ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

Программалық инженерия кафедрасы

Мүсілімов Дінмұхамед Бақытұлы

Тақырыбы: Жеке тұлғаларға қызмет көрсетуге арналған IOS мобильді әзірлемесі

Дипломдық жобаға
ТҮСІНІКТЕМЕЛІК ЖАЗБА

5В070400 – «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 кафедра меңгерушісі,
PhD докторы
_____ М. Тұрдалыұлы
“06” 06 2021 ж.

Дипломдық жобаға

ТҮСІНІКТЕМЕЛІК ЖАЗБА

Тақырыбы Жеке тұлғаларға қызмет көрсетуге арналған IOS мобильді әзірлемесі

Мамандығы 5В070400 – Есептеу техникасы және бағдарламалық қамтамасыз ету

Орындаған Мүсілімов Д.Б

Ғылыми жетекші
техн. ғыл. магистрі, лектор
_____ А.М. Кайрбеков
“06” маусым 2021 ж.

Алматы 2021

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

Программалық инженерия кафедрасы

БЕКІТЕМІН
ПИ кафедра меңгерушісі,
PhD доктор,
_____ М. Тұрдалыұлы
"_06_" _06_ 2021 ж.

Дипломдық жоба орындауға
ТАПСЫРМА

Білім алушыға *Мусілімов Дінмұхамед Бақытұлы*
Тақырыбы: Жеке тұлғаларға қызмет көрсетуге арналған IOS мобильді әзірлемесі
Университет ректоры бұйрығының №2131 б «24» 11 2021 ж. Шешімімен бекітілген.
Орындалған жобаның өткізу мерзімі «07» маусым 2021 ж.
Дипломдық жобаның бастапқы мәліметтері: *Жобаның төлқұжаты, технология бойынша техникалық құжаттама, техникалық құжаттаманы әзірлеу кезінде құрылған жобаның UML диаграммалары, техникалық тапсырма.*
Есеп – түсініктеме жазбаның талқылауға берілген сұрақтардың тізімі:
а) жалға беру және алу жүйелерін жалпы сипаттау;
б) жобаны модельдеу және UML диаграммаларын сызу;
в) мобильді қосымша шешімдерін жобалау;
г) тесттерден өткізу, кітапханалармен жұмыс жасау, бар жүйелермен салыстыру;
Графикалық материалдар тізімі (міндетті суреттердің нақты көрсетілуімен):
презентацияның 17 слайдпен берілген құжат түрінде ұсынылған.
Ұсынылған негізгі әдебиеттер: *10 пайдаланылған әдебиеттер тізімінен.*

Дипломдық жобаны орындау

КЕСТЕСІ

Бөлімдердің атаулары, зерттелген мәселелердің тізімі	**Ғылыми жетекшіге және кеңесшілерге ұсыну мерзімі**	**Ескерту**
1. Диплом жұмысының жоспарын құру.	22.01.2021	
2. Тапсырма қойылымы және бағдарламау ортасын таңдау.	01.02.2021	
3. Зерттеу тақырыбы бойынша ғылыми – теориялық материалдарды жинау және негізгі бөлім бойынша есеп беру жазбасын дайындау.	05.02.2021	
4. Жобалау бөліміне сызбаларды дайындау.	15.02.2021	
5. "Alu.kz" мобильді қосымшасын құру, тесттен өткізу.	01.03.2021	
6. Дипломдық жобаға түсініктеме жазба жазу.	10.04.2021	

Дипломдық жұмыс бөлімдерінің кеңесшілерінің аяқталған жұмысқа қойылған қойған **қолтаңбалары**

Бөлімдер атауы	Кеңес берушілер (аты-жөні, тегі, ғылыми дәрежесі, атағы)	Қолтаңба қойылған мерзімі	Қолы
Нормалық бақылаушы	Марғұлан Қ. Тех. ғыл. магистрі, лектор	06.06.2021	
Бағдарламалық бөлім	Рамазан А.Б. Тех. ғыл. магистрі, ассистент	31.05.2021	

Ғылыми жетекші _____ А.М. Кайрбеков

Тапсырманы орындауға кабылдап алған студент _____ Д.Б. Мүсілімов

Күні «06» маусым 2021 ж.

АҢДАТПА

Жобаның өзектілігі жеке тұлғаларға күнделікті қолданыстағы құралдарды немесе заттарды жалға беруге арналған мобильді қосымшаны жобалау және жасау болып табылады. Заманауи ақпараттық технология құралдары даму сатысында біршама белестерді бағындырғаны мәлім және сол құралдардың бірі ұялы телефон немесе басқа сөзбен – смартфон болып табылады. Республикамызда әлі күнге дейін тиянақты түрде қарастырылмаған жалға беру жүйесін, жоғарыда атап айтылған ұялы телефонға сай баптап, егжей-тегжейіне дейін жобалап, мобильді қосымша әзірлеу – осы жұмыстың басты мақсаты.

Бұл жүйенің жобадағы сипаттамасы кіріспеден, үш тараудан (Зерттеу, технологиялық, жобалау) тұратын негізгі бөлімнен, қорытындыдан, пайдаланылған әдебиеттер тізімінен тұрады.

Зерттеу бөлімінде жобаның өзектілігі және қажеттілігі қарастырылды, қолданыстағы жүйеге талдау жүргізілді. Әзірленетін технология және құралдар түрі анықталып, сипатталды.

Технологиялық бөлімде жобаны құрауға қолданылатын бағдарламалық қамтама таңдалынды, технологияларға сипаттау жүргізілді. Бағдарламалау ортасын баптау және әзірлеу барысы талданды.

Жобалау бөлімінде проектілеу, сонымен қоса, қолданылған әдістер сипатталды.

Жалпы алғанда дипломдық жұмыс 62 бет, 14 сурет, 2 қосымшадан тұрады. Жүзеге асыру барысын сипаттауға 10 әдебиеттер қолданылды.

АННОТАЦИЯ

Актуальность проекта заключается в проектировке и разработке мобильного приложения для сдачи в аренду инструментов или предметов повседневного пользования физическим лицам. Как известно, современные средства информационных технологий находятся на стадии развития и одним из таких инструментов является мобильный телефон или другими словами – смартфон. Основная цель этой работы – досконально спроектировать систему аренды, которая до сих пор полноценно не разработана в нашей республике, в соответствии с вышеизложенным мобильным телефоном.

Описание данной системы в проекте состоит из введения, основной части, состоящей из трех глав (исследовательских, технологических, проектных), заключения, списка использованной литературы.

В разделе исследования рассмотрены актуальность и необходимость проекта, проведен анализ существующей системы. Определены и описаны виды разрабатываемых технологий и инструментов.

В технологическом разделе выбрано программное обеспечение, используемое для составления проекта, проведено описание технологий. Проанализирован ход настройки и разработки среды программирования.

В проектной части описано проецирование, а также используемые методы.

Всего дипломная работа состоит из 62 страницы, 14 рисунков, 2 приложений. Для описания хода реализации использовано 10 источников.

ANNOTATION

The relevance of the project lies in the design and development of a mobile application for renting tools or items for daily use to individuals. As you know, modern means of information technology are at the stage of development and one of these tools is a mobile phone or in other words-a smartphone. The main goal of this work is to thoroughly design the rental system, which is still not fully developed in our republic, in accordance with the above mobile phone.

The description of this system in the project consists of an introduction, a main part consisting of three chapters (research, technological, project), a conclusion, and a list of references.

The research section examines the relevance and necessity of the project, and analyzes the existing system. The types of technologies and tools being developed are identified and described.

In the technology section, the software used for drawing up the project is selected, and the technologies are described. The course of setting up and developing the programming environment is analyzed.

The project part describes the projection, as well as the methods used.

In total, the thesis consists of 62 pages, 14 figures, and 2 appendices. 10 sources were used to describe the implementation process.

МАЗМҰНЫ

КІРІСПЕ

Қазіргі заман – бұл технологиялар мен ақпараттың даму кезеңі. Сол уақытта қазіргі заман мен технологияның, ақпарат таратуға арналған түрлі көздер мен құрылғылардың үздіксіз дамуы көптеген мүмкіндіктерге жол ашып отырғаны мәлім. Мұндай мүмкіндіктерде электронды компьютерлер ерекше орын алады. Бұл күнделікті өмірде, жұмыс орындарында үнемі қолданылатын компьютерлер, ноутбуктер. Осы құрылғылардың көмегімен біз әр түрлі бағдарламалау тілдерін қолдана отырып, күнделікті жұмысымызға қажетті қосымшаларды жасай аламыз. Өтініштерге сұраныстар саны да артып келеді. Адамзат қоршаған әлеуметтік ортамен тұрақты байланыстардан алатын және өңдейтін ақпараттың саны, мазмұны, көлемі мен құрылымы артып келеді. Өз кезегінде ақпараттық технологиялар да қарқынды дамуда, адамдардың көпшілігі Интернетке қол жеткізе алады. Көптеген мекемелер ішкі процестерді ұйымдастыру және басқару үшін ақпараттық жүйелерді енгізуде.

«Компьютерлер және бағдарламалық қамтамасыз ету» – ақпаратты іздеу, жинау, сақтау, түрлендіру және адам қызметінің әр түрлі салаларында қолдануға байланысты ақпараттық технологиялар мәселелерімен айналысады. Біріншіден, бұл жоғары технологиялар саласы. Ақпараттық технологияларды иемдену, қазіргі заманның нормасы деп айтуға болады.

Осы бағыт шеңберінде кәсіби бағдарламашылар – жүйелік және қолданбалы бағдарламалық жасақтама әзірлеушілері дайындалады.

Мамандық түлектеріне деген қажеттілік өте жоғары – көптеген жылдар бойы осы саланың мамандарына деген сұраныс ұсыныстан асып түсті. Түлектер жетекші IT-компанияларда жұмыс істейді.

Ал, мобильді қосымша туралы айтатын болсақ, мобильді қосымшаны әзірлеу – бұл PDA, смартфон немесе ұялы телефон сияқты шағын қол құрылғыларына арналған қосымшалар жасау процесі. Бұл қосымшалар өндіріс кезінде құрылғыға алдын ала орнатылуы, қолданушы бағдарламалық жасақтаманы тарату үшін әр түрлі платформаларды қолдана отырып жүктеуі немесе клиентте немесе сервер жағында өңделетін веб-қосымшалар болуы мүмкін.

Қосымшаны әзірлеуде Xcode – MacOS, iOS, watchOS және tvOS платформаларына арналған Apple компаниясы жасаған интегралды бағдарламалық жасақтама ортасы (IDE) және Swift программалау тілі таңдалынды. Xcode-тің бірінші нұсқасы 2003 жылы шыққан. Тұрақты нұсқалар Mac App Store арқылы ақысыз таратылады. Тіркелген әзірлеушілер Apple Developer сайты арқылы бета-жинақтарға қол жеткізе алады. Swift – бұл ашық

көзді, жалпыға арналған, Apple компаниясы құрған бағдарламалау тілі. Ол оқуға жеңілдеу және бағдарламашының қателіктеріне көңіл аударады, тез түзеуге көмек береді, оның алдындағы Objective-C-ге қарағанда сенімді болды. Swift бағдарламалары XCode 6 және одан жоғары IDE құрамына кіретін LLVM көмегімен құрастырылады. Swift Objective-C жұмыс уақытын қолдана алады, бұл екі тілді де (сонымен қатар C) бір бағдарлама аясында пайдалануға мүмкіндік береді. Ашықтығы мен қол жетімділігі, жетілдіруге арналған ақысыз және ыңғайлы құралдардың болуына байланысты олар үлкен сұранысқа ие, сондықтан осы технологиялар осы жобаны әзірлеу кезінде таңдалды.

Мобильді қосымшаның негізгі аудиториясы: Алматы қаласының тұрғындары, қолданыстағы құралды немесе затты ақылы жалға беру арқылы ақша табу жолын іздеген кез-келген жеке тұлға.

Дипломдық жұмыстың мақсаты: Күнделікті қолданыстағы құралдарды немесе заттарды жалға бергісі келетін жеке тұлғалардан өтініштерді электронды түрде қабылдап және бірыңғай жүйе арқылы оған баға тағайындап, хабарландыру парақшасына жариялау жүйесін жасап шығару.

1 Зерттеу бөлімі

1.1 Жобаның өзектілігі

Бұл жобаның жүйесін құрар алдында, жүйенің жеке тұлғалар арасындағы сұранысы қаншалықты қарқынды болатынын зерттеу керек болатын. Сондықтан, мен зерттеу жұмысының ең қарапайым түрін қолданып көргенім жөн деп ойладым. Менің қуанышыма орай, қазіргі заманауи әлеуметтік желілердің дамып тұрған шағында, қарапайым әлеуметтік сауалнама жүргізу үлкен пайдасын тигізді. Сауалнама бетінде бұл жобаның идеясын жалпылама бейнелеп, «Ия» немесе «Жоқ» деген жауаптар қойдық. Сауалнама қорытындысы:

1.1.1-сурет – Сауалнама сұрағы және жауаптар

– сауалнаманың жауаптарына қарай, көптеген адамдарда өздері қолданбайтын біршама жеке құралдарының бар екенін білдік. Келесі сұрағымыз олардың қаншауы сол заттарды пайдала отырып, ақша тапқысы келетінін анықтау жайында болатын.

1.1.2-сурет – Сауалнама сұрағы және жауаптар

– келесі сауалнамада, сол заттарды жалға беру арқылы ақша табу ойы 95,1% адамға қызық болғаны көрініп тұр.

1.1.3-сурет – Сауалнама сұрағы және жауаптар

– және соңғы сауалнама бойынша біз құрастырылатын жүйенің мобильді қосымшада құралғаны көпшілік қауымға ыңғайлырақ болатынын түсіндік.

1.2 Мобильді қосымшалардың маңызы

Бұл жобаның жүйесін зерттеу барысында, түрлі сұрақтар мен қиыншылықтар кездесті. Алайда, солардың ең бастысы, қазіргі қарқынды ақпараттық даму барысында, түрлі технологиялар мен ақпараттық құралдардың сан-алуан түрінен қайсысын таңдау тиімді, әрі дұрыс болатыны бізді қатты алаңдатты. Соңғы жүргізген сауалнама қорытындысы біздің сұраққа бағыт берді.

Технологияның дамуымен біздің өміріміз барлық мүмкін мобильді құрылғылармен толықты, олардың рөлін бүгінде бағалау қиын, өйткені ұялы телефонның көмегімен біз кез-келген уақытта достарымызбен, туыстарымызбен немесе жұмыс әріптестерімізбен байланысуға болады, ақпаратты жылдам табу немесе беру үшін. Телефонда контактілерден басқа көптеген басқа ақпараттар сақталады: олар өздерінің идеялары мен ойларын, несие карталарының нөмірлерін және басқа да жеке деректерді жазады. Заманауи ұялы телефондармен жабдықталған барлық мүмкін болатын пайдалы бағдарламалар, функциялар бұл шағын құрылғыны соншалықты көпфункционалды етеді, оларға таңданбау мүмкін емес.

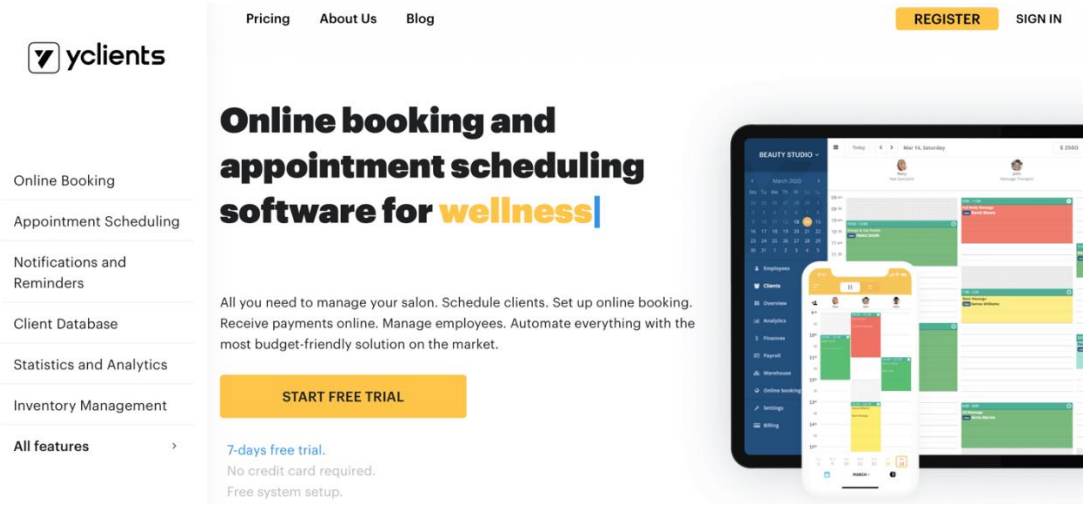
Мобильді қосымшалар өз кезегінде адамның күнделікті іс-әрекетін жеңілдетуге байланысты көптеген процестерді автоматтандыру арқылы заманауи ақпараттық технологияларды дамытуда үлкен рөл атқарады.

Кәсіпкерлік саласындағы адамдар үшін ұялы құрылғының пайдасы қажетті келіссөздер жүргізу, кез келген жерде және кез келген уақытта келісімге келу қабілетінде. Егер кәсіпкерлер кеңседен тыс жерде келісім жасай алмаса, бұйрық бере алмаса немесе басқа мәселелерді шеше алмаса, компаниялардың өнімділігі қаншалықты төмендейтіндігін елестету қиын емес.

1.3 Қолданыста бар жалға беру жүйесін зерттеу

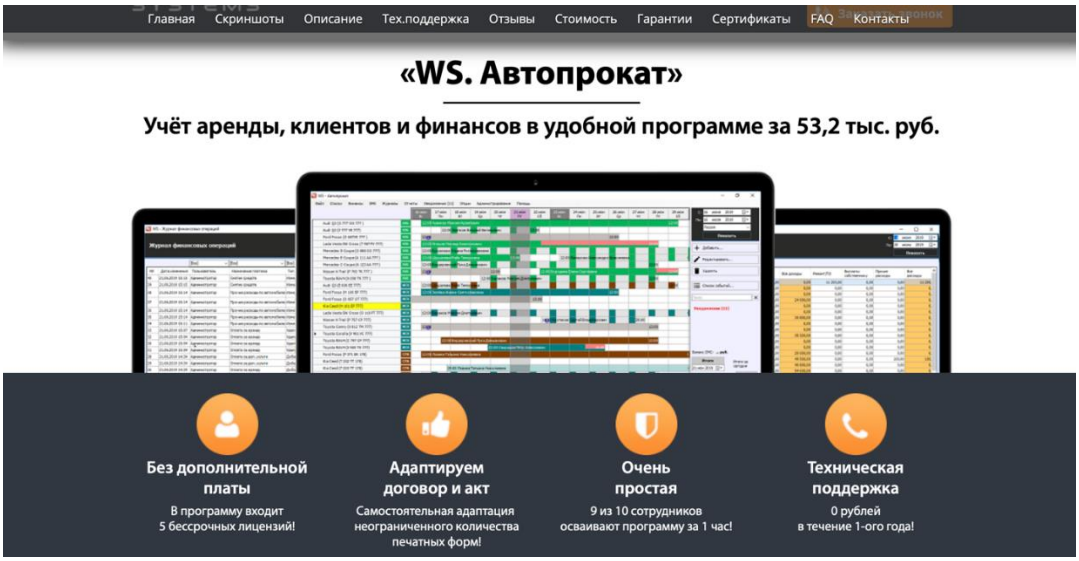
Менің ойымша, әр жобаның жетістігі оның бәсекеге қабілеттілігіне байланысты. Сондықтан, әр тұлға жобасының құрылуы барысында жүйесінің артықшылықтары мен кемшіліктеріне толықтай талдау жасап, мобильді қосымшалар мен түрлі жүйелер нарығындағы бәсекелестерді бақылауы қажет. Осы кеңеске орай, әртүрлі CRM – жүйелерін зерттей отыра, біршама бәсекелестердің тізімін құра алдым, олардың ішінде:

1. YCLIENTS – Интернетте брондау және қызмет көрсету саласын автоматтандыру CRM жүйесі. Кез-келген жалға беру жүйесін автоматтандыруға арналған жүйе. Басты артықшылығы: ыңғайлы қолданыс барысы мен қызмет көрсету саласындағы нақты уақыт бойынша жұмыс. Кемшілігі: ақылы нұсқасының қымбаттылығы.



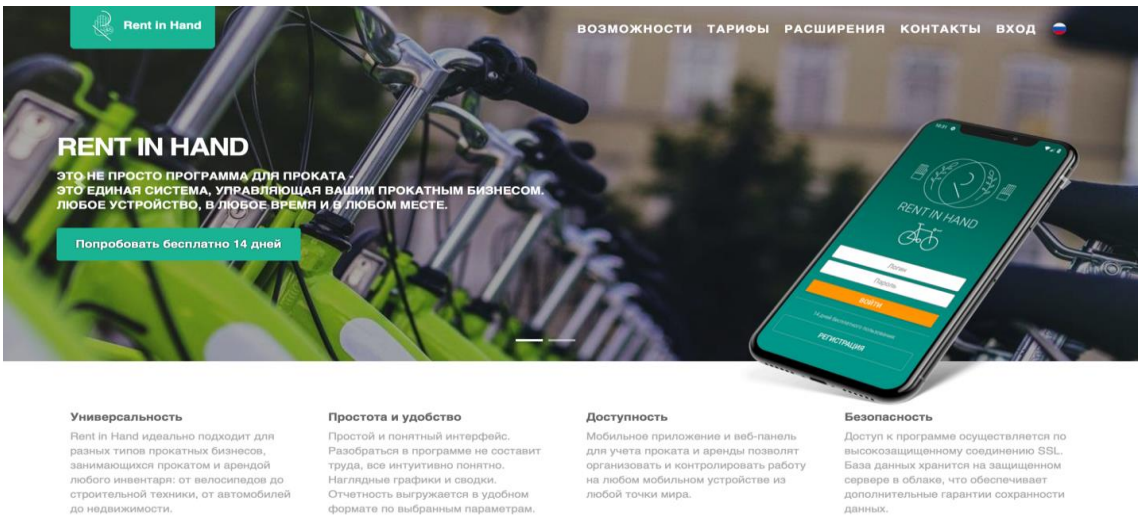
1.3.1-сурет – «YCLIENTS» CRM жүйесінің басты беті

2. WS. Автопрокат – Ыңғайлы бағдарламада жалдау төлемдерін, клиенттерді және қаржыны есепке алу. Ақысыз автокөлікті қарапайым және ыңғайлы көрнекі іздеу. Артықшылығы: автопарктің жұмыс кестесі «Шахмат» кестесімен орналасуы, келісімшартты басып шығару үшін сізге клиентті, автокөлікті және жалға алу мерзімін таңдау қажет, қалғаны бағдарлама бойынша жүзеге асырылады. Кемшілігі: тегін пайдалану мерзімі жоқ.



1.3.2-сурет – «WS. Автопрокат» жүйесінің басты беті

3. Rent in Hand – бұл бухгалтерлік есепті және жалдауды автоматтандыруға арналған қызмет. Жалға беру және жалға беру бизнесімен айналысатын автомобильдерді жалға алудың әртүрлі түрлері үшін өте қолайлы. Көбіне велосипед және моторлы кішігірім қозғалыс құралына арналған. Ерекшелігі: кішігірім қозғалыс құралына ыңғайлы. Кемшілігі: көптеген өзге құралдарға арналған ерекшеліктері жоқ.



1.3.3-сурет – «Rent in Hand» жүйесінің басты беті

4. PSoft. Негізгі PSoft бағдарламалық жасақтамасы автокөлік кәсіпорындарын автоматтандыруға арналған. Бұл бағдарламалық қамтамасыздандыру такси, көлік жалдау, ломбардтар, автомектептер, автопарктер мен такси компанияларын жіберуге арналған. Ерекшелігі: Өзіңізге ыңғайлы сайт немесе қосымша әзірлеуге арналған жүйе. Кемшілігі: ақылы нұсқасының қымбаттылығы.

1.3.4-сурет – «PSoft» жүйесінің басты беті

Жоғарыда айтылып өткен жүйелерді анализ жасай отыра, олардың артықшылықтары мен кемшіліктерін көре алдық. Бұл бізге, кейін, өзіміз құрастыратын жүйеде сан-алуан жайттарды қарастыра отырып, өзге жүйелерден ерекшеленуге мүмкіндік береді. Бұлардың барлығы көбіне автокөлік жүйесіне, қозғалыс құралдарына арналғанын байқадық. Алайда біз алға қойған мақсаттан әлі де, алыста жүрміз.

2 Технологиялық бөлім

2.1 Фреймворк ұғымына түсінік

Дипломдық жұмысын құру барысында соңғы жылдары күрделі бағдарламар жасап шығаруда қолданысқа ие болып жүрген Swift программалау тілі және Xcode таңдалынды. Аталған ұғымдар келесі бөлімшелерде жеке анықталатын болады.

Фреймворк – бұл қосымшаны құруды және техникалық жағынан күрделі немесе ауыр жобаларды пайдалануды жеңілдетуге арналған бағдарламалық өнім. Оның екі негізгі қызметі бар: клиент бөлімінде жұмыс (frontend) және сервер бөлімінде жұмыс (backend).[2]

Клиенттік шеңбер – бұл қолданушының оқуға, бейнелеуге және іске қосу мүмкіндіктерін анықтайтын қолданушыға көрінетін қосымшаның сыртқы бөлігі. Қазіргі уақытта пайдаланушыларға клиенттік жағын жүзеге асыратын көптеген құрылымдар ұсынылады, ең танымал: [4]

– Flutter фреймворкі;
– Ionic фреймфоркі;
– Xcode IDE және Swift, Objective-C программалау тілдері;
– React Native фреймворкі;
– Xamarin.

Серверлік шеңбер – қолданушының браузерінде немесе компьютерінде мүлдем көрсетілмеген қосымшаның логикасын іске асыратын, қосымшаның ішкі серверлік бөлігі үшін жауап беретін шеңбер. Бағдарламашылар арасында ең кең тарағаны:

– Node.js (Express); – Django;
– Rails;
– Laravel;
– Spring.

Қарастырылатын дипломдық жұмыстың жүйесінің құрылымының ажыратылуына бір функционалымен мысал келтірер болсақ: затын жалға бергісі келген жеке тұлға мобильді қосымшада белгілі бір форманы толтыру қажет, толтыру барысында керекті мәліметтер енгізеді және ол модераторға бірден жіберіледі. Мұндағы жарнама мәліметтер базасында сақталу және керекті тұлғаларға жіберілу процесінің қалай жүзеге асатыны қолданушыға белгісіз, процесс көрінбейді. Бұл функциялардың жүзеге асуын бэкенд бөлігі қамтамасыз етеді. Ал фронтенд бөліміне қолданушының жүйеге логин мен құпиясөз енгізуі, жарнама толтыру барысында керекті формаларға мәліметтер енгізуі, құрылған жарнаманың статусын көруі секілді қолданушы әрекеттері жатады. Іскерлік логикасы бар және жоғары жылдамдықты, сенімділікті және қауіпсіздікті талап ететін жобаларда фреймворктарды қолдану SaaS, CMS немесе CMF сияқты платформалар типіне қарағанда тиімдірек. [7]

Фреймерлерді қолданбай-ақ, бағдарлама жасаушы өзі қалаған кез-келген қосымшаны толығымен жаза алады. Бірақ жазбаша өтініш әртүрлі платформаларда көптеген қосымша кодтауды қажет етеді. Фреймворк көптеген дайын компоненттерді қамтитындықтан, олардың негізгі мақсаты – бағдарламалаушыға бағдарламаны оңай және қысқа мерзімде құруға мүмкіндік беру.

2.2 Клиенттік бөлім – IOS мобильді қосымшасы

Бағдарламашы мамандығы – ең беделді және сұранысқа ие мамандық, бұл факт. Бағдарламалық жасақтама жасау индустриясының өзіндік орны бар. Олардың ішінде ең жылдам өсетін және сонымен бірге ең көп төленетіні – мобильді құрылғыларға арналған бағдарламалық жасақтама. Өздеріңіз білетіндей, бәсекелестерді артта қалдырып, үш тұрақты қарсылас көш бастады. Микросхема осы үшеудің біреуі сөзсіз көшбасшы болатындай етіп орналасады, біреу әрқашан қуып жетеді.

Мобильді технологиялар сегментінде Apple iPhone-мен көшбасшы екені сөзсіз. Қыркүйек айында өткен презентацияда Cupertino компаниясы мобильді құрылғының 11-ші нұсқасын көрсетіп үлгерді. Онымен бірге Apple мобильді операциялық жүйесінің iOS 14 жаңа нұсқасын ұсынды. Қазір бұл алдыңғы жүйелер сияқты әлемдегі ең маңызды операциялық жүйе. Бұдан шығатыны, iOS дамуын үйрену уақыт пен ақшаның ең тиімді инвестициясы болып табылады.

Бүгін біз iOS үшін қарапайым мобильді қосымшаны жасамақпыз. Әрине, бұған жергілікті Mac ең қолайлы. Егер сіз Windows қолдаушысы болсаңыз немесе жай Mac болмаса, сіз macOS амалдық жүйесін Windows жүйесінде виртуалды машинаның астына орната аласыз. Интернетте сізге MacOS-тың компьютер үшін арнайы қайралған, халық арасында хакинтош деп аталатын нұсқасын табу қиын болмайды. Оны VMware виртуалды машинасына, VirtualBox-қа қоюға қымсынбаңыз – өз қалауыңыз бойынша.

Жалпы және әсіресе iOS үшін бағдарламалау үшін сізге көп нәрсе білу керек. Математика мен логика басында қажет болмауы мүмкін, бірақ кейінірек сұранысқа ие болады. Заманауи технологиялар бағдарламашыны компьютердің архитектурасын жетік білу қажеттілігінен босатты, бірақ санау жүйелері, олардың түрлендірулері, ішкі бағдарламалардың жылдамдығы немесе алгоритмдердің тиімділігі (үлкен О) сияқты негізгі механизмдерді түсіну қажет.

Жоғары деңгейде iOS жасаушыға macOS амалдық жүйесі мен iOS өзі туралы терең білім қажет. Сонымен қатар, сізге міндетті түрде «алма» бағдарламалау тілін меңгеру қажет. Delphi, C ++, C # немесе VB.NET-ті білу сізге көп көмектеспейді. Apple-дің өзінің экожүйесі бар, оның тілдері бар: Objective-C және Swift. Әрине, C ++ туралы білім әлі ешкімге зиянын тигізбеді, ондағы бағдарламалар мен кірістірулер барлық жерде, тіпті Apple-де бар. Бірақ анықтама бойынша ең көп қолданылатыны – «Objective-C», өйткені ол әлдеқайда

ертерек пайда болды (өткен ғасырдың сексенінші жылдарының ортасында), ал Swift 7 жыл бұрын ғана (2014 жылы). Apple компаниясы үлкен үміт артып, жаңа тілге көп қаражат салуда. Objective-C ескі бағдарламалық жасақтаманы қолдау үшін қолданылады, ал жаңасы Swift-те жазылған. Сондықтан сізге екеуін де білу жақсы. [1]

2.3 Xcode және MacOS

MacOS (бастапқыда Mac OS X ретінде енгізілген, 2012 жылы OS X деп өзгертілген, 2016 жылы macOS деп өзгертілген) – бұл Apple компаниясы жасаған меншікті операциялық жүйе.

MacOS жүйесінде Mach микро ядросына негізделген және Apple кодынан, NeXTSTEP және FreeBSD кодтарынан тұратын XNU ядросы қолданылады. 10.3 нұсқасына дейін ОЖ тек PowerPC процессоры бар компьютерлерде жұмыс істеді. 10.4 және 10.5 шығарылымдары PowerPC және Intel процессорларына қолдау көрсетеді. 10.6-дан бастап, macOS тек Intel және Apple Silicon процессорларымен жұмыс істейді.

Соңғы жылдары macOS пен iOS – Apple мобильді құрылғыларына арналған операциялық жүйе (iPhone, iPad және iPod touch) өзара интеграциялануда. Компанияның өзі екі операциялық жүйені бірыңғай платформа ретінде қарастырады.

Өздеріңіз білетіндей, macOS және онымен бірге iOS-тың негізгі құралы – бұл Xcode бағдарламалау ортасы. Оған Mac, iPhone, iPad, Apple TV, Apple Watch қосымшаларын жасауға арналған құралдар кіреді. Барлық Apple платформалары. Xcode кодында интерфейс құрастырушы бар, ол барлық заманауи кодтау құралдарын қолдайды. Сонымен қатар, Xcode-ден шықпай-ақ, сіз қосымшаны тексере аласыз; егер ол сыртқы құрылғыға арналған болса, оны эмуляторда іске қосуға болады.

Жүйеге барлық құрылғылардың эмуляторлары кіреді, олардың жаңа нұсқаларын жүктеуге болады. Сонымен қатар, Xcode бағдарламалық жасақтаманың өнімділігін талдауға арналған графикалық құралдарды, соның ішінде процессордың пайдаланылуын диагностикалау құралдарын, сақтауды пайдалануды (HDD, SDD және басқалары), графикалық адаптердің жүктелуін (OpenGL жағынан) қамтиды. [6]

13 қыркүйекте 8.0 даму ортасының жаңа, ыстық ықыласпен күтілген нұсқасы шықты. Тұрақты нұсқаларын App Store дүкенінен тегін жүктеуге болады. Альфа және бета-нұсқа әзірлеушінің жазылымы бойынша таратылады. Сегізінші нұсқаға мыналар кіреді: Swift 3 тілінің жаңа нұсқасы, iPad үшін Swift Playground оқу құралы, Interface Builder жаңа нұсқасы, ол тезірек болды және бағдарламаның орналасуын әртүрлі платформаларда көруге мүмкіндік береді, тренажерге қолданудың өзі.

2.4 Swift программалау тілі

2014 жылы Apple жаңа бағдарламалау тілін Swift-ті енгізді. Бұл тез әлемдегі ең көп айтылатын және тез дамып келе жатқан тілге айналды. Бұл Apple құрылғыларының барлық парағында macOS, iOS, tvOS және watchOS операциялық жүйелеріне арналған қосымшаларды жасауға арналған сенімді, қауіпсіз, интуитивті тіл. Swift – тиімді бағдарламалау тілі. Оларды оқудың мақсаты, Objective-C-ге қарағанда оңай, бірақ сонымен бірге Swift одан ең жақсы идеяларды алады. Сонымен бірге, әзірлеушілер тілдерді үйлесімді етті, яғни бір бағдарламада Swift және Objective-C кодтары болуы мүмкін. Әлемде Objective-C-де жазылған миллиардтаған жолдар мен жүздеген мың бұрынғы бағдарламалар бар, сондықтан оны қолдаудан бас тартпайды.

Swift-ті дамытуда мақсат ең жылдам, қауіпсіз және мәнерлі жүйелік бағдарламалау тілін құру болды.

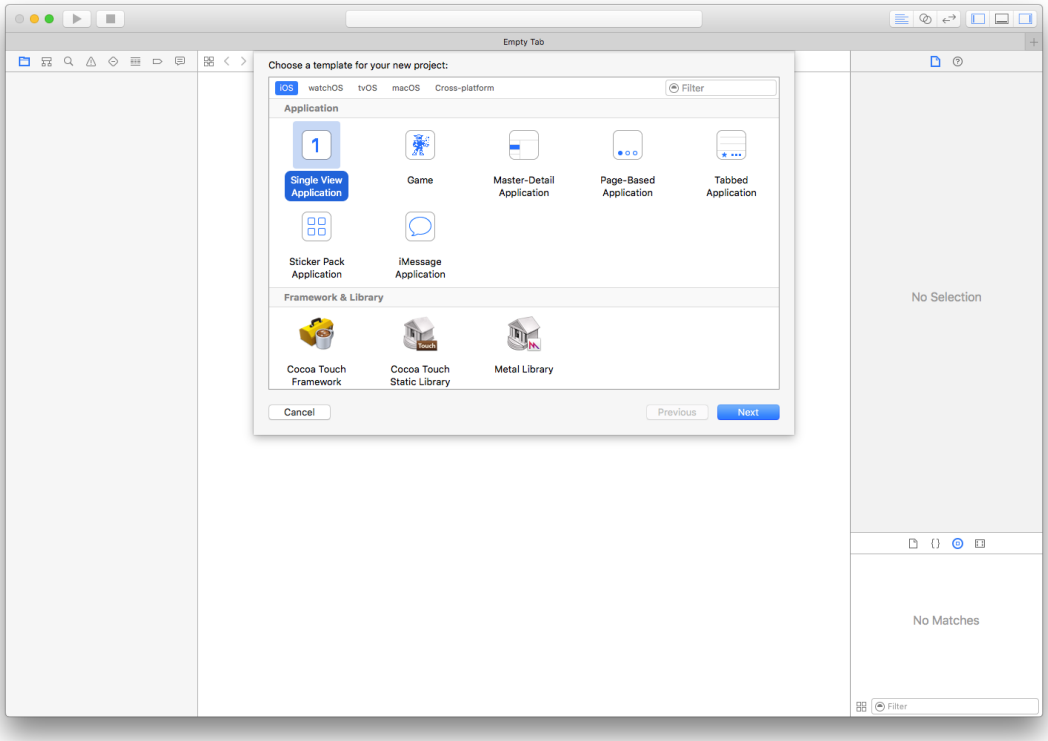
Swift-тағы алгоритм Objective-C-тағы бірдей алгоритмден 2,6 есе, ал Python-дан 8,4 есе жылдам. Swift қолдану бағдарламаларды жасау процесін икемді және ыңғайлы етеді. [3]

Экспоненциалды дамуына байланысты, 2015 жылдың 3 желтоқсанында Swift тілі ашық кодты қоғамдастыққа берілді. Сонымен бірге, Apple оның дамуын қатаң қадағалайды, оны дамыту комитетін ұйымдастырады. Swift енді Apple жүйелерінде ғана емес, Linux жүйесінде де қол жетімді. Тілмен жұмыс істеуге арналған барлық қосымша құралдар, соның ішінде түзеткіш, стандартты кітапхана, бума менеджері ақысыз және ашық қайнар көз болып табылады.

2.5 Программалау құрылымы және жұмыс жасау принципі

Шаблон ретінде біз өте қарапайым қосымша жасадық. Қасиетті дәстүрден таймай, ғаламға сәлем берейік. Алайда, бізде графикалық қосымшасы болғандықтан, оны әртараптандырамыз: оны басу арқылы батырманы қосамыз, «әлемге сәлем» деген жазуды көрсетеміз немесе жасырамыз.

Пайда болған терезеде Xcode іске қосыңыз, жаңа жоба жасауды таңдаңыз немесе негізгі мәзірден «Файл -> жаңа -> жоба» таңдаңыз. Келесі терезе сізден мақсатты платформаны және бағдарлама түрін таңдауды сұрайды.



2.5.1-сурет – «Xcode» программасында жаңа проект бастау беті

Бұл жағдайда бізді iOS платформасы қызықтырады. Қолданбалардың жеті түрі бар. Олардың алтауы – стандартты iOS қосымшалары, олар әдепкі бойынша басқа компоненттер жиынтығын қамтиды. Жетінші түрі – ойын. Қосымша түрлері:

– Single View қосымшасы қарапайым бір экрандық бағдарлама ретінде жасалған. Бос орын интерфейс құрастырушысы формасының дизайнері көмегімен қосымшаның сыртқы түрін реттеуге мүмкіндік беретін View Controller компонентін қамтиды;

– Master Detail Қолданба кесте көрінісінде объектілер жиынтығын көрсететін қосымшаны жасайды. Олардың біреуін таңдағаннан кейін осы объект туралы толық ақпарат көрсетіледі. Бірінші түрі – шебер, екіншісі – бөлшектер;

– Page-Based негізделген қосымшаның көмегімен сіз кітаптың парақтары сияқты бірнеше экраны бар қосымшалар жасайсыз. Сондықтан, мысалы, оқырмандар осы шығармадан жасалған;

– Tabbed application қосымшасы кез-келген уақытта әр экранға жылжытуға болатын қосымшаларды құруға мүмкіндік береді, демек, әр экранда тақырып көрсетілетін жерде оны іске қосатын өз батырмасы болады. ITunes – мысал;

– Game – ойын бланкісін жасау үшін қолданылады. Ойынды құру үшін төрт шеңбер ұсынылады: SpriteKit, SceneKit, OpenGL ES, Metal. Оларды жылдам қарастырайық;

– SpriteKit – бұл құрылымды Sprite тікбұрыштарына арналған екі өлшемді визуализация және анимация жүйесі. Фреймдер көрсетілген кезде стандартты

цикл қолданылады, кадр көріністегі барлық мазмұн көрсетілгеннен кейін көрсетіледі;

– SceneKit – бұл 3D графикасын OpenGL жоқ көрсетуге арналған жоғары деңгейлі Framework. Ол үш өлшемді объектілерді жүктеуді, манипуляцияны қолдайды. Оған физикалық қозғалтқыш, бөлшектер генераторы және қарапайым сценарий әдісі кіреді;

– OpenGL ES – бұл компьютерлік графика саласындағы стандарт. 2D және 3D көріністерін көруге мүмкіндік береді. Бұл бейне картаға арналған сызықты сипаттауға мүмкіндік береді: шыңдар екі өлшемді кескінмен расталып, экранда көрсетілетін примитивтерге айналады, өзгертіледі. Pipline бағдарламаланатын көлеңкелерді қосуға болады;

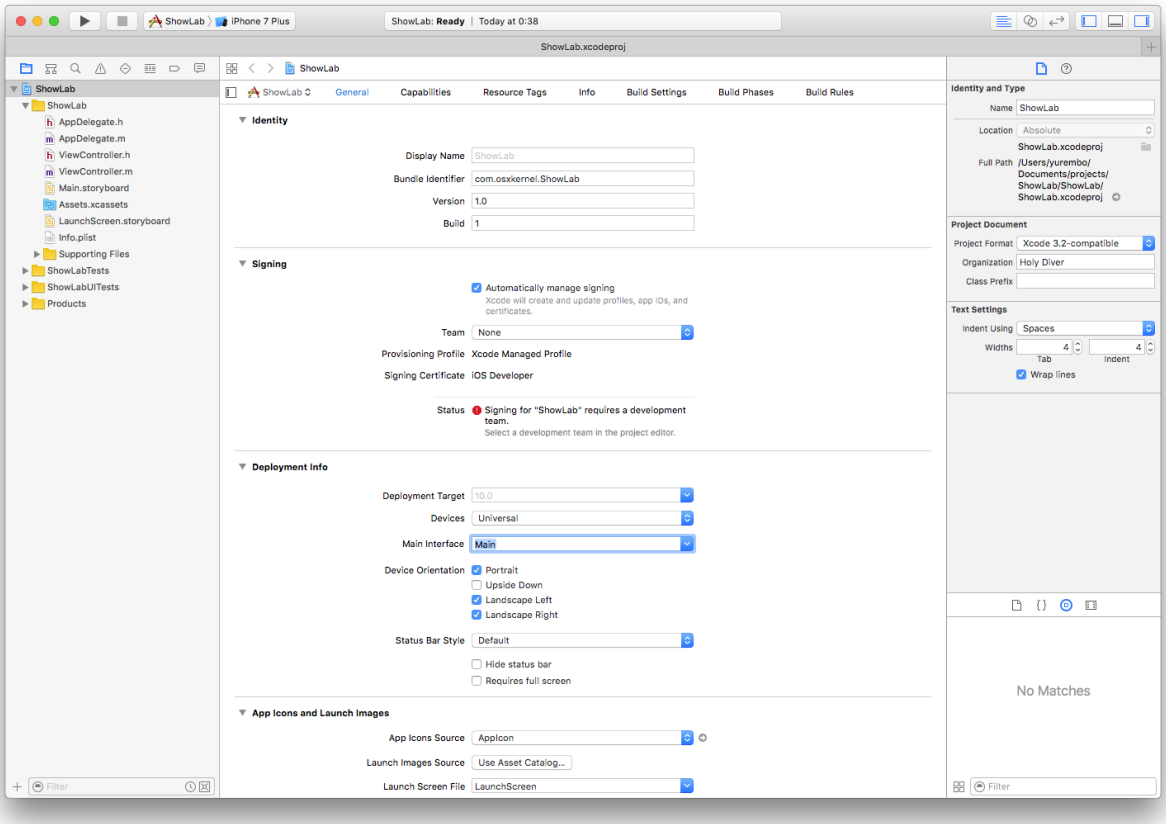
– Metal – бұл бейне адаптердің толық қуатын алуға мүмкіндік беретін төменгі деңгейлі API. Алдын ала салынған көлеңкелі және үлкен ағыны бар теңшелетін API сіздің ойыныңызға өнімділік пен сапаның жаңа деңгейіне шығуға мүмкіндік береді; [4]

– Sticker Pack қосымшасы – бұл iOS 10 және Xcode 8-де пайда болатын қосымшаның жаңа түрі. Бұл жаңа iMessage-де қолданылатын қарапайым немесе анимациялық суреттер жиынтығы. Оны құру үшін кодтау қажет емес;

– iMessage қосымшасы – бұл iOS 10 және Xcode 8-де пайда болатын қосымшаның жаңа түрі, ол iMessage қондырмаларын жасауға мүмкіндік береді, мысалы, стикерлер бумасын сатып алу және жүктеу. Сіз iMessage API көмегімен аудио, бейнені ойнату, стикерлерді пайдалану және басқаларын қоса, осы бағдарламаның аналогын жасай аласыз.

Қосымшаға шаблон ретінде «бір көрініс» қосымшасын таңдайық. Біз үлкен бағдарламаны жасамайтындықтан, бұл тренингтің материалы жеткілікті болады. Келесі түймешігін басыңыз. Шебердің келесі бетіне жобаның атауын енгізіңіз, мысалы, «ShowLab». Тіл ашылмалы тізімінде әдепкі Objective-C тілін қалдырыңыз. содан кейін құрылғының ашылмалы тізімінде «Universal» таңдауын қалдырыңыз. Бұл қосымшаның қай құрылғыда (iPhone немесе iPad) жасалатынын анықтайды. «Universal» сөзі екі мағынаны білдіреді. Біз «Include Unit Tests» және «Include UI Tests» ұяшықтарын өшіреміз, бізге тест қажет емес. Содан кейін. Жобаны сақтау үшін қалтаны таңдайық. Енді «Create» батырмасын басыңыз.

Бұл жобаға қолданылатын барлық нұсқалардың тізімі бар терезені ашады. Бұл терезеде сіз Шеберде бұрын конфигурацияланған параметрлерді өзгерте аласыз: бағдар, мақсатты құрылғы және т.б.



2.5.2-сурет – «Xcode» программасында жаңа проект баптаулары беті

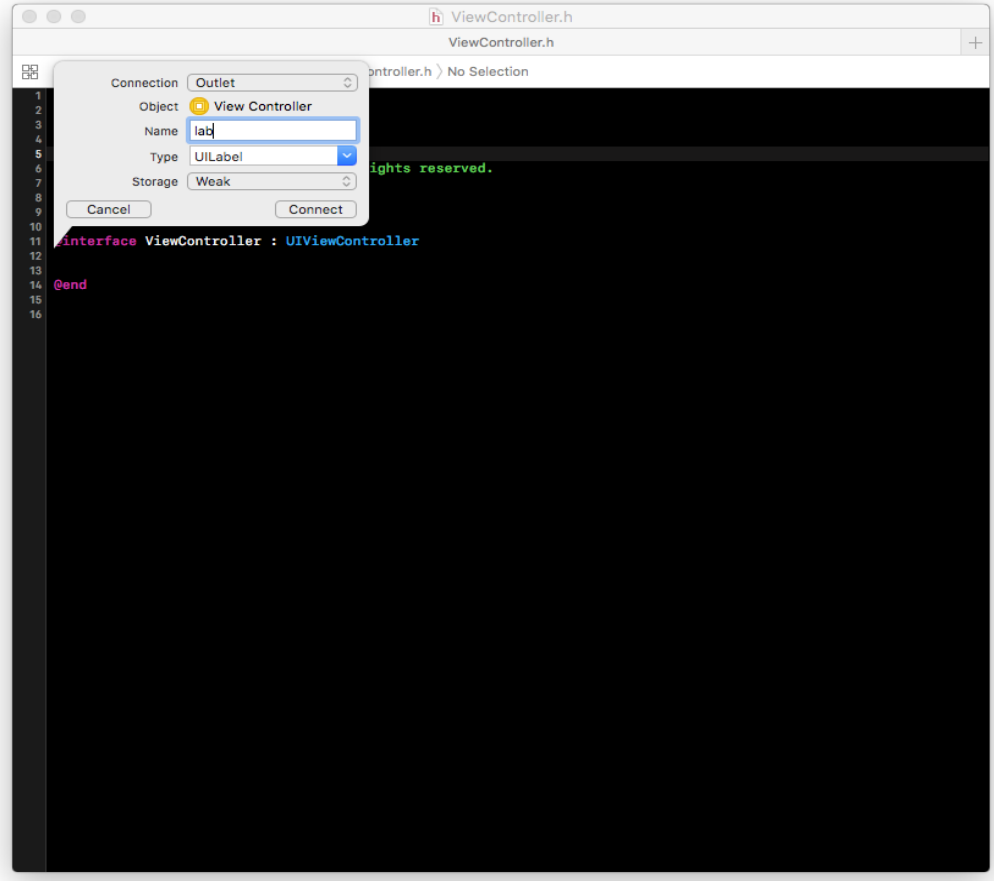
Біріншіден, біз қосымша интерфейсін құруымыз керек. Мұны істеу үшін «Settings» батырмасын басыңыз. Сол жақтағы тізімнен «Main.storyboard» файлын таңдаңыз (егер бұл файл көрінбейтін болса, «ShowLab» қалтасының мазмұнын кеңейтіңіз). Тізімнің оң жағында барлық Терезе «Interface Builder» орналастырылған. Орталықта құрылғының орналасқан жері көрсетіледі. Терезенің төменгі оң жақ бұрышында компоненттер панелі орналасқан. Біз «Label» және «Button» компоненттерін сол жерден сүйреп апарамыз. компонент панелінің жоғарғы жағында қасиеттер тізімі бар, егер ол жоқ болса, интерфейстің оң жағындағы терезе тақырыбының астында орналасқан «Атрибут инспекторын көрсету» батырмасын басамыз.

Дизайнда біз «Label» компонентін таңдап, оның «Text» қасиетін орнатамыз: ашылмалы тізімде «Plain» опциясын қалдырамыз, төменгі жолда біз қажетті белгіні енгіземіз, біздің жағдайда «Hello, World». Егер мәтін өрістің шеттерімен сәйкес келмесе, біз оларды түймені компоненттің шеттеріне апару арқылы өзгертеміз. Оны көлденеңінен орталықтандыру үшін біз өлшем инспекторы бетіне «Size Inspector» батырмасын басу арқылы барамыз (атрибуттар инспекторы экранының оң жағында). Бұл бетте «Arrange» ашылмалы тізімінде «Center Horizontally in Container» таңдаңыз.

Енді біз «Button» компонентін таңдаймыз, біз оның «Text» қасиетін қажетті белгіге ауыстырамыз: «Switch». Біз оны жоғарыда сипатталғандай орталықтандыруға болады. [5]

Visual Studio-да (немесе Delphi-де) визуалды компонент формаға енгізілген сәтте сіздің кодыңыздағы объект автоматты түрде жасалады. Бұл Xcode-де болмайды, бірақ бұл проблема емес.

«ViewController.h» тақырыптық файлының мазмұнын екі рет басу арқылы бөлек терезеде ашыңыз. Бұл файлда «@interface» кілт сөзімен белгіленген «uiviewcontroller» сыныпты кеңейту туралы декларация бар. Бұл функция Objective-C екінші нұсқасында қосылды. енді біз келесі әрекеттерді орындаймыз: тінтуір меңзерін компоненттің үстіне жылжытыңыз: мәтіндік жапсырма, Ctrl пернесін және тышқанның сол жақ батырмасын басыңыз. Меңзерді коды бар терезеге жылжытыңыз (ViewController.h файлы), курсор артында көк сызық шығады. Біз тышқанды және кілтті «ViewController» интерфейсінің сипаттамасында жібереміз. «Outlet» құру терезесі пайда болады.



2.5.3-сурет – «Outlet» құру терезесі

Бұл басқа объектіге сілтеме жасайтын объектілік қасиет (бұл жағдайда Visual компоненті). Біз зертханалық болып табылатын Visual компонентіне кіретін Outlet объектісінің атауын енгізуіміз керек. Содан кейін сәтті таңдалған нысан түрі таңдалды: UILabel.

Сақтау тізімінде объектінің одан да төмен сілтеме түрі таңдалады: әлсіз немесе күшті. Егер сіз күш таңдайтын болсаңыз, қасиетті көрсететін объект қасиет оны көрсеткен уақытта болады, бұл жағдайда оны қолданбаған кезде оны

автоматты түрде жою мүмкін емес. Екінші жағынан, әлсіз сілтеме күшіне енген кезде, объект өздігінен жойылуы мүмкін. Сондықтан әлсіз сілтеме түрін таңдап, «Қосылу» батырмасын басыңыз. Нәтижесінде кодқа келесі жол қосылады:

*@property (weak, nonatomic) IBOutlet UILabel *lab;*

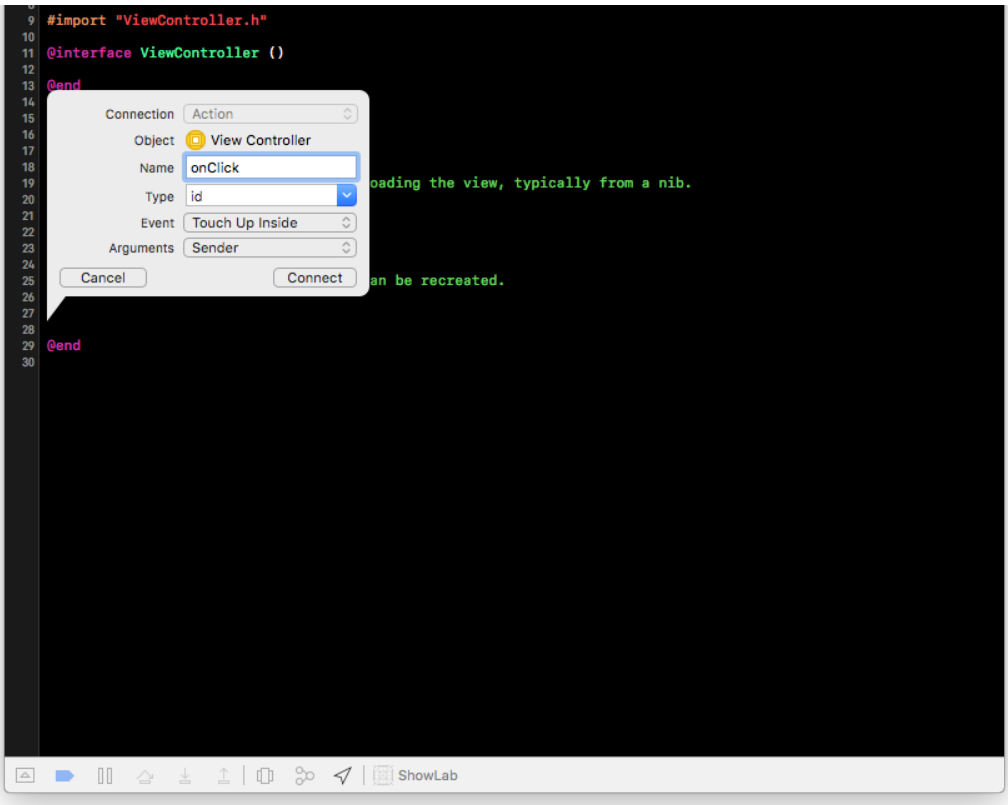
Outlet-тің меншік екеніне көз жеткіземіз.

Енді батырма үшін розетка жасайық. Алгоритм өзгеріссіз қалады. Тек «Name» қасиеті үшін басқа атау енгізу керек, мысалы, «but». Жол кодқа қосылады:

*@property (weak, nonatomic) IBOutlet UIButton *but;*

Нәтижесінде бізде визуалды компоненттер үшін екі индикатор бар: «lab» және «but», сәйкесінше, жарлық пен батырма. Енді көрсеткіштерді қолдана отырып, біз кодтағы компоненттермен жұмыс істей аламыз.

Әрі қарай, батырманы басу оқиғасы үшін өңдеуші жасау керек. Ол үшін біз «ViewController.m» іске асыру файлын бөлек терезеде ашамыз. «Outlet» құру үшін жолды тақырыптық файлға сүйреп апарып, батырмадан бастап, жолды іске асыру файлына апарып, орналастырамыз. ол жақшаның жабылу командасы арасында – «@end». Ұяшық құру терезесіне ұқсас оқиға жасайтын терезе пайда болады. Айырмашылықты көресіз: тақырыптық файлда объектке сілтеме жасалады және әдіс жасалады іске асыру файлы [6].



2.5.4-сурет – Оқиға жасау терезесі

Мәні әдіс қасиетінің атын білдіретін атау өрісін толтырыңыз. Оны «OnClick» жасаңыз. Әдепкі бойынша, «Type» өрісіне біз «id» мәнін қалдырамыз. Objective-C-де бұл тип басқалардың ата-анасы болып табылады. Әдепкі оқиға ашылмалы тізімінде курсорды (тышқанды) батырманың үстінен жібергенде пайда болатын «Touch Up Inside» оқиғасы таңдалады, яғни батырманы басудың соңғы қадамы. Бұл бізге қажет. Аргументтер тізімінде біз әдепкі мәнді қалдырамыз: жіберуші – бұл сигнал жіберген объект, біздің жағдайда бұл әрқашан батырма. Қосылу түймесін басыңыз. Нәтижесінде келесі код қосылады:

```
- (IBAction)onClick:(id)sender {
}
```

Басынан аз нәрсе жеке әдісті білдіреді. Interface Builder визуалды компоненттерінің оқиғалары (әдістері) IBAction кілт сөзімен көрсетілген.

Біз батырманы басқан кезде жұмыс істейтін кодты жақшаға жазамыз:

```
_lab.hidden = !_lab.hidden;
```

Бұл код жолында біз жасырын сипаттың мәнін аударамыз. Оның екі мәні бар. BOOL типті: ИӘ – ақиқат және ЖОҚ – жалған (Windows бағдарламашылары үшін біршама ерекше, мұнда ақиқат және жалған).

Нысан – белгілер деп аталатын сценарийге назар аударыңыз (_lab). Онсыз компиляция сәтсіздікке ұшырайды. Жүктеу автоматты түрде басқа объектілердің объектілеріне қосылады; яғни, бұл жағдайда зертханалық объект «ViewController» түрінде болады. Бұл сынып мүшелері жариялаған нысандар мен жергілікті объектілерді шектеу үшін қолданылатын шартты ереже болды. Қазір бұл тіл компиляторында жүзеге асырылатын қатаң ереже.

Енді біз қосымшаны құрастырып, тренажерде іске қосамыз. Біз «Xcode»-ге интеграцияланған «iPhone 7» тренажерін таңдадық. Компиляция және орындау батырмасы интерфейстің жоғарғы жағында орналасқан қара төртбұрыш.

Қосымшаны құрастырып, тренажерді іске қосып, біздің қосымшаны жүктегеннен кейін оның интерфейсі тренажер экранында көрсетіледі: «сәлем, әлем» жазуы және «Switch» батырмасы. Соңғы жазбаны басу жоғалады және оны басу қайтадан пайда болады.[1]

2.6 Ағындарды басқару

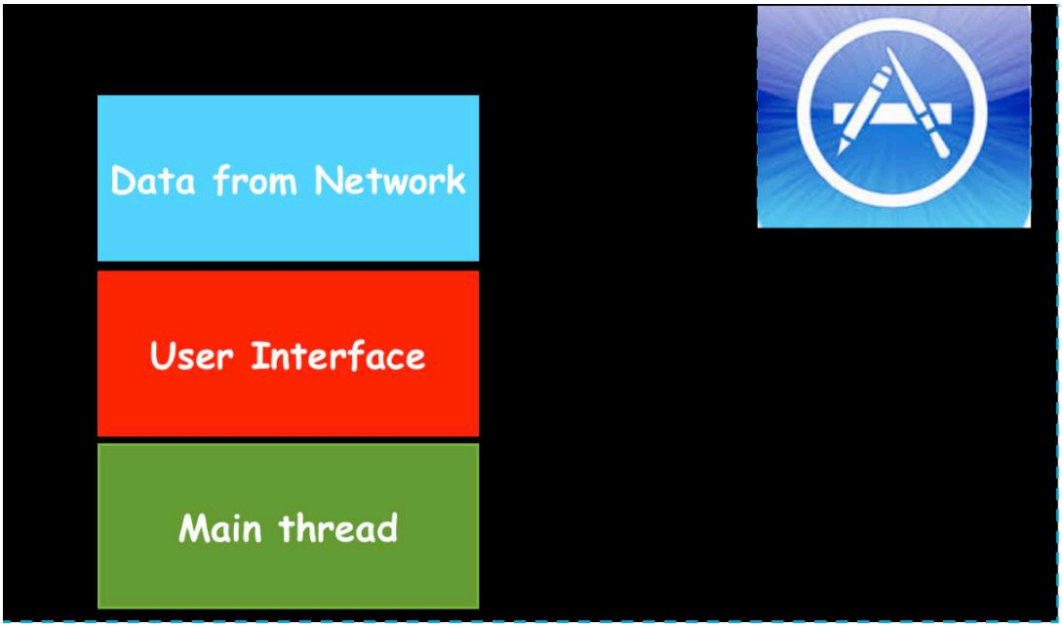
Swift қазіргі уақытта кез-келген жергілікті көп жұмыс функциясын қамтымайды.

Мысалы, Go-де сіз жеке ағынмен жұмыс істейтін функцияны оның алдына «go» кілт сөзімен жібере аласыз. Swift-те бұл әлі жоқ, сондықтан егер сіз өзіңіздің

өтініміңіздегі кез-келген тапсырманы параллельдеуді жоспарласаңыз және бір уақытта жарыс жағдайына және көпжоспарлы әлемнің басқа қиыншылықтарына тап болмасаңыз, сізде екі жол бар. Сыртқы кітапханаларды пайдаланыңыз, мысалы libDispatch (aka GCD) немесе Foundation немесе OS ұсынған синхрондау примитивтері [synchronization primitives] (мутекс, семафор, құлыптар және т.б.).

IOS-тағы көп жұмыс (concurrency) әрдайым iOS қосымшаларын жасаушылармен сұхбат кезінде қойылған сұрақтарға, сондай-ақ iOS қосымшаларын әзірлеу кезінде бағдарламашылар жіберетін қателіктер қатарына қосылады деп айтуға болады. Сондықтан бұл құралды жетік меңгеру өте маңызды.

Сонымен, сізде қосымша бар, ол сіздің қолданушы интерфейсін (UI) көрсететін кодты орындауға жауап беретін негізгі ағынмен жұмыс істейді. Қосымшаңызға желіден деректерді жүктеу немесе негізгі ағынға (негізгі ағын) кескіндерді өңдеу сияқты «уақытты қажет ететін» код бөліктерін қосуды бастағаннан кейін, сіздің интерфейсіңіз баяулай бастайды және мүмкін тіпті оның толық «қатуына» әкеледі.



2.6.1-сурет – IOS қосымшасының ағындары

Бұл проблемалар туындамас үшін қолданбаның архитектурасын қалай өзгертуге болады? Бұл жағдайда көпжоспарлау (concurrency) көмекке келеді, бұл бір уақытта екі немесе одан да көп тәуелсіз тапсырмаларды (tasks) орындауға мүмкіндік береді: есептеулер, желіден немесе дискіден мәліметтерді жүктеу, кескіндерді өңдеу және т.б.

Процессор уақыттың кез-келген сәтінде сіздің тапсырмаңыздың бірін орындай алады және оған сәйкес ағын бөлінеді.

Бір ядролы процессор (iPhone және iPad) жағдайында параллельділікке бірнеше ядролы процессордағы тапсырмаларды бір уақытта орындаудың сенімді көрінісін жасай отырып, тапсырмалар орындалатын ағындар арасында бірнеше

рет ауысу арқылы қол жеткізіледі. Көп ядролы процессорда (Mac) көп ағынды тапсырмаға байланысты әрбір «жіпке» тапсырмаларды орындау үшін өз ядросын беру арқылы қол жеткізіледі. Бұл екі технологияда параллельдік туралы жалпы түсінік қолданылады.

Сіздің қосымшаңызға көп сызықты енгізу үшін төленетін ақы – бұл жіптің қауіпсіздігін қамтамасыз етудің қиындығы. Тапсырмаларды қатар жүргізуге мүмкіндік бере салысымен, әр түрлі тапсырмалар (tasks) бірдей ресурстарға қол жеткізгісі келетіндігіне байланысты мәселелер туындайды, мысалы, олар бір айнымалыны әртүрлі тізбектерде өзгерткісі келеді немесе өздері қалайды басқа тапсырмалармен бұғатталған ресурстарға қол жеткізу. Бұл басқа ағындардағы тапсырмалар пайдаланатын ресурстарды жоя алады.

IOS бағдарламалауында көпжоспарлау бірнеше құралдар түрінде – Thread, Grand Central Dispatch (GCD ретінде қысқартылған) және Operation түрінде ұсынылады және пайдаланушы интерфейсінің өнімділігі мен жауаптылығын арттыру үшін қолданылады. Біз бұл тақырыпты қозғамаймыз, өйткені бұл төменгі деңгейдегі механизм, бірақ осы дипломдық жұмыста GCD және кейінгі жарияланымдағы Operation (GCD-дің үстіне салынған объектілі-бағытталған API) туралы тоқталамыз.[9]

Swift 3 пайда болғанға дейін Grand Central Dispatch (GCD) сияқты қуатты құрылымда C тіліне негізделген API болған, ол бір қарағанда жай сиқыр кітабы сияқты көрінеді және бұл бірден емес пайдаланушының пайдалы тапсырмаларын орындау үшін оның мүмкіндіктерін қалай жұмылдыру керектігін түсіндіру.

Swift 3-те бәрі күрт өзгерді. GCD-де жаңа, толығымен Swift-ке ұқсас синтаксис бар, оны қолдану өте оңай. Егер сіз ескі GCD API-мен аздап таныс болсаңыз, онда барлық жаңа синтаксис сізге оңай серуендейтін болады; егер олай болмаса, онда сіз iOS бағдарламалауының тағы бір жалпы бөлімін үйренуіңіз керек. Жаңа GCD құрылымы Swift 3-те барлық Apple құрылғыларында жұмыс істейді, Apple Watch-тан бастап, барлық iOS құрылғыларын қоса, Apple TV мен Mac-қа дейін.[7]

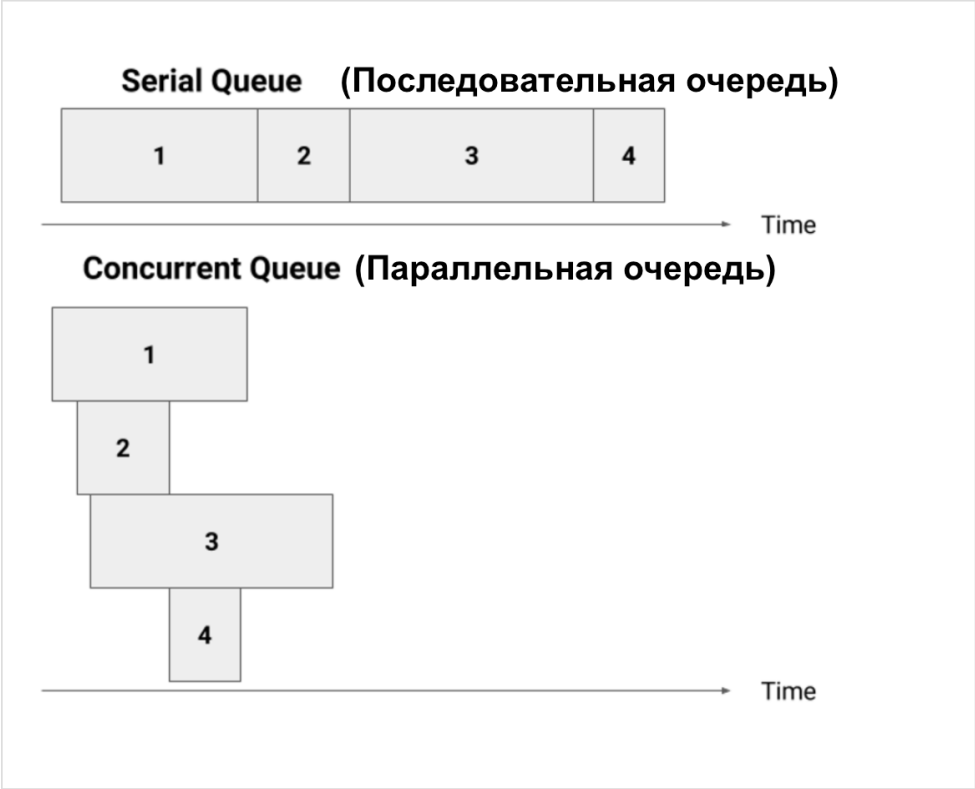
Тағы бір жақсы жаңалық – Xcode 8-ден бастап GCD және Операцияны үйрену үшін Playgroud сияқты қуатты және интуитивті құралды пайдалануға болады. Xcode 8 жаңа көмекші класын – PlaygroudPage ұсынады, ол Playgroud-қа шексіз өмір сүруге мүмкіндік береді. Бұл жағдайда DispatchQueue жұмыс аяқталғанға дейін жұмыс істей алады. Бұл әсіресе желі сұраныстары үшін өте маңызды. PlaygroudPage класын пайдалану үшін PlaygroudSupport модулін импорттау керек. Бұл модуль сонымен қатар іске қосу циклына қол жеткізуге, интерфейстің интерфейсін көрсетуге және Playgroud-та асинхронды операцияларды орындауға мүмкіндік береді. Төменде біз бұл параметрдің жұмыс істеп тұрғанын көреміз. Xcode 8-дегі бұл жаңа ойын алаңы Swift 3-те көпсалалы оқуды өте оңай және интуитивті етеді.

Параллельдікті жақсы түсіну үшін Apple GCD және Operation екеуі де жұмыс жасайтын кейбір дерексіз ұғымдарды ұсынды. Негізгі түсінік – кезек. Сондықтан, iOS қосымшасын жасаушының көзқарасы бойынша iOS-та

мультитрединг туралы сөйлескен кезде кезек туралы айтылады. Кезек дегеніміз – адамдар кезекке тұрған қарапайым кезектер, мысалы, кино билетін, бірақ біздің жағдайда жабылулар (анонимді код блоктары) кезекке тұрады. Жүйе оларды кезекке сәйкес орындайды, келесі кезекті «шығарып» алады және оны осы кезекке сәйкес жіпке орындау үшін іске қосады. Кезектер FIFO үлгісімен жүреді (First In, First Out), демек кезекте кім бірінші кезекте тұрса, бірінші орындауға жіберіледі. Сізде бірнеше кезек болуы мүмкін, және жүйе әр кезектен бір-бірден жабылады және оларды өз жіптерімен басқарады. Осылайша сіз көп мәтінді аласыз.

Бірақ бұл iOS-та параллелдіктің қалай жұмыс істейтіні туралы жалпы идея ғана. Бастапқыда бұл кезектер тапсырмаларды бір-біріне қатысты (дәйекті немесе параллель) орындау мағынасында бейнеленеді және олардың көмегімен қандай функциялар (синхронды немесе асинхронды) кезекке қойылады, сол арқылы ағымдағы кезекті блоктайды немесе блоктамайды.

Кезектер «сериялық» болуы мүмкін, мұнда кезектің жоғарғы жағында тұрған тапсырма (жабылу) iOS арқылы тартылады және ол аяқталғанға дейін жұмыс істейді, содан кейін келесі элемент кезектен алынады және т.б. Бұл сериялық кезек немесе сериялық кезек. Кезектер «қатарлас» болуы мүмкін, мұнда жүйе кезектің жоғарғы жағындағы жабуды «тартып алады» және оны белгілі бір жіппен басқарады. Егер жүйеде әлі де ресурстар болса, онда ол кезекті элементті алады және оны бірінші функция әлі жұмыс істеп тұрған кезде басқа тізбекте орындау үшін бастайды. Сонымен, жүйе барлық функциялар спектрін шығара алады. Көпжоспарлаудың жалпы тұжырымдамасын «қатарлас кезектермен» (көп тізбектелген кезектермен) шатастырмау үшін, біз «қатарлас кезекті» параллель кезек деп атаймыз, бұл оған бір-біріне қатысты жұмыс орындарының орындалу ретін білдірмейді. осы параллельді техникалық іске асыру. [2]

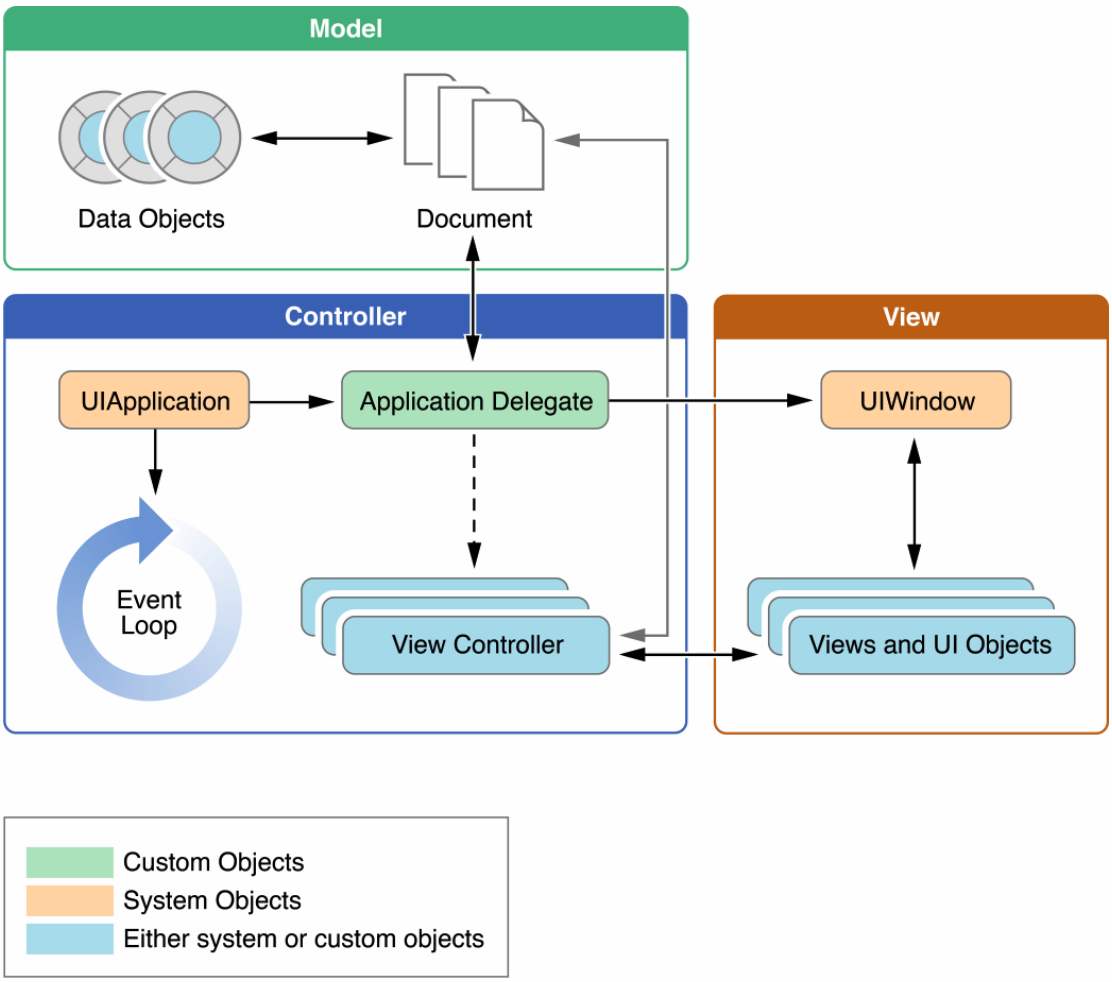


2.6.2-сурет – Serial Queue және Concurrent Queue

Тізбектелген (дәйекті) кезекте жабылулар оларды орындау үшін берілген дәл тәртіпте аяқталатынын көреміз, ал қатарлас (параллель) кезекте жұмыс күтпеген түрде аяқталады. Сонымен қатар, сіз белгілі бір жұмыс тобының сериялық кезектегі орындалуының жалпы уақыты қатар тұрған кезектегі жұмыс орындарын аяқтауға кететін уақыттан әлдеқайда көп екенін көре аласыз. Тізбектелген (дәйекті) кезекте кез-келген уақытта бір ғана жұмыс жүреді, ал қатарлас (параллель) кезекте кез-келген уақытта жұмыс саны өзгеруі мүмкін.[10]

2.7 IOS мобильді қосымшасының өмірлік циклы

Іске қосу кезінде UIApplicationMain функциясы бірнеше негізгі объектілерді таңдап, қосымшаны іске қосады. Әрбір iOS қосымшасының негізінде UIApplication нысаны жатыр, ол жүйе мен қосымшаның басқа объектілері арасындағы байланысты жеңілдетеді. Суретте көптеген қосымшаларда кездесетін негізгі объектілер көрсетілген. IOS қосымшасында MVC архитектурасы қолданылатындығын ескеріңіз. Бұл үлгі қолданбалы деректерді іскери логикадан және деректердің визуалды ұсынуынан бөледі. Бұл архитектура экранның өлшемдері әр түрлі құрылғыларда жұмыс істей алатын қосымшаны құрудың кілті болып табылады.



2.7.1-сурет – Қосымшаларда кездесетін негізгі объектілер

Күйлер арасындағы негізгі қозғалыстар қолданбалы нысанда көрсетілген арнайы әдістерді шақырады. Бұл әдістер күйдің өзгеруін басқаруға мүмкіндік береді.

– application: willFinishLaunchingWithOptions: – сіздің қосымшаңыздың бұл әдісі қосымшаны іске қосу кезінде кодты орындауға мүмкіндік береді;

– application: didFinishLaunchingWithOptions: – Бұл әдіс қолданбаны қолданушыға көрсетпес бұрын іске қосуды тоқтатпас бұрын кодты орындауға мүмкіндік береді;

– applicationDidBecomeActive: – Оның алдыңғы қатарға қосылатындығын білуге мүмкіндік береді. Бұл әдісті соңғы дайындық үшін қолданыңыз;

– applicationWillResignActive: – Бағдарламаның алдыңғы режимнен шығатынын хабарлаңыз. Қолданбаны демалыс режиміне қою үшін осы әдісті қолданыңыз;

– applicationDidEnterBackground: – қосымшаның фонда жұмыс істейтінін және оны кез келген уақытта өшіруге болатындығын білуге мүмкіндік береді;

– applicationWillEnterForeground: – Сіздің қосымшаңыз фоннан алдыңғы орынға ауысқанын, бірақ ол әлі белсенді емес екенін білуге мүмкіндік береді;

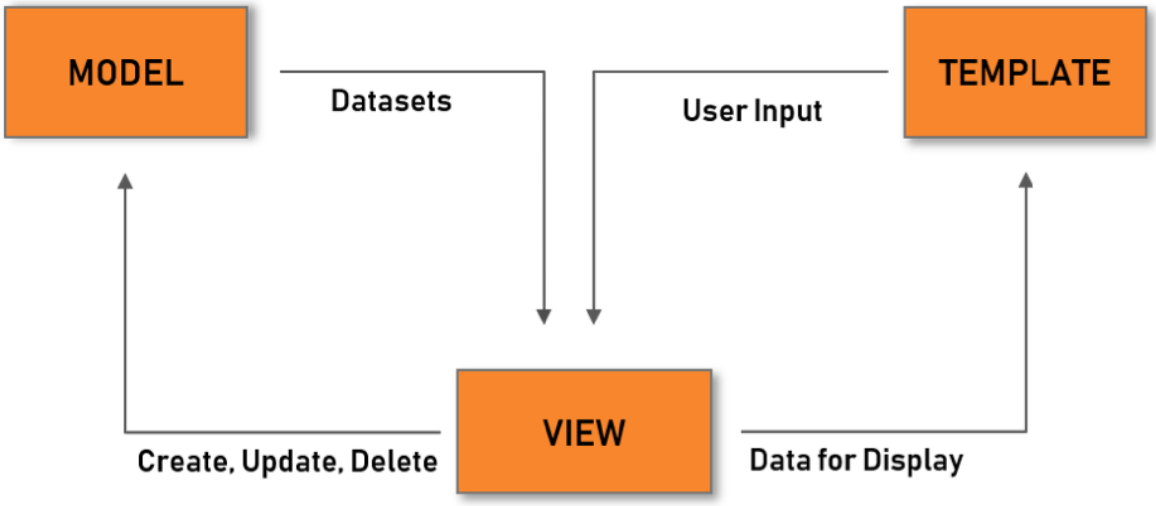
– applicationWillTerminate: – қосымшаның жабылғанын білуге мүмкіндік береді. Егер бағдарлама уақытша тоқтатылған болса, бұл әдіс шақырылмайды.

Қосымша әрқашан оны кез-келген уақытта өшіруге болатындығына дайын болуы керек және кез-келген параметрлердің немесе деректердің сақталуын күтпеуі керек. Өшіру – бұл қолданбалы қолдану циклінің қалыпты бөлігі. Әдетте жүйе жадыны тазарту немесе қолданушы іске қосатын басқа қосымшаларды іске қосуға дайындалу үшін қосымшаларды өшіреді, бірақ жүйе дұрыс жұмыс істемейтін немесе жауап бермейтін қосымшаларды да өшіре алады.[2]

Тоқтатылған қолданбалар тоқтату туралы хабарлама алмайды. Жүйе процесті өлтіреді және тиісті жадыны қалпына келтіреді. Егер қосымша фондық режимде жұмыс істеп тұрса және жауап бермей жатса, жүйе applicationWillTerminate: қосымшаны өшіруге дайындау үшін шақырады. Құрылғы қайта жүктелген кезде жүйе әдісті шақырмайды.

2.8 Серверлік бөлім – Django фреймворкі

Django бұл веб-қосымшаларды әзірлеу үшін MVC дизайн үлгісін қолданатын ақысыз Python фреймворкі. Оны Lawrence-Journal World әзірлеушілері Адриан Головати мен Саймон Уиллисон іске асырды, өйткені басылымның жаңалықтарын жариялау үшін интернет-сайт қажет. Django сәулеті модель-қарау-контроллер (MVC) жүйесімен сипатталады. Классикалық MVC моделінің контроллері Django (View) көрінісіне сәйкес келеді, ал презентация логикасы Үлгі деңгейінде жүзеге асырылады. Осы себепті Django-ның деңгейлік архитектурасы көбіне «модель-үлгі-көрініс» (MTV) деп аталады. Сәулет бір қарағанда күрделі болып көрінгенімен, қолданушыларға көптеген нұсқалар қол жетімді. Төменде көрсетілген суреттен байқауға болады.[8]



2.8.1-сурет – Django архитектурасы

Сонымен қатар, DRY (өзіңізді қайталамаңыз) қағидаты құрылымдық құрылымда жақсы дамыған. Олар қолданбаны құру кезінде пайдаланушыларға

жеңілдіктер жасау мақсатында әзірленген. Яғни, бір кодты бірнеше рет қайталап жазудың қажеті жоқ. Python туралы түсінігі бар жаңадан бастаушылар үшін қолайлы, жедел даму жобаларында өте тиімді. Django прагматикалық және таза дизайн үлгісіне құрылған және күрделі веб-қосымшаларды әзірлеуге қажетті барлық негізгі компоненттерден тұрады.

2.9 Сервермен қарым-қатынас жүйесі

Сонымен, бізде серверден деректерді өңдеу қажеттілігі туындайды. Бұл тапсырма барлық мобильді қосымшаларда бар.

Бұл үшін жергілікті құрал бар – URLSession, бірақ онымен жұмыс істеу біз қалағандай қиынырақ. Бұл процесті жеңілдету үшін, Alamofire жақтауы бар, ол URLSession-ті орайтын, сервермен жұмыс істеу кезінде өмірді едәуір жеңілдетеді.

Alamofire – бұл iOS және Mac OS X жүйелеріне арналған Swift HTTP желілік кітапханасы, ол Apple компаниясының Foundation желілік стекі үшін талғампаз интерфейсті ұсынады, ол бірқатар жалпы желілік тапсырмаларды жеңілдетеді.

Alamofire жауап беру / сұрау салу әдісін, JSON параметрін және сериялandыру жауабын, аутентификация және басқа да көптеген мүмкіндіктерді ұсынады. Бұл Alamofire-ді файлдарды жүктеу және үшінші тараптың RESTful API интерфейстерінен деректер сұрау сияқты негізгі желілік тапсырмаларды орындау үшін қолданасыз.

Alamofire талғампаз, өйткені ол Swift-те нөлден жазылған және оның Objective-C аналогы AFNetworking-тен ештеңе мұраға алмайды.

Сізде HTTP туралы жалпы түсінік және Apple компаниясының NSURLSession және NSURLConnection сияқты желілік класстары туралы біраз түсінік болуы керек.

Alamofire іске асырудың егжей-тегжейіне тоқталғандықтан, кейбір негізгі білімдер пайдалы болады, себебі сіздің желілік сұраныстарыңызды жою қажет болуы мүмкін. Alamofire проект жобасына енгізу үшін сізге CocoaPods орнатылған болуы керек.

Жобаның негізгі каталогында келесі мазмұнмен Podfile деп аталатын файл жасау керек. [3]

```
platform :ios, '9.0'

inhibit_all_warnings!
use_frameworks!

target 'PhotoTagger' do
  pod 'Alamofire', '~> 3.1.2'
end
```

2.9.1-сурет – Podfile файлының іші

Содан кейін CocoaPods тәуелділіктерін «pod install» командасы арқылы орнату керекпіз.

Егер сіз осы Alamofire оқулығына дейін үшінші тараптың интернет қызметін пайдаланбаған болсаңыз, онда сіз бұл қысқартулардың нені білдіретінін білмеуіңіз мүмкін!

HTTP – бұл веб-серверден сіздің экраныңызға деректерді тасымалдау үшін веб-сайттар қолданатын қолданбалы хаттама немесе ережелер жиынтығы. Сіз өзіңіздің веб-шолғышыңызға кіретін барлық URL мекенжайларының алдыңғы жағында HTTP (немесе HTTPS) арқылы кездестірдіңіз. Сіз FTP, Telnet және SSH сияқты басқа қолданбалы хаттамалар туралы естіген боларсыз. HTTP клиенттің (сіздің веб-шолғышыңыздың немесе қосымшаңыздың) қажетті әрекетті көрсету үшін қолданатын бірнеше сұрау әдісін немесе етістігін анықтайды:

1. GET: веб-беттер сияқты деректерді алу үшін қолданылады, бірақ сервердегі деректерді өзгертпейді;
2. HEAD: GET-пен бірдей, бірақ нақты ақпаратты емес, тек тақырыптарды жібереді;
3. POST: Деректерді серверге жіберу үшін қолданылады, әдетте форма толтырылып, жіберу батырмасы басылғанда;
4. PUT: белгілі бір алдын ала орнатылған жерге дерек жіберу үшін қолданылады;
5. DELETE: Көрсетілген орыннан деректерді жояды.

REST немесе REpresentational State Transfer – бұл жүйелі, қолдануға ыңғайлы және қызмет ететін веб-API құру ережелерінің жиынтығы. REST-те сұраныстар арасындағы құбылмалы күйлерді орындайтын, кэштелетін сұраныстар жасайтын және бірдей интерфейстерді қамтамасыз ететін бірнеше архитектуралық ережелер бар. Мұның бәрі сіз сияқты қосымшаны әзірлеушілерге API-ді қосымшаларыңызға оңай қосуға мүмкіндік береді – бұл сұраныстың негізінде деректердің күйін қадағаламай-ақ.

JSON (JavaScript Object Notation), екі жүйе арасында деректерді берудің қарапайым, оқуға оңай және тасымалданатын механизмін ұсынады. JSON деректер типінің шектеулі санына ие: жол, логикалық, массив, объект / сөздік, нөл (0) және сан. Толық сандар мен ондық сандар арасында айырмашылық жоқ.

Apple объектілерді жадтан JSON-ға және керісінше түрлендіруге көмектесетін NSJSONSerialization класын ұсынады.

HTTP, REST және JSON тіркесімі – бұл сізге әзірлеуші ретінде қол жетімді веб-қызметтердің негізгі бөлігі. Әр элементтің оқшауланған түрде қалай жұмыс істейтінін түсінуге тырысу қиын мәселе болуы мүмкін. Alamofire сияқты кітапханалар осы қызметтермен өзара әрекеттесудің күрделілігін азайтуға көмектеседі – сондықтан сіз оларды игеріп, тезірек жұмыс істеуге үйренесіз.

Бізге Alamofire не үшін қажет? Apple қазірдің өзінде HTTP арқылы мазмұнды жүктеу үшін NSURLSession және басқа сыныптарды ұсынады, сондықтан үшінші тарап кітапханасымен қиындықтар тудырудың қажеті не?

Қысқаша айтқанда, Alamofire NSURLSession-ге негізделген, ол сізді код жазуды жеңілдетеді, ол сізді плитаның кодын жазудан босатады. Сіз Интернеттегі деректерге оңай қол жеткізе аласыз, сонда сіздің кодыңыз әлдеқайда таза және оқуға оңай болады.[3]

Alamofire-де бірнеше негізгі мүмкіндіктер бар:

– upload: көп компонентті, ағындық, файлдық және деректер әдістерін қолдана отырып файлдарды жүктеу.

– download: файлдарды жүктеу немесе жүктеу жалғасуда.

– request: Файлдарды тасымалдаумен байланысты емес кез келген басқа HTTP сұрауы.

Бұл Alamofire функциялары класс немесе құрылым емес, модуль шеңберінде. Alamofire негізінде менеджер, сұраныс және жауап сияқты сыныптар мен құрылымдар орналасқан; дегенмен, оны қолдануды бастау үшін Alamofire құрылымын толық түсінудің қажеті жоқ.

Apple-дің NSURLSession және Alamofire-дағы сұраныс функциясы арқылы сол желілік операцияның қалай жұмыс істейтініне мысал келтірілген:

```swift
// With NSURLSession
public func fetchAllRooms(completion: ([RemoteRoom]?) -> Void) {
  let url = NSURL(string: "http://localhost:5984/rooms/_all_docs?include_docs=true")!

  let urlRequest = NSMutableURLRequest(
    URL: url,
    cachePolicy: .ReloadIgnoringLocalAndRemoteCacheData,
    timeoutInterval: 10.0 * 1000)
  urlRequest.HTTPMethod = "GET"
  urlRequest.addValue("application/json", forHTTPHeaderField: "Accept")

  let task = urlSession.dataTaskWithRequest(urlRequest)
    { (data, response, error) -> Void in
    guard error == nil else {
      print("Error while fetching remote rooms: \(error)")
      completion(nil)
      return
    }

    guard let json = try? NSJSONSerialization.JSONObjectWithData(data!,
      options: []) as? [String: AnyObject] else {
        print("Nil data received from fetchAllRooms service")
        completion(nil)
        return
    }

    guard let rows = json["rows"] as? [[String: AnyObject]] {
      print("Malformed data received from fetchAllRooms service")
      completion(nil)
      return
    }

    var rooms = [RemoteRoom]()
    for roomDict in rows {
      rooms.append(RemoteRoom(jsonData: roomDict))
    }

    completion(rooms)
  }

  task.resume()
}
```

2.9.2-сурет – SURLSession кітапханасымен жазылған код бөлігі

Сіз Alamofire параметрінің функциядан гөрі қысқа әрі түсінікті екенін көре аласыз. Жауапты responseJSON-ге айналдырдыңыз (options: completeHandler:) және Response объектісіне validate()-пен қоңырау шалу қатені өңдеуді жеңілдетеді.

```swift
// With Alamofire
func fetchAllRooms(completion: ([RemoteRoom]?) -> Void) {
  Alamofire.request(
    .GET,
    "http://localhost:5984/rooms/_all_docs",
    parameters: ["include_docs": "true"],
    encoding: .URL)
    .validate()
    .responseJSON { (response) -> Void in
      guard response.result.isSuccess else {
        print("Error while fetching remote rooms: \(response.result.error)")
        completion(nil)
        return
      }

      guard let value = response.result.value as? [String: AnyObject],
        rows = value["rows"] as? [[String: AnyObject]] else {
          print("Malformed data received from fetchAllRooms service")
           completion(nil)
           return
      }

      var rooms = [RemoteRoom]()
      for roomDict in rows {
        rooms.append(RemoteRoom(jsonData: roomDict))
      }

      completion(rooms)
  }
}
```

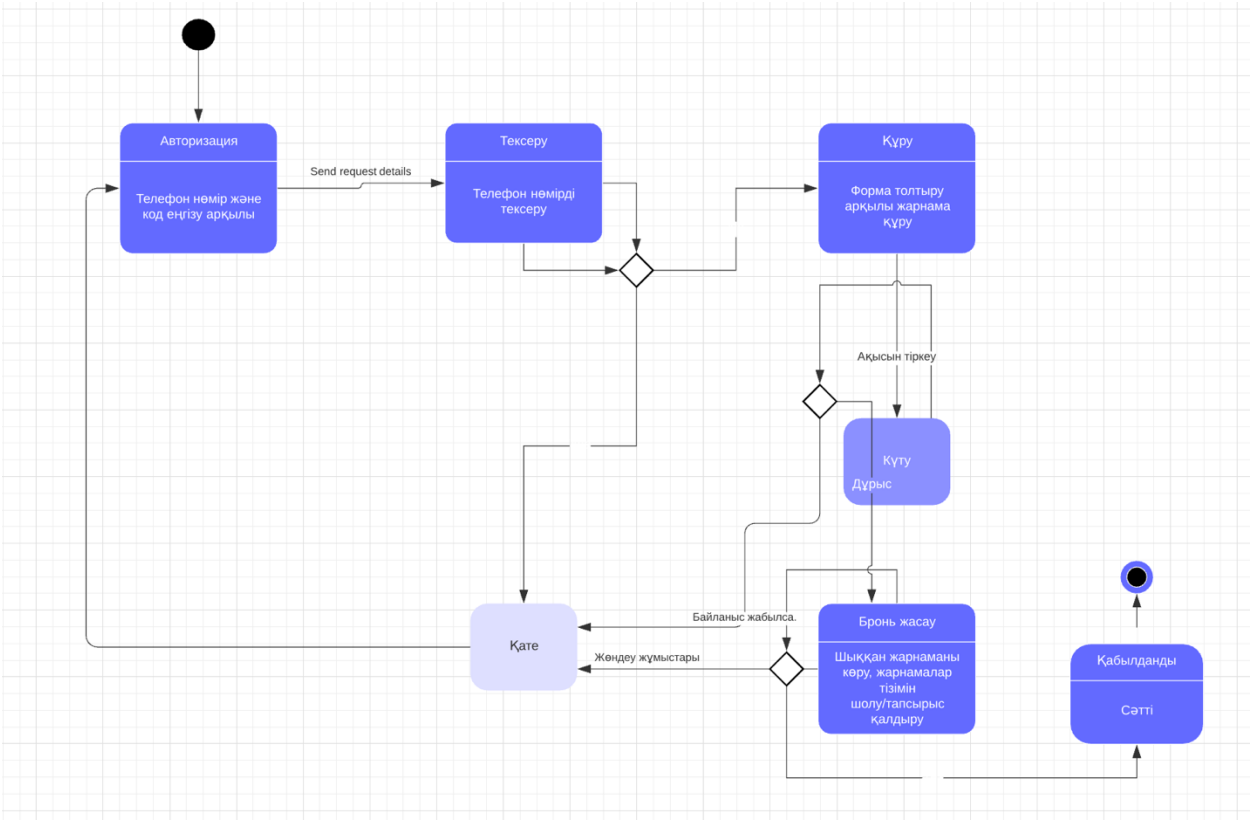
2.9.3-сурет – Alamofire кітапханасымен жазылған код бөлігі

3 Жобалау бөлімі

3.1 Мобильді қосымшаның логикалық құрылымы

Ал, енді біз жалға беру жүйесінің құрылым барысындағы ең маңызды бөлігі, осы дипломдық жобаның басты өзегі – мобильді қосымшаның құрылымына келдік. Біздің мобильді қосымшаны таңдаған себебіміз, қазіргі уақытта смартфон – адам өмірінің ажырамас бөлшегі, адам қайда барса да, қолынан әрқашан да ұялы телефон түсер емес. Барша егжей-тегжейін А қосымшасындағы техникалық тапсырмадан көре аламыз, алайда, осы тақырып барысында негізгі логикалық маңызы бар бөліктеріне тоқталып өтейік.

Бастапқыда UML диаграммасына көңіл аударайық:



3.1.1-сурет – Жобаның UML диаграммасы

Диаграммада көріп тұрғанымыздай біздің жүйенің басты ерекшелігінің бірі ол тіркелу, аутентификация жүйесі ұялы телефон нөмірі мен оған жіберілген код арқылы жүзеге асады. Кейін, қолданушы екі басты іс-әрекет жасай алады:

1. Жарнамалар тізімін көру, ол жайында ақпарат қарау;
2. Өзі қалаған затты, форма толтыру арқылы жарнамаға шығару.

Барлық кезекте де, біздің модератор ақпаратты тексеру жұмысын жүргізеді.

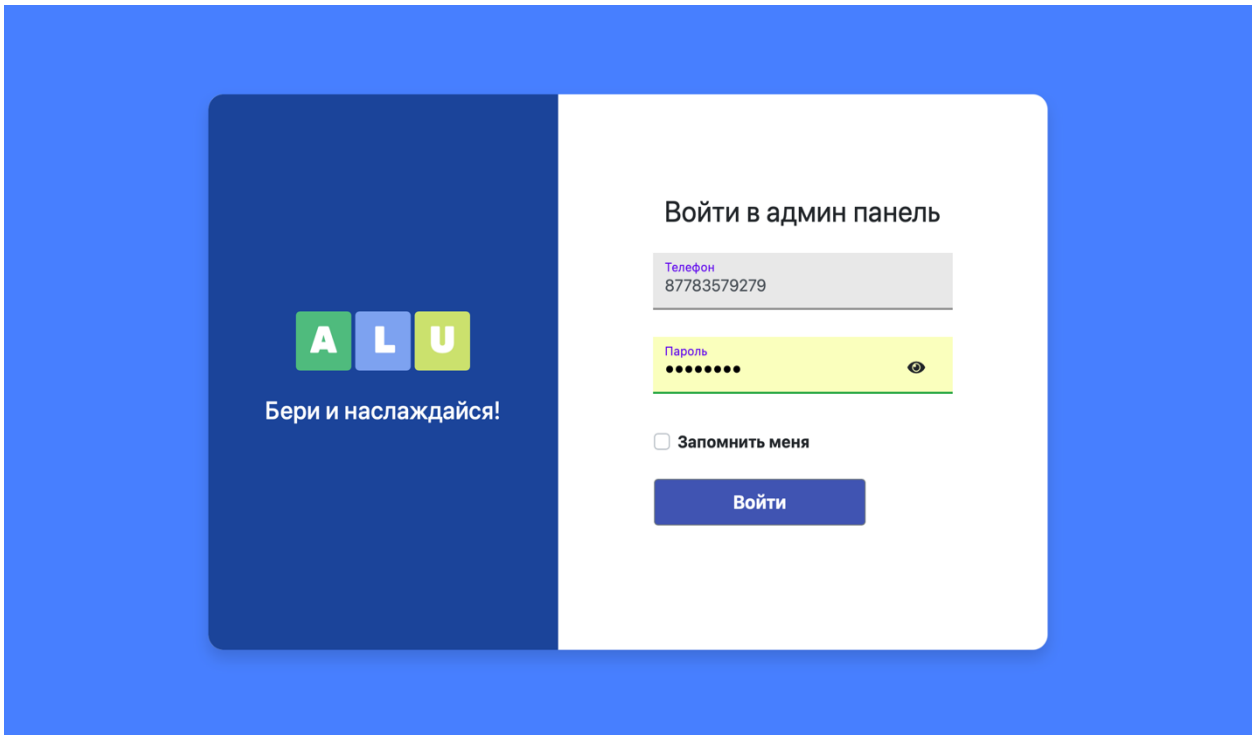
3.2 Жүйенің логикалық құрылымы

Бұл жүйені құрудағы бізге түскен қиыншылығы – бұған дейін осы жүйеге ұқсас аналогының болмауы. Республикамыздағы жалға беру жүйелерінің ішіндегі ыңғайлы, анық нұсқасын құрастыру – біздің мақсат. Жалпы жүйенің логикасына көңіл аударатын болсақ, ол бірнеше сатыдан тұратынын көреміз:

Жалға беруші тарапынан товардың күйлері:

1. Жарнамаға шығару жөніндегі тапсырыс;
2. Жарияланған;
3. Товарды алып кету;
4. Тапсырыстарды беру пунктінде;
5. Жалға берілген;
6. Товарды қайтару;
7. Қайта тапсырыстарды беру пунктінде.

Егер де көңіл аударатын болсақ, берілген жүйені іске асыру үшін бізге офис және жеткізу пунктіне жұмыскер керегін түсінеміз. Тапсырыс келу барысында офистағы модератор тапсырысты зерттеп шығуы керек. Дұрыстығы расталған кезде, оны жарнамалар тізіміне жібереді. Товардың күйі «жарияланған»-ға ауысады. Егер де товарды жалға алу жөнінде сұраныс келетін болса, жалға беруші push-хабарландыру арқылы сұраныстың түскенін көреді, алып кетуші келгенге дейін товарды дайындап қоюы керек. Алған сәттен бастап товардың күйі «жалға берілген» болып өзгереді.



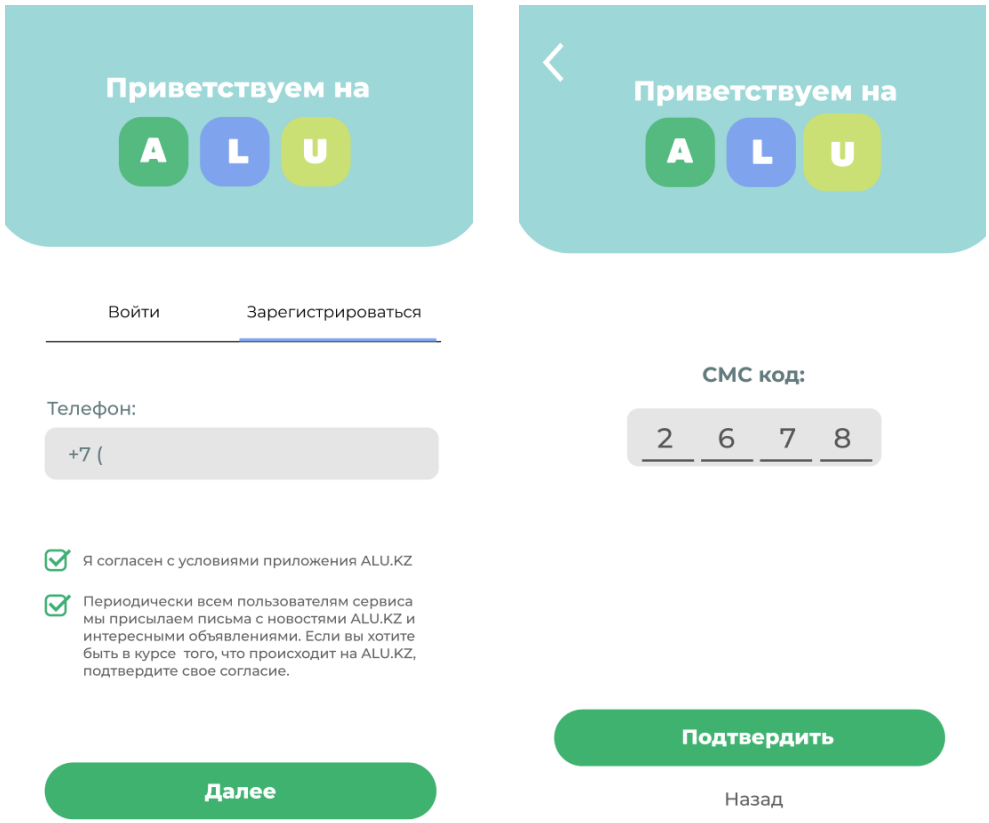
3.2.1-сурет – «Админ» қосымшасының бірінші беті

Жалға алушы тарапынан товардың күйлері:
1. Жалға алу жөніндегі сұраныс;
2. Қалыптасуда;
3. Жеткізу;
4. Өздігінен алып кету;
5. Қайтару (жеткізу, өздігінен алып келу);
6. Процесс аяқталды.

Демек, біздің мобильді қосымшаны қолданушы жеке тұлға жарнамалар тізімінен өзіне қажетті товарды көрген сәтте, оған сұраныс қалдырады. Осы сәтте бұл жағдайды бақылап отырған модератор админ қосымшасынан товардың мекен-жайын жұмыскерге хабарлау арқылы товарды алып кетіп, сұраныс қалдырушы адамға жеткізеді. Осы сәтте біз алып кету мен жеткізудің екі типін көріп тұрмыз. Бұл жағдайды анықтау үшін сұраныс қалдырушыға таңдау еркін береміз. Соңында, жалға алу мерзімі біткенге дейін, процесс аяқталған болып табылады.

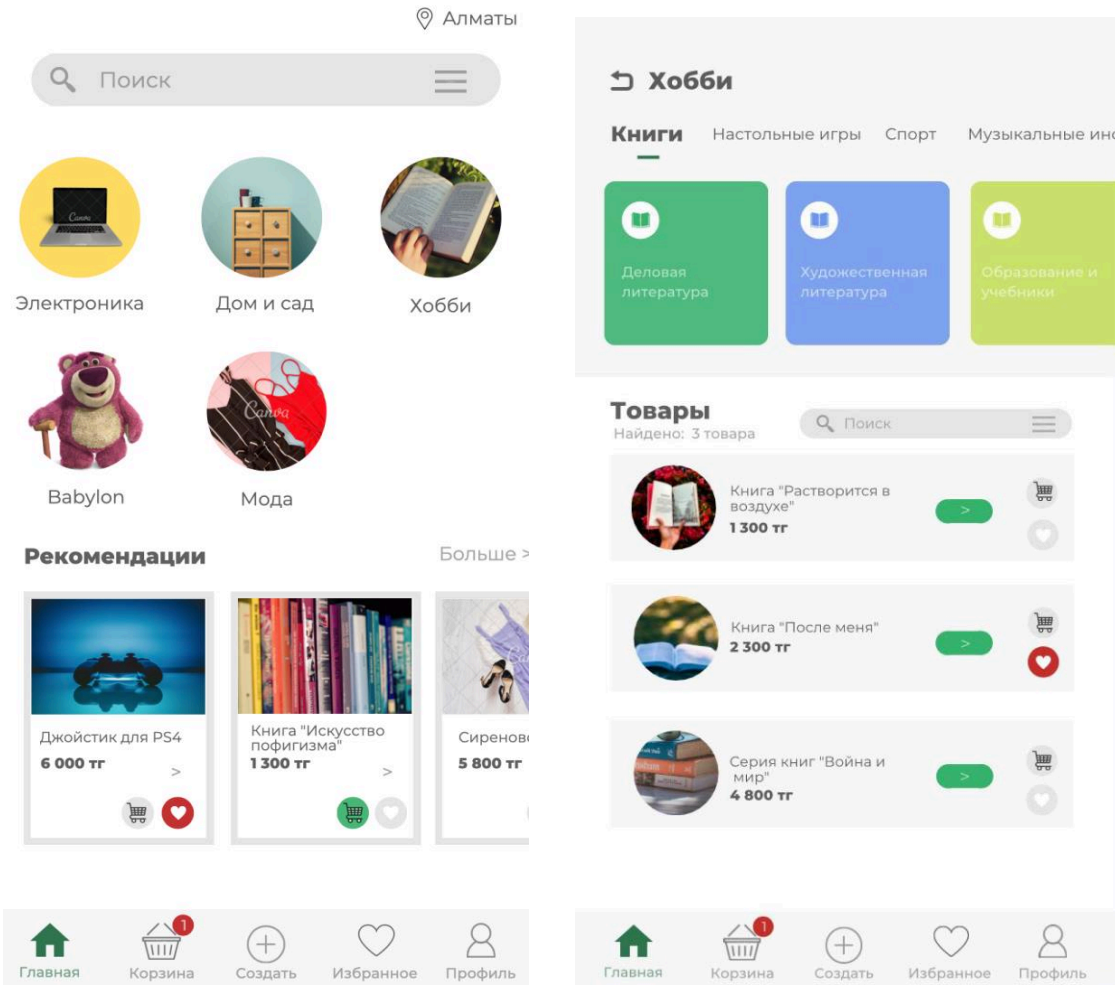
3.3 Қолданушы интерфейсі

«Alu.kz» мобильді қосымшасының интерфейсі басты бет (tab view controller) және бірнеше бөлімнен тұрады. Олар: басты бет, жарнама құру беті, профиль беті, себет беті, тіркелу беті (авторизация/регистрация).



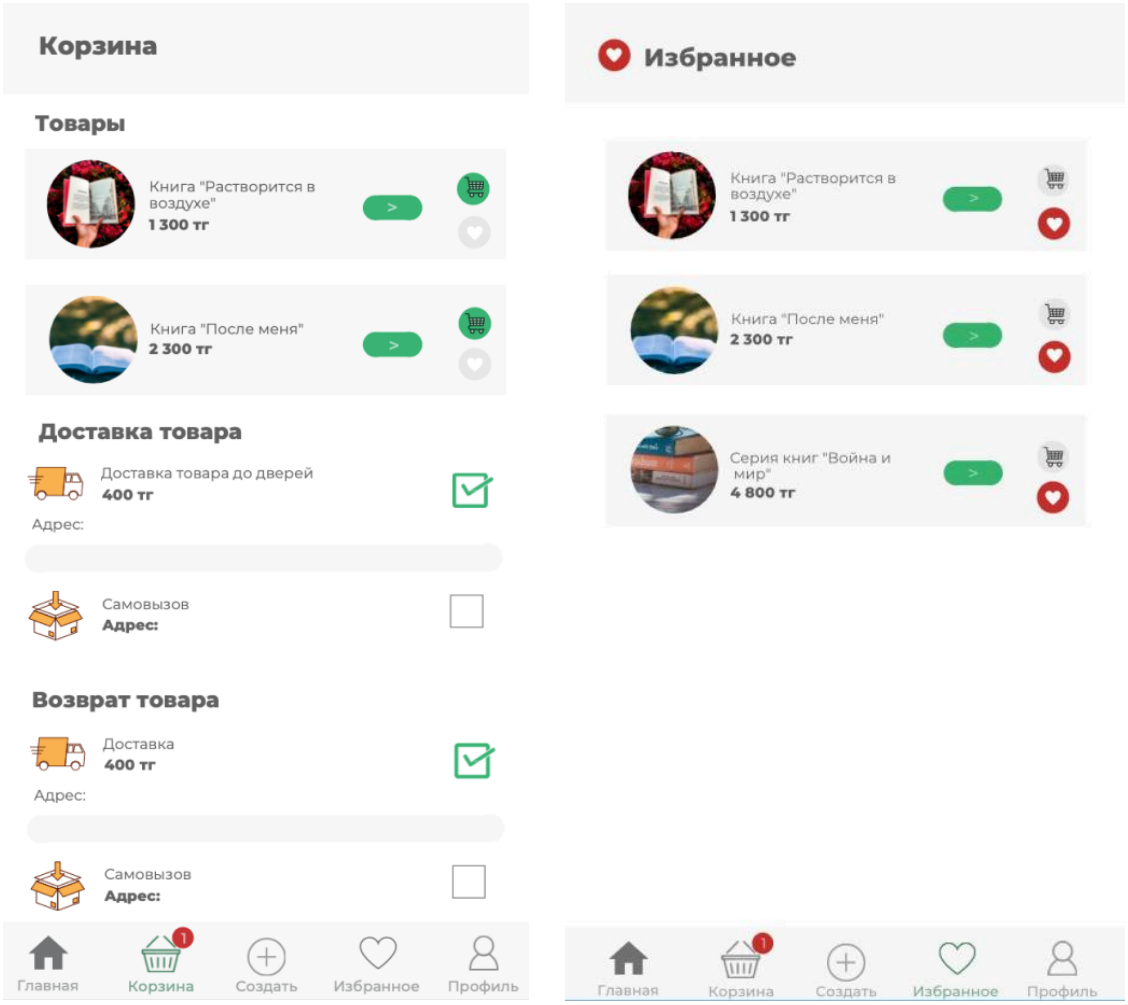
3.3.1-сурет – Тіркелу беттері

Жоғарыда атап айтқандай, тіркелу беті ұялы телефон нөмірін еңгізіп «Далее» батырмасын басу керек. Шыққан бетте ұялы телефон нөміріне жіберілген кодты растау арқылы жүзеге асады. Код 4 саннан тұрады. Мобильді қосымшаға тіркелген бетте, оны басты бетке (Tab view controller) өткіземіз.



3.3.2-сурет – Басты бет және Категория таңдалған бет

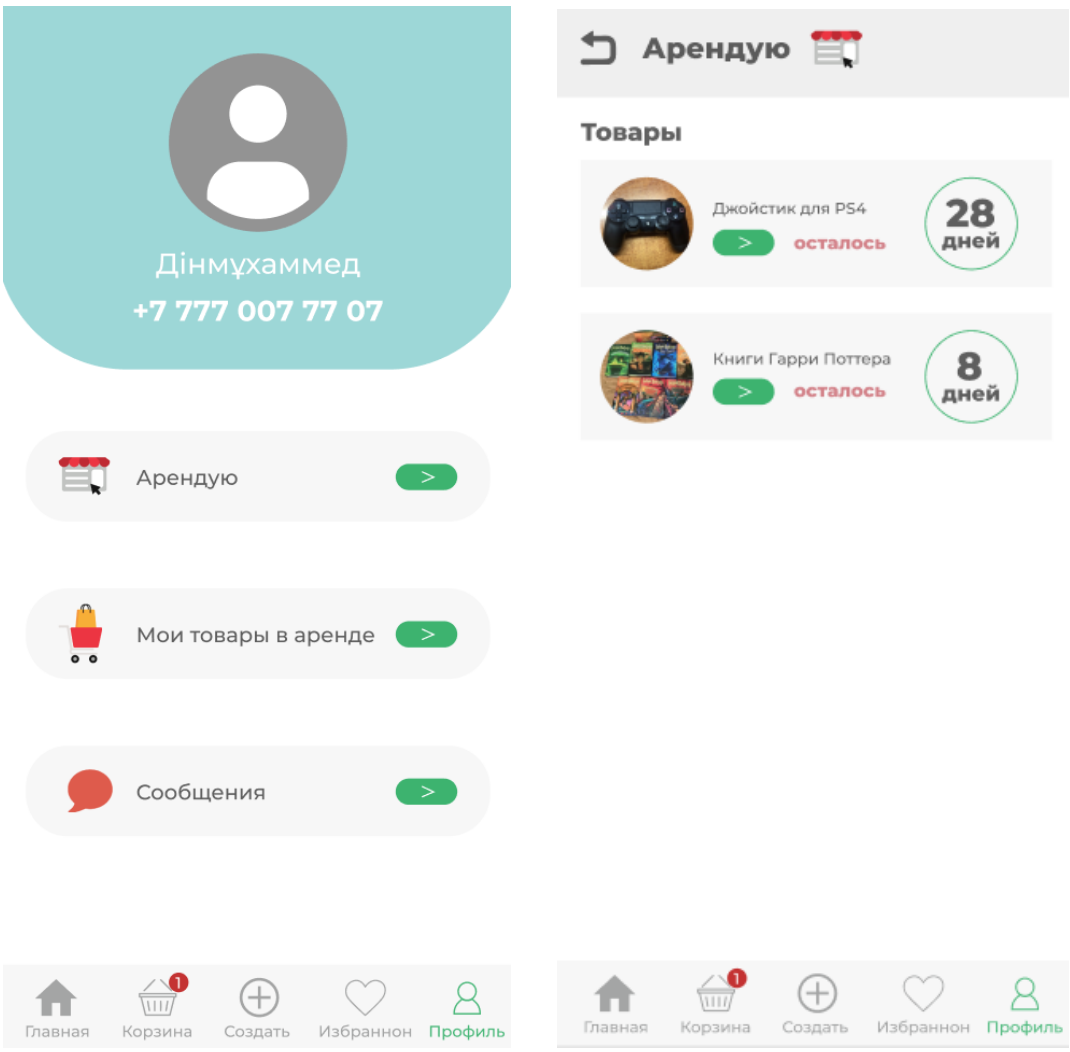
Жоғарыдағы скриншоттан көріп тұрғанымыздай, басты бет үш негізгі бөлімнен тұратынын көреміз. Олар: Іздеу, категория таңдау, ұсыным бөлімдері. Іздеу бөлімі біз жолға жазған сөз бойынша, бізге қажетті товарды іздеп, бізге ұсынады. Келесі бөлімде біз категория таңдап, келесі бетке өтеміз. Бұл бетте керекті товардың сипаттамасын таңдай отыра, тізімнен бар товарларды көреміз. Ұсыным бөлігінде біз қызыққан товарларға ұқсас немесе жаңадан жарнамаға шыққан товарларды көреміз.



3.3.3-сурет – Себет және Таңдаулылар беттері

Келесі беттер: себет және таңдаулы беттері. Себет бетінде біз өзімізге қажетті, жалға алуға таңдалған товарлардың тізімін көреміз. Осы бетте тұлға өзінің ыңғайына қарай қажетті опцияны таңдайды. Екінші таңдаулы бетінде, қолданушы қызық деп санаған товарлардың тізімін көреді. Ол үшін ол товардың картасындағы жүрекше батырмасын басуы керек.

Тағы басты беттердің екеуі профиль және жалға алулы товарлар беттері. Профиль бетінде қолданушының аты-жөні мен ұялы телефон нөмірі бейнеленген. Бұл беттегі кезекті үш батырма жалға алулы товарлардың тізімі, өзі жалға берген товарлардың тізімі және хабарландыру мәтіндері.



3.3.4-сурет – Профиль және жалға алулы товарлар беттері

ҚОРЫТЫНДЫ

Дипломдық жобамен жұмыс барысында «Alu.kz» деген атаумен, жалға беру және алу жүйесі негізінде жасалған IOS мобильді қосымшасы әзірленді. Әзірленген жүйе осы күнге дейінгі әлі автоматтандыруға келмеген жалға беру жүйесін жеткілікті түрде қарастырды. Қосымшаны пайдалану барысында әр жеке тұлға өзіне қажетті құралдар мен товарларды жарнама тізімінен көріп жалға алу мүмкіндігін иеленді. Сонымен қоса, өз меншігіндегі құралды немесе товарды жалға беру арқылы ақылы сияқы ала алады. Жүйені максималды ыңғайлы жасау үшін профиль бетіне жалға берілген товарлардың күйін көретін бөлім қосып, өзі жалға алған құралдың мерзімін бақылай алатын бөлім қостық. Әр қажетті мағлұматты қолданушыға дереу жеткізу үшін, push-хабарландыру жүйесін де қоса алдым. Бұл дипломдық жобаның және біз ойлап тауып жүзеге асырған жалға алу немесе беру жүйесінің басты мақсаты – ақпараттық даму заманында Республикамыздың ақпараттық және технологиялық даму үрдісін жеделдету, Қазақстан Республикасының тұрғындары мен барша адамзаттың ыңғайлы өмір сүру салтын дамыту.

Әзірлеу барысында қолданылған технологиялар тізімі: бэкенд бөлігіне Django фреймворкі, фронтенд бөлігіне Xcode IDE және Swift программалау тілі, деректер қорын жүзеге асыру мақсатында PostgreSQL тілі.

Болашаққа жоспар ретінде мобильді қосымшадағы ұсыныс бөлігіне машиналық оқытуды қосу арқылы қажетті товарлардың тізімін құруды алып отырмын. Сондай-ақ, жеткізу және алып кету мерзімін нақтылау үшін «Google Maps» сервисін қосу қажет. Модератор жұмысын автоматтандыру үшін тапсырыстарды тексеру жүйесін құру керек. Аталған жаңалықтарды іске асыру жүйемен жұмысты жақсартуға және қамтаманың құндылығын арттыруға мүмкіндік береді.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1. Унифицированный язык моделирования UML // Электрондық нұсқасы http://samara.mgpu.ru/~dzhadzha/dis/15/200.html
2. Swift. Основы разработки приложений под iOS, iPadOS и macOS. 6-е изд. дополненное и переработанное // Электрондық нұсқасы https://habr.com/ru/company/piter/blog/534380/
3. Нативная разработка мобильных приложений М. Данн, Ш. Льюис: Электрондық нұсқасы https://www.studentlibrary.ru/kk/book/ISBN9785970608456.html.
4. Программирование для iOS 7. Основы Objective-C, Xcode и Cocoa. Руководство // Электрондық нұсқасы http://www.williamspublishing.com/PDF/978-5-8459-1895-6/part.pdf
5. Вандад Нахавандипур iOS. Приемы программирования // Электрондық нұсқасы https://www.spbdk.ru/catalog/item-ios_priemy_programmirovaniya/
6. Программирование под iOS. Для профессионалов // Электрондық нұсқасы https://www.litres.ru/aaron-hillegass/programmirovanie-pod-ios-dlya-professionalov-6058672/
7. Дэвид Марк, Джек Наттинг, Джефф Ламарш, Фредрик Олссон, Ким Топли Swift. Разработка приложений в среде Xcode для iPhone и iPad с использованием iOS SDK // Электрондық нұсқасы https://monster-book.com/swift-3-razrabotka-prilozheniy-v-srede-xcode
8. Alamofire в среде Xcode // Электрондық нұсқасы https://swiftbook.ru/post/tutorials/tutorial-po-alamofire/
9. Архитектурные паттерны в iOS // Электрондық нұсқасы https://habr.com/ru/company/badoo/blog/281162/
10. Учебник Swift — разработка приложения для iOS8 // Электрондық нұсқасы https://habr.com/ru/post/231541/

А Қосымшасы
(міндетті)

«Alu.kz» мобильді қосымшасын құруға арналған
техникалық тапсырма

А.1 Кіріспе

Қазіргі кезде кез-келген жүйені автоматтандыруға арналған мобильді қосымшалар пайда болуда. Оны күнделікті қолданып жүрген отандық немесе әлемдік үздік мобильді қосымшаларынан байқауға болады. Бұл өз кезегінде дәстүрлі жолмен жұмыс істеуді жылжытып, барлық мағлұматтарды электрондандыру жүйесіне әкеліп соғады. Сондықтан жүйені электронды түрде автоматтандырушы мобильді косымшалар қазіргі күннің өзекті тақырыбы. Сол себепті «Alu.kz» мобильді қосымшасы жалға беру жүйесіндегі мәселені шешуші құрал.

«Alu.kz» мобильді қосымшасы негізінен жалға алу немесе беру жүйесін қолдаңғысы келетін жеке тұлғаларға арналған. Қолданушы жарнама тізімін көру арқылы өзіне қажет товарды таңдап оны жалға ала алады. Бұл мобильді қосымша Қазақстан Республикасы немесе барша өзге мемлекет тұрғындарына жарамды мобильді қосымша болып табылады.

А.1.2 Қолдану саласы

Қолдану саласы – мобильді телефон, смартфон қолдана білетін кез-келген қолданушылар пайдалана алады. Жүктеу үшін iOS операциалық жүйесі орнатылған смартфон болуы қажет. Қосымшаны жүктеу үшін AppStore қосымшалар жүктеу программасын пайдалану қажет.

А.1.3 Анықтамалар, терминдер және қысқартулар

Анықтамалар, терминдер және қысқартулар А.1-кестеде көрсетілген.

А.1.4 Адаптация бойынша талаптар

«Alu.kz» мобильді қосымшасы кез-келген iOS ОЖ орнатылған құрылғыға адаптацияланған.

А Қосымшасының жалғасы

А.1-кесте – Анықтамалар, терминдер және қысқартулар

Терминдер немесе қысқартулар	Анықтамалар
Смартфон	Смартфон (ағылшынша смартфон – смартфон) – қалта дербес компьютерінің функционалдығымен толықтырылған ұялы телефон (заманауи – әдетте, сенсорлық экраны бар).
iOS	iOS (2010 жылдың 24 маусымына дейін – iPhone OS) – бұл американдық Apple компаниясы жасаған және өндірген смартфондарға, электронды планшеттерге және басқа да құрылғыларға арналған мобильді операциялық жүйе.
Xcode	MacOS, iOS, watchOS және tvOS платформаларына арналған Apple компаниясы жасаған интегралды бағдарламалық жасақтама ортасы (IDE).
Swift	Ашық көзді, мультипарадигма, жалпы мақсаттағы жинақталған бағдарламалау тілі. Apple негізінен iOS және macOS жасаушыларына арналған (қазіргі уақытта бұл операциялық жүйелер шеңберінен тыс).
App Store	Қосымшалар дүкені, iPhone ұялы телефондарына, iPod Touch ойнатқыштарына және iPad құрылғыларына, сондай-ақ Mac дербес компьютерлеріне арналған әр түрлі қосымшалардан тұратын және оларды тегін сатып алуға немесе жүктеуге мүмкіндік беретін iTunes Store интернет-дүкенінің бөлімі.

А.2 Жалпы сипаттамасы

Жалпы мобильді қосымша дерекқоры бұлттық режимде жұмыс істейтін App Store онлайн қосымшалар дүкенінде сақталынған. Дерекқор мен клиент арасындағы сұранымдарды Django жүргізеді, кейін дерекқордан алынған ақпараттар клиенттік бетке дереу шығарылатын болады.

А.2.1 Пайдаланушы интерфейстер

IOS ОЖ орнатылған құрылғыларда жұмыс істейтін бұл мобильді қосымшаның түрін қолданушылар қол жетімді әрі түсінікті интерфейсте қолдана алады.

А.2.2 Аппаратттық интерфейстер

Смартфонға қойылатын жалпы талаптар:
– iOS ОЖ;
– оперативті жады – 1GB;
– кеңістік – 42 MB
– ғаламтор желісі;
– сенсорлы экран.

А.2.3 Программалық интерфейстер

Жобаға қатысты бағдарламалық компоненттер:
– Mac OS X операциялық жүйесі (соңғы жаңартудағы нұсқа);
– Visual Studio Code – бастапқы кодты өзгертетін редактор ;
– Figma – веб дизайнерлік бағдарлама.
– Postman бағдарламасы – REST сұранымдарды жүргізу бағдарламасы.
– Xcode – интегралды бағдарламалық жасақтама ортасы.

А.2.4 Коммуникациялық интерфейстер

Қолданушылардың жоғары жылдамдықты интернет желісі болу керек. Сервер мен клиенттерді байланыстыратын TCP/IP протоколы қолданылады.

А.2.5 Жады бойынша шектеулер

Деректер сақталатын digitalocean бұлттық сервисінде бірқатар шектеулер бар. Ашып айтқанда, бұл онлайн деректер сервисінде тегін және ақылы бөлімдері бар. Мобильді қосымша жұмыс жасау барысында тегін ақылы бөлігін тандадым, өйткені оның мүмкіншіліктері кең болды. Жады бойынша шектеулері жойылған, жадының көлемі өскен.

Б Қосымшасы
(міндетті)

Бағдарлама мәтіні

1. Телефон нөмір мен код арқылы кіру бетінің мәтіні

```
import UIKit
import Alamofire
import SwiftyJSON

class AuthorizationController: UIViewController, UITextFieldDelegate {

    @IBOutlet weak var numberTextField: UITextField!
    @IBOutlet weak var nameTextField: UITextField!
    @IBOutlet weak var nameLabel: UILabel!
    @IBOutlet weak var phoneConstraint: NSLayoutConstraint!
    @IBOutlet weak var nameConstraint: NSLayoutConstraint!
    @IBOutlet weak var firstText: UIImageView!
    @IBOutlet weak var secondText: UIImageView!
    @IBOutlet weak var checkmark_1: UIImageView!
    @IBOutlet weak var checkmark_2: UIImageView!
    @IBOutlet weak var line_1: UIImageView!
    @IBOutlet weak var line_2: UIImageView!

    let defaults = UserDefaults.standard
    var isChecked_1 = 1
    var isChecked_2 = 1
    var isSignIn = 0

    override func viewDidLoad() {
        super.viewDidLoad()
        numberTextField.addTarget(self, action: #selector(textFieldDid-
Change(_:)), for: .editingChanged)
        addDoneButtonOnKeyboard()
        line_1.alpha = 0.0

        let tapGestureRecognizer_1 = UITapGestureRecognizer(target: self, ac-
tion: #selector(imageTapped_1(tapGestureRecognizer:)))
        let tapGestureRecognizer_2 = UITapGestureRecognizer(target: self, ac-
tion: #selector(imageTapped_2(tapGestureRecognizer:)))
        let tapGestureRecognizer_3 = UITapGestureRecognizer(target: self, ac-
tion: #selector(imageTapped_3(tapGestureRecognizer:)))
```

```
        firstText.addGestureRecognizer(tapGestureRecognizer_1)

        checkmark_1.addGestureRecognizer(tapGestureRecognizer_2)
        checkmark_2.addGestureRecognizer(tapGestureRecognizer_3)
    }

    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        self.view.endEditing(true)
        return false
    }

    func textFieldDidBeginEditing(_ textField: UITextField) {
        if textField == numberTextField{
            if self.numberTextField.text!.count < 6{
                self.numberTextField.text = "+7 ("
            }
        }
    }

    func addDoneButtonOnKeyboard()
    {
        let doneToolbar: UIToolbar = UIToolbar(frame: CGRect.init(x: 0, y: 0,
width: UIScreen.main.bounds.width, height: 50))
        doneToolbar.barStyle = .default

        let flexSpace = UIBarButtonItem(barButtonSystemItem: .flexibleSpace,
target: nil, action: nil)
        let done: UIBarButtonItem = UIBarButtonItem(title: "Done", style: .done,
target: self, action: #selector(self.doneButtonAction))

        let items = [flexSpace, done]
        doneToolbar.items = items
        doneToolbar.sizeToFit()

        numberTextField.inputAccessoryView = doneToolbar
    }

    @objc func doneButtonAction()
    {
        self.view.endEditing(true)
    }
```

```
        @IBAction func nextButton(_ sender: UIButton) {
            if isSignIn == 1{
                if numberTextField.text!.count == 20 && numberTextField.text!.count
> 3{
                    let number = numberTextField.text![1..<2] + numberText-
Field.text![4..<7] + numberTextField.text![9..<12] + numberTextField.text![14..<16]
+ numberTextField.text![18..<20]
                    let parameters = ["phone" : String(number)]
                    AF.request("**********", method:
.post, parameters: parameters, encoding: JSONEncoding.default, headers: nil).re-
sponseJSON { response in
                        let json = try? JSON(data: response.data!)
                        if (json!["status"] == "ok") {
                            self.defaults.set(String(number), forKey: "PhoneNumber")
                            let storyboard = UIStoryboard(name: "Main", bundle: nil)
                            let viewController = storyboard.instantiateViewController(with-
Identifier :"ValidationController")
                            self.present(viewController, animated: true)
                        }
                        else{
                            let alert = UIAlertController(title: "Внимание!", message: "Вы
не зарегистрированы. Пожалуйста, зарегистрируйтесь!", preferredStyle: .alert)
                            alert.addAction(UIAlertAction(title: "OK", style: .default, han-
dler: nil))
                            self.present(alert, animated: true)
                        }
                    }
                }
                else{
                    let alert = UIAlertController(title: "Внимание!", message: "Имя или
номер сотового неверного формата.", preferredStyle: .alert)
                    alert.addAction(UIAlertAction(title: "OK", style: .default, handler:
nil))
                    self.present(alert, animated: true)
                }
            }
            else{
                if isChecked_1 == 1{
                    if numberTextField.text!.count == 20 && numberText-
Field.text!.count > 3{
```

```
                    let number = numberTextField.text![1..<2] + numberText-
Field.text![4..<7] + numberTextField.text![9..<12] + numberTextField.text![14..<16]
+ numberTextField.text![18..<20]
                    let parameters = ["phone" : String(number), "name" : nameText-
Field.text!]
                    AF.request("**********************************",
method: .post, parameters: parameters, encoding: JSONEncoding.default, headers:
nil).responseJSON { response in
                      let json = try? JSON(data: response.data!)
                      if (json!["status"] == "ok") {
                        self.defaults.set(String(number), forKey: "PhoneNumber")
                        self.defaults.set(self.nameTextField.text!, forKey: "Name")
                        let storyboard = UIStoryboard(name: "Main", bundle: nil)
                        let viewController = storyboard.instantiateViewControl-
ler(withIdentifier :"ValidationController")
                        self.present(viewController, animated: true)
                      }
                    }
                  }
                  else{
                    let alert = UIAlertController(title: "Внимание!", message: "Имя
или номер сотового неверного формата.", preferredStyle: .alert)
                    alert.addAction(UIAlertAction(title: "OK", style: .default, handler:
nil))
                    self.present(alert, animated: true)
                  }
                }
                else{
                  let alert = UIAlertController(title: "Извините", message: "Нужно
согласие на обработку данных.", preferredStyle: UIAlertController.Style.alert)
                  alert.addAction(UIAlertAction(title: "OK", style: UIAlertAc-
tion.Style.default, handler: nil))
                  self.present(alert, animated: true, completion: nil)
                }
              }
            }
          }
```

2. Басты бетің мәтіні

```
import UIKit
import Foundation
import SwiftyJSON
```

```
import Alamofire
import Kingfisher

class MainController: UIViewController, UITextFieldDelegate, UICollectionViewDelegate, UICollectionViewDataSource{

    @IBOutlet weak var mainCollectionView: UICollectionView!
    @IBOutlet weak var activityIndicator: UIActivityIndicatorView!
    @IBOutlet weak var searchTextField: UITextField!

    let defaults = UserDefaults.standard
    var orderArray = [JSON]()

    override func viewDidLoad() {
        super.viewDidLoad()
        activityIndicator.startAnimating()
        searchTextField.borderStyle = .none
        searchTextField.setLeftPaddingPoints(35)
        searchTextField.layer.cornerRadius = 15
        searchTextField.clipsToBounds = true
        searchTextField.isUserInteractionEnabled = false
        GlobalVariables.basket = []
        GlobalVariables.favorites = []
        if self.defaults.bool(forKey: "isSignIn") == true{
            getFavorites()
            getBasket()
        }
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        navigationController?.setNavigationBarHidden(true, animated: animated)
        getRecomendation()
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)
    }

    @IBAction func searchButtonTapped(_ sender: UIButton) {
        let storyboard = UIStoryboard(name: "Main", bundle: nil)
```

```
        let viewController = storyboard.instantiateViewController(withIdentifier
:"SearchController")
        self.navigationController?.pushViewController(viewController,
        animated: true)
    }

    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        self.view.endEditing(true)
        return false
    }

    func collectionView(_ collectionView: UICollectionView, number-
OfItemsInSection section: Int) -> Int {
        return orderArray.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt in-
dexPath: IndexPath) -> UICollectionViewCell {
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "In-
terestingCollectionViewCell", for: indexPath) as! InterestingCollectionViewCell
        cell.name.text = orderArray[indexPath.row]["title"].string
        cell.price.text = orderArray[indexPath.row]["price_14"].stringValue + "тг"
        let urlOfImage = URL(string: orderArray[indexPath.row]["product_im-
age"][0]["image"].stringValue)!
        cell.imageView.kf.setImage(with: urlOfImage)

        if GlobalVariables.favorites.contains(orderArray[index-
Path.row]["id"].int!){
            cell.favoriteButton.setBackgroundImage(UIImage(named: "heart1"),
for: .normal)
        }
        else{
            cell.favoriteButton.setBackgroundImage(UIImage(named: "heart"), for:
.normal)
        }

        if GlobalVariables.basket.contains(orderArray[indexPath.row]["id"].int!){
            cell.basketButton.setBackgroundImage(UIImage(named: "basket1"),
for: .normal)
        }
        else{
```

```
                cell.basketButton.setBackgroundImage(UIImage(named: "basket"), for:
.normal)
            }


            if orderArray[indexPath.row]["is_rented"].int == 1{
                cell.bgView.backgroundColor = UIColor(red: 0.9882, green: 0.898,
blue: 0.7569, alpha: 1.0)
                cell.basketButton.isUserInteractionEnabled = false
            }
            else{
                if orderArray[indexPath.row]["owner"]["id"].stringValue == de-
faults.string(forKey: "UID")!{
                    cell.basketButton.isUserInteractionEnabled = false
                }
                else{
                    cell.basketButton.isUserInteractionEnabled = true
                }
                cell.bgView.backgroundColor = UIColor.white
            }

            cell.favoriteButton.addTarget(self, action: #selector(favor-
iteAction(sender:)), for: .touchUpInside)
            cell.favoriteButton.tag = orderArray[indexPath.row]["id"].int!
            cell.basketButton.addTarget(self, action: #selector(basketAction(sender:)),
for: .touchUpInside)
            cell.basketButton.tag = orderArray[indexPath.row]["id"].int!

            return cell
        }

        func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
            let id = indexPath.row
            defaults.set(orderArray[id].rawString(), forKey: "AboutProductInfo")

            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let viewController = storyboard.instantiateViewController(withIdentifier
:"OrderInfoController")
            self.navigationController?.pushViewController(viewController,
            animated: true)
        }
```

```
        @IBAction func electronicButton(_ sender: UIButton) {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let viewController = storyboard.instantiateViewController(withIdentifier
:"OrdersController")
            self.navigationController?.pushViewController(viewController,
            animated: true)

            defaults.set("Электроника", forKey: "mainTitle")
            defaults.set("4", forKey: "Category_id")
        }
        @IBAction func forHomeButton(_ sender: UIButton) {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let viewController = storyboard.instantiateViewController(withIdentifier
:"OrdersController")
            self.navigationController?.pushViewController(viewController,
            animated: true)

            defaults.set("Дом и сад", forKey: "mainTitle")
            defaults.set("3", forKey: "Category_id")
        }
        @IBAction func hobbyButton(_ sender: UIButton) {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let viewController = storyboard.instantiateViewController(withIdentifier
:"OrdersController")
            self.navigationController?.pushViewController(viewController,
            animated: true)

            defaults.set("Хобби", forKey: "mainTitle")
            defaults.set("1", forKey: "Category_id")
        }
        @IBAction func babylonButton(_ sender: UIButton) {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
            let viewController = storyboard.instantiateViewController(withIdentifier
:"OrdersController")
            self.navigationController?.pushViewController(viewController,
            animated: true)

            defaults.set("Babylon", forKey: "mainTitle")
            defaults.set("2", forKey: "Category_id")
        }
        @IBAction func styleButton(_ sender: UIButton) {
            let storyboard = UIStoryboard(name: "Main", bundle: nil)
```

```
            let viewController = storyboard.instantiateViewController(withIdentifier
:"OrdersController")
            self.navigationController?.pushViewController(viewController,
            animated: true)

            defaults.set("Мода", forKey: "mainTitle")
            defaults.set("5", forKey: "Category_id")
        }

        @objc func favoriteAction(sender: UIButton){
            if self.defaults.bool(forKey: "isSignIn") == true{
                let headers: HTTPHeaders = [
                    "Authorization": "Token " + defaults.string(forKey: "Token")!,
                  "Accept": "application/json"
                ]
                let parameters = ["product" : String(sender.tag)]
                AF.request("**********************************", method:
.post, parameters: parameters, encoding: JSONEncoding.default, headers: headers).re-
sponseJSON { response in
                    switch response.result {
                    case .success(_):
                        let json = try? JSON(data: response.data!)
                        if (json!["status"] == "ok") {
                            if sender.backgroundImage(for: .normal) == UIImage(named:
"heart1"){
                                sender.setBackgroundImage(UIImage(named: "heart"), for:
.normal)
                                GlobalVariables.favorites.remove(at: GlobalVariables.favor-
ites.firstIndex(of: sender.tag)!)
                            }
                            else{
                                sender.setBackgroundImage(UIImage(named: "heart1"), for:
.normal)
                                GlobalVariables.favorites.append(sender.tag)
                            }
                        }
                    case .failure(let error):
                        print(error)
                    }
                }
            }
            else{
```

```
            let alert = UIAlertController(title: "Внимание!", message: "Вы не зарегистрированы! Зарегистрироваться?", preferredStyle: UIAlertController.Style.alert)

            alert.addAction(UIAlertAction(title: "Да", style: .default, handler: { (action: UIAlertAction!) in
                let storyboard = UIStoryboard(name: "Main", bundle: nil)
                let viewController = storyboard.instantiateViewController(withIdentifier :"AuthorizationController")
                self.present(viewController, animated: true)
            }))

            alert.addAction(UIAlertAction(title: "Нет", style: .cancel, handler: { (action: UIAlertAction!) in
            }))

            present(alert, animated: true, completion: nil)
        }

    }

    @objc func basketAction(sender: UIButton){
        if self.defaults.bool(forKey: "isSignIn") == true{
            let headers: HTTPHeaders = [
                "Authorization": "Token " + defaults.string(forKey: "Token")!,
              "Accept": "application/json"
            ]
            let parameters = ["product" : String(sender.tag)]
            AF.request("**********", method: .post, parameters: parameters, encoding: JSONEncoding.default, headers: headers).responseJSON { response in
                switch response.result {
                case .success(_):
                    let json = try? JSON(data: response.data!)
                    if (json!["status"] == "ok") {
                        if sender.backgroundImage(for: .normal) == UIImage(named: "basket1"){
                            sender.setBackgroundImage(UIImage(named: "basket"), for: .normal)
                            GlobalVariables.basket.remove(at: GlobalVariables.basket.firstIndex(of: sender.tag)!)
                        }
```

```
                    else{
```

```
                        sender.setBackgroundImage(UIImage(named: "basket1"), for: .normal)
                        GlobalVariables.basket.append(sender.tag)
                    }
                }
                case .failure(let error):
                    print(error)
                }
            }
        }
        else{
            let alert = UIAlertController(title: "Внимание!", message: "Вы не зарегистрированы! Зарегистрироваться?", preferredStyle: UIAlertController.Style.alert)

            alert.addAction(UIAlertAction(title: "Да", style: .default, handler: { (action: UIAlertAction!) in
                let storyboard = UIStoryboard(name: "Main", bundle: nil)
                let viewController = storyboard.instantiateViewController(withIdentifier :"AuthorizationController")
                self.present(viewController, animated: true)
            }))

            alert.addAction(UIAlertAction(title: "Нет", style: .cancel, handler: { (action: UIAlertAction!) in
            }))

            present(alert, animated: true, completion: nil)
        }
    }

    func getFavorites(){
        let headers: HTTPHeaders = [
            "Authorization": "Token " + defaults.string(forKey: "Token")!,
          "Accept": "application/json"
        ]
        DispatchQueue.global().async {
            AF.request("**********", method: .get, parameters: nil, encoding: JSONEncoding.default, headers: headers).responseJSON { response in
                switch response.result {
```

```
                case .success(_):
                    let json = try? JSON(data: response.data!)
```

Б Қосымшасының жалғасы

```
                    if json != nil {
                        for i in json!.arrayValue{
                            GlobalVariables.favorites.append(i["id"].int!)
                        }
                    }
                case .failure(let error):
                    print(error)
                    self.getFavorites()
                }
            }
        }
    }

    func getBasket(){
        let headers: HTTPHeaders = [
            "Authorization": "Token " + defaults.string(forKey: "Token")!,
          "Accept": "application/json"
        ]
        DispatchQueue.global().async {
            AF.request("**********", method: .get, parameters: nil,
encoding: JSONEncoding.default, headers: headers).responseJSON { response in
                switch response.result {
                case .success(_):
                    let json = try? JSON(data: response.data!)
                    if json != nil {
                        for i in json!.arrayValue{
                            GlobalVariables.basket.append(i["id"].int!)
                        }
                    }
                case .failure(let error):
                    print(error)
                    self.getBasket()
                }
            }
        }
    }

    func getRecomendation(){
        DispatchQueue.global().async {
            let configuration = URLSessionConfiguration.default
```

```
        configuration.timeoutIntervalForRequest = 10 // seconds
        configuration.timeoutIntervalForResource = 10
```

Б Қосымшасының жалғасы

```
        configuration.requestCachePolicy = .useProtocolCachePolicy

        let headers: HTTPHeaders
            headers = ["Connection": "keep-alive"]

        AF.request("**********", method: .get, parameters: nil, encoding: JSONEncoding.default, headers: headers).validate().responseJSON { [self] response in
            switch response.result {
            case .success(_):
                let json = try? JSON(data: response.data!)
                if json != nil {
                    self.orderArray = json?.array as! [JSON]
                    self.mainCollectionView.reloadData()
                    self.activityIndicator.stopAnimating()
                    self.activityIndicator.alpha = 0.0
                }
            case .failure(let error):
                print(error)
                self.getRecomendation()
            }
        }
    }
```

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

Мамандығы 5В070400 – Есептеу техникасы және бағдарламалық қамтамасыз ету

Мүсілімов Дінмұхамед Бақытұлы

Дипломдық жобаға

ТҮСІНІКТЕМЕЛІК ЖАЗБА

Тақырыбы: Жеке тұлғаларға қызмет көрсетуге арналған IOS мобильді әзірлемесі

ҒЫЛЫМИ ЖЕТЕКШІНІҢ ПІКІРІ

Диплом жобасын жасаушы Мүсілімов Дінмұхамед алдына жеке тұлғаларға қызмет көрсетуге арналған IOS мобильді әзірлемесін жасау тапсырмасы қойылған. Дипломдық жобада жеке тұлғаларға күнделікті қолданыстағы құралдарды немесе заттарды жалға беруге арналған кешенді мобильді қосымша құрылған.

Жұмыс барысындағы басты мәселелер: жарнамаға өтініштерді электронды түрде жіберу және қабылдау, өтініштерді мұқият модерациядан өткізу, тексеру әрі жариялау, жалға беруші мен алушы арасындағы қарым-қатынасты қамтамасыз ету, онлайн түрде хабарландыру жіберіп алу уақытын айқындауды қамтамасыз ету болып табылады. Бұл өз кезегінде әлемдік дәрежеде жеке тұлғалар арасындағы уақыт ресурсын үнемдеуде, іздегенін дереу табуға таптырмас мобильді қосымша бола алады.

Менің пікірімше, диплом жазушы алдына қойылған тапсырманы толығымен орындады және ақпараттық жүйелердің заманауи технологияларын меңгергендігін көрсетті.

Жоба жетекшісі ретінде бұл дипломдық жобаны өз деңгейіне сәйкес деп есептей отырып Мүсілімов Дінмұхамедке 5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандығы бойынша «Техника және технологиялар бакалавры» академиялық дәрежесін тағайындауға болады деп есептеймін.

Ғылыми жетекші: «Программалық инженерия» кафедрасының лекторы

 А.М. Кайрбеков «04» 05 2021 жыл

Метаданные

Название
Мүсілімов Дінмұхамед.docx

Автор
Мүслімов Дінмұхаммед

Научный руководитель
Даулет Байымбетов

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	5
Интервалы	A→	0
Микропробелы	·	12
Белые знаки	ß	0
Парафразы (SmartMarks)	a	2

Объем найденных подобий

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

0.21% 0.21% КП1

0.00% 0.00% КП2

1.72% 1.72% КЦ

25
Длина фразы для коэффициента подобия 2

7166
Количество слов

57289
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	**IT-мамандықтарының талапкерлеріне арналған ыңғайлы мобилді қосымшасын жасау** Сапарғалиев Алмат Қалтайұлы **5/14/2018** Satbayev University (ИКиИТ)	15	0.21 %

из базы данных RefBooks (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из домашней базы данных (0.21 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	**IT-мамандықтарының талапкерлеріне арналған ыңғайлы мобилді қосымшасын жасау** Сапарғалиев Алмат Қалтайұлы **5/14/2018** Satbayev University (ИКиИТ)	15 (1)	0.21 %

из программы обмена базами данных (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из интернета (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	ИСТОЧНИК URL	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

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: Мүсілімов Дінмұхамед Бақытұлы, Кайрбеков Абылай

Название: Жеке тұлғаларға қызмет көрсетуге арналған IOS мобильді әзірлемесі

Координатор: Сейтбекова Е.С.

Коэффициент подобия 1: 0.21

Коэффициент подобия 2: 0.0

Замена букв: 5

Интервалы: 0

Микропробелы: 12

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:

.....

Дата 27.05.2021г *Подпись Научного руководитель* 

Форматы	Зонасы	Поз-сы	Белгілеулер	Атаулары	Саны	Қорытынды

					Дипломдық жоба	Бет
Өлш	Бет	Құжат №	Қолы	Күні		62