація запитується з пам'яті пристрою і не витрачає трафік пристрою на додаткові запити в мережу Інтернет.

Зовнішній вигляд програми при зміні параметрів

На рис. 3.3 зображений зовнішній вигляд головного вікна програми.

Кольори, використані в додатку можуть бути змінені за бажанням власника інтернет-магазину. Наприклад, верхній колір - це primaryColor, колір панелі з кнопками і фону товарів accentColor. Колір тексту - textColor, а загальний фоновий колір вікна - backgroundColor. Також кольору іконок бувають двох кольорів - чорного або білого, цей параметр також змінюємо і вибирається власником інтернет-магазину.

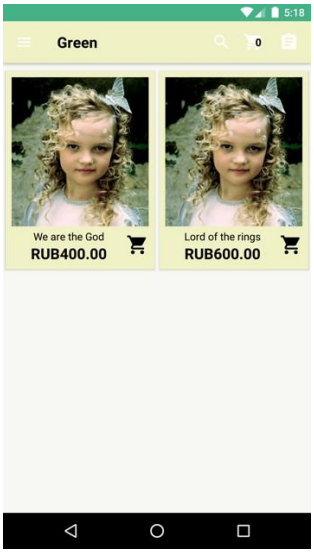


Рис. 3.3. Зовнішній вигляд програми

Інструменти розробки збирача

Як середовище розробки була використана Pycharm останньої на момент розробки версії. Ця IDE (integrated development environment - інтегроване середовище розробки), спеціально налаштована для розробки на мові Python. Pycharm Community Edition доступна безкоштовна кожному під ліцензією Apache License 2.0.

Середовище розробки Pycharm доступна на операційних системах Windows, Mac OS X,

Linux.

Перевага використання саме Pycharm складається в наявності в середовищі розробки аналізатора коду і підказок при написанні коду на мові Python, рефакторінга на мові Python, вбудованого відладчика Python і інших функцій.

Основна мова розробки - Python версії 3. Третя версія використовується для спрощення деяких операцій над файлами і багатозадачності.

4. НАПОВНЕННЯ ДОДАТКІВ ДАНИМИ (CMS)

4.1. Використання віддаленого сервера для збірки

У другому розділі обґрунтовано використання віддаленого сервера Digital Ocean. Далі будуть розглянуті кроки, необхідні для налаштування збирача.

Підключення до віддаленої віртуальній машині під пристроєм ОС Linux Ubuntu і управління нею здійснюється по протоколу SSH, настройка і експлуатація віртуальної машини буде описана далі, в розділі «2.1.2 Налаштування збирача». Щоб забезпечити доступ ззовні до системи, на віртуальній машині встановлено сервер nginx і встановлений фреймворк для веб-додатків Django, який дозволяє по HTTP-запиту до серверу розпочинати процес складання.

Для запуску процесів складання асинхронно і для того, щоб фреймворк Django працював постійно в тлі використовується чергу завдань Celery. Щоб відновити після завантаження використовується gUnicorn.

Також на сервері встановлена версія Java Development Kit [35] і Android SDK для збірки проекту програми операційної системи Android.

В результаті, після завершення розгортання і налаштування віддаленого збирача, отриманий IP-адреса сервера, а завдяки налагодженому сервера nginx і фреймворку Django [22] до сервера можна передавати запити виду:

http://188.226.131.144/?appId=x3sa&package=me.sgayazov&appName=MarketApp&force=true & Debug = false

Цей запит додає поточний запит на складання з переданими в запиті параметрами в чергу збірки.

А запит виду:

http://188.226.131.144/downloadapk/?appId=x3sa

Дозволяє отримати завантажити зібраний підсумковий APK-файл за ідентифікатором додатка.

За допомогою таких запитів відбувається управління запуском збірки і передача параметрів майбутнього програми. Ці запити в даній системі викликаються бібліотекою модулів, яка запитує додаток з певними параметрами. Після завершення збирання віддалений збирач повідомляє бібліотеку модулів і передає посилання на APK-файл.

4.2. Налаштування збирача

Нижче буде наведено кроки по налаштуванню щойно створеного "Droplet" в сервісі Digital Ocean (або ж будь-який інший віртуальної віддаленій машині під управлінням ОС Linux Ubuntu) до робочого стану збирача.

Після реєстрації в сервісі Digital Ocean (або в іншому сервісі), необхідно створити віртуальну машину, серед обов'язкових вимог наступні:

- Не менш 1GB оперативноїпам'яті для збірки Java / Android проекту через Gradle (при 512MB javac - Java Compiler, Компілятор мови Java - видає помилку "Unable to allocate memory" через нестачу оперативної пам'яті для збірки
- Операційна система Linux Ubuntu версії вище 16 і з розрядністю x64
- Не менш 20GB пам'яті для зберігання проекту, Android SDK і зібраних файлів Створивши віртуальну машину з такими настройками можна приступати до розгортання.

У більшості таких сервісів доступ до машини здійснюється за допомогою протоколу SSH.

Тут і далі будуть описані процедури доступу до віртуальної машини за допомогою ОС

Mac OS X і програми командного рядка Terminal.

Ввівши в Terminal команду і написавши пароль, виданий в Digital Ocean,

ssh root@188.226.131.144

де root - ім'я користувача, отримане при створенні віртуальної машини (найчастіше root), а 188.226.131.144 - адреса віртуальної машини, буде відкритий SSH доступ до машини. Після цієї команди всі дії будуть виконуватися в командному рядку віртуальній машині, про що говорить курсор введення команд, змінився на

root @ ubuntu-512mb-ams2-01 : ~ #

Що говорить про те, що ми під ім'ям користувача root підключені до машини з назвою ubuntu-512mb-ams2-01.

Установка останньої версії Android SDK і JDK з командного рядка

Для збірки Android додатків необхідна наявність Android SDK в системі, так само як і

Java Development Kit, так як вихідний код додатків Android написаний на мові Java.

Установка правильного (правильним є оригінальна JDK від Oracle, що не OpenJDK) виконується трохи складніше, ніж команда apt-get install, тому що пакет Oracle JDK знаходиться в іншому репозиторії і потрібно додати репозиторій:

sudo apt-get install python-software-properties

sudo add-apt-repository ppa: webupd8team / java sudo apt-get update

Після цього можна виконати команду установки:

sudo apt-get install oracle-java8-installer

Для установки потрібно завантажити Android SDK за допомогою завантажувача Linux Ubuntu wget і розпакувати файл:

wget http://dl.google.com/android/android-sdk_r23-linux.tgz tar -xzf android-sdk_r23-linux.tgz

Перейшовши в тільки що розпакувати папку, можна завантажити необхідні компоненти. За допомогою команди

android list sdk -all

Можна отримати список доступних для завантаження компонентів. Зазвичай, основними є Build Tools і Platform Tools останньої версії. Встановимо то, що потрібно зі списку командою

android update sdk -u -a -t 1,2,3,4, .., n

Після цього установка завершена. Необхідно оновлювати компоненти Android SDK на сервері в міру виходу нових компонентів.

4.3. Установка Django, створення віртуального оточення Python 3, настройка nginx і gunicorn

Потрібно почати з команд:

sudo apt-get update sudo apt-get upgrade

Ці команди оновлять і завантажать інформацію про пакетах. Після поновлення можна

встановити інтерпретатор мови Python 3, тому що і Django використовує цю мову, і сам скрипт написаний на цій мові. [21] Зробити це можна за допомогою команди:

sudo apt-get install python3

Установка nginx виконується також за допомогою менеджера пакетів:

sudo apt-get install nginx

Для зручності потрібно встановити менеджер Python пакетів третьої версії:

sudo apt-get install python3-setuptools sudo easy_install3 pip

Тепер можна створити віртуальне Python оточення:

pip3 install virtualenv virtualenv / opt / myenv

source / opt / myenv / bin / activate

Де myenv - назва оточення, може бути будь-яким. Перша команда встановлює пакет Python для створення віртуальних оточень, друга команда створює таке оточення по шляху / opt / myenv, а третя переходить в це віртуальне оточення.

Усередині віртуального оточення потрібно встановити Django і gunicorn за допомогою менеджера пакетів Python "pip":

pip install django gunicorn

Після цього можна вважативсі установки завершеними і можна приступити до налаштування. Для створення проекту в віртуальному оточенні слід виконати команду:

django-admin.py startproject myproject

demyproject - назва проекту.

Для настройки проекту Django, потрібно відкрити папку myproject і командою

nano settings.py

відкрити для редагування файл настройок.

У файлі налаштувань прописати наступні рядки:

STATIC_ROOT = '/ opt / myenv / myproject / static /'

І виконати команду:

python manage.py collectstatic

Для настройки nginx потрібно перейти по шляху:

cd / etc / nginx / sites-available /

і відкрити для редагування файл default:

nano default

в якому прописати конфігурацію, змінивши шлях до проекту і IP-адреса сервера.

```
server {
listen 80;
server_name 111.222.333.44; # IP-адреса сервера
```

```
access_log /var/log/nginx/example.log;


location / static / {
root / opt / myenv / myproject /; # Шлях до проекту
expires 30d;
}


location / {
proxy_pass http://127.0.0.1:8000; proxy_set_header Host $ server_name; proxy_set_header X-Real-IP $ remote_addr;
proxy_set_header X-Forwarded-For $ proxy_add_x_forwarded_for;
}
}
```

Схема 1. Файл конфігурації Django

Після цього потрібно перезапустити nginx командою

```
sudo service nginx restart
```

і запустити gunicorn:

```
gunicorn myproject.wsgi: application
```

Тепер по IP-адресою сервера доступна сторінка Django. Використовуючи Django, можна змінити те, що відбувається при зверненні до сервера. Це розписано в наступному розділі.

Написання програм для Django

В папці проекту Django потрібно створити сторінку [12], яка буде обробляти запити командою

python manage.py startapp myapp

Після цього структура файлів буде виглядати як на схемі 1.

```
├── django_project
│       ├── __init__.py
│       ├── settings.py
│       ├── urls.py
│       └── wsgi.py
```

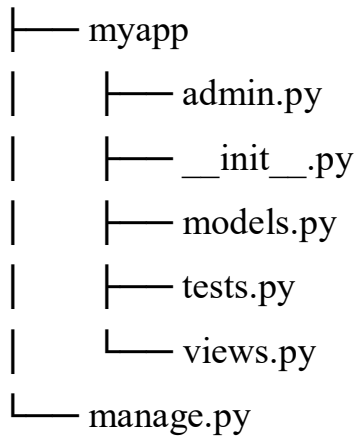


Схема 2. Структура файлів проекту Django

Після цього можна редагувати список обслуговуваних адрес у файлі urls.py, і редагувати поведінку myapp в файлі views.py

Після таких змін потрібно перезапускати gunicorn командою

service gunicorn restart

4.4. Асинхронність і Celery

В Django ми можемо прописувати то, як відображається сторінка і що повертається в запиті, але під час запиту можна збирати додаток. Збірку слід здійснювати в тлі, і для цього використовується Celery [18]. Перед початком в віртуальному оточенні так само як і раніше потрібно встановити пакет Python:

pip install celery

Також потрібно встановитиRedis, тому що Celery використовує «брокерів» для передачі повідомлень між проектом Django і процесами Celery. Тут використовується Redis в якості брокера повідомлень. У тих же налаштуваннях settings.py слід додати рядки для настройки Redis:

CELERY STUFF

BROKER_URL = 'redis: // localhost: 6379' CELERY_RESULT_BACKEND = 'redis: // localhost: 6379' CELERY_ACCEPT_CONTENT = ['application / json'] CELERY_TASK_SERIALIZER = 'json' CELERY_RESULT_SERIALIZER = 'json' CELERY_TIMEZONE = Moscow '

Після цього в коді Django в views.py можна позначати методи декоратором

@task (name="Build_task")

А завдання тепер будуть виконуватися у фоновому режимі, тобто запит не чекатиме закінчення зборки, а просто почне збірку і поверне відповідь про успішний старт зборки.

Збільшення обсягу оперативної пам'яті за допомогою swap-файлу

У зв'язку з тим, що збірка складних проектів займає багато оперативної пам'яті (512 МБ було недостатньо для збірки простого проекту), можна додатково використовувати файл підкачки, для того щоб використовувати пам'ять на жорсткому диску в якості оперативної. На щастя, на Digital Ocean машинах встановлена SSD пам'ять, яка швидше в рази звичайних жорстких дисків. [17]

Зробити це можна командою

sudo fallocate -l 4G / swapfile

Ця команда виділить4 ГБ пам'яті на жорсткому диску під файл підкачки. Для використання цього місця потрібно встановити правильні права:

sudo chmod 600 / swapfile

І використовувати місце:

sudo swapon / swapfile

Щоб файл підкачки не зникав після перезавантаження, потрібно відкрити файл

sudo nano / etc / fstab

І прописати рядок

/ swapfile none swap sw 0 0

Після цього в системі буде використовуватися файл підкачки з необхідним обсягом пам'яті. Після завантаження правильного файлу views.py настройка і установка збирача завершена.

Запуск збірки можливий за допомогою запиту:

http://188.226.131.144/?appId=x3sa&package=me.sgayazov&appName=MarketApp&fo rce = true & debug = false

де потрібно поміняти параметри:

appId- внутрішній ідентифікатор додатки, використовується для отримання з сервера даних програми;

package- ім'я пакета Android;

appName- назва програми в системі Android;

ВИСНОВОК

Очікується, що система стане зручним інструментом для власників інтернет-магазинів і буде вільно доступна. Система може використовуватися без знання мов програмування для створення мобільних додатків інтернет-магазинів. Система розробляється як масштабується, що розширюється і високопродуктивна.

З виконаних завдань варто відзначити вивчення поширених способів розробити додаток таким чином, що воно буде легко змінним під різні інтернет-магазини. В рамках даного завдання було визначено, що поширених способів не існує, і був обраний спосіб з установкою ідентифікатора додатка при складанні, який використовується для отримання даних про зовнішній вигляд програми з віддаленого сервера згодом під час запуску програми. Також був проведений аналіз підходів до розробки додатків під операційну систему Android і визначено набір бібліотек та інструментів, а також мова програмування

— Java версії 7, останньої версії мови, підтримувана Android SDK.

Розглянуто підходи збірки додатку через інтерфейс командного рядка в APK файл, готовий до поширення через магазин додатків Google Play, а також підходи розміщення збирача в хмарної віртуальній машині в стійкою до збоїв середовищі. Для складання використовується складальник Gradle, проект завантажується для збірки віддалено з системи GitHub і файл APK після складання зберігається в системі збирача і стає доступний за посиланням. Складальник знаходиться в хмарі на серверах Digital Ocean під управлінням ОС Linux, налаштований перезапуск збирача після перезавантаження / падіння системи.

Розроблено структуру взаємодії мобільного додатка з даними інтернет-магазину (продукти, категорії і т.д.), архітектура мобільного додатка. Обрана архітектура Clean Architecture і патерн Model-View-Presenter.

Розроблене рішення складається з збирача і проекту мобільного додатка. Складальник розташований на віддаленому сервері і за запитом з зовнішньої системи збирає проект програми з заданим ідентифікатором. Зібране мобільний додаток є мобільним додатком для клієнтів інтернет-магазину, підтримує відображення асортименту, категорій інтернет-магазину, дозволяє створювати і оплачувати замовлення. Додаток отримує дані про конкретному магазині з зовнішнього сервера, при запитах до якого передає ідентифікатор додатки для ідентифікації.

Проект дозволяє збирати кастомізовані мобільні додатки для інтернет- магазинів. Проект може використовуватися власниками інтернет-магазинів для розширення свого бізнесу за допомогою створення мобільного застосування.

Відповідно до обраних технологіями була розроблена програма, проведені її тестування і налагодження. Також була розроблена технічна документація.

Однак система створює тільки додатки для ОС Android і не підтримує iOS. Зробити цю систему доступною для користувачів iOS - одне з подальших напрямків роботи.

Можливе використання зовнішніх модулів також є перспективним напрямком розробки. Модулі - це адаптери, «view», аналітичні модулі, розроблені сторонніми розробниками. Користувач зможе завантажувати ці модулі як архівний файл, і система буде використовувати їх для розширення функцій мобільного додатка.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Intellij IDEA - JetBrains [Електронний ресурс] // URL: https: //www.jetbrains.com/idea/(Дата звернення: 12.04.2017, режим доступу: вільне володіння).
2. Maven - Welcome to Apache Maven [Електронний ресурс] // URL: https://maven.apache.org/ (Дата звернення: 11.03.2017, режим доступу: вільне володіння).
3. Патерни для новачків: MVC vs MVP vs MVVM [Електронний ресурс] // URL: https://habrahabr.ru/post/215605/ (Дата звернення: 22.03.2017, режим доступу: вільне володіння).
4. Введення в JSON [Електронний ресурс] // URL:http://www.json.org/json-ru.html (Дата звернення: 21.03.2017, режим доступу: вільне володіння).
5. Safely Storing User Passwords: Hashing vs. Encrypting [Електронний ресурс] // URL:http://www.darkreading.com/safely-storing-user-passwords-hashing-vs-encrypting/a/d- id / 1269374 (Дата звернення: 21.04.2017, режим доступу: вільне володіння).
6. Android Application Framework [Електронний ресурс]. / режим доступу:https://developer.android.com/guide/index.html, вільний. (Дата звернення: 10.05.16)
7. Фрісен, Д. Learn Java for Android development // Джефф Фрісен - Apress, 2014.
8. Коуco, К. Gradle for Android // Кеннет Коуco - O'Reilly Media, 2016.
9. Матлофф, Н. The Art of Programming: Tour of Statistical Software Design // Норман Матлофф
- No Starch Press, 2011 року.

10. Шоттс, У. The Linux Command Line. A Complete Introduction // Вільям Шоттс - No Starch Press, 2012.
11. Хайд, P. With great code: thinking low-level, writing high-level // Рендалл Хайд - No Starch Press, 2004.
12. Create reactive mobile apps in a fraction of the time [Електронний ресурс] // URL: https://realm.io/docs/ (Дата звернення: 05.04.2017, режим доступу: вільне володіння).
13. Android Developers [Електронний ресурс] // URL:http://developer.android.com/index.html (Дата звернення: 11.02.2017, режим доступу: вільне володіння).
14. RxJava - Reactive Extensions for the JVM - a library for composing asynchronous and event- based programs using observable sequences for the Java VM. [Електронний ресурс] // URL: https://github.com/ReactiveX/RxJava (Дата звернення: 01.03.2017, режим доступу: вільне володіння).
15. Kotlin Programming Language. [Електронний ресурс] // URL: https://kotlinlang.org (Дата звернення: 05.04.2017, режим доступу: вільне володіння).
16. MVP for Android: how to organize the presentation layer. [Електронний ресурс] // URL:http://antonioleiva.com/mvp-android/ (Дата звернення: 02.12.2016, режим доступу: вільне володіння).
17. How To Add Swap on Ubuntu 14.04. [Електронний ресурс] // URL: https: //www.digitalocean.com/community/tutorials/how-to-add-swap-on-ubuntu-14-04 (Дата звернення: 05.03.2017, режим доступу: вільне володіння).

18. Asynchronous Tasks With Django and Celery. [Електронний ресурс] // URL: https://realpython.com/blog/python/asynchronous-tasks-with-django-and-celery/ (Датазвернення: 25.02.2017, режим доступу: вільне володіння).
19. django-background-tasks 1.1.9. [Електронний ресурс] // URL:https://pypi.python.org/pypi/django-background-tasks (Дата звернення: 12.01.2017, режим доступу: вільне володіння).
20. How To Use the Django One-Click InstallImage. [Електронний ресурс] // URL:https: //www.digitalocean.com/community/tutorials/how-to-use-the-django-one-click-install- image (Дата звернення: 02.12.2016, режим доступу: вільне володіння).
21. How do I install Python 3.6 using apt-get ?. [Електронний ресурс] // URL: https://askubuntu.com/questions/865554/how-do-i-install-python-3-6-using-apt-get (Датазвернення: 01.02.2017, режим доступу: вільне володіння).
22. How To Set Up Django with Postgres, Nginx, and Gunicorn on Ubuntu 16.04. [Електронний ресурс] // URL: https: //www.digitalocean.com/community/tutorials/how-to-set-up-django-with- postgres-nginx-and-gunicorn-on-ubuntu-16-04 (Дата звернення: 11.03.2017, режим доступу: вільне володіння).
23. Django + Python3 + Nginx + Gunicorn + DO. [Електронний ресурс] // URL: https://djbook.ru/examples/62/ (Дата звернення: 25.01.2017, режим доступу: вільне володіння).
24. Architecting Android ... The clean way? [Електронний ресурс] // URL: https://fernandocejas.com/2014/09/03/architecting-android-the-clean-way/ (Дата звернення: 26.02.2017, режим доступу: вільне володіння).

25. Architecting Android ... The evolution. [Електронний ресурс] // URL: https://fernandocejas.com/2015/07/18/architecting-android-the-evolution/ (Дата звернення: 14.01.2017, режим доступу: вільне володіння).
26. Clean Architecture: Dynamic Parameters in Use Cases. [Електронний ресурс] // URL: https://fernandocejas.com/2016/12/24/clean-architecture-dynamic-parameters-in-use-cases/ (Дата звернення: 30.12.2016, режим доступу: вільне володіння).
27. Walmart. Google Play Market [Електронний ресурс] // URL: https://play.google.com/store/apps/details?id=com.walmart.android&hl=ru (Дата звернення: 15.10.2016, режим доступу: вільне володіння).
28. Розробка інтернет-магазинів в Росії набирає обертів. [Електронний ресурс] // URL: https://rg.ru/2015/09/22/magaziny.html (Дата звернення: 15.10.2016, режим доступу: вільне володіння).
29. OZON.ru - інтернет магазин. Google Play Market. [Електронний ресурс] // URL: https://play.google.com/store/apps/details?id=ru.ozon.app.android&hl=ru (Дата звернення: 15.10.2016, режим доступу: вільне володіння).
30. The Rise of M-Commerce: Mobile Shopping Stats & Trends. [Електронний ресурс] // URL:http://www.businessinsider.com/mobile-commerce-shopping-trends-stats-2016-10 (Датазвернення: 12.11.2016, режим доступу: вільне володіння).

31. Mobile Commerce Statistics and Trends [Infographic]. [Електронний ресурс] // URL:http://www.invespcro.com/blog/mobile-commerce/ (Дата звернення: 25.11.2016, режим доступу: вільне володіння).
32. Android Pay. [Електронний ресурс] // URL: https: //www.android.com/pay/ (Дата звернення: 12.01.2017, режим доступу: вільне володіння).
33. Amazon Web Services (AWS). [Електронний ресурс] // URL: https://aws.amazon.com/ru/ (Дата звернення: 13.01.2017, режим доступу: вільне володіння).
34. Digital Ocean. Cloud computing, designed for developers. [Електронний ресурс] // URL: https: //www.digitalocean.com/(Дата звернення: 13.02.2017, режим доступу: вільне володіння).
35. Java SE Development Kit 8 Downloads. [Електронний ресурс] // URL:http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html (Дата звернення: 14.03.2017, режим доступу: вільне володіння).
36. How to install Android SDK Build Tools on the command line? [Електронний ресурс] // URL: https://stackoverflow.com/questions/17963508/how-to-install-android-sdk-build-tools-on-the- command-line (Дата звернення: 13.03.2017, режим доступу: вільне володіння).
37. Model-View-Controller [Електронний ресурс] // URL: https: //www.wikiwand.com/ru/Model- View-Controller (Дата звернення: 11.02.2017, режим доступу: вільне володіння).
38. Model-View-ViewModel [Електронний ресурс] // URL: https: //www.wikiwand.com/ru/Model- View-ViewModel (Дата звернення: 11.02.2017, режим доступу: вільне володіння) .Java SE Development Kit 8 Downloads. [Електронний ресурс] // URL:http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html (Дата звернення: 14.03.2017, режим доступу: вільне володіння).
39. How to install Android SDK Build Tools on the command line? [Електронний ресурс] // URL: https://stackoverflow.com/questions/17963508/how-to-install-

android-sdk-build-tools-on-the- command-line (Дата звернення: 13.03.2017, режим доступу: вільне володіння).

40. 5 Reasons Why You ShouldChoose Realm Over CoreData / SQLite [Електронний ресурс] // URL: https://sebastiandobrincu.com/blog/5-reasons-why-you-should-choose-realm-over-coredata (Дата звернення: 12.05.2017, режим доступу: вільне володіння).