ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

Сейітмат Мұхаммед-Сұлтан

«Жеке тұлғалардың қызмет көрсетуіне арналған Андроид жүйесінде қолданба әзірлеу»

Дипломдық жобаға
ТҮСІНІКТЕМЕЛІК ЖАЗБА

5B070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандығы

Алматы 2021

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
ПИ кафедрасының меңгерушісі,
PhD докторы
__________ М. Тұрдалыұлы
«06» ____06____ 2021 ж.

Дипломдық жобаға
ТҮСІНІКТЕМЕЛІК ЖАЗБА

Тақырыбы: Жеке тұлғалардың қызмет көрсетуіне арналған Андроид жүйесінде қолданба әзірлеу

5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандығы

Орындаған Сейітмат Мұхаммед-Сұлтан

Ғылыми жетекші
__________ А. Қайрбеков.

«04» мамыр 2021ж.

Алматы 2021

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

БЕКІТЕМІН
ПИ кафедрасының меңгерушісі,
PhD докторы
_____________ М. Тұрдалыұлы
«06» 06 2021 ж.

ТАПСЫРМА

Білім алушы: Сейітмат Мұхаммед-Сұлтан
Тақырыбы: Жеке тұлғалардың қызмет көрсетуіне арналған Андроид жүйесінде қолданба әзірлеу
Университет ректоры бұйрығының №2131-б «24» қараша 2021ж. Шешімімен бекітілген.
Аяқталған жұмысты тапсыру мерзімі: 2021 жылғы "07" маусым
Дипломдық жұмыстың бастапқы берілістері: *Ұсынылатын дипломдық жобада* жеке тұлғалардың қызмет көрсетуіне арналған Андроид жүйесінде қолданба әзірлеу.
Дипломдық жұмыста қарастырылатын мәселелер тізімі:
а) зерттеу бөлімі
б)технологиялық бөлімі
в)жобалау бөлімі
г) тәжірбиелік бөлімі
Сызба материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс)
Сызба материалдарының 15 бет слайдта көрсетілген.
Ұсынылатын негізгі әдебиет 8 атаудан тұрады.

Дипломдық жұмысты (жобаны) дайындау
КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекші мен кеңесшілерге көрсету мерзімдері	Ескерту
Дипломдық жоба тақырыбына байланысты ізденіс жасау, зерттеу жүргізу.	20.01.2021	
Дипломдық жұмысты жобалау: мобильді қосымшаның UI интерфейсін құру	01.02.2021	
Дипломдық жобаның мобильді қосымшасының функционалдарын дайындау, іске асыру.	05.02.2021	
Дипломдық жобаны диплом алды есеп алу комиссия мүшелеріне таныстыру.	10.03.2021	
Дипломдық жобаға түсіндірме жазба жазу.	06.05.2021	

Дипломдық жұмыс (жоба) бөлімдерінің кеңесшілері мен норма бақылаушының аяқталған жұмысқа (жобаға) қойған қолтаңбалары

Бөлімдер атауы	**Кеңесшілер, аты, әкесінің аты, тегі (ғылыми дәрежесі, атағы)**	**Қол қойылған күні**	**Қолы**
Бағдарламалық бөлім	Рамазан Акерке Техн.ғыл. магистрі, ассистент	31.05.2021	
Норма бақылау	Марғұлан Қ. Техн.ғыл. магистрі, лектор	04.06.2021	

Ғылыми жетекші________________________________А. Қайрбеков.
Тапсырманы орындауға алған білім алушы____________________М. Сейітмат.

Күні «04» маусым 2021ж.

АҢДАТПА

Бұл дипломдық жоба Android операциялық жүйесінде қауіпсіз және оңай жалға алу мен жалға беру қызметтерін жүзеге асыратын мобильді қосымша әзірлеуге арналған. Жоба тұтынушы тұтынушыға принципіне негізделген. Өзара қарым-қатынастың қауіпсіз болуына үшінші тарап кепілгер ретінде қатысып, және тұтынушы қалаған жағдайда жеткізу қызметін өзінің жауапкершілігіне алады.

Дипломдық жоба бірнеше логикалық бөлімнен тұрады.

Бірінші бөлімде жобаның өзектілігі, мақсаты қарастырылатын болады. Екінші бөлімде жобаны жүзеге асыру үшін қолданылған технологиялар мен кітапхана, фреймворктар сипатталған. Үшінші бөлімде бағдарламаның архитектурасы мен жобалау шаблонын жазылған. Соңғы бөлім қосымшаның функционалдарын сипаттайды

АННОТАЦИЯ

Этот дипломный проект посвящен разработке мобильного приложения, которое предоставляет безопасные и простые услуги аренды и проката на операционной системе Android. Проект основан на принципе потребителя к потребителю. В качестве гаранта безопасности взаимоотношений выступает третья сторона и по желанию потребителя берет на себя ответственность за службу доставки.

Дипломный проект состоит из нескольких логических разделов.

В первой части будут рассмотрены актуальность, цель проекта. Во второй части описаны технологии и библиотека, фреймворки, которые использовались для реализации проекта. В третьей части изложена архитектура и шаблон проектирования программы. Последний раздел описывает функциональные возможности приложения

ANNOTATION

This diploma project is dedicated to the development of a mobile application that provides secure and easy rental and rental services on the Android operating system. The project is based on the principle of"from consumer to consumer". The third party acts as a guarantor of the security of the relationship and, at the request of the consumer, assumes responsibility for the delivery service. The diploma project consists of three main sections.

The diploma project consists of several logical sections.

In the first part, the relevance and purpose of the project will be considered. The second section describes the technologies and Library, Frameworks used for the implementation of the project. The third section describes the architecture and design template of the program. The last section describes the functionality of the application

МАЗМҰНЫ

	Кіріспе	9
1	Жалпы сипаттама	10
1.1	Жүйе әзірлеудің мақсаты	10
1.2	Анықтамалар, терминдер, қысқартулар	10
1.3	Жобаның өзектілігі	11
2	Технологиялық бөлім	12
2.1	Бағдарламалау тілі	12
2.2	Android SDK	13
2.3	IDE	13
2.4	Құрастыру жүйесі	14
2.4.1	Gradle	14
2.4.2	Maven	14
2.4.3	Gradle vs Maven	16
2.5	Android UI	16
2.6	Android Jetpack	16
2.6.1	Android KTX	17
2.6.2	ViewModel	18
2.6.3	LiveData	19
2.6.4	Navigation	20
2.6.5	Retrofit 2	20
2.7	Firebase Cloud хабар алмасу	21
3	Жобалау бөлімі	22
3.1	MVVM – қосымша архитектурасының жобалау үлгісі	22
3.2	UI Архитектурасы	23
3.3	Клиент-серверлік қосымша архитектурасы	23
3.4	Прецеденттер диаграммасы	24
3.5	Белсенділік диаграммасы	25
4	Тәжірибелік бөлім	26
4.1	Қолданушы интерфейсі	26
4.1.1	Авторизация	26
4.1.2	Басты бет	27
4.1.3	Тауарлар тізімі	28
4.1.4	Тауар сипаттамасы	29
4.1.5	Жалға алу	30
4.1.6	Хабарлама	31
4.1.7	Профиль	32
4.1.8	Жалға беретін тауарлар	33
	Қорытынды	34
	Пайдаланылған әдебиеттер тізімі	35
	А Қосымшасы. Техникалық тапсырма	36
	Б Қосымшасы. Бағдарлама мәтіні	38

КІРІСПЕ

Қазіргі таңда заманауи цифрлық технология адам өміріне және әртүрлі салалар мен өндірістерге кеңінен ену барысында. Цифрлық технология дәстүрлі қызмет көрсету процесстеріне қарағанда өзінің тиімділігі жағынан басым түскендіктен өнеркәсіп, бизнес және басқа да салалар дәстүрліден цифрлы технологияға көшуде.

Негізгі мақсат жалға беру және жалға алу қызметін цифрландыру үшін электронды алаң жасалып оның қауіпсіздігін және тиімділігін қамтамасыз ету. Электронды алаң арқылы жалға алатын тауарлар бір жерге шоғырланып іздеу барысын жеңілдетеді.

Қызмет көрсету барысында үш тарап қатысатын болады. Олар: тұтынушы(жалға беруші), тұтынушы(жалға алушы), қызмет көрсетуші. Үшінші қызмет көрсетуші тараптың қатысуы арқылы жеткізу қызметі қауіпсіз және сенімді болады.

Бұл жобаның толықтай жүзеге асып, жоғарыдағы критреийлерге сай болу үшін әр тарапқа жеке бағдарлама жасалынады. Тұтынушыға: Android және IOS операциялық жүйелерінде жазылған мобильді қосымша. Қызмет көрсетуші тарапқа: админ панелі жасалынатын болады. Бұл дипломдық жоба Android операциялық жүйесіне қосымша жасалуына арналған.

1 Жалпы сипаттама

1.1 Жүйе әзірлеудің мақсаты

Бұл дипломдық жобаның негізгі мақсаты жалға алу мен жалға беру қызметін цифрландыру мақсатында электронды алаң жасау. Жобада қызметтер заңды түрде келісімшарт арқылы орындалуының негізінде тұтынушыларға электронды алаңды сенімді ету.

Негізгі міндеттері:

– авторизация және тіркеле алу;
– тауарды жалға алу немесе жалға беру;
– осы қызметтегі шоғырланған тауарларды жүйелі түрде көру ;
– кілт сөз арқылы жалпылама іздеу;
– өз тауарын сала алу мүмкіндігі;
– тауарды алу не жеткізу қызметінде таңдау жасай алу.

1.2 Анықтамалар, терминдер, қысқартулар

Мобильді қсымшаный әзірлеу барысында кездескен қысқартулар мен термин сөздер төмендегі 1-кестеде көрсетілген

1 - кесте. Анықтамалар, қысқартулар және терминдер

ОЖ	Операциялық жүйе
SDK	Software Development Kit
API	Application Programming Interface
UI	User Interface
JVM	Java Virtual Machine
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
XML	eXtensible Markup Language
MVVM	Model – View – ViewModel
KTX	Kotlin Extensions
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
FCM	Firebase Cloud Messaging
APK	Andorid Packege

1.3 Жобаның өзектілігі

Жобаның негізі қызметі жалға алу мен жалға беру процесстерін цифрландыру. Тұрмыста әр түрлі жағдайларға байланысты белгілі бір заттардың бір реттілік қажеттілігі туындап жатады. Және белігі бір затты сатып алмас бұрын өзіне керекті мәселені шешеді ме, шешпейді ме соны байқап көру үшін затты қолданып көру мұқтаждылығы пайда болады. Сол секілді тұрмыста көптеген оқиғалар болып, осындай кезде затты уақытша қолданысқа тексеріп көру қажеттілігі туындамақ.

Жалға беру қызметі бұл кей адамдардың негізгі табыс көзі. Кей адамдардың шағын бизнесіне айналған.

Түйіндей келе екі жақта да қажеттілік бар екенін көреміз. Алайда адамдарға осы қызметтке арналған электронды алаң жоқ. Еліміздегі нарықта бірнеше лидер компаниялар бар. Ол компаниялардың продукттарының негізгі бағыты сату қызметіне арналған. Ал жалға алу мен жалға беру қызметтері қосымша бөлік ретінде ғана жүреді. Тіпті кейбір электронды алаңдарда адамдар өздері сату алаңына жалға беретін тауарларды салып, қолмен қызмет түрлерін жазуға мәжбүр болып жатыр.

Біздің жобада тек бір ғана қызметке арналған электронды алаң жасалып, қосымша, үрдісті жеңілдетеін функционалдарды қамтамасыз ететін болады.

2 Технологиялық бөлім

2.1 Бағдарламалау тілі

Андроид ОС ресми түрде екі бағдарламалау тілін қолдайды:

– Java;
– Kotlin.

Алайда I/O 2019 конференциясында Google Kotlin бағдарламалау тілін Android әзірлеушілеріне қолдаунын тиімділігін хабарлады. Оған себеп жаңа Api және Jetpack кітапханалары бірінші Kotlin тілінде шығуы.

Kotlin – бұл әмбебап, еркін, ашық, статикалық түрде терілген "прагматикалық" бағдарламалау тілі, бастапқыда JVM (Java виртуалды машинасы) және Android үшін жасалған, ол объектіге бағытталған және функционалды бағдарламалау функцияларын біріктіреді. Ол үйлесімділікке, қауіпсіздікке, айқындыққа және құралдарды қолдауға бағытталған. Kotlin тілінде жазғылған код қысқа әрі түсінікті және типті қауіпсіз.

Kotlin 2010 жылы IntelliJ IDEA артындағы JetBrains компаниясында пайда болды және 2012 жылдан бастап ашық болып табылады. Қазіргі уақытта Котлин командасының JetBrains-тен 90-нан астам мүшесі бар, ал Github-тегі Котлин жобасының 300-ден астам мүшесі бар. JetBrains Котлинді көптеген өнімдерінде, соның ішінде IntelliJ-тің жетекші идеясын қолданады.

2.2 Android SDK

Android SDK – Android операциялық жүйесіне арналған мобильді қосымшаларды әзірлеудің әмбебап құралы. Кодтарды жазуға арналған қарапайым редакторлардан ерекшелігі-бастапқы кодтарды тестілеу мен күйін келтіруді бастауға, қосымшаның жұмысын Android ОЖ-нің әртүрлі нұсқаларымен үйлесімділік режимінде бағалауға және нәтижені нақты уақытта (міндетті емес) бақылауға мүмкіндік беретін кең функционалдылықтың болуы. Ол көптеген мобильді құрылғыларды қолдайды, олардың ішінде: ұялы телефондар, планшеттік компьютерлер, ақылды көзілдіріктер (соның ішінде Google Glass), Android ОЖ-де борттық компьютерлері бар заманауи автомобильдер, кеңейтілген функционалдығы бар теледидарлар, қол сағаттарының ерекше түрлері және басқа да көптеген мобильді гаджеттер, жалпы техникалық құрылғылар.[1]

Android SDK 2009 жылдың қазан айында шығарылды. Әзірлеу ортасы кросс-платформа болып табылады, жиынтықтың едәуір бөлігі Java бағдарламалау тілінде жазылған. SDK құрамына әр түрлі даму құралдары кіреді, соның ішінде түзеткіш, кітапханалар жиынтығы, QEMU қозғалтқышына негізделген телефон эмуляторы, құжаттама жиынтығы, қосымшалар мен

оқулықтардың мысалдары (нұсқаулықтар). Android SDK ортасын Linux, Mac OS X 10.5.8 және одан жаңа, Windows 7 және одан жаңа компьютерлерде іске қосуға болады. Алайда, 2015 жылғы наурыздағы жағдай бойынша SDK жүйесін Android ОЖ басқаратын құрылғыларда тікелей іске қосу мүмкін емес.

2.3 IDE

Android Studio – Android платформасындағы ресми интеграцияланған әзірлеу ортасы. Бұл IntelliJ Idea интеграцияланған әзірлеу ортасы негізінде жазылып, Google компаниясының қолдауында. IntelliJ IDEA бағдарламалық жасақтамасына негізделген. IntelliJ IDEA ұсынған редактор коды мен әзірлеуші құралдарынан бөлек Android Studio бағдарлама жасау өнімділігін арттыру үшін төмендегідей функцияларды ұсынады:

- Gradle негізінде икемді құрастырау жүйесі;
- жылдам әрі көп функционалды эмулятор;
- барлық android құрылғыларына бағдарлама жасауға арналған бір орта;
- визуалды бағдарламалау;
- apk анализаторы;
- нақты уақыттағы профильдеуші.

2.4 Құрастыру жүйесі

Android build жүйесі қолданба ресурстарын және бастапқы кодты жинақтайды және оларды APK-ға жинайды, оны сіз тексере, орналастыра, қол қоя және тарата аласыз. Құрылым жасау процесін автоматтандыру және басқару үшін Android Studio бағдарламасы Gradle-ді қолданады, сонымен қатар икемді тұтынушылық құрастыру конфигурацияларын анықтауға мүмкіндік береді. Әрбір құрастыру конфигурациясы сіздің кодыңыздың және ресурстарыңыздың жиынтығын анықтай алады, сонымен бірге сіздің қосымшаңыздың барлық нұсқаларына ортақ бөліктерді қайта қолдана аласыз. Gradle-ге арналған Android плагині Android қосымшаларын құру мен тексеруге тән процестер мен конфигурацияланатын параметрлерді қамтамасыз ету үшін құрастыру құралдар жинағымен жұмыс істейді.

Gradle және Android плагині Android Studio-дан тәуелсіз жұмыс істейді. Бұл дегеніміз, сіз өзіңіздің Android бағдарламаларыңызды Android Studio ішінен, пәрмен жолынан немесе Android Studio орнатылмаған машиналардан (мысалы, үздіксіз интеграциялық серверлерден) құра аласыз. Егер сіз Android Studio қолданбасаңыз, сіз өзіңіздің бағдарламаңызды қалай құруға және басқаруға болатындығын командалық жолдан біле аласыз. Жобаны пәрмен жолынан, қашықтағы құрылғыдан немесе Android Studio қолданбасын құрғаннан кейін де нәтиже бірдей болады.

2.4.1 Gradle

Gradle – бұл толықтай ашық қайнар көзі болып табылатын және Apache Maven мен Apache Ant-та көрген тұжырымдамаларды қолданатын автоматтандыру жүйесі. Ол Groovy бағдарламалау тіліне негізделген Apache Maven-тен ерекшеленетін доменге тән тілді қолданады, ол жобаның конфигурациясы үшін XML пайдаланады. Ол сонымен қатар бағытталған ациклдік графиктің көмегімен орындалатын тапсырмалардың ретін анықтайды. Бірнеше жасаушылар Gradle-ді құрып, алғаш рет 2007 жылы шығарды, ал 2013 жылы Google оны Android жобаларына арналған жүйе ретінде қабылдады. Ол өте үлкен болады деп күтіліп отырған көп жобалы жобаларды қолдауға арналған. Сонымен қатар, ол сіздің құрылысыңызға біртіндеп қосуға мүмкіндік береді, өйткені ол сіздің жобаңыздың қай бөліктері жаңартылатынын біледі. Жаңартылған бөліктерге тәуелді тапсырмалар енді қайта орындалмайды. Қазіргі уақытта, ең соңғы тұрақты шығарылым - 3.4 нұсқасы, ол 2017 жылдың ақпан айында іске қосылды. Ол Java, Scala және Groovy бағдарламаларын қолдана отырып әзірлеуді және кейіннен орналастыруды қолдайды, сонымен қатар болашақта басқа жобалардың жұмыс процестері мен тілдері енгізіледі.[2]

2.4.2 Maven

Maven Java көмегімен жобаларды құрастыруды автоматтандыру үшін қолданылады. Бұл белгілі бір бағдарламалық жасақтаманың қалай құрылатынын, сондай-ақ оның әртүрлі тәуелділіктерін анықтауға көмектеседі. Ол сіз құрып жатқан жобаны, бағдарламалық жасақтаманың үшінші тарап модульдері мен бөлшектеріне тәуелділіктерін, құрастыру ретін, сондай-ақ қажетті плагиндерді сипаттау үшін XML файлын пайдаланады. Қаптама және компиляция сияқты тапсырмалар үшін алдын-ала белгіленген мақсаттар бар.[2]

Maven кітапханалар мен плагиндерді әртүрлі репозитарийлерден жүктеп алады, содан кейін олардың барлығын жергілікті компьютердегі кэшке салады. Java жобалары үшін басым болғанымен, сіз оны Scala, Ruby және C #, сонымен қатар көптеген басқа тілдер үшін қолдана аласыз.

2.4.3 Gradle vs Maven

Екі жүйені құру тәсілінде кейбір түбегейлі айырмашылықтар бар. Gradle тапсырмаға тәуелділік графигіне негізделген - онда тапсырмалар жұмыс жасайтын нәрселер болып табылады, ал Maven фазалардың тұрақты және сызықтық моделіне негізделген. Maven көмегімен мақсаттар жобаның

кезеңдеріне қосылады, ал мақсаттар Gradle міндеттеріне ұқсас функцияны орындайды, бұл «жұмысты орындайтын заттар».

Өнімділікке сәйкес, екеуі де көп модульді құрылымдарды қатар жүргізуге мүмкіндік береді. Алайда, Gradle қосымша тапсырмаларды жасауға мүмкіндік береді, өйткені ол қай тапсырмалардың жаңартылғанын немесе жаңартылмағандығын тексереді. Егер ол болса, онда тапсырма орындалмайды, бұл сізге салу уақытын әлдеқайда қысқа етеді. Gradle-ден таба алатын басқа өнімділік сипаттамалары:

- Java сыныптарына арналған қосымша жинақтар;
- Java үшін компиляция жаса;
- қосымша тапсырмалар үшін арі қолдану;;
- компилятор демоны, ол компиляцияны тезірек жасайды.

Тәуелділікті басқару туралы айтатын болсақ, Gradle де, Maven де динамикалық және өтпелі тәуелділіктерді басқара алады, тәуелділіктің үшінші жақ кэштерін қолдана алады және POM метадеректерінің форматын оқи алады. Сіз сондай-ақ кітапхананың нұсқаларын орталық нұсқасының анықтамасы арқылы жариялауға және орталық нұсқасын қолдануға мәжбүр ете аласыз. Екеуі де артефакт қоймасынан өтпелі тәуелділіктерді жүктейді. Maven-де Maven Central, Gradle-де JCenter бар, және сіз өзіңіздің жеке компанияңыздың репозиторийін де анықтай аласыз. Егер бірнеше тәуелділіктер қажет болса, Maven оларды бір уақытта жүктей алады.

Алайда, Gradle API және іске асыруға тәуелділіктер туралы, сонымен қатар бір уақытта қауіпсіз кэштерге мүмкіндік беру кезінде жеңеді. Ол сондай-ақ кэштелген тәуелділіктермен бірге репозитарийдің метадеректерін сақтайды, сол кэшті қолданатын екі немесе одан да көп жобалардың бірін-бірі қайта жазбайтындығына кепілдік береді және оның бақылау сомасына негізделген кэші бар және кэшті репозиториймен синхрондауы мүмкін. Сонымен қатар, Gradle IVY метамәліметтерімен үйлесімді, бұл сізге динамикалық тәуелділікке нұсқаны көрсету үшін нұсқаулық ережелерін анықтауға және қайшылықтарды шешуге мүмкіндік береді. Бұл Maven-де жоқ. Тек Gradle-ден таба алатын тәуелділікті басқарудың басқа мүмкіндіктері:

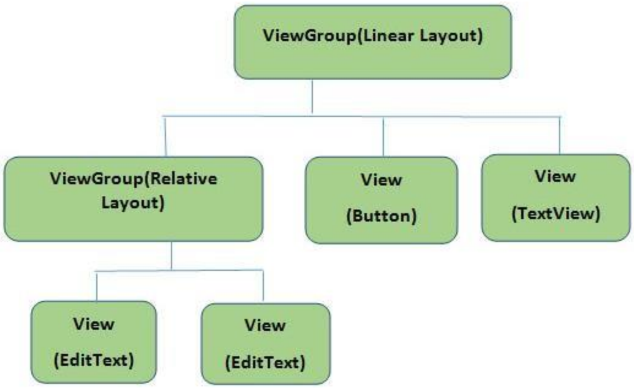
- үйлесімді кітапханалар үшін ауыстыру ережелерін қолдану;
- ауыстырылған ережелер бойынша қолдану;
- метадеректердің жақсы шешімі;
- жобалық тәуелділікті сыртқы тәуелділікке, керісінше, динамикалық түрде ауыстыру мүмкіндігі.

Gradle сонымен қатар композициялық құрылымдармен жұмыс істеу кезінде сізге жеңілірек уақыт береді және бұл сізге уақытша және тұрақты композиттік құрылымдармен жұмыс істеуге мүмкіндік береді, сонымен қатар IntelliJ IDEA Eclipse ішіне әртүрлі құрылымдарды біріктіреді және композициялық құрылымды импорттайды.

2.5 Android UI

XML-бұл деректерді сипаттау үшін қолданылатын HTML-ге өте ұқсас белгілеу тілі. XML тегтері XML-де алдын-ала анықталмаған. Біз өз тегтерімізді анықтауымыз керек. XML-ді адам да, машина да жақсы оқиды. Сонымен қатар, ол масштабталатын және қарапайым. Android-де біз XML - ді макеттерімізді жасау үшін қолданамыз, өйткені xml-бұл жеңіл тіл, сондықтан ол біздің орналасуымызды ауырлатпайды.

Android UI-нің барлық тұжырымдамасы View және ViewGroup нысандарының иерархиясымен анықталады. ViewGroup-бұл балалар көріністерін ұйымдастыратын көрінбейтін контейнер. Бұл балалар көріністері-бұл пайдаланушы интерфейсінің әртүрлі бөліктерін жасау үшін қолданылатын басқа виджеттер. Төмендегі суретте көрсетілгендей, бір ViewGroup-да бала элементі ретінде басқа ViewGroup болуы мүмкін (2.1-сурет):



2.1-сурет – View иерархиясы

2.6 Android Jetpack

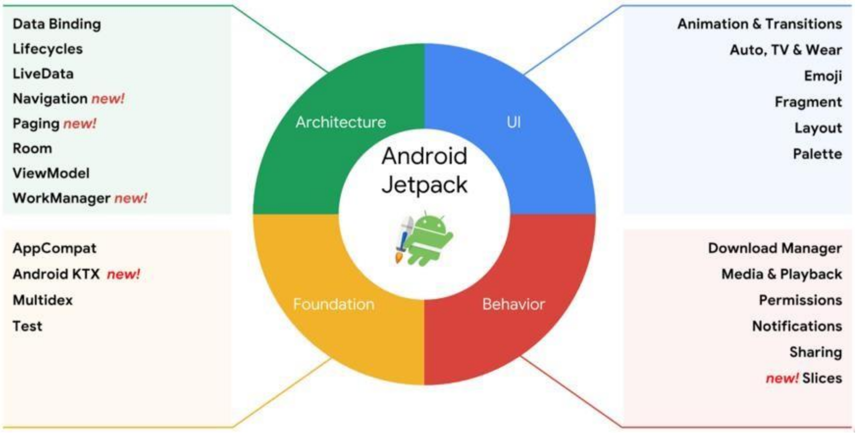
Android jetpack – әзірлеуіші жұмысын оңай ететін және тез әрі жүйелі қосымша жасауға арналған кітапханалар, құралдар мен архитектуралық нұсқаулар жиыны. Олар төртке бөлінеді (2.2-сурет):

Foundation – шаблонды кодтарды азайтуға арналған негізгі кітапхана. Қосымшаның тестілеуіне Test, барлық версияларда жұмыс жасауына AppCompat және Kotlin тілін қолдауға Android KTX кітапханаларын қамтиды.

Architecture Components – бұл кітапхана қосымшаның архитектурасын құруға көмектеседі. Оларға: NavigationComponent, ViewModel, Room, WorkManager және т.с.с кітапханалар жатады.

Behavior – Android Sdk ұсынатын функциионалдардың wrapper кітапханаларының жиынтығы. Олар: Permissions, Notifcations, CameraX

UI – қолданушы интерфейсін құруға арналған Fragment, Layout,Animation обьектілеріне функционалдар қосып жұмыс жасауды жеңілдетеді.[3]



2.2-сурет – Android Jetpack компоненттері

2.6.1 Android KTX

Android KTX – бұл Android Jetpack және Басқа Android кітапханаларына кіретін Котлин кеңейтімдерінің жиынтығы. KTX кеңейтімдері Jetpack, Android платформасы және басқа API үшін қысқа идиомалық Котлинді ұсынады. Ол үшін бұл кеңейтімдер Котлин тілінің бірнеше функцияларын қолданады, соның ішінде:

- кеңейту функциялары
- кеңейту қасиеттері
- ламбды
- аталған параметрлер
- әдепкі параметр мәндері
- сопрограммалар

Мысалы, SharedPreferences-пен жұмыс істеу кезінде сіз осы преференцияларға өзгерістер енгізбес бұрын редактор жасауыңыз керек.[3] Сондай-ақ, келесі мысалда көрсетілгендей өңдеуді аяқтаған кезде осы өзгерістерді қолдану немесе түзету керек (2.3-сурет)::

```
sharedPreferences
        .edit()
        .putBoolean("key", value)
        .apply()
```

2.3-сурет – SharedPreferences обьектін қолдану

Котлин Ламбдалары бұл пайдалану үшін өте қолайлы. Олар редакторды жасағаннан кейін кодты іске қосу үшін код блогын өткізіп, кодты орындауға мүмкіндік береді, содан кейін SharedPreferences API-ге өзгерістерді атомдық түрде қолдануға мүмкіндік береді.

Мұнда Android ktx Core, SharedPreferences функцияларының бірінің мысалы келтірілген.өңдеу функциясын SharedPreferences - ке қосатын өңдеу. Бұл функция қосымша логикалық жалаушаны өзгертулерді бекіту немесе қолдану керектігін көрсететін алғашқы дәлел ретінде қабылдайды. Ол сонымен қатар Lambda өрнегі түрінде SharedPreferences редакторында орындалатын әрекетті алады (2.4-сурет):

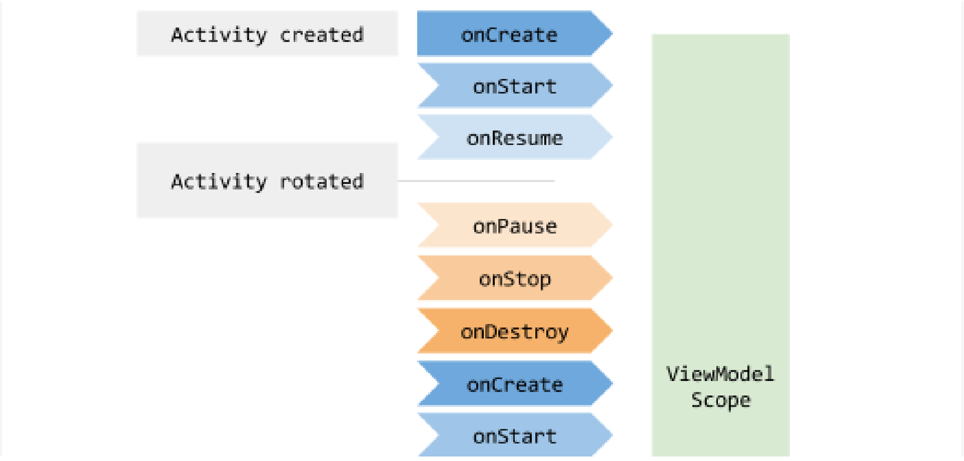
```
sharedPreferences.edit { putBoolean("key", value) }
sharedPreferences.edit(commit = true) { putBoolean("key", value) }
```

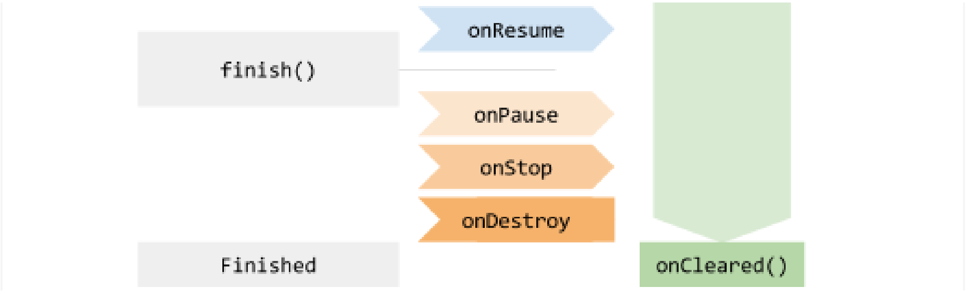
2.4-сурет – SharedPreferences обьектісін KTX көмегі арқылы қолдану

2.6.2 ViewModel

ViewModel – қолданушы интерфейсімен байланысты деректерді өмірлік циклінде сақтайтын және басқаратын класс.[6]

Android жүйесі қолданушы интерфейсінің контроллерінің яғни Activity мен Fragment өмірлік циклін басқарады. Қолдаунышының әрекетіне байланысты немесе жүйедегі белгілі бір себептерге байланысты жүйе контроллерді жойып қайтадан аша алады. Контроллер жойылып қайта ашылған кезде ол жақтағы деректерде бірге жойылып кетеді. Осындай жағдайда деректерді сақтап қалу үшін ViewModel классы көмектеседі. ViewModel классы контроллер толықтай жойылмайынша өзінің өмір сүруін тоқтатпайды. Төменде ViewModel мен контроллердің өмірлік циклі көрсетілген (2.5-сурет):



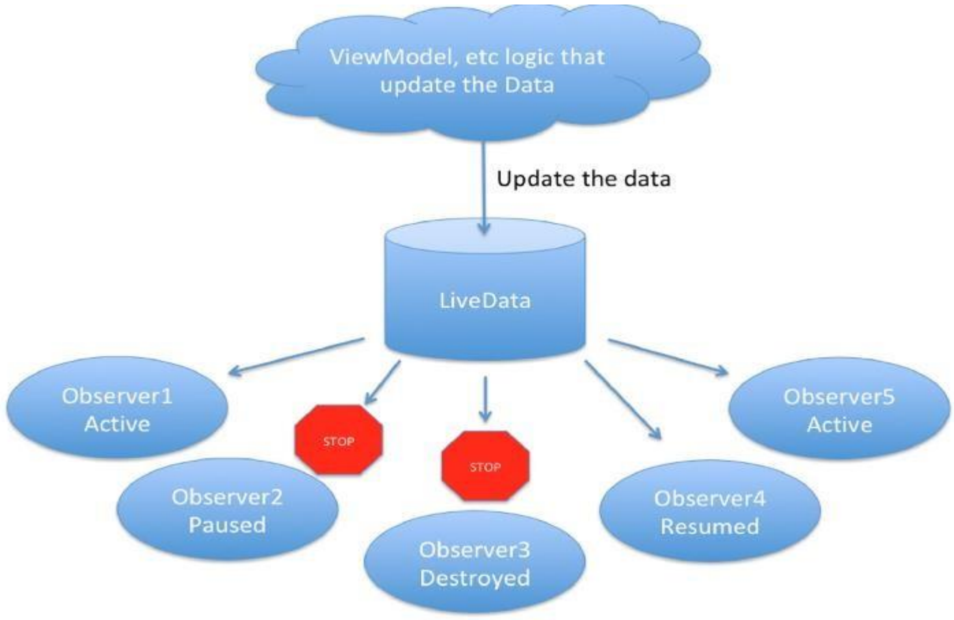


2.5-сурет – ViewModel өмірлің циклі

2.6.3 LiveData

LiveData – деректерді сақтайтын және Observable паттернін жүзеге асыратын класс. Қарапайым Observable класстан айырмашылығы мұнда Activity, Fragment, Service-тің өмірлің циклін бақылайды. LiveData деректерді белсенді өмірлік циклде тұрғана контроллерге яғни жағдайы тек STARTED немесе RESUMED болған кезде ғана деректерде жаңартады. LiveData классының жұмыс істеу схемасы төменде көрсетілген (2.6-сурет).

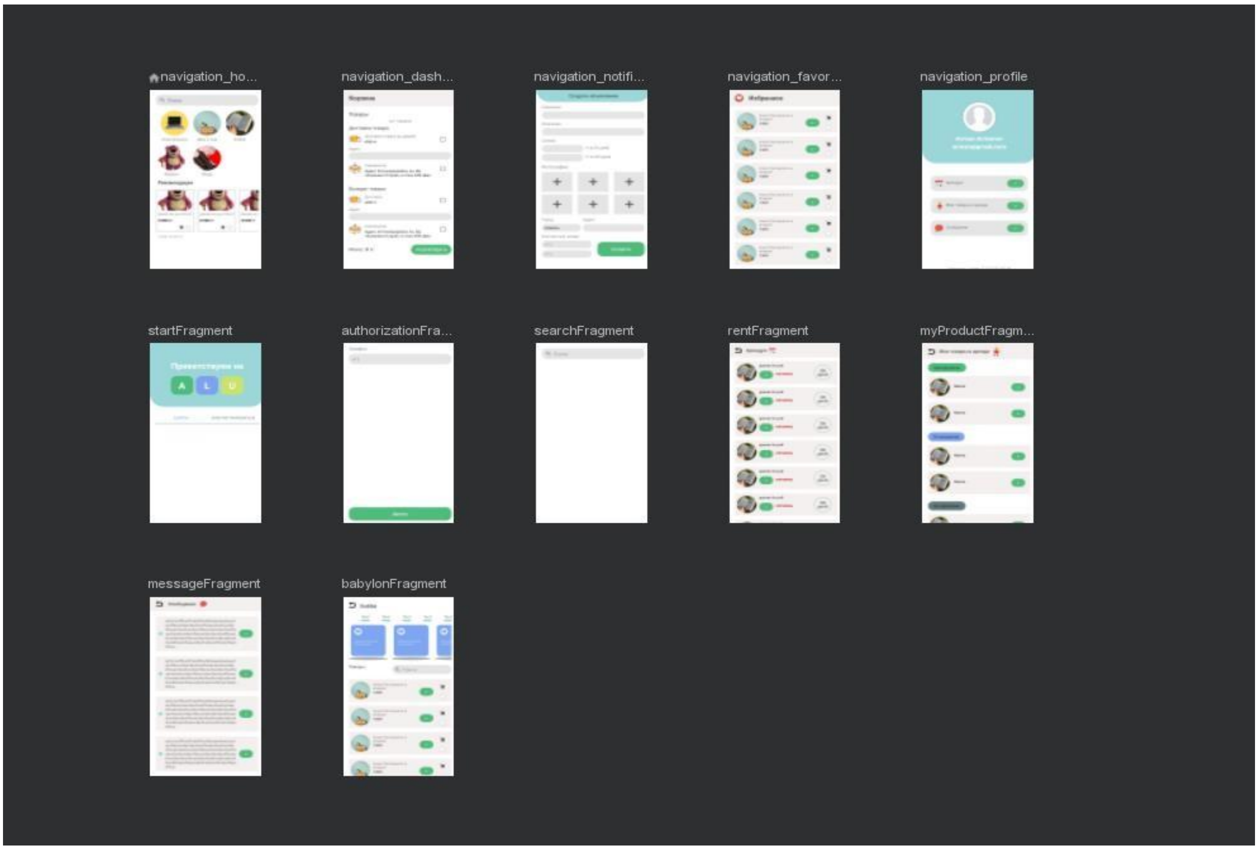
LifecycleOwner интерфейсін жүзеге асыратын объектімен бірге бақылаушыны тіркей алуға болады. Бұл байланыс тиісті өмірлік цикл объектісінің күйі "жойылды"деп өзгерген кезде бақылаушыны алып тастауға мүмкіндік береді. Бұл әсіресе әрекеттер мен фрагменттер үшін пайдалы, өйткені олар LiveData нысандарын қауіпсіз бақылап, жадының ағып кету туралы алаңдамайды—өмірлік циклдары жойылған кезде әрекеттер мен фрагменттер бірден жойылады.



2.6-сурет – LiveData жұмысының кескіні

2.6.4 Navigation

Navigation – бұл навигация қызметін жеңілдететін Android Jetpack архитектуралық компонентіне кіретін кітапхана. Бұл кітапхана арқылы жазатын код қысқарып, жалпы қосымша ішіндегі парақшалардың навигациясын визуалды түрде көруге көмектеседі (2.7-сурет). Және қосымша жасау үдерісін біршама жеңілдетеді. Бұл навигация компоненті негізігі үш бөлктен тұрады: NavigationGraph – мұнда қосымшадағы xml ресурстарды бір орталықтандырылған жерде ұстайды, және жеке бөліктерге ұстауға мүмкіндік береді; NavHost – бос контейнер компоненттердің бағытын көрсететін; NavController – қосымшадағы навигацияларды басқаратын обьект.



2.7-сурет – NavigationComponent мысалы

2.6.5 Retrofit 2

Retorfit – бұл Android пен Java бағдарлмалау тілдеріне арналған типті қауіпсіз http – клиент. Кітапхана арқылы сұранныстардың денесі, тақырыптары және API-сұраныстардың параметрлерін аннотация арқылы басқара отырып кодты түсінікті әрі қысқа етеді. Сұраныстар жасағанда басқа поток ашуды талап етпейді. Сонымен қатар шақыруларды синхронды және асинхронда жасауға

мүмкіндік береді. Автоматты түрде JSON объектілерін конвератция жасайды, қателерді өңдейді, файл жіберугі мүмкіндік береді.

API (қолданбалы бағдарламалау интерфейсі) - сыртқы бағдарламалық өнімдерде қолдануға арналған дайын сыныптар, процедуралар, функциялар, құрылымдар, константалар жиынтығы.

API көбінесе веб-қосымшалар мен қызметтерді жобалау кезінде қолданылады. Әдетте, API HTTP сұрауларының белгілі бір жиынтығын анықтайды және жауап беру үшін XML немесе JSON сияқты құрылымдар қолданылады. Веб-арі веб-қызметтің синонимі болып табылады, оның негізгі технологиялары SOAP және REST болып табылады.

Rest (Representational State Transfer) - Бағдарламалық Қамтамасыз Етуді Әзірлеу принциптерін сипаттайтын, үлестірілген жүйелерді құрудың сәулеттік стилі. REST HTTP протоколын қолданады, деректер жалпы стандартты форматтар (HTML, XML, JSON) түрінде беріледі. REST-тің негізгі артықшылықтары-оның қарапайымдылығы, сонымен қатар жүйенің жылдамдығы мен масштабталуын, функционалды үйлесімділігін арттыратын кэштеу. Осы мәліметтерден SOAP протоколдар мен стандарттардың бүкіл тобы деп қорытынды жасауға болады, бұл оның артық салмағы мен басқаларға қарағанда үлкен салмағын көрсетеді. SOAP http-ді көлік протоколы ретінде пайдаланады, ал REST оған негізделген, REST архитектурасында сервер деңгейіндегі кэш, масштабталу сияқты барлық технологиялар қолданылады, ал SOAP-та басқа құралдарды іздеу керек. Сонымен қатар, rest-тің SOAP-тен артықшылығы-SOAP XML-мен байланысты, ал REST json және басқа деректер форматтарын қолдана алады.

2.7 Firebase Cloud хабар алмасу

Firebase Cloud Messaging - бұл хабарламаларды ақысыз жіберуге мүмкіндік беретін платформалық хабар алмасу шешімі.

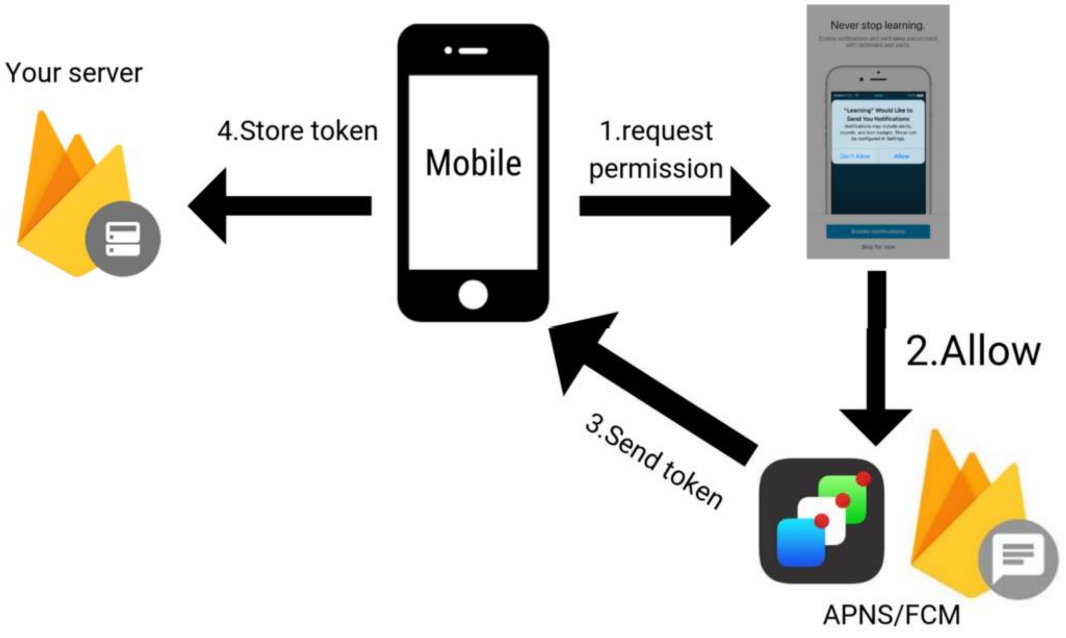
FCM көмегімен жаңа электрондық поштаны немесе басқа деректерді синхрондау үшін қол жетімді екендігі туралы клиенттік қосымшаны хабардар ете аласыз. Пайдаланушыларды қайта тарту және ұстап қалу үшін хабарландырулар жібере аласыз. Лездік хабар алмасу сияқты пайдалану жағдайларында хабарлама 4 КБ дейінгі пайдалы жүктемені клиенттік қоымшаға жібере алады. FCM енгізу жіберу және алу үшін екі негізгі компонентті қамтиды.

Firebase үшін бұлтты мүмкіндіктер немесе хабарламалар жасауға, теңшеуге және жіберуге болатын Бағдарлама сервері сияқты сенімді орта.

Платформаға тәуелді тиісті көлік қызметі арқылы хабарлама алатын iOS, Android немесе веб (JavaScript) клиенттік қосымшасы.

Push хабарландыруы-бұл қолданбада немесе шолғышта қысқа қалқымалы хабарлама. Ол пайдаланушыларға жаңартулар, жаңалықтар мен акциялар туралы сөйлесу үшін жіберіледі. Push хабарландыруының негізгі мақсаты-клиентке

келісімді сақтау үшін тиісті ақпаратты жеткізу. Төменде Push хабарламаның жұмыс істеу принципі көрсетілген (2.8-сурет):



2.8-сурет – Push харбаламаның жұмыс істеу принципі

3. Жобалау бөлімі

3.1 MVVM – қосымша архитектурасының жобалау үлгісі.

Бұл жобалау архитектурасының қолданунын себебі Google ресми түрде ұсынды. MVVM жобалау үлгісі Android Jetpack компоненттерімен жұмыс жасауды ыңғайлы етеді. Оның ішінде DataBinding. Және Android Repository архитектурасын жүзеге асыруға өте қолайлы.

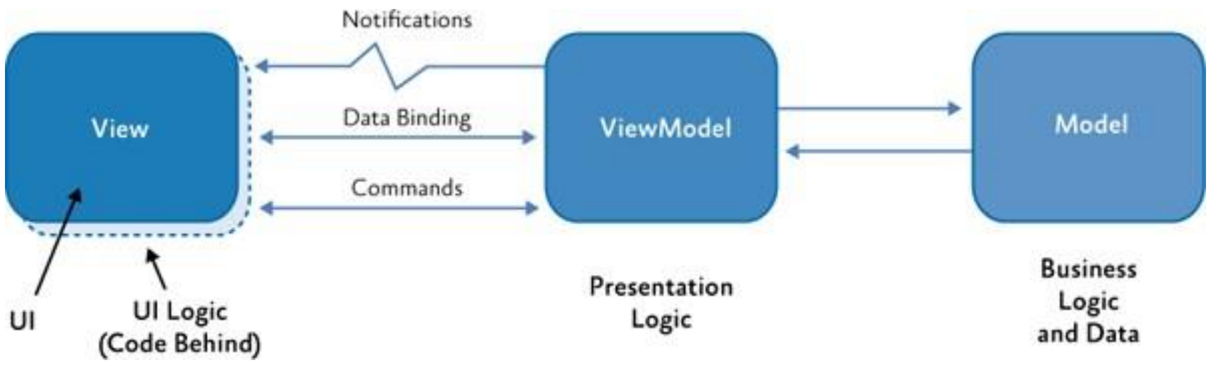
Қолданбадағы модель мен көрініс арасындағы тығыз байланысты бір-біріне ажырату үшін, яғни графикалық интерфейс пен деректермен жұмыс істеу логикасында болатын өзгерістерді бір-біріне тәуелсіз ету үшін қоладанылады (3.1-сурет). MVVM үлгісі үшке бөлінеді:

Модель (Model) – қолданбаның жұмыс істеуіне қажетті деректермен жұмыс логикасы мен фундаментальді деректерді сипаттайды.

Көрініс – бұл графикалық интерфейстер(баттырма, сурет, сөз). Көрініс моделі ұсынатын деректерде болып жатқан өзгерістер мен оқиғалардың жазылушысы ретінде қызмет атқарады. Көрініс моделінде бір өзгеріс болса онда ол, бұл өзгеріс жайлы өзінің барлық жазылуышыларына хабарлайды, ал көрініс өз кезегінде жаңартылған деректерді сұрайды.

Көрініс моделі (ViewModel) – Көрініс пен Модель арасындағы көпір ретінде қызмет атқарады. Яғни Көрініске керекті деректерді Моделден алып

өңдеп Көрініске ұсынады. Бұл қабат, Модель деректерді қайдан алып жатқанын білмеуі керек деректер қорынан ба әлде веб-серверден бе. Сонымен қатар Көріністің бар екенінде білмеуі керек.[8]



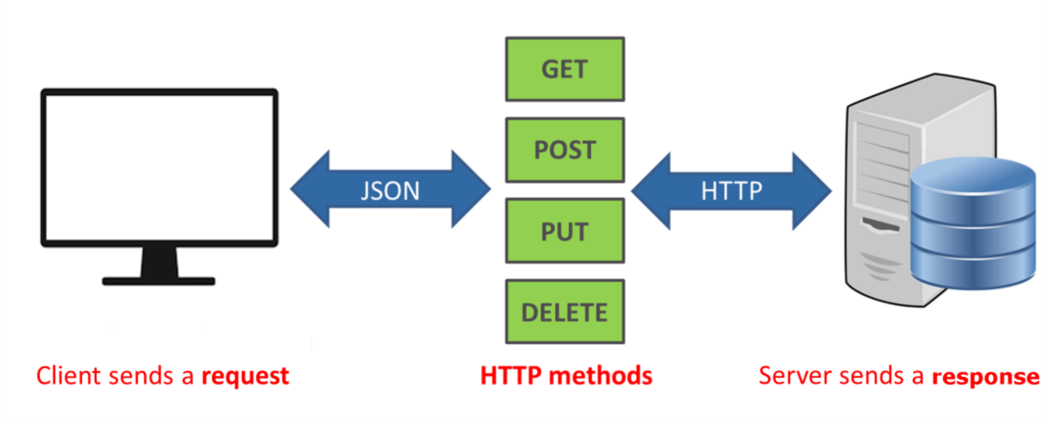
3.1-сурет – MVVM шаблонының схемасы

3.2 UI Архитектурасы

Қосымшаны SingleActivity архитектурасымен жазылған. Бұл архитектурада бір ғана Activity болады. Бір activity бір экранды бейнелеудің орнына, біз әрекетті экранды білдіретін әрекеттің ішіндегі фрагменттері бар үлкен контейнер ретінде қарастырамыз. Google бұл тәсілді қолдануды ұсынады, себебі бұл Navigation компоненнтің мүкіндігін кең көлемде қолдануға мүмкіндік береді.

3.3 Клиент-серверлік қосымша архитектурасы

Клиент-сервер архитектурасы, көптеген клиенттер (қашықтағы процессорлар) орталықтандырылған серверден (хост-компьютерден) қызмет сұрайтын және алатын компьютерлік желі архитектурасы. Клиенттік компьютерлер компьютер пайдаланушысына сервер қызметтерін сұрауға және сервер қайтарған нәтижелерді көрсетуге мүмкіндік беретін интерфейсті ұсынады.Серверлер клиенттерден сұраныстарды күтеді, содан кейін оларға жауап береді. Ең дұрысы, сервер клиенттерге стандартталған мөлдір интерфейсті ұсынады, сондықтан клиенттер қызмет көрсететін ерекшелігін білмейді. Клиенттер көбінесе жұмыс станцияларында немесе жеке компьютерлерде болады, ал серверлер желінің басқа жерлерінде, әдетте неғұрлым қуатты машиналарда орналасқан. Бұл есептеу моделі әсіресе клиенттер мен серверде әдетте орындалатын жеке тапсырмалар болған кезде тиімді болады. (3.2-сурет).

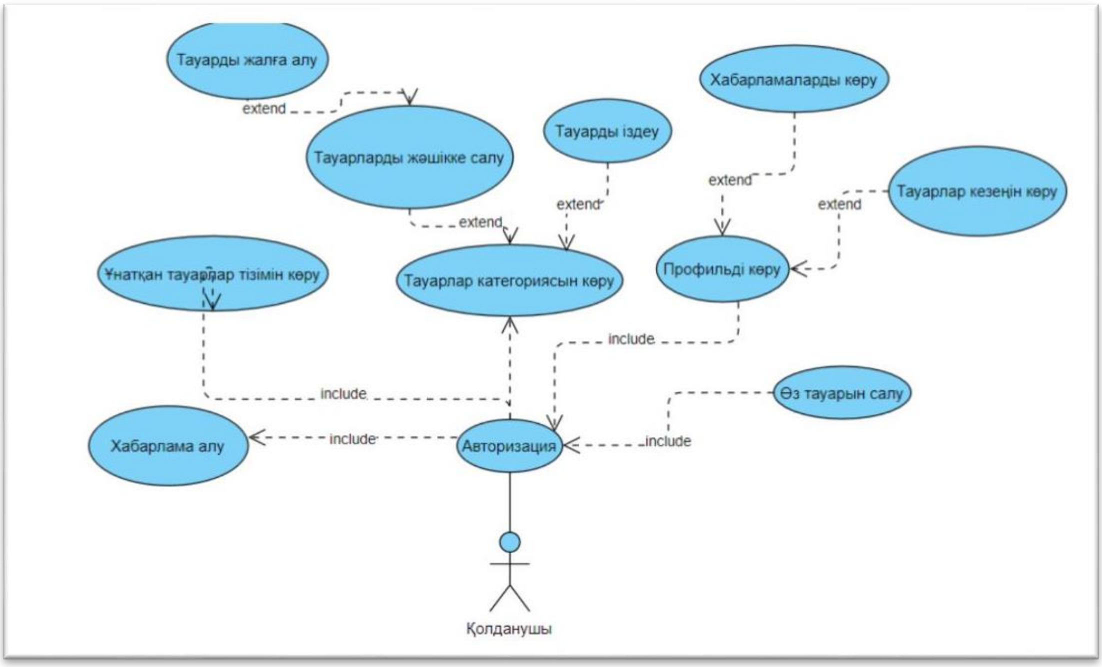


3.2-сурет – Клинет-серверлік бағдарламаның жұмыс істеу схемасы

3.4 Прецеденттер диаграммасы

Прецеденттер диаграммасы қолданушы мен прецеденттердің өзара әрекетін көрсетеді (3.3-сурет) . Прецеденттер диаграммасында бірнеше терминде қолданылады. Олар:

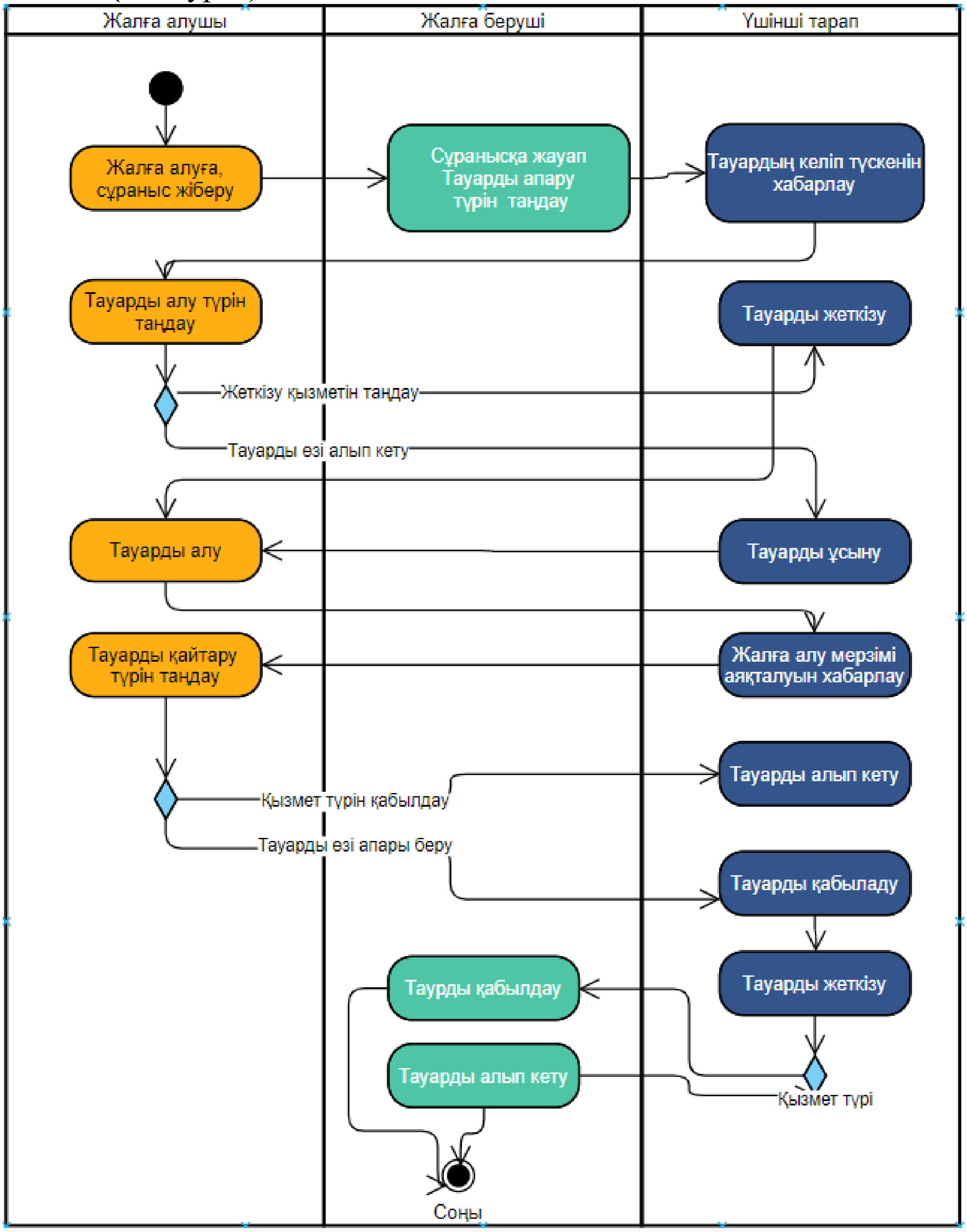
- актер – пайдаланушының жүйеге қатысты атқаратын рөлі.
- прецедент – жүйеде болаты іс-әрекеттерді көрсетеді
- include – прецеденттегі күрделі қадамды басқа прецедентпен ұсынуға болады. uml тілінде біз бірінші прецедент екінші прецедентті (includes) қамтиды деп айтамыз.



3.3-сурет. Прецеденттер диаграммасы

3.5 Белсенділік диаграммасы

Диаграмма көмегімен бизнес-процестерді, компания жұмыс істейтін алгоритмдерді бейнелеу ыңғайлы. Төмендеге суретте белсенділік диаграммасы көрестеліген (3.4-сурет):



3.4-сурет – Белсенділі диаграммасы

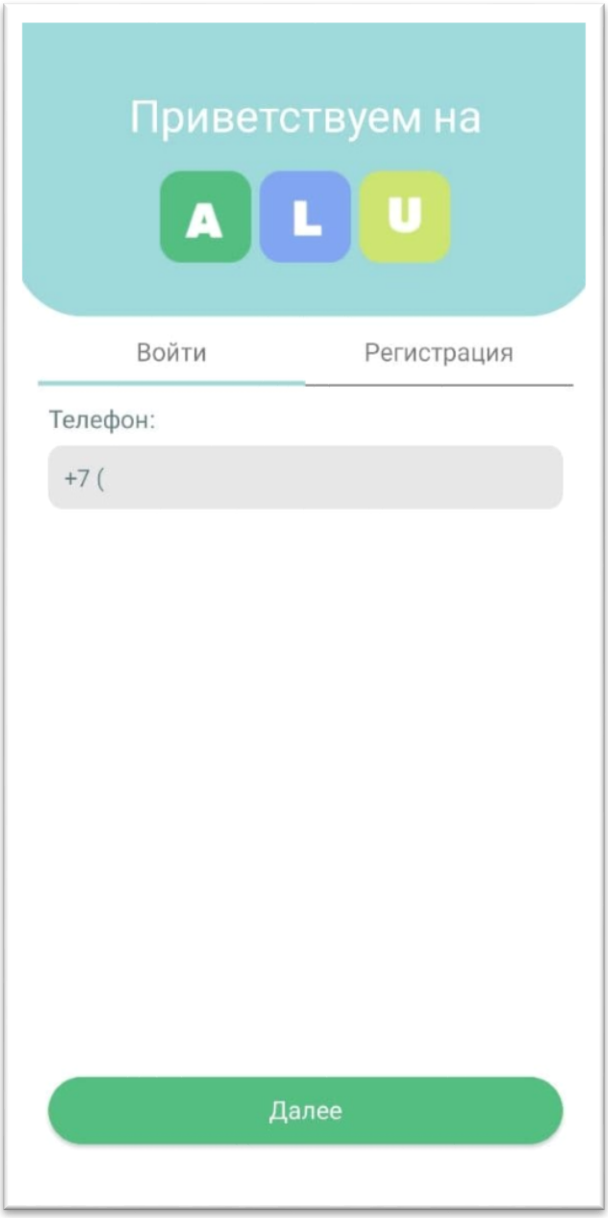
4 Тәжірибелік бөлім

4.1 Қолданушы интерфейсі

4.1.1 Авторизация

Жоғарыда айтылғандай әрбір парақша бір Fragment болып табылады. Бұл Single Activity принципіне негізделіп жасалған.

Мобильді қосымшаға кірген кезде қолданушы тіркелу керек. Тіркелу нөмір телефоны енгізіп растау коды арқылы жүзеге асады. (4.1-сурет)

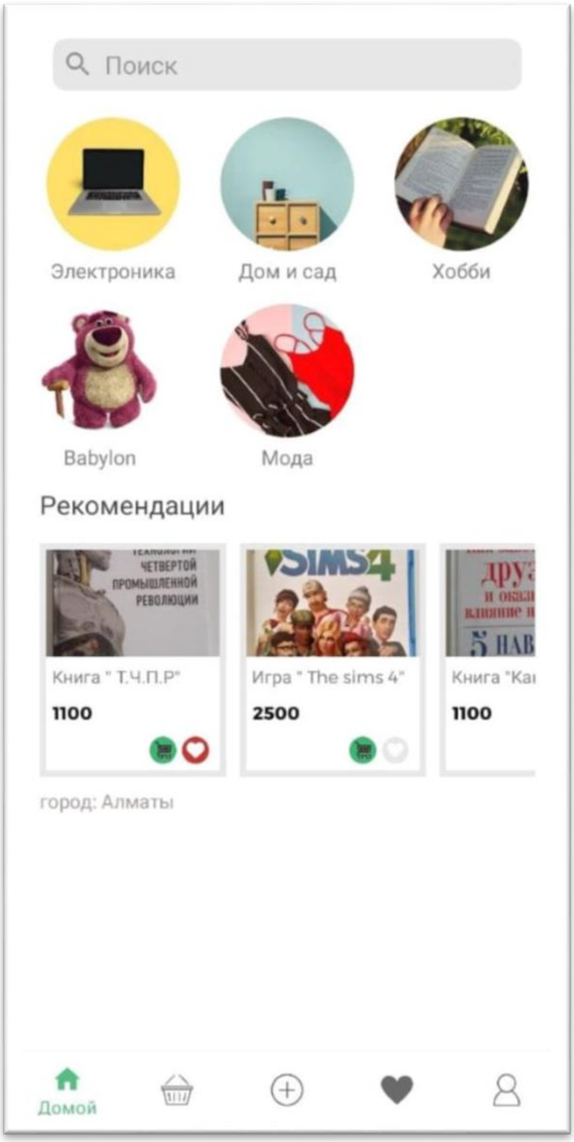


4.1-сурет – Қолданбаға авторизация жасау беті

4.1.2 Басты бет

Қолданбада негізгі мәзір ретінде BottomNavigation қолданылды (4.2-сурет). Мәзірде негізігі бес фрагментке кіруге болады. Төмендегі басты бетте тауарлардың категориялары және іздеу баттырмасы мен ұсыныстар көрсетіледі. Тауарларды бес категорияға бөлдік:

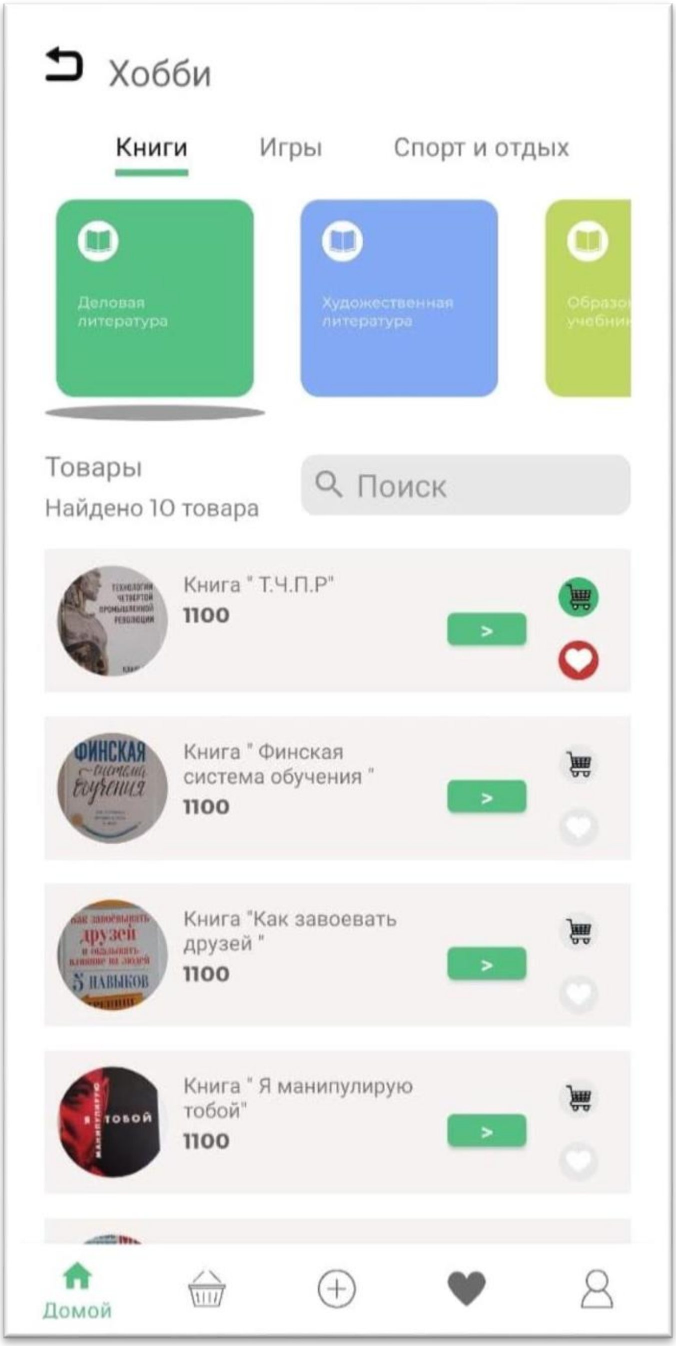
- электроника
- дом и сад
- хобби
- babylon
- мода.



4.2-сурет – Қосымшадағы басты бет

4.1.3 Тауарлар тізімі

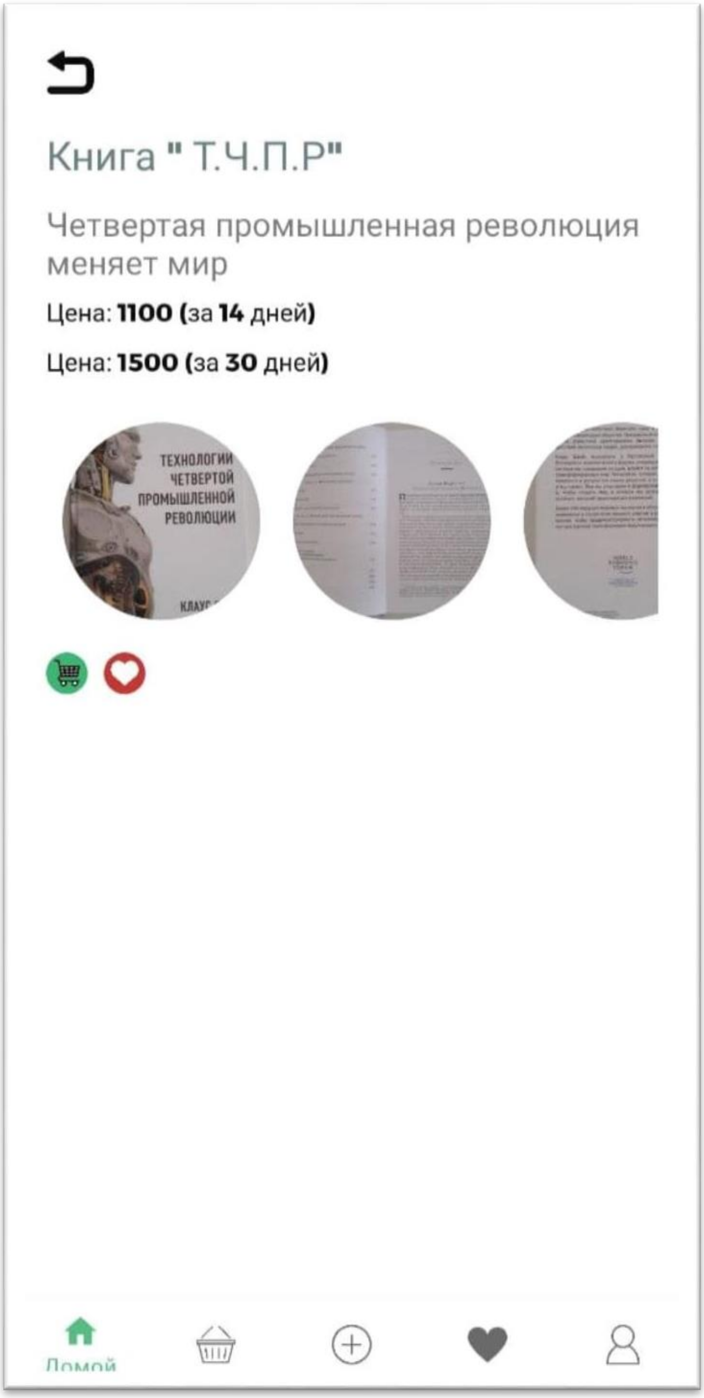
Басты беттегі категориялардың біріне кірген кезді төмендегі парақшаны ашады (4.3-сурет). Ол жерде тағы да таңдалынған категорияға сәйкес бірнеше бөлімдерге бөлінеді. Және тек қана таңдалынған тауарлар бойныша іздеу функциясы қарастырылған.



4.3-сурет – Категориялар мен тауарлардың тізімі

4.1.4 Тауар сипаттамасы

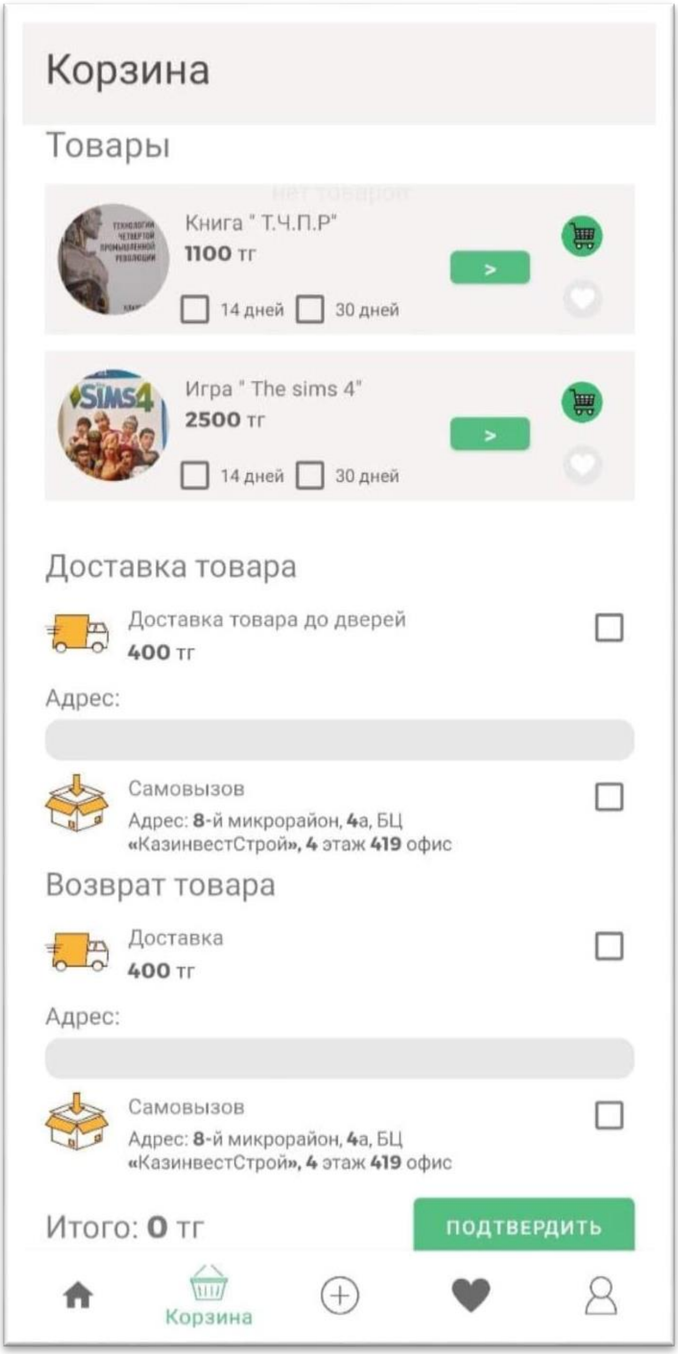
Таңдалынған тауар жайлы ақпаратты көре аламыз (4.4-сурет). Және тауарларды жәшіке немесе ұнатқан тауарлар сақталанын бөлімге жібере аламыз.



4.4-сурет – Тауардың сипаттамасы

4.1.5 Жалға алу

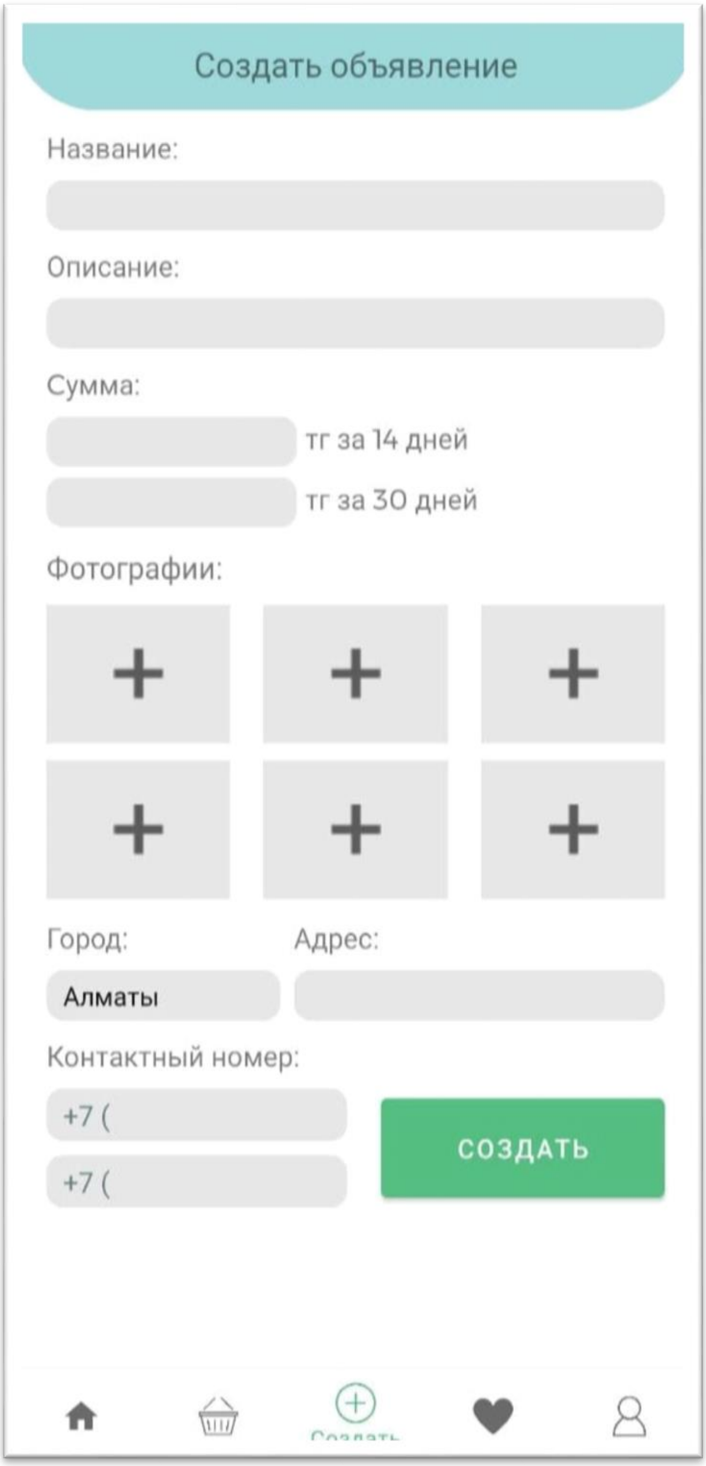
Төменде RecyclerFragment парақшасы (4.5-сурет). Бұл бетте қолданушы өзі таңдаған тауараларды жалға ала алады, немес алған тауарды қайтара алады. Бірнеше тауарға жалға алуға бірден өтінім қалдыруға мүмкіндік бар. Және жеткізу қызметінде таңдай алады.



4.5-сурет – Тауарларды жалға алатын немесе қайтаруға арналғаан бет

4.1.6 Хабарлама

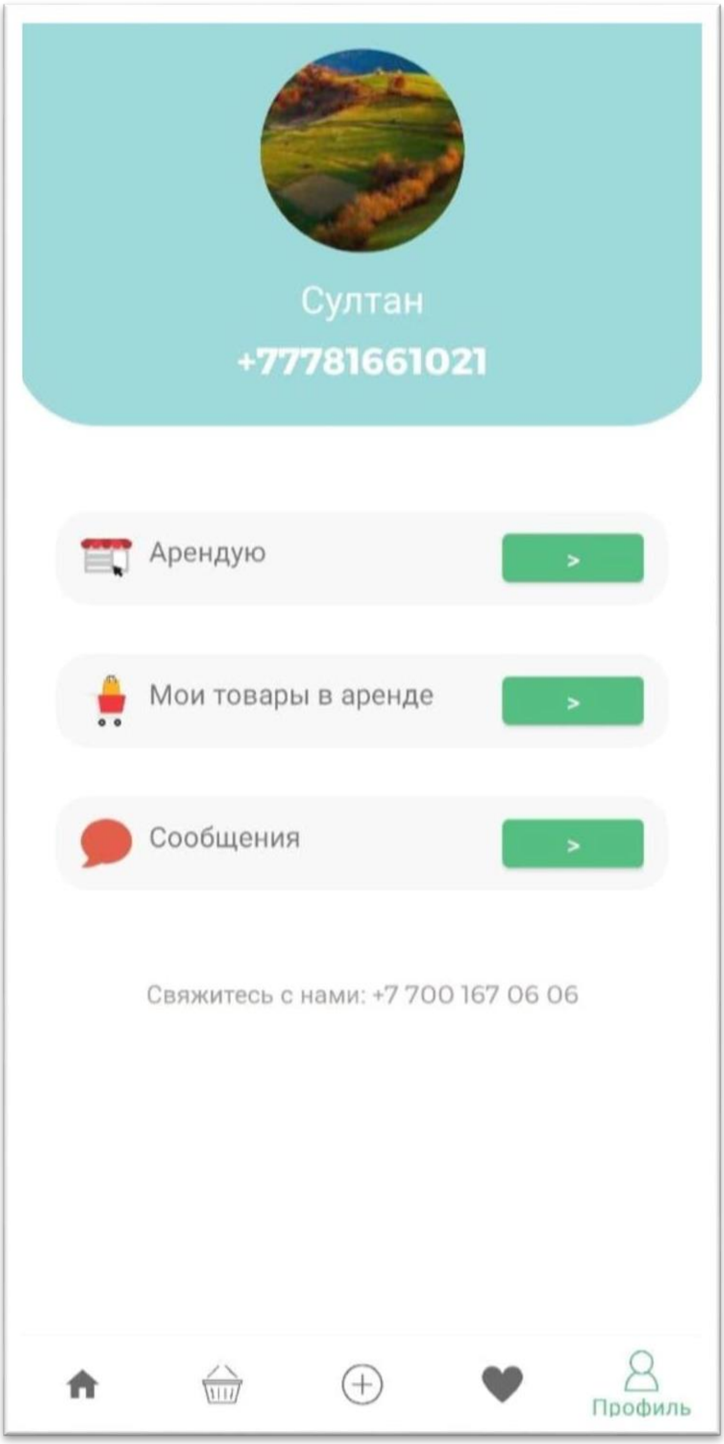
Бұл бетте қолданушы өзінің тауарын жалға беруге хабарлама жасай алады (4.6-сурет). Тауар тексеріс бөлімне түседі.



4.6-сурет – Өз тауарын салатын парақша

4.1.7 Профиль

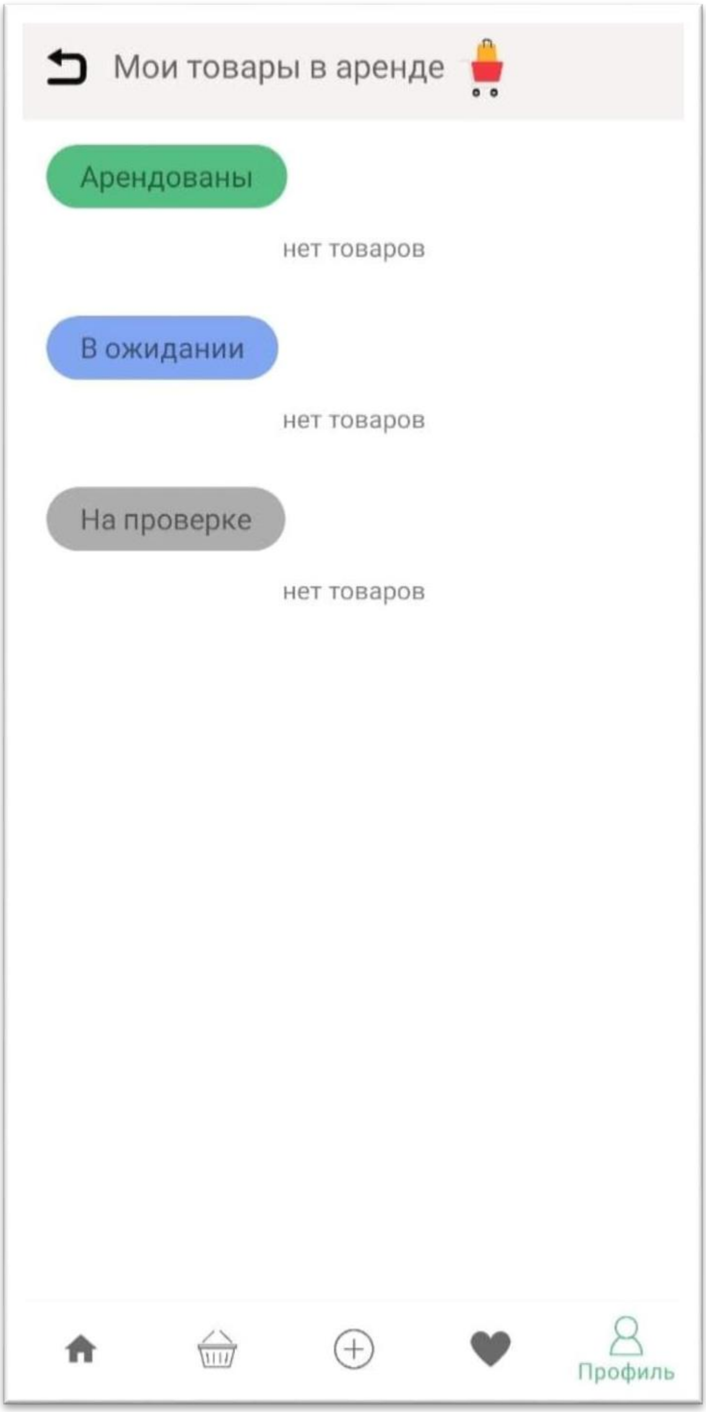
Қолдануышың профилінде үш бөлімге өтуге болады. Бірінші жалға алып жатқан тауарларының тізім көре алады. Екінші жалға беріп жатқан тауарларының тізімі, үшінше push хабарламлар тізімі жазылған (4.7-сурет).



4.7-сурет – Қолданушы профилі

4.1.8 Жалға беретін тауарлар

Қолданышы салынған тауар бірінші кезекті модераторлардың өңдеуінен өтіп сонан соң ғана жарыққа шығады. Бұл жағдайда тауар тексеріс бөлімінде тұрады. Одан соң күту бөлімінде тұрып, тауарды басқа қолданушы жалға алғаннан кейін ғана Жалға алынған бөліміне өтеді (4.8-сурет).



4.8-сурет – Тауардың кезеңін көрсететін бе

ҚОРЫТЫНДЫ

Дипломдық жұмыстың нәтижесінде Android ОЖ-де «ALU.KZ» деп аталатын мобильді қосымша жасалынып, Google Play Store платформасына жүктелді. Мобильді қосымшаны кез келген адам қолдана алады. Қосымша жалға алу мен жалға беру қызметтерін жүзеге асыру үшін электронды алаңды ұсынады. Сонымен қатар бірнеше тиімді функционалдар мен қауіпсіз қызметтерге ие.

Жұмыс барысында тиімді технологиялар мен кітапханалар таңдалынып, мобильді қосымша жылдам, ыңғайлы, әдемі интерфейстермен қамтасыз етілді. Әрі қарай мобильді қосымшаға өзгерістер енгізіп, дамытуға негізінде бағдарламалық бөлімі белгілі бір архитектураға жүгініп жасалынды. Жалға алу мен жалға беру мәселесін жеңілдету үшін барынша тиімді әрі оңай бизнес логика құрылып, және жобаның толық жүзеге асуына үшінші тарап қатысатын болып шешілді. Осы бизнес процесстер цифрлы түрге көшу үшін мобильді қосымша әзірленді.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1. Android SDK // Электрондық нұсқа https://ru.wikipedia.org/wiki/Android_SDK

2. Gradle және Maven // Электрондық нұсқа https://dzone.com/articles/gradle-vs-maven

3. Android Jetpack // Электрондық нұсқа https://developer.android.com/jetpack

4. Android KTX // Электрондық нұсқа https://developer.android.com/kotlin/ktx

5. FCM // Электрондық нұсқа https://firebase.google.com/docs/cloud-messaging

6. View Model // Электрондық нұсқа https://developer.android.com/topic/libraries/architecture/viewmodel

7. Retrofit 2 // Электрондық нұсқа https://square.github.io/retrofit/

8. MVVM архитектурасы // Электрондық нұсқа https://ru.wikipedia.org/wiki/Model-View-ViewModel

А Қосымшасы
(міндетті)

Техникалық тапсырма

А.1.1 Мобильді қосымшаны әзірлеуге арналған техникалық тапсырма

Техникалық тапсырма Android ОЖ-де мобильді қосымша әзірлеу мақсатында қолданылады. Қосымшаны кез келген жалға алу және жалға беру қызметтеріне қажеттілігі бар тұтынушы қолдана алады. Мобильді қосымша қолдаунышыларға қажетті қызметтерін жүзеге асыруына тез әрі оңай жетуіне көмектеседі.

А.1.1.1 Әзірлеу негізі

Қосымша дипломдық жұмыс негізігнде әзірленеді.

А.1.1.2 Мақсаты

Жеке тұлғалардың қызмет көрсетуі бойынша Android жүйесінде қолданба әзірлеу. Қосымшаның негізгі қызметі қауіпсіз жалға алу және жалға беру қызметтерін жүзеге асыру.

А.1.1.2 Қосымшаның сыртқы түріне қойылатын талаптар

Қосымшаның сыртқы түрі интуитивті түсінікті және пайдалануға ыңғайлы болуы керек.

А.1.1.3 Функционалдық сипаттамаларына қойылатын талаптар

- қоланушылар тіркеле және авторизация жасай алу
- жалға тауар алу және беру
- жалға алатын тауарларды категориялар бойынша жүйелі түрде көрек алу
- тауарларды глобальді түрде кілт сөзбен іздеу
- ұнатқан тауарларын сақтап қоя алу
- жеткізу қызметін (тауарды алу немесе беру) таңдау

А.1.1.4 Сенімділікке қойылатын талаптар

Қолдануышалар енгізетін ақпарат алдын ала тексерістен өтіп содан соң ғана жарияланады. Барлық сервермен барлық байлыныстар қауіпсіз протоколмен жүзеге асады. Деректердің толықтығы сақталады.

А.1.1.5 Құрамы мен техникалық құралдар параметрлеріне қойылатын талаптар

Қосымшаның толықтай жұмыс жасауына интернет желісімен байланыс қажет

А.1.1.6 Бағдарламалық құжаттамаға қойылатын талаптар

Бағдарлама коды бойынша түсінектеме міндетті түрде болуы қажет.

Б Қосымшасы
(міндетті)

Бағдарлама коды

Желіден келетін деректердің wrapper классы (Data.kt)

```
data class Data<RequestType>(val status: Status, val data: RequestType? = null, val message:
String? = null)

enum class Status{SUCCESFULL, ERROR, LOADING}
```

Локальді деректерді сақтайтын қойма (SharedPreference.kt)

```
import android.content.Context
import android.content.SharedPreferences
import androidx.core.content.edit
import com.example.alu.utills.Constants

class SharedPreference {

    companion object {
        @Volatile private var  INSTANCE : SharedPreferences? = null

        fun getInstance(context: Context) : SharedPreferences =
            INSTANCE ?: synchronized(this) {
                INSTANCE ?: context.getSharedPreferences(Constants.NAME_PREFENCES,
Context.MODE_PRIVATE)
            }
    }
}
```

Желімен жұмыс жасайтын сервис (NetworkService.kt)

```
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

object NetworkService{

 const val BASE_URl = "https://**********/"

    private val retrofit = Retrofit.Builder()
        .baseUrl(BASE_URl)
        .addConverterFactory(GsonConverterFactory.create())
        .build()
    val instance: ApiService = retrofit.create(
        ApiService::class.java)
}
```

User модель классы (User.kt)

```
package com.example.alu.model
import com.google.gson.annotations.SerializedName

data class User(
    @SerializedName("avatar")
    val avatar: String,
    @SerializedName("nickname")
    val nickname: String,
    @SerializedName("phone")
    val phone: String
)
```

ProfileFragment.kt коды

```
import android.app.Activity
import android.content.Intent
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.net.Uri
import android.os.Bundle
import android.util.Base64
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.TextView
import android.widget.Toast
import androidx.fragment.app.Fragment
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProviders
import androidx.navigation.fragment.findNavController
import com.bumptech.glide.Glide
import com.example.alu.R
import com.example.alu.data.Status
import com.example.alu.viewmodel.ProfileViewModel
import com.theartofdev.edmodo.cropper.CropImage
import com.theartofdev.edmodo.cropper.CropImageView
import de.hdodenhof.circleimageview.CircleImageView
import java.io.ByteArrayOutputStream

class ProfileFragment : Fragment() {

    private lateinit var viewModel: ProfileViewModel
    private lateinit var avatar : CircleImageView
    private lateinit var name: TextView
```

```
private lateinit var phone: TextView

override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?


): View? {
    val view = inflater.inflate(R.layout.profile_fragment, container, false)
    viewModel = ViewModelProviders.of(this).get(ProfileViewModel::class.java)
    initView(view)

    view.findViewById<LinearLayout>(R.id.prof_irent).setOnClickListener {
        findNavController().navigate(R.id.action_navigation_profile_to_irentFragment)
    }
    view.findViewById<LinearLayout>(R.id.prof_myrent).setOnClickListener {
        findNavController().navigate(R.id.action_navigation_profile_to_messsageFragment)
    }
    view.findViewById<LinearLayout>(R.id.prof_message).setOnClickListener {
        findNavController().navigate(R.id.action_navigation_profile_to_myRentedFragment)
    }

    avatar.setOnClickListener {
        CropImage.startPickImageActivity(activity?.baseContext!!, this)
    }

    getUser()

    viewModel.userResponse.observe(viewLifecycleOwner, androidx.lifecycle.Observer {
        when (it.status) {
            Status.ERROR -> {
                showToast(R.string.internet_false)
            }
            Status.SUCCESFULL -> {
                name.text = it.data?.nickname
                phone.text = it.data?.phone
                Glide.with(this).load(it.data?.avatar).into(avatar)
            }
        }
    })
    viewModel.addAvatarResponse.observe(viewLifecycleOwner, Observer {
        when (it.status) {
            Status.ERROR -> {
                showToast(R.string.internet_false)
            }
            Status.LOADING -> {
                avatar.isEnabled = false
            }
            Status.SUCCESFULL -> {
```

```
                avatar.isEnabled = true
            }

        }
    })
    return view
}

private fun initView(view: View) {
    avatar = view.findViewById(R.id.profile_avatar)
    name = view.findViewById(R.id.profile_name)
    phone = view.findViewById(R.id.profile_phone)


}

private fun getUser() {
    viewModel.getUser()
}

private fun showToast(text: Int) {
    Toast.makeText(activity, text, Toast.LENGTH_SHORT).show()
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    if (requestCode == CropImage.PICK_IMAGE_CHOOSER_REQUEST_CODE &&
resultCode == Activity.RESULT_OK) {
        val imageUri = CropImage.getPickImageResultUri(activity?.baseContext!!, data)
        if (CropImage.isReadExternalStoragePermissionsRequired(
                activity?.baseContext!!,
                imageUri
            )
        ) {

requestPermissions(arrayOf(android.Manifest.permission.READ_EXTERNAL_STORAGE), 0)
        } else {
            startCrop(imageUri)
        }
    }
    if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE && data !=
null) {
        val result: CropImage.ActivityResult = CropImage.getActivityResult(data)
        if (resultCode == Activity.RESULT_OK) {
            setPic(avatar, result.uri)
        }
    }
}

private fun startCrop(imageuri: Uri) {
```

```
        CropImage.activity(imageuri)
            .setGuidelines(CropImageView.Guidelines.ON)
            .setMultiTouchEnabled(true)
            .start(activity?.baseContext!!, this)
        }

    private fun setPic(imageView: ImageView, uri: Uri) {
        val targetW: Int = imageView.width
        val targetH: Int = imageView.height

        val bmOptions = BitmapFactory.Options().apply {

            inJustDecodeBounds = true

            BitmapFactory.decodeFile(uri.path, this)

            val photoW: Int = outWidth
            val photoH: Int = outHeight

            val scaleFactor: Int = Math.max(1, Math.min(photoW / targetW, photoH
/ targetH))

            inJustDecodeBounds = false
            inSampleSize = scaleFactor
            inPurgeable = true
        }
        BitmapFactory.decodeFile(uri.path, bmOptions)?.also { bitmap ->
            imageView.setImageBitmap(bitmap)
            viewModel.addAvatar(toBase64(bitmap))
        }
    }

    private fun toBase64(bitmap: Bitmap): String {
        val byteArrayOutputStream = ByteArrayOutputStream()
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, byteArrayOutputStream)
        val byteArray: ByteArray = byteArrayOutputStream.toByteArray()
        return Base64.encodeToString(byteArray, Base64.DEFAULT)
    }
}
```

ProfileViewModel.kt коды

```
package com.example.alu.viewmodel

import android.app.Application
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.LiveData
```

```
import androidx.lifecycle.MutableLiveData
import  androidx.lifecycle.viewModelScope
import com.example.alu.data.Data
import com.example.alu.net.NetworkService
import com.example.alu.data.SharedPreference
import com.example.alu.data.Status
import com.example.alu.model.AddAvatar
import com.example.alu.model.User
import com.example.alu.net.ApiService
import com.example.alu.utills.Constants
import kotlinx.coroutines.launch
import java.io.IOException

class ProfileViewModel(application: Application) : AndroidViewModel(application) {

    private val _addAvatarResponse: MutableLiveData<Data<AddAvatar>> = MutableLiveData()
    private val _userResponse: MutableLiveData<Data<User>> = MutableLiveData()

    private val _sharedPreference = SharedPreference.getInstance(getApplication())
    private val _apiService: ApiService = NetworkService.instance

    val addAvatarResponse: LiveData<Data<AddAvatar>>
        get() {
            return _addAvatarResponse
        }

    val userResponse: LiveData<Data<User>>


        get() {
            return _userResponse
        }

    fun addAvatar(base64: String) {
        _addAvatarResponse.value = Data(Status.LOADING)
        viewModelScope.launch {
            try {
                val token = getToken()
                val request = _apiService.addAvatar("Token $token", base64)
                val result: Data<AddAvatar> = if (request.isSuccessful) Data(
                    Status.SUCCESFULL,
                    data = request.body()
                ) else Data(Status.ERROR, data = null, message = "Server error")
                _addAvatarResponse.postValue(result)
            } catch (exception: IOException) {
                _addAvatarResponse.postValue(
                    Data(
                        status = Status.ERROR,
                        data = null,
```

```
                message = "Internet exception"
              )
            )
          }
        }
      }

    fun getUser() {
      _userResponse.value = Data(Status.LOADING)
      viewModelScope.launch {
        try {
          val token = getToken()
          val uid = getUID()
          val request = _apiService.getUser("Token $token", uid!!)
          val result: Data<User> = if (request.isSuccessful) Data(
            Status.SUCCESFULL,
            data = request.body()
          ) else Data(Status.ERROR, data = null, message = "Server error")
          _userResponse.postValue(result)
        } catch (exception: IOException) {
          _userResponse.postValue(
            Data(
              status = Status.ERROR,
              data = null,
              message = "Internet exception"
            )
          )
        }
      }
    }

    private fun getUID() = _sharedPreference.getString(Constants.UID, " ")

    private fun getToken() =
      _sharedPreference.getString(Constants.KEY, " ")

  }
```

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
СӘТБАЕВ УНИВЕРСИТЕТІ

5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандығы

Сейітмат Мұхаммед-Сұлтан

Тақырыбы: Жеке тұлғалардың қызмет көрсетуіне арналған Андроид жүйесінде қолданба әзірлеу

ҒЫЛЫМИ ЖЕТЕКШІНІҢ ПІКІРІ

Диплом жобасын жасаушы Сейітмат Мұхмамед-Сұлтан алдына жеке тұлғалардың қызмет көрсетуіне арналған Андроид жүйесінде қолданба әзірлеу жасау тапсырмасы қойылған. Дипломдық жобаның мақсаты жалға алу немесе жалға беру процестерін автоматтандыру үшін мобильді қосымша жасау. Мобильді қосымша арқылы электронды алаң жасалып, онда кез келген қолданушы тауарын жалға беріп немесе жалға ала алады.

Жұмыс барысында қойылған тапсырмалар: қолданушы интерфейсін құру, қосымшаны архитектураға жүгініп жасау. Менің пікірім бойынша, диплом жазушы үміткер алдына қойылған тапсырманы толығымен орындады және ақпараттық жүйелердің заманауи технологияларын жақсылап меңгергендігін көрсетті деп ойлаймын.

Жоба жетекшісі ретінде бұл дипломдық жұмысты өз деңгейінде жасады деп есептей отырып Сейітмат Мұхаммед-Сұлтан 5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандығы бойынша «Техника және технологиялар бакалавры» академиялық дәрежесін тағайындауға болады деп есептеймін.

Ғылыми жетекші: «Программалық инженерия» кафедрасының техникалық ғылымдарының магистрі, лектор______________________________А. Қайрбеков.

Тапсырманы орындауға алған білім алушы_______________________М.Сейітмат.

Күні " 04 " маусым 2021ж.

StrikePlagiarism.com SATBAYEV UNIVERSITY

Дата отчета 5/26/2021
Дата редактирования ---

Отчет не был оценен.

Метаданные

Название
Сейітмат Мұхаммед-Сұлтан (1).docx

Автор
Сейітмат Мұхаммед-Сұлтан

Научный руководитель
Алибек Байгаринов

Подразделение
ИКиИТ

Список возможных попыток манипуляций с текстом

В этом разделе вы найдете информацию, касающуюся манипуляций в тексте, с целью изменить результаты проверки. Для того, кто оценивает работу на бумажном носителе или в электронном формате, манипуляции могут быть невидимы (может быть также целенаправленное вписывание ошибок). Следует оценить, являются ли изменения преднамеренными или нет.

Замена букв	ß	0
Интервалы	A→	0
Микропробелы	·	3
Белые знаки	ß	0
Парафразы (SmartMarks)	a	13

Объем найденных подобий

Обратите внимание!Высокие значения коэффициентов не означают плагиат. Отчет должен быть проанализирован экспертом.

КП1	КП2	КЦ
3.72% / 3.72%	0.59% / 0.59%	0.59% / 0.59%
25 — Длина фразы для коэффициента подобия 2	5074 — Количество слов	43927 — Количество символов

Подобия по списку источников

Просмотрите список и проанализируйте, в особенности, те фрагменты, которые превышают КП №2 (выделенные жирным шрифтом). Используйте ссылку «Обозначить фрагмент» и обратите внимание на то, являются ли выделенные фрагменты повторяющимися короткими фразами, разбросанными в документе (совпадающие сходства), многочисленными короткими фразами расположенные рядом друг с другом (парафразирование) или обширными фрагментами без указания источника ("криптоцитаты").

10 самых длинных фраз

Цвет текста

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ И АДРЕС ИСТОЧНИКА URL (НАЗВАНИЕ БАЗЫ)	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	https://melimde.com/saba-5-tairibi-android-sdk-rali-jospar-android-sdk-rali.html	30	0.59 %
2	https://developer.android.com/training/camera/photobasics	17	0.34 %
3	https://developer.android.com/training/camera/photobasics	15	0.30 %
4	**YFCNU/2013/phys/phys_2013_019.pdf** YFCNU **10/29/2019** Yuriy Fedkovych Chernivtsi National University(CNU) (Deanery)	14	0.28 %
5	https://developer.android.com/training/camera/photobasics	14	0.28 %
6	https://melimde.com/saba-5-tairibi-android-sdk-rali-jospar-android-sdk-rali.html	14	0.28 %

7	https://developer.android.com/training/camera/photobasics	12	0.24 %
8	https://developer.android.com/training/camera/photobasics	12	0.24 %
9	**YFCNU/2013/phys/phys_2013_019.pdf** YFCNU **10/29/2019** Yuriy Fedkovych Chernivtsi National University(CNU) (Deanery)	11	0.22 %
10	https://melimde.com/saba-5-tairibi-android-sdk-rali-jospar-android-sdk-rali.html	11	0.22 %

из базы данных RefBooks (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из домашней базы данных (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из программы обмена базами данных (0.95 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	**YFCNU/2013/phys/phys_2013_019.pdf** YFCNU **10/29/2019** Yuriy Fedkovych Chernivtsi National University(CNU) (Deanery)	41 (4)	0.81 %
2	**Марғұлан Соңғы ДИПЛОМ (1).docx** Марғұлан Ерболұлы **6/2/2020** Turan University (TU) (Экономический факультет)	7 (1)	0.14 %

из интернета (2.78 %)

ПОРЯДКОВЫЙ НОМЕР	ИСТОЧНИК URL	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	https://developer.android.com/training/camera/photobasics	70 (5)	1.38 %
2	https://melimde.com/saba-5-tairibi-android-sdk-rali-jospar-android-sdk-rali.html	55 (3)	1.08 %
3	https://github.com/ArthurHub/Android-Image-Cropper/issues/758	16 (2)	0.32 %

Список принятых фрагментов (нет принятых фрагментов)

ПОРЯДКОВЫЙ НОМЕР	СОДЕРЖАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Сейітмат Мұхаммед-Сұлтан

Название: Жеке тұлғалардың қызмет көрсетуіне арналған Андроид жүйесінде қолданба әзірлеу

Координатор: Еркежан Сейтбекова

Коэффициент подобия 1: 3.72%
Коэффициент подобия 2: 0.59%

Замена букв: 0
Интервалы: 0
Микропробелы: 3
Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

- ☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;
- ☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
- ☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:
……

……26.05.2021…………   ……………………………………………

Дата   Подпись Научного руководителя

Форматы	Зонасы	Поз-сыz	Белгілеулер	Аталулары	Саны	Қоры-тынды

					Дипломдық жоба	Бет
						46
Өлш	Бет	Құжат №	қолы	күні		