ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Ақпараттық және телекоммуникациялық технологиялар институты
Программалық инженерия кафедрасы

Токенова Улар Салимгерыйқызы

Тақырыбы: Дәріханалардың мобильді қосымшасы-агрегаторының Back-end бөлігін әзірлеу

Дипломдық жобаға
ТҮСІНІКТЕМЕЛІК ЖАЗБА

5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыздандыру» мамандығы

Алматы 2021

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Ақпараттық және телекоммуникациялық технологиялар институты

Программалық инженерия кафедрасы

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

Дипломдық жобаға
ТҮСІНІКТЕМЕЛІК ЖАЗБА

Тақырыбы: Дәріханалардың мобильді қосымшасы-агрегаторының Back-end бөлігін әзірлеу

5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыздандыру» мамандығы

Орындаған: Токенова Улар Салимгерыйқызы

Ғылыми жетекші, техн. ғыл.
магистрі, лектор
__ _ Д.А. Баймбетов
«_04_» ____маусым____ 2021ж.

Алматы 2021

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Ақпараттық және телекоммуникациялық технологиялар институты

Программалық инженерия кафедрасы

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

Дипломдық жоба орындауға
ТАПСЫРМА

Білім алушыға Токенова Улар Салимгерыйқызы
Тақырыбы: «Дәріханалардың мобильді қосымшасы - агрегаторының Back-end бөлігін әзірлеу»
Университет ректоры бұйрығының №_2131-б_«_24_» _11__2021 ж. Шешімімен бекітілген.
Орындалған жобаның өткізу мерзімі «_08_» _маусым__ 2021 ж.
Дипломдық жобаның бастапқы мәліметтері: Жобаның төлқұжаты, технология бойынша техникалық құжаттама, техникалық тапсырма, жоба диаграммалары түрінде ақпаратты жинау, деректер қорына сақтау, тестілеу, тексеруге арналған программалық қамтамаларды жасау жүргізілген.

Есеп – түсініктеме жазбаның талқылауға берілген сұрақтардың тізімі:
а) тақырып бойынша талдау және есептің қойылымын;

б) жобаны жобалау және пәндік сала бойынша талдау;

в) пайдаланушы интерфейсін жобалау және дамыту;

г) бағдарламаны құру, кітапханаларды қосу, деректерді қосу және тестілеу

Графикалық материалдар тізімі (міндетті суреттердің нақты көрсетілуімен):
презентацияның 16 слайдпен берілген құжат түрінде ұсынылған.

Ұсынылған негізгі әдебиеттер: 8 пайдаланылған әдебиеттер тізімінен

Дипломдық жобаны орындау

КЕСТЕСІ

Бөлімдердің атаулары, зерттелген мәселелердің тізімі	**Ғылыми жетекшіге және кеңесшілерге ұсыну мерзімі**	**Ескерту**
1. Диплом жұмысының жоспар құрылымын құру	19.01.2021	жоқ
2. Тапсырма қойылымы және бағдарламау ортасын таңдау	05.02.2021	жоқ
3. Зерттеу тақырыбы бойынша дипломдық жұмысты жобалау: деректер базасын құру.	10.02.2021	жоқ
4. Дипломның екінші бөлімі – жобалау сызбаларын дайындау.	05.03.2021	жоқ
5. Жобаның мобильді-қосымшасын тестілеуден өткізу.	04.04.2021	жоқ
6. Дипломдық жобаға түсіндірме жазба жазуды аяқтау	06.05.2021	жоқ

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

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

Ғылыми жетекші _____ Д.А. Баймбетов

Тапсырманы орындауға қабылдап алған студент _____ У.С.Токенова

Күні «_06_» _маусым_ 2021ж.

АҢДАТПА

Бұл дипломдық жоба онлайн дәріхана мобильді – агрегатор қосымшасы құрылған. Онлайн дәріхана қажетті дәрі-дәрмектерді ұялы телефон, ақпараттық жүйелер және мобильді қосымшаны қолдану арқылы табуға мүмкіндік береді. Сонымен қатар мобильді қосымша сізге дәріхананың мекен-жайы туралы толық ақпарат береді, орналасқан жері мен телефон нөмірі. Белгілі бір дәріханада қазіргі кезде қанша дәрі-дәрмек қалғаны туралы ақпарат ала аласыз. Бұл мүмкіндітер тек қарапайымқолданушыларда. Бұл агрегатор қосымшасы дәріхана иелеріне арналған. Қоймадағы дәрі-дәрмек саның қадағалап, жарамдылық мерзімін қадағалап отыруға арналған, яғни өз тауарларына мониторинг жасай алады.

АННОТАЦИЯ

Дипломный проект представляет собой мобильное приложение – агрегатор интернет-аптек. Интернет-аптека позволяет найти необходимые лекарства с помощью мобильного телефона, информационных систем и мобильного приложения. В мобильном приложении вы также найдете подробную информацию об адресе аптеки, ее местонахождении и номере телефона. Вы можете получить информацию о том, сколько лекарств сейчас осталось в той или иной аптеке. Эти функции предназначены только для обычных пользователей. Приложение-агрегатор предназначено для владельцев аптек. Он предназначен для отслеживания количества и сроков хранения лекарственных средств на складе, т.е. может контролировать свою продукцию.

ANNOTATION

The diploma project is a mobile application-aggregator of online pharmacies. The online pharmacy makes it possible to find the necessary medicines using a mobile phone, information systems and a mobile application. In the mobile application, you will also find detailed information about the pharmacy's address, location and phone number. You can get information about how many drugs are currently left in a particular pharmacy. These features are for general users only. The aggregator application is intended for pharmacy owners. It is designed to track the quantity and shelf life of medicines in the warehouse, i.e. totrack their products.

МАЗМҰНЫ

КІРІСПЕ

Қазіргі кезде ақпараттық технологиялардың қарқынды даму кезеңі. Интернетті кез келген уақыт аралығында пайдалану арқылы кезек күту немесе көлік кептелестерінде тұру қиындықтар туғызады, осындай кездерде мобильді қосымашалар өте ыңғайлы ресурс болып табылады. Мысалы онлайн дәріхана агрегаторы арқылы өзіңіз қалаған дәрі-дәрмектің көрсеткіштерін мен қарсы көрсеткіштерін оқып, бағасы, жарамдылық мерзіміжәне саны туралы толық ақпарат ала аласыз. Сонымен қатар, дәріхана иелері өзінің қоймасындағы дәрі-дәрмек және тауарларына мониторинг жасай алады, яғни учет жүргізе алады, мәселен қоймада қанша тауар саны қалғаны және қай тауардың жарамдылық мерзімі аяқталғанын push-хабарландыру функциясы арқылы біле алады.

Осылайша, онлайн дәріхана агрегаторы сізге осындай артықшылықтар ұсынады, препараттың қол жетімділігі туралы нақты ақпарат, препаратты қолдану туралы нақты ақпарат, оның құрамы, өндірушісі және т.б., бағасы туралы нақты ақпарат, дәріхана жайлы толық ақпарат, дәріханаларда болып жатқан жеңілдіктер мен акциялар жайлы push-хабарламалар.

Онлайн дәріхана агрегаторының басты мақсаттарының бірі қолданушыдың уақытын үнемдеу және дәріхана иелеріне тиімді функционал ұсыну. Өзіміз көріп тұрғандай көп қиындықтардың шешімін мобильді қосымшалар арқылы шешуге болады.

1 Пәндік аймаққа талдау және есептің қойылымы

1.1 Пәндік аймақтың сипаттамасы, мобильдік бағдарлама құрудың басты мақсаты және оның артықшылықтары

Осы уақытта дәріханаларға арналған мобильді қосымшалар өте кең таралған. Дәріханаларға арналған мобильді қосымша – бұл қызметтерді ілгерілетуге, компания мен клиент арасындағы байланысты жеңілдетуге, көптеген процестерді автоматтандыруға және сатылымды арттыруға мүмкіндік беретін қуатты құрал. Дәріхана қосымшасы жекелендірудің арқасында сұранысқа ие және тұтынушыларға тиімді. Компания үшін бұл орташа чекті көбейту, адал клиенттер жинау және жаңа клиенттерді тарту мүмкіндігі. Бұл сонымен қатар көптеген ресурстарды және адамдар үшін уақытты үнемдейді. Мұндай қосымшалардың тіпті бірнеше санаты бар, толық талдау жасайық.

Жұмыстың мақсаты – белгілі бір дәріхананың тұтынушылары үшін уақытты үнемдеу, яғни қосымша арқылы қандай дәрі-дәрмек бар екенін бақылау. Атап айтқанда, бұл онлайн қосымша агрегатор дәріхана иелеріне арналған, дәлірек айтсақ, қосымшаның көмегімен сіз қоймадағы дәрі-дәрмектің жарамдылық мерзімін бақылай алу. Push-ескерту арқылы дәріхана иелері қойманы бақылаусыз оңай басқару.

Жұмыстың өзектілігі – бұл ақпараттық жүйе арқылы қолданушылар өздеріне қажетті дәрі-дәрмектерді іздеп таба алады. Сонымен қатар, қолданушыға жақын орналасқан дәріхананы тауып, мәліметтеріне қол жеткізе алады. Осы мәселелерді шешу үшін «Онлайн дәріхана агрегатор» ақпараттық жүйесі жасалынды.

Дәріханалық қосымшалардың бірнеше санаттарының бірі – бұл онлайн дәріханалық агрегатор қосымшасы. Дәріхана агрегаторының мобильді қосымшасы – бұл ақпараттық қызмет, дәлірек айтсақ, әр дәріханадан белгілі бір дәріханадағы дәрінің бағасы мен қол жетімділігі туралы ақпарат жинау. Дәлірек айтқанда, дәрі-дәрмектерге толық шолу, қарсы көрсеткіштер, қолдану әдісі, жарамдылық мерзімі, дәрі-дәрмектің болуы, осы дәріхана туралы толық ақпарат және т.б. Қарапайым пайдаланушылар үшін белгілі бір дәріханадан акциялар мен жеңілдіктер туралы хабарлама алу мүмкіндігі бар. Бұл қосымша ақпараттық қызмет ретінде ғана емес, оның артықшылығы бар, дәлірек айтсақ, дәріхана иелеріне көбірек қызмет етеді. Олар өз тауарларының қоймасын қадағалай алады, яғни қоймада қанша тауар қалады және итермелеу арқылы хабарландыру арқылы есірткінің жарамдылық мерзімін бақылауға болады. Дәріхана иесі өзінің дәріханасының есебін жүргізе алады.

Бұл «Онлайн дәріхана агрегатор ақпараттық жүйесінің» артықшылықтары:

– ыңғайлы интерфейс;

– дәріханаға барар алдында нақтылау ұсыныс жүйесі (рекомендательные системы);

– дәрі-дәрмектерді жылдам іздеу;

– дәрі-дәрмектерге қолданушылардың пікір блогы;

– қолдандабада дәрілердің сатылымдағы саны көрсетіліп отырады;

– қолданбада дәріхана иелерінің өз рұқсат құқығы бар, яғни өз кабинетінде қандай тауардың саны азайып бара жатқандығы жайлы ақпарат ала алады;

– қолданбаданың push-хабарламалары арқылы жеңілдіктер жайлы біліп отыруға мүмкіндік бар;

– дәріхана иелерінің учет жүргізу мүмкіндігі бар.

1.2 Мобильдік бағдарлама жүйесіндегі деректер қорына қойылатын негізгі талаптар

Мобильді агрегатор қосымшасының Back-end бөлігін әзірлеуге қойылатын талаптар тізімі. Мобильді агрегатор қосымшасының Back-end бөлігін әзірлеу кезінде Django фреймворкі қолданылды. Қойылған талаптар:

– мәліметтер базасына тұрақты байланыс болуы керек;

– мәліметтер қорының байланыстарын үнемі басқару;

– кодтау;

– мәліметтер базасын басқару жүйелерімен дұрыс конфигурациялау;

– мәліметтердің тұтастығы мен тәуелсіздігін сақтай отырып,

– мәліметтер қорын құру;

– мәліметтер базасына қосылу;

– кестелер құру;

– модельдер құру.

1.3 Анықтамалар және терминдер

Жобалау кезінде қолданылған анықтамалар және терминдер тізімі 1.1 кестеде көрсетілген.

1.1-кесте – Анықтамалар және терминдер

Django	веб-қосымшаларға арналған фреймворк
Digital Ocean Ubuntu	инфрақұрылым жеткізушісі

1.1-кестенің жалғасы

PostgreSQL	ақысыз объектілік-реляциялық мәліметтер қорын басқару жүйесі
View	Представление
JSON	JavaScript Object Notation
HTML	HyperText Markup Language

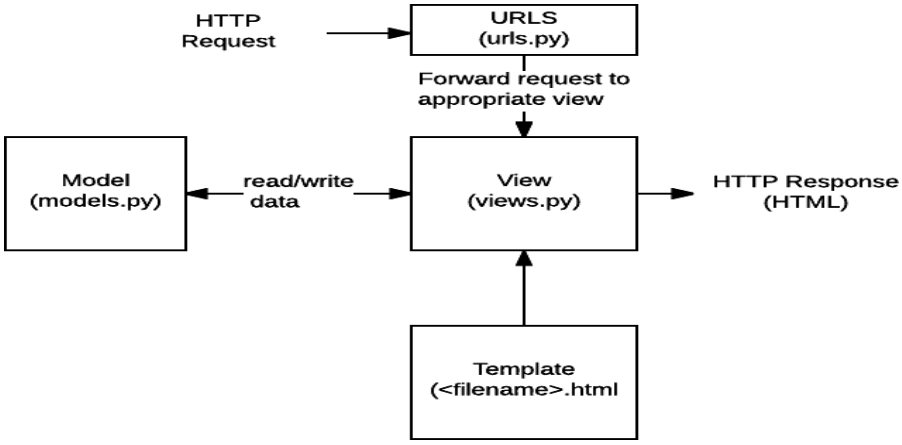
2 Техникалық зерттеу бөлімі

2.1 Қолданылған құралдар мен технологияларға шолу

Django – бұл қауіпсіз және қолдай алатын веб-сайттарды жылдам құруға болатын жоғары деңгейлі Python веб-құрылымы. Тәжірибелі әзірлеушілер жасаған Django веб-қосымшаларды жазуға көңіл бөлу үшін веб-сайттарды құрудағы қиындықтардың көп бөлігін алады. Бұл ақысыз және ашық кодта жүктелген, өсіп келе жатқан және белсенді қоғамдастық, керемет құжаттама және көптеген ақысыз және ақылы қолдау нұсқалары. [1]

Django 2003 және 2005 жылдар аралығында газет веб-сайттарын құратын және жүргізетін топпен жасалды. Бірнеше веб-сайттарды салғаннан кейін команда көптеген жалпы кодтар мен дизайн үлгілерін қайта қолдана бастады. Онда көптеген веб-әзірлемелерге арналған компоненттер жиынтығы және оларды пайдаланудың бір (немесе екі) тәсілдері ұсынылған. Алайда,бұл Django архитектурасы сіз әдетте бірнеше түрлі нұсқаларды таңдай аласыз немесе қажет болған жағдайда мүлдем жаңа опцияларға қолдау қосасыз дегенді білдіреді.

Дәстүрлі жаңалықтар сайтында веб-бағдарлама веб-шолғыштан (немесе басқа клиенттен) HTTP сұрауларын күтеді. Сұраныс түскен кезде, бағдарлама URL-мекен-жайы мен мүмкін POST немесе GET сұраныстарындағы мәліметтер негізінде қажет нәрсені өңдейді. Оның қажеттіліктеріне байланысты ол содан кейін мәліметтер базасынан ақпараттыоқи немесе жаза алады немесе сұранысты орындау үшін қажетті басқа әрекеттерді орындай алады. Содан кейін қосымша веб-шолғышқа жауап қайтарады, көбінесе алынған деректерді HTML үлгісіне енгізу арқылы браузерде көрсету үшін HTML парағын динамикалық түрде жасайды. Django веб-қосымшалары әдетте осы қадамдардың әрқайсысын өңдейтін кодты бөлек файлдарға топтайды:



2.1-сурет - Django-ғы файлдар құрылымы

URL-мекен-жайлар: Әрбір URL мекен-жайынан сұраныстарды бір функциямен өңдеу мүмкін болғанымен, әр ресурс үшін бөлек функция жазу әлдеқайда ыңғайлы. URL маршрутизаторы HTTP сұраныстарын URL мекен-жайы негізінде тиісті көрініске қайта бағыттау үшін қолданылады. Сонымен қатар, URL маршрутизаторы белгілі бір үлгі бойынша URL-ден деректерді шығарып, оны тиісті қарау функциясына аргумент ретінде бере алады.[2]

View: бұл HTTP сұраныстарын қабылдап, жауаптарын қайтаратын сұраныстарды өңдеу функциясы. Көрермен талаптарды қанағаттандыру үшін қажет деректерге қол жеткізеді және модельдер көмегімен шаблондарға жауаптар береді.

Models: моделдер – бұл қосымшаның деректер құрылымын анықтайтын және мәліметтер базасын редакциялау (қосу, өзгерту, жою) және сұрау тетіктерін қамтамасыз ететін Python объектілері.

Templates: Template – бұл нақты мазмұнды көрсету үшін қолданылатын іздеу өрістері бар беттің құрылымын немесе орналасуын (мысалы, HTML парағы) анықтайтын мәтіндік файл. Көрініс HTML шаблондарын пайдаланып HTML беттерін динамикалық түрде құра алады және оларды шаблон деректерімен толықтыра алады. Үлгіні кез-келген типтегі файлдың құрылымын анықтау үшін қолдануға болады, міндетті түрде HTML емес.

DigitalOcean, Inc. – бұл Нью-Йоркте және бүкіл әлемдегі деректер орталықтарында орналасқан бұлтты инфрақұрылымның (облачные инфраструктуры) американдық провайдері. Тапсырыс беруші Нью-Йорк, Сан-Франциско, Амстердам, Франкфурт, Лондон, Торонто, Сингапур және Бангалордағы деректер орталықтарының бірін таңдай алады, осылайша барлық ірі халықаралық трафик алмасу орталықтарын қамтиды. [6]

3 Онлайн дәріхана бағдарлама агрегаторының деректер қорының құрылымын талдау

3.1 Модельдер мен деректер қорын сипаттау

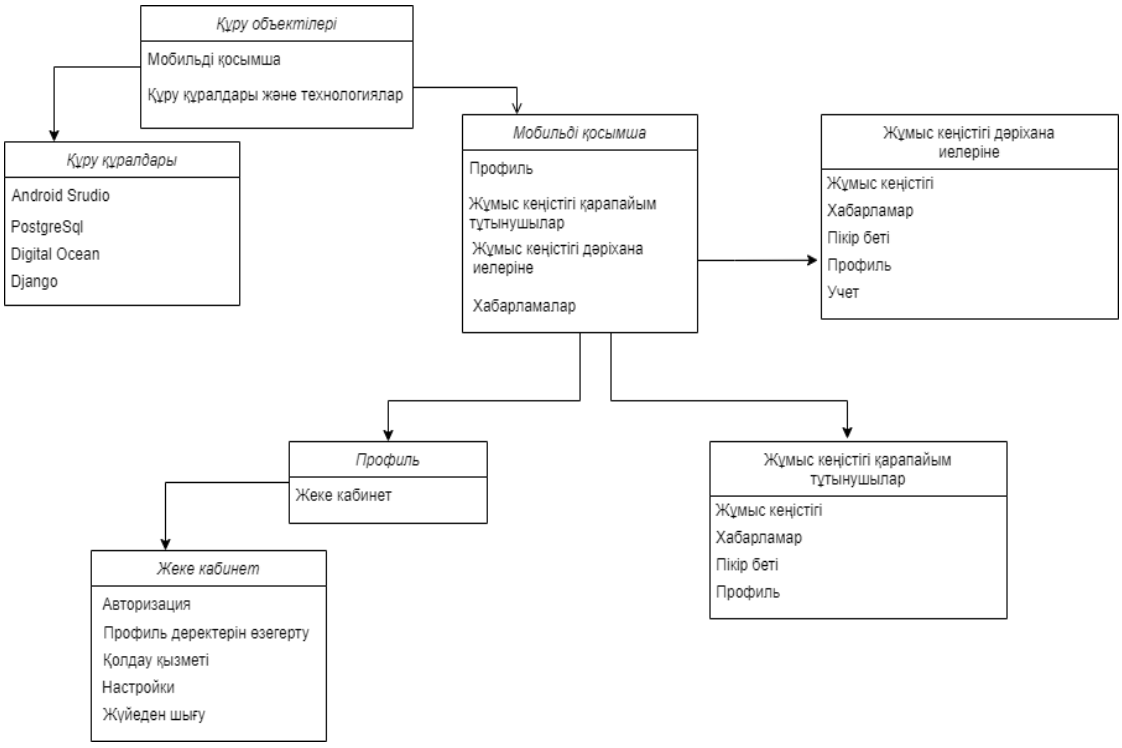
Мәліметтер қорының концептуалдық моделі – бұл қабылданған нотада салынған және объектілер мен олардың сипаттамалары арасындағы байланысты егжей-тегжейлі көрсететін визуалды диаграмманың бір түрі. Концептуалды модель базаны әрі қарай жобалау және оны реляциялық мәліметтер базасына аудару үшін жасалады. Тұжырымдамалық модель деректер объектілері мен олардың сипаттамалары арасындағы байланысты визуалды түрде ыңғайлы түрде тіркейді.

Концептуалдық мәліметтер базасы үшін мәліметтер базасын бағдарламалаудың бірізділігі үшін келесі ұғымдар енгізілді:

– Нысан(объект) немесе мән (сущность). Бұл пайдаланушы (сатып алушы) байқағысы келетін нақты нәрсе немесе объект (адамдар үшін). Мысалы, Иванов Иван Иванович;

– Атрибут – бұл оның мәніне сәйкес келетін объектінің сипаттамасы. Мысалға. Біз өзімізге сұрақ қоямыз: Иванов Иван Иванович туралы қандай ақпаратты сақтау керек? Бұл сұраққа жауаптар объектінің атрибуттары болады Иванов Иван Иванович;

– Концептуалды мәліметтер қорын жобалау кезіндегі үшінші түсінік-объектілер арасындағы қатынастар немесе жай қатынастар. [8]



3.1-сурет – Жүйенің концептуалды моделі

Деректердің логикалық моделін құру.

Мәліметтердің логикалық моделі – бұл мәліметтер құрылымдарының, олардың атрибуттары мен байланыстарының визуалды графикалық көрінісі. Логикалық модель деректерді бизнес пайдаланушылары оңай түсінетін етіп ұсынады. Логикалық модельді әзірлеу кезінде платформаға және іске асыруға тілдік талаптар болмауы керек немесе кейінірек деректер қалай пайдаланылатын болады.

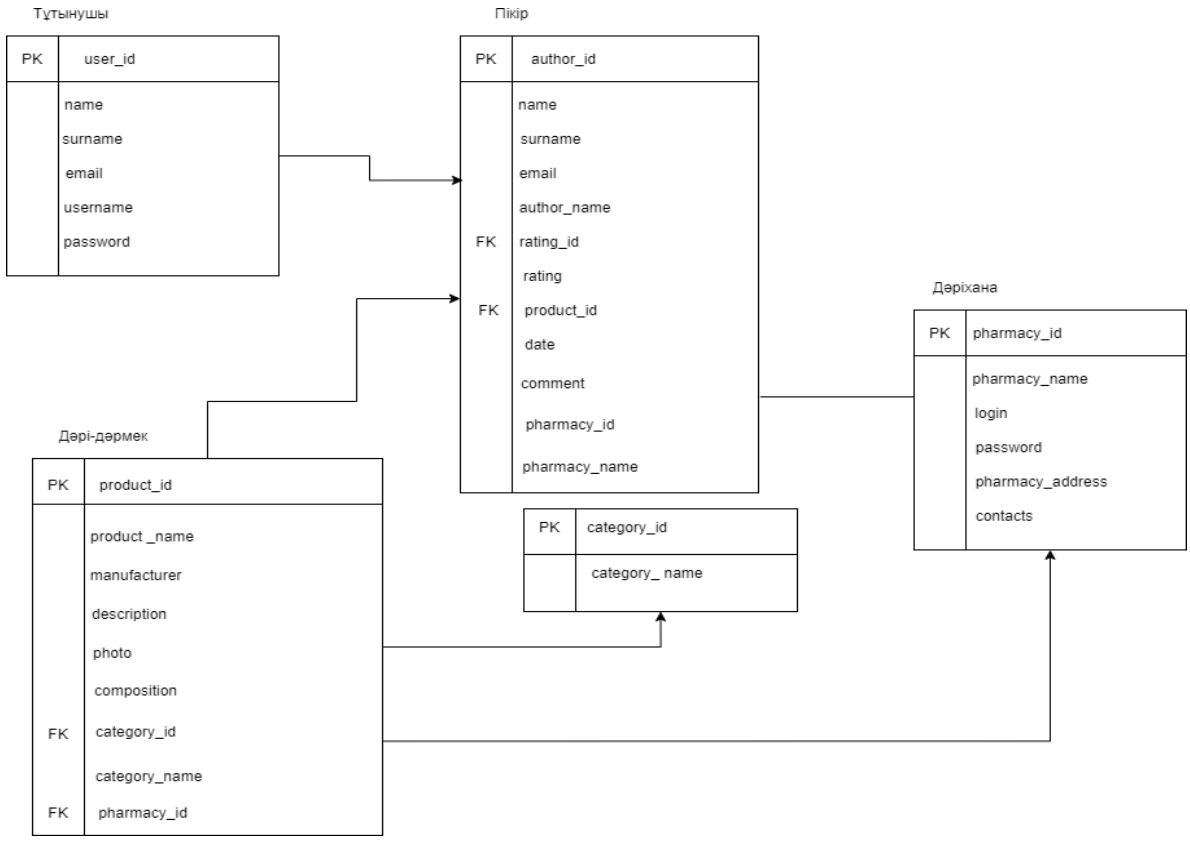
Даму деректердің логикалық моделін қалыптастыру үшін мәліметтерге қажеттіліктер мен талдау нәтижелерін қолданады. Логикалық модель үшінші қалыпты формаға ауыстырылады және технологиялық модель көмегімен тексеріледі.

Логикалық модельдің негізгі компоненттері:

– нысандар;

– нысан атрибуттары;

– субъектілер арасындағы қатынастар. [7]

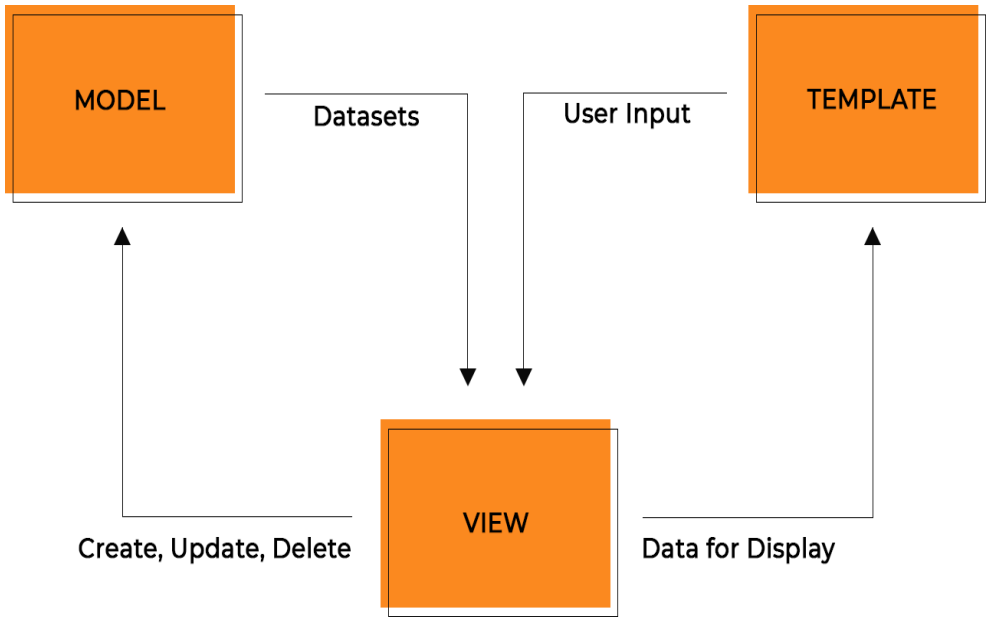


3.2-сурет – Деректердің логикалық моделі

4 Онлайн дәріхана бағдарламасының жоба бөлімі

4.1 Жобаның өзара әрекеттесу жүйесінің архитектурасын талдау

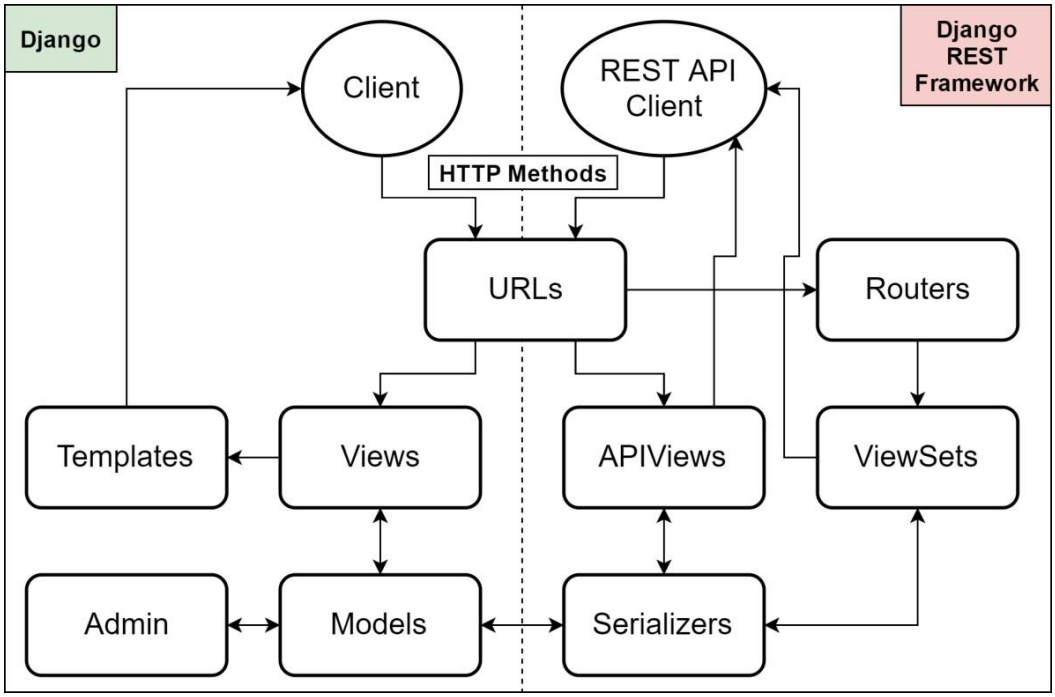
Django архитектурасы модель-көрініс-контроллер (MVC) моделіне ұқсас. Классикалық MVC модель контроллері Django көрінісі деп атайды, ал көрініс презентациясының логикасы Django-да модель деңгейінде жүзеге асырылады. Осы себепті Джангоның қабатты архитектурасы көбіне Model-Template View (MTV) деп аталады, архитектурасы 4.1-суретте көрсетілген. [3]



4.1-сурет – Django архитектурасының схемасы

Django дамыту ортасы Python, Django және мәліметтер базасы жүйесін орнатудан және конфигурациялаудан тұрады. Django веб-қосымшамен айналысатындықтан, веб-серверді де орнату қажет екенін айта кеткен жөн.

Ең алдымен, біз қажетті ресурстарды, пакеттерді орнатамыз және даму ортасын орнатамыз. Ортаны орнатқаннан кейін дерекқорды орнатуға кірісеміз, бұл жағдайда дерекқор PostgreSQL-де сақталады. Содан кейін біз серверге байланыс орнаттық (Digital Ocean Ubuntu). Содан кейін біз жобаны дамытуға көшеміз, серверді және мәліметтер базасын құрып, орнатқаннан және қосқаннан кейін біз интерфейсті іске қосамыз.Django әкімшісі, содан кейін әкімші интерфейсі және суперпайдаланушы қосылымдары арқылы біз қолданба модельдерін толтыра аламыз (CRUD операцияларын орындау). Келесі кезекте модельдер жасаймыз, деректермен манипуляция жасаймыз (CRUD), дәлірек айтқанда, көріністі құруға көшеміз, Djangoархитектурасының схемасы 4.2-суретте көрсетілген.

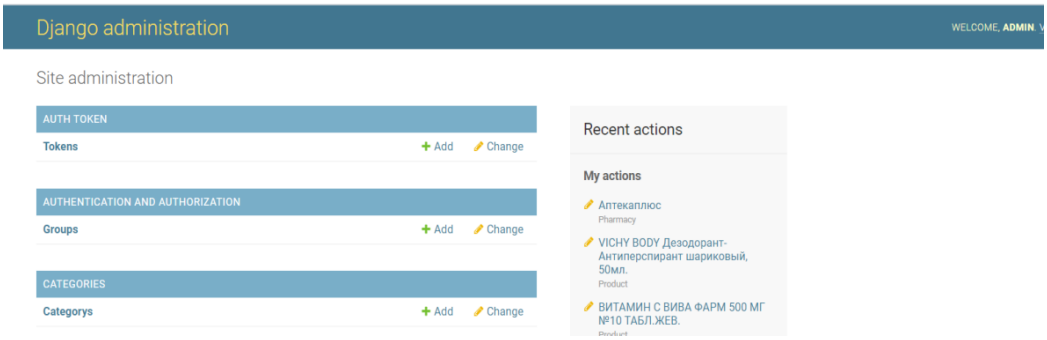


4.2-сурет – Django архитектурасының схемасы

4.2 Мәліметтер базасын жобалау

PostgreSQL дерекқоры деректерді сақтау үшін қолданылады. Қосымшадан келетін деректер үлкен JSON ағашында сақталады. Әрбір қолданушы үшін оның жеке құжаты жинақта жасалады. Құжат өз кезегінде CRUD операциялары кезінде мән қабылдайтын өрістерді қамтиды. [5]

Біз суперпайдаланушының тіркелгі деректерімен жергілікті серверге өтіп (деректермен толтырып) 4.3-суретте көрсетілген, JSON-да жауап беретін API жазуды жалғастырамыз.



4.3-сурет – Локальді сервер

Сервер параметрлерімен барлық манипуляциялардан кейін біз мәліметтер қорының моделін құрамыз, модель дегеніміз кестеден мәліметтерді ұсынатын объект 4.4-суреттер көрсетілген. Модель жасаймыз:

```python
class Product(models.Model):
    name = models.CharField(max_length=150)
    manufacturer = models.ForeignKey(Manufacture, on_delete=models.CASCADE)
    description = models.TextField(null=True, blank=True)
    photo = models.ImageField(upload_to=product_photos_dir, default="default/default.png", null=True, blank=True)
    composition = models.TextField(null=True, blank=True)
    category = models.ForeignKey("categories.Category", on_delete=models.CASCADE, null=True, blank=True)

    def __str__(self):
        return self.name
```

4.4-сурет – Мәліметтер қорының моделі

Біз запрос параметрін қабылдайтын және жауап қайтаратын view (функция) құрамыз. View бойынша біз қайтқымыз келетін сөздік нысанын құрдық және сол сөздікті Django ұсынған JsonResponse класына жібереміз 4.5-суретте көрсетілген.

```python
class createProduct(APIView):
    permission_classes = [permissions.IsAuthenticated,]

    def post(self, request):
        s = CreateProductSer(data=request.data)
        if s.is_valid():
            p = Product.objects.create(
                name = s.validated_data['name'],
                manufacturer = Manufacture.objects.get(id=s.validated_data['manufacturer']),
                description = s.validated_data['description'],
                composition = s.validated_data['composition'],
                category = Category.objects.get(id = s.validated_data['category']),
                photo = s.validated_data['photo']
            )
            c = CountProduct.objects.create(
                product = p,
                pharmacy = request.user.my_pharmacy.all()[0],
                price = s.validated_data['price'],
                count = s.validated_data['count']
            )
            return Response({'status': 'ok'})
        else:
            return Response(s.errors)
```

4.5-сурет – View құру

Келесі қадам, осы View-ға қол жеткізу үшін url/endpoint құру 4.6- суретте көрсетілген.

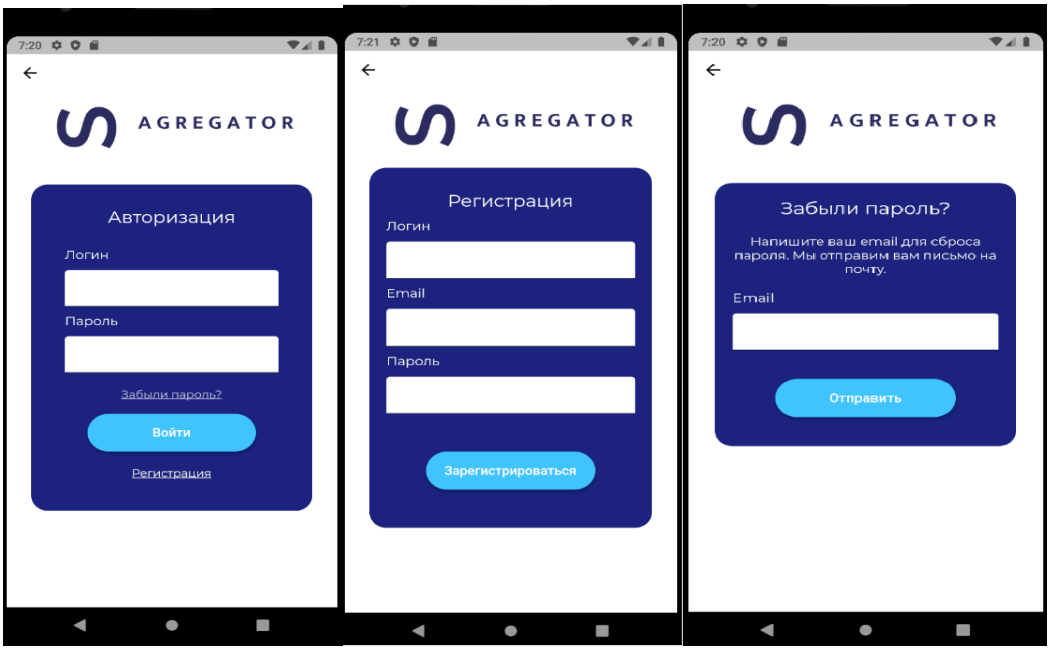
```python
urlpatterns = [
    path('', getProduct.as_view({'get': 'list'})),
    path('create/', createProduct.as_view()),
```

4.6-сурет – View-ға қол жеткізу

4.7-сурет – JSON API

4.3 Мобильді қосымшаның интерфейсіне қысқаша шолу

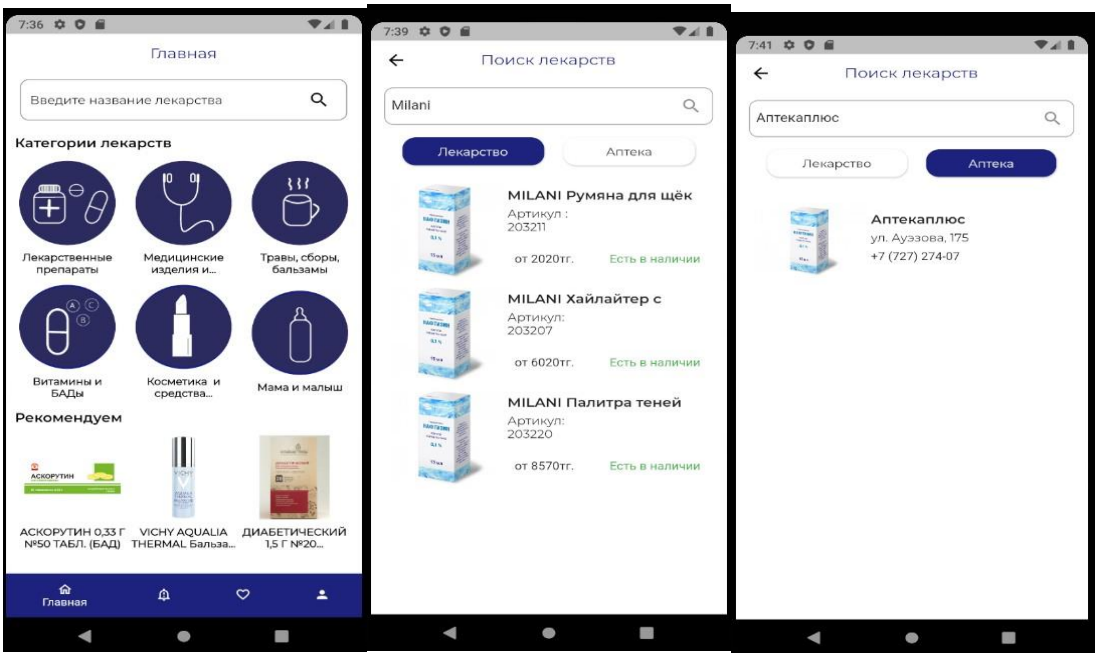
Мобильді қосымшаның алғашқы беті, авторизация, тіркелу және құпия сөз бетін қалыптастыратын бөлімдер 4.8-суретте көрсетілген.



4.8-сурет – Мобильді тіркелу және авторизация беттері

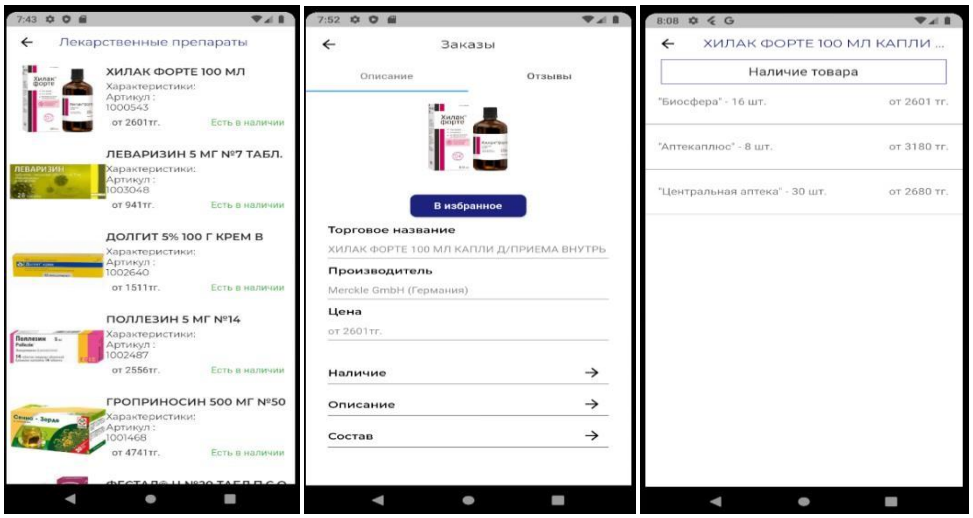
Тіркелу бетінен өткеннен кейін, ең алдымен тауарлар категориясы мен «ұсыныс» беті ашылады, яғни бұл тұтынушының ең жиі қарастыратын

тауарларына байланысты ұқсас тауарлар шығарып береді. Мобильді қосымша - агрегаторының басты мақсаты дәрі-дәрмек пен дәріхана жайлы мәлімет іздеу болып табылады. Яғни сіз дәрінің атауын енгізу арқылы табу функциясы қарастырылған, 4.9-суретте көрсетілген.



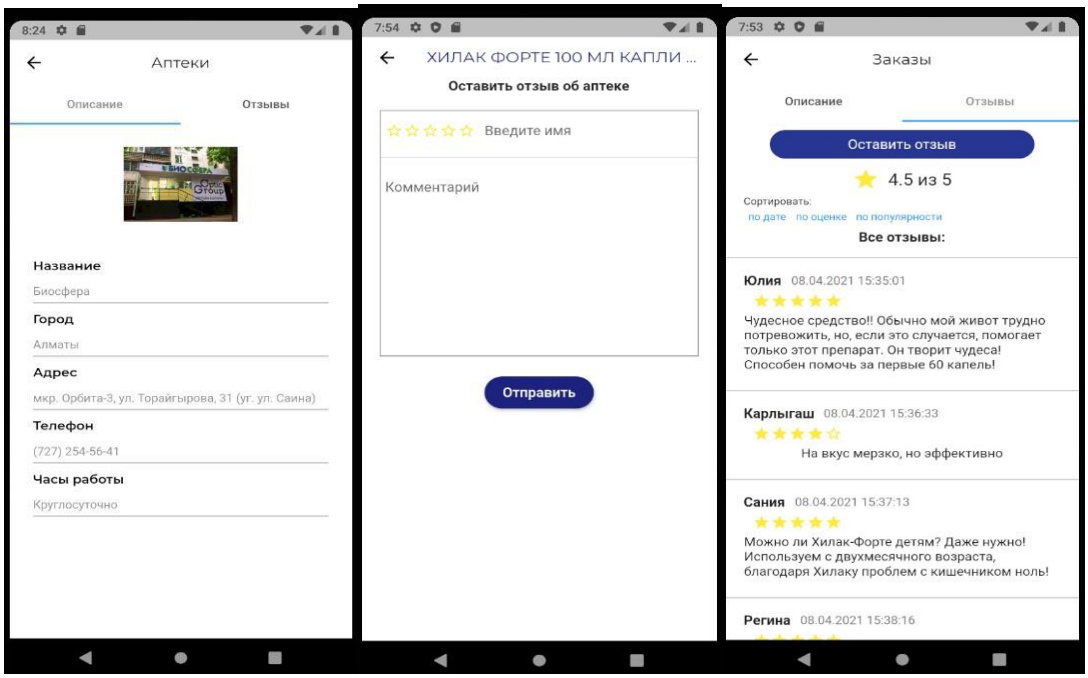
4.9 -сурет – Дәрі-дәрмек іздеу бөлімі

Келесі беттерде тауарлар жайлы толық ақпарат ала аласыз. Қай дәріханада бар сізге қажетті тауадың бар екенін көруге болады. Дәрі-дәрмек жайлы толық ақпарат, яғни оның құрамы, қарсыкөрсеткіштері, бағасы және т.б. ақпарат толықтай алуға болады. Көріп тұрғандарыңыздай іздеген дәрі-дәрмек қай дәріхана және қанша саны қалғандығын көрсетеді, 4.10 - суретте көрсетілген.

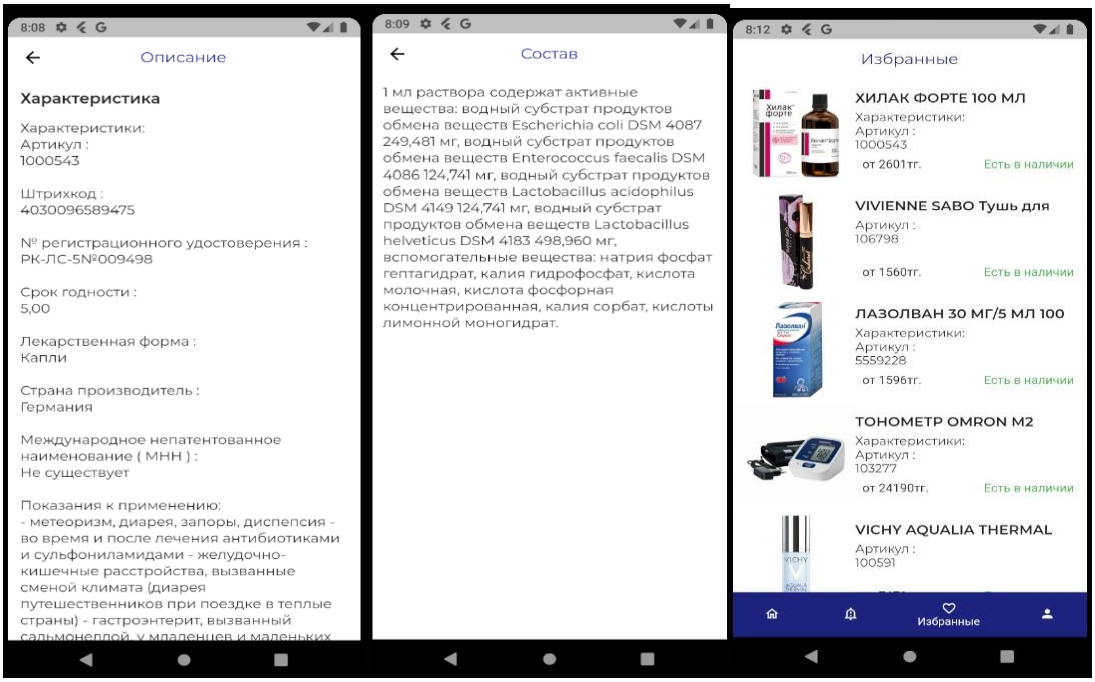


4.10-сурет – Дәрі-дәрмектер тізімі беті

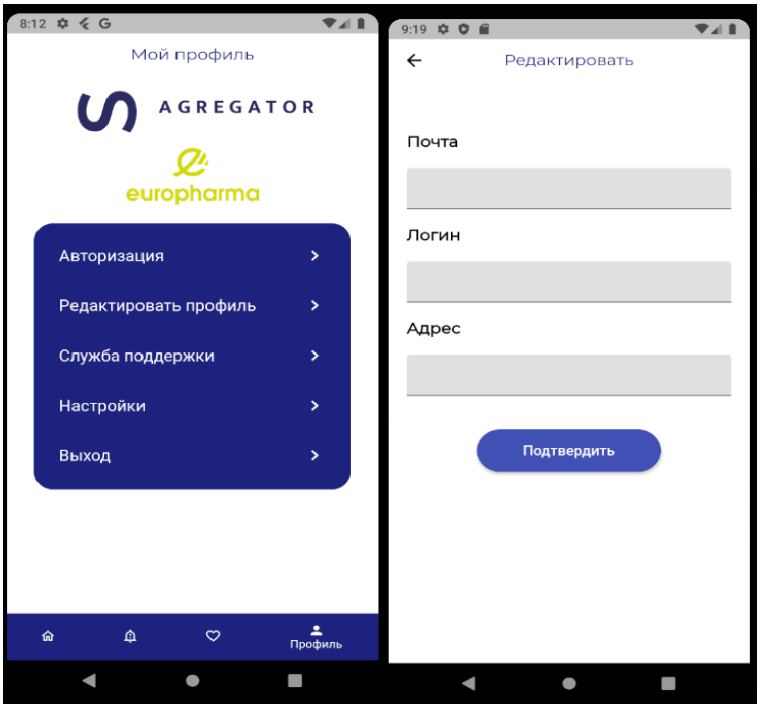
4.11-суретте аптеканың өзіне басқанда сол дәріхана жайлы толық ақпарат ала аласыз, яғни дәріхананың мекен жайы, жұмыс істеу уақыты және т.б. басқа ақпарат ала аласыз. Сонымен қатар дәріхана жайлы өзіңіздің пікіріңізді қалдыра аласыз.



4.11-сурет – Дәріхана және пікір қалдыру беттері

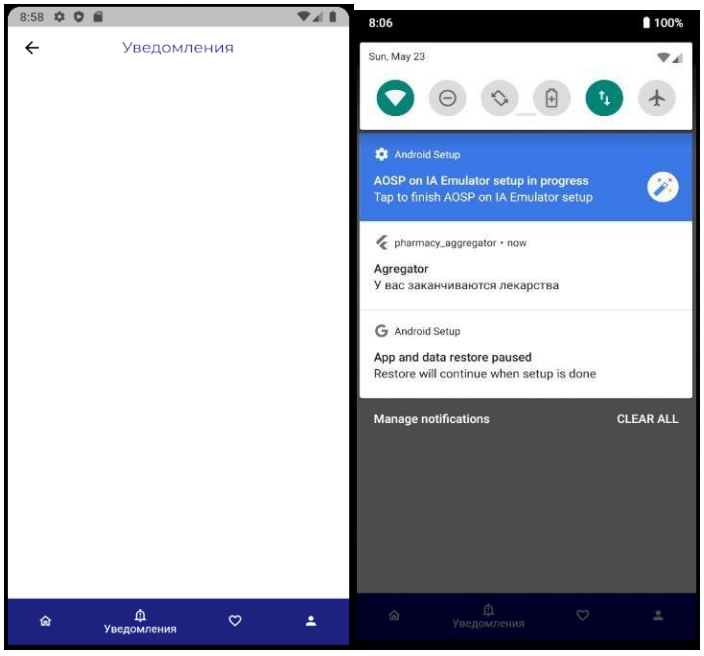


4.12-сурет – Дәрі-дәрмек туралы ақпарат және избранное беті



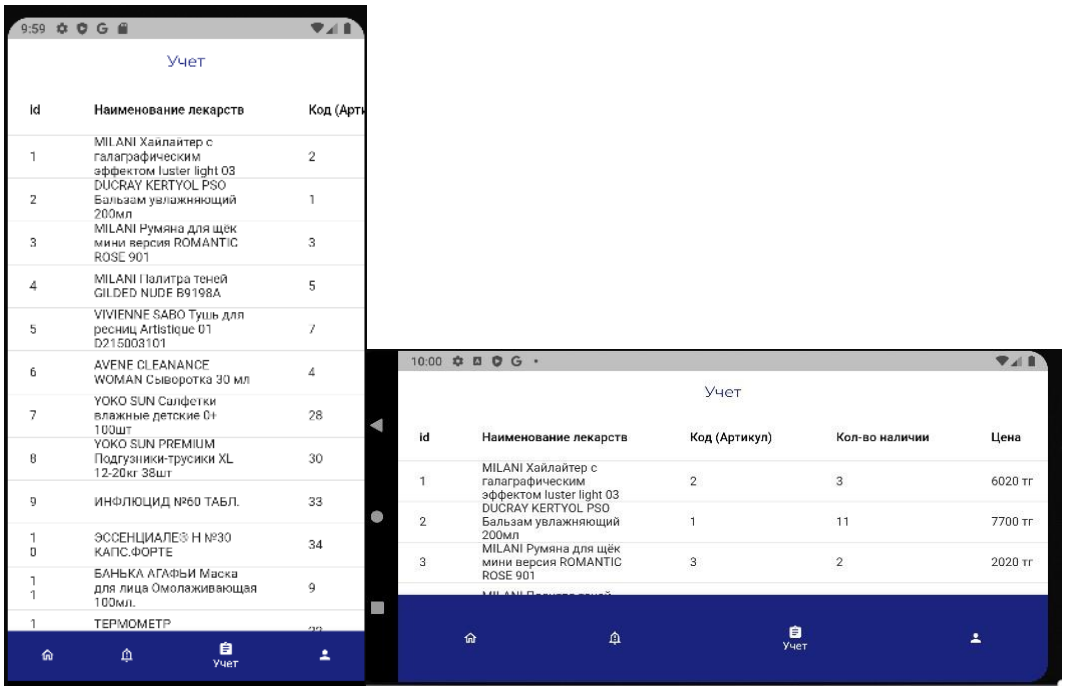
4.13-сурет – Жеке кабинет беті

Push-хабарламандыру бетінде қарапайым тұтынушыларға белгілі бір аптекадағы акция және жеңілдіктер жайлы хабарламалар жіберіліп тұрады. 4.14-суретте дәріхана иелеріне қоймадағы дәрі немесе тауар саны, тауардың жарамдылық мерзімі біткені жайлы хабарламар жіберіліп тұрады.



4.14-сурет – Push-хабарламандыру беті

4.15-суретте дәріхана иелері өз тауарларына учет жүргізе алады. Бұл қарапайым тұтынушыларға көрінбейтін функция. Яғни тауарлар жайлы ақпарат көре алады.



4.15-сурет – Учет жүргізу беті

ҚОРЫТЫНДЫ

Бұл дипломдық жұмыстың түпкі мақсаты – дәріханалық агрегатор қосымшасының Back-end бөлігін жасау, дәлірек айтқанда, дұрыс интеграцияланған мәліметтер базасымен қосымшаны ұсыну. Бұл соңғы өнім ақпараттық қызмет түріндегі қарапайым қолданушыларға да, дәріхана иелеріне де арналған. Дәріхана иелері өз тауарларын қоймада қадағалау мүмкіндігі бойынша бірнеше ерекше артықшылықтарға ие, дәлірек айтсақ, қанша қалды және хабарлама түріндегі дәрілік заттардың жарамдылық мерзімі. Дәріхана иелері үшін маңызды артықшылықтардың бірі олардың тауарларының есебін жүргізу болып табылады. Қарапайым пайдаланушылар үшін бұл қосымша ақпараттық қызмет сияқты, бірақ сонымен бірге адамдардың уақытын үнемдейді және кезектердің пайда болуына жол бермейді.

Интеграция процесінде жүйеге қойылатын кейбір қосымша талаптар нақтыланды.

Пәндік саланы зерттеу барысында мұндай қосымшалардың жетіспейтіндігі анықталды.

Бұл қосымша бағдарламалық жасақтаманы жақсартуға ашық. Болашақта оны қолданыстағы сауда жүйесімен біріктіруге және жаңа функциялармен жабдықтауға болады, мысалы, тауарларды брондау және т.б. Бұл қосымша ретінде тартымдылықты жоғарылатып, дәріханаларға тиімді болар еді.

ПАЙДАЛЫНЫЛҒАН ӘДЕБИТТЕР ТІЗІМІ

1 Django кіріспе // Электрондық нұсқа https://metanit.com/python/django/.

2 Django Python // Электрондық нұсқа https://www.educative.io/blog/what-is-django-python.

3 Django архитектурасы // Электрондық нұсқа https://habr.com/ru/post/31180/.

4 Django архитектурасы // Электрондық нұсқа https://habr.com/ru/company/vivid_money/blog/544856/ .

5 Django жобасының серверлік бөлігі // Электрондық нұсқа https://developer.mozilla.org/ru/docs/Learn/Server-side/Django.

6 Digital Ocean Ubuntu // Электрондық нұсқа https://ru.wikipedia.org/wiki/ DigitalOcean.

7 Деректердің логикалық моделі // Электрондық нұсқа https://poisk-ru.ru/s17872t2.html.

8 Деректердің концептуалдық моделі // Электрондық нұсқа https://webonto.ru/kontseptualnaya-model-bazyi-dannyih/.

А Қосымшасы
(міндетті)

Техникалық тапсырма

А.1.1 Мобильдік қосымшаның back-end бөлігін әзірлеуге арналған техникалық тапсырма

Бұл техникалық тапсырма дәріханаларға арналған мобильдік қосымшаның back-end бөлігін жасауға қатысты. Дәріхана қызметкерлері, фармацевтикалық компания иелері және қарапайым қолданушылар бұл қосымшаны қолдана алады. Қосымша фармацевтикалық және медициналық салада қызмет көрсетуге бағытталған. Қойылатын талаптар:

– деректер қорына тұрақты байланыс болуы керек;

– деректер қорының тұтастығын мен тәуелсіздігін сақтай отырып, мәліметтер базасын құру;

– модельдер құру;

– мәліметтер қорының байланыстарын үнемі басқару;

– жүктеме кезінде төтеп беру;

– сервермен байланысты үнемі қолдау.

Б Қосымшасы

(міндетті)

1. product/models.py

```
from django.db import models
def pharmacy_photos_dir(instanse, filename):
    usrnme = f'{instanse.name}'
    folder_name = f"{usrnme}/{filename}"
    return folder_name

class Pharmacy(models.Model):
    name = models.CharField(max_length=150)
    address = models.CharField(max_length=255)
    working_hours = models.CharField(max_length=255)
    phone = models.CharField(max_length=15)
    city = models.CharField(max_length=150)
    photo = models.ImageField(upload_to=pharmacy_photos_dir,
default="default/default.png", null=True, blank=True)
    owner = models.ForeignKey('users.User',on_delete=models.CASCADE,
null=True, blank=True, related_name="my_pharmacy")

    def__str__(self):
        return self.name


class Review(models.Model):
    text = models.TextField()
    rating = models.FloatField(null=True, blank=True)
    author = models.ForeignKey("users.User",
on_delete=models.CASCADE)
    pharmacy = models.ForeignKey(Pharmacy,
on_delete=models.CASCADE)
    created_date = models.DateTimeField(auto_now_add=True, blank=True,
null=True)

    def__str__(self):
        return self.pharmacy.name


class Manufacture(models.Model):
    name = models.CharField(max_length=150)

    def__str__(self):
```

```
            return self.name
        def product_photos_dir(instanse, filename):
            usrnme =  f'{instanse.name}'
            folder_name = f"{usrnme}/{filename}"
            return folder_name

        class Product(models.Model):
            name = models.CharField(max_length=150)
            manufacturer = models.ForeignKey(Manufacture,
on_delete=models.CASCADE)
            description = models.TextField(null=True, blank=True)
            photo = models.ImageField(upload_to=product_photos_dir,
default="default/default.png", null=True, blank=True)
            composition = models.TextField(null=True, blank=True)
            category = models.ForeignKey("categories.Category",
on_delete=models.CASCADE, null=True, blank=True)

            def__str__(self):
                return self.name

        class ReviewProduct(models.Model):
            text = models.TextField()
            rating = models.FloatField(null=True, blank=True)
            author = models.ForeignKey("users.User",
on_delete=models.CASCADE)
            product = models.ForeignKey(Product, on_delete=models.CASCADE)
            created_date = models.DateTimeField(auto_now_add=True, blank=True,
null=True)

            def__str__(self):
                return self.product.name


        class CountProduct(models.Model):
            pharmacy = models.ForeignKey(Pharmacy,
on_delete=models.CASCADE, null=True, blank=True)
            product = models.ForeignKey(Product, on_delete=models.CASCADE,
null=True, blank=True, related_name="available")
            count = models.BigIntegerField(null=True, blank=True)
            price = models.BigIntegerField(null=True, blank=True)
```

```
 def__str__(self):
      return self.product.name + " " + self.pharmacy.name
```

2. product/views.py

```
from django.shortcuts import render
from django.http import JsonResponse
from rest_framework import generics, permissions, status, views, viewsets
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.decorators import permission_classes
from .serializers import *
import random
from django.core.mail import send_mail
from django.conf import settings
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework import filters
from users.models import User
from categories.models import Category
from .models import *


class getProduct(viewsets.ModelViewSet):
   permission_classes = [permissions.AllowAny,]
   queryset =  Product.objects.all()
   serializer_class = ProductSer
   filter_backends = [DjangoFilterBackend, filters.SearchFilter]
   search_fields = ('name', 'description')
   filter_fields = ('category', )


class PharmacyS(viewsets.ModelViewSet):
   permission_classes = [permissions.AllowAny,]
   queryset = Pharmacy.objects.all()
   serializer_class = PharmacySer
   filter_backends = [DjangoFilterBackend, filters.SearchFilter]
   search_fields = ('name',)



class createProduct(APIView):
permission_classes = [permissions.IsAuthenticated,]
```

```
    def post(self, request):
        s = CreateProductSer(data=request.data)
        if s.is_valid():
            p = Product.objects.create(
                name = s.validated_data['name'],
                manufacturer=
Manufacture.objects.get(id=s.validated_data['manufacturer']),
                description = s.validated_data['description'],
                composition = s.validated_data['composition'],
                category = Category.objects.get(id = s.validated_data['category']),
                photo = s.validated_data['photo']
            )
            c = CountProduct.objects.create(
                product = p,
                pharmacy = request.user.my_pharmacy.all()[0],
                price = s.validated_data['price'],
                count = s.validated_data['count']
            )
            return Response({'status': 'ok'})
        else:
            return Response(s.errors)



class PharmacyApi(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def post(self, request):
        s = PharmacySer(data=request.data)
        if s.is_valid():
            p = Pharmacy.objects.create(
                name = s.validated_data['name'],
                owner = request.user,
                address = s.validated_data['address'],
                working_hours = s.validated_data['working_hours'],
                city = s.validated_data['city'],
                phone = s.validated_data['phone'],
                photo = s.validated_data['photo']
            )
            return Response({'status': 'ok'})
        else:
            return Response(s.errors)
```

```
class Accounting(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def get(self, request):
        user = Pharmacy.objects.get(owner=request.user)
        queryset = CountProduct.objects.filter(pharmacy = user)
        s = CountProductSer2(queryset, many=True, context={'request': request})
        return Response(s.data)


class ReviewApi(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def get(self, request, id):
        queryset = Review.objects.filter(pharmacy__id = id)
        s = ReviewSer(queryset, many=True)
        return Response(s.data)


class CreateReview(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def post(self, request):
        s = CreateReviewSer(data=request.data)
        if s.is_valid():
            Review.objects.create(
                text = s.validated_data['text'],
                author = request.user,
                pharmacy = s.validated_data['pharmacy'],
                rating = s.validated_data['rating']
            )
            return Response({'status': 'ok'})
        else:
            return Response(s.errors)


class ReviewProductApi(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def get(self, request, id):
```

```
        queryset = ReviewProduct.objects.filter(product__id = id)
        s = ReviewProductSer(queryset, many=True)
        return Response(s.data)


class CreateProductReview(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def post(self, request):
        s = CreateReviewProductSer(data=request.data)
        if s.is_valid():
            ReviewProduct.objects.create(
                text = s.validated_data['text'],
                author = request.user,
                product = s.validated_data['product'],
                rating = s.validated_data['rating']
            )
            return Response({'status': 'ok'})
        else:
            return Response(s.errors)


class Recomendation(APIView):
    permission_classes = [permissions.AllowAny,]

    def get(self, request):
        queryset = Product.objects.all()
        queryset = list(queryset)
        random.shuffle(queryset)
        s = ProductSer(queryset, many=True, context={'request': request})
        return Response(s.data)
```

3. product/urls.py

```
from django.urls import path, include
from .views import *
from rest_framework.routers import DefaultRouter

# router = DefaultRouter()
# router.register(r'list', UserList, basename='users')

urlpatterns = [
```

```
  path('', getProduct.as_view({'get': 'list'})),
  path('create/', createProduct.as_view()),
  path('pharmacy/create/', PharmacyApi.as_view()),

  path('accounting/', Accounting.as_view()),
  path('review/<id>', ReviewApi.as_view()),
  path('create/review/', CreateReview.as_view()),

  path('review/p/<id>', ReviewProductApi.as_view()),
  path('create/review/p/', CreateProductReview.as_view()),

  path('recomendation/', Recomendation.as_view()),
  path('pharmacy/', PharmacyS.as_view({'get': 'list'}))
]
# urlpatterns += router.urls
```

4. product/serilalizers.py

```
from rest_framework import serializers
from .models import *


class PharmacySer(serializers.ModelSerializer):
    class Meta:
        model = Pharmacy
        fields = "__all__"
        read_only_fields = ("owner",)


class CountProductSer(serializers.ModelSerializer):
    pharmacy = PharmacySer()
    class Meta:
        model = CountProduct
        fields = "__all__"


class ManufSer(serializers.Serializer):
    name = serializers.CharField()


class ProductSer(serializers.ModelSerializer):
    available = CountProductSer(many=True)
```

```
        manufacturer = ManufSer()
        photo = serializers.SerializerMethodField('get_avatar_url',
read_only=True)
        class Meta:
            model = Product
            fields = "__all__"
        def get_avatar_url(self, obj):
            return self.context['request'].build_absolute_uri(obj.photo.url)


    class CreateProductSer(serializers.Serializer):
        name = serializers.CharField()
        manufacturer = serializers.IntegerField()
        description = serializers.CharField()
        composition = serializers.CharField()
        price = serializers.IntegerField()
        count = serializers.IntegerField()
        photo = serializers.FileField()
        category = serializers.IntegerField()


    class ProductSer2(serializers.ModelSerializer):
        photo = serializers.SerializerMethodField('get_avatar_url',
read_only=True)
        class Meta:
            model = Product
            fields = "__all__"
        def get_avatar_url(self, obj):
            return self.context['request'].build_absolute_uri(obj.photo.url)

    class CountProductSer2(serializers.ModelSerializer):
        product = ProductSer2()
        class Meta:
            model = CountProduct
            fields = "__all__"


    class Author(serializers.Serializer):
        username = serializers.CharField()


    class ReviewSer(serializers.ModelSerializer):
```

```
    author = Author()
    class Meta:
        model = Review
        fields = "__all__"

class CreateReviewSer(serializers.ModelSerializer):
class Meta:
        model = Review
        fields = "__all__"
        read_only_fields = ('author',)


class ReviewProductSer(serializers.ModelSerializer):
    author = Author()
    class Meta:
        model = ReviewProduct
        fields = "__all__"

class CreateReviewProductSer(serializers.ModelSerializer):
    class Meta:
        model = ReviewProduct
        fields = "__all__"
        read_only_fields = ('author',)
```

5. users/models.py

```
from django.contrib.auth.models import AbstractUser
from django.db import models
class User(AbstractUser):
    address = models.CharField(max_length=150)


    def__str__(self):
        return self.username
```

6. users/views.py

```
from django.shortcuts import render
from django.http import JsonResponse
from rest_framework import generics, permissions, status, views, viewsets
from rest_framework.views import APIView
from rest_framework.response import Response
```

```
from rest_framework.authtoken.models import  Token
from rest_framework.decorators import permission_classes
from .serializers import *
import random
from django.core.mail import send_mail
from django.conf import settings
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework import filters
from .models import User
import uuid
from push_notifications.models import APNSDevice, GCMDevice

class Register(APIView):
    permission_classes = (permissions.AllowAny,)

    def post(self, request):
        s = RegisterSer(data=request.data)
        if s.is_valid():
            username = s.validated_data['username']
            email = s.validated_data['email']
            password = s.validated_data['password']
            if User.objects.filter(username=username).exists():
                return Response({'status': 'username exists'})
            elif User.objects.filter(email=email).exists():
                return Response({'status': 'email exists'})
            else:
                user = User.objects.create(username=username, email=email)
                user.set_password(password)
                user.save()
                return Response({'status': 'ok'})
        else:
            return Response(s.errors)



class Login(APIView):
    permission_classes = (permissions.AllowAny,)

    def post(self, request):
        s = LoginSer(data=request.data)
        if s.is_valid():
            username = s.validated_data['username']
```

```
            password = s.validated_data['password']
            user = User.objects.get(username=username)
            success = user.check_password(password)
            if success:
                if Token.objects.filter(user=user).exists():
                    token = Token.objects.get(user=user)
                else:
                    token = Token.objects.create(user=user)
                return Response({'key': token.key, 'uid': user.pk})
            else:
                return Response({'status': 'wrong'})
        else:
            return Response(s.errors)
class UserList(viewsets.ModelViewSet):
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = UserSer
    queryset = User.objects.filter(is_superuser=False, is_staff=False)

class changePassword(APIView):
    permission_classes = (permissions.AllowAny,)

    def post(self, request):
        s = EmailSer(data=request.data)
        if s.is_valid():
            email = s.validated_data['email']
            pwd = uuid.uuid4().hex[:10]
            user = User.objects.filter(email=email)
            if user.exists():
                user = user[0]
                user.set_password(pwd)
                user.save()
                send_mail(
                    'Agregator password',
                    f'Your password: {pwd}',
                    settings.EMAIL_HOST_USER,
                    [email,],
                    fail_silently=False)
                return Response({'status':'ok'})
            else:
                return Response({'status': 'not found'})
        else:
            return Response(s.errors)
```

```
class pushRegister(APIView):
    permission_classes = [permissions.IsAuthenticated,]

    def post(self, request):
        s = pushSerializer(data=request.data)
        if s.is_valid():
            android = GCMDevice.objects.filter(user=request.user)
            if android.exists():
                android = GCMDevice.objects.get(user=request.user)
                android.registration_id = s.validated_data['reg_id']
                android.save()
            else:
                GCMDevice.objects.create(user=request.user, active=True,
                                registration_id=s.validated_data['reg_id'],
                                cloud_message_type="FCM")
            return Response({'status': "ok"})
        else:
            return Response(s.errors)
```

7. users/urls.py

```
from django.urls import path, include
from .views import *
from rest_framework.routers import DefaultRouter

router = DefaultRouter()
router.register(r'list', UserList, basename='users')

urlpatterns = [
    path('register/', Register.as_view()),
    path('login/', Login.as_view()),
    path('change/password/', changePassword.as_view()),
    path('push/register/', pushRegister.as_view()),
]
urlpatterns += router.urls
```

8. users/serializers.py

```
from rest_framework import serializers
from .models import User

class RegisterSer(serializers.Serializer):
```

```
    username = serializers.CharField()
    email = serializers.CharField()
    password = serializers.CharField()


class LoginSer(serializers.Serializer):
    username = serializers.CharField()
    password = serializers.CharField()


class EmailSer(serializers.Serializer):
    email = serializers.CharField()


class UserSer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ('id', 'email', 'username', 'address')
    def update(self, instance, validated_data):
        instance.email = validated_data.get('email', instance.email)
        instance.username = validated_data.get('username', instance.username)
        instance.address = validated_data.get('address', instance.address)
        instance.save()
        return instance

class pushSerializer(serializers.Serializer):
reg_id = serializers.CharField()
```

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

5В070400 – «Есептеуіш техника және программамен қамтамасыз ету»

Токенова Улар Салимгерыйқызы

Тақырыбы: «Дәріханалардың мобильді қосымшасы-агрегаторының Back-end бөлігін әзірлеу»

ҒЫЛЫМИ ЖЕТЕКШІНІҢ СЫН-ПІКІРІ

Бұл жобаның өзектілігі көптеген себептермен расталады. Дипломдық жобада талқыланған мәселелер фармацевтика саласында мобильді қосымшаларды қолдануға байланысты өзекті болып табылады.

Дипломдық жоба кіріспеден, мазмұннан, негізгі бөлімнен, дәлірек айтсақ, технология теориясынан, екі тараудан, әр тарауға арналған қорытындыдан, қорытындыдан, пайдаланылған әдебиеттер тізімінен және сөздіктен тұрады. Бұл құрылымдау әдісі ғылыми тәжірибеде кең таралған және түсінуге оңай. Сондай-ақ, дипломдық жобаның дизайны қабылданған стандарттарға сәйкес келеді. Кіріспеде мақсаттар мен міндеттер тұжырымдалады, зерттеу объектісі сипатталады, жұмыс тақырыбын зерттеудің теориялық және әдістемелік әдістері көрсетіледі, қарастырылып отырған тақырыптың өзектілігінің себептері түсіндіріледі.

Осы дипломдық жобаның ғылыми жаңалығы мен практикалық маңыздылығы ақпараттық қызмет түріндегі қарапайым қолданушылар үшін, сондай-ақ фармацевтикалық дәріханалардың иелері үшін олардың тауарларын қадағалау есепке алуды жүргізу құралы түріндегі мобильді қосымшаның өзектілігі мен қажеттілігінде.

Жобаның бірінші тарауында пәндік аймақты зерттеу және осы қосымшаның даму мақсаттары қарастырылған.

Жобаның екінші тарауында агрегатордың мобильді қосымшасының back-end жағын жобалау кезінде қолданылатын құралдар мен технологиялардың теориялық негіздері келтірілген.

Дипломдық жұмыстың үшінші тарауында мәліметтер базасын толық талдау және қойылған міндеттерге сәйкес модельдердің сипаттамасы келтірілген. Мәліметтер базасын талдау және зерттеу кезінде логикалық және тұжырымдамалық модельдер қарастырылды.

Төртінші тарауда архитектура, мәліметтер базасының дизайны және мобильді агрегатор қосымшасының back-end жағын жобалау кезең-кезеңдік жоспары келтірілген.

Қорытындыда қойылған міндеттерге сәйкес келетін қорытындылар тізімі келтірілген. Шығарма авторы жасаған ұсыныстар мен ұсыныстар түпнұсқа болып табылады және егжей-тегжейлі қарастыруға тұрарлық.

Пайдаланылған дерек көздерінің тізімі дұрыс.

Жалпы, Төкенова У.С. дипломдық жобаның тақырыбы толық және егжей-тегжейлі ашылды, бұл оң шолуға ықпал етті. Дипломдық жұмыстың кемшіліктеріне сатып алу жүйесінің жоқтығы жатады, бірақ жұмыс бастапқыда агрегаторға арналған мобильді қосымша ретінде жасалғандықтан, бұл жасалу сапасына әсер етпейді.

Алдын ала бағалау нәтижелері бойынша мен авторды қорғауға қабылдауды қажет деп санаймын, орындаған жұмыс Төкенов У.С. қорғауға ұсынылған талаптарға толық сәйкес келеді.

Ғылыми жетекші: «Программалық инженерия» кафедрасының техн. ғыл.

магистрі, лектор _____Д.А. Баймбетов

«_04_»_____маусым____2021ж.

Метаданные

Название
Токенова Улар(Дәріхана агрегатор мобильді қосымшасы)методичка (1).docx

Автор
Токенова Улар

Научный руководитель
Даулет Байымбетов

Подразделение
ИКиИТ

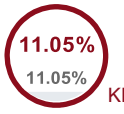
Список возможных попыток манипуляций с текстом

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	2
Парафразы (SmartMarks)	a	35

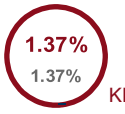
Объем найденных подобий

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



КП1

КП2



КЦ

25
Длина фразы для коэффициента подобия 2

4731
Количество слов

43695
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://pypi.org/project/djangorestframework-files/	16	0.34 %
2	https://www.bbsmax.com/A/mo5kyPlKzw/	15	0.32 %
3	https://www.django-rest-framework.org/api-guide/serializers/	15	0.32 %
4	https://ultimatedjango.com/learn-django/lessons/create-the-subscriber-view/	12	0.25 %
5	https://www.bbsmax.com/A/mo5kyPlKzw/	12	0.25 %

6	https://www.django-rest-framework.org/api-guide/serializers/	12	0.25 %
7	https://bbs.huaweicloud.com/blogs/139090	12	0.25 %

8	https://codedec.com/tutorials/how-to-send-email-in-django/	12	0.25 %
9	https://bbs.huaweicloud.com/blogs/139090	12	0.25 %
10	https://blog.csdn.net/weixin_43796109/article/details/102497048	11	0.23 %

из базы данных RefBooks (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из домашней базы данных (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из программы обмена базами данных (5.50 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	**Дослідження та програмна реалізація рекомендаційної системи для розширеної фільтрації відео** КУЩ Анастасія Володимирівна **12/8/2020** National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби")	155 (20)	3.28 %
2	**YFCNU/2019m/iftc/iftc_2019_213.pdf** YFCNU **10/28/2019** Yuriy Fedkovych Chernivtsi National University(CNU) (Deanery)	63 (10)	1.33 %
3	**Автоматизації формування освітньої траєкторії студентів з урахуванням інтересів стейкхолдерів.** В. С. Задорожній **12/11/2020** Kryvyi Rih National University (Кафедра автоматизації, комп'ютерних наук і технологій)	42 (5)	0.89 %

из интернета (5.56 %)

ПОРЯДКОВЫЙ НОМЕР	ИСТОЧНИК URL	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	https://www.bbsmax.com/A/mo5kyPlKzw/	66 (7)	1.40 %
2	https://www.django-rest-framework.org/api-guide/serializers/	36 (3)	0.76 %
3	https://www.pianshen.com/article/7243377107/	33 (4)	0.70 %
4	https://blog.csdn.net/weixin_43796109/article/details/102497048	32 (5)	0.68 %
5	https://bbs.huaweicloud.com/blogs/139090	24 (2)	0.51 %
6	https://pypi.org/project/djangorestframework-files/	23 (2)	0.49 %
7	https://stackoverflow.com/questions/59116235/reduce-duplicate-queries-in-django-modelform	18 (2)	0.38 %
8	https://codedec.com/tutorials/how-to-send-email-in-django/	12 (1)	0.25 %
9	https://ultimatedjango.com/learn-django/lessons/create-the-subscriber-view/	12 (1)	0.25 %
10	https://gist.github.com/z4none/67b9f1d8ae7f972c678dabfeaadaa951	7 (1)	0.15 %

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: Токенова Улар Салимгерыйқызы, Баймбетов Даулет

Название: Дәріханалардың мобильді қосымшасы-агрегаторының Back-end бөлігін әзірлеу

Координатор: Сейтбекова Е.С.

Коэффициент подобия 1: 11.05

Коэффициент подобия 2: 0.0

Замена букв: 1

Интервалы: 0

Микропробелы: 0

Белые знаки: 2

После анализа Отчета подобия констатирую следующее:

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

Обоснование: .....

Дата 27.05.2021г *Подпись Научного руководитель* 

Форматы	Зонасы	Поз-сы	Белгілеулер	Атаулары	Саны	Қоры-тынды

Өлш.	Құжат	№Бет	Қолы	Күні	Дипломдық жоба	Бет
						43