ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
ПИ кафедрасының меңгерушісі,
PhD докторы
_____ М. Тұрдалыұлы
“ 06 ” 06 2021 ж.

Дипломдық жобаға
ТҮСІНІКТЕМЕЛІК ЖАЗБА

Тақырыбы: Жеке тұлғалардың қызметтерін ұсынуға арналған веб-платформаны әзірлеу

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

Орындаған Қуаныш А.Ж

Ғылыми жетекші
_____А. Қайрбеков.

«04»____06____ 2021ж.

Алматы 2021

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

БЕКІТЕМІН
ПИ кафедрасының меңгерушісі,
PhD докторы
_____М. Тұрдалыұлы
“ 06 ” 06 _____2021 ж.

ТАПСЫРМА

Білім алушы Қуаныш Ақниет Жаңабекұлы
Тақырыбы: Жеке тұлғалардың қызметтерін ұсынуға арналған веб-платформаны әзірлеу
Университет Ректорының *№2131-б «24» 11 2020 жылғы* бұйрығымен бекітілген.
Аяқталған жұмысты тапсыру мерзімі 2021 жылғы "07" маусым
Дипломдық жұмыстың бастапқы берілістері: *Ұсынылатын дипломдық жобада* жеке тұлғалардың қызметтерін ұсынуға арналған веб-платформаны әзірлеу.
Дипломдық жұмыста қарастырылатын мәселелер тізімі:
а) зерттеу бөлімі
б)технологиялық бөлімі
в)жобалау бөлімі
г) программалау бөлімі
Сызба материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс)
Сызба материалдарының 15 бет слайдта көрсетілген.
Ұсынылатын негізгі әдебиет 8 атаудан тұрады.

Дипломдық жұмысты (жобаны) дайындау

КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекші мен шілерге көрсетумерзімдері	Ескерту
Дипломдық жоба тақырыбына байланысты ізденіс жасау, зерттеу жүргізу.	20.01.2021	
Дипломдық жұмысты жобалау: деректер базасын құру.	01.02.2021	
Дипломдық жобаның веб-косымшасын дайындау, іске асыру.	05.02.2021	
Дипломдық жобаны диплом алды есеп алу комиссия мүшелеріне таныстыру.	10.03.2021	
Дипломдық жобаға түсіндірме жазба жазу.	06.05.2021	

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

Бөлімдер атауы	**Кеңесшілер, аты, әкесінің аты, тегі (ғылыми дәрежесі, атағы)**	**Қол қойылған күні**	**Қолы**
Бағдарламалық бөлім	Рамазан Акерке Техн.ғыл. магистрі, асассистент	31.05.2021ж	
Норма бақылау	Марғұлан Қ. Техн.ғыл. магистрі, лектор	03.06.2021ж	

Ғылыми жетекші _____А. Қайрбеков.

Тапсырманы орындауға алған білім алушы _____А. Қуаныш.

Күні "04" 06 2021ж.

АҢДАТПА

Бұл дипломдық жоба жалға алу немесе жалға беру процестерін автоматтандыруға арналған. Қолданушылар кез-келген затты оңай әрі тез жалға ала алады, сондай-ақ кез-келген өзінің затын жалға қоя алады.

Дипломдық жұмыс кіріспеден, негізгі үш бөлімнен және қорытындыдан тұрады:

Кіріспеде таңдалған тақырыптың мақсаты, оның өзектілігі және тапсырмаларын көрсетіледі.

Негізгі бөлімде мобильді қосымшаның толық техникалық сипаттамасын қарастырамыз. Дипломдық жұмысты жасауда қандай технологиялар қолдандым және не үшін осы технологиялар пайдаланғаным, олардың басқа технологиялардан айырмашылығын талқылып өтеміз.

Қорытынды бөлімде жасалған жұмыстың оң және теріс жақтары қарастырылады.

АННОТАЦИЯ

Это дипломный проект предназначен для автоматизации процесса аренды или лизинга. Пользователи могут легко и быстро арендовать все, что угодно.

Дипломный проект состоит из введения, трех основных разделов и заключения:

Во введении указывается цель выбранной темы, ее актуальность и задачи.

В основном разделе мы рассматриваем подробные технические характеристики мобильного приложения. Мы обсудим, какие технологии я использовал в своей диссертации и почему я использовал эти технологии, их отличия от других технологий.

В заключительном разделе рассматриваются плюсы и минусы проделанной работы.

ANNOTATION

This graduation project is designed to automate the rental or leasing process. Users can easily and quickly rent whatever they want.

The graduation project consists of an introduction, three main sections and a conclusion:

The introduction indicates the purpose of the selected topic, its relevance and objectives.

In the main section, we review the detailed technical specifications of the mobile application. We will discuss what technologies I used in my dissertation and why I used these technologies, how they differ from other technologies.

The final section discusses the pros and cons of the work done.

МАЗМҰНЫ

КІРІСПЕ

Бұл заманда ғаламтор желілері қаттты дамып келе жатыр. Онымен қоса қазіргі кезде ғаламтор қызметтері де жақсы дамуда. Мысалыға алатын болсақ: ғаламтордан керекті затты тапсырыс беру, мейлі ол тамақ болсын немесе киім болсын, онлайн кезекке тұру, мысалы, шаштаразға, кез-келген қызмет көрсету орталығына және т.б. Бұл ғаламтор арқылы жасалатын қызметтер адамдар өмірін біршама жеңілдетті деуге болады және сондай қосымшаларға кун сайын сұраныстар саны да артуда. Сондай қосымшаның бірін біз осы дипломдық жұмысқа алып отырмыз. Бұл қосымша адамдарға қандайда бір затты, мысалы электроника, кітап және т.б. жалға алуға мүмкіндік береді. Бұрынғыдай адамдар кездесіп, оны алып кетудің керегі жоқ, барлығы қосымша арқылы жүзеге асады.

Мен бұл қосымшаның бэк бөлігін жасадым. Бэк бөлігі дегеніміз браузерде емес, серверде жасалатын жұмыстың барлығы деуге болады. Ол қосымшаның ішкі функционалдарына жауап береді. Оны жасау үшін Python тілінің Django фреймворкін таңдадым. Django фреймворкі қазіргі кезде қарқынды дамып келе жатқан фреймворктардың бірі дей аламыз. Бұл фреймворкта қосымшаны жасау процесін жылдамдататын, жеңілдететін дайын компонеттер бар.

Дипломдық жұмыстың мақсаты: жалға алу процесін автоматтандыру болып табылады. Сол арқылы адамдар уақытын тиімді пайдалана алады.

1 Зерттеу бөлімі

1.1 Жобаның өзектілігі

Қазіргі кезде ғаламтор желілері қатты дамып келе жатыр және олар адам өмірінің ажырамас бөлігі болып табылады. Онымен қоса қазіргі кезде ғаламтор қызметтері де жақсы дамуда. Сол арқылы көптеген процестерді үйден шыкпай жасауға болады. Бұл дипломдық жобада сондай бір қызметті автоматтандыру.

Жоба жалға алу және жалға беру процесін автоматтандыру болып табылады. Себебі бұл кезге дейін ондай электронды алаң болмады. Қандай да бір затты жалға алғымыз келсе, алдымен біз заттың иесімен кездесіп, келісіп отыру керек және қолма-қол келісім жұмысы ыңғайсыз, әрі сенімсіз. Бұл былайша айтқанда уақыт жоғалту. Осылардан шығатын қорытынды жалға беру жүйесі дамымаған. Сондықтан жобаның өзектілігі жеке тұлғаларға күнделікті қолданыстағы құралдарды және заттарды жалға беруге немесе алуға арналған мобильді қосымшаны жобалау және жасау болып табылады.

Маған қойылған негізгі тапсырмалар:

– дерекқор құру;
– API шығару;
– админ интерфейсін жасау.

1.2 UML тілінде жобаны модельдеу

Негізінен қандайда бір жобаны жасап бастар алдында оны толықтай зерттеу қажет. Жобаның орындалатын этаптарын құрастырып, функционалдарын ойлап тауып, толықтай модельдеп, зерттеу керек. Бұл былай қарағанда өажет емес процес болып көрінуі мүмкін, бірақ негізінен ол жімыс барысын тездетуге, алда шығатын қателерді азайтуға жәнеде жұмыс барысын нақты қалай жүргізу, яғни бастау керек екенін көрсетеді. Былайша айтқанда, бәріміз «Жеті рет өлшеп, бір рет кес» деген сөзді естіген шығармыз. Бағдарламалауда да солай. Ең дұрысы оны іске асыруға уақыт бөлмей тұрып, оны әрқашан қарастырған жөн. Жүзеге асыру кезінде көбінесе сыныптар құру, олардың өзара әрекеттесуін ойластыру қажет. Көбінесе мұның визуалды көрінісі мәселені ең дұрыс жолмен шешуге көмектеседі. Бұл үшін UML қолдансақ болады.

Егер сіз іздеу жүйелеріндегі суреттерді қарасаңыз, UML диаграммалар, сызықтар мен квадраттар туралы болатыны түсінікті болады.

Маңыздысы – UML (Unified Modeling Language) бірыңғай модельдеу тілі ретінде аударылады. Мұнда Unified сөзі маңызды. Яғни, біздің суреттерді біз ғана емес, UML-ді білетіндер де түсінетін болады. Диаграмма салуға арналған

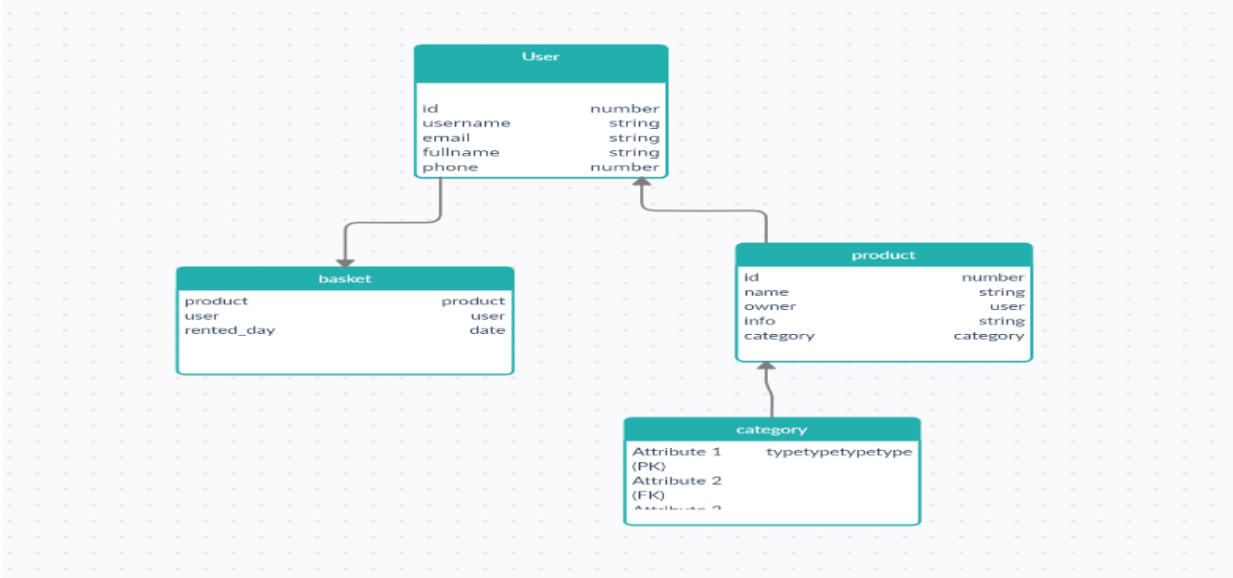
халықаралық тіл осындай екен.

UML – бағдарламалық жасақтама, бизнес-процестерді модельдеу, жүйелік инженерия және ұйымдық құрылымдарды бейнелеу кезінде объектілерді модельдеуге арналған графикалық сипаттама тілі [1].

UML артықшылықтары:

1. тапсырмаға әр түрлі көзқараспен қарау мүмкіндігі;
2. басқа бағдарламашыларға тапсырманың мәнін және оны қалай жүзеге асыруды түсіну оңайырақ;
3. диаграммаларды олардың синтаксисімен тез танысқаннан кейін оқуға салыстырмалы түрде оңай.

UML тек сыныптардың құрылымын сипаттаумен шектелмейді. UML диаграммаларының көптеген түрлері бар. Айталық, сіз жүйені жобалауыңыз керек. Бірнеше сабақты енгізуді бастамас бұрын, сіз жүйені тұжырымдамалық түрде түсінгіңіз келеді – маған қандай сыныптар керек? Бұл сыныптарда қандай функционалдылық пен ақпарат болады? Олар бір-бірімен қалай әрекет етеді? Бұл сабақтарды кім көре алады? Тағыда басқа. Осы кезде кластар диаграмасын қолданамыз. Кластар диаграммалары – жүйеде сабақтарды кодтауды бастамас бұрын оларды бейнелеудің тамаша тәсілі. Олар сіздің жүйеңіз құрылымының статикалық көрінісі [2]. Жобаның кластар диаграммасы 1.1-суретте көрсетілген.



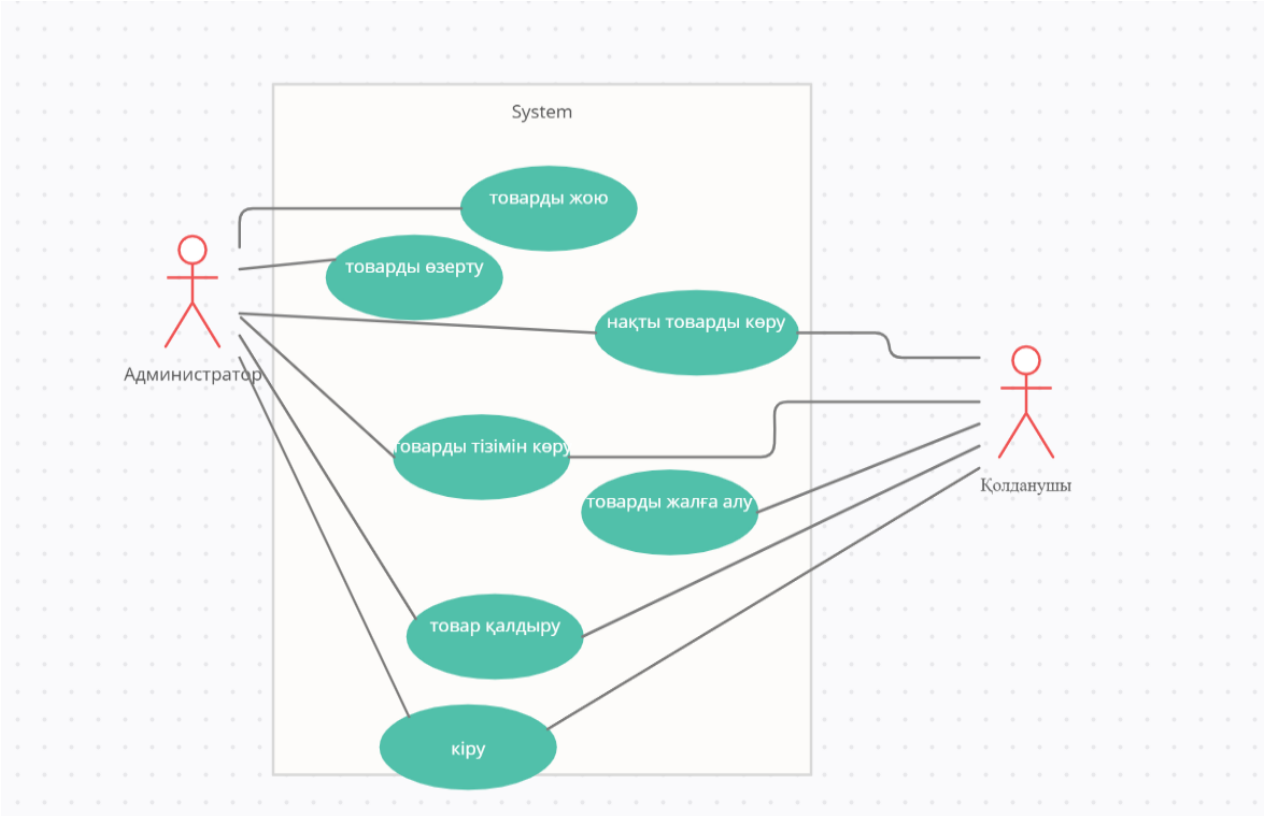
1.1-сурет – Кластар диаграммасы

Қолдану жағдайының (UseCase) диаграммасы өте егжей-тегжейлі емес, мысалы, қадамдар орындалу ретін модельдейді деп күтпеңіз. Керісінше, дұрыс пайдалану жағдайының диаграммасы пайдалану жағдайлары, актерлер мен жүйелер арасындағы байланысты жоғары деңгейлі шолуды бейнелейді. Сарапшылар мәтіндік сипаттаманы сипаттайтын кейсті толықтыру үшін пайдалану схемаларын қолдануды ұсынады.

UML қолдану жағдайының (UseCase) диаграммалары:

1. Жүйе қолданушылардың өзара әрекеттесу мақсаттарын ұсыну;
2. Жүйедегі функционалдық талаптарды анықтау және ұйымдастыру;
3. Жүйенің мазмұны мен талаптарын көрсету;
4. Қолдану жағдайындағы оқиғалардың негізгі ағымын модельдеу.

Жобаның UseCase диаграммасы 1.2-суретте көрсетілген.



1.2-сурет – UseCase диаграммасы

2 Технологиялық бөлім

2.1 Фреймворк ұғымына түсінік

Фреймворктер – бұл техникалық жағынан күрделі немесе ауыр жобаларды құруды және оларға қызмет көрсетуді жеңілдететін бағдарламалық өнімдер. Фреймворк, әдетте, тек бағдарламалық жасақтаманың негізгі модульдерін қамтиды және жобаның барлық компоненттерін әзірлеуші олардың негізінде жүзеге асырады. Осылайша, дамудың жоғары жылдамдығына ғана емес, сонымен қатар шешімдердің жоғары өнімділігі мен сенімділігіне қол жеткізіледі.

Фреймворктерді қолданудың басты артықшылықтарының бірі – оның бір бірыңғай құрылымы болады. Сондықтан құрылымдарға негізделген қосымшаларды сақтау және нақтылау әлдеқайда жеңіл, өйткені компоненттерді ұйымдастырудың стандартталған құрылымы осы платформадағы барлық әзірлеушілерге түсінікті және принципін түсіну үшін архитектураны түсіну көп уақытты қажет етпейді. Веб-қосымшаларды әзірлеуге арналған көптеген фреймворктар MVC парадигмасын қолданады (модель-представление-контроллер), яғни көптеген фреймворктерде қолданбалы компоненттерді ұйымдастырудың бірдей тәсілі бар, және бұл қолданбаның архитектурасын тіпті фреймворкта түсінуді жеңілдетеді. әзірлеушіге таныс емес.

Фреймворктер кітапхана емес.

Кітапхана – бағдарламалық жасақтама архитектурасының қарапайым бөлігі. Бағдарламалық кітапхананы негізгі бағдарламалық өнімнің архитектурасына әсер етпей және оған ешқандай шектеулер қоймай, ұқсас функционалдылықтың ішкі жүйелерінің жиынтығы ретінде пайдалануға болады.

Екінші жағынан, рамка әзірлеушіге қажетті функционалдылықты қамтамасыз етіп қана қоймайды, сонымен қатар қолданбалы архитектураны құру ережелерін белгілейді, дамудың бастапқы сатысында әдепкі мінез-құлықты орнатады, кеңейту қажет болатын және көрсетілген талаптарға сәйкес өзгертілді. Фреймаске сонымен қатар үлкен бағдарламалық жасақтаманың әртүрлі компоненттерін жасау мен біріктіруді жеңілдететін утилиталық бағдарламалар, код кітапханалары, сценарийлер тілі және басқа бағдарламалық қамтамасыздандыру кіруі мүмкін.

Фреймворктардың жақсы жақтары:

1. фреймворк жасалған жобаны, толықтай қолмен жазылған жобаға қарағанда, әрі қарай колдау, жетілдіру ыңғайлы;

2. бастапқыда жүйеге енгізілгендерді ғана емес, кез-келген бизнес-процестерді жүзеге асыруға болады (және салыстырмалы түрде қарапайым). Сондай-ақ, фреймворктарға негізделген жобалар оңай масштабталады және жаңартылады;

3. рамалық шешімдер CMS пен өздігінен жазылатын жүйелерге

қарағанда әлдеқайда жылдам жұмыс істейді және жұмыс жүктемесіне төтеп береді. Сондықтан көптеген танымал интернет-дүкендер қораптағы CMS-те емес, фреймворктарда жұмыс істейді. Қауіпсіздік тұрғысынан фреймворктарға негізделген шешімдер өздігінен жазылатын жүйелерден едәуір жоғары және оларды CMS-пен салыстыруға болады (әдетте, фреймворктарға негізделген сайттар тіпті қауіпсіз).

Қазіргі таңда ең танымал және көп қалнылатын фремворктар олар:

1. Node.js (Express);
2. Django;
3. Rails;
4. Laravel;
5. Spring.

Мен бұл дипломдық жұмыста бэк, яғни серверлік жағын жасаймын. Сондықтан, мен қазіргі таңда ең танымал фреймворк – Django фреймворкін қолданамын.

2.2 Серверлік бөлім – Django фреймворкі

Серверлік бөлім – бұл сервистиің бағдарламылық-ақпараттық бөлігі. Ол веб-сайттың, приложениенің логикасын құрастыруға қажетті құралдар жиынтығы. Былайша айтқанда ол көзге көрінбейтін, өолданушы мен серверді байланыстыратын бөлік деуге болады.

Бэкенд әзірлеуші осыларды құрастыру үшін өзіне ыңғайлы немесе тапсырыс беруші сұраған технологияларды қолдана алады. Мысалы: PHP, Ruby, Python, Java. Одан басқа, backend СУБД мен байланысады (MySQL, PostgreSQL, SQLite, MongoDB және т.б.). Мен Python тіліндегі Django фреймворгін, ал СУБД-ға PostgreSQL қолдандым. Жалпы бэкенд әзірлеуші мына тапсырамаларды қарастырады:

1. приложение функцияларының және логикасын тұрақты жұмыс жасауын;
2. дерекқорымен жұмыс жасау;
3. приложениенің жұмыс жасау логикасын құру;
4. API;
5. тестілеу;
6. басқа сервистермен байланыстыру.

Django – бұл Python тіліндегі фреймворк. Оның негізгі принциптерінің бірі DRY (don't repeat yourself). DRY (don't repeat yourself) ол бағдарламылық технологияны құрастырудың принципі, әр-түрлі ақпараттың қайталануын азайтуға бағытталған. Бұл принцип дерекқор схемаларына, тестілеуге, құрастыру кезінде қолданылады. Django бір немесе бірнеше бір-бірімен байланыса беретін приложениелерден құрастырылады. Оның архитектурасы «Модель-Представление-Контроллер» (MVC) құсайды. Бұл жерде контроллер деп фреймвортың View-ін айта аламыз, ол негізгі логиканы

қамтиды, ал представление содан шыққан урлде көрстегілген шаблон бөлігі, ол ақпаратты шығару, көру үшін. Модель бұл дерекқордағы таблицалар деуге болады [3]. Веб-фреймворк Django көбісі білетін ірі компанияларда қолданылады. Мысалы: Instagram, Disqus, Mozilla, The Washington Times, Pinterest, lamoda және т.б.

2.3 Django фреймворкінің өзіндік ерекшеліктері

Django – бұл Python-да жазылған тегін және ашық бастапқы коды бар веб-қосымшаның құрылымы. Фреймворк – бұл веб-сайттарды жылдам және оңай дамытуға көмектесетін компоненттер жиынтығы.

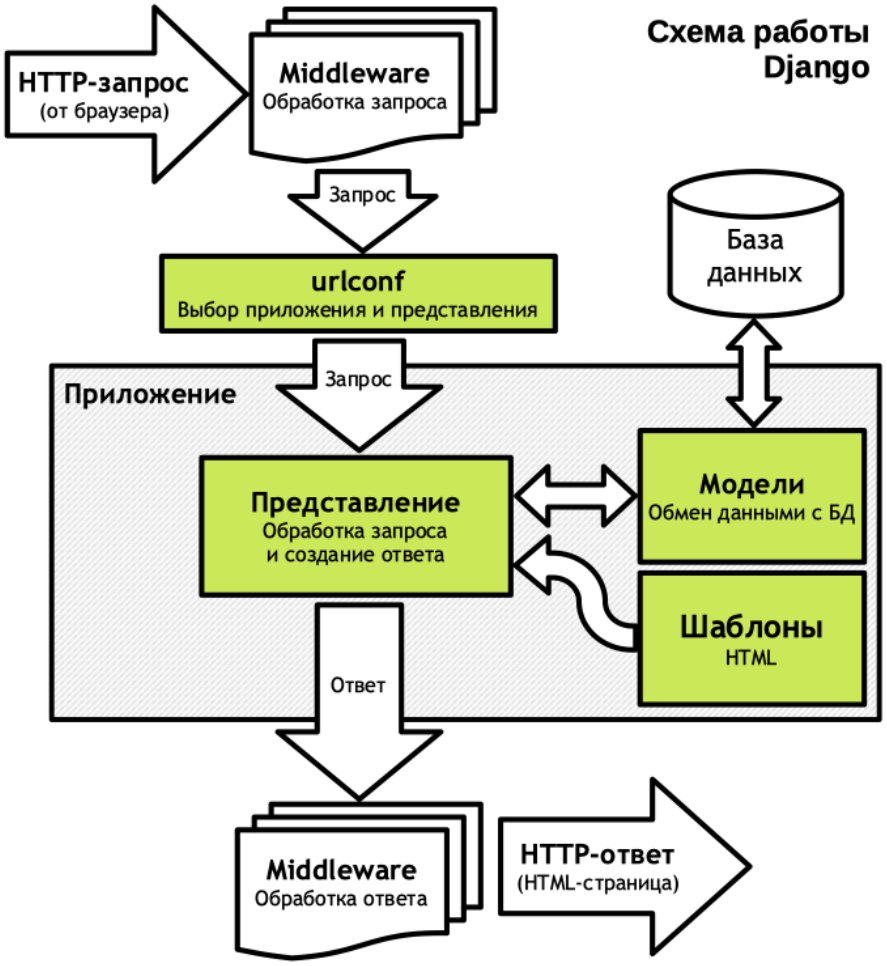
Веб-сайттарды жасаған сайын сізге ұқсас компоненттер қажет: пайдаланушылардың аутентификациясы (кіру, шығу, тіркелу) тәсілі, сайттың бақылау тақтасы, формалар, файлдарды жүктеуге арналған құралдар және т.б. Бақытымызға орай, басқа адамдар веб-дамудың ұқсас проблемаларын байқады, сондықтан олар бізді қолдануға дайын шаблондармен қамтамасыз ететін құрылымдар құрды (Django және басқалары).

Серверге сұраныс түскен кезде, ол Django-ға жіберіледі, ол одан нақты не сұралып жатқанын анықтауға тырысады. Алдымен ол веб-парақтың мекен-жайын алып, не істеу керектігін анықтауға тырысады. Django-дағы процестің бұл бөлігі urlresolver арқылы жүзеге асырылады (сайттың мекен-жайы URL – Uniform Resource Locator деп аталады, сондықтан urlresolver, resolver == resolver, белгілі бір мағынаға ие). Ол өте ақылды емес, сондықтан шаблондардың тізімін алып, оларды URL мекенжайына сәйкестендіруге тырысады. Django шаблондарды жоғарыдан төмен қарай тексереді, егер бірдеңе сәйкес келсе, ол сұранысты тиісті функцияға бағыттайды (көрініс деп аталады).

Хатпен почта қызметкерін елестетіп көріңіз. Ол көшеде өтіп, үй нөмірлерін хаттағы мекен-жаймен салыстырады. Егер олар сәйкес келсе, онда ол хат қалдырады. Urlresolver осылай жұмыс істейді!

Бірақ ең қызықты нәрселер view функциясында болады: мысалы, белгілі бір ақпарат үшін мәліметтер базасына қол жеткізе аламыз. Мүмкін пайдаланушы кейбір ақпаратты өзгертуді сұраған шығар? Хатта «Менің жұмыс сипаттамамды өзгертіңізші» деген сияқты. View функциясы сіздің рұқсатыңыздың бар-жоғын тексеріп, содан кейін жұмыс сипаттамасын жаңартып, «дайын!» Деп жібере алады. Содан кейін қарау функциясы жауап тудырады және Django оны қолданушының веб-шолғышына жібере алады.

Django-ның жұмыс істеу принципі 2.1-суретте көрсетілген.



2.1-сурет – Django-ның жұмыс істеу принципі

Django DRY (don’t repeat yourself) принципін жүзеге асырады. Осының арқасында сайттар құру уақыты қысқарады. Яғни, Django қолданған кезде бір кодты бірнеше рет қайта жазудың қажеті жоқ. Фреймворк компоненттерден веб-сайт құруға мүмкіндік береді. Оны Lego-мен бірге бекініс салумен салыстыруға болады. Django фреймворкасы Python бағдарламалау тілінде жазылған, сондықтан оның құрылымы тілдің ерекшелігіне сәйкес келеді. Авторлар MVC үлгісін Django-да іске асырды және ол фреймворктің қазіргі нұсқасында қолданылады [3].

MVC архитектурасы әзірлеушіге қосымшаның визуалды презентациясымен және іскери логикасымен бөлек жұмыс істеуге мүмкіндік береді. Айтпақшы, Django-мен жұмыс жасағанда сарапшылар MVT – Model-View-Template немесе Model-View-Template терминдерін жиі қолданады. MVT компоненттерін бір-біріне тәуелсіз қолдануға болады.

Djangodағы модель мәліметтердің негізгі өрістері мен мінез-құлқын қамтитын мәліметтер туралы ақпарат көзі болып табылады. Әдетте бір модель мәліметтер базасындағы бір кестені көрсетеді. Django PostgreSQL, MySQL, SQLite және Oracle мәліметтер базаларын қолдайды.

Модельдерде мәліметтер туралы ақпарат бар. Бұл деректер атрибуттармен немесе өрістермен ұсынылған. Модель қарапайым класс

болғандықтан, Джангоның басқа қабаттары туралы ештеңе білмейді. Қабаттар арасындағы өзара әрекеттесу API арқылы жүзеге асады.

Модель іскери логикаға, әдістерге, қасиеттерге және деректерді манипуляциялауға байланысты басқа элементтерге жауап береді. Модельдер әзірлеушілерге мәліметтер базасындағы объектілерді құруға, оқуға, жаңартуға және жоюға мүмкіндік береді.

View үш мәселені шешеді: HTTP сұраныстарын қабылдайды, әдістер мен қасиеттермен анықталған іскери логиканы жүзеге асырады және сұраныстарға жауап ретінде HTTP жауабын жібереді. Яғни көрініс модельден деректерді қабылдайды және шаблондарға (шаблондарға) осы деректерге қол жеткізуді қамтамасыз етеді немесе деректерді алдын-ала өңдейді, содан кейін оларды шаблондарға қол жетімді етеді.

Django-да қуатты қарбалас қозғалтқышы және өзіндік белгілеу тілі бар. Шаблондар – бұл мәліметтерді көрсететін HTML коды бар файлдар. Файлдардың мазмұны тұрақты немесе динамикалық болуы мүмкін. Оларда ешқандай бизнес логикасы жоқ. Сондықтан олар тек деректерді көрсетеді.

2.4 Django Rest Framework

REST API – бұл басқа қолданбаларға деректерді берудің стандартталған тәсілі. Содан кейін бұл қосымшалар деректерді өз қалауынша қолдана алады. Кейде, API-лер басқа қолданбаларға деректерге өзгеріс енгізу әдісін ұсынады.

– REST API сұранысының бірнеше негізгі нұсқалары бар;

– GET – ең көп таралған опция, сіз кірген соңғы нүктеге және берілген параметрлерге негізделген кейбір деректерді API-ден қайтарады;

– POST – мәліметтер базасына қосылатын жаңа жазба жасайды;

– PUT – берілген URI мекен-жайы бойынша жазбаны іздейді. Егер ол бар болса, бар жазбаны жаңартыңыз. Егер жоқ болса, жаңа жазба жасаңыз;

– DELETE – берілген URI кезіндегі жазбаны жояды;

– PATCH – жазбаның жеке өрістерін жаңартыңыз.

Әдетте, API – мәліметтер қорының терезесі. API backend дерекқорға сұранысты және жауапты форматтауды өңдейді. Сіз алатын нәрсе – бұл сіз сұраған кез келген ресурстың статикалық жауабы, әдетте JSON форматында.

REST API интерфейстері бағдарламалық жасақтамада кеңінен таралған, әзірлеушіге олардың қалай жұмыс істейтінін білу маңызды дағды болып табылады. API – бұл қосымшалардың бір-бірімен немесе тіпті өз ішіндегі байланыс тәсілі [4].

Аты айтып тұрғандай, Django REST Framework – бұл Django үшін жеңіл құрылым, ол өкілдікті мемлекеттік тасымалдау (REST) идеологиясын қолдайды. Осы фреймворкті қолдану мәліметтер қорына сұраныстарды стандарттауды жеңілдетеді және сонымен бірге біздің сайтқа RESTful WEB API құруды жеңілдетеді [5].

Мүмкіндіктері:

1. Django-ның Class Based Views функциясын пайдаланып ресурстарды көрсету;

2. ModelResources қолдайды және енгізуді тексереді;

3. қосылатын талдаушылардың болуы, кескінделуі, авторизация және қол жеткізу құқықтары бәрі оңай конфигурацияланады;

4. HTTP сұраныстарының тақырыптарында Access көмегімен мазмұн түрін көрсету;

5. расталған формаларды қолдау.

2.5 Дерекқор қоры

Дерекқор қоры – бұл өзара байланысты ақпаратты, негізінен үлкен көлемді сақтауға, өзгертуге және өңдеуге арналған ұйымдастырылған құрылым. Деректер қоры деректердің айтарлықтай көлемі бар динамикалық сайттар үшін белсенді қолданылады – көбінесе интернет-дүкендер, порталдар, корпоративті сайттар. Мұндай сайттар әдетте серверлік бағдарламалау тілін (мысалы, PHP) қолдана отырып немесе CMS (мысалы, WordPress) негізінде әзірленеді және HTML сайттарына ұқсас дайын беттері болмайды. Динамикалық сайттардың парақтары веб-серверге клиенттің тиісті сұранысынан кейін сценарийлер мен мәліметтер базаларының өзара әрекеттесуі нәтижесінде «жылдам» қалыптасады.

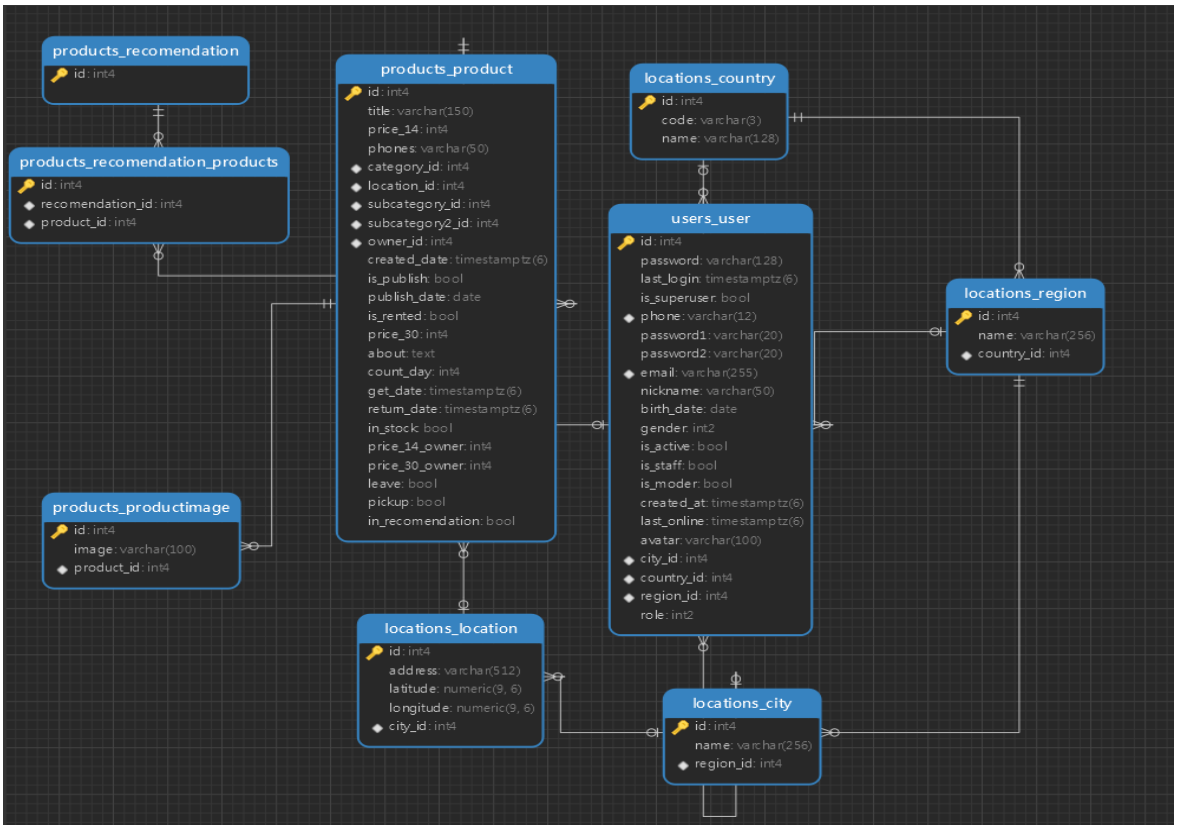
Деректер базасы тұрғысынан ДҚБЖ тұжырымдамасын қарастырған жөн. Деректер базасын басқару жүйесі (ДҚБЖ) – бұл мәліметтер қорының жаңа құрылымын құруға, оны толтыруға, мазмұнын редакциялауға және ақпаратты бейнелеуге қажетті бағдарламалық құралдар жиынтығы. Ең көп таралған МҚБЖ-ға MySQL, PostgreSQL, Oracle, Microsoft SQL Server жатады [8].

2.6 PostgreSQL

PostgreSQL – бұл тек реляциялық емес, объектілік-реляциялық мәліметтер қорын басқару жүйесі. Бұл MySQL, MariaDB және Firebird сияқты басқа ашық бастапқы дерекқорлар SQL дерекқорларына қарағанда біршама артықшылықтар береді [6].

Объектілік-реляциялық мәліметтер қорының іргелі сипаттамасы – бұл тапсырыс беруші объектілерге және олардың мінез-құлқына, оның ішінде мәліметтер типтеріне, функцияларына, операцияларына, домендеріне және индекстеріне қолдау. Бұл Postgres-ті керемет икемді және сенімді етеді. Басқа нәрселермен қатар, ол күрделі деректер құрылымын құруды, сақтауды және

алуды біледі. Postgres қолдайтын деректер түрлерінің кең тізімі бар. PostgreSQL сандық, өзгермелі нүкте, мәтіндік, логикалық және басқа да күтілетін деректер түрлерінен (сонымен қатар олардың көптеген вариацияларынан) басқа Uuid, ақшалай, санақ, геометриялық, екілік, желілік адрестерді, биттік жолдарды, мәтіндік іздеуді, xml-ді, json, массивтер, құрама типтер мен диапазондар, сонымен қатар объектілерді идентификациялау және журналдардың орналасуы үшін кейбір ішкі типтер. MySQL, MariaDB және Firebird-те де осындай кейбір деректер түрлері бар, бірақ тек Postgres ғана олардың барлығын қолдайды деп айту әділетті. Postgres объектілік-реляциялық мәліметтер қоры болғандықтан, мәндер массивін бар мәліметтер типтерінің көпшілігінде сақтауға болады.
Жобаның дерекқор құрылымы 2.2-суретте көрсетілген.



2.2-сурет – Дерекқордың құрылымы

2.7 Vue.js Javascript-framework

Vue – JavaScript бағдарламалау тілінде қолданушы интерфейстерін дамытуға арналған веб-фреймворк. Vue қолданыстағы қосымшаға біртіндеп енгізу үшін жасалған. Ол презентация деңгейінің әр түрлі міндеттерін шешеді

(қарау), басқа кітапханалармен жұмысты жеңілдетеді және күрделі бір парақты қосымшаларды (SPA, Single-Page Applications) құруға мүмкіндік береді [7].

Фрейворкпен жұмыс істеу үшін HTML, CSS және әрине JavaScript-ті кем дегенде негізгі деңгейде білу керек. Vue тек «презентация қабатында» жұмыс істейтіндіктен және орта бағдарламалық жасақтама мен артқы жағында қолданылмайтындықтан, ол басқа жобалармен және кітапханалармен оңай интеграцияланады. Vue.js функциялары артықшылықтары:

1. реактивті интерфейстер;
2. декларативті көрсету;
3. мәліметтерді байланыстыру;
4. директивалар (барлық директивалар «v-» префиксімен жазылған. мемлекеттік мән директиваға беріледі, және html атрибуттары немесе vue js оқиғалары аргумент ретінде қолданылады);
5. үлгі логикасы;
6. компоненттер;
7. оқиға өңдеу;
8. қасиеттері;
9. CSS ауысулары мен анимациялары;
10. сүзгілер.

Vue-дің Angular және React арасындағы негізгі айырмашылықтарын қарастырайық.

1. *Компоненттер*

Бұл құрылымдардың барлығы компоненттерге негізделген. Айтуынша, React және Vue мылқау деп аталатын компоненттермен – кіріс және қайтару элементтерін шығыс ретінде қабылдайтын кішігірім, азаматтығы жоқ функциялармен жақсы жұмыс істейтінін ескеру қажет. Vue.js компоненттерінде олардың атауларына арналған арнайы талаптар жоқ, бірақ теңшелетін компоненттерге арналған W3C ережелерін ұстану ұсынылады – кіші әріптермен және дефиспен бөлінген белгілерді қолданыңыз.

2. *Кітапханаға немесе фреймворк*

Angular – бұл кітапхана емес, фреймворк, өйткені онда қолданбаның құрылымына қатысты нақты нұсқаулар бар, сонымен қатар кең функционалдылыққа ие. Angular – бұл басқа қосымшаларға талдау жасауды немесе қосымша құралдарды қолдануды қажет етпейтін корпоративті қосымшаларға арналған толық шешім. React және Vue, керісінше, жан-жақты. Олардың кітапханалары барлық пакеттер түрлерімен байланыса алады, дегенмен Vue-де олардың саны аз, өйткені ол әлі де жас.

3. Қолданудың икемділігі

Javascript кітапханасын бастапқы кодқа қосу арқылы сіз React немесе Vue-мен жұмыс істей аласыз. Angular-да ол мүмкін емес, өйткені ол күрделі тапсырмаларға арналған. Микроқызметтер мен микроқолданбалар туралы айтатын болсақ, React және Vue қолданбаларыңыздың көлемін көбірек басқаруға мүмкіндік береді, бұл сізге белгілі бір жағдайларда қажет элементтерді ғана таңдауға мүмкіндік береді. Олар сонымен қатар бір беттік қосымшалардан микросервистерге көшудің икемділігін ұсынады, бұл сіздің ескі қолданбаңыздың бөліктерін пайдалануға мүмкіндік береді. Бұрыштық, кең функционалдылығына байланысты, бір парақты қосымшаларды өздері жасауға ыңғайлы.

4. Өнімділік және файл өлшемі

Angular құрылымы айтарлықтай көлемді. Кең функционалдығы арқасында, ықшамдалған файл өлшемі шамамен 143к құрайды. React оған қарағанда 3 есе ықшам, ал Vue 4 есе ықшам болып келеді.

React және Vue құрылымдық құжаттың объектілік көрінісінің көшірмесін жасайтын және көріністің өзінен гөрі визуалды көшірмемен жұмыс істеуге мүмкіндік беретін виртуалды DOM-ға (құжат нысаны моделі) ие. Бұл тәсіл фреймворктардың жұмысын жақсартуға көмектеседі, осылайша сіздің қосымшаңыз тезірек жұмыс істейді. Vue, әсіресе, керемет өнімділікке және жадыны терең бөлуге ие.

3 Жобалау бөлімі

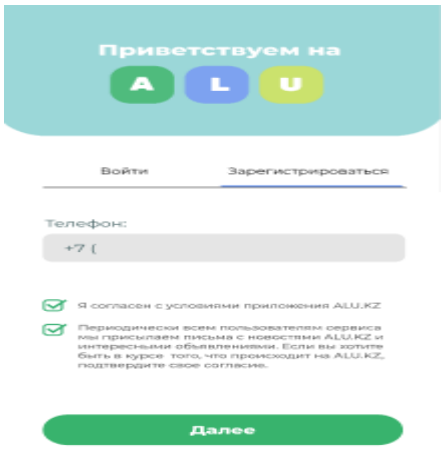
3.1 Жоба туралы

Бұл дипломдық жобада былайша айтқанда кез-келген заттарды жалға алуға, беруге арналған платформа. Бағдарламаның жұмыс жасау этаптары төмендегідей:

– бағдараламаға тіркелу;
– кіру;
– басты бетте категориялар тізімі және товарлар тізімі шығады;
– керекті категорияны басқанда сол категорияға тиісті товарлар шығады;
– керекті товарды басқанда сол товар туралы толықтай ақпарат шығады;
– товарларды себетке (корзина) сала аламыз;
– себетте товарды калай алу керек екені және төлем жасау беті ашылады.

3.2 Веб қосымшаның логикалық құрылымы

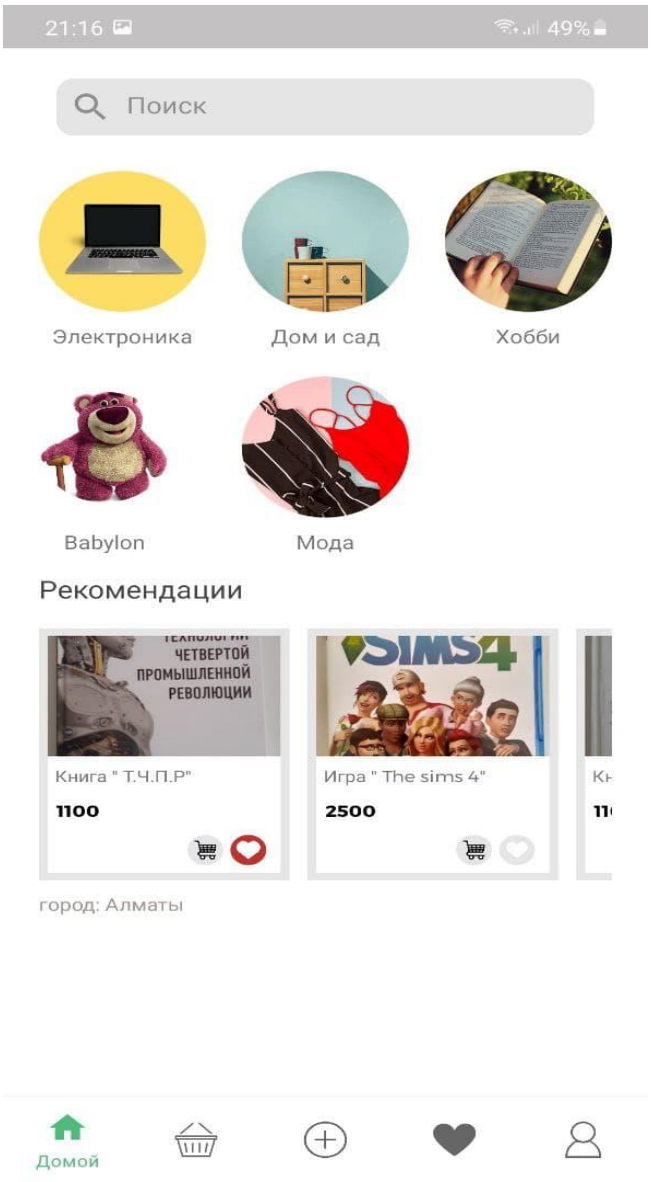
Қолданбаны ашқанда бірінші тіркелу немесе кіру беті ашылады. Бұл бетте телефон нөмері енгізетін арнайы орынан және «Далее» батырмасы тұрады. Тағы телефон нөмері мен құпия сөз орындары қатаң ережелерден өту керек. Телефон нөмері тек сандардан тұруы керек және +7 ден басталады.



3.1-сурет – Тіркелу / Кіру беті көрінісі

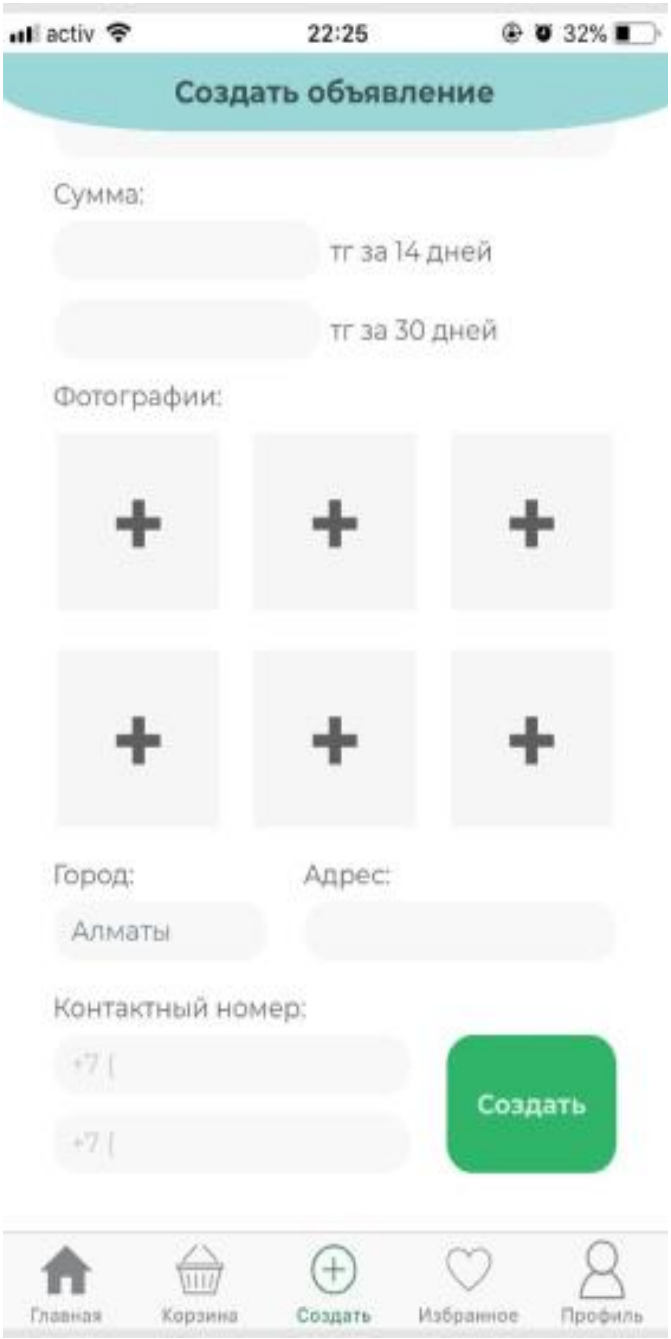
Одан кейін басты бетке көше аламыз. Бұл бетте товарларды іздейтін арный орын, категориялар тізімі, товарлар тізімі бар. Нақты бір категорияны басқанда сол категорияға қатысты товарлар шығады. Астыңғы жағында Tab bar орналасқан. Онда 5 пункт бар. Олар:

1. Басты бет «Домой»;
2. Себет (корзина) таңдалған товарлар;
3. Жаңа товар салу (Жалға беру үшін);
4. Таңдаулылар (ұнаған товарлар);
5. Менің парақшам.



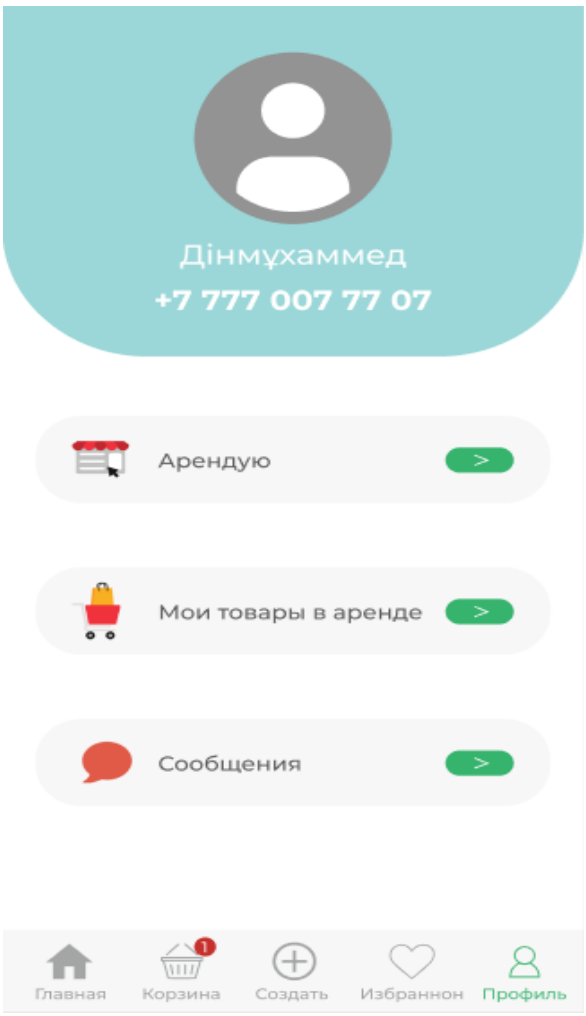
3.2-сурет – Басты бет көрінісі

Жаңа товар салу (құру) беті. Бұл бетте қолданушы жалға беру үшін өзінің товарын сала алады.



3.3-сурет – Товар құру бетінің көрінісі

Менің парақшам (Профиль) бетінде қолданушы өзі туралы ақпаратты көре алады. «Арендую» бетінде қолданушыны қазір жалға алған товарлары көрсетіледі. «Мой товары в аренде» бетінде қолданушығың жалға берілген товарлары көрсетіледі. «Сообщения» бетінде келіп түскен хабарламаларды көре алады.



3.4-сурет – Менің парақшам бетінің көрінісі

ҚОРЫТЫНДЫ

Дипломдық жұмысты қорытындылай келе, біздің алған жоба адамдар өмірін сәл де болса жеңілдетеді деген сенімдеміз. Бұл жобаның артықшылығы, ерекшелігі деугеде болады, ол жалға алу, жалға беру процестерін автоматтандыру.

Жобаны жасауда ең соңғы технологияларды қолдандық дей аламыз. Сондай технологияның бірі – Django framework. Оның басқа қарсыластармен салыстырғанда артықшылықтары өте көп. Соның негізінде бағдарлама жылдам жасайды және жобаны жасау барысыда тез әрі оңай және қызықты болды. Сондықтан бағдарламаны алдыңғы уақытта әрі қарай жетілдіру мақсаты тұр.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1 Зачем нам UML? // Сайттың электронды нұсқасы https://habr.com/ru/post/458680/

2 UML для самых маленьких: диаграмма классов // Сайттың электронды нұсқасы https://habr.com/ru/post/511798/

3 Почему Django — лучший фреймворк для разработки сайтов // Сайттың электронды нұсқасы https://ru.hexlet.io/blog/posts/pochemu-django-luchshiy-freymvork-dlya-razrabotki-saytov

4 Что такое Django REST Framework? // Сайттың электронды нұсқасы https://mkdev.me/posts/chto-takoe-django-rest-framework

5 Кратко о Django Rest Framework // Сайттың электронды нұсқасы https://django.fun/tutorials/kratko-o-django-rest-framework/

6 Чем PostgreSQL лучше других SQL баз данных // Сайттың электронды нұсқасы https://habr.com/ru/post/282764/

7 Vue.js: особенности, применение и отличия от других Javascript фреймворков // Сайттың электронды нұсқасы https://stfalcon.com/ru/blog/post/vue-js-guide-to-tech

8 Что такое база данных // Сайттың электронды нұсқасы https://hostiq.ua/wiki/database/

А Қосымшасы
(міндетті)

Техникалық тапсырма

А.1 Жалпы сипаттама

Жоба жалға алу және алу процестерін автоматтандыруға, оларды ыңғайлы және де жылдамдатуға арналған.

А.2 Функционалдық сипаттама

Орындалатын функциялар:
- авторизация (телефон нөмерін жазу);
- СМС кодты енгізу;
- товарларды көру;
- оларды керекті категориялар бойынша іріктеу;
- таңдаулыларға сала алу;
- себетке (корзина) салу;
- жаңа товар құру;
- товарын жоя алу;
- товарларының тарихын көре алу.

А.3 Сенімділікке талап

Бағдаламадағы сенімділікке қатысты талаптар:
- тіркелмеген адамдар жалға ала алмайды;
- тіркелмеген адамждар жаңа товар құра алмайды.

А.4 Программалық интерфейске талап

Жүйе әзірлеуге қажетті бағдарламалық компоненттер:
– VS Code редакторы;
– PostgreSQL және PgAdmin бағдарламалары;
– Python;
– PIP менеджерлік пакеті;
– Django framework;
– node.js;
– npm менеджерлік пакеті
– Vue.js.

А.5 Пайдаланушы интерфейстер

Бағдарлама IOS және Android платформаларында жұмыс жасайды

А.6 Терминдер және қысқартулар

Терминдер, қысқартулар және олардың анықтамалары төмендегі кестеде көрсетілген.

А.1-кесте – Анықтамалар, терминдер және қысқартулар

Терминдер және қысқартулар	Анықтамалар
Фреймворк	техникалық жағынан күрделі немесе ауыр жобаларды құруды және оларға қызмет көрсетуді жеңілдететін бағдарламалық өнімдер.
Сервер	Сервер – бұл бағдарламалық кодтарды орындау, ақпаратты сақтау, пайдаланушылар мен мәліметтер базасына қызмет етудің белгілі бір міндеттерін шешуге арналған компьютер.

А.1-кестенің жалғасы

REST	дистрибутивтік жүйелерге арналған бағдарламалық жасақтама стилі (мысалы, танымал www). Әдетте, ол веб-қызметтерді құру үшін қолданылады.
Django	Python тіліндегі фреймворк
PostgreSQL	объектілік-реляциялық мәліметтер қорын басқару жүйесі. Бұл MySQL, MariaDB және Firebird сияқты басқа ашық бастапқы дерекқорлар SQL дерекқорларына қарағанда біршама артықшылықтар береді.
MVC	MVC моделді қарау-контроллер дегенді білдіреді. Бұл әртүрлі мәселелерді шешуге жауап беретін блоктарды бөлуді көздейтін кодты ұйымдастыру тәсілі. Бір блок қосымшаның деректері үшін жауап береді, екіншісі сыртқы түріне жауап береді, ал үшіншісі қосымшаның жұмысын басқарады
API	бір компьютерлік бағдарламаның екінші бағдарламамен өзара әрекеттесу тәсілдерінің сипаттамасы.

Б Қосымшасы

(міндетті)

Бағдарламаның мәтіні

1. *models.py*

```
class Product(models.Model):
    title = models.CharField(max_length=150)
    about = models.TextField(blank=True, null=True)
    price_14 = models.IntegerField()
    price_30 = models.IntegerField(blank=True, null=True)

    price_14_owner = models.IntegerField(blank=True, null=True)
    price_30_owner = models.IntegerField(blank=True, null=True)
    phones = ArrayField(models.CharField(max_length=50),  10)
    location = models.ForeignKey("locations.Location",
on_delete=models.CASCADE, null=True, blank=True,
related_name="location_pr")
    category = models.ForeignKey("categories.category",
on_delete=models.SET_NULL, null=True, blank=True)
    subcategory = models.ForeignKey("categories.subcategory",
on_delete=models.SET_NULL, null=True, blank=True)
    subcategory2 = models.ForeignKey("categories.sub_subcategory",
on_delete=models.SET_NULL, null=True, blank=True)
    owner = models.ForeignKey("users.User", on_delete=models.CASCADE,
related_name="my_product", null=True, blank=True)
    is_publish = models.BooleanField(default=False, blank=True, null=True)
    in_recomendation = models.BooleanField(default=False, blank=True,
null=True)
    is_rented = models.BooleanField(default=False, blank=True)
    in_stock = models.BooleanField(default=False, blank=True)
    leave = models.BooleanField(default=False, blank=True)
    pickup = models.BooleanField(default=False, blank=True)
    count_day = models.IntegerField(blank=True, null=True)
    get_date = models.DateTimeField(blank=True, null=True)
    return_date = models.DateTimeField(blank=True, null=True)
    publish_date = models.DateField(auto_now=False, auto_now_add=False,
blank=True, null=True)
```

```
created_date = models.DateTimeField(auto_now_add=True, blank=True,
null=True)

    def __str__(self):
        return self.title + ", " + str(self.id)
class Rented(models.Model):
    TYPE_DELIVERY = 1
    TYPE_PICKUP = 2
    TYPE_GET_PRODUCT = (
        (TYPE_DELIVERY, 'Достовка'),
        (TYPE_PICKUP, 'Самовывоз')
    )
    product = models.ManyToManyField("products.Product", blank=True,
related_name="rented_obj")
    user = models.ForeignKey("users.User", on_delete=models.CASCADE,
related_name="i_rent")
    amount = models.IntegerField()
    rented_day = models.DateField(blank=True, null=True)
    deadline = models.DateField(blank=True, null=True)
    get_product = models.SmallIntegerField(choices=TYPE_GET_PRODUCT,
blank=True, null=True)
    return_product = models.SmallIntegerField(choices=TYPE_GET_PRODUCT,
blank=True, null=True)
    get_address = models.TextField(blank=True, null=True)
    return_address = models.TextField(blank=True, null=True)
    get_date = models.DateTimeField(blank=True, null=True)
    return_date = models.DateTimeField(blank=True, null=True)
    is_checked = models.BooleanField(default=False, blank=True, null=True)
    is_rented = models.BooleanField(default=False, blank=True, null=True)
    is_canceled = models.BooleanField(default=False, blank=True, null=True)
    is_ended = models.BooleanField(default=False, blank=True, null=True)

    def __str__(self):
        return "id {}, user {}".format(self.id,self.user.phone)

    def chech(self):
        p = self.product.all()
```

```
        c = "False"
        for i in p:
            if i.in_stock == True:
                c = "True"
            else:
                c = "False"
                break
        return self.is_checked

class User(AbstractBaseUser, PermissionsMixin):
    ROLE_SALES_DEPARTMENT = 1
    ROLE_CONTROL_DEPARTMENT = 2
    ROLE_CHOICES = (
        (ROLE_SALES_DEPARTMENT, 'Отдел продаж'),
        (ROLE_CONTROL_DEPARTMENT, 'Отдел контроля')
    )
    GENDER_MALE = 1
    GENDER_FEMALE = 2
    GENDER_CHOICES = (
        (GENDER_MALE, _('Male')),
        (GENDER_FEMALE, _('Female')),
    )
    phone_regex = RegexValidator( regex = r'^\+?1?\d{7,12}$',
                        message = "Phone number in the format '+77777777'. Up to 12
digits")
    phone = models.CharField(max_length=12,validators = [phone_regex],
unique=True)
    password1 = models.CharField(max_length=20, blank=True, null=True)
    password2 = models.CharField(max_length=20, blank=True, null=True)
    email = models.EmailField(max_length=255, unique=True, blank=True,
null=True)
    nickname = models.CharField(max_length=50, blank=True, null=True)
    birth_date = models.DateField(null=True, blank=True,
                            auto_now=False,
                            auto_now_add=False)
    gender = models.SmallIntegerField(choices=GENDER_CHOICES,
```

```
 null=True, blank=True)
    country = models.ForeignKey("locations.Country",
on_delete=models.CASCADE, blank=True, null=True)
    region = models.ForeignKey("locations.Region",
on_delete=models.CASCADE, blank=True, null=True)
    city = models.ForeignKey("locations.City", on_delete=models.CASCADE,
blank=True, null=True)
    role = models.SmallIntegerField(choices=ROLE_CHOICES, blank=True,
null=True)
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)
    is_moder = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    last_online = models.DateTimeField(null=True, blank=True)

    favorites = models.ManyToManyField("products.Product",
related_name="favs", blank=True)
    basket = models.ManyToManyField("products.Product",
related_name="basket", blank=True)
    USERNAME_FIELD = 'phone'
    REQUIRED_FIELDS = []
    objects = UserManager()

    def __str__(self):
        return str(self.id) + ", " + self.phone
```

2. *urls.py*

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('users/', include('users.urls')),
    path('categories/', include('categories.urls')),
    path('products/', include('products.urls')),
    path('locations/', include('locations.urls')),
    path('basket/', include('basket.urls')),
    path('message/', include('message.urls')),
```

```
]
urlpatterns = [
    path('phone/otp/', PhoneCode.as_view()),
    path('register/', Register.as_view()),
    path('phone/check', LoginUser.as_view()),
    path('login', Logined.as_view()),
    path("admin/login", login_admin.as_view()),
    path("detail/<id>", detailUser.as_view()),
    path('avatar', Avatar.as_view()),
    path("push", pushRegister.as_view()),
    path("private/policy", privatepolicy)
]

urlpatterns = [
    path("recomendation", recomendations.as_view()),
    path("create", product.as_view()),
    path('filter', getProduct.as_view({'get': 'list'})),
    path("favorites", favorites.as_view()),
    path('publish/<id>', ProductPublish.as_view()),
    path("change", ProductChange.as_view()),
    path('delete', Delete.as_view()),
    path("admin/get", GetProductPublish.as_view()),
    path('return', ReturnApi.as_view()),
    path('return/pickup', RetrunPickup.as_view()),
    path('returned', ReturnProduct.as_view()),
    path("inStock", productInStock.as_view())
]
```

3. *views.py*

```
class Register(APIView):
    permission_classes = [permissions.AllowAny,]
    def post(self, request):
        s = RegisterSerializer(data=request.data)
        if s.is_valid():
            print('register: ', s.validated_data['phone'], s.validated_data['code'])
            phone = s.validated_data['phone']
```

```
if phone[0] != "+":
            phone = "+" + phone
        u = PhoneOTP.objects.get(phone=phone)
        if u.otp == str(s.validated_data['code']):
            nickname = u.nickname
            if User.objects.filter(phone=phone).exists():
                us = User.objects.get(phone=phone)
                uid = us.pk
                us.nickname = nickname
            else:
                us = User.objects.create(phone=phone, nickname=nickname)
                uid = us.pk
            if Token.objects.filter(user=us).exists():
                token = Token.objects.get(user=us)
            else:
                token = Token.objects.create(user=us)
            return Response({'key': token.key, 'uid': uid, 'status': 'ok', 'nickname':
us.nickname})
        else:
            return Response({'status': 'otp error'})
    else:
        return Response(s.errors)

class getProduct(AutoPrefetchViewSetMixin, viewsets.ModelViewSet):
    permission_classes = [permissions.AllowAny,]
    queryset = Product.objects.filter(is_publish=True)
    serializer_class = getProductSerializer
    filter_backends = [DjangoFilterBackend, filters.SearchFilter]
    search_fields = ('title', 'about')
    filter_fields = ('category','subcategory', 'subcategory2', "price_14", "price_30")
    def get_queryset(self):
        minheight = self.request.GET.get('minprice')
        maxheight = self.request.GET.get('maxprice')
        if(minheight and maxheight):
            return self.queryset.filter(price_14__gte=minheight,
price_14__lte=maxheight)
```

```
 return self.queryset

class BasketView(AutoPrefetchViewSetMixin, APIView):
    permission_classes = (permissions.IsAuthenticated,)
    def get(self, request):
        queryset = request.user.basket.all()
        s = getProductSerializer(queryset, many=True, context={'request': request})
        return Response(s.data)
    def post(self, request):
        s = productIdSer(data=request.data)
        if s.is_valid():
            product = Product.objects.get(id=s.validated_data['product'])
            if product in request.user.basket.all():
                request.user.basket.remove(product)
            else:
                request.user.basket.add(product)
            return Response({'status': 'ok'})
        else:
            return Response(s.errors)
```

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
СӘТБАЕВ УНИВЕРСИТЕТІ

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

Қуаныш Ақниет

Дипломдық жоба

Тақырыбы: Жеке тұлғалардың қызметтерін ұсынуға арналған веб-платформаны әзірлеу

ҒЫЛЫМИ ЖЕТЕКШІНІҢ ПІКІРІ

Диплом жобасын жасаушы Қуаныш Ақниет алдына жеке тұлғалардың қызметтерін ұсынуға арналған веб-қосымша жасау тапсырмасы қойылған. Дипломдық жоба тақырыбы жалға алу немесе жалға беру процестерін автоматтандыру, қолданушылар кез-келген затты оңай әрі тез жалға ала алу, сондай-ақ кез-келген өзінің затын жалға қоя алатын кешенді веб-қосымшасын құру болып табылады.

Жұмыс барысында қойылған тапсырмалар: дерекқор құру, бэк бөлігін дайындау, админ интерфейсін жасау болып табылады. Менің пікірім бойынша, диплом жазушы үміткер алдына қойылған тапсырманы толығымен орындады және ақпараттық жүйелердің заманауи технологияларын жақсылап меңгергендігін көрсетті деп ойлаймын.

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

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

«27»____05____2021ж

Метаданные

Название
диплом3.docx

Автор
Куаныш Акниет

Научный руководитель
Даулет Байымбетов

Подразделение
ИКиИТ

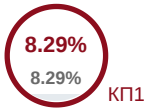
Список возможных попыток манипуляций с текстом

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	1
Интервалы	A→	0
Микропробелы	·	0
Белые знаки	ß	0
Парафразы (SmartMarks)	a	38

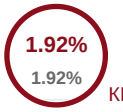
Объем найденных подобий

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



КП1

КП2



КЦ

25
Длина фразы для коэффициента подобия 2

4885
Количество слов

40077
Количество символов

Подобия по списку источников

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

10 самых длинных фраз

Цвет текста

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ И АДРЕС ИСТОЧНИКА URL (НАЗВАНИЕ БАЗЫ)	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	https://github.com/django-import-export/django-import-export/issues/1234	32	0.66 %
2	**Сейдалы Гульзира** Сейдалы Гульзира **5/6/2019** Satbayev University (ИКиИТ)	20	0.41 %
3	https://stackoverflow.com/questions/52469973/django-how-to-override-unique-together-error-message-on-admin-side	17	0.35 %
4	https://github.com/django-import-export/django-import-export/issues/1234	13	0.27 %
5	https://pastebin.com/zaPUeLjc	13	0.27 %

6	https://laptrinhx.com/how-to-populate-your-database-from-an-external-api-in-django-3761339001/	12	0.25 %
7	**Сұлулық салонына жазылуды ұйымдастыратын “Ве.Сұлу” веб қосымшасын құру** Қожантаева Бақыт **5/5/2018** Satbayev University (ИКиИТ)	11	0.23 %
8	https://www.djangoproject.com/community/q-and-a/?page=413	11	0.23 %
9	https://www.djangoproject.com/community/q-and-a/?page=413	10	0.20 %
10	https://stackoverflow.com/questions/52469973/django-how-to-override-unique-together-error-message-on-admin-side	10	0.20 %

из базы данных RefBooks (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из домашней базы данных (1.54 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	**Сейдалы Гульзира** Сейдалы Гульзира **5/6/2019** Satbayev University (ИКиИТ)	44 (4)	0.90 %
2	**Сұлулық салонына жазылуды ұйымдастыратын “Ве.Сұлу” веб қосымшасын құру** Қожантаева Бақыт **5/5/2018** Satbayev University (ИКиИТ)	31 (4)	0.63 %

из программы обмена базами данных (0.14 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	**YFCNU/2019m/iftc/iftc_2019_213.pdf** YFCNU **10/28/2019** Yuriy Fedkovych Chernivtsi National University(CNU) (Deanery)	7 (1)	0.14 %

из интернета (6.61 %)

ПОРЯДКОВЫЙ НОМЕР	ИСТОЧНИК URL	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	https://www.djangoproject.com/community/q-and-a/?page=413	107 (14)	2.19 %
2	https://github.com/django-import-export/django-import-export/issues/1234	65 (5)	1.33 %
3	https://stackoverflow.com/questions/52469973/django-how-to-override-unique-together-error-message-on-admin-side	39 (4)	0.80 %
4	https://pastebin.com/zaPUeLjc	36 (5)	0.74 %
5	https://laptrinhx.com/how-to-populate-your-database-from-an-external-api-in-django-3761339001/	31 (3)	0.63 %
6	https://stackoverflow.com/questions/60159793/django-filter-and-django-tables2-filter-with-modelchoice-values-not-filtering	30 (5)	0.61 %
7	https://forum.djangoproject.com/t/dynamic-filtering/7705	15 (2)	0.31 %

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: Қуаныш Ақниет Жаңабекұлы
Название: Жеке тұлғалардың қызметтерін ұсынуға арналған веб-платформаны әзірлеу
Координатор: Еркежан Сейтбекова

Коэффициент подобия 1: 8.29%
Коэффициент подобия 2: 0.66%

Замена букв: 1
Интервалы: 0
Микропробелы: 0
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:
……
……

27.05.2021г …….

Дата Подпись Научного руководителя

Форматы	Зонасы	Поз-сы.	Белгілеулер	Аталулары	Саны	Қорытынды

						Бет
					Дипломдық жоба	
Өлш	Бет	Құжат №	қолы	күні		39