

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Сәтбаев университеті

Ақпараттық және телекоммуникациялық технологиялар институты

Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

Сапарбай Ажар

«Андройд платформасында Ломбард мобильді қосымшасын әзірлеу»

ДИПЛОМДЫҚ ЖҰМЫС

5B070300 – «Ақпараттық жүйелер» мамандығы

Алматы 2019

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Сәтбаев университеті

Ақпараттық және телекоммуникациялық технологиялар институты

Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
КАӨЖС кафедра меңгерушісі,
техн. ғыл. канд. ,
ассистент профессор
Н.А.Сейлова
« 16 » мамыр 2019 ж.

ДИПЛОМДЫҚ ЖҰМЫС

Тақырыбы: Android платформасында Ломбард мобильді қосымшасын әзірлеу

5B070300 – «Ақпараттық жүйелер» мамандығы бойынша

Орындаған :

Сапарбай А.Н.

Пікір беруші:

PhD докторы, аға ғыл.оқытушы

А. С. Шаяхметова
« 14 » мамыр 2019 ж.

Ғылыми жетекші :

Сеньор-лектор

О.В. Киселева
« 14 » мамыр 2019 ж.

Алматы 2019

Дипломдық жобаны дайындау
КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекші мен кеңесшілерге көрсету мерзімдері	Ескерту
Мәселенің қазіргі жағдайына шолу және оны талдау	11.01.2019 - 08.03.2019	
Ақпараттық қамтаманы құру	06.02.2019 - 10.03.2019	
Программалық қамтаманы құру	12.03.2019 - 28.04.2019	

Дипломдық жұмысының бөлімдерінің кеңесшілері мен
норма бақылаушыларының аяқталған жобаға қойған
қолтаңбалары

Бөлімдер атауы	Кеңесшілер, аты, әкесінің аты, тегі (ғылыми дәрежесі, атағы)	Қол қойылған күні	Қолы
Норма бақылаушы	А.А.Кабдуллин, тьютор	15.05.19	
Программалық қамтама	М.Б. Бауыржан Техн.ғыл.магистр. лектор.	16.05.19	

Ғылыми жетекші  О.В.Киселева

Тапсырманы орындауға алған білім алушы  А.Н.Сапарбай

Күні

«08» 01 2019ж

ҒЫЛЫМИ ЖЕТЕКШІНІҢ СЫН-ПІКІРІ

дипломдық жобасына

Сапарбай Ажар

Мамандық 070300 – Ақпараттық жүйелер

Тақырыбына: **«Android платформасында Ломбард мобильді қосымшасын әзірлеу»**

Берілген дипломдық жұмыста Қазақстан Республикасындағы Ломбардтың жұмыс жасау үрдістері қарастырылған. Алматыдағы Ломбард бөлімшелерінің мекен-жайы, жұмыс уақыты, қызмет көрсету негіздері көрсетілген.

Ломбард бөлімдерінің негізгі көрсеткіштері қарастырылған. Басқа да Ломбард қызметтерінің жұмысымен салыстырылған.

Сонымен қатар тауарды сатушы әртүрлі тауарларды сатылымға қойып, бағасын беріп, кез келген клиентке сатуды жүзеге асыра алады. Немесе қаржылай көмек қажет болған кезде, өзінің техникаларын Ломбард қызметкерімен келісім-шартқа отырып, уақытша мерзімге кепілге қоя алатындығы келтірілген.

Негізгі бөлімде барлық қызметтің түрлері, бағалары, келісім-шарттар, сату және сатып алу жұмыстарының орындалуы толықтай көрсетілген. Қаржылай тиімділіктің көрсеткіштері есептелген.

Еңбек қорғау шаралары бойынша сатып алушының құқығы қорғалған, кепілге қою заңды және ешқандай қиындықсыз орындалады, сату мәселелері көтеріліп, сәйкес есептеулер жүргізілген.

Бітіруші Сапарбай Ажар дипломдық жұмысты жасауда өздігінен жұмыс істеу қабілетін көрсете алды. Дипломант, Сапарбай Ажар алдына қойған инженерлік есептерді шеше алатынын, әдебиеттермен жұмыс істей алатынын көрсетті. Жалпы дипломдық жұмысты 95/А/«өте жақсы» деп бағалап, ал студент Сапарбай Ажар 070300 - «Ақпараттық жүйелер» мамандығы бойынша «бакалавр» біліктілігіне сай деп санаймын.

Ғылыми жетекші

КАӨЖС каф. сеньор лектор

 О.В. Киселева

«19»  2019ж

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӨТБАЕВ УНИВЕРСИТЕТІ

РЕЦЕНЗИЯ

дипломдық жоба

Сапарбай Ажар

Мамандық 5B070300 – Ақпараттық жүйелер

Тақырыбына: **«Андройд платформасында Ломбард мобильді қосымшасын әзірлеу»**

Орындалды:

а) түсіндірме жазбасы 40 бет.

Дипломдық жобада Сапарбай Ажар **«Андройд платформасында Ломбард мобильді қосымшасын әзірлеу»** мәселелерін қарастырған.

Дипломдық жұмыста Ломбард саласын талдау, олардың ерекшеліктері, қызметтері, техникалық тапсырмалары жазылған. Жалпы деректер базасының моделін жасау толық тоқталып, сипатталды. Программалық қамтаманы құру мен іске асыру туралы қарастылып, әр бөлімі тиянақты жасалды.

Бұл дипломдық жұмыс жоғарғы оқу орындарының талаптарына сай жеткілікті жоғары дәрежеде жазылған, алынған нәтижелер ақпаратты өңдеп тарату технологиялардағы ғылыми бағытқа жауап береді.

ЖҰМЫСҚА ЕСКЕРТУ ЖАСАУ

Белгілі бір семантикалық және стилистикалық дәлсіздіктер сияқты шағын кемшіліктер, сондай-ақ әртүрлі әдебиетке деген сілтемелер аз. Алайда, бұл кемшіліктер орындалатын жұмыстардың жалпы деңгейін төмендетпейді.

Жұмыс бағасы

Жалпы дипломдық жұмысты "90" деген бағаға, ал студент Сапарбай Ажар 5B070300 - «Ақпараттық жүйелер» мамандығы бойынша «бакалавр» академиялық дәрежесіне ұсынылады..

РЕЦЕНЗЕНТ

Аға ғылыми қызметкер, PhD доктор А.С.Шаяхметова

«15» мамыр 2019 ж.

Протокол анализа Отчета подобия

заведующего кафедрой / начальника структурного подразделения

Заведующий кафедрой / начальник структурного подразделения заявляет, что ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Сапарбай Ажар

Название: Андроид платформасында Ломбард мобильди косымшасын азырлеу

Координатор: Ольга Киселева

Коэффициент подобия 1:0,1

Коэффициент подобия 2:0

Тревога:4

После анализа отчета подобия заведующий кафедрой / начальник структурного подразделения констатирует следующее:

- ☒ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, работа признается самостоятельной и допускается 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, работа не допускается к защите.

Обоснование:

.....
.....
.....
.....
.....
.....

Дата 16.05.2023

Подпись заведующего кафедрой /

начальника структурного подразделения


И.Б.О.И.И.

Окончательное решение в отношении допуска к защите, включая обоснование:

Вопросе не верну

Дата 16.05.192

Подпись заведующего кафедрой /

начальника структурного подразделения

ИИ
ИИИИИИ

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: Сапарбай Ажар

Название: Андроид платформасында Ломбард мобильди косымшасын азирлеу

Координатор: Ольга Киселева

Коэффициент подобия 1:0,1

Коэффициент подобия 2:0

Тревога:4

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:

запущены в работу не
обязательно

14.05.19

Дата

О.К. —

Подпись Научного руководителя

Краткий отчет



Университет:	Satbayev University
Название:	Андройд платформасында Ломбард мобильди қосымшасын азирлеу
Автор:	Сапарбай Ажар
Координатор:	Ольга Киселева
Дата отчета:	2019-05-06 05:34:25
Коэффициент подобия № 1:	0,1%
Коэффициент подобия № 2:	0,0%
Длина фразы для коэффициента подобия № 2:	25
Количество слов:	4 750
Число знаков:	38 998
Адреса пропущенные при проверке:	
Количество завершенных проверок:	42



К вашему сведению, некоторые слова в этом документе содержат буквы из других алфавитов. Возможно - это попытка скрыть позаимствованный текст. Документ был проверен путем замещения этих букв латинским эквивалентом. Пожалуйста, уделите особое внимание этим частям отчета. Они выделены соответственно. Количество выделенных слов 4

>> Самые длинные фрагменты, определенные, как подобные

№	Название, имя автора или адрес гиперссылки (Название базы данных)	Автор	Количество одинаковых слов
1	URL_ http://adilet.zan.kz/kaz/docs/V17016013_2/compare		5

>> Документы, содержащие подобные фрагменты: Из домашней базы данных

Не обнаружено каких-либо заимствований

>> Документы, содержащие подобные фрагменты: Из внешних баз данных

Не обнаружено каких-либо заимствований

>> Документы, содержащие подобные фрагменты: Из интернета

Документы, выделенные жирным шрифтом, содержат фрагменты потенциального плагиата, то есть превышающие лимит в длине коэффициента подобия № 2

№	Источник гиперссылки	Количество одинаковых слов (количество фрагментов)
1	URL_ http://adilet.zan.kz/kaz/docs/V17016013_2/compare	5 (1)

Copyright © Plagiat.pl 2002-2019

АНДАТПА

Дипломдық жұмыстың тақырыбы: Android платформасында «Ломбард» мобильді қосымшасын әзірлеу

Дипломдық жұмыс мәтіндік баспаның 40 парағын, 35 суретін, пайдаланылған көздер тізімін – 20 атауды қамтиды.

Дипломдық жұмыстың мақсаты - «Lombard». бағдарламасына өтініш жасау. Бұл қарапайым пайдаланушыларға кез-келген ломбардтың орналасқан жерін оңай табуға мүмкіндік береді және фотосурет тауарларын алдын-ала бағалау үшін аударуға мүмкіндік береді және кепілді қамтамасыз ету ретінде жақсы аударымды береді. Және сатып алушыларға кепілдіктерді сатып алу ломбардтың мобильді қосымшасы арқылы тауарларды алуға мүмкіндік береді. Дипломдық жұмыстың мақсатына жету үшін жұмыс бірнеше кезеңде жүзеге асырылды. Жасалатын мобильді қосымшаның қолданыстағы аналогтарын талдау жүргізілді және қолдануға арналған технологиялар мен құралдар зерттелді. Біз Android операциялық жүйесінде жүйелердің үшінші тараптық өзара әрекеттесу қағидаттарын терең түсіндік.

АННОТАЦИЯ

Тема дипломной работы Разработка мобильного приложения «Lombard» на платформе Android

Дипломная работа содержит 40 страниц машинописного текста, 35 рисунков, список использованных источников - 20 наименований.

Цель дипломной работы является создание приложения для компании «Lombard +». Это позволит простым пользователям с легкостью узнавать местоположение любого ломбарда и позволит передавать фототоваров для предварительной оценки и дальнейшего передачи в залог. А скупщикам залогов позволит следить за поступлением товаров через мобильное приложение ломбарда. Для достижения цели дипломного проекта работа выполнялась в несколько этапов. Был проведен анализ существующих аналогов создаваемого мобильного приложения и исследовались технологии и инструменты для разработки приложения. Получен глубокий опыт и понимание принципов стороннего взаимодействия систем в операционной системе Android.

ANNOTATION

Theme of the thesis: Development of the mobile application "Lombard" on the Android platform

The diploma project contains 40 pages of typewritten text, 35 drawings, a list of sources used – 20 titles.

The purpose of the thesis is to create an application for «Lombard». This will allow simple users to easily locate the location of any pawnshop and will allow the transfer of photographic goods for preliminary evaluation and give the best transfer as collateral. And to buyers of pledges will allow you to follow the receipt of goods through the mobile application of the pawnshop. To achieve the goal of the diploma project, the work was carried out in several stages. An analysis of existing analogues of the mobile application being created was conducted and technologies and tools for application development were explored. We gained deep experience and understanding of the principles of third-party interaction of systems in the Android operating system.

МАЗМҰНЫ

	КІРІСПЕ	9
1	АНАЛИТИКАЛЫҚ БӨЛІМ	11
1.1	«Ломбард+» компаниясымен танысу	11
1.2	Ломбардтың қызметін автоматтандыру	12
1.3	Карталарда іздеу жұмысының функционалдылығы мен маңыздылығы	13
1.4	Аналогтық талдау	20
1.5	Тапсырмаларды орнату	20
2	ТЕОРИЯЛЫҚ БӨЛІМ	22
2.1	Бағдарламалау тілдерімен танысу және таңдау негіздері	22
3	Практикалық бөлім ПРАТИКАЛЫҚ БӨЛІМ	33
3.1	Қосымшаны құру процесі	33
3.2	Қосымшаның жұмысымен танысу	33
	ҚОРЫТЫНДЫ	42
	ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ	43
	А қосымшасы	44

КІРІСПЕ

Қазақстанда мемлекеттік қарыз алушыларға арналған көптеген ломбардтар құрыла бастады, бұл қарыз алушыға қысқа мерзімге және белгілі бір пайызға несие берді.

1992 жылғы 29 наурыздағы «Кепіл туралы» Заңды қабылдау арқылы ломбардтардың қызметін мемлекет мойындады. Ол оның қоғамдағы қажеттіліктеріне негізделген. Нәтижесінде, жеке ломбардтар ұйымдастырыла бастады.

Кейінірек 01.01.1995 ж. Бастап Қазақстан Республикасының Азаматтық кодексінің бөлігі ломбардтың қызметіне тікелей сілтеме жасап, ол түзетілді.

Ломбард операцияларын есепке алу және салық салу ұйымдардың осы тобында ерекше орын алады. «Ломбардтың қызметі» концептуалдық шекарасы, өнертабыс ережелерінің ережелерінен анықталады. Қазақстан Республикасы Азаматтық кодексінің 358 және 919-баптарына сәйкес әзірленді. Бұл қызмет, ең алдымен, азаматтардың жылжымалы мүлікімен қамтамасыз етілген несиелерді беруден және халыққа тиесілі мүлікті сақтаудан тұрады.

Өртүрлі мобильді гаджеттердің пайдаланушылары кеңінен таралған, және бүгінгі күннің өзінде көптеген оқушыларда смартфондар немесе планшеттер бар. Әсіресе, қашықтағы платформаларда өз бизнесінің бір бөлігін құратын іскер адамдарға қатысты. Бүгінгі күні гаджеттерді қамтамасыз ететін мүмкіндіктер тізімін айтарлықтай кеңейтті. Әлеуметтік желілерге смартфондар мен iPhone арқылы іздеу жүйелеріне қол жеткізу арнайы қосымшалар арқылы жүзеге асырылады. Өте үлкен орынды ойын мобильді қосымшалары иеленеді, бұл кез-келген жерде пайдалана алуға мүмкіндік береді. Дегенмен, арнайы мобильді қосымшалар коммуникация мен ойын-сауықпен қатар, әр түрлі бизнес-міндеттерді орындауға мүмкіндік береді. Бизнес иелері әрқашан кеңейтуге және қосымша пайда алуға көмектесетін шешімдер іздейді. Осы мүмкіндіктердің бірі - мобильді гаджеттерді пайдалану арқылы бизнесті жүргізуге мүмкіндік беретін арнайы бағдарламаларды әзірлеу. Бұл ұялы құрылғылардан үлкен сандарда орындалған корпоративтік сайттарға баруды талдау кезінде түсінікті болады. Бұл бизнестің кеңеюі үшін серіктес және тұтынушы гаджеттерінің әлеуетті пайдалануының жоғары деңгейін көрсетеді.

Мобильдік қосымшаларды сапалы және жоғары сапалы әзірлеу - бұл қажетті жұмыс түріне және көлеміне байланысты уақытты талап ететін ұзақ процесс, сондықтан мобильді құрылғыларға бейімделу бірнеше аптаға созылуы мүмкін және тәуелсіз бизнес құралдарын жобалауға және іске асыруға байланысты күрделі жұмыстар талап етілуі мүмкін. Серверлік-клиенттік шешімдердің дамуы күрделі және ұзақ тапсырма болып есептеледі, себебі ол сонымен бірге серверлік бөліктің жобалау жұмыстарына да әсер етеді. Тәжірибе көрсеткендей, ұялы гаджеттерге арналған бизнес-қосымшаларды әзірлеуге жұмсалған қаражаттар оларды пайдалану ыңғайлылығы мен клиенттік базаның кеңеюіне байланысты өздігінен төлейді. Сондықтан дамудың бұл түрі кәсіпкерлердің санының артуына алып келеді.

Дипломдық жұмыстың тақырыбы: Android платформасында «Ломбард » мобильді қосымшасын құру.

Мақсаты: Ломбард қосымшасын жасаңыз. Бұл қарапайым пайдаланушыларға кез-келген ломбардтың орналасуын оңай анықтауға мүмкіндік береді және алдын-ала бағалау және кепілдік беру үшін тауарлардың фотосуреттерін аударуға мүмкіндік береді. Ал ипотека сатып алушылары ломбард ұялы қосымшасы арқылы тауардың түсуін бақылай алады.

Дипломдық жұмыстың жаңалығы: ломбардтың алғашқы өтінішін жасау. Қолданбаны жасаған кезде Ionicframework және javascript, typescript және mongoDB бағдарламалау тілі дерекқор ретінде пайдаланылады.

1 АНАЛИТИКАЛЫҚ БӨЛІМ

1.1 Ломбард + ломбардымен танысу

«Ломбард +» компаниясы 1995 жылы құрылған және клиенттердің үнемі өсіп келе жатқан қажеттіліктерін қанағаттандыруға тырысып, өз клиенттерімізге несиелендіру мен қызмет көрсету сапасын арттыру міндетін қойып, қаржы институтын белсенді түрде қалыптастырады. Компанияның негізгі қызметі зергерлік бұйымдармен және электроникамен қамтамасыз етілген тұлғаларға қысқа мерзімді несиелер беру болып табылады. Компания Қазақстанның барлық қалаларында филиалдар желісі болып табылады.

Компанияның негізгі құндылығы оның қызметкерлері және корпоративтік мәдениеті болып табылады. Ломбард + қызметкерлерге мансаптық мүмкіндіктер мен дағдыларды дамытуға, лайықты жалақыға, ұзақ мерзімді болашаққа және тұрақтылықты қамтамасыз етеді. Осы мақсатта ол әр қызметкер өз мансабын дамытуға және жоспарлауға қатысатын «Мансапты басқару орталығы» бағдарламасының кадрлық резервін құрып, белсенді түрде жұмыс жасап жатыр.

Компанияның әлеуметтік пакетіне мыналар кіреді:

- мобильдік төлем;
- қызметкерлерге арналған жылдық бонус;
- медициналық сақтандыру;
- қызметкерлердің өміріндегі елеулі оқиғалар кезінде (туған күндер және т.б.) қаржылық көмек көрсету;
- пайызсыз.

2012, 2013, 2014 және 2015 жылдары Қазақстанның Ұлттық тұтынушылар лигасының Ломбард + ұлттық сапа белгісіне ие болды.

2013 жылы «Қазақстан» HR-брендін үшін «Ломбард +» номинациясы бойынша «Қазақстан» санатында ІІІ орынды жеңіп алып, «Жеңімпаздық шешім: тарту және ұстау» номинациясы бойынша жеңімпаз атанды. 2013 жылғы HR-брендін марапатының бас серіктесі Smart Solutions. Алматылық ломбардтың ұзақ уақыттан бері ең маңызды аспектісі әр клиентке өте құзыретті көзқарас болып табылады және ол өте күрделі уақытта қарыз алушыға көмектесу болып табылады, әсіресе клиенттер үшін дәл және түсінікті шарттар жасалады.

Ломбард + та тек қана білікті мамандар ғана жұмыс істейді, бұл құжаттармен және қарыздармен ең тиімді және жоғары сапалы операция жасауға мүмкіндік береді.

1.2 Ломбардтың қызметін автоматтандыру

Ломбард + мобильді қосымшасы ломбардтың жұмысын автоматтандыруға арналған. Ломбардтың барлық операцияларын есепке алу туралы шешім қабылданды:

- мүлікті кепілдік ретінде алу;
- талдау;
- қарыз беру;
- борыштарды пайдаланғаны үшін сыйақы есептеу және жинау, заттарды сақтау үшін өтеу;
- кепіл берушінің қарызын өтеу;
- аукционнан квалификациялық емес заттарды сатуға, сондай-ақ құндылықтарды сатып алу;

қамтамасыз ету / қарыз / займ бойынша жұмыспен қамтамасыз ету;

- филиалдар желісінде жұмыс істеу мүмкіндігі; сауда жүйесі;

- құндылықтарды сатып алу жүйесі;
- есеп беру.

Негізгі мүмкіндіктер:

- Ломбард филиалдарының желісінде жұмыс істеу;
- Интернеттегі деректерді ұйымдастыруды өзгерту;
- Құжат нөміріне басқа префиксті пайдалану;
- кассирге ақшалай қаражаттардың қозғалысын аналитикалық есепке алуды жүргізу;

- Федералдық қаржылық бақылау қызметімен байланыс;
- қамтамасыз ету / қарыздар / ломбардтық несиелер бойынша жұмыстар;
- қатаң есептілік нысандарын есепке алу (bso);
- кешенді билет нөмірлерін автоматты түрде жасау;
- бағалы қағаздар бойынша ломбардтың негізгі операцияларын тіркеу;
- несиелік сұлбаларды дұрыс реттеу;
- кепіл туралы ақпаратты сақтау;
- файлдар мен суреттерді сақтау;
- билет үшін мәмілелерді жасағанда басқа да қолдаушы ақпаратты көрсету;

- Қажетті типті басып шығару үшін қауіпсіздік нысанын оңай орнату;
- автоматтандырылған жұмыс орнында сарапшы тауарлар;
- қауіпсіздік билеттерінің тізімін және олардың негізгі параметрлерін көрсету;

- Таңдалған кепіл берушімен контакт тарихына қатысты ақпараттың қысқаша сипаттамасы;

- қолданыстағы билетте мәмілелерді енгізу және алдын-ала жасалған мәмілелерді көрсету мүмкіндігі;

- түрлі сайлауды қолдану мүмкіндігі;
- Ломбард Ломбард алаңы;

- мерзімі өткен тауардың сауда-саттықсыз жеңілдетілген сату мүмкіндігі;
- ломбардта құндылықтарды сатып алудың кіші жүйесі;
- арнайы есеп беру;
- менеджерге арналған ломбардтың монитори;

1.3 Карталарда іздеу жұмыстарының функционалдылығы мен маңыздылығы

Карталар түрлері:

Бұл құжат Google Maps JavaScript API-лері арқылы көрсетілетін карталардың түрлерін сипаттайды.

MapType нысаны осы карталар туралы ақпаратты сақтау үшін API-ді пайдаланады.

MapType - бұл карта парақтарының көрсетілуін және пайдаланылуын анықтайтын интерфейс, сондай-ақ экранның координаттарын картаның географиялық координаттарына түрлендіру. Әрбір MapType нысанында түбіртектерді өңдеу және карталар парақтарын жоюдың бірнеше әдістері, сондай-ақ оның көрнекі әрекетін анықтайтын функциялар болуы керек. JavaScript Maps API ішіндегі карта түрлерінің ішкі құрылымы күрделі мәселе. Көптеген әзірлеушілер төменде көрсетілген карталардың негізгі түрлерін ғана пайдалана алады. Сонымен қатар, карта түрлерін пайдалана отырып, карталарды көрсете аласыз немесе қолданыстағы түрлердің көрсетілімін өзгертуге болады.

Негізгі карталар түрлері

Google Карталары JavaScript API карталардың төрт түрін ұсынады. Таныс «сызылған» жол карталарынан басқа Карталар үшін JavaScript API-да басқа түрлері де қолданады.

Карталар JavaScript API-де келесі карталар түрлері бар:

- Жол картасы - жол картасының стандартты өкілдігі. Бұл түрі әдепкі бойынша пайдаланылады;
- спутниктік - Google Earth спутниктік картасында;
- дәстүрлі және спутниктік карталардың гибриділік комбинациясы;
- Жер, жер туралы деректерге негізделген физикалық карта.

Пайдаланылатын картаның түрін өзгерту үшін Map объектісінің TypeId сипатын өзгертуіңіз керек. Сіз мұны әзірлеуші тарапынан, картаңыздың нысанын өзгерту арқылы немесе осы карта үшін setMapTypeID () әдісін шақыру арқылы жасай аласыз. Әдепкі mapTypeID сипаты - бұл жол картасы.

Карта жасаған кезде PropertyTypeId сипаты:

```
var=myLatLng=newgoogle.LatLng(-34.396,150.654);
var=mapOptions={
zoom:9,
centere:my_LatLng,
map_TypeId:'satellite'
```

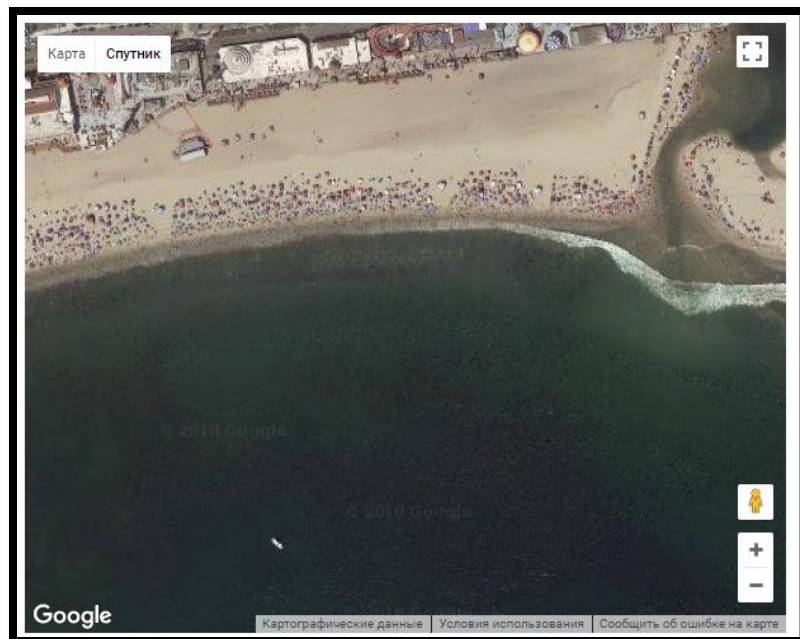
`Var=map=newgoogle.Мар`

Динамикалық `mapTypeId` сипатын өзгерту:

`map.setMapTypeId('terrain');`

Карта түрін орнатпағаныңызды, бірақ `MapType`-ке идентификаторды қолданатын картаның `mapTypeId` сипатын көрсетіңіз. Бұл сілтемелерді басқаруға арналған Карталар JavaScript API төменде сипатталған карта тізілімінің түрін пайдаланады. Google Maps JavaScript API нақты аймақтар үшін 45° бейнелерді қолдайды. Бұл жоғары ажыратымдылықтағы суреттер әр негізгі бағыттың (солтүстік, оңтүстік, шығыс, батыс) перспективалық көрінісін көрсетеді. Қолдау көрсетілетін карта түрлері үшін бұл кескіндер үлкен ауқымда қол жетімді. 1.1-суретте Санта Крус бұрышында тротуар көрсетілген.

Спутниктердің және гибриді карталардың түрлері бұл опция қол жетімді болған кезде қуатты күшейіп, 45° бұрышта бейнені қолдайды. Пайдаланушы осындай суреттер бар аймақты кеңейтсе, карта көріністері осы түрлерін төмендегідей автоматты түрде өзгертеді. Спутниктік немесе гибридік кескіндер ағымдағы орналасқан жеріңізде 45° бейнелермен ауыстырылады. Әдеттегі бойынша бұл фотосуреттер солтүстікке арналған. Егер пайдаланушы зонаға түссе, спутниктік немесе гибридік бейнелер әдеттегідей бойынша қайта пайда болады. Санта-Крус тротуарының көрінісі 1.1 суретте көрсетілген. Бұрылыспен басқару бұрышты және айналмалы карталарды реттеуге мүмкіндік береді. Егер `rotateControl` шын болса, көлбеу басқару элементі 45° сурет үшін көрсетіледі. Бұл басқару элементі кескінді 45° бұрышпен бұруға мүмкіндік береді.



Сурет 1.1 - Санта Крус көшесінің түрі

Сурет қисайған кезде, экранда көрсеткіш сызық 90° бұрылуға мүмкіндік береді.

Картаның өлшемі 45 ° дейін төмендегенде, бұл өзгерістердің барлығы жойылады және бастапқы карта түрі қалпына келтіріледі. Карта нысанында setTilt (0) әдісін шақыру арқылы 45 ° бұрышта суреттерді өшіруге болады. Қолдау көрсетілетін карта түрлеріне арналған 45 ° бейнені қосу үшін setTilt (45) әдісіне қоңырау шалуыңыз қажет.

Map объектісінің getTilt () әдісі әрқашан кескіннің ағымдағы көлбеуін картада көрсетеді. Егер кескінді қисайту үшін орнатсаңыз және оны жойсаңыз (мысалы, картаның өлшемі бойынша), getTilt () картасының әдісі 0 мәнін қайтарады. 45 ° бұрыштағы сурет әрбір негізгі бағытта (солтүстік, оңтүстік, шығыс, батыс) суреттер жиынтығынан тұрады. Картада 45 ° бұрыштағы суреттер көрсетілсе, оларды Map объектісіне setHeading () әдісін шақыру және солтүстіктен ауытқу деңгейіне сәйкес келетін сандық мәнді беру арқылы негізгі бағыттардың біріне бағыттауға болады.

Төмендегі мысалда түйме басылғанда автоматты түрде әрбір 3 секунд сайын ауысатын аэрофотосъемкамен біріктірілген картаны көрсетеді.

Карточкалық тіркеу түрін өзгерту

mapTypeId - бірегей мәнді MapType түріне біріктіру үшін пайдаланылатын жол идентификаторы. Әрбір Map объектісі MapTypeRegistry тіркелушісі болып табылады, ол осы карта үшін қол жетімді MapType түрлерінің жиынтығын қамтиды. Бұл тізбе MapType картасын басқару үшін қол жетімді карталардың түрлерін таңдау үшін пайдаланылады. Оқу карта түрінің тіркелімінен тікелей жұмыс істемейді. Оның орнына тізбе өздерінің карталар түрлерін қосу және таңдалған идентификаторлар ауқымына байланыстыру арқылы өзгертіледі. Карталардың негізгі түрлерін өзгерту немесе өзгерту мүмкін болмайды (картаTypeControlOptions параметрінің көрінісін өзгерту арқылы картадан жойылуы мүмкін болса да).

Келесі мысал MapTypeControlOptions параметрін пайдаланып картаның екі түрін ғана көрсету үшін және нақты MapType іске асыру интерфейсіне идентификаторды қосу үшін тізімді өзгерту әдісін көрсету жолын көрсетеді.

Карта стильдері

StyledMapType жолдар, саябақтар және даму аймақтар сияқты элементтердің сурет стиліндегі өзгерістерге негізделген стандартты Google карталарының көрінісін теңестіруге мүмкіндік береді.

Жеке карталар түрлері

Google Maps JavaScript API енді жеке карталарыңызды немесе картадағы мозаикалық қабаттасуларды пайдалануға мүмкіндік беретін өзіңіздің карталар түрлерін көрсетуді және басқаруға қолдау көрсетеді.

JavaScript Maps API ішіндегі көптеген мүмкін карталарды енгізу түрлері бар.

Суреттерден тұратын карталардың жалпы жиынтығы тұтас картаны білдіреді. Парақтардың бұл бағандары негізгі карталар түрлері деп аталады. Бұл карталар түрлері бар әдеттегі карталар түрлеріне ұқсас:

- roadmap;
- satellite;
- hybrid;

– terrain.

Картаның картасына жеке түрін қосуға болады. Түрлердің массиві Maps API интерфейсында пайдаланушы интерфейсі өзіңіздің карта түрін стандарт ретінде түсіндіре алады (мысалы, оны MapType басқаруымен біріктіру).

Мозаика кескіндері негізгі карталар түрлерінің үстінен көрсетіледі. Әдетте бұл карталардың түрлері қолданыстағы түрлерге жарамды және қосымша ақпаратты көрсету үшін пайдаланылады. Оларды пайдалану белгілі бір аудандарда және масштабтарда жиі шектеледі. Мозаика қабаттары мөлдір болуы мүмкін, сондықтан сіз бар карталарға элементтер қосуға болады. туралы ақпаратты көрсетуді ең негізгі деңгейде өзгертуге мүмкіндік беретін графикалық емес карта түрлері.

Бұл параметрлердің әрқайсысы MapType интерфейсін жүзеге асыратын сынып құруына негізделген. Сонымен қатар, ImageMapType класы MapType кескіндерін жасауды жеңілдететін кіріктірілген функционалдығы бар.

Картаны үйлестіру

Google Maps JavaScript API бірнеше координаттық жүйені қолданады.

Кез-келген географиялық нүктені бірегей сәйкестендіруге мүмкіндік беретін ендік және бойлық шамалары. (Google WGS84 әлемдік стандартты геодезиялық жүйені қолданады). Белгілі бір шкаламен картадағы нақты парақты көрсететін карта парағының координаттары.

Пиксельдің координаттары

Әлемнің координаттары көрсетілген проектордағы абсолютті позицияларды көрсетеді, бірақ масштабтаудың белгілі бір деңгейінде пикселдерді жоюды анықтау үшін олар пикселдердің координатына айналуы керек. Пиксельдің координаттары келесі формула бойынша есептеледі:

$$Pixel_Coordinate = world_Coordinate * 3zoomLevel$$

Жоғарыда келтірілген тендеудің көмегімен масштабтаудың әрбір келесі деңгейі x және y осьтерімен екі есе артық екенін көруге болады. Осылайша, әрбір кейінгі масштабтау масштабы алдыңғы төрт еседен жоғары. Мысалы, 1 ауқымының масштабында картада 4 парақ бар, олар 256x256 пикселді өлшейді, яғни, пиксел аймағы - 512x512. Масштабтау деңгейі 19 болған кезде, картадағы әрбір пиксель x және y мәнін 0-ден 256 * 219-ға дейін анықтауға болады. Әлемдік координаттар карталар парағының өлшеміне негізделгендіктен, пиксель координаттарының бүтін бөлігі қазіргі кездегі масштаб деңгейіне нақты пиксельді анықтайды. 0 ауқымында пикселдердің координаттары әлемдегі координаттарға сәйкес келеді.

Google Maps JavaScript API осы орталықтың көру ортасын картаның сәйкес деңгейінде (LatLng) және оның құрамындағы DOM өлшемінде жасайды және осы аймақтың шекараларын пикселдер координатына түрлендіреді. API содан кейін көрсетілген пиксель шектеріндегі карталардың барлық парақтарын логикалық түрде анықтайды. Осы парақтардың әрқайсысына сілтеме картаның суреттерін көрсетуді жеңілдететін парақ координаттары арқылы жіберіледі.

APIJavaScriptGoogleMaps әдетте пайдаланылатын ең жоғары стандартты масштаб деңгейімен барлық карталар суреттерін жүктей алмайды. Оның орнына, APIJavaScript Карталар нақты бағдарламада логикалық түрде орналасқан карталар жиынын жиынтығында әрбір масштабта сурет жасайды. Картаны жаңа орынға жылжытқанда немесе ұлғайту деңгейін өзгерткен кезде MapsMapsJavaScriptAPI қажетті парақтарды анықтау үшін пиксель координаттарын пайдаланады және осы мәндерді алу керек тізімге түрлендіреді. Парақтың осы координаттары кез-келген нүктеге бейнелерді қамтитын парақтардың мәнін логикалық түрде жеңілдететін схема арқылы көрсетіледі. APIJavaScriptGoogleMaps әдетте пайдаланылатын ең жоғары стандартты масштаб деңгейімен барлық карталар суреттерін жүктей алмайды. Оның орнына, APIJavaScript Карталар нақты бағдарламада логикалық түрде орналасқан карталар жиынтығында әрбір масштабта сурет жасайды. Картаны жаңа орынға жылжытқанда немесе ұлғайту деңгейін өзгерткен кезде MapsMapsJavaScriptAPI қажетті парақтарды анықтау үшін пиксель координаттарын пайдаланады және осы мәндерді алу керек тізімге түрлендіреді. Парақтың осы координаттары кез-келген нүктеге бейнелерді қамтитын парақтардың мәнін логикалық түрде жеңілдететін схема арқылы көрсетіледі. Google Карталарындағы кестелер пикселдермен қатар есептеледі. Google Mercator болжауында, бастапқы парақ әрқашан картаның солтүстік-батыс бұрышында орналасқан, ал x мәндері батыстың шығысына қарай өсіп, солтүстіктен оңтүстікке қарай бағаның өсуі. Кестелер осы нүктеге байланысты x , y координаттары арқылы индекстеледі. Мысалы, 1.3 суретте көрсетілгендей, масштабтау 2 деңгейінде жер 16 параққа бөлінген, әр парақ бірегей x , y координаттары бойынша анықталады:



Сурет1.2 Жер арнайы X координаттары арқылы белгіленеді

Әлемдік координаттар

Google карталары JavaScript API географиялық нүктені картаға бір нүктеге айналдыру қажет болғанда, ең алдымен, ендік пен бойлық мәндерін «әлемдік» координаттарына түрлендіру қажет. Бұл өзгеріс картаның проекциясы арқылы жасалады. Осы мақсатта Mercator проекциясы Google

Maps қызметінде пайдаланылады. Сондай-ақ, google.maps.Projection іске қосу арқылы өз жобаңызды анықтауға болады. Пиксельдің координаттарын есептеуге ыңғайлы болу үшін, 0 масштабтағы масштабтау картасы - негізгі өлшем парағы. Содан кейін әлемнің координатасы пикселдердің координаталарына қатысты масштабтау деңгейінде ендік пен бойлық мәндерін негізгі беттегі пикселдер жағдайына айырбастау үшін проекция көмегімен анықталады. Google Maps картасындағы әлем координаттары Mercator проекциясының бірінші нүктесінен (картаның солтүстік-батыс бұрышы 180 градустық бойлық және ендік 85 градус) өлшенеді және шығыс (оң) бойымен x осімен және оңтүстіктегі (төмен) осьтері бойынша артады. Mercator проекциясының негізгі Google Maps беті 256 x 256 пиксель болғандықтан, әлемдегі ең пайдалы үйлестіру кеңістігі - {0-256}, {0-256}. 1.2-сурет X Y координатасын көрсетеді.

Mercator проекциясында карта бөлімінің белгілі бір ені бар, бірақ ендік биіктігі бойынша белгіленуі керек. Меркатортордың квадрат формасындағы ені 85 ° проекциясын қолданып, парақтың логикасын таңдауды жеңілдете отырып, картаның кескінін «қиып аламыз». Мысалы, орналасулар полюстарға жақын жерде көрсетілсе, карта координаттардың пайдалы координаталар аймағынан тыс әлемге көрсетілуін қамтамасыз етеді.



Сурет 1.3 - XY Координатасы

Ескерту: егер сіз пиксель координатын парақ өлшемімен бөліп, нәтиженің толық бөлігін алсаңыз, сіз ағымдағы масштабтау деңгейінде парақ координатын аласыз.

MapType интерфейсі

Карта түрлерімен MapType интерфейсі іске асырылуы керек. Бұл интерфейс API ағымдағы көру терезесінің және қазіргі масштабтаудың деңгейіне арналған карталардың парақтарын көрсетуі қажет болған кезде, API сұрауларыңызды картаңыздың түрлеріне жіберуге мүмкіндік беретін кейбір

функциялар мен әдістерді көрсетеді. Қолданысыңыз осы сұрауларды өңдейді және жүктелетін парақты анықтайды.

Тасымалдау түрлері

Кейбір карталар түрлері басқа да бір карта түрін алмастыру үшін жасалды. Осы карталарда қолданушыларға арналған қосымша мәліметтер болады және ол мөлдір қабаттан тұрады. Мұндай жағдайларда картаның түрімен жеке объектіде айналысуға болмайды. Оның орнына, сіз оны MapType объектісіне тікелей Map объектісінің overlayMapTypes сипатын пайдаланып қосуға болады.

Бұл сипат MVCArray MapType нысандарының жиынын қамтиды. Карталардың барлық түрлері (база және тасымалдау) карта қабатының картасында көрсетіледі. Overlays олар Map.overlayMapTypes массивінде орындалған тәртіптегі негізгі карталардың үстінде көрсетіледі. Келесі мысал алдыңғы мысалмен бірдей, MapType қабатын қоспағанда ROADMAP картасында жасадық:

Мозаика карталарының түрлері

Карта түрін негізгі картаның түрі ретінде қолдану үшін көптеген сағаттар мен ресурстарды талап етеді. API интерфейсі MapType интерфейсін жеке кескін файлдары бар парақтардан тұратын ең көп таралған карталар түрлеріне орындайтын арнайы сынып ұсынады. ImageMapType деп аталатын бұл сынып ImageMapTypeOptions Specification Image нысанын пайдаланып жасалды, ол келесідей қажетті функцияларға сілтеме жасайды:

tileSize (міндетті) - парақтың өлшемін анықтайды (мысалы, google.maps.Size). Парақ тікбұрышты болуы керек, бірақ міндетті түрде шаршы болмауы керек.

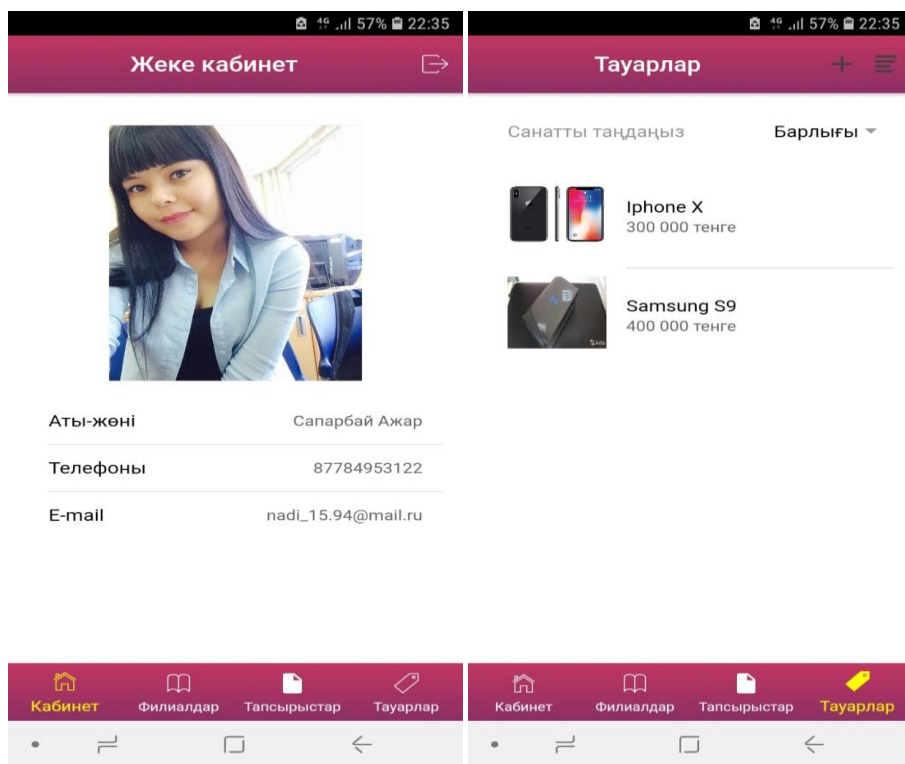
getTileUrl (талап етіледі) - әлемнің белгілі координаттарына негізделген сәйкес суреттер парағын және масштабтау деңгейін тандауды жеңілдету үшін функцияны (әдетте, жолдағы функцияны) анықтайды.

Бұл код Google Карталарды бір ай ішінде ImageMapType негінде іске асырылуын көрсетеді. Бұл мысалда, қалыпқа келтіру функциясы парақтарды картаның x осімен қайталау үшін пайдаланылады, бірақ y осі жоқ.

Осындай нарықта көптеген бағдарламалар бар. Бірақ бізге өтініш жасау керек.

1.4 Анализді талдау

Ломбард: Жеке кабинет. (1.4 суретке сәйкес)



1.4-сурет - «Ломбард жеке кабинеті» қосымшасы

Ломбардқа мүлікті кепілге салу кезінде сіз кепілдік бойынша қосымша төлемдер жасай аласыз.

Қосымша сізге кепілдікке қатысты барлық маңызды нәрселерді, мысалы өтеу мерзімін немесе қарыздың кешігуін жіберуге мүмкіндік бермейді, бірақ бұл жерде сіз өзіңіздің несиелеріңіз, өткен және алдағы өтетін оқиғалар туралы толық ақпараттарды көре аласыз.

Қалай пайдалануға болады:

Депозиттік билетті қосымшаға қосу өте оңай: кепілдік билеті түрінен бірегей кіру кодын енгізу жеткілікті. 1С: Кәсіпорын 8.VDGB: Ломбард бағдарламасын қолдана отырып, ломбардтармен ақпарат алмасуды қолдайды.

1.5 Тапсырманы орнату

Мақсаты - көп функционалдылықты, көп нұсқалы, икемділік пен тиімділікті қамтамасыз ететін жаңа ақпараттық технологияларды қолдану арқылы ломбардтың мобильдік қосымшасын құру.

Шарттары:

Жүйе - бұл технологияның, ұйымның, аппараттық және бағдарламалық қамтамасыз етудің сипаттамасын, ақпаратты және конверсиялау технологияларын беруді, алгоритмді қолдауды қамтитын прототип.

Ішкі материалдар.

Стандарттық бағдарлама қамтамасыз ету.

Критерийдің тиімділігін енгізу:

Уақытты азайту, кепілдік мүлікті бағалау;

- хабардарлықты арттыру;

Сатып алушылар үшін интернет-дүкен құру;

- ломбард пен адам арасындағы келісімнің ыңғайлылығы

Талап етіледі:

- даму ортасын дайындау;

Деректерді сақтау үшін дерекқорды құру;

- Ломбард компаниясына мобильдік қосымшаны құру;

- тұтынушылар үшін ақпараттық ортаны құру;

- ломбард үшін «алдын-ала бағалауға» мүмкіндік жасау;

Сатып алушыларға, фото тауарларға арналған интернет-дүкен құру;

Ionic framework және android studio негізінде түпнұсқа мобильді қосымшаны әзірлеу;

Қосымшаны әзірлеу.

Мәселелерді шешу әдістері қазіргі заманғы ақпараттық технологияларды басқару теориясы әдістерін, өлшеу дәлдігін жақсарту, шудан қорғау әдістерін және динамикалық бағалар мен уақыттық сериялар туралы деректерді өңдеу технологияларын кеңінен қолдану болып табылады.

«Ломбард» қосымшасының артықшылығы: меншікті қосымшаны платформалар алаңында және mongoDB дерекқорында жасау. Сонымен қатар, артықшылығы - сіздің үйден шықпай-ақ мүлікіңізді тексеру мүмкіндігі.

2 ТЕОРИЯЛЫҚ БӨЛІМ

2.1 Бағдарламалау тілдерімен танысу және таңдау негіздері

Мобильді қосымшаны әзірлеу үшін жылдам және тиімді даму үшін бірнеше бағдарламалар мен құралдар қажет.

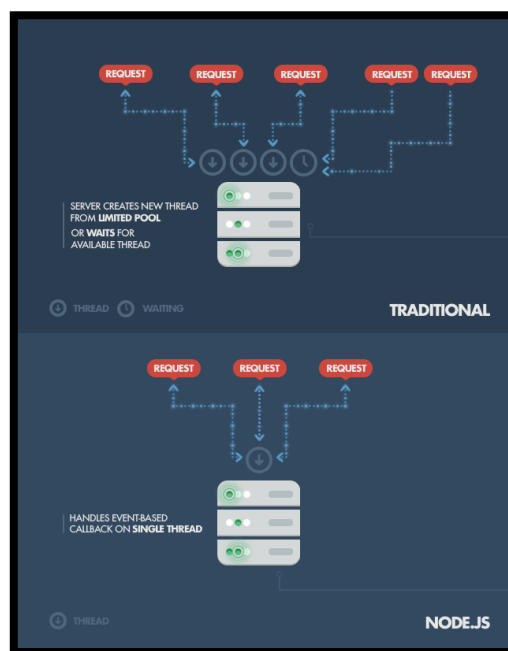
Node.js - Google V8 JavaScript механизмін қамтитын пакет, платформаның абстракциялық қабаты libuv кітапханасы және JavaScript-де дәл жазылған негізгі кітапхана. Node.js-тің авторы Райан Дал, нақты уақыт режимінде жұмыс сайттарын құруға және Gmail сияқты қосымшалардан бас тартатын функцияны қамтамасыз етуге тырысады. Node.js-те ол бағдарламашыларды оқиғаларға шоғырландырылмаған күту құралдарымен қамтамасыз етті. Node.js нақты уақыттағы бағдарламаларда жұмыс істейді, себебі ол веб-розеткалар арқылы push технологиясын қолданады. Революционер дегеніміз не? Клиентке және серверге қосылуға болатын қос бағыттық бағдарламалар пайда болады, содан кейін деректердің еркін алмасуын жалғастырады.

Бұл технология клиенттің әрқашан байланыс жасайтын Интернеттің стандартты ұсынуындағы айырмашылығымен ерекшеленеді. Сонымен қатар, барлық технология ашық веб-стекке (HTML, CSS және JS) негізделген, тапсырма стандартты порт 80 арқылы өтеді.

Node.js, өзінің барлық артықшылықтарымен қазіргі уақытта сайттың бірегей қасиеттеріне тікелей тәуелді, қалған көптеген компаниялардың технологиялық стекінде маңызды рөл атқарады. Node.js-дің ең маңызды идеясы үлкен деректерді нақты уақытта өңдейтін және таратылған құрылғыларда жұмыс істейтін қолданбалармен жұмыс істеу кезінде жеңіл және тиімдірек болу үшін бұғатталған оқиғалардың әрекеттерін пайдалану болып табылады. Негізінде, бұл Node.js веб дамыту әлемінде басым болатын барлық жағдайлар үшін платформа емес дегенді білдіреді. Екінші жағынан, бұл қатаң белгіленген тапсырмаларды шешуге арналған алаң. Бұны түсіну өте маңызды болып табылады. Әрине, Node.js-те қатты жүктелген процессор жүктемелерін пайдаланбауыңыз керек, сонымен қатар күрделі есептеулерде Node.js-ті пайдалану арқылы оның барлық артықшылықтарын жоққа шығарады. Node.js жылдам ауқымды желілік қосымшаларды құру үшін тамаша қосымша болып табылады, себебі ол бір мезгілде жоғары өткізу қабілеттілігі бар үлкен қосылыстардың көп мөлшерін өңдеуге мүмкіндік береді.

Әрбір қосылу (сұрау) жана ағын тудыратын дәстүрлі веб-қызметтермен салыстырғанда, жұмыс жадысын жылдамырақ жүктейді. Node.js неғұрлым үнемді:

- бір ағын бойынша жұмыс істейді;
- ішкі кіріс қоңыраулар қоңыраулар үшін белсенді емес;
- мыңдаған бәсекелестік байланыстарды қолдауға мүмкіндік береді.



2.1-Node.js жұмыс барысы

Қарапайым есептеу: мысалы, әр ағын 2 Мбайт жадты талап ете алады және

8 Гб оперативті жады бар жүйемен жұмыс істей алады. Бұл жағдайда біздің теория 4000 бірдей қосылымдар және ағындар арасындағы контексті жылжытудың мәнін күтуге болады. Бұл жағдайда сіз дәстүрлі веб-сервистерді қолданумен айналысуыңыз керек. Мұның бәрін болдырмау арқылы Node.js миллионнан астам бәсекелес байланыста өлшенеді (тұжырымдаманы растау үшін эксперимент ретінде). Олардың ең көп тарағандары шексіз модуль немесе ескірген жүйелік құралдар мен мониторлармен жұмыс жасауы.

Node.js пакет менеджері

Node.js файлын талқылау үшін кез келген Node.js.-та орнатылған NPM құралы пайдаланылады, мұнда пайдаланылған пакетті басқарудың бекітілген қолдауын көрсету керек. NPM модульдерінің идеясы Ruby Gems-ге ұқсас: ол онлайн-репозиторий арқылы оңай орнатылатын қол жетімді компоненттер жиынтығы; қолдау көрсетілетін нұсқалар мен тәуелділіктерді қамтиды. Қапталған модульдердің толық тізімі NPM веб-сайтында npmjs.org қол жетімді және Node.js.-да автоматты түрде орнатылатын CLI NPM құралында қол жетімді. Модульдердің экожүйесі толығымен ашық, бұл жерде әрбір модуль NPM репозиторийі тізімінде пайда болатын өз модулін жариялай алады. Ең танымал заманауи NPM модульдерінің кейбірі:

express - хабарлау –Sinatra- да жазылған Node.js үшін веб-әзірлеудің негізі, іс жүзінде қолданыстағы Node.js қосымшаларының көпшілігі үшін норма;

connect– Node.js-пен бірге «аралық» деп аталатын жоғары сапалы «қосылатын модульдер» жиынтығын ұсынатын HTTP сервері ретінде жұмыс істейтін кеңейтілетін орта; экспресс үшін негіз болып табылады;

jade – HAML арқылы жазылған танымал үлгілік шаблондардың бірі, әдетте Express.js-те пайдаланылады;

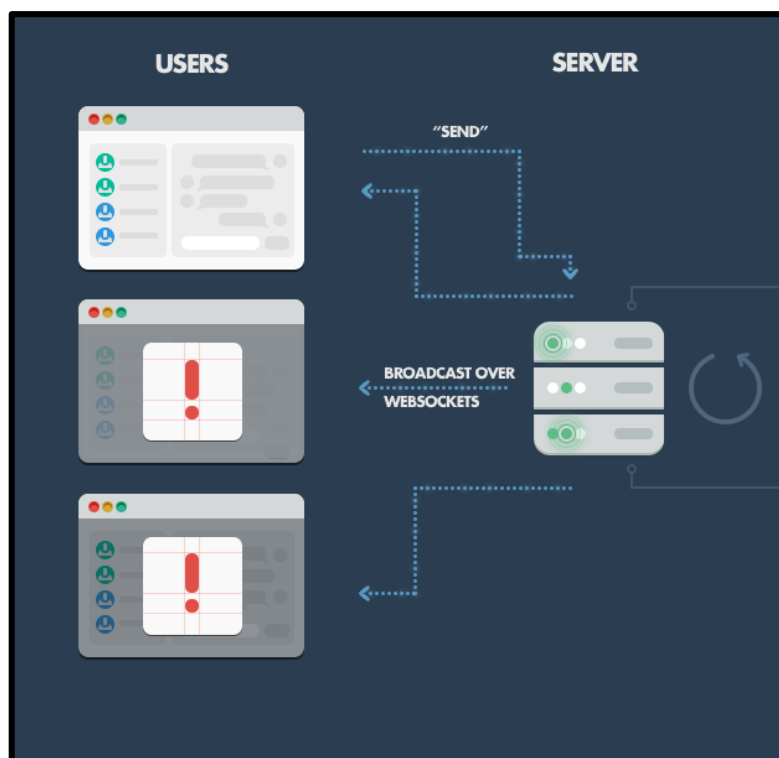
mongo және mongojs – MongoDB орауыштары Node.js ішіндегі MongoDB нысан дерекқорлары үшін API қамтамасыз етеді;

redis - Клиенттің кітапханасын өңдеу;

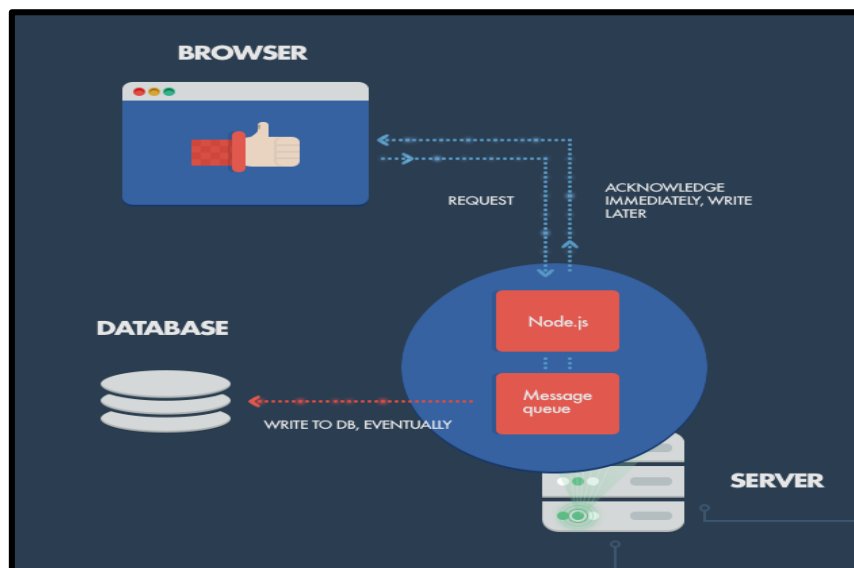
coffee-script– CoffeeScript компиляторы, әзірлеушілерге Node.js арқылы бағдарламаларды жазуға мүмкіндік береді;

underscore– Node.js-те пайдалану үшін бумаға салынған ең танымал JavaScript-көмекші кітапханалары, сондай-ақ жақсартылған өнімділікті қамтамасыз ететін екі ұқсас кітапхана;

Веб-сокет қосылғандағы сервер бөлігі хабарды қабылдайды және оны барлық басқа қосылған клиенттерге таратады. 2.2-сурет веб-сокет қосылымы серверінің бір бөлігін көрсетеді. Барлық тұтынушылар веб-бетте жұмыс істейтін веб-компонентті пайдаланып, жаңа басу хабарламасын алады. Содан кейін олар хабардың мазмұнын шығарып, форумға жаңа жазба енгізетін веб-бетті жаңартады.



2.2 сурет - Серверлік компонент



2.3 сурет –Клиент сервер

Егер бағдарлама процессорды жүктейтін қарқынды есептеулерді орындамаса, MongoDB және JSON нысандарының дерекқорын пайдалансаңыз, оны JavaScript-та, оның ішінде дерекқорға жаза аласыз. Бұл дамуды ғана емес, мамандарды іріктеуді жеңілдетеді. Процессорды жүктейтін кез-келген қарқынды есептеу Node.js динамикасына кедергі келтіреді, сондықтан бұл жағдайда көп ағынды платформаны пайдалану жақсы. Есептеулерді көлденең өлшеуге де болады. Осындай қарқынды есептеуіш процессорды ауыстыра отырып, клиенттік сұрауларды синхронды түрде өңдейтін етіп, дерекқор интерфейсінде өңделетін ауқымды MQ ортасын жасай аласыз.

MongoDB



2.4 сурет–MongoDB

MongoDB - құжатқа бағдарланған ашық бастапқы дерекқор. Деректер базасына арналған деректерді басқару жүйесін таңдау, реляциялық әріптестерден айырмашылығы, құжатта өрістер жиынтығында шектеулер жоқ екеніне негізделеді. Аймақтық және синхронды өңделген клиенттік сұраулар. 2.4 суретте MongoDB логотипі көрсетілген. Құжаттарға бағдарланған дерекқорлар реляциялық ретінде әлі де таралмағанымен, жыл сайын олардың танымалдығы артып келеді, құжатқа бағытталған деректер базасының терминологиясының кейбір нюанстарын түсіндіру ұсынылады. MongoDB негізгі түсініктері:

MongoDB - бұрынғы деректер базасына тұжырымдамалық түрде ұқсас (немесе Oracle, схема бойынша). MongoDB ішінде нөлдік немесе одан да көп дерекқорлар болуы мүмкін, олардың әрқайсысы басқа нысандар үшін контейнер болып табылады.

Дерекқорда нөл немесе одан көп «жинақтар» болуы мүмкін. Бұл коллекцияға қатысты дерекқордан алынған кестеге ұқсас. Коллекцияларда көптеген «құжаттар» бар. Құжатта бір немесе бірнеше өрістер, бағандар бар.

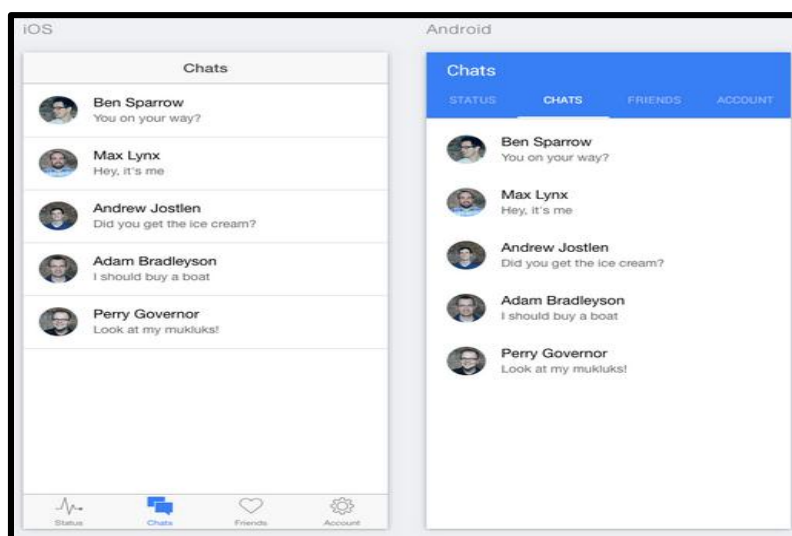
Ionic framework - ең талқыланатын тақырыптардың бірі. Ionic - AngularJS, SASS, Apache Cordova негізіндегі гибриді мобильді қосымшаларды, CSS және JS компоненттерінің жиынтығын жасау үшін құрылған. 2.5-суретте ionic логотипі көрсетілген.



2.5 сурет-Ionic

Ionic репозиторийінің осы уақытқа дейін 15 300 жұлдызы бар, ал оны продукциялайтын Drifty компаниясына инвестицияланған сома 3,7 миллион долларға жетті.

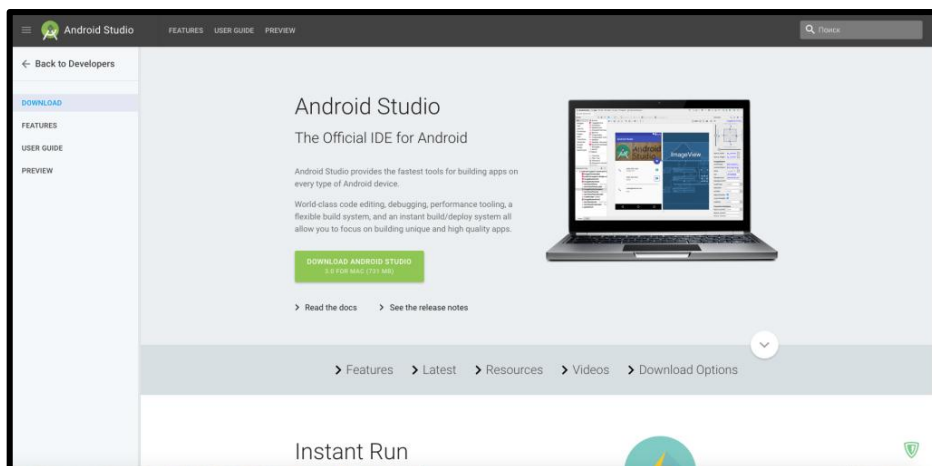
ionic serve - lab iOS және Android платформалары үшін бір уақытта сіздің браузеріңізді көрсетеді. Бұл опция сонымен қатар тікелей қайта жүктеуді қолдайды. (2.6-суретке сәйкес).



2.6сурет – iOS және Android платформасы
Android Studio-ны орнату

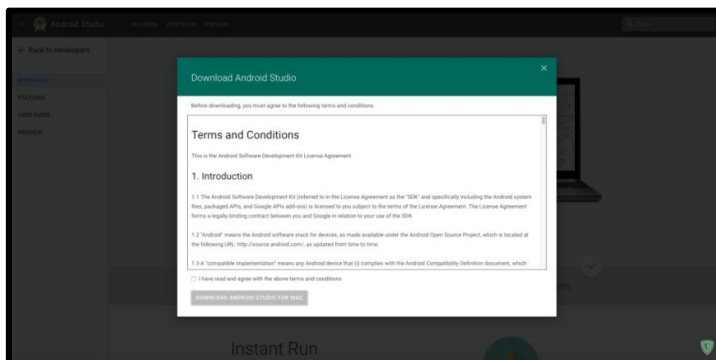
Android Studio бетіне өтіп, платформаңыздың ең соңғы нұсқасын жүктеу үшін түймені басамыз.

2.15 суретте Android Studio басты беті көрсетілген.



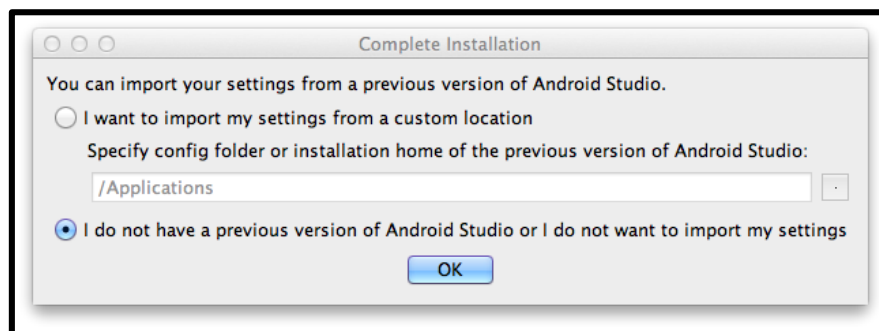
2.15 сурет –Android Studio-ның басты беті

Жүктеу түймесін басып, бағдарламалық жасақтаманың пайдалану шарттарын растаймыз. Мұқият оқудан және қабылдаудан кейін жүктеу басталады. Осыдан кейін Android Studio бағдарламасын кез келген басқа бағдарлама сияқты орнатуға болады. Бастапқы бет Mac және Windows үшін орнату нұсқауларын қамтиды. 2.16, 2.17 суретте. Android Studio бағдарламасының орнатылуы көрсетілген.



2.16 сурет–Android Studio орнатылуы

Android Studio бағдарламасын орнатқаннан кейін бағдарлама басталады. Бағдарлама параметрлерді импорттағыңыз келе ме деп сұрайды. Сіз басынан бастап, екінші элементті таңдап, жалғастырыңыз.



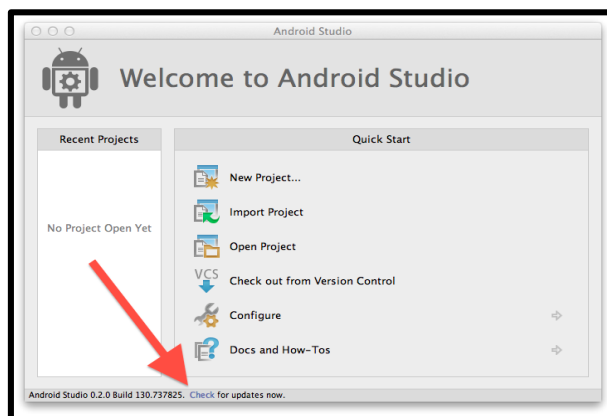
2.17 сурет - Android Studio параметрлерін импорттау

Жүктеу экранын Material Design стилінде көресіз (2.18-сурет бойынша)



Сурет 2.18 - Android Studio

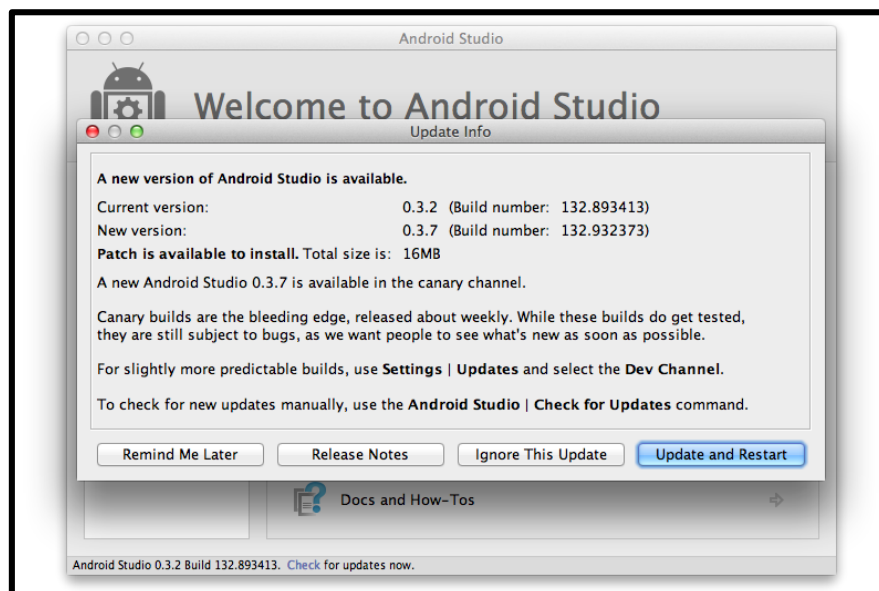
Осыдан кейін келесі бөлімін орындаймыз (2.19-суретке сәйкес)



2.19-сурет - Жаңа жобаны құру

Android студиясын ғана жүктеген болсаңыз да, сізде соңғы нұсқасы болмауы мүмкін. Нұсқалармен кейінірек проблемаларды болдырмау үшін, Check батырмасын басыңыз.

Қазір жаңартуларды тексеріңіз және қажет болса, соңғы нұсқаны алу үшін барлық нұсқауларды орындаңыз. Кейде студиялар сізге 2.20 суретке ұқсас экранды жаңарту туралы хабар береді.



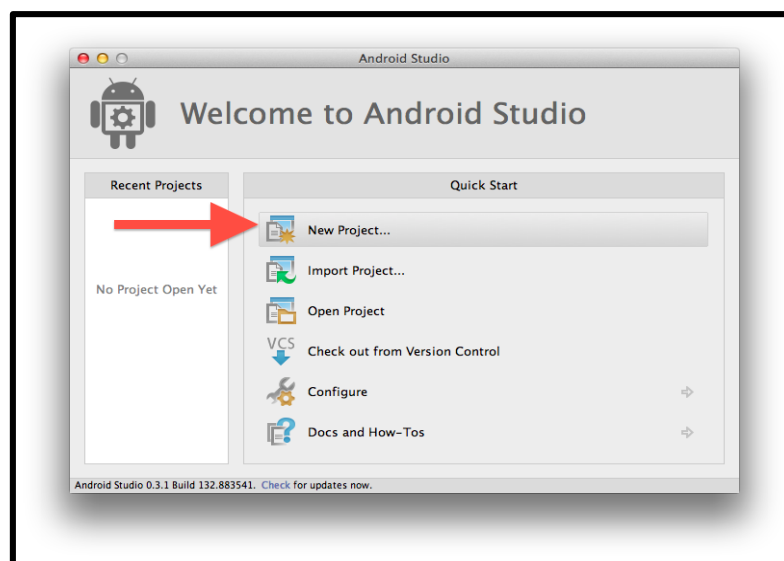
2.20 сурет - Бағдарламаны жаңарту

Бұл жағдайда Update and Restart пәрменін таңдаңыз.

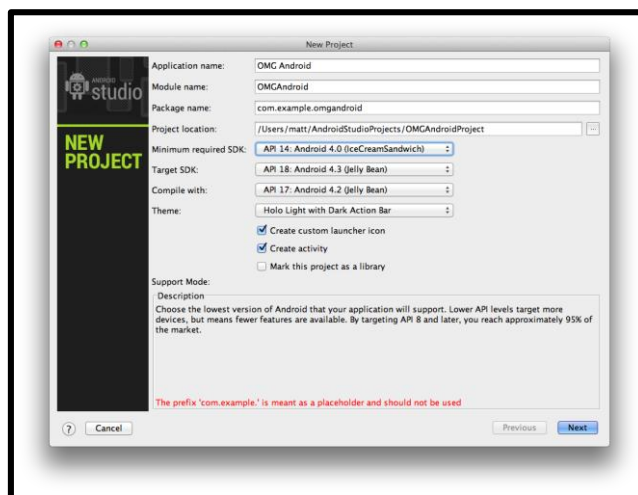
Бірінші Android жобасын жасау

Android Studioда жобаңызды жасауға көмектесетін шағын қадам бар.

Бастау экранында «Жаңа жоба» түймесін басыңыз: 2.21-суретте көрсетілгендей



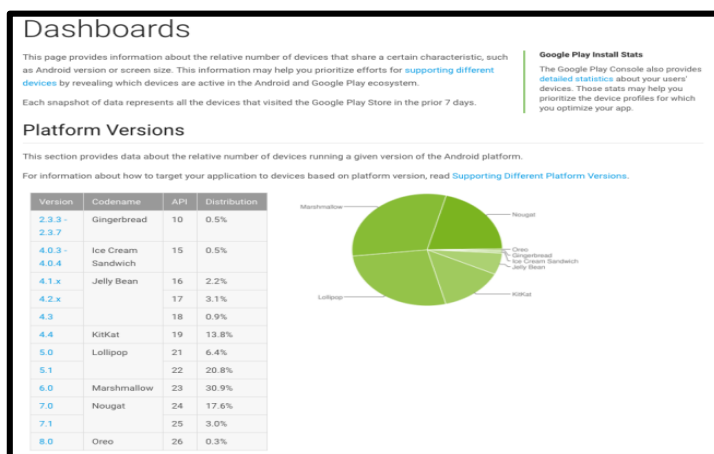
Сурет 2.21 - Жаңа Android Studio жобасы



2.22-сурет - Жаңа жобаны құрудың бірінші кезеңі

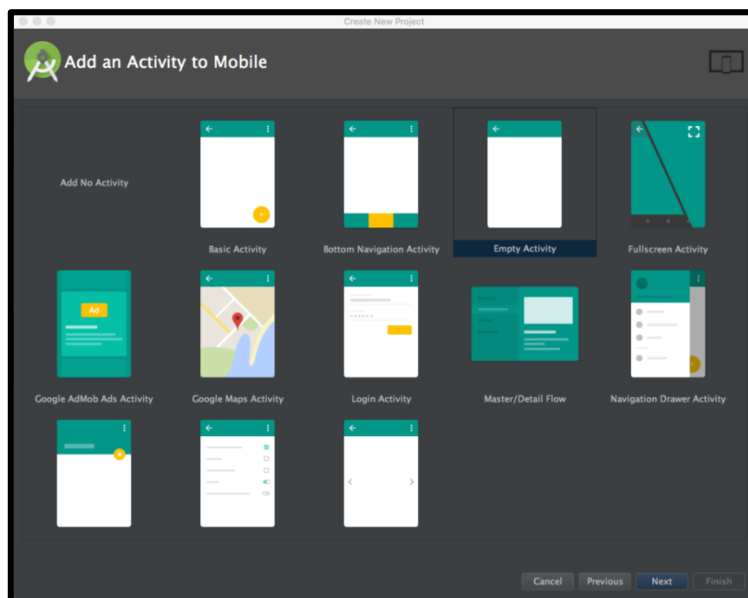
Оны дәл осылай толтырыңыз. Экранның төменгі жағындағы ескертуді алып тастау үшін бума атауындағы «мысал» дегенді ауыстырыңыз. Сондай-ақ, жобаның орнын қатқыл дискідегі кез-келген папканы көрсете отырып орнатуға болады.

Ашық SDK нұсқаларында диалогтық терезенің төменгі жағындағы «Сипаттама» бөліміне назар аударыңыз. Бұл әрбір параметр үшін түсіндіріледі. Скриншотта көрсетілгендей ең төменгі SDK-ні орнатыңыз. Бағдарламаны іске қосу үшін Android нұсқасын орнату қажет. Өзіңіздің жобаларыңыз үшін осы соманы таңдау сіз қалаған SDK мүмкіндігін теңестіру мәселесі және 2.23-суреттегідей қолдау көрсететін құрылғылар.



2.23-сурет. API және оларды әзірлеуші веб-сайтында пайдалану

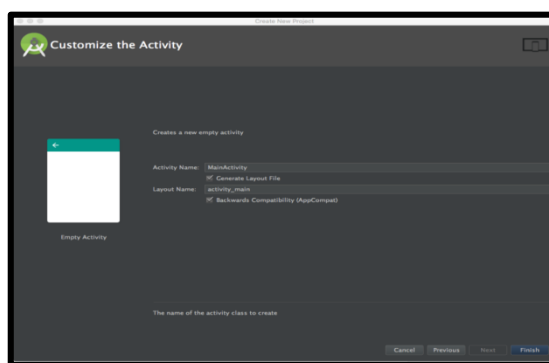
Нұсқаны таңдағаннан кейін іске қосу үлгісін таңдау экраны ашылады. API google карталарымен өзара әрекеттесетін және картаны көрсететін бағдарламаны жасай алады. Сынақ жағдайында «Әрекетті босату» пәрменін таңдап, «Келесі» түймешені басыңыз. (2.24 суретке сәйкес).



2.24 сурет. Main Activity класы

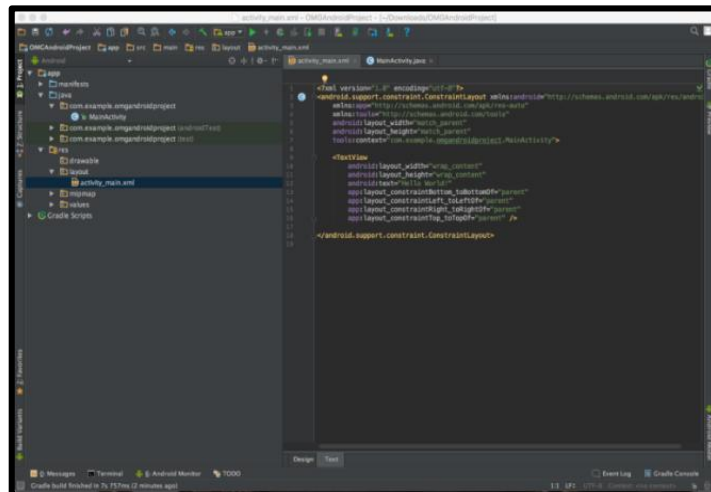
Енді сіз қосымшаны құру процесінің соңғы сатысында тұрсыз. Аяқтауды бастамас бұрын кейбір нәрселерге назар аударыңыз. Мұнда сіз кез-келген өтініштің негізгі дизайн компоненттеріне сілтемелер таба аласыз.

Қызметтің атауы бірінші болып табылады, бірақ соңғы сөз емес, ескерту сөз Activity. Android контексінде әрекетті әдетте қолданбаңыздағы «экран» ретінде қарастырады. Бұл тармақ өте икемді. Android Studio негізгі ағыны MainActivity жасаған кезде, ол Android SDK-ден белсенділік класынан алады. Нысаналы-бағдарланған бағдарламамен танысу, бұл тұжырымдаманы түсінеді, бірақ жаңадан бастағандар үшін MainActivity қолданылады. Бұл қызметтің реттелетін нұсқасы (2.25-суретте)



Сурет 2.25 - Android Studio жобасын жасау және жүктеу

Аяқтау түймесін басыңыз. Жобаны жасау мен жүктеуге біраз уақыт кетеді. Бір сәттен кейін Android Studio жобаңыздың құрылысын аяқтайды. Әрине, жоба әлі де бос, бірақ Android құрылғысында немесе эмуляторда жұмыс істеу үшін бәрі бар. (2.26.2.27 суретке сәйкес)



Сурет 2.27-Android бағдарламалық жасақтамасы бар

Бағдарламаны эмуляторда іске қосу. Енді эмулятор туралы айта кетсек. Android Studio бағдарламасында қолданбаларды іске қосу, вебті шолып шығу, жөндеу және т.б. үшін Android құрылғысын эмуляциялауға болатын бағдарламалық жасақтама бар. Бұл мүмкіндік Android Virtual Device Manager (AVD) арқылы қамтамасыз етіледі. Қаласаңыз, кейбір эмуляторларды баптай аласыз, әр жаңа эмулятор үшін экран өлшемін және платформа нұсқасын орнатуға болады. Бұл функция өте пайдалы, өйткені ол әзірлеушілерді сынақ бағдарламалары үшін кейбір құрылғыларды сатып алудан босатады. Іске қосу үшін Run нүктесін басу керек.

3 ПРАКТИКАЛЫҚ БӨЛІМ

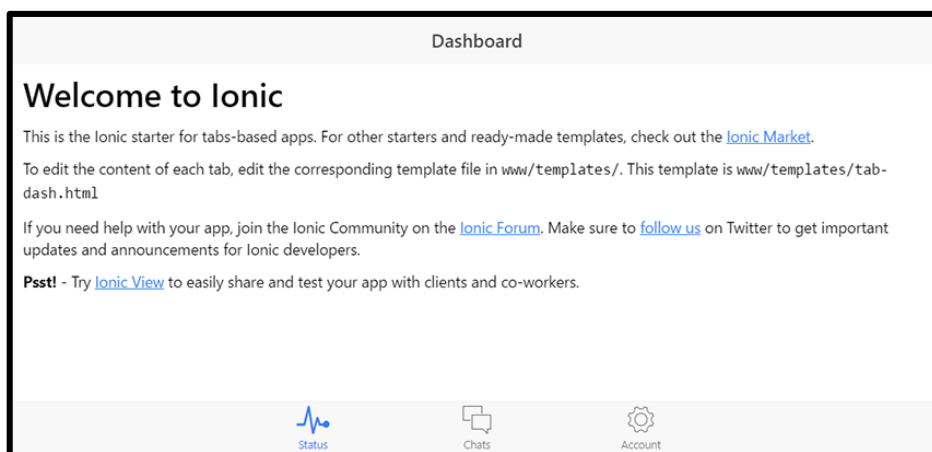
3.1 Қосымшаны құру процесі

Мобильді қосымшаны құру үшін, біз `ionic` негіздемесінің көмегімен гибридті дамытамыз. Жұмысты бастау үшін бізге `javascript`, `html` және қосымша элементтер бойынша тәжірибе керек. Бұған қоса, бұрын `AngularJS`-мен жұмыс істеген болсаңыз пайдасы көп. Сондай-ақ бөлек сайттар немесе қызметтердің кейбір түрін жасасаңыз, бұл да септігін тигізеді. `Ionic`-ке жаңа жоба орнату үшін, біздің жүйеге `Node.js` орнату қажет. Орнатқаннан кейін `npm` пакетінің менеджеріне (пакет менеджері) қосыла аласыз. `Ionic-framework` орнату үшін пәрмен жолына кіру қажет:

Қолданбамыз гибридті бағдарлама болғандықтан, артықшылығы бар, браузердің ішінде мобильді қосымшаны іске қосуға болады. Браузерде мобильді қосымшаны іске қосу үшін келесі пәрменді іске қосу керек:

`ionic serve`

3.1-суретке сәйкес жүктелген бағдарламамен браузер ашылады.



3.1 сурет – Қосымшаның мысалы

3.2 Қолданбаның жұмысымен танысу

Android негізіндегі «Ломбард +» мобильдік қосымшасы клиенттерге ыңғайлы болу үшін және ломбардтық қызметке тарту үшін арналған. 3.11-суретте мобильді қосымшаның негізгі экранда орналасуы көрсетілген. Бұл бет авторизацияға арналған. Ол үшін пайдаланушы жеке логин мен парольді енгізу керек.



3.11 сурет – «Ломбард» мобильді қосымшасының басты бетінің макеті

3.12-суретте «Карталар» және «Ломбардтар атауы» бөлімдерінің орналасуы көрсетіледі. Карта учаскесінде барлық ломбард филиалдарының орналасқан жері көрсетілген. Ломбардты таңдағаннан кейін, ең жақын орналасқан ломбард филиалы туралы барлық ақпарат ашылады.

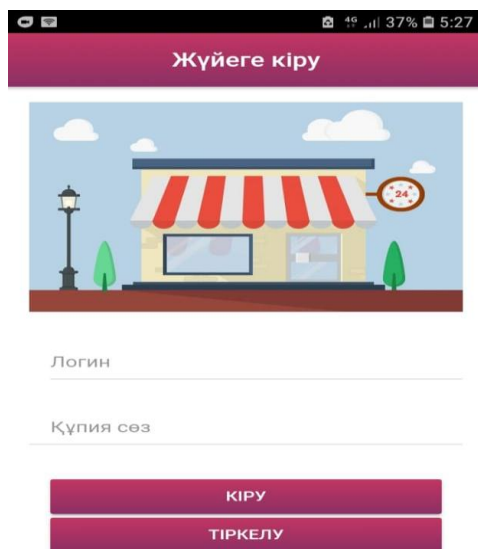


3.12 сурет – «Карталар» және «Ломбардтың атауы» секцияларының орналасуы

Әрі қарай 3.13 суретте «Өтініш беру» және «Қолдану» бөлімдерінің орналасуы көрсетіледі.

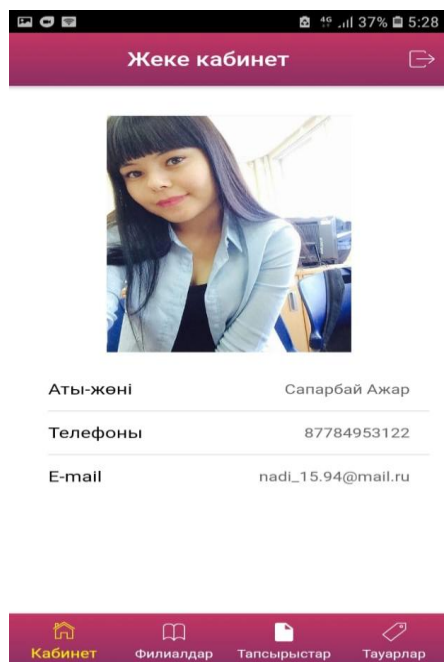


3.13 сурет – «Өтініш жіберу» және «Қолданбалар» бөлімдерінің орналасуы.

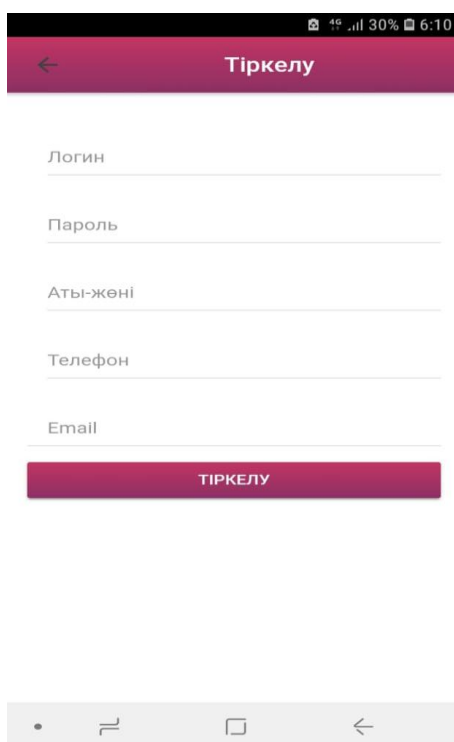


3.15 сурет – Кіру беті

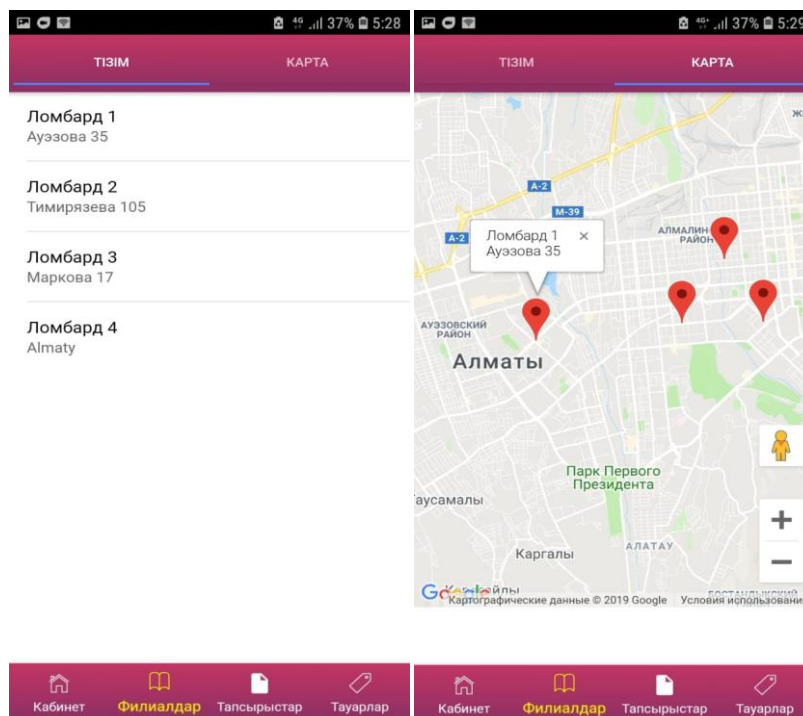
Кіру бетінде деректерді енгізу үшін өрістерді көруге болады. Атап айтқанда, логин мен пароль. Сондай-ақ «енгізу» батырмасы. Дұрыс тіркеуден өтіп, «енгізу» түймесін басу керек. Қолданба біздің жеке шотымызда іске қосылады. Есептік жазбаңызда толық аты-жөні, телефон нөмірі, сондай-ақ электронды поштаны көруге болады.



3.16 сурет – Жеке кабинет беті

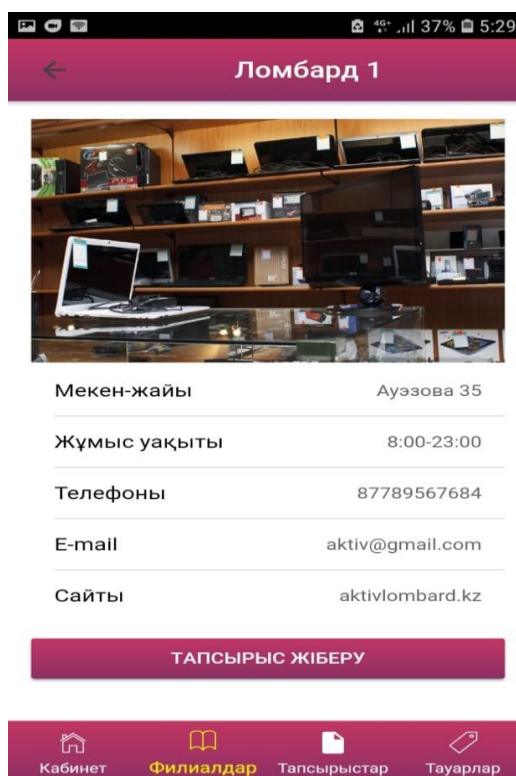


3.17 сурет – Тіркелу беті



3.18 сурет – Ломбардтар беті

Ломбардтың бетінде тізім мен картаны ауыстыру үшін табтарды көруге болады. Бұл пайдаланушыға ыңғайлы болу үшін жасалған. Кейбір ломбардтар алыс болуы мүмкін.



3.19 сурет – Ломбард беті

Ломбардтың беті пайдаланушыға қажетті барлық ақпаратты көрсетеді. Сондай-ақ, осы бетте кепілзатты бағалау туралы сұрау жібере аласыз. «Өтініш жіберуді» басқан кезде. Одан басқа да терезені көруге болады.

Одан мыналарды көрсетуі керек:

1. өнімнің атауы;
2. ұсынылатын баға;
3. санат;
4. қосымша ақпарат.
5. жіберу туралы сұрау батырмасын басыңыз.

Санатты таңдаған кезде сіз нақты тауарлар санатын көрсетуіңіз керек. Админ бұл барлық қосымшаларды қадағалайды. Өтінішті жібергеннен кейін оны жоюға болады. Бірақ егер админ өтінішті қабылдаса, қолданба бас тарта алмайды.

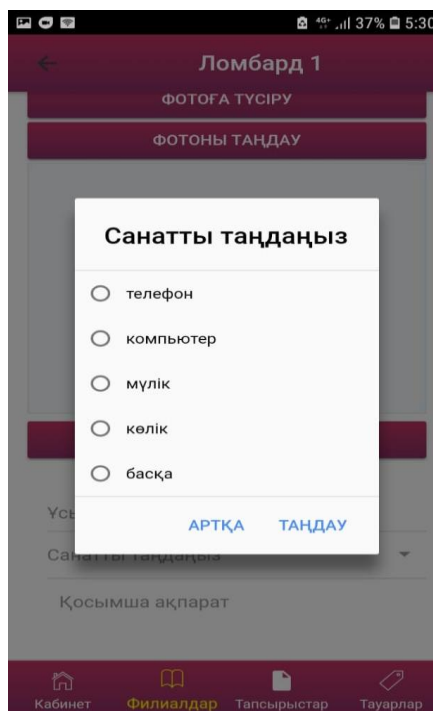
Барлық өтініштер қосымшалар бетінде пайда болады. Онда олар табтармен бөлінеді.

Табтар:

6. күтілетін;
7. қабылданғандар;
8. қабылданбағандар.



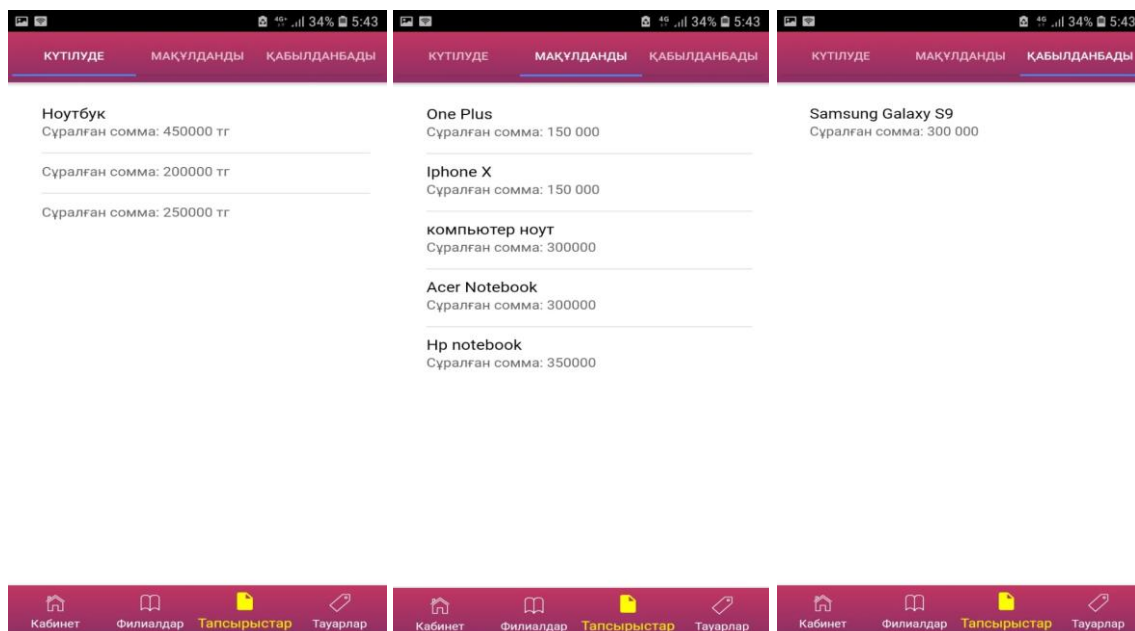
3.20 сурет – Өтініш қалдыру беті



3.21 сурет – Тауарлар санатының терезесі



3.22 сурет – Өтініш беті

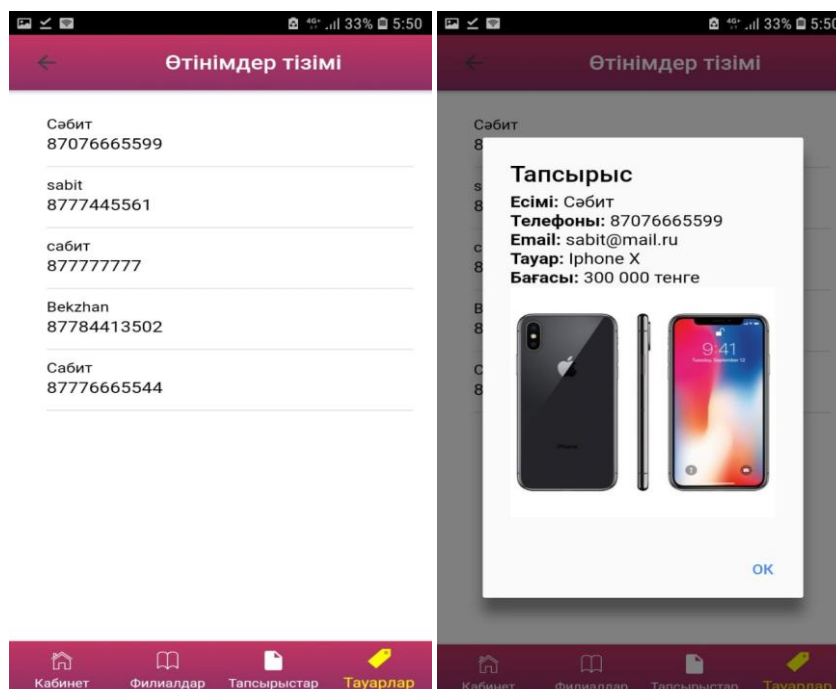


3.23 сурет – Барлық өтініштер беті

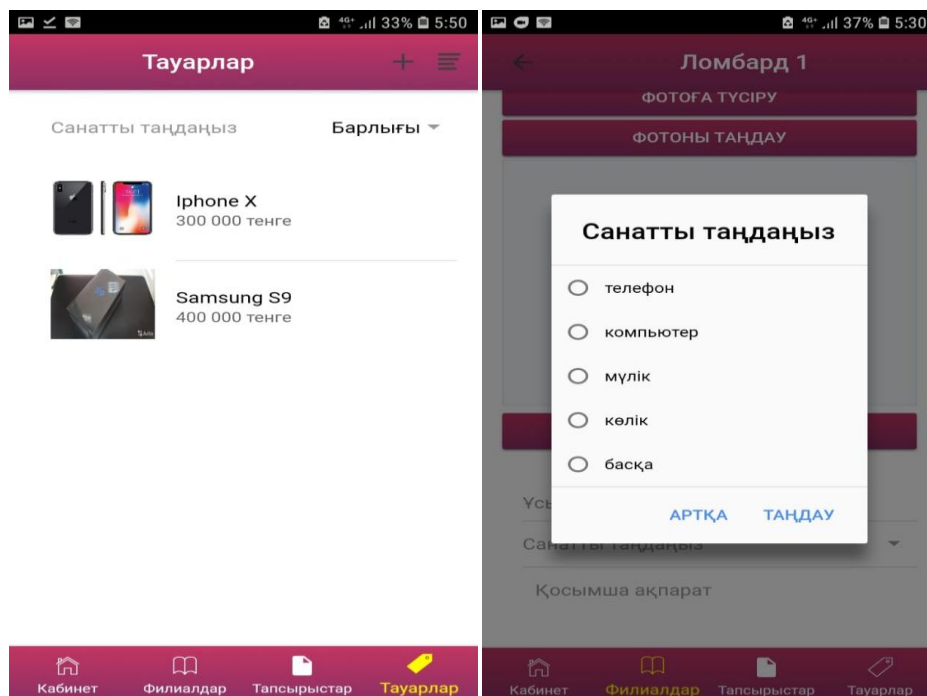
Барлық өтініштер бетінде сіз өзіңіз жіберген барлық өтініштерді көре аласыз.

Ломбардтың беті сіздің салымыңыз үшін ұсынылатын бағаны көрсетеді.

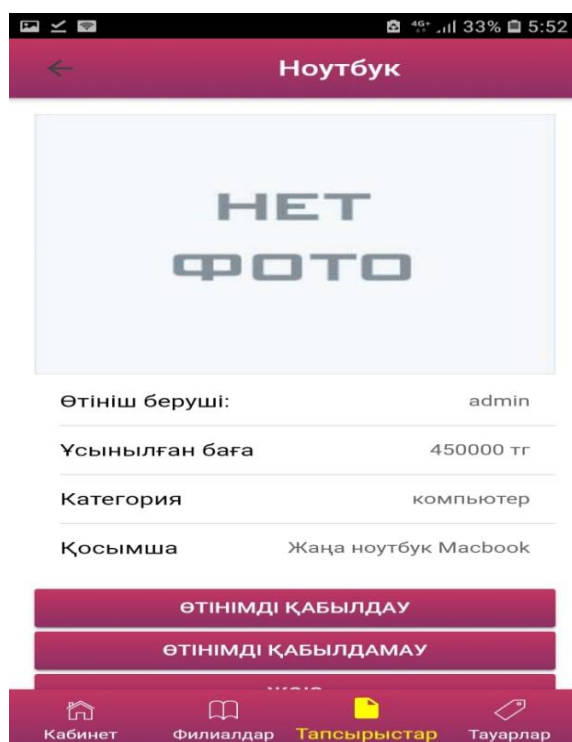
Ломбардтар өнімдері бетінде сіз желідегі өнімнің көрмесін көре аласыз. Бұл бөлімде қажетті өнімдерді арзан бағамен таба аласыз.



3.24 сурет – Барлық өтініштер беті - қабылданбағандар



3.26 сурет – Ломбард тауарлары беті



3.27 сурет – Тауар беті

ҚОРЫТЫНДЫ

Көптеген жылдар бойы жастарымыздың көптеген азаматтары кәсіпкерлік қызметтің әртүрлі түрлеріне қатысады. Кәсіпкерлік қызметті бағалауда ең тиімді кәсіпорындардың бірі - пайыздық несие беру болып табылады. Осыған байланысты көптеген кәсіпкерлер ломбардты табыс көзі ретінде қарастырады.

Мысалы, ломбард - мамандандырылған коммерциялық ұйым ретінде негізгі қызметі - азаматтардың жылжымалы мүлкімен және заттарды кепілге қою арқылы алынған қысқа мерзімді несиелер беру. Диссертациялық жұмысты орындау барысында келесі программалау тілдері мен технологиялары пайдаланылды: Ionicframework, typescript, javascript and mongoDB дерекқоры.

Диссертациялық жұмыстың барлық талаптары орындалды:

- даму (разраб.) ортасы дайындалды;
- деректерді сақтауға арналған дерекқорды құрды;
- ломбард+ компаниясына арналған мобильдік қосымшаны құрды;
- тұтынушылар үшін ақпараттық ортаны қалыптастырады;
- ломбард үшін «алдын-ала бағалау» мүмкіндігі жасалды;
- сатып алушыларға, бағамен, өнімнің фотосуреттерімен онлайн-дүкен

құрылды;

Ionicframework және androidstudio негізіндегі түпнұсқа мобильді қосымшаны әзірледі.

ӘДЕБИЕТТЕР ТІЗІМІ

- 1 Иванова Г.С. Основы программирования: Учебник для вузов. - М.: Издательство МГТУ им. Н.Э. Баумана, 2012. - 416 с.
- 2 Кормен Т., Ривест Р. Алгоритмы: построение и анализ. - М.: МЦНМО, 2009. - 960с.
- 3 Левитин В. Алгоритмы: введение в разработку.: Пер. с англ. - М. : Вильямс, 2016. 523 с.
- 4 Окулов С.М. Основы программирования. - 3-е изд. - М.: БИНОМ. Лаборатория знаний, 2016. – 440с.
- 5 Орлов С. Технологии разработки программного обеспечения. - СПб.:Питер, 2005. 325 с.
- 6 Сырецкий Г. Информатика. Фундаментальный курс. Том 1. Основы информационной и вычислительной техники.: БХВ-Петербург, 2015 - 832 с.
- 7 Михеева Е.В. Информатика. 7-е изд., испр., 2012 - 400с.
- 8 Информатика. Базовый курс. Учебник для Вузов/под общей редакцией. С.В. Симоновича, - СПб.: Питер, 2012. – 245 с.
- 9 Симонович С. В., Евсев Г.А., Прак. информатика, Учебное пособие. М.: АСТпресс, 2013. – 362 с.
- 10 Денисов А., Белов А.. Самоучитель Интернет. - СПб: Питер, 2011. - 461 с.
- 11 ShklarL., RosenR. WebApplicationArchitecture: Principles, Protocols and Practices. Wiley, 2009 – 440 с.
- 12 Касторнова, В.А. Структуры данных и алгоритмы их обработки на языке программирования: Учебное пособие / В.А. Касторнова. — СПб.: ВHV, 2016. — 304 с.
- 13 Кауфман, В.Ш. Языки программирования. Концепции и принципы / В.Ш. Кауфман. — М.: ДМК, 2011. — 464 с.
- 14 Дэвид Флэнаган, JavaScript. Подробное руководство, 6-е издание. - Символ-Плюс, 2016. – 1080с.
- 15 Эрл Каслдайн, Изучаем JQuery, 2-е издание. - М.:Питер, 2015. – 402с.

А ҚОСЫМША

```
import { NgModule, ErrorHandler } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { IonicApp, IonicModule, IonicErrorHandler } from 'ionic-angular';
import { MyApp } from './app.component';

import { TabsPage } from '../pages/tabs/tabs';
import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';
import { AgmCoreModule } from '@agm/core';
import { Camera } from '@ionic-native/camera';
import { Geolocation } from '@ionic-native/geolocation';

import { LombardPage } from '../pages/lombard/lombard';
import { RequestPage } from '../pages/request/request';
import { AuthorPage } from '../pages/author/author';
import { ProductPage } from '../pages/product/product';

import { LeaverequestPage } from '../pages/leaverequest/leaverequest';

import { LombardDetailPage } from '../pages/lombard-detail/lombard-detail';
import { ProductDetailPage } from '../pages/product-detail/product-detail';
import { RequestDetailPage } from '../pages/request-detail/request-detail';

import { HttpClientModule } from '@angular/common/http';
import { HttpModule } from '@angular/http';
import { LombardService } from '../pages/shared/services/lombard.service';
import { RequestService } from '../pages/shared/services/request.service';
import { LoginPage } from '../pages/login/login';
import { LoginService } from '../pages/shared/services/login.service';
import { CabinetPage } from '../pages/cabinet/cabinet';
import { ProductService } from '../pages/shared/services/product.service';
import { RegPage } from '../pages/reg/reg';
import { SendRequestToProductPage } from
"../pages/send-request-to-product/send-request-to-product";
import { AddProductPage } from "../pages/add-product/add-product";
import { OrderService } from "../pages/shared/services/order.service";
```

А қосымшаның жалғасы

```
import {OrdersListPage} from "../pages/orders-list/orders-list";
```

```
@NgModule({
  declarations: [
    MyApp,
    TabsPage,
    CabinetPage,
    LombardPage,
    RequestPage,
    ProductPage,
    AuthorPage,
    LombardDetailPage,
    LeaverequestPage,
    RequestDetailPage,
    ProductDetailPage,
    LoginPage,
    RegPage,
    SendRequestToProductPage,
    AddProductPage,
    OrdersListPage
  ],

  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp),
    HttpClientModule,
    HttpModule,
    AgmCoreModule.forRoot({
      apiKey: 'AIzaSyD_25gMznhrYHcnh45LY6QujM8TJy2Xo-Q'
    })
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    TabsPage,
    CabinetPage,
```

А қосымшаның жалғасы

```
LombardPage,  
  RequestPage,  
  ProductPage,  
  AuthorPage,  
  LombardDetailPage,  
  LeaverequestPage,  
  RequestDetailPage,  
  LoginPage,  
  ProductDetailPage,  
  RegPage,  
  SendRequestToProductPage,  
  AddProductPage,  
  OrdersListPage  
],  
providers: [  
  StatusBar,  
  SplashScreen,  
  {provide: ErrorHandler, useClass: IonicErrorHandler},  
  LombardService,  
  RequestService,  
  LoginService,  
  ProductService,  
  Camera,  
  Geolocation,  
  OrderService  
]  
})  
export class AppModule {}
```

<ion-header>

<ion-navbar>

<ion-title>Жүйеге кіру</ion-title>

</ion-navbar>

А қосымшаның жалғасы

```
</ion-header>
```

```
<ion-content padding>
```

```
  <div style="text-align: center;">
```

```
    
```

```
  </div>
```

```
  <ion-list>
```

```
    <ion-item>
```

```
      <ion-label floating>Логин</ion-label>
```

```
      <ion-input type="text" [(ngModel)]="username"></ion-input>
```

```
    </ion-item>
```

```
    <ion-item>
```

```
      <ion-label floating>Құпия сөз</ion-label>
```

```
      <ion-input type="password" [(ngModel)]="password"></ion-input>
```

```
    </ion-item>
```

```
</ion-list>
```

```
  <div padding>
```

```
    <button ion-button block (click)="signIn()">Kipy</button>
```

```
    <button ion-button block (click)="navToRegPage()">Тіркелу</button>
```

```
  </div>
```

```
</ion-content>
```

```
import { Component } from '@angular/core';
```

```
import { NavController, NavParams } from 'ionic-angular';
```

```
import { LoginService } from '../shared/services/login.service';
```

```
import { TabsPage } from '../tabs/tabs';
```

```
import { AlertController } from 'ionic-angular';
```

```
import { RegPage } from '../reg/reg';
```

```
@Component({
```

```
  selector: 'page-login',
```

```
  templateUrl: 'login.html',
```

А қосымшаның жалғасы

```
    })
    export class LoginPage {

        username: string = "";
        password: string = "";

        constructor(public navCtrl: NavController, public navParams: NavParams,
private loginService: LoginService,
        private alertCtrl: AlertController) {
        }
        presentAlert() {
            let alert = this.alertCtrl.create({
                title: 'Қайталап көріңіз!',
                subTitle: 'Логин немесе құпия сөз дұрыс емес.',
                buttons: ['Ок']
            });
            alert.present();
        }

        signIn() {
            this.loginService.loginByUsernameAndPassword({"where":{"and":[{"login":
this.username}, {"password": this.password}]}})
                .subscribe(data => {
                    if (data[0] && data[0].login == this.username && data[0].password ==
this.password) {
                        localStorage.setItem("login", JSON.stringify(data[0]));
                        this.navCtrl.setRoot(TabsPage);
                    } else {
                        this.presentAlert();
                        console.log("ERROR");
                    }
                });
        }

        navToRegPage() {
            this.navCtrl.push(RegPage);
        }
    }
}
```

А қосымшаның жалғасы

```
}
```

```
}
```

```
<ion-header>
```

```
<ion-toolbar>
```

```
<ion-segment [(ngModel)]="type">
```

```
<ion-segment-button value="list">
```

```
<span class="lombard-list">Тізім</span>
```

```
</ion-segment-button>
```

```
<ion-segment-button value="map">
```

```
<span class="lombard-list">Карты</span>
```

```
</ion-segment-button>
```

```
</ion-segment>
```

```
</ion-toolbar>
```

```
</ion-header>
```

```
<ion-content >
```

```
<div [ngSwitch]="type">
```

```
<ion-list *ngSwitchCase="'list'">
```

```
<button ion-item *ngFor="let item of items" (click)="itemSelected(item)">
```

```
<h2>{{ item.title }}</h2>
```

```
<p>{{ item.adress }}</p>
```

```
</button>
```

```
</ion-list>
```

```
<agm-map [latitude]="lat" [longitude]="lng" *ngSwitchCase="'map'"  
[zoom]="13">
```

```
<agm-marker [latitude]="myLat" [longitude]="myLon"  
[label]="labelOptions"></agm-marker>
```

```
<agm-marker *ngFor="let item of items" [latitude]="item.lat"  
[longitude]="item.lng">
```

```
<agm-info-window>
```

```
<div (click)="itemSelected(item)">
```

А қосымшаның жалғасы

```
{{item.title}} <br>
    {{item.adress}}
</div>
</agm-info-window>
</agm-marker>
</agm-map>

</div>

</ion-content>

import { Component } from '@angular/core';
import { NavController, NavParams } from 'ionic-angular';
import { LombardService } from '../shared/services/lombard.service';
import { Lombard } from '../shared/models/lombard.model';
import { LombardDetailPage } from '../lombard-detail/lombard-detail';
import { Geolocation } from '@ionic-native/geolocation';

@Component({
  selector: 'page-lombard',
  templateUrl: 'lombard.html',
})
export class LombardPage {

  items: Lombard;
  lat: number = 43.240472;
  lng: number = 76.923791;
  type = 'list';
  myLat: number;
  myLon: number;
  labelOptions = {
    color: '#cc001e',
    fontFamily: '',
    fontSize: '14px',
    fontWeight: 'bold',
    text: 'Мен осы жердемін!',
  }
```

А қосымшаның жалғасы

```
};

constructor(
  public navCtrl: NavController,
  public navParams: NavParams,
  private lombardService: LombardService,
  private geolocation: Geolocation
) {
  this.lombardService.getLombards().subscribe(data => this.items = data);
  this.geolocation.getCurrentPosition().then((resp) => {
    // resp.coords.latitude
    // resp.coords.longitude
    console.log(resp.coords);
    this.myLat = resp.coords.latitude;
    this.myLon = resp.coords.longitude;
  }).catch((error) => {
    console.log('Error getting location', error);
  });
}

itemSelected(item: Lombard) {
  this.lombardService.selected = item;
  this.navCtrl.push(LombardDetailPage);
}

}

<!--
Generated template for the LombardDetailPage page.

See http://ionicframework.com/docs/components/#navigation for more info on
Ionic pages and navigation.
-->
<ion-header>

  <ion-navbar>
```

А қосымшаның жалғасы

```
<ion-title>{{item.title}}</ion-title>
</ion-navbar>
</ion-header>
```

```
<ion-content padding>
  
```

```
<ion-list>
  <ion-item>
    Мекен-жайы
    <p item-end>{{item.adress}}</p>
  </ion-item>
```

```
<ion-item>
  Жұмыс уақыты
  <p item-end>{{item.worktime}}</p>
</ion-item>
```

```
<ion-item>
  Телефоны
  <p item-end>{{item.phone}}</p>
</ion-item>
```

```
<ion-item>
  E-mail
  <p item-end>{{item.mail}}</p>
</ion-item>
```

```
<ion-item>
  Сайты
  <p item-end>{{item.website}}</p>
</ion-item>
```

```
</ion-list>
<button ion-button block (click)="leaveRequest(item.title)"> Тапсырыс
жіберу</button>
</ion-content>
```

А қосымшаның жалғасы

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import { LombardService } from '../shared/services/lombard.service';
import { Lombard } from '../shared/models/lombard.model';
import { LeaverequestPage } from '../leaverequest/leaverequest';
```

```
@Component({
  selector: 'page-lombard-detail',
  templateUrl: 'lombard-detail.html',
})
```

```
export class LombardDetailPage {
```

```
  item: Lombard;
```

```
  constructor(public navCtrl: NavController, public navParams: NavParams,
    private lombardService: LombardService) {
    this.item = this.lombardService.selected;
  }
```

```
  leaveRequest(title){
    this.navCtrl.push(LeaverequestPage, { title: title });
  }
}
```

```
<ion-header>
```

```
<ion-toolbar>
```

```
<ion-title>Таярлар</ion-title>
```

```
<ion-buttons end *ngIf="user.role == 'admin'">
```

```
<button ion-button icon-only (click)="navToAddProduct()">
```

```
<ion-icon name="add"></ion-icon>
```

```
</button>
```

```
<button ion-button icon-only (click)="navToListOrder()">
```

```
<ion-icon name="list"></ion-icon>
```

```
</button>
```

```
</ion-buttons>
```

А қосымшаның жалғасы

```
</ion-toolbar>
</ion-header>

<ion-content padding>
  <!-- <ion-list>
    <button ion-item *ngFor="let item of items" (click)="itemSelected(item)">
      
      <h2>{{ item.name }}</h2>
      <p>{{item.price}}</p>
    </button>
  </ion-list> -->

  <ion-list>
    <ion-item>
      <ion-label>Санатты таңдаңыз</ion-label>
      <ion-select [(ngModel)]="category" (ionChange)="onSelectChange($event)">
        <ion-option value="all">Барлығы</ion-option>
        <ion-option value="телефон">Телефондар</ion-option>
        <ion-option value="телевизор">Телевизорлар</ion-option>
        <ion-option value="компьютер">Компьютерлер</ion-option>
      </ion-select>
    </ion-item>
  </ion-list>

  <ion-list>
    <ion-item *ngFor="let item of items" (click)="itemSelected(item)">
      <ion-thumbnail item-start>
        
      </ion-thumbnail>
      <h2>{{item.name}}</h2>
      <p>{{item.price}}</p>
    </ion-item>
  </ion-list>
</ion-content>

import { Component } from '@angular/core';
import { NavController, NavParams } from 'ionic-angular';
```

А ҚОСЫМШАНЫҢ ЖАЛҒАСЫ

```
import { Products } from '../shared/models/product.model';
import { ProductService } from '../shared/services/product.service';
import { ProductDetailPage } from '../product-detail/product-detail';
import { AddProductPage } from '../add-product/add-product';
import { OrdersListPage } from '../orders-list/orders-list';

@Component({
  selector: 'page-product',
  templateUrl: 'product.html',
})
export class ProductPage {

  category = 'all';
  items: Products;
  user = JSON.parse(localStorage.getItem('login'));

  constructor(
    public navCtrl: NavController,
    public navParams: NavParams,
    private productService: ProductService
  ) {
    this.productService.getProducts().subscribe(data => this.items = data);
  }

  ionViewDidEnter() {
    this.productService.getProducts().subscribe(data => this.items = data);
  }

  itemSelected(item: Products) {
    this.productService.selected = item;
    this.navCtrl.push(ProductDetailPage);
  }

  onSelectChange(selectedValue: any) {
    if (selectedValue == 'all')
      this.productService.getProducts().subscribe(data => this.items = data);
  }
}
```

А қосымшаның жалғасы

```
else
    this.productService.getProductsByCategory(selectedValue).subscribe(data =>
this.items = data);
}

navToAddProduct() {
    this.navCtrl.push(AddProductPage);
}

navToListOrder() {
    this.navCtrl.push(OrdersListPage);
}
}
```