ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Ақпараттық және телекоммуникациялық технологиялар институты

«Программалық инженерия» кафедрасы

Қабылқақулы Айдын

Android платформасында «Online Hospital» мобильді қосымшасын әзірлеу

ТҮСІНДІРМЕ ЖАЗБА

дипломдық жобаға

5В060200 – «Ақпараттану» мамандығы

Алматы 2019

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Ақпараттық және телекоммуникациялық технологиялар институты

Программалық инженерия кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
ПИ кафедра меңгерушісі
тех. ғыл. кандидаты,
ассистент-профессор
_____Р. Юнусов
"30" мая 2019ж.

Дипломдық жобаға
ТҮСІНІКТЕМЕЛІК ЖАЗБА

Тақырыбы: Android платформасында «Online Hospital» мобильді қосымшасын әзірлеу

5В060200 – «Ақпараттану» мамандығы

Орындаған Қабылқақұлы А.

Ғылыми жетекші
Сениор- лектор
_____Б.М.Мустафина
"____" _____ 2019 ж.

Алматы 2019

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Ақпараттық және телекоммуникациялық технологиялар институты

Программалық инженерия кафедрасы

5В060200 – «Ақпараттану»

БЕКІТЕМІН
ПИ кафедра меңгерушісі,
тех. ғыл. кандидаты,
ассистент-профессор,
_____ Р. Юнусов
"20" мая 2019ж.

Дипломдық жоба орындауға
ТАПСЫРМА

Білім алушыға *Қабылқақұлы Айдын*
Тақырыбы: *« Android платформасында Online Hospital мобильді қосымшасын әзірлеу»*
Университет ректоры бұйрығының *№ 1841-б "14" наурыз 2019 ж.* шешімімен бекітілген.
Орындалған жобаның өткізу мерзімі *" 20 " мамыр 2019 ж.*
Дипломдық жобаның бастапқы мәліметтері: *Жобаның төлқұжаты, технология бойынша техникалық құжаттама, техникалық тапсырма, жоба диаграммлары түрінде ақпаратты жинау, деректер қорына сақтау, тестілеу, тексеруге арналған программалық қаматамаларды жасау жүргізілген.*
Есеп – түсініктеме жазбаның талқылауға берілген сұрақтардың тізімі:
а) тақырып бойынша талдау және есептің қойылымынжасау;
b) 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 13 слайдпен берілген құжат түрінде ұсынылған.
Ұсынылған негізгі әдебиеттер: *10 пайдаланылған әдебиеттер тізімінен*

Дипломдық жобаны орындау

КЕСТЕСІ

Бөлімдердің атаулары, зерттелген мәселелердің тізімі	**Ғылыми жетекшіге және кеңесшілерге ұсыну мерзімі**	**Ескерту**
1. Диплом жұмысының жоспар құрылымын құру.	14.01.2019	жоқ
2. Тапсырма қойылымы және бағдарламалау ортасын таңдау	18.01.2019	жоқ
3. Зерттеу тақырыбы бойынша ғылыми теориялық материалдарды жинау және негізгі бөлім беру бойынша есеп беру жазбасын дайындау	01.02.2019	жоқ
4. Дипломның екінші бөлімі – жобалау сызбаларын дайындау.	15.02.2019	жоқ
5. Жобаның веб-қосымшасын тестілеуден өткізу.	18.03.2019	жоқ
6. Дипломдық жобаға түсіндірме жазба жазуды аяқтау	26.04.2019	жоқ

Дипломдық жұмыс бөлімдерінің кеңесшілерінің аяқталған жұмысқа қойған

қолтаңбалары

Бөлімдер атауы	**Кеңес берушілер (аты-жөні, тегі, ғылыми дәрежесі, атағы)**	**Қолтаңба қойылған мерзімі**	**Қолы**
Нормалық бақылаушы	С.Б.Қалдыбеков Сениор-лектор, магистр	16.05.19	
Бағдарламалық бөлім	А.А.Таурбекова Тьютор	16.05.19	

Ғылыми жетекші _____ Б.М.Мустафина

Тапсырманы орындауға қабылдап алған студент _____ А. Қабылқақұлы

Күні «29» 04 2019ж.

АҢДАТПА

Бұл проектінің жасалу барысында қолданушыға деген барлық қолайлылық көзделуде. Басқа да ұқсас қосымшаларға қарағанда артықшылықтарды арттыру басты мақсаттың бірі.

Қосымшаның басты қызметі қолданушылардың уақытын үнемдеу. Ол үшін түсінікті әрі ыңғайлы интерфейс ұсынылған.

Қосымшаның басты артықшылығы қазіргі заман талабына сай платформаларда жұмыс жасай береді, айта кетсек Android, IOS, Windows Phone, BlackBerry. Дипломдық жұмыстың мақсаты пайдаланушылар мобильді қосымша арқылы дәрігерлерге жазылуды жедел және өздеріне тиімді етіп таңдауға көмектесетін қосымшасын құру.

АННОТАЦИЯ

При разработке данного проекта предусматривается все удобства для пользователей. Одной из главных целей является повышение преимуществ по сравнению с другими аналогичными приложениями.

Основной функцией приложения является экономия времени пользователей. Для этого представлен понятный и удобный интерфейс.

Главным преимуществом приложения является использование современных платформ, таких как Android, IOS, Windows Phone, BlackBerry. Целью дипломной работы является создание приложений, которые помогут пользователям быстро и эффективно выбрать запись к врачам через мобильные приложения.

ANNOTATION

In the development of this project provides all the convenience for users. One of the main goals is to increase the benefits compared to other similar applications.

The main function of the application is to save users ' time. To do this, presented a clear and user-friendly interface.

The main advantage of the application is the use of modern platforms such as Android, IOS, Windows Phone, BlackBerry. The aim of the thesis is to create applications that will help users quickly and efficiently select an entry to doctors through mobile applications.

МАЗМҰНЫ

КІРІСПЕ

Қазіргі интернет пен смарфондар дамыған заманда мобильді қосымшалардың алатын орны ерекше. Смартфон озық функциялары бар ұялы телефон. Смартфонда жоғары ажыратымдылығы бар сенсорлық экран, WiFi байланысы, веб-браузинг мүмкіндіктері және күрделі қосымшаларды қабылдау мүмкіндігі бар. Осы құрылғылардың көпшілігі Android, iOS, Blackberry операциялық жүйесі және Windows операциялық жүйелері сияқты танымал мобильді операциялық жүйелердің кез келгенінде жұмыс істейді. Смартфондардың технологиясы мен ерекшеліктері бір операциялық жүйеден екіншісіне байланысты. Смартфондарда қолданылатын операциялық жүйеге байланысты смартфондар жіктеледі. Бұл жіктеу негізінен Android, iOS және Windows сияқты үш негізгі операциялық жүйелерге бағытталған.

Кез-келген интернетке мүмкіндігі мен смартфоны бар тұтынушы уақыт пен орнына қарамастан өзіне керек ақпартты ала алды. Белгілі бір сипаттағы ақпарттарға жиі жүгінуне қажетті тұтынушылар ортасы кеңейді. Бұл осы қажеттіліктерді қанағаттандыра алатын арнайы қосымшалардың құрылуына қажеттілігін тудырады. Қосымшалардың мүмкіндіктерін төмендегіше анықтауға болады:

– сұранымды қанағаттандыратын мамандырылған қосымшалар;
– лезде әрекет ету ұлғайтылған;
– тек қажетті ақпарат жүктеледі,яғни интернет трафикті үнедеу;
– жармана болмауы;
– қаралым үшін аса ыңғайлы интерфейс;
– аса ыңғайлы функцияналдылық.

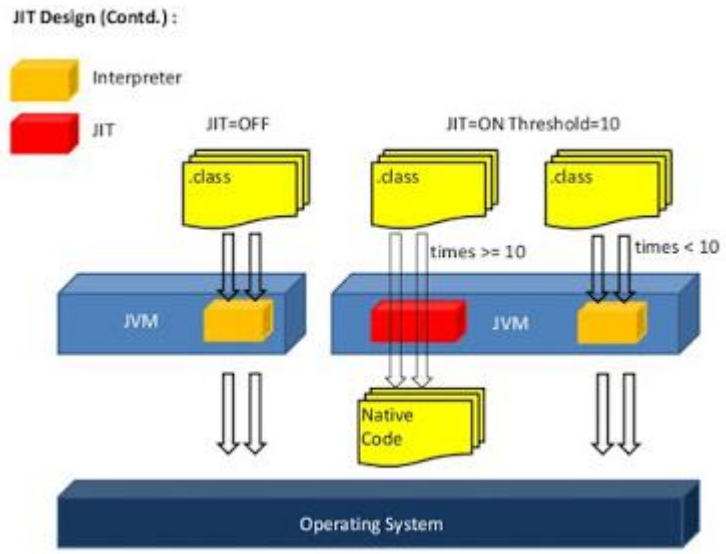
Дипломдық жобаның мақсаты – онлайн түріндегі клиниканың мобильді қосымшасын құру. Бұл жобаның артықшылықтары:

– үйде отырып клинакадағы дәрігерлер жайлы ақпарат алу;
– мобильді қосымша арқылы дәрігерлерге жазылу;
– дәрігерлердің қызметтерін қарау.

1 Қосымшаларды әзірлеудің теориялық негіздері

1.1 Android бағдарламалардың құрылғыларда орындалуы

Егер сізге Linux операциялық жүйесі мен процесс түсінігі таныс болса, Android-қосымшалардың қалай орындалатынын түсіну қиын болмайды. Әдепкі бойынша, Android операциялық жүйесі әрбір қолданба бірегей пайдаланушы идентификаторын береді. Android қосымшаларын іске қосқаннан кейін, олардың әрқайсысы өзінің виртуалды машинасында өз процесінде орындалады. 1.1-суретте Android бағдарламалардың құрылғыларда орындалуы көрсетілген.



1.1-сурет – Android бағдарламалардың құрылғыларда орындалуы

Қажет болған жағдайда, Android операциялық жүйесі қосымшалар процестерін іске қосуды және тоқтатуды басқарады. Бұл дегеніміз, Android-тың барлық қосымшалары бір-бірінен бөлек жұмыс істейді, бірақ, әрине, аппараттық және басқа да жүйелік ресурстарға қол жеткізуді сұрата алады.

Егер сіз мобильді қосымшаларды әзірлеумен таныс болсаңыз, мысалы, J2ME-де, қол жеткізу құқығы (permissions) ұғымымен бетпе-бет келген боларсыз. Android қолданбасын орнату немесе іске қосу кезінде, ол Интернетке, телефон кітабына немесе басқа да жүйелік ресурстарға қол жеткізу үшін қажетті құқықтарды сұратады. Пайдаланушы бұл құқықтарды анық береді, әйтпесе әрекет бас тартылады.

Барлық осы қол жеткізу құқықтары Android бағдарламасының манифест файлында сипатталған. Java-ға қарағанда, Android манифесті-қосымшаның

айырмашылығы барлық компоненттері мен оларға арналған теңшелімдер XML-файлда көрсетілген.

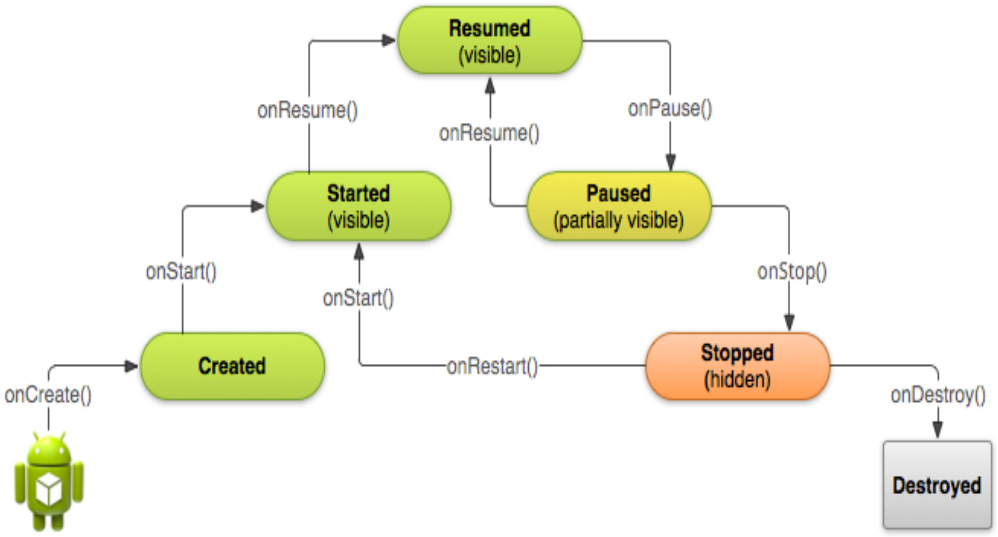
Android бағдарламасының төрт негізгі компоненті: белсенділік, сервистер, контент жеткізушілері және кең тарату қабылдағыштары (broadcast receivers). Олардың ішінде көбінесе Android қосымшасының жеке экрандық формасына сәйкес белсенділік бар. Мысалы, Android операциялық жүйесіне арналған ойында бірнеше экрандар болуы мүмкін: жүйеге кіру үшін, рекордтар, нұсқаулықтар және ойынның экраны. Осы элементтердің әрқайсысы қолданбадағы әр түрлі белсенділікке сәйкес келеді.

Java сияқты, Android ОЖ-де әзірлеушінің орнына кейбір міндеттерді орындайды, мысалы, белсенді нысандарды жасайды. Белсенділікті ұйымдастыруға System сыныбы жауап береді. Егер белсенділікті іске қосу керек болса, Intent нысаны бар startActivity() әдісін параметр ретінде шақыру жеткілікті. Бұл қоңырауға жауап ретінде, System сыныбы немесе жаңа белсенділік нысанын жасайды немесе ескісі қайта пайдаланады.

Жадты қайта пайдаланудың өте маңызды міндеті үшін жауап беретін Java тіліндегі қоқыс жинау сияқты, Android қосымшаларды іске қосуды, тоқтатуды, жасауды және жоюды басқарады. Мүмкін, ол тым көп шектейді, бірақ бұл олай емес. Android осы үдеріске араласу үшін қайта анықтауға болатын өмірлік цикл оқиғаларын ұсынады.

1.2 Android қолданбасының өмірлік циклі

Android-дегі қосымшаның өмірлік циклы жүйемен тығыз байланысты және пайдаланушы қажеттіліктеріне, қол жетімді ресурстарға және т.б. байланысты. Мысалы, пайдаланушы браузерді іске қосқысы келеді. Өтінішті қабылдау туралы шешім жүйе арқылы жасалады. Соңғы сөз жүйеге қалдырылғанымен, ол белгілі бір қисынды ережелерге бағынады, ол жүктеуге, оны тоқтатуға немесе оны тоқтатуға болатынын анықтайды. Егер пайдаланушы қазіргі уақытта белгілі бір терезеде жұмыс істесе, жүйе тиісті бағдарламаға басымдық береді. Керісінше, егер терезе көрінбейтін болса және жүйе қосымша ресурстарды жұмылдыру үшін бағдарламаның жұмысы тоқтатылуға тиіс деп шешсе, төменгі басымдылығы бар бағдарлама тоқтатылады. Android-де ресурстары шектеледі, сондықтан Android қосымшаларды қатаң бақылайды. Android қолданбасының өмірлік циклі 1.2-суретте көрсетілгендей алты күйден тұрады.



1.2-сурет – Android қолданбасының өмірлік циклі

Қосымшаның өмірлік циклінің негізгі әдістері:

– protected void onCreate();
– protected void onStart();
– protected void onResume();
– protected void onPause();
– protected void onStop();
– protected void onDestroy().

OnCreate(). OnCreate () әдісі әрекетті жасау немесе қайта іске қосу кезінде шақырылады. Жүйе оқиғаларға байланысты ағымдағы терезелерді бастауға және тоқтатуға болады. Бұл әдіс ішінде статикалық белсенділік интерфейсін орнатыңыз. Статикалық әрекет деректерін инициалдайды, деректерді тізімдермен байланыстырады және т.б. Қажетті деректер мен ресурстармен байланыстырады. Көріністі setContentView () әдісі арқылы орнатады.

Осы әдіс арқылы пайдаланушы интерфейсін жүктеңіз, класстың сипаттарына сілтемелер жіберіңіз, деректерді басқару элементтеріне байланыстырып, қызметтер мен ағындарды жасаңыз. OnCreate () әдісі соңғы қоңырауда сақталған пайдаланушы интерфейсінің күйін қамтитын Bundle нысанын onSaveInstanceState өңдегішіне қабылдайды. GUI-ді өзінің алдыңғы күйінде қалпына келтіру үшін осы айнымалы мәнді пайдалану керек: onCreate ішінде () немесе onRestoreInstanceState () әдісін елемеу.

Уақытты жұмсайтын инициализациялау операциялары onCreate () әдісін пайдаланудың орнына фондық үрдісте орындалуы керек. Әйтпесе ANR тілқатысу терезесін алуға болады (Қолданба жауап бермейді, бағдарлама жауап бермейді).

OnStart(). OnCreate() үшін onStart() шақыруы керек, бірақ onStart() алдында onCreate() міндетті емес, өйткені onStart () тоқтатылған қосымшаның жұмысын қайта бастау үшін (қолданба onStop() әдісімен тоқтатылады). Шақыру кезінде onStart() терезесі тағы пайдаланушыға көрінбейді, бірақ көп ұзамай болады көрініп тұр. Белсенді пайдаланушыға көрінетін болғанға дейін тікелей шақырылады. OnResume() әдісі, егер белсенділік алдыңғы жоспарды алса, немесе onstop() әдісі жасырын болса, шақырумен сүйемелденеді.

OnResume(). OnResume() әдісі onStart() кейін шақырылады, тіпті терезе басым режимде жұмыс істесе де, пайдаланушы оны бақылай алады. Осы уақытта пайдаланушы сіз жасаған тереземен өзара әрекеттеседі. Қосымша монополиялық ресурстарды алады. Анимация, аудио және бейне ойнатылады. Сондай-ақ, onPause() кейін шақырылуы мүмкін.

Жүйе бұл әдісті сіздің белсенділігіңіз алдыңғы жоспарда, соның ішінде, бірінші құру кезінде жүргенде шақыратынын есте ұстаңыз. Осылайша, сіз onpause()-да босаған/тоқтата тұрған кез келген кең таратылатын қабылдағыштарды немесе басқа да процестерді тіркеу үшін onResume() жүзеге асыруыңыз және белсенділік қайтадан белсенді болған кезде орын алуы тиіс кез келген басқа да бастамашылықтарды орындауы тиіс.

Қолданба экраннан жасыру немесе алдыңғы жоспарға шығу кезінде жауапты болу үшін салыстырмалы түрде жылдам және жеңіл кодты орналастыруға тырысыңыз.

Бұл функциялар onCreate() және onRestoreInstanceState өңдеушілеріне жүктелген болғандықтан, пайдаланушы интерфейсінің күйін қайта жүктеудің қажеті жоқ.

OnPause(). Жаңа тереземен жұмыс істеуге ауысуды шешкен кезде, жүйе үзілген терезеге onPause() әдісі шақырады. Шын мәнінде, белсенділіктің ұюы болады. Тіркелмеген деректерді сақтайды. Монополиялық ресурстарды белсендіреді және шығарады. Бейне, аудио және анимация ойнатуды тоқтатады. OnPause() шақыруға onResume() немесе onStop() өтуге болады.

Бұл әдіс анимацияны және процессорды жүктейтін басқа да әрекеттерді тоқтату керек. Сақталмаған деректерді, мысалы, жазу жазбасын тіркеу, себебі оны орындағаннан кейін белсенділік жұмысы ескертусіз үзілуі мүмкін. GPS деректерді өңдеу сияқты жүйелік ресурстарды босату.

Қолданба экраннан жасыру немесе алдыңғы жоспарға шығу кезінде жауапты болу үшін салыстырмалы түрде жылдам және жеңіл кодты орналастыруға тырысыңыз.

Қолданба архитектурасын негізге ала отырып, сіз алдыңғы жоспарда белсенділік пайда болғанша ағындарды, процестерді немесе кең таратылатын қабылдағыштарды орындауды тоқтата аласыз.

Мысалы, камерамен жұмыс істеу кезінде әдіс 1.3-суретте көрсетілгендей қолданылады:

```
@Override
public void onPause() {
    super.onPause();

    // Release the Camera because we don't need it when paused
    // and other activities might need to use it.
    if (mCamera != null) {
        mCamera.release()
        mCamera = null;
    }
}
```

1.3-сурет – Камерамен жұмыс әдісі

Сонымен қатар, сіз тұрақты сақтау үшін пайдаланушы өзгерістерін сақтау үшін onPause() пайдаланбауыңыз керек. Пайдаланушылар автоматты түрде сақталатын өзгерістерді күтетініне сенімді болғанда ғана рұқсат етіледі (мысалы, электрондық пошта жасау кезінде). Сонымен бірге, сіз onPause()-да деректер базасындағы жазба сияқты қарқынды жұмысты орындаудан аулақ болу керек, себебі бұл келесі белсенділікке ауысуды баяулатуы мүмкін (оның орнына onStop() өшіру операциясы кезінде ауыр жүктемені орындау керек.

Белсенді тоқтаған кезде, барлық компоненттер жадта сақталады және қайта жаңғырту кезінде оларды қайта бастау қажет емес.

OnStop(). OnStop() әдісі терезе пайдаланушыға көрінбейтін болған кезде туындайды. Бұл оны жою кезінде немесе ағымдағы белсенділік терезесін жабатын басқа белсенділік (бар немесе жаңа) іске қосылса, орын алуы мүмкін. Кез келген onRestart() әдісінің кез келген шақыруы әрқашан жүреді, егер белсенділік пайдаланушымен өзара әрекет ету үшін қайтарылса, немесе onDestroy() әдісінің, егер бұл әрекет жойылса.

Белсенділік тоқтаған кезде, белсенділік объектілері жадта сақталады және белсенділік өз жұмысын жаңартқан кезде қалпына келтіріледі. Бұрын жасалған компоненттерді қайта бастау қажет емес. Сонымен қатар, жүйе әр көрініс үшін ағымдағы жағдайды қадағалайды, сондықтан пайдаланушы мәтінді мәтін өрісіне енгізсе, оның мазмұны сақталады және оны сақтау және қалпына келтірудің қажеті жоқ.

Ескертпе: Егер жүйе тоқтаған кезде сіздің белсенділігіңізді жабса да, ол әлі де арнайы Bundle объектісіндегі EditText мәтіні сияқты нысандардың жағдайын сақтайды (кілт-мән түрінде) және пайдаланушы сол белсенділік данасына қайта ауысса, оларды қалпына келтіреді.

Бұл әдіс деректерді сақтау үшін күрделі операцияларды жасауға болады: күрделі анимацияны, ағындарды, датчиктердің көрсеткіштерін, GPS сұрауларын, таймерлерді, Сервистерді немесе пайдаланушы интерфейсін жаңарту үшін қажет басқа да процестерді тоқтату үшін. Интерфейсті жаңарту үшін ресурстарды

(орталық процессордың тактілерін немесе желілік трафикті) тұтынудың мәні жоқ, ал ол экранда көрінбейді. Onstart() немесе onRestart() әдістерін белсенділік қайтадан көрінетін болған кезде осы процестерді жаңарту немесе қайта іске қосу үшін қолданыңыз.

Жад жетіспеген жағдайда, жүйе onstop() әдісін айналып өтіп, onDestroy() әдісін шақыра отырып, жасырын белсенділікті жоя алады.

OnRestart(). Егер терезе onStop() шақыруынан кейін басым режимге оралса, онда onRestart() әдісі шақырылады. Яғни, белсенділік тоқтағаннан кейін және пайдаланушы қайтадан іске қосылғаннан кейін туындайды. Әрқашан onStart() әдісін шақырумен жүреді.

onRestart onStart() (біріншісінен басқа) әдісінің алдында. Оны "толық" күй шеңберінде белсенділікті қайта іске қосқан кезде ғана орындалуы тиіс арнайы әрекеттер үшін пайдаланыңыз.

OnDestroy(). Әдіс белсенділік жұмысы аяқталғаннан кейін, finish() әдісін шақырғанда немесе жүйе ресурстарды босату үшін белсенділіктің осы данасын жойған жағдайда шақырылады. Бұл екі жою сценарийі isfinishing() әдісін шақыруға болады. Белсенділікті жою алдында туындайды. Бұл жүйенің белсенділігін алатын соңғы сұраныс. Егер белгілі бір терезе әйнектің жоғарғы жағында болса, бірақ пайдаланушыға көрінбейтін болса және жүйе бұл терезені аяқтауды шешеді, onDestroy() әдісі шақырылады. Бұл жағдайда әдіс белсенділіктің барлық статикалық деректерін жояды. Барлық пайдаланылатын ресурстарды береді.

Ресурстарды босату бойынша барлық қажетті операцияларды сіз onStop() әдісінде жасадыңыз, онда бұл әдіс бойынша сіз сақтандыру және барлық бос емес ресурстарды тағы да тексере аласыз.

Іс жүзінде сіз onCreate(), onResume() және onPause() әдістерімен жиі кездестіресіз. OnCreate() әдісі тереземен жұмыс істеу үшін пайдаланушы интерфейсін жасау кезінде шақырылады. Бұл әдіс сізге деректерді компоненттермен байланыстыруға және оқиғаларды өңдеушілерді пайдаланушы интерфейсінің компоненттеріне қосуға мүмкіндік береді. OnPause() көмегімен маңызды ақпаратты қолданбаның деректер базасында сақтай аласыз. Бұл жүйе бағдарламаның жұмысын аяқтамас бұрын шақыратын соңғы қауіпсіз әдіс. OnDestroy() әдісі міндетті түрде шақырылмайды, сондықтан сыни логиканы жүзеге асыру кезінде осы әдіске сүйенбеңіз.

2 Жалпы бөлім

2.1 Ionic Framework

Ionic Freamwork – кеңінен талқыланған шеңберлердің бірі. Ресми сайттың мәліметі бойынша, Ionic – AngularJS, SASS, Apache Cordova негізіндегі гибридті мобильді қосымшаларды, CSS және JS компоненттерінің жиынтығын құруға арналған SDK.

Ionic репозиторийде 15 300 жұлдыз бар және Drifty компаниясының өнімдері Ionic-қа тиесілі инвестиция көлемі 3,7 миллион долларға жетті.

Шын мәнінде, бұл қосымша мүмкіндіктер беретін Cordova CLI үстінен орау:

– үлгіні таңдаумен базалық қолданбаны жасау (мысалы, бүйірлік мәзірі бар бағдарлама, Google Maps табақтары, карталары бар бағдарлама, бос бағдарлама);

– эмуляторда, нақты құрылғыда, браузерде құрастыру және іске қосу;

– браузерде және құрылғыда live reload.

Ionic CLI болуы міндетті емес, бірақ бұл бағдарламаны әзірлеуді әлдеқайда жеңілдетеді. Біздің ойымызша, маңызды мүмкіндіктер:

– $ionic resources.

Бұл пәрмен сіз бастапқы файлдардан (.psd, .png, .ai) мақсатты платформалар үшін барлық өлшемді шкалалардың экранының белгішелерін жасауға мүмкіндік береді. Ол үшін бастапқы белгіше өлшемі 192 × 192 пиксельден кем болмауы керек, ал арнайы PSD үлгісімен дайындалған кем дегенде 2208 × 2208 пиксель мөлшері бар шашыраңқы экран үшін бастапқы кескін жеткілікті.

Ionic View iOS және Android үшін мобильді қосымша болып табылады, онда сіз өзіңіздің әзірленген қосымшаларыңызды тұтынушылар, тестерлер, әріптестермен бөлісе аласыз. «$Ionic upload» командасы қосымшаны сіздің компьютеріңізде сервермен синхрондайды, содан кейін оны телефонда іске қосуға болады.

Айта кету керек, қазіргі уақытта Ionic View beta нұсқасында. Android-де қолдану барысында біз жиі түсініксіз қателіктерге ұшырадық – өмірдің белгілері жоқ ақ бет, бағдарлама мүлдем басталмады. Белгілі болғандай Cordova плагиндерінің барлығы Ionic View нұсқасының ағымдағы нұсқасында қолдаулы емес.

2.2 Apache Cordova

Apache Cordova – ашық бастапқы кодты мобильді қосымшаларды әзірлеу платформасы. Ол кросс-платформаны дамыту үшін HTML5, CSS3 және JavaScript

секілді стандартты веб-технологияларды қолдануға мүмкіндік береді, мобильді платформалардың әрқайсысы үшін өзіндік даму тілінен аулақ болады. Қолданбалар әр платформаға бағытталған қабықтың ішінде жұмыс істейді және құрылғы сенсорларына, деректерге және желі күйіне кіру үшін стандартты API-ке сүйенеді.

Apache Cordova инкубациялық кезеңді 2012 жылдың қазан айында Apache Software Foundation (ASF) шеңберінде негізгі жоба ретінде аяқтады. АСФ-ға қатысу арқылы Кордованың болашақ дамуы ашық жобаларды басқаруды қамтамасыз етеді. Ол Apache лицензиясының 2.0 нұсқасы бойынша әрдайым ашық және ашық күйде қалады. Қосымша ақпарат алу үшін cordova.apache.org сайтына кіріңіз.

Apache Cordova-ді келесі жағдайларда қолданыңыз:

– мобилді әзірлеуші және әр платформаны дамыту тіліне және құралдар қорабына қайта енгізуді қажет етпей, қосымшаны бірнеше платформаға кеңейтуді қалайды;

– веб-әзірлеуші және әртүрлі қолданбалар дүкендерінде тарату үшін бумаға салынған веб-бағдарламаны орналастырғысы келеді;

– WebView (Арнайы Браузер терезесі) көмегімен өз қосымшасының компоненттерін араластыруға мүдделі ұялы әзірлеуші құрылғы деңгейіндегі API-ке қол жеткізе алады немесе егер сіз жергілікті және WebView компоненттері арасында қосылатын модульді жасауды қаласаңыз.

Cordova қосымшалары бағдарлама туралы ақпараты бар және бағдарламаның жұмыс істеуіне әсер ететін параметрлерді анықтайтын жалпы config.xml файлына сүйенеді, мысалы, ол құрылғының бағдарындағы өзгерістерге жауап береді. Бұл файл W3C спецификациясының Packaged Web Applications немесе виджетіне сәйкес келеді.

Бағдарламаның өзі веб-бет ретінде әдепкі бойынша кез-келген CSS, JavaScript, суреттер, мультимедиалық файлдарды немесе іске қосу үшін қажетті басқа ресурстарға сілтеме жасайтын index.html деп аталатын жергілікті файл болып табылады. Қолданба қолданбалар дүкендеріне таратылатын қолданбаның қабығы ішінде веб-шолғыш ретінде жұмыс істейді.

Cordova қолдауымен WebView бағдарламалары және оның барлық пайдаланушы интерфейсі көрсетілуі мүмкін. Кейбір платформаларда ол WebView-ді қосымшаның басқа компоненттерімен біріктіретін үлкен, гибридті қосымшаларда компонент болуы мүмкін. (Толық ақпаратты «Интеграцияланған веб-шолулар» бөлімінен қараңыз).

Қосылатын модуль Cordova және басқа компоненттер үшін бір-бірімен өзара әрекеттесу үшін қол жетімді. Бұл JavaScript тілінен платформа тілінде кодты шақыруға мүмкіндік береді. Ең дұрысы, бұл машина коды үшін javascript API бірнеше құрылғының платформаларында сәйкес келеді. 3.0 нұсқасынан бастап қосылатын модульдер құрылғының стандартты API интерфейстеріне

байланыстырады. Үшінші тарап қосылатын модульдер барлық платформаларда міндетті түрде қол жетімді емес мүмкіндіктерге арналған қосымша байламдарды қамтамасыз етеді. Сіз бұл үшінші тарап қосылатын модульдерін плагиндер тізімінде таба аласыз және оларды қолданбаңызда қолданасыз. Сондай-ақ, «Плагинді дамыту жөніндегі нұсқаулық» бөлімінде сипатталғандай өз плагиндеріңізді де дамыта аласыз. Плагиндер, мысалы, Кордова мен оның компоненттері арасындағы байланыс үшін қажет болуы мүмкін.

Ескерту: 3.0 нұсқасынан бастап, Cordova жобасын жасаған кезде оның ешқандай плагины жоқ. Бұл жаңа әдепкі мінез. Қалаған плагиндер, тіпті негізгі плагиндер де нақты қосылуы керек.

Кордова ешқандай пайдаланушы интерфейс виджеттерін немесе MV шеңберлерін қамтамасыз етпейді. Кордова тек қана орындалатын жұмыс уақытын қамтамасыз етеді. Егер сіз UI виджеттерін және / немесе MV құрылымын пайдаланғыңыз келсе, оларды таңдап, оларды үшінші тараптың ресурстары ретінде қолданбаңыз.

3.0 нұсқасынан бастап, мобильдік қосымшаларды жасау үшін екі негізгі жұмыс үрдісін пайдалануға болады. Бірдей тапсырманы орындау үшін кез-келген жұмыс процесін пайдалана алсаңыз да, осы жолдардың әрқайсысының артықшылығы бар:

– кросс-платформалық жұмыс үрдісі: егер сіздің қолданбаңыз мүмкіндігінше платформаға тән дамудың минималды қажеттіліктері бар көптеген мобильді платформаларда жұмыс істеуін қаласаңыз, осы жұмыс процесін пайдаланыңыз. Бұл жұмыс процесі Cordova 3.0-ден бастап енгізілген Cordova CLI деп аталатын Cordova утилитасының жанында пайда болады. CLI – жоғары дәрежелі құрал, ол сізге төмен деңгейлі сценарийлердің функционалдығын мүмкіндігінше тез сюрлеуге мүмкіндік беретін бірнеше платформаға арналған жобаларды жасауға мүмкіндік береді. CLI веб-ресурстардың жалпы жиынтығын әр ұялы платформаға арналған ішкі каталогтарға көшіреді, әр платформа үшін теңшелімге қажет өзгертулер енгізеді, орындалатын қолданбалар файлдарын жасау үшін құрастыру сценарийлерін іске қосады. CLI сонымен қатар қосымшаға плагиндерді қолдану үшін ортақ интерфейсті ұсынады. CLI туралы қосымша ақпарат алу үшін «Пәрмен жолы интерфейсі» бөлімін қараңыз. Егер сізге жұмыс үрдісінің тұғырын ортаға қою қажет болса, кросс-платформа жұмыс үрдісі ұсынылады;

– платформаға негізделген әзірлеу процесі: Егер сіз бір платформаға арналған қосымшаны құруға назар аударғыңыз келсе, осы процесті қолданыңыз және өзгертулерді төмен деңгейде жасауыңыз керек. Сіз бұл тәсілді пайдалануыңыз керек, мысалы, егер сіз өзіңіздің бағдарламаңызды Cordova веб-негізіндегі компоненттерімен бірге WebViews Integration бөлімінде сипатталғандай біріктіруін қаласаңыз. Әдетте SDK көмегімен жобаны өзгерту қажет болса, бұл жұмыс процесі қолданылады. Бұл жұмыс процесі әр қолдау

көрсетілетін платформаға және жеке плагин утилитасына арналған төменгі деңгейлі сценарийлер жиынтығына негізделген, бұл плагиндерді жеке таңдаған платформаға қолдануға мүмкіндік береді. Бұл жұмыс процесін кросс-платформалық қосымшаларды жасау үшін қолдануға болады, бірақ әдеттегідей қиынырақ болады, себебі жоғары деңгейдегі утилиталардың болмауы жеке жинақтау циклдарын және әрбір платформаға арналған қосылатын модульдерді өзгертуді білдіреді. Дегенмен, бұл жұмыс процесі әр SDK ұсынған әзірлеу параметрлеріне үлкен қол жеткізуге мүмкіндік береді және күрделі гибридтік қосымшалар үшін маңызды.

Бастапқыда, «Пәрмен жолы интерфейсі» бөлімінде сипатталғандай, бағдарламаны жасау үшін кросс-платформалық жұмыс үрдісін пайдалану оңай болуы мүмкін. Содан кейін SDK ұсынған басқарудың үлкен дәрежесі қажет болса, сіз платформаға бағдарланған даму процесіне баруға мүмкіндігіңіз бар. Төмен деңгейдегі утилиталар cordova.apache.org сайтында CLI-ге қарағанда бөлек тарату тізімінде қол жетімді. CLI-тің бастапқыда жасаған жобалары үшін бұл құралдар жобаның әр түрлі жобаларында / платформаларында / * / cordova каталогтарында қол жетімді.

Ескертпе: CLI негізіндегі жұмыс үрдісінен платформаға бағдарланған үрдіс пен пәрмен жолы құралдарына ауысқаннан кейін, сіз қайта орала алмайсыз. CLI платформаға тән бастапқы кодты өзгерту үшін әрбір құрылым үшін пайдаланатын кросс-платформалық бастапқы кодтың жалпы жиынтығын қолдайды. Платформа үшін арнайы ресурстарға жасаған кез-келген өзгерістерді сақтау үшін, сіз кросс-платформаның бастапқы кодын елемейтін және платформа үшін арнайы бастапқы кодқа сүйенетін платформаға бағдарланған құралдарға баруыңыз керек.

Cordova қондырғысы жоғарыдағы жұмыс үрдісіне байланысты әр түрлі болады, сіз таңдайсыз:

– кросс-платформалық жұмыс процесі: «Командалық жол интерфейсі» бөлімін қараңыз;

– жұмыс процесінің платформасының ортасында: «Қолдау көрсетілетін платформалар нұсқаулығы» бөлімін қараңыз;

– Cordova-ны орнатқаннан кейін сіз дамитын мобильді платформалар үшін «Қолдау көрсетілетін платформаларға арналған нұсқаулық» бөлімін оқып шығу ұсынылады. Сондай-ақ, Құпиялылық нұсқаулығы, Қауіпсіздік нұсқаулығы және Келесі қадамдарды шолуды ұсынамыз.

2.3 Firebase (RealtimeDatabase)

Firebase Realtime дерекқоры бұлт дерекқоры болып табылады. Деректер JSON форматында сақталады және әр қосылған клиентпен нақты уақыт режимінде

синхрондалады. IOS, Android және JavaScript үшін SDK-лерді пайдаланып кросс-платформа қосымшаларын жасаған кезде, барлық клиенттер нақты уақыт режимінде дерекқордың бір данасын ортақ пайдаланады және жаңартуларды соңғы деректермен автоматты түрде алуы мүмкін.

Негізгі мүмкіндіктері 1.1-кестеде көрсетілген.

1.1-кесте – Firebase негізгі мүмкіндіктері

Нақты уақыт кезінде	Typical HTTP сұрауларының орнына, Firebase Realtime дерекқоры деректерді синхрондауды қолданады - деректер өзгерген сайын, кез келген қосылған құрылғы бұл жаңартуды миллисекунд ішінде алады. Желілік код туралы ойламай, бірлескен және терең тәжірибе жасаңыз.
Желіде емес	Firebase бағдарламалары тіпті дербес күйде қалады, себебі Firebase Realtime дерекқоры SDK сіздің деректеріңізді дискіге сақтайды. Қосылым қалпына келтірілгеннен кейін, клиенттік құрылғы сервердің ағымдағы күйімен
Клиент құрылғыларынан қол жетімді	Firebase Realtime дерекқорына тікелей мобильді құрылғыдан немесе веб-шолғыштан қол жеткізуге болады; Бағдарламалық серверге қажеттілік жоқ. Деректерді оқу немесе жазу кезінде орындалатын өрнектерге негізделген Firebase Realtime дерекқордың қауіпсіздік ережелері арқылы қауіпсіздік және деректерді тексеру қол жетімді.
Бірнеше дерекқорлар бойынша масштабтау	Blaze бағасының жоспарында Firebase Realtime дерекқорын пайдалану арқылы, деректерді бірегей Firebase жобасында бірнеше дерекқор даналарына деректерді бұзу арқылы деректер қолданбасының қажеттіліктерін кеңейте аласыз. Сіздің жобаңыз үшін Firebase түпнұсқалық растамасымен түпнұсқалықты оңтайландырыңыз және барлық дерекқор даналарындағы пайдаланушылардың шынайылығын тексеріңіз. Әрбір дерекқордағы деректерге қолжетімділікті бас

1.1-кестенің жалғасы

	қару Әрбір дерекқор данасы үшін пайдаланатын Firebase нақты уақыттық дерекқор ережелері арқылы.

Firebase Realtime дерекқоры дерекқорға тікелей клиенттік кодтан қауіпсіз кіруді қамтамасыз ету арқылы бай ынтымақтастық бағдарламаларын жасауға мүмкіндік береді. Деректер жергілікті түрде сақталады, тіпті желіден тыс, нақты уақыттағы оқиғалар соңғы пайдаланушыны жауапты тәжірибемен қамтамасыз етіп, өртеніп тұрады. Құрылғы қосылымды қалпына келтіргенде, нақты уақыт деректер базасы жергілікті деректердің өзгеруін клиент дербес болған кезде қақтығыстар автоматты түрде біріктірілетін қашықтағы жаңартулармен синхрондайды.

Нақты уақыттағы дерекқор деректеріңіздің құрылымы мен деректерді оқуға немесе жазуға болатындығын анықтау үшін Firebase-тің Real-Time деректер қорының қауіпсіздік ережелері деп аталатын икемді, білдіруге негізделген ережелер тілін қамтамасыз етеді. Firebase түпнұсқалық растамасымен біріктіру кезінде әзірлеушілер қандай деректерге және оларға кіруге болатыны туралы кімге қол жеткізе алатынын анықтай алады.

Нақты уақыттағы деректер базасы – бұл NoSQL дерекқоры және осылайша, реляциялық деректер базасымен салыстырғанда әр түрлі оңтайландырулар мен функционалдылықтар бар. Нақты уақыттағы деректер базасы API операцияларды жылдам орындау үшін арналған. Бұл реакцияны құрметтеместен миллиондаған қолданушыларға қызмет көрсете алатын керемет нақты уақыттағы мүмкіндіктерді жасауға мүмкіндік береді. Осыған байланысты пайдаланушыларға сіздің деректеріңізге қалай кіру керектігі туралы ойлау маңызды, содан кейін оны тиісті түрде құрастырыңыз.

2.4 Android Studio

Android Studio – Android OS платформасында қосымшаларды жасау үшін әзірлеушілерге қол жетімді болатын Google өндірісінің интеграцияланған ортасы. Android Studio Windows, Mac және Linux орнатуға болады. Google Play App Store-да app әзірлеушінің есептік жазбасы $25. Android Studio IntelliJ IDEA базасында құрылған.

IDE жүктеуге және тегін пайдалануға болады. Онда UI құру үшін макеттер бар, әдетте қосымшамен жұмыс басталады. Studio бағдарламасында смартфондар мен планшеттерге арналған шешімдерді әзірлеу құралдары, сондай-ақ Android TV,

Android Wear, Android Auto, Glass және қосымша контекстуалды модульдер үшін жаңа технологиялық шешімдер бар.

Android Studio ортасы Мобильді қосымшаларды әзірлеушілердің шағын командаларына (тіпті бір адам саны) немесе GIT немесе басқа ұқсас нұсқаларды басқару жүйелері бар ірі халықаралық ұйымдарға арналған. Тәжірибелі әзірлеушілер ауқымды жобалар үшін неғұрлым қолайлы құралдарды таңдай алады. Android шешімдері Java немесе C++пайдаланып Android Studio-да әзірленеді. Android Studio жұмыс процесінің негізінде бар проблемаларды бірден анықтауға мүмкіндік беретін үздіксіз интеграциялау концепті салынған. Кодты ұзақ тексеру әзірлеушілермен тиімді кері байланыс мүмкіндігін қамтамасыз етеді. Бұл опция мобильді қосымшаның нұсқасын Google Play App Store-да жылдам жариялауға мүмкіндік береді. Ол үшін LINT, Pro-Guard және App Signing құралдарын қолдау да бар.

Өнімділікті бағалау құралдарының көмегімен қолданбалы бағдарламалар пакеті бар файлдың күйі анықталады. Графиканы визуализациялау қолданба Google бағдарына сәйкес келе ме, 16 миллисекунд. Жадты визуализациялау құралы арқылы әзірлеуші оның қолданбасы тым көп жедел жадты пайдаланғанда және қашан "қоқыс жинау" болатынын біледі. Батареяларды талдау құралдары құрылғыға қандай жүктеме тиетінін көрсетеді.

Android Studio Google App Engine платформасымен жаңа API және мүмкіндіктер бұлтта жылдам интеграциялау үшін үйлесімді. Даму ортасында сіз Google Play, Android Pay және Health сияқты түрлі API таба аласыз. Android 1.6 бастап барлық Android платформаларын қолдау бар. Google Android нұсқасынан айтарлықтай ерекшеленетін Android нұсқалары бар. Олардың ең танымал – Amazon Fire OS. Android Studio осы ОЖ үшін ҚХА құруға болады. Android Studio қолдау онлайн форумдармен шектеледі.

2.5 Android Emulator

2016 жылдың сәуірінде Android Studio 2.0 нұсқасы жаңа эмулятормен шығарылды. Біртіндеп барлық материалдар жаңа нұсқамен ауыстырылады.

Жаңа нұсқада тезірек орнату үшін apk-файлды эмулятор терезесіне сүйреп апаруға болады. Сондай-ақ терезе өлшемін өзгерту, мультитач операциялары және тағы басқалар.

Жаңа эмулятормен жұмыс істеу үшін SDK құралдарын 25.1.1 немесе одан жоғары нұсқаға жаңартып, x86 жүйесінің суретімен жаңа виртуалды құрылғыны жасау керек. Құжаттар беті.

Android эмуляторы әзірлеуші үшін маңызды құрал болып табылады. Оның ерекшеліктерін зерттеу және оны дамудың бастапқы кезеңінде пайдалану қажет.

Дегенмен, эмулятор нақты құрылғының жалпы мінез-құлқын симуляциялау ғана есте сақтау керек. Сондықтан, соңғы тестілеу осы телефонда жүргізілуі тиіс.

Эмулятордағы экрандық пернетақтаның орнына тінтуірді жалаң және үстел үсті пернетақтасының орнына пайдалануға болады.

Аспапты тестілеу кезінде AVD артықшылығының бірі экранның ажыратымдылығы мен пикселдің тығыздығы үшін ерікті мәндерді орнату мүмкіндігі. Бұл сізді нарықтағы барлық құрылғыларды сатып алудан құтқарады. Дегенмен, елеулі компаниялар бұл жасайды.

Виртуалды құрылғы

Қолданбаны жасамас бұрын эмуляторы бар виртуалды құрылғыны жасау керек. Құрылғы өзі Eclipse-да жасалады. Бірақ көріністердің артында не болады? Windows 7 жүйесінде C: \ Users \ user_name \ .android \ avd қалтасында Android 2.1, 2.3, және т.б. үшін әрбір құрылғы түрі үшін бөлек қалталар жасалады. Linux жүйесінде /home/user_name/.android/ каталогын іздеңіз.

Бұл білім сізге Windows пайдаланушысының орыс тілінде қолданылған жағдайда көмектесуі мүмкін. Бұл жағдайда эмулятор іске қосылмайды және қате жібереді. Ini-файлды ашыңыз және виртуалды құрылғының жолын жазыңыз, сонда жолда орыс әріптері жоқ (тиісінше, * .avd файлы басқа жерге көшірілуге тиіс).

Genymotion. Бұл Ionic экожүйесінің бөлігі емес, бірақ назар аудару керек. Genymotion – бұл VirtualBox негізінде салынған жылдам Android эмулятор. Ол соншалықты жылдам, басқаша пайдалану мүмкін емес. Тек көріңіз.

Genymotion+Ionic қолданба іске қосу сәл өзгереді, орнына $ Ionic emulate android орындау керек $ ionic run android, өйткені бұл эмулятор виртуалды емес, нақты құрылғы ретінде анықталады.

Егер сізде Android Studio тұрса, онда сіз genymotion жобасын іске қосу оңай, диалогтық терезеде іске қосылған виртуалды машиналардың бірін таңдай аласыз. Мысалы, әр түрлі Android нұсқасы бар 4 виртуалды машинаны іске қосуға және олардың әрқайсысында өз қолданбасын тез ашуға болады – өте ыңғайлы. Сонымен қатар, виртуалды машиналарды (эмуляторларды) басқаруға мүмкіндік беретін Android Studio үшін Genymotion плагині бар.

Минустардан-барлық виртуалды машиналар бірдей тез жұмыс істемейді. Мысалы, Nexus 5 (Android 5.0) баяу жұмыс істейді, мысалы, Galaxy S3 (Android 4.1) найзағай жұмыс істейді. Genymotion Жеке пайдалану үшін тегін.

2.6 PhpStrom

PhpStorm – бұл қазіргі заманғы және классикалық жобалар үшін PHP 5.3-7.3 қолдайтын, кодты терең түсінетін интеллектуалды редакторы бар PHP

интеграцияланған даму ортасы, кодты автодотолыту, рефакторингтер, ұшу қателерін болдырмау және тілдерді араластыруды қолдайды.

Жүздеген инспекциялар жобаны әзірлеу кезінде толығымен талдай отырып, кодты верификациялау туралы қамқорлық жасайды. PHPDoc, code (re) arranger, код мәнері конфигурациясы бар код пішімі және басқа мүмкіндіктер жасаушыларға ұқыпты және оңай қол жетімді кодты жазуға көмектеседі.

HTML5, CSS, Sass, SASS, less, Stylus, Compass, CoffeeScript, TypeScript, ECMAScript Harmony, Jade, Zen Coding, Emmet, және, әрине, JavaScript.

PhpStorm WebStorm (HTML/CSS редактор, JavaScript редактор) бүкіл функционалдығын қамтиды және PHP және дерекқор / SQL толық функциялы қолдауын қосады.

Негізгі мүмкіндіктері:

– синтаксисті бөлектеу, кодты аяқтау, кеңейтілген код пішімдеу параметрлері, ұшақта қателердің алдын-алу мүмкіндігі бар интеллектуалды PHP код редакторы;

– PHP 5.3-7.3-ді, генераторларды, корутинді және барлық синтаксистік жақсартуларды қолдайды;

– PHP-нің реформаторлары, коды (қайта) ұйымдастырушысы, детектордың детекторы;

– Docker, Composer, кірістірілген REST клиенті, Пәрмен жолы құралдары, SSH консолі;

– негіздемелік қолдау (Symfony2, Yii үшін MVC көрінісі) және жетекші PHP шеңберіндегі мамандандырылған плагиндер (Laravel, Symfony, Magento, Drupal, Yii, CakePHP, WordPress, Joomla және көптеген басқа);

– PHP қосымшалары үшін көрнекі түзету құралы, кодты қамтуы бар PHPunit және Codeception (PHPunit 6-ға қолдау көрсету) көмегімен кодтау, сондай-ақ профайлмен біріктіру;

– HTML, CSS, JavaScript редакторы. JS үшін отладтау және сынау. HTML5, CSS, Sass, SCSS, Less, Stylus, Compass, CoffeeScript, TypeScript, ECMAScript Harmony, Emmet және басқа дамыған веб-технологияларды қолдайды;

– алдын-ала әзірлеу құралдарының жиынтығы;

– Code styles support, PSR1 / PSR2, Laravel, Symfony, Zend, Drupal және басқа кірістірілген стильдер;

– біріктірілген интерфейсті қоса, нұсқаларды басқару жүйелерімен біріктіру;

– FTP, SFTP, FTPS және т.б. арқылы қашықтан қолдануды қолдану және автоматты синхрондау;

– Live Edit: Код өзгерістерін браузерде бетті қайта жүктеместен бірден қарауға болады;

– PHP UML;

– қателерді бақылаушылармен біріктіру;

– дерекқор құралдарының, SQL редакторы;
– кросс-платформа (Windows, Mac OS X, Linux).

2.7 AngularJS

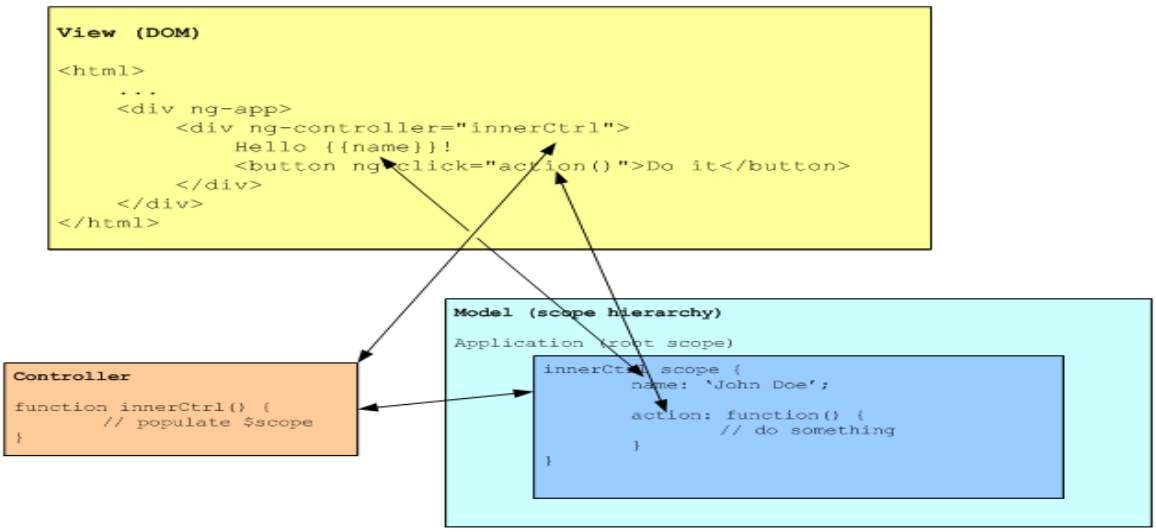
AngularJS – веб-қосымшаларды құру үшін тамаша фреймворк. Оның мысалдармен жабдықталған тамаша құжаттамасы бар. Оқыту "сынақ" қосымшаларында (TodoMVC Project сияқты) басқа жақтаулар арасында өзін лайықты көрсетеді. Ол тамаша презентациялар мен скринкастар бар.

Алайда, егер әзірлеуші бұрын Angular сияқты фреймворкалармен ешқашан кездеспесе және jQuery сияқты негізгі кітапханаларда жұмыс істесе, онда ол өзінің ойлау бейнесін өзгерту қиын болуы мүмкін.

Angular туралы түсіну керек бірінші нәрсе: бұл мүлдем басқа құрал. jQuery-кітапхана. AngularJS-бұл фреймворк. Сіздің кодыңыз кітапханамен жұмыс істегенде, ол қандай да бір функцияны шақыруды өзі шешеді. Фреймворк жағдайында сіз оқиғаны өңдеушілерді іске асырасыз және фреймворк оларды қандай сәтте шақыруын шешеді.

Бұл айырмашылықты орындау кезінде не болып жатқанын ойласақ түсіну оңай. JQuery рантайм кезінде не істейді? Іс жүзінде ештеңе . JQuery коды сіздің кодында болған нәрсе жауап ғана шақырылады-DOM оқиғасы туындаған функцияларды кез келген триггер жұмыс істеген кезде.

Angular жүктеу кезеңінде сіздің DOM Ағашын және angular-қолданбаға кодты айналдырады. HTML-белгілеу беттер angular-директиваларын және сүзгілермен онда компиляцияланады ағашқа шаблондарды, тиісті облыстың көріну (scope) және контроллерлер қосылады, оларға қажетті орындарда, ішкі цикл қосымша жұмысын қамтамасыз етеді дұрыс байланыстыруға арасындағы деректерді ұсына отырып, модель. Бұл MVC принциптерімен толық келісімдегі нақты жұмыс сызбасы, ол ұсыныстың, контроллердің және модельдің арасында өте таза бөлінуді қамтамасыз етеді. Егер жалпы оқиғалар циклі туралы айтатын болсақ, бетті суреттеу және деректерді байлау, онда сіз қажет болғанда ғана сіздің контроллерлеріңіздің кодын шақыра отырып, ол үздіксіз барлық уақытта орындалады деп санауға болады. AngularJS жұмысы істеу принципі 2.1-суретте көрсетілген.



2.1-сурет – AngularJS жұмысы істеу принципі

Модельдің әрбір жаңаруы кезінде (маңызды емес, асинхронды AJAX-сұрау арқылы немесе тікелей Контроллерден деректерді өзгерту арқылы) Angular деректерді байланыстыруды жаңартатын және барлық жүйені өзекті күйде сақтайтын $digest арнайы процедурасының циклын қайта іске қосады.

2.8 JavaScript

Қазіргі заманғы JavaScript – бұл жалпы мақсаттағы "қауіпсіз" бағдарламалау тілі. Ол жадымен, процессормен жұмыс істеудің төменгі деңгейлі құралдарын бермейді,себебі бастапқыда қажет емес браузерлерге бағытталған.

Басқа мүмкіндіктерге келетін болсақ, олар JavaScript іске қосылған қоршаған ортаға байланысты. JavaScript браузерінде бет манипуляциясына, келушімен өзара іс-қимылға және сервермен қандай да бір шамада барлық нәрселерді жасай алады:

– жаңа HTML тегтерін жасау, бар элементтерді жою, элементтердің стилін өзгерту, жасыру, элементтер мен т. б. көрсету;

– келушінің іс-әрекетіне жауап беру, тінтуірдің кликаларын өңдеу, курсорды жылжыту, пернетақтаға және т. б. Басу;

– серверге сұраныстарды жіберу және бетті қайта жүктеусіз деректерді жүктеу (бұл технология "AJAX"деп аталады);

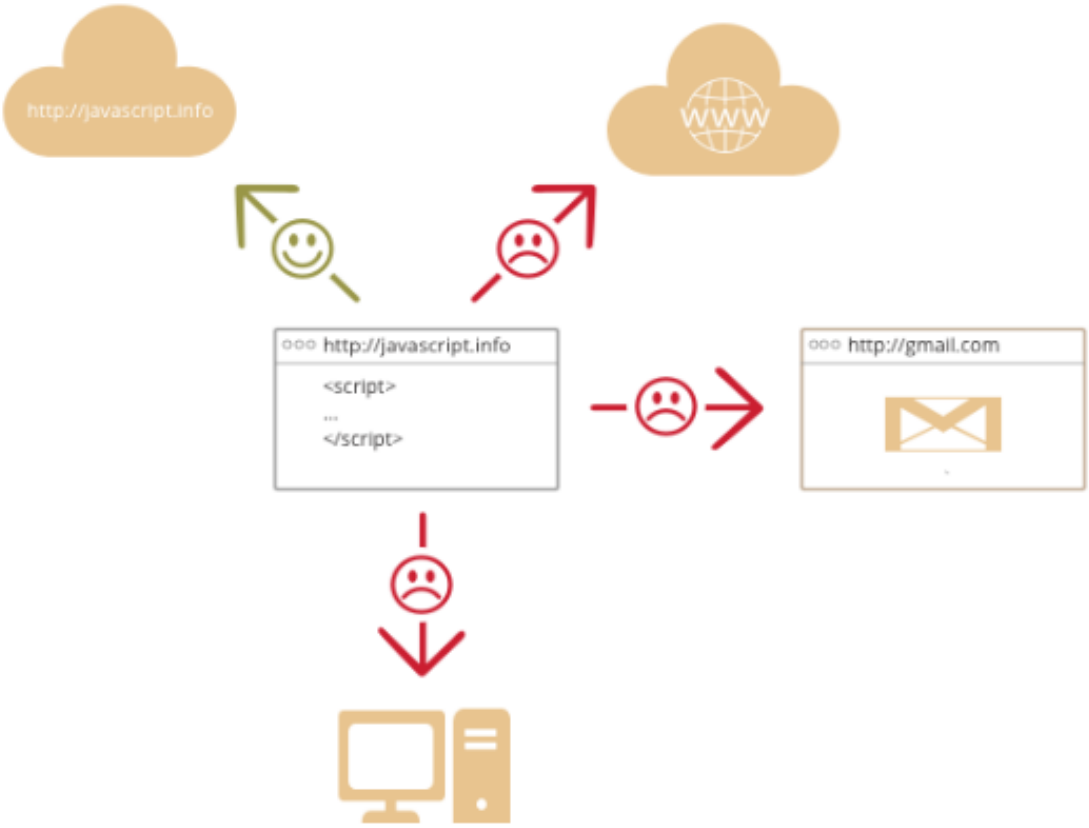
– cookie алу және орнату, деректерді сұрату, хабарларды шығару.

JavaScript – жылдам және қуатты тіл, бірақ браузер оны орындауға кейбір шектеулер қояды.

Бұл қолданушылардың қауіпсіздігі үшін жасалған, зиянкес JavaScript көмегімен жеке деректерді ала алмауы немесе қолданушының компьютеріне зиян келтірмеуі үшін.

Бұл шектеулер JavaScript браузерден тыс, мысалы, серверде қолданылады. Сонымен қатар, қазіргі заманғы браузерлер кеңейтілген мүмкіндіктерге ие плагиндер мен кеңейтулерді орнату бойынша өз механизмдерін ұсынады, бірақ пайдаланушыдан орнату бойынша арнайы әрекеттерді талап етеді.

Браузердегі JavaScript мүмкіндіктерінің көпшілігі ағымдағы терезе және бетпен шектелген. JavaScript жұмыс істеу принципі 2.2-суретте көрсетілген.



2.2-сурет – JavaScript жұмыс істеу принципі

JavaScript еркін файлдарды қатты дискіге оқуға/жазуға, оларды көшіруге немесе бағдарламаларды шақыруға болмайды. Ол операциялық жүйеге тікелей қол жеткізе алмайды.

Қазіргі браузерлер файлдармен жұмыс істей алады, бірақ бұл мүмкіндік арнайы бөлінген директориямен шектелген – "құмшы". Құрылғыларға қол жеткізу мүмкіндіктері қазіргі заманғы стандарттарда да өңделеді және кейбір браузерлерде ішінара қол жетімді.

Бір қойындыда жұмыс істейтін JavaScript, ол осы терезені немесе бір көзден бірнеше қойындыны (бірдей домен, порт, хаттама) ашқан жағдайды қоспағанда, басқа қойындылармен және терезелермен сөйлесе алмайды.

Бұл айналып өту жолдары бар және олар оқулықта ашылған, бірақ олар түрлі қойындыларда немесе терезелерде бар екі құжат үшін арнайы кодты талап етеді. Ол жоқ, қауіпсіздік мақсатында, JavaScript көмегімен бір қойындыдан екіншісіне ұшуға болмайды.

JavaScript-тен бет келген серверге сұраныстарды оңай жіберуге болады. Басқа доменге сұраныс болуы мүмкін, бірақ ыңғайлы, өйткені мұнда қауіпсіздік шектеулері бар.

2.9 HTML бағдарламалау тілінің құрылымы

HTML (HyperText Markup Language) – интернеттегі гипермәтін беттерін белгілеу стандартты тілі. Басқа гипермәтін белгілеу тілдері бар, бірақ интернет сайттарының көп бөлігі HTML тілінде белгіленген. Мұндай беттер адам үшін ыңғайлы түрде түрлі электрондық құрылғылардың экрандарында оларды көрсететін браузерлермен табысты түсіндіріледі.

HTML гипермәтін белгілеу тегтік тілі болып табылады: мәтінді гипермәтінге айналдыру үшін бөлгіштерді (дескрипторларды) пайдаланады. Тегтің мысалы: – бұл ашушы тег жабатын тег кездеспейінше мәтінді қалың қаріппен шығаруды қамтамасыз етеді.

WordPress арқылы сайтты жасауды бастап, сайт HTML тілінде жасалатынын білу жеткілікті, бірақ уақыт өте келе HTML тілін сәл меңгеруге тура келеді.

Мысал үшін жиі кездесетін жағдай: сіз Мәтін бетін жазасыз және іздеу жүйелері сізді плагиатормен ажыратпауы үшін дәйексөз ретінде осы фрагментті дұрыс белгілеп, оған бөтен мәтін фрагментін қосу туралы шешім қабылдадыңыз.

Режимнен мәтінді көзбен көріп, осы HTML тег терезесінде мәтіннің өзінен көп екенін анықтаңыз. Бұл мәтін көптеген сайттарға Түсіп, әр мәтінді толықтырып, күшейтіп, "тереңдетуге" тырысты, ал нәтижесінде өте қатал HTML-код болды.

HTML тіліндегі таңбаны қамтитын мәтіндік құжаттар (мұндай құжаттар дәстүрлі түрде кеңейтіледі .html немесе .htm), құжатты пішімделген түрде көрсететін арнайы қосымшалармен өңделеді. "Браузерлер" немесе "интернет шолғыштар" деп аталатын мұндай қосымшалар әдетте пайдаланушыға веб-беттерді сұрау, оларды қарау (және өзге сыртқы құрылғыларға шығару) және қажет болған жағдайда пайдаланушы енгізген деректерді серверге жіберу үшін ыңғайлы интерфейсті ұсынады. Бүгінгі таңда ең танымал браузерлер-Google Chrome, Mozilla Firefox, Opera, Internet Explorer және Safari.

HTML – құжаттарды белгілеудің тегтік тілі. HTML тіліндегі кез келген құжат элементтердің жиынтығы болып табылады және әр элементтің басы мен соңы арнайы тегтермен белгіленеді. Элементтер бос болуы мүмкін, яғни ешқандай мәтін және басқа деректер жоқ. Бұл жағдайда әдетте жабатын тег көрсетілмейді (мысалы,
 - жолды көшіру тегі-жеке және оны жабу қажет емес) . Сонымен қатар, элементтердің қандай да бір қасиеттерін анықтайтын атрибуттары болуы мүмкін (мысалы, href=" сілтеме). Атрибуттар ашушы тегте көрсетіледі. Мұнда HTML құжат үзінділерінің мысалдары:

– екі тег — ашу және жабу арасындағы Мәтін.< / strong>

– <a href="http://www.example.com" >мұнда элемент href атрибутын, яғни гиперсілтемені қамтиды.< / a>.

2.10 CSS стильдер тілінің негізі

CSS – HTML-құжаттарды көрсететін мәнерлер тілі. CSS қаріптермен, символдар мен фон түстерімен, өрістері бар, жолдары бар, биіктігі бар және бейнелеу элементтерінің ені бар, фондық бейнелері бар, элементтерді позициялаумен және көптеген басқалармен жұмыс істейді.

Егер HTML бет мазмұнын құрылымдау үшін қажет болса, онда CSS құрылымдық мазмұнды пішімдеу үшін қажет.

CSS қолдану сайттың барлық беттері бірыңғай стильде ұсталатынына сенімді болу үшін арнайы css-файлдарда сайт беттерінің жеке элементтерінің стилін қоюға мүмкіндік бере отырып, сапалы сайттар жасауды жеңілдетеді.

CSS түстерді, қаріптерді, жеке блоктарды және осы веб-беттердің сыртқы көрінісін көрсететін басқа да аспектілерді жасау үшін веб-беттерді жасаушылар қолданады. CSS әзірлеудің негізгі мақсаты веб-беттің логикалық құрылымын сипаттауды осы веб-беттің сыртқы түрін сипаттаудан (енді CSS формальды тілінің көмегімен жасалады) бөлу болып табылады. Мұндай бөлу құжаттың қол жетімділігін арттырып, оны ұсынудың үлкен икемділігі мен басқаруына мүмкіндік береді, сондай-ақ құрылымдық мазмұнда күрделілік пен қайталанушылықты азайтуы мүмкін. Сонымен қатар, CSS экрандық көрініс, баспа көрінісі, дауыспен оқу (арнайы дауыстық браузер немесе экраннан оқу бағдарламасы) немесе Брайль шрифтін пайдаланатын құрылғылар шығарған кезде түрлі стильдерде немесе шығару әдістерінде бір құжатты ұсынуға мүмкіндік береді.

CSS ережелері CSS ресми тілінде жазылады. Ережелер сыртқы көрінісі сипатталатын веб-құжаттың өзінде де, CSS пішімі бар сыртқы файлдарда да орналасуы мүмкін. CSS пішімі-бұл мәтіндік файл, онда CSS ережелері мен оларға түсініктеме тізімі бар.

CSS стильдері олар сипаттаған веб-құжатқа <head>элементіне қосылған <link> элементі арқылы қосылуы мүмкін:

```
<html>
<head>
<link rel="stylesheet" href="style.css">
</head>
<body>
</body>
</html>
```

3 Деректер базасының арасындағы байланыстар

3.1 ER диаграммасы

Ақпараттық жүйе (АЖ)-қандай да бір пәндік сала туралы ақпаратты сақтауға және өңдеуге арналған бағдарламалық-аппараттық кешен.

АЖ құру процесі бірқатар кезеңдерге бөлінеді. Әдетте АЖ құрудың келесі кезеңдерін белгілейді:

– жүйеге қойылатын талаптарды қалыптастыру (талдау),
– жобалау,
– іске асыру,
– тестілеу,
– іске қосу,
– пайдалану және сүйемелдеу.

Кез келген ақпараттық жүйенің маңызды компоненті деректер қоры (ДБ) болып табылады. Деректер қоры (DataBase) – белгілі бір таңдалған модельге сәйкес құрылымдалған, ұйымдастырылған деректер жиынтығы, бірлестіктер және қандай да бір физикалық немесе виртуалды жүйенің сипаттамасын сипаттайтын.

Дәл осы ДБ АЖ-ны пайдалануға, оған ағымдағы қызмет көрсетуді орындауға, кәсіпорынды (ұйымды) жаңғырту немесе ақпараттық ағындарды, заңнаманы және кәсіпорынның (ұйымның) есептілік нысандарын өзгерту кезінде оны түрлендіруге және дамытуға мүмкіндік береді.

Қазіргі заманғы методологияға сәйкес, АЖ құру процесі АЖ өмірлік циклінің барлық кезеңдерінде (ӨЦ) келісілген модельдердің қатарын құру және дәйекті қайта құру процесі болып табылады. Әрбір кезеңде ӨЦ үлгілері құрылады: ұйымдар, АЖ талаптары, АЖ жобасы, қосымшаларға қойылатын талаптар және тағы басқа.

АЖ жобалау үш негізгі саланы қамтиды:

– деректер базасында іске асырылатын деректер нысандарын жобалау (деректер үлгілерін жасау);

– деректерге сұрау салуды орындауды қамтамасыз ететін бағдарламаларды, экрандық нысандарды, есептерді жобалау;

– нақты ортаны немесе технологияны есепке алу, атап айтқанда: желі топологиясы, аппараттық құралдардың конфигурациясы, пайдаланылатын сәулет (файл-сервер немесе клиент-сервер), параллель өңдеу, деректерді үлестірілген өңдеу және т. б.

Модель – жүйенің және оның компоненттерінің көрінісі (бейнесі) болып табылатын жасанды объект.

Деректер моделі (Data Model) – ұйымның қызметін басқару және бағалау үшін оның миссиясына, функцияларына, мақсаттарына, стратегияларына қол

жеткізу мақсатында қажетті деректерді анықтайтын талдаудың графикалық немесе мәтіндік көрінісі. Деректер моделі мәнін, домендерді (атрибуттарды) және басқа деректермен байланысты анықтайды, сондай-ақ деректер мен деректер арасындағы байланыстың концептуалды көрінісін ұсынады.

Деректер моделін құру мақсаты АЖ әзірлеушісін бір модель немесе кез келген деректер базасына салыстырмалы түрде оңай біріктірілуі мүмкін бірнеше жергілікті модель нысанында деректер базасының тұжырымдамалық схемасымен қамтамасыз етуден тұрады.

Деректер модельдерін жасау кезінде семантикалық модельдеу әдісі қолданылады. Семантикалық модельдеу құрылымдық компоненттердің мәніне немесе деректердің сипаттамаларына негізделеді, бұл олардың дұрыс түсіндірілуіне (түсінуіне, түсіндірілуіне) ықпал етеді. Құралы ретінде семантикалық модельдеу пайдаланылады әр түрлі нұсқалары диаграммалар «мәні-байланыс» (ER – Entity-Relationship) – ERD.

Бірақ барлық диаграммалар мәні-байланыс бір идеядан тұрады-сурет әрдайым мәтіндік сипаттамадан көрнекі. ER-диаграммалар пәндік аймақ мәндерінің графикалық бейнесін, олардың қасиеттерін (атрибуттарын) және мәндер арасындағы өзара байланысты қолданады.

Мәні (кесте, қатынас) – бұл бір топқа бөлуге болатын нақты немесе абстрактылы объектілердің (адамдар, заттар, орындар, оқиғалар, идеялар, комбинациялар және т.б.) жиынтығын ұсыну, өйткені олар бірдей сипаттамаларға ие және ұқсас байланыстарға қатыса алады. Әрбір мәнде сингулярлық зат есімімен көрсетілген атау болуы тиіс. Модельдегі әрбір мән атауы бар тіктөртбұрыш түрінде бейнеленеді.

Жалпы атрибуттары немесе сипаттамалары бар көптеген нақты немесе абстрактылы заттар (адамдар, объектілер, оқиғалар, идеялар және т.б.) деп айтуға болады.

Нысан данасы (жазба, кортеж) – бұл нысанның нақты өкілі.

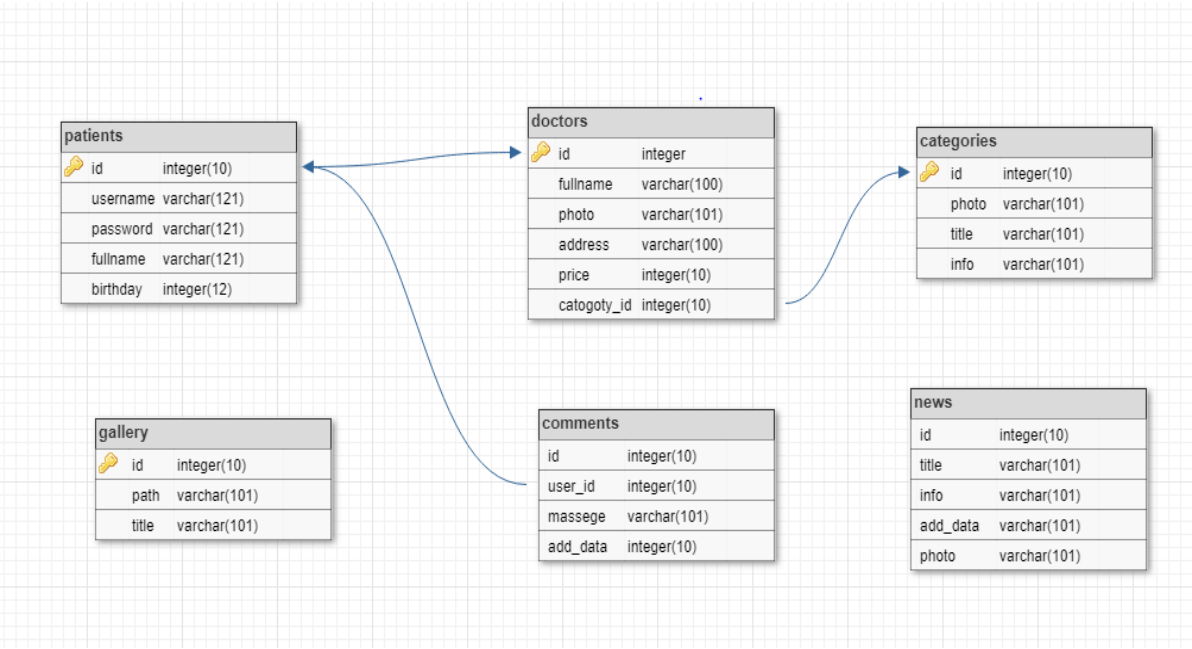
Нысан атрибуты (өріс, домен) – бұл нысанның кейбір қасиеттері болып табылатын атаулы сипаттама.

Байланыс – бұл екі нысан арасындағы кейбір қауымдастық. Бір мән басқа нысанмен немесе өзімен байланысты болуы мүмкін. Байланыс бір мағынада онымен байланысты басқа мәндерді табуға мүмкіндік береді.

Әрбір байланыс келесі байланыс түрлерінің бірі болуы мүмкін:

Бірдің-бірге, Көптің-көпке, бірдің-көпке.

Бірдің-бірге типті байланыс бірінші нысанның бір данасы (сол жақ) екінші нысанның бір данасымен (оң жақ) байланысты екенін білдіреді. Бір-біріне байланыс көбінесе біздің шын мәнінде екіге дұрыс бөлінбеген бір ғана мәні бар екенін көрсетеді.



3.1-сурет – Server-дегі кестелердің ER диаграммасы

Байланыс типіндегі көптің-көпке білдіреді әрбір данасы бірінші мәніне байланысты болуы мүмкін бірнеше данасын екінші мәні, және әрбір данасы екінші мәніне байланысты болуы мүмкін бірнеше данасын бірінші мәні. Байланыс түрі көп-көп адамдар модельді әзірлеудің ерте кезеңдерінде рұқсат етілген байланыстың уақытша түрі болып табылады.

Бірдің-көпке типті байланыс бірінші нысанның бір данасы (сол жақ) екінші нысанның бірнеше данасымен (оң жақ) байланысты екенін білдіреді. Бұл ең жиі қолданылатын байланыс түрі.

ER-модельдерді әзірлеу кезінде пәндік облысты (ұйымды, кәсіпорынды) тексеру және:

– ұйымда (кәсіпорында) деректер сақталатын мәндер, мысалы, адамдар, орындар, идеялар, оқиғалар және т. б. (блоктар түрінде ұсынылады);

– осы нысандар арасындағы байланыстар (осы блоктарды қосатын желілер түрінде ұсынылады);

– осы мәндердің қасиеттері (осы блоктардағы атрибуттар атаулары түрінде ұсынылады).

3.2 Класстар диаграммасы

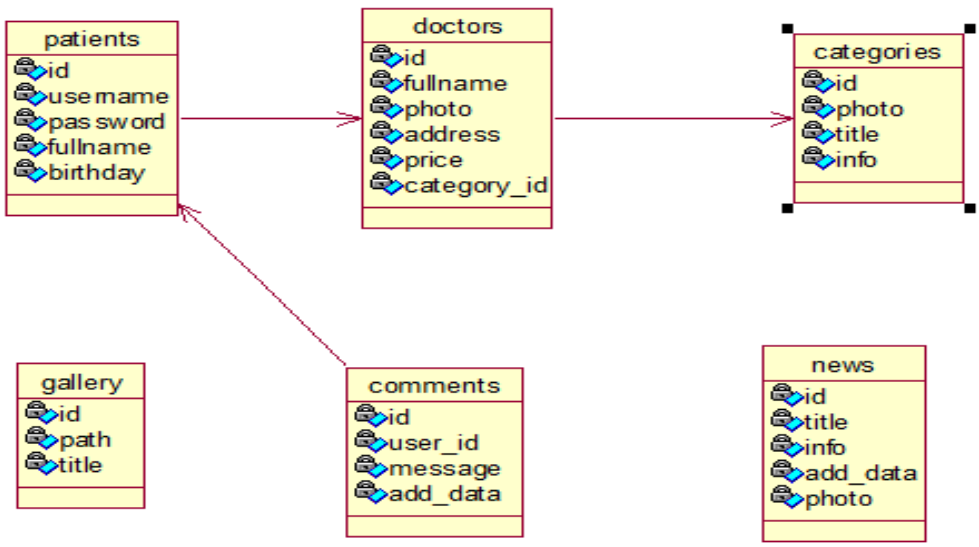
Класстар диаграммасы жүйенің, класстың түрлерін және олардың арасында бар статикалық қатынастардың әртүрлі түрлерін анықтайды. Класс

диаграммалары класстың атрибуттарын, класс әрекеттерін және класстар арасындағы қатынастарға қолданылатын шектеулерді көрсетеді. Класс диаграммасының түрі мен түсіндірілуі, көзқарасқа (абстракция деңгейіне) байланысты: класстар доменнің субъектілерін (талдау процесінде) немесе бағдарламалық жасақтама жүйесінің элементтерін (жобалау және іске асыру процестерінде) ұсына алады.

Негізгі элементтер класстар мен олардың арасындағы байланыстар болып табылады. Класстар атрибуттар мен операциялардың көмегімен сипатталады.

Атрибуттар класс объектілерінің қасиеттерін сипаттайды. Класстағы объектілердің көпшілігі олардың атрибуттарындағы айырмашылықтарға және басқа объектілермен өзара байланысына байланысты өз даралығын алады. Алайда, атрибуттар мен өзара байланыстардың бірдей мәндері бар объектілер болуы мүмкін. Яғни, объектілердің даралығы олардың қасиеттеріндегі айырмашылықтар емес, олардың өмір сүру фактісімен анықталады. Атрибуттың аты класс шегінде бірегей болуы керек. Атрибуттың атауында оның түрі мен әдепкі мән болуы мүмкін.

Негізгі элементтер класстар мен олардың арасындағы байланыстар болып табылады. Класстар атрибуттар мен операциялардың көмегімен сипатталады. Класстар диаграммасы 3.2-суретте көрсетілген.



3.2-сурет – Server-дегі кестелердің каласс диаграммасы

Негізгі элементтер класстар мен олардың арасындағы байланыстар болып табылады. Класстар атрибуттар мен операциялардың көмегімен сипатталады.

Атрибуттар класс объектілерінің қасиеттерін сипаттайды. Класстағы объектілердің көпшілігі олардың атрибуттарындағы айырмашылықтарға және басқа объектілермен өзара байланысына байланысты өз даралығын алады. Алайда, атрибуттар мен өзара байланыстардың бірдей мәндері бар объектілер болуы мүмкін. Яғни, объектілердің даралығы олардың қасиеттеріндегі айырмашылықтар емес, олардың өмір сүру фактісімен анықталады. Атрибуттың аты класс шегінде бірегей болуы керек. Атрибуттың атауында оның түрі мен әдепкі мән болуы мүмкін.

Қауымдастық (association) –класс даналарының арасындағы қарым-қатынас.

Қауымдастықтың әр соңы жиілікке ие (синоним-қуаты, ориг. - multiplicity), ол ассоциацияның тиісті соңынан орналасқан қанша нысан осы қатынаста қатыса алатынын көрсетеді. Жалпы жағдайда еселік кез келген жиынмен берілуі мүмкін.

Қауымдастықтарға атау берілуі мүмкін. Ат ретінде әдетте байланыстың мәні мен мақсатын көрсететін етістік немесе етістік сөз тіркесі таңдалады. Сонымен қатар, қауымдастықтың соңында рольдің аты, яғни қауымдастықтың осы шетіндегі объектілер қандай рөл атқарады.

Агрегация (aggregation) – бұл "бүтін бөлік"типті қауымдастық. UML-дегі Агрегация соңында ромбпен тура түрінде ұсынылады.

Композиция (composition) – бұл объектілер-бөліктер өздігінен өмір сүре алмайтын және агрегациялық кластың объектісін жою кезінде жойылатын агрегация. Композиция ассоциация ретінде бейнеленген, тек ромбик боялған.

Агрегация мен композиция арасындағы айырмашылықты түсіну маңызды: агрегация кезінде объектілер-бөліктер өздері өмір сүре алады, ал композиция кезінде – жоқ. Агрегация үлгісі: автомобиль-дөңгелек, композиция үлгісі: үй-бөлме.

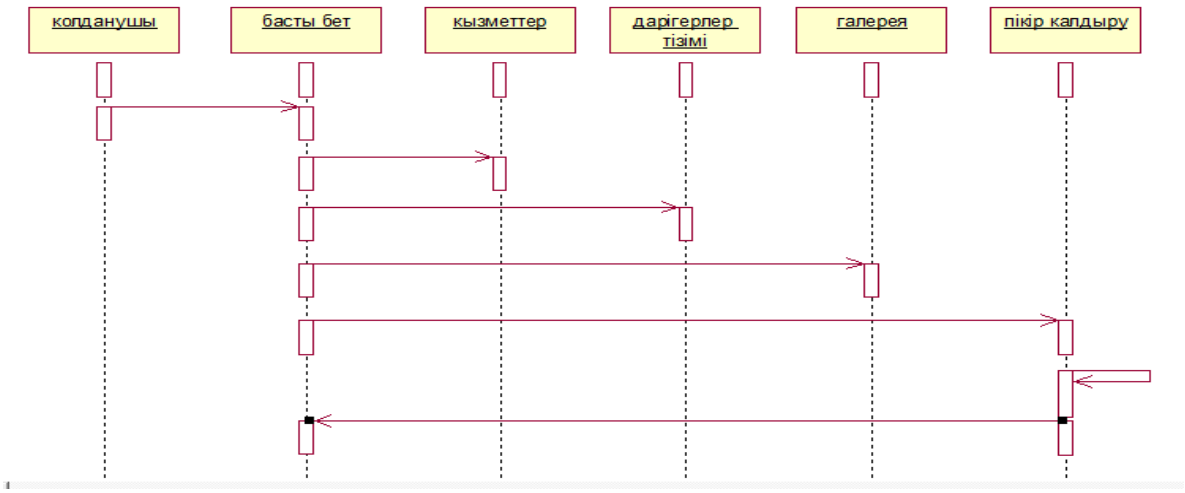
Мұрагерлік (inheritance) – бұл "жалпы-жеке"типті қатынас. Бір сынып басқа сыныптардың мінез-құлқы мен құрылымына ие болған сыныптар арасындағы осындай қарым-қатынасты анықтауға мүмкіндік береді. Базалық (бір немесе бірнеше) негізінде туынды класты құру кезінде мұрагерлік иерархиясы пайда болады. Мұрагерлік принциптерін іске асыру кодты қайта пайдалану мүмкіндігінің негізгі алғышарты болып табылады, өйткені бұл полиморфизмге жетудің негізгі құралы.

3.3 Тізбек және Күй диаграммалары

АЖ жобалау барысында талдаушы жалпы тұжырымдамадан оның логикалық құрылымын физикалық іске асыруды сипаттайтын неғұрлым егжей-тегжейлі модельдерге түсіну арқылы кезең-кезеңімен түседі.

Прецеденттер диаграммасының (пайдалану нұсқаларының) көмегімен жүйенің негізгі пайдаланушылары және осы жүйе шешуі тиіс міндеттер анықталады. Қызмет диаграммасының көмегімен біз қойылған мақсатқа жету үшін қажетті әрбір прецедент үшін іс-қимыл реттілігін сипаттаймыз.

Осылайша, талдаушылар белгілі бір нәтижеге қол жеткізу үшін қажетті бір немесе бірнеше прецеденттер шеңберіндегі іс-қимыл алгоритмі, сондай-ақ келтірілген іс-әрекеттерді орындау барысында объектілердің жай-күйін өзгерту сияқты мінез-құлық аспектілерін тіркейді. Server-дегі кестелердің тізбек диаграммасы 3.3-суретте көрсетілген.



3.3-сурет – Server-дегі кестелердің тізбек диаграммасы

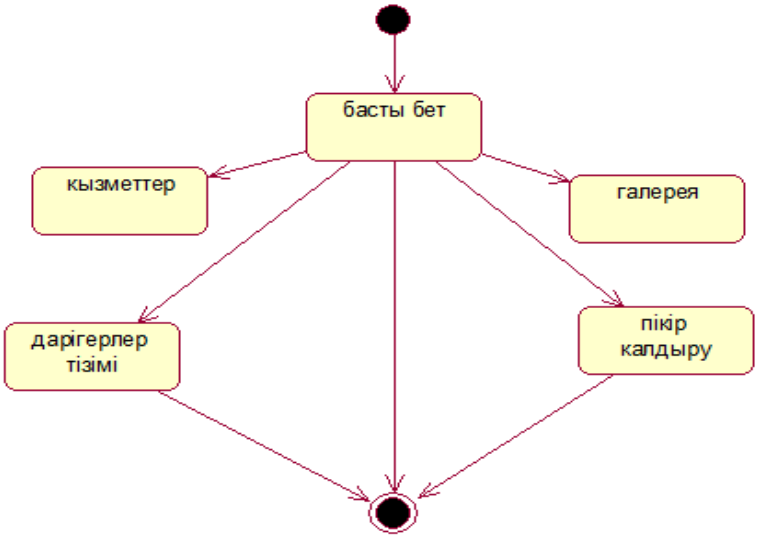
Жүйе жұмысының реттілігін (алгоритмін) ғана көрсететін қызмет диаграммасынан айырмашылығы, өзара іс-қимыл диаграммалары әзірлеушілердің назарын объектінің (класстың) белгілі бір операцияларын шақыруға бастамашылық ететін немесе операцияны орындау нәтижесі болып табылатын хабарламаларға аударады.

Тізбек диаграммасы өзара әрекеттесу диаграммаларының әр түрлілігінің бірі болып табылады және жүйе нысандарының өзара әрекеттесуін уақытында модельдеуге, сондай-ақ олардың арасында хабар алмасуға арналған.

UDP негізгі принциптерінің бірі бір-бірінен хабарламаларды жіберу және алу арқылы көрінетін жүйе элементтері арасындағы ақпараттық алмасу тәсілі болып табылады. Осылайша, тізбек диаграммасының негізгі ұғымдары Объект және хабар ұғымымен байланысты.

Күй диаграммасы кейбір автоматты ұсынатын арнайы түрдің графы болып табылады. UML контекстіндегі автомат ұғымы автоматтар теориясына негізделген өте ерекше семантикаға ие. Бұл бағанның ұштары автомат элементтерінің күйі мен басқа да түрлері (жалған күйлер) болып табылады, олар тиісті графикалық

символдармен бейнеленеді. Доғалар баған күйден күйге ауысуларды белгілеу үшін қызмет етеді. Күй диаграммалары модельдің жеке элементтерін егжей-тегжейлі көрсететін ішкі диаграммаларды жасай отырып, бір-біріне салынуы мүмкін. Жай-күйдің нақты диаграммасының семантикасын түсіну үшін модельдеуші мәннің мінез-құлқының ерекшеліктерін ғана емес, сонымен қатар автоматтар теориясы бойынша жалпы мәліметтерді білу қажет. Күй диаграммасы 3.4-суретте көрсетілген.



3.4-сурет – Server-дегі кестелердің күй диаграммасы

Күй диаграммасының негізгі элементтері "күй" және "өту"болып табылады. Жай-күй диаграммасы қызмет диаграммасымен ұқсас семантикасы бар, тек мұнда қызмет жай-күймен ауыстырылды, өтпелер іс-қимылды бейнелейді. Осылайша, егер қызмет диаграммасы үшін "қызмет" және "әрекет" ұғымдары арасындағы айырмашылық одан әрі декомпозицияның мүмкіндіктеріне негізделген болса, онда күй диаграммасында қызмет объект ұзақ уақыт болатын жағдайды, ал әрекет дереу бейнелейді.

3.4 Мобильді қосымшаның терезелерін сипаттау

Қала тұрғындарының уақытын жоғалтпай "Online Hostital" қосымшасы арқылы өзіне керекті аурухана және ондағы дәрігерлер жайлы ақпаратты ала алды. Өз кезегінде қолданушылар жобаға тіркелу, дәрігерлерге жазылу және т.б секілді

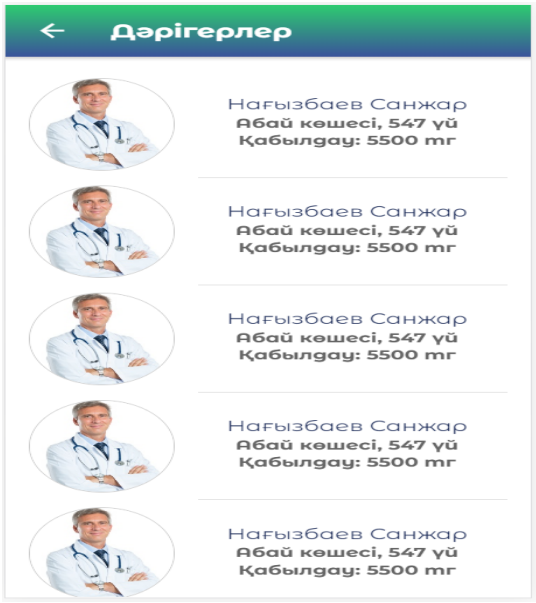
әрекеттерді орындай алады. Төмендегі суреттен жобаның басты бетін көре аламыз (3.5-сурет).

3.5-сурет – Қосымшаның басты беті

Төменде келтірілген суретте "Қызметтер" терезесінің пішімі көрсетілген (3.6-сурет). Бұл терезеден қосымшада көрсетілген дәрігерлер тізмдерін көре аламыз.

3.6-сурет – Қызметтер терезесі

Төмендегі суретте біз таңдаған мамандықтағы дәрігерлердің тізімдері көрсетілген (3.7-сурет).



3.7-сурет – Дәрігерлер терезесі

Суреттегі "Дәрігірлер жайлы ақпарат" терезесінен дәрігерлердің көрсететін қызметтері жайлы және сол қызметтің дәрігер ұсынған бағасы жайлы ақпарат ала аламыз. Сонымен қатар таңдаған дәрігерімізге жазылу уақытын таңдай аламыз.

3.8-сурет – Дәрігерлер жайлы ақпарат терезесі

ҚОРЫТЫНДЫ

Дипломдық жұмыстың қорытындысын шығара келсек. Ionic Cordova ортасында жасалынып, және де Java объектіге бағытталған бағдарламалау тілін қолданылды Бұл проектінің жасалу барысында қолданушыға деген барлық қолайлылық көзделуде. Басқа да ұқсас қосымшаларға қарағанда артықшылықтарды арттыру басты мақсаттың бірі.

«Online Hospital» мобильді қосымшаны жасауға қажетті технологияларға талдау жасалды. Мобильдік жаңалық порталының жалпы сипаттамасы құрастырылды. Жұмысты орындауға Android Studio бағдарламалау ортасының негізі, онын негізгі мақсаты мен мобильді қосымшалар жасау мүмкіншіліктері қолданылды. Сонымен қатар Java бағдарламалау тілі, ондағы класстар, қасиеттер ашылып жазылды. Мобильді құрылғыларға қосымшалар құруды жоспарлаудың теориялық негіздері айқындалды, мобильді құрылғылар платформалары мен қосымшаларды құру жабдықтары талданып, Android операциялық жүйесінің артықшылығы мен кемшілігі келтірілді. Android архитектурасына түсініктеме берілді. Android бағдарламасының жалпы жұмыс істеу сұлбасы келтірілді. Android Studio бағдарламалау ортасы, Google Play, Java құрылымы және өзгешеліктері, XML тілінің ерекшеліктері берілді. Android SDK құралы көмегімен мобильді қосымшалар жасау және оның жұмыс істеу принциптері зерттелді. Ең бастысы мобилді қосымша толықтай қазақ тілінде жұмыс жасайды.

Қосымшаның Android Studio платформасында жүзеге асырылу қадамдары сипатталды. Қойылған мәселелердің толық шешімі бағаланды. Алға қойылған мақсат орындалды.

Жұмыстың нәтижелерін нақты қолдану бойынша ұсыныстар мен шығыс мәліметтері зерттелді.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1. Голощапов А. Google Android: программирование для мобильных устройств. – СПб.: БХВ-Петербург, 2010. – 448 с.
2. Коматинэни С., Маклин Д., Хэшими С. Google Android: программирование для мобильных устройств = Pro Android 2. – 1-е изд. – СПб.: Питер, 2011. – 736 с.
3. Сатия Коматинени, Дэйв Маклин. Android 4 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов = Pro Android 4. – М.: Вильямс. – 880 с.
4. Роджерс Р., Ломбардо Д. Android. Разработка приложений. – М.: ЭКОМ Паблишерз, 2010. – 400 с.
5. Донн Фелкер. Android: разработка приложений для чайников = Android Application Development For Dummies. – М.: Диалектика, 2011. – 336 с.
6. Автоматизированные системы управления предприятием класса ERP: идеи, решения, проблемы // Компьютерные вести. – 2009. – 44с.
7. Э. Фримен, Э. Фримен. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. – П.: «Питер», 2010. – 656 с.
8. Эд Титтел, Джефф Ноубл. HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. – М.: «Диалектика», 2011. – 400 с.
9. Питер Лабберс, Брайан Олберс, Фрэнк Салим. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений = Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development. – М.: «Вильямс», 2011. – 272 с.
10. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования = Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development. – 3-е изд. – М.: Вильямс, 2006. – 736 с.
11. Джозеф Шмуллер. Освой самостоятельно UML 2 за 24 часа. Практическое руководство = Sams Teach Yourself UML in 24 Hours, Complete Starter Kit. – М.: Вильямс, 2005. – 416 с.

А Қосымшасы
(міндетті)

Техникалық тапсырма

А.1 Жүйенің мақсаты

Бұл дипломдық жобада алға қойылатын техникалық тапсырма клиникаға онлайн түрдегі мобильді қосымша жасау және оны қолданысқа беру болып табылады, қызмет көрсету орталықтары мен қарапайым халық ортасында байланыс орнату.

Бірінші кезде мобильді қосымшаны жасауға пайдаланылатын бағдарламалық қосымшалар: HTML, TypeScript, JS, CSS тілдерінде жасалу тиіс болды. Және сол тілдердегі көмекші бағдарламалар арқылы мобильді қосымшаны жасап шығару.

Екінші этапта жалпы мобильді қосымшаның базамен байланысын орнату, ол қалай жасалады, құрылымы қандай болады деген өзекті жұмыстарды жасау. Әр қызмет көрсету орталықтарының жеке беттерін жасау, оның ішінде бір бетті мысал ретінде қарастыру. Мобильді қосымшаның нәтижелерін скриншоттар арқылы көрсету.

Үшінші этапта сервистермен жұмыс жасау.Толығырақ ашып жазу. Мобильді қосымшаның қабілеттілігін арттыру, сонымен қатар тұтынушылардың сұраныстарын қанағаттандыру мақсатында, қосымшаның қазіргі заманға сай автоматтандырылған ақпараттық жүйесін жасап шығару болып табылады.

Мобильді қосымша келесідей пунктерден тұруы қажет:

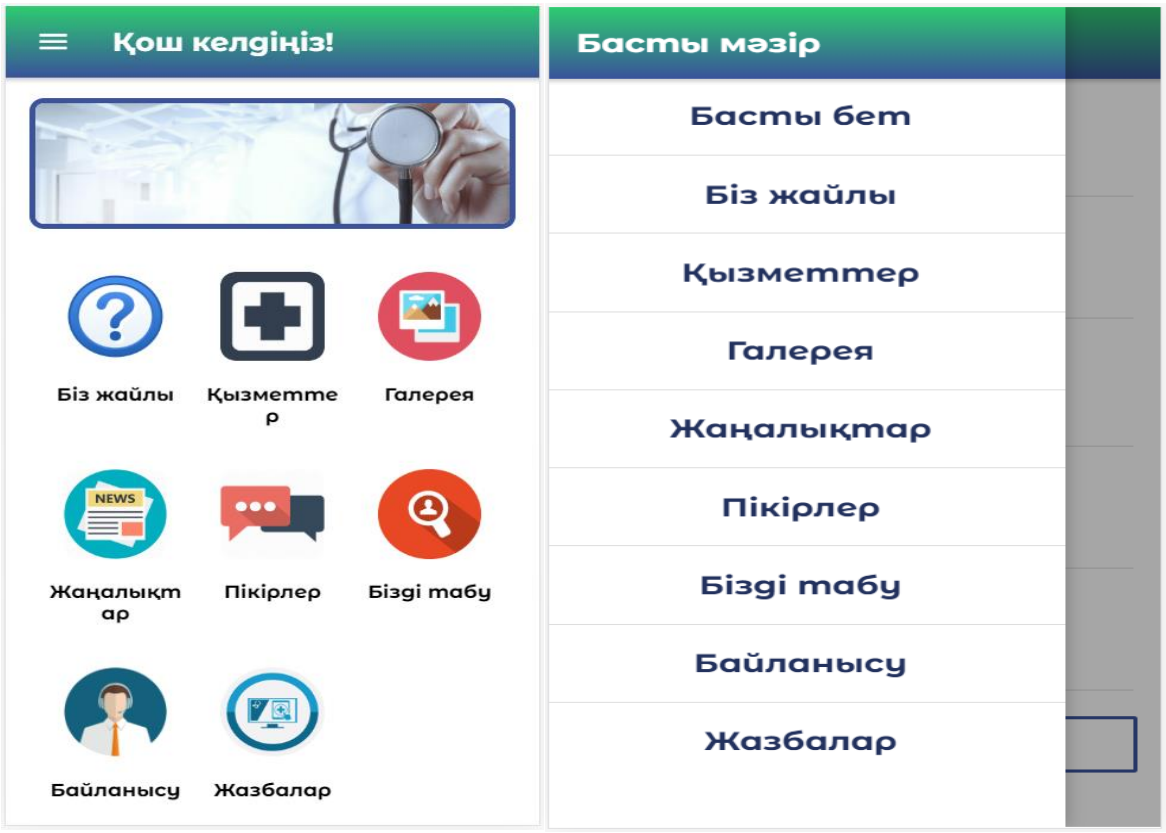
– Тіркелу;
– басты мәзір;
– жоба жайлы;
– қызметтер;
– дәрігерлер;
– дәрігерлер жайлы ақпарат;
– галерея;
– жаңалықтар;
– пікірлер;
– бізді табу;
– байланысу.

А.2 Қолданушылық модулі

Программалық жабдықтама үшін қолданушыға қойылатын басты талап:
– смартфон Android операциялық жүйесінде жұмыс жасау тиіс;
– ОЖ Android 4.4 (Kitkat) төмен болмауы қажет.

Жалпы мобильді қосымшада мағлұматтар енгізудің ерекше белгісі ретінде тез, әрі жылдам мағлұматтар қосу жұмысын автоматтандыру. Осылайша, қосымшаға мағлұматтар енгізу өте ыңғайлы қылып жасалуын қамтамасыз ету. Мұндай шешімдерді қабылдау заманауи ақпараттық технологияларды қолданбай жету мүмкін емес. Алайда, ҚР ақпараттық технологиялар нарығы бай емес болғандықтан, менің мобильді қосышам ҚР нарығында өз орнын жылдам тауып алады деген сеніммен қосымша жасап шығару.

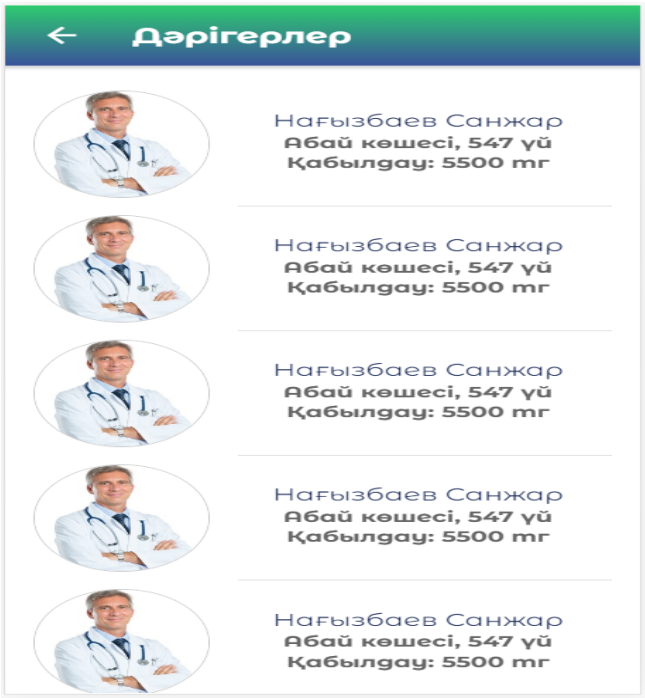
Біздің жасаған мобильді қосымша төмендегідей парақшалардан тұрады.



А.1-сурет – Қосымшаның басты беті

А Қосымшасының жалғасы

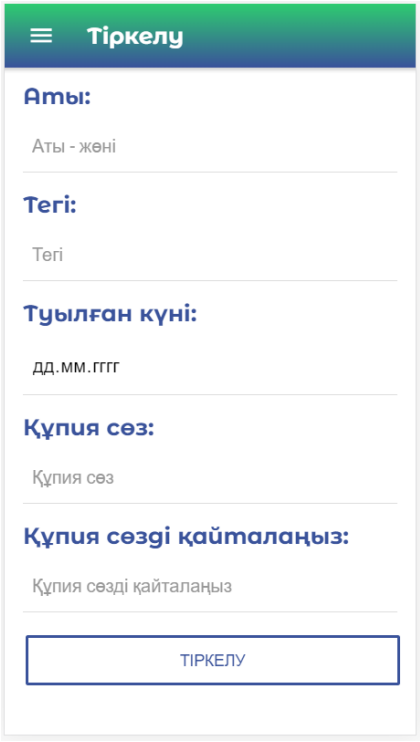
А.2-сурет – Қызметтер терезесі



А.3-сурет – Дәрігерлер терезесі

А Қосымшасының жалғасы

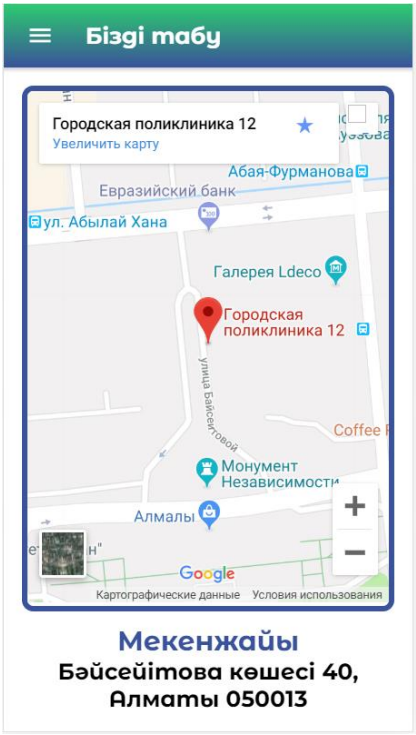
А.4-сурет – Дәрігерлер жайлы ақпарат терезесі



А.5-сурет – Тіркелу беті

А Қосымшасының жалғасы

А.6-сурет – Галерея терезесі



А.7-сурет – Бізді табу терезесі

Б Қосымшасы
(міндетті)

Бағдарлама мәтіні

```
//signin.html авторизация беті
<ion-header>
  <ion-navbar color="hospital">
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Авторизация</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <h3>Қолданушы аты:</h3>
  <ion-list>
    <ion-item>
      <ion-input [(ngModel)]="username" type="text" placeholder="Қолданушы
аты:"></ion-input>
    </ion-item>
  </ion-list>
  <h3>Құпия сөз:</h3>
  <ion-list>
    <ion-item>
      <ion-input [(ngModel)]="password" type="password" placeholder="Құпия
сөз:"></ion-input>
    </ion-item>
  </ion-list>
  <button ion-button outline class="recording-btn" (click)="auth()">Жүйеге
кіру</button>
  <button ion-button outline class="recording-btn" (click)="reg()">Тіркелу</button>
</ion-content>

//
// signin.ts авторизация беті
import { Component } from '@angular/core';
import { NavController, NavParams } from 'ionic-angular';
import {RegistrationPage} from "../registration/registration";
import {ServiceProvider} from "../../providers/service/service";
```

Б Қосымшасының жалғасы

```
import {AngularFireDatabase} from "angularfire2/database-deprecated";
import {HomePage} from "../home/home";
@Component({
  selector: 'page-authorization',
  templateUrl: 'authorization.html',
})
export class AuthorizationPage {
  username = '';
  password = '';
  constructor(public navCtrl: NavController, public navParams: NavParams, private
service: ServiceProvider) {
  }
  auth() {
    this.service.list('/patients').subscribe((res: any) => {
      let result = res.filter((patient: any) => {
        return patient.username == this.username && patient.password == this.password
      });
      if (result.length) {
        this.service.toast('Сәтті авторизация!');
        localStorage.setItem('user', JSON.stringify(result[0]));
        this.navCtrl.setRoot(HomePage);
      }else{
        this.service.toast('Қолданушы атауы немесе құпия сөз дұрыс емес. Тағы
қайталап көріңіз!');
      }
    });
  }
  reg() {
    this.navCtrl.push(RegistrationPage);
  }
}

//
//comments.html пікірлер беті
<ion-header>
  <ion-navbar>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
```

```
    <ion-title>Пікірлер</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <ion-list>
    <ion-item>
      <ion-input [(ngModel)]="text" type="text" placeholder="Пікір
қалдырыңыз"></ion-input>
    </ion-item>
  </ion-list>
  <button ion-button outline class="recording-btn" (click)="add()">Пікір
қалдыру</button>
  <div class="comments-wrapper">
    <div class="comment-item" *ngFor="let item of comments">
      <h4>{{item.author}}</h4>
      <p>{{item.text}}</p>
    </div>
  </div>
</ion-content>

//
//comments.ts пікірлер беті
import { Component } from '@angular/core';
import { NavController, NavParams } from 'ionic-angular';
import {ServiceProvider} from "../../providers/service/service";
@Component({
  selector: 'page-comments',
  templateUrl: 'comments.html',
})
export class CommentsPage {
  text: string = '';
  comments = [];
  user = JSON.parse(localStorage.getItem('user'));
  constructor(
      public navCtrl: NavController,
      public navParams: NavParams,
      private service: ServiceProvider
  ) {
    this.service.list('/comments').subscribe(res => this.comments = res);
```

```
  }
  add() {
    this.service.push('/comments', {
      author: this.user.firstname,
      text: this.text
    });
    this.service.toast('Пікір сәтті қосылды!');
    this.text = '';
  }
}

//
// Бізбен байланысу беті
<ion-header>
  <ion-navbar>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Байланысу</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <h3>Бізбен байланысу</h3>
  <ion-list>
    <ion-item>
      <ion-input [(ngModel)]="contact.fio" type="text" placeholder="Аты-
Жөніңіз"></ion-input>
    </ion-item>
  </ion-list>
  <ion-list>
    <ion-item>
      <ion-input [(ngModel)]="contact.phone" type="text" placeholder="Телефон
нөміріңіз"></ion-input>
    </ion-item>
  </ion-list>
  <ion-list>
    <ion-item>
      <ion-input [(ngModel)]="contact.email" type="text" placeholder="Эл.
поштаңыз"></ion-input>
```

```
    </ion-item>
  </ion-list>
  <button ion-button outline class="recording-btn" (click)="add()">Жіберу</button>
  <h3>Біздің телефон нөмірлеріміз</h3>
  <a href="tel: 87774733322">87774733322</a>
  <a href="tel: 87774733322">87774733322</a>
</ion-content>

//
//Бізбен байланысу беті
import { Component } from '@angular/core';
import { NavController, NavParams } from 'ionic-angular';
import {ServiceProvider} from "../../providers/service/service";
@Component({
  selector: 'page-contact',
  templateUrl: 'contact.html',
})
export class ContactPage {
  contact = {
    fio: '',
    phone: '',
    email: ''
  };
  constructor(public navCtrl: NavController, public navParams: NavParams, private
service: ServiceProvider) {}
  add() {
    this.service.push('/contacts', this.contact);
    this.service.alert('Пікіріңіз сәтті қосылды!');
    this.contact = {
      fio: '',
      phone: '',
      email: ''
    };
  }
}

//
// Дәрігер жайлы ақпарат бөлімі
<ion-header>
```

Б Қосымшасының жалғасы

```
  <ion-navbar color="hospital">
    <ion-title>Дәрігер жайлы ақпарат</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <div class="doctor-photo">
    <img [src]="item.photo" alt="">
  </div>
  <h3>{{item.fio}}</h3>
  <p>Қабылдау адресі: <span>{{item.address}}</span></p>
  <p>Стажы: <span>{{item.exp}}</span></p>
  <h2 text-center style="color: red;">{{item.price}} тг</h2>
  <ion-list>
    <ion-item>
      <ion-label floating>Уақытты таңдаңыз</ion-label>
      <ion-datetime cancelText="Артқа" doneText="Таңдау" [min]="minTime"
displayFormat="DD MMM HH mm" [(ngModel)]="time"></ion-datetime>
    </ion-item>
  </ion-list>
  <button ion-button outline class="recording-btn" (click)="send()">Жазылу</button>
</ion-content>

//
// Дәрігер жайлы ақпарат бөлімі
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import {ServiceProvider} from "../../providers/service/service";
@IonicPage()
@Component({
  selector: 'page-doctor-detail',
  templateUrl: 'doctor-detail.html',
})
export class DoctorDetailPage {
  item = {} as any;
  user = JSON.parse(localStorage.getItem('user'));
  time = '';
  minTime = new Date().toISOString();
  constructor(public navCtrl: NavController, public navParams: NavParams, private
service: ServiceProvider) {
```

```
    this.item = this.navParams.data;
  }
  send() {
    let toSave = {
      user: this.user,
      item: this.item,
      date: this.time
    };
    this.service.push('/orders', toSave);
    this.service.alert('Сіздің сұранысыңыз сәтті жіберілді!');
    this.time = '';
  }
}

//
//Дәрігерлер бөлімі
<ion-header>
  <ion-navbar color="hospital">
    <ion-title>Дәрігерлер</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <ion-list>
    <button *ngFor="let doctor of doctors" ion-item (click)="detail(doctor)">
      <ion-avatar item-start>
        <img [src]="doctor.photo">
      </ion-avatar>
      <h2>{{doctor.fio}}</h2>
      <p>Қабылдау: {{doctor.price}} тг</p>
    </button>
  </ion-list>
</ion-content>

//
//Дәрігеолер бөлімі
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import {DoctorDetailPage} from "../doctor-detail/doctor-detail";
import {ServiceProvider} from "../../providers/service/service";
```

Б Қосымшасының жалғасы

```
@IonicPage()
@Component({
  selector: 'page-doctors',
  templateUrl: 'doctors.html',
})
export class DoctorsPage {
  private key: string = this.navParams.get('key');
  public doctors = [];
  constructor(public navCtrl: NavController, public navParams: NavParams, private
service: ServiceProvider) {
    this.service.list('/doctors').subscribe((res: any) => {
      this.doctors = res.filter((doctor: any) => {
        return +doctor.service === +this.key;
      });
      console.log(this.doctors);
    });
  }
  detail(item: any) {
    this.navCtrl.push(DoctorDetailPage, item);
  }
}

//
//menu.html Меню бөлімі
<ion-header>
  <ion-navbar color="hospital">
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Қош келдіңіз!</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <img src="assets/imgs/main.jpg">
  <ion-grid>
    <ion-row>
      <ion-col col-4>
        <img (click)="go_to_page('about')" src="assets/icon/Help-2-256.png" alt="">
        <p>Біз жайлы</p>
```

```
    </ion-col>
    <ion-col col-4>
      <img (click)="go_to_page('services')" src="assets/icon/Public_signs-12-512.png"
alt="">
      <p>Қызметтер</p>
    </ion-col>
    <ion-col col-4>
      <img (click)="go_to_page('gallery')" src="assets/icon/Images-icon.png" alt="">
      <p>Галерея</p>
    </ion-col>
  </ion-row>
  <ion-row>
    <ion-col col-4>
      <img (click)="go_to_page('news')" src="assets/icon/4-512.png" alt="">
      <p>Жаңалықтар</p>
    </ion-col>
    <ion-col col-4>
      <img (click)="go_to_page('comments')" src="assets/icon/Comments-01-512.png"
alt="">
      <p>Пікірлер</p>
    </ion-col>
    <ion-col col-4>
      <img (click)="go_to_page('findus')" src="assets/icon/find-icon.png" alt="">
      <p>Бізді табу</p>
    </ion-col>
  </ion-row>
  <ion-row>
    <ion-col col-4>
      <img (click)="go_to_page('contact')" src="assets/icon/Customer-Supprt.png"
alt="">
      <p>Байланысу</p>
    </ion-col>
  </ion-row>
 </ion-grid>
</ion-content>

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

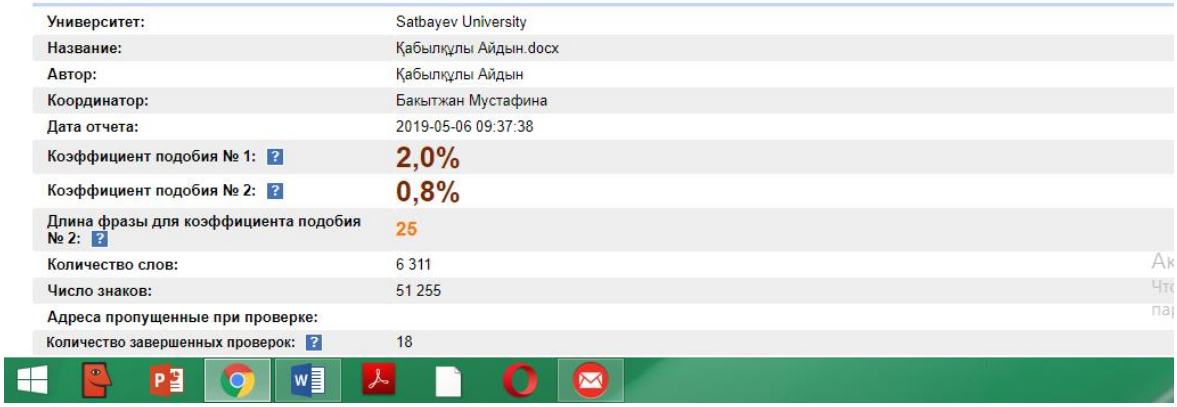
```
import { NavController } from 'ionic-angular';
import {ServicesPage} from "../services/services";
import {AboutPage} from "../about/about";
import {GalleryPage} from "../gallery/gallery";
import {NewsPage} from "../news/news";
import {CommentsPage} from "../comments/comments";
import {FindusPage} from "../findus/findus";
import {ContactPage} from "../contact/contact";
import {RecordsPage} from "../records/records";
@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
  constructor(public navCtrl: NavController) {
  }
  go_to_page(p) {
    switch (p) {
      case 'about':
        this.navCtrl.push(AboutPage);
        break;
      case 'services':
        this.navCtrl.push(ServicesPage);
        break;
      case 'gallery':
        this.navCtrl.push(GalleryPage);
        break;
      case 'news':
        this.navCtrl.push(NewsPage);
        break;
      case 'comments':
        this.navCtrl.push(CommentsPage);
        break;
      case 'findus':
        this.navCtrl.push(FindusPage);
        break;
      case 'contact':
        this.navCtrl.push(ContactPage);
        break;
```

```
      case 'records':
        this.navCtrl.push(RecordsPage);
        break;
    }
  }
}

//
//news.html
<ion-header>
  <ion-navbar>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Жаңалықтар</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <div class="news-wrapper">
    <div class="news-item" *ngFor="let item of news" (click)="detail(item)">
      <div class="left-side">
        <img style="width: 60px; height: 60px;" [src]="item.photo">
      </div>
      <div class="right-side">
        <h3>{{item.title}}</h3>
        <span>{{item.date}}</span>
      </div>
    </div>
  </div>
</ion-content>

//
//news.ts
import { Component } from '@angular/core';
import { NavController, NavParams } from 'ionic-angular';
import {ServiceProvider} from "../../providers/service/service";
import {NewsDetailPage} from "../news-detail/news-detail";
@Component({
  selector: 'page-news',
```

```
  templateUrl: 'news.html',
})
export class NewsPage {
  news: any = [];
  constructor(public navCtrl: NavController, public navParams: NavParams, private
service: ServiceProvider) {
    this.service.list('/news').subscribe(res => this.news = res);
  }
  detail(item: any) {
    this.navCtrl.push(NewsDetailPage, item);
  }
}
```



Университет:	Satbayev University
Название:	Қабылқұлы Айдын.docx
Автор:	Қабылқұлы Айдын
Координатор:	Бақытжан Мустафина
Дата отчета:	2019-05-06 09:37:38
Коэффициент подобия № 1:	2,0%
Коэффициент подобия № 2:	0,8%
Длина фразы для коэффициента подобия № 2:	25
Количество слов:	6 311
Число знаков:	51 255
Адреса пропущенные при проверке:	
Количество завершенных проверок:	18

Ғылыми жетекші
сениор-лектор

_____ Б. М. Мустафина
"_____" _____ 2019 ж.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

5В060200 – «Ақпараттану» мамандығы

Студент Қабылқақұлы Айдын

Тақырыбы: Андроид платформасында «онлайн hospital» мобильді қосымшасын әзірлеу

ҒЫЛЫМИ ЖЕТЕКШІНІҢ СЫН-ПІКІРІ

Диплом жобасын жасаушы Қабылқақұлы Айдын Ionic фреймворкі мен Apache Cordova-ны пайдаланып мобильді қосымша құру міндеті қойылды.

Ionic фреймворкі қазіргі замандағы үздік он фреймворктердің қатарына кіреді, мүмкіншіліктері өте көп болып табылады. Деректер қорын басқару жүйесіне MySQL алынды.

Қабылқақұлы Айдын Firebase-і мен Apache Cordova технологиясын жақсы меңгере отырып, деректер қорын құрып, мобильді қосымша құру жолдарын жүзеге асырды. Дипломдық жоба аурухана жүйесі үшін тиімді, жеңіл әрі ыңғайлы қосымша болмақ. Дипломдық жобаның негізгі мазмұнын құрайтын барлық нәтижелер студенттің өз бетімен алынған.

Қорытындылай келе, дипломдық жоба 5В060200 – «Ақпараттану» мамандығының бітіру жұмыстарына қойылатын талаптарына сәйкес және 5В060200 – «Ақпараттану» мамандығы бойынша «Техника және технологиялар бакалавры» академиялық дәрежесін тағайындауға болады және дипломдық жобаны қорғауға жіберіледі деп есептеймін.

Ғылыми жетекші
«Программалық инженерия»
кафедрасының сениор-лектор

_____ Мустафина Б.

«16» 05 2019ж