ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«Ғұмарбек Дәукеев атындағы Алматы энергетика және байланыс университеті» коммерциялық емес АҚ
Ақпараттық технологиялар институты
IT-инжиниринг кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
PhD, доцент
_____ А.У. Утегенова
«____» _____ 2021 ж.

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: Үміткердің кәсіби шеберлігі бойынша бос жұмыс орындарын іздеу үшін мобильді қосымшаны әзірлеу

Мамандығы: 5В060200 – «Информатика»

Орындаған: Смаилов Қ.А. Тобы: ИНФк-17-1
Ғылыми жетекші: PhD, доцент Утегенова А.У.

Кеңесшілер:
Экономикалық бөлім: э.ғ.к., доцент _____ Е.М. Нұрпейіс
«____» _____ 2021 ж.

Өміртіршілік қауіпсіздігі: аға оқытушы _____ Т.М. Сапаев
«____» _____ 2021 ж.

Программалық қамтама бөлімі: аға оқытушы _____ Ш.П. Жұмағулова
«____» _____ 2021 ж.

Норма бақылаушы: аға оқытушы _____ П. Оралхан
«____» _____ 2021 ж.

Сын-пікір беруші: т.ғ.к., асс.-профессор _____ Ф.Н. Абдолдина
«____» _____ 2021 ж.

Алматы, 2021

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«Ғұмарбек Дәукеев атындағы Алматы энергетика және байланыс университеті» коммерциялық емес АҚ

Ақпараттық технологиялар институты

IT-инжиниринг кафедрасы

Мамандығы 5В060200 – «Информатика»

Дипломдық жобаны орындауға берілген
ТАПСЫРМА

Білім алушы Смаилов Қайсар Адлетұлы

Жобаның тақырыбы: Үміткердің кәсіби шеберлігі бойынша бос жұмыс орындарын іздеу үшін мобильді қосымшаны әзірлеу

2021 жылғы «18» ақпан № 25 университет бұйрығымен бекітілген.

Аяқталған жобаны тапсыру мерзімі: «24» мамыр 2021 ж.

Дипломдық жобаның бастапқы мәліметтері (зерттеу (жоба) нәтижелерінің талап етілген параметрлері мен объектінің бастапқы мәліметтері): Ұсынылып отырған дипломдық жобада жұмыс орындарды беруші тарап заңды тұлғамен шектелмей, сонымен қатар жеке тұлғалар да ұсына алатын платформа жасалынды. Бұл қосымшада SQL базасы мен локалды SQLite базасы және MainActivity, LogInActivity және RegistrationActivity, ал төртінші SplashActivity қолданылды.

Дипломдық жобада қарастырылған мәселелер тізімі немесе дипломдық жобаның қысқаша мазмұны:
- талдау бөлімі;
- жобалау бөлімі;
- жобаны жүзеге асыру және тестілеу бөлімі;
- экономикалық бөлім;
- өміртіршілік қауіпсіздігі;
- А қосымшасы. Техникалық тапсырма;
- Ә қосымшасы. Программа листингі.

Графикалық материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс): 11 кесте, 39 сурет ұсынылған.

Ұсынылатын негізгі әдебиеттер:

1 Машнин, Т.С. JavaFX 2.0: разработка RIA приложений / Т.С. Машнин. – СПб.: BHV, 2012. - 320 с.

2 Кирстен, В. Постреляционная СУБД Cache 5 Объектно-ориентированная разработка приложений / В. Кирстен, М. Иринге, Б. Рериг. – М.: Бином-Пресс, 2011. - 416 с.

3 Фелкер Д. Android. Разработка приложений для чайников; Диалектика / Вильямс – М., 2012. - 288 с.

4 Кузнецов, С. Д. Основы баз данных / С.Д. Кузнецов. – М.: Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2017. - 488 с.

Дипломдық жобаның бөлімдеріне қатысты белгіленген кеңес берушілер

Бөлімдер	Кеңесшілер	Мерзімі	Қолы
Экономикалық бөлім	Нұрпейіс Е.М.		
Өміртіршілік қауіпсіздігі	Сапаев Т.М.		
Программалық қамтама	Жұмағулова Ш.П.		
Норма бақылау	Оралхан П.		

Дипломдық жобаны дайындау
КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Талдау бөлімі		
Жобалау бөлімі		
Жобаны жүзеге асыру және тестілеу бөлімі		

Тапсырманың берілген күні «___» _____ 20__ ж.

Кафедра меңгерушісі _____ А.У. Утегенова

Жобаның ғылыми жетекшісі _____А.У. Утегенова

Тапсырманы орындауға алған
білім алушы _____Қ.А. Смаилов

Аңдатпа

Бұл дипломдық жобада менің алдыма қойылған тапсырма – үміткердің кәсіби шеберлігі бойынша бос жұмыс орындарын іздеу үшін мобильді қосымшаны әзірлеу. Бұл мобильді қосымшаны жасаған кезде Java бағдарлама тілін қолдандым. Сонымен қатар, жобаның платформасын жасау кезде бағдарламалау тілінен бөлек, қосымшаның XML дизайнын, мобильді платформаның деректер базасын, UI/UX сервис жүйесін қарастырдым. Бұл қосымшада жұмыс іздеуші мен беруші арасындағы алмасқан ақпараттар болса немесе жүйеге жұмыс жайлы салынған болса қауыпсіздігіне басты назар аударамын. Сонымен қатар тіркеу кезінде пациент туралы деректерді қате енгізу мәселесін шешу.Бұл мәселелер заманауи технологияны пайдалану арқылы жүзеге асады. Соның ішінде бэкенд жағында веб-сервер құру Java тілінің мүмкіндіктерін қолданыла отырып жүзеге асады.

Жобаны жүзеге асырғандағы экономикалық тиімділік есептелінді және онсы жобаға кететін шығын мөлшерін және пайда есептеледі.

Жұмысшылардың жобамен жұмыс істеу барысында тіршілік қауіпсіздігі ескерілу керек: Жұмыс ортасындағы электромагниттік сәуле бөлетін құралдармен мейлінше дұрыс жұмыс жасау керек.

Аннотация

Задача, поставленная передо мной в данном дипломном проекте – создание мобильного приложения для поиска вакансий по профессиональным навыкам претендента. При создании этого мобильного приложения я использовал язык приложений – Java. Кроме того, при разработке платформы проекта, помимо языка программирования, я рассмотрел XML дизайн приложения, базу данных мобильной платформы, систему сервиса UI/UX. В этом приложении я обращаю особое внимание на то, есть ли обмен информацией между соискателем и поставщиком или если работа построена в системе. А также решение проблемы ошибочного ввода данных о пациенте при регистрации. Эти проблемы реализуются с использованием современных технологий. В том числе создание веб-сервера на стороне бэкенда осуществляется с использованием возможностей языка Java.

Рассчитывается экономический эффект от реализации проекта и рассчитывается сумма затрат на проект и прибыль.

При работе над проектом необходимо учитывать безопасность жизнедеятельности рабочих: максимально правильно работать с приборами, излучающими электромагнитное излучение в рабочей среде.

Annotation

In this diploma project, we were faced with the task of creating a mobile application for job search. When we created this mobile app, we used the Java programming language. In addition, when developing the project platform, in addition to the programming language, we considered the XML design of the application, the database of the mobile platform, and the user interface/UX service system. In this application, we pay special attention to the safety of information exchanged between the applicant and the employer, or if it is built into the system about the work. As well as solving the problem of incorrect entry of patient data during registration. These problems are solved using modern technologies. In particular, the creation of a web server on the back side is carried out using the capabilities of the Java language.

The economic effect of the project is calculated and the amount of costs and profit is calculated for each project.

When working on the project, it is necessary to take into account the safety of life of employees: it is necessary to work as correctly as possible with means that emit electromagnetic radiation in the working environment.

Мазмұны

Кіріспе

Кез-келген кәсіпорынның болашағы үшін мобильді немесе веб-қызметтерді дамыту шешуші болып табылады, өйткені компания үшін мобильді қосымшаны құру аудиторияның қызығушылығын арттыру және бизнесті алға жылжыту тәсілі болып табылады.

Android, iOS немесе Windows платформаларына қосымшалар жасау – бұл компанияның трендте екендігінің айқын белгісі. Пайдаланушыға сайттың үлкен нұсқасында болу әрдайым ыңғайлы емес, сондықтан сізге жеңіл және қол жетімді балама қажет.

Таңдалған тақырыптың өзектілігі Android, iOS және WindowsPhone операциялық жүйелеріндегі ұялы телефондарды пайдаланушылар санының күн сайын өсіп келе жатқандығына байланысты. Адамдар смартфонның көмегімен шексіз ақпаратқа қол жеткізе алатындығын түсінеді: олар бухгалтерлік есеп жүргізе алады, көңіл көтереді, медиа мазмұнды қарайды, пайдалы бағдарламалар мен ойындар орната алады, демалыстарын жоспарлай алады. Осының арқасында мобильді қосымшалар нарығын көптеген адамдар жұмыс істейтін перспективалы сала деп атауға болады.

Күннің көп бөлігін телефонымен өткізетін адам іс жүзінде оның құрылғысындағы қосымшаның қол жетімділігінің шегінде. Мысалы, ол белгілі бір бағдарламаның ескертулерін көреді, оған жұмыс үстеліндегі белгішені қолдана отырып кіре алады және сәйкесінше оның мазмұнымен таныса алады. Осыны ескере отырып, әзірлеушілер қарапайым пайдаланушылардың смартфондары мен планшеттері үшін нақты соғыс жүргізуде. Шынында да, заманауи технологиялардың көмегімен сіз ойын-сауық бағдарламаларын немесе ойындарды шығарып қана қоймай, олардағы жарнамадан ақша таба аласыз, сонымен қатар клиенттерге өз қызметтерін, өнімдерін ұсына аласыз, олардан пікірлер жинай аласыз және т.б. Мобильді қосымшалар кез-келген бизнестің қуатты қозғалтқышы бола алады.

Осылайша, бүгінде клиенттер үшін мобильді қосымшаларды бұқаралық секторларда жұмыс істейтін ұйымдар да, корпоративтік нарықта шоғырланған ұйымдар да ала алады. Кейбіреулер үшін мобильді қосымшаны шығару – бұл сән, ал басқалары үшін – шұғыл қажеттілік, ал басқалары үшін – жаңа бизнес мүмкіндіктерін игеру.

Біздің дипломдық жұмыста қарастырып отырған жоба бұл бос жұмыс орындарын іздейтін қосымша жасау болып табылады. Бұл қосымшаның адамдар үшін пайдалы жағы үйінен шықпай-ақ, өзінің қолынан келетін жұмыс түрлеріне өзінің түйіндемесін білдіре алады. Егер жұмысшы келтірілген жұмыстың талаптарына сай келетін болса, онда кадрлар бөлімінен жұмысшыға қоңырау шалынып, тілдесуге шақыртылады. Бұл қосымшаның ыңғайлы тұсы, жұмыс іздеген адамадар уақыттарын үнемдей алады.

1 Талдау бөлімі

1.1 Мобильді қосымшаны әзірлеудегі бағдарламалау тілдерінің ерекшеліктері

Мобильді қосымшаларды әзірлеу – бұл көбінесе Android Және iOS үшін смартфондар мен сандық көмекшілерге арналған бағдарламалық жасақтама жасау процесі. Бағдарламалық жасақтаманы құрылғыға алдын-ала орнатуға, мобильді қосымшалар дүкенінен жүктеуге немесе мобильді веб-шолғыш арқылы алуға болады. Бағдарламалық жасақтаманы әзірлеу үшін қолданылатын бағдарламалау және белгілеу тілдеріне Java, Swift, C# және HTML5 кіреді.

Мобильді қосымшаларды әзірлеу қарқынды дамып келеді. Бөлшек сауда, телекоммуникация және электрондық коммерциядан бастап сақтандыру, денсаулық сақтау және мемлекеттік басқаруға дейін барлық салалардағы ұйымдар нақты уақыт режимінде транзакцияларды жүргізудің және Ақпаратқа қол жеткізудің ыңғайлы тәсілдеріне қатысты пайдаланушылардың үміттеріне сәйкес келуі керек. Бүгінгі таңда мобильді құрылғылар және олардың құндылығын ашатын мобильді қосымшалар адамдар мен компанияларды интернетке қосудың ең танымал тәсілі. Өзекті, жауапты және сәтті болу үшін ұйымдар клиенттері, серіктестері және қызметкерлері талап ететін мобильді қосымшаларды әзірлеуі керек.

Алайда, мобильді қосымшаларды әзірлеу қиын міндет болып көрінуі мүмкін. ОЖ платформасын немесе платформаны таңдағаннан кейін, сіз мобильді құрылғылардың шектеулерін жеңіп, бағдарламаңызға ықтимал тарату кедергілерін толығымен жеңуге көмектесуіңіз керек. Бақытымызға орай, бірнеше негізгі нұсқаулар мен нұсқауларды орындау арқылы сіз қосымшаларды әзірлеу процесін жеңілдете аласыз [1].

Кез-келген платформада мобильді қосымшаларды әзірлеу туралы көбірек білу үшін iOS және Android қосымшаларын әзірлеу туралы мақалаларымызды оқыңыз.

Мобильді қосымшалардың дамуын зерттеу үшін Android-де дауыстық чат құру үшін қарапайым IBM оқулығын қараңыз. Қазіргі заманғы адамдардың көпшілігі "тек қоңырау шалу үшін" смартфонын пайдаланып, өз күндерін елестету қиын болар еді. Көптеген мобильді құрылғылар телефон қоңырауларын шалу және қарапайым жедел хабарламалар жіберу сияқты негізгі мақсаттардан асып түсті. Бүгінгі таңда тіпті қарапайым ұялы телефон – бұл күнделікті қолдануға арналған әртүрлі қосымшаларды орналастыруға арналған платформа, ол мыңдаған адамдарды көбірек қосымшалардың ризашылық білдірушілеріне айналдырады. Қарапайым немесе күрделі, ақпараттық немесе ойын-сауық, минималистік немесе жарқын және есте қаларлық бөлшектерге толы, практикалық немесе қуанышты, пайдаланушының қажеттіліктерін қанағаттандырады, олар өмірді жақсартады. Бүгінгі таңда қол жетімді қосымшалардың әртүрлілігі пайдаланушыларға тек

ұялы телефондарын қолдана отырып, әр түрлі әрекеттерді жасауға мүмкіндік береді. Ертең дабылды қою, келесі аптадағы шығындарды есептеу немесе Messenger қосымшасы арқылы анама селфи жіберу арқылы пайдаланушылардың басым көпшілігі күнделікті қарапайым операциялардың артында қанша адам тұрғанын елестете де алмайды [8].

Бұрын біздің жобаларымызда біз мобильді қосымшалар үшін интерфейс дизайнын жасаудың әдеттегі кезеңдерін аштық. Бүгін идеяны орнатудан бастап App Store-да шығаруға дейін мобильді қосымшаны құрудың толық жолын белгілейік.

Кез-келген шығармашылық процесс сияқты, ауадан мобильді қосымшаны құру – бұл әр жағдайда өзіндік ерекшеліктері мен ерекшеліктері бар күрделі процесс. Алайда, Tubik Studio-ның әртүрлі қосымшаларды жасаудағы кең тәжірибесіне сүйене отырып, осы процестің бірнеше типтік шығармашылық кезеңдерін анықтауға болады [17].

Сіз кезеңдердің реттілігін көрсеңіз де, бұл әр келесі кезең алдыңғы кезең аяқталған кезде басталады дегенді білдірмейді. Мұндай сызықтық тәуелділікті елестету мүмкін емес, өйткені көптеген процестер мен кезеңдер ұсынылған тізімде дәйекті болмаса да өзара байланысты. Сонымен қатар, тестілеу немесе бағалау сияқты олардың кейбіреулері осы жерде және сол жерде пайда болады, бұл қосымшаларды құрудың бүкіл процесіне таралады. Енді кіші идеяның нақты мобильді қосымшаға айналғанын көру үшін осы жолдан қадам-қадамнан өтейік.

1.1-сурет – Мобильді қосымшаларды әзірлеу ерекшеліктері

1.2 Қазақстан бойынша мобильді қосымшалар нарығына талдау жасау және жасаумен айналысатын компаниялар

Мобильді қосымшаны құруды шешкеннен кейін алдымен оның тақырыбы мен форматын анықтау керек. Бұл жүктеуге, әлеуметтік желіге, мессенджерге және т.б. кіруге арналған бағдарлама болуы мүмкін.

Бұл кезеңде жаңа өнімнің міндеттері тұжырымдалады: ол пайдаланушыға не беруі керек, аналогтардан гөрі, кімге арналған және т. б.;

UI/UX жобасын әзірлеу. Тұжырымдама болашақ өнімнің прототипін жасауды қамтиды. Онда жүзеге асырылуда барлық үйлерінде техзадании функциялары. Бұл кезде қосымшаның қалай жұмыс істейтіні, экранда қандай қосымша түймелер орналастырылатыны туралы ойлану керек;

Тұжырымдаманы әзірлеу. Дизайн – бұл қосымшаның сұлулығын, оның басқа жобаларға ұқсамауын, ыңғайлылығын анықтайтын маңызды құрал;

Экрандарды сызу. Олар қабылданған дизайнда жасалады. Бұл кезеңде белгішелер мен түймелердің орналасуы, дизайн қарастырылады;

Беттеу. Бұл кезең орналасудың статикалық суретін интерактивті жұмыс өніміне айналдыру үшін қажет. Жаңа қосымшаның клиенті мен серверін дұрыс қосу маңызды. [13].

Шын мәнінде, осы қадамдарды орындағаннан кейін дайын бағдарлама алынады, бірақ ол әлі де "шикі". Сондықтан оны тексеру, жұмыс режимінде қуу қажет. Қателер әрдайым дерлік анықталады, оларды дереу жою керек. Содан кейін екінші тестілеу, бақылау өткізіледі. Егер қателер табылмаса, қосымшаны құру туралы мәлімдеуге болады. Құралдардың кем дегенде біреуін өткізіп жіберуге болмайды, олар өзара байланысты. Википедияда мобильді құрылғыларға арналған қосымшаларды әзірлеу өте егжей-тегжейлі сипатталған және іс жүзінде ең егжей-тегжейлі сценарийді қамтиды. Әр жағдайда жұмысты орындау тәсілдері әр түрлі болуы мүмкін.

1.2.1 Қорытынды кезеңдер

Қолыңда дайын жоба болса, оған "белгішені" ойлап табу керек. Ол тәуелсіз графикалық элемент ретінде әрекет етеді. Ол 6 түрлі мөлшерде жасалады, ал суреттің әртүрлі бағдарламалық жасақтамалары бар мобильді құрылғыларда дұрыс және әдемі көрсетілгеніне көз жеткізу керек.

Барлық жұмыстардан кейін соңғы кезең – өнімді жариялау басталады. Бағдарлама AppStore немесе Google play-де орналасқан. Маркетингте көрсетілмес бұрын, ол сайт стандарттарына сәйкестігі тексеріледі. Өзгерістер енгізу туралы өтініш болуы мүмкін, содан кейін қосымша жарияланады. Өнімді тексеру, бекіту және каталогқа енгізу үшін жетерліктей уақыт кетеді.

Қазақстан Республикасындағы мобильді қосымшалар нарығы қазір пайда болу сатысында. Мысал ретінде батыс бизнес моделін алуға болады, онда көптеген туристік фирмалардың iOS немесе Android смартфондарында жұмыс істеуге бейімделген бағалары мен турларының тізімі бар жеке бағдарламасы

бар. Әрине, біздің отандық болмысымыз туралы не айтуға болмайды. Бұл, мүмкін, 5 жылдан кейін бізді дәл солай күтеді. Қазақстандағы мобильді құрылғылардың үштен екісінен астамы танымал болып келе жатқан Android-де жұмыс істейді. Қазақстандағы iOS перспективалары өте күмәнді, атап айтқанда Apple-дің iPhone-ның құнын 70 жылдың желтоқсан айында теңгеге шаққанда 2014% - ға арттыру туралы шешіміне байланысты. Windows Phone нарықтың маңызды үлесін ала алмады [16].

1.1-кесте – ҚР мобильді қосымшаларды әзірлеушілердің рейтингі 2020 жылғы санақ бойынша

Компания атауы	Баллдық көрсеткіш	Тренд	Жаслған проекттер саны
Magora Systems	50	Сұраныста	69
«Максимус»	42,87	Жаңа	63
Sailet	11,64	Жаңа	13
Crystal Spring	3,44	-	48
v-jet group	3,15	Жаңа	19
«Ракетная фирма»	1,07	Жаңа	6

1.3 Қолданыстағы мобильді платформаларға шолу

Операциялық жүйенің (ОЖ) болуы – смартфонды әдеттегі ұялы телефоннан ерекшелейтін басты қасиет. Телефонның немесе құрылғының нақты моделін таңдағанда, амалдық жүйе көбінесе анықтаушы фактор болып табылады.

Смартфондарға арналған ең көп таралған операциялық жүйелер және платформалар:

– Symbian OS – ОС смартфондар нарығының көп бөлігін соңына дейін иеленді 2010 ж. 2010 жылдың басында ОС базасында тек 1 Платформа қалды: Series 60, ол негізінен Nokia құрылғыларында, сондай-ақ кейбір Samsung модельдерінде қолданылады;

– BlackBerry OS (RIM) – жүйе құрылғыларда кеңінен қолданылады.

бірінші кезекте Америка Құрама Штаттарында, өйткені кейбір елдердің арнайы қызметтері елде смартфондарды қолдануға қызығушылық танытпайды, өйткені барлық кіріс және шығыс деректер AES шифрлау алгоритмін қолдана отырып шифрланған;

– Windows Mobile және Windows CE ықшам Операциялық жүйе 1996 жылдан бастап шығарылған және 2010 жылға қарай смартфондарға арналған ОС нарығының ең үлкен сегментін иемденген Microsoft қазіргі уақытта қолдау мен дамудан біртіндеп бас тартуда;

– Windows Phone 7 - бұл Windows Mobile-тан түбегейлі ерекшеленетін Microsoft әзірлемесі.;

– Palm OS – қазіргі уақытта Palm OS негізіндегі ұялы телефондар сирек кездесетін танымал платформалардың бірі. Операциялық жүйенің соңғы смартфоны 2007 жылдың соңында шығарылды (Palm Centro);

– Linux – бұл операциялық жүйе мобильді құрылғыларда кең таралмады, бірақ оның дамуы дәстүрлі түрде перспективалы бағыт болып саналады. Linux негізіндегі смартфондар негізінен Азияда таратылады;

– Bada – компания әзірлеген ең жаңа мобильді платформа Samsung. Жаңа платформадағы алғашқы телефон S8500 Wave болды;

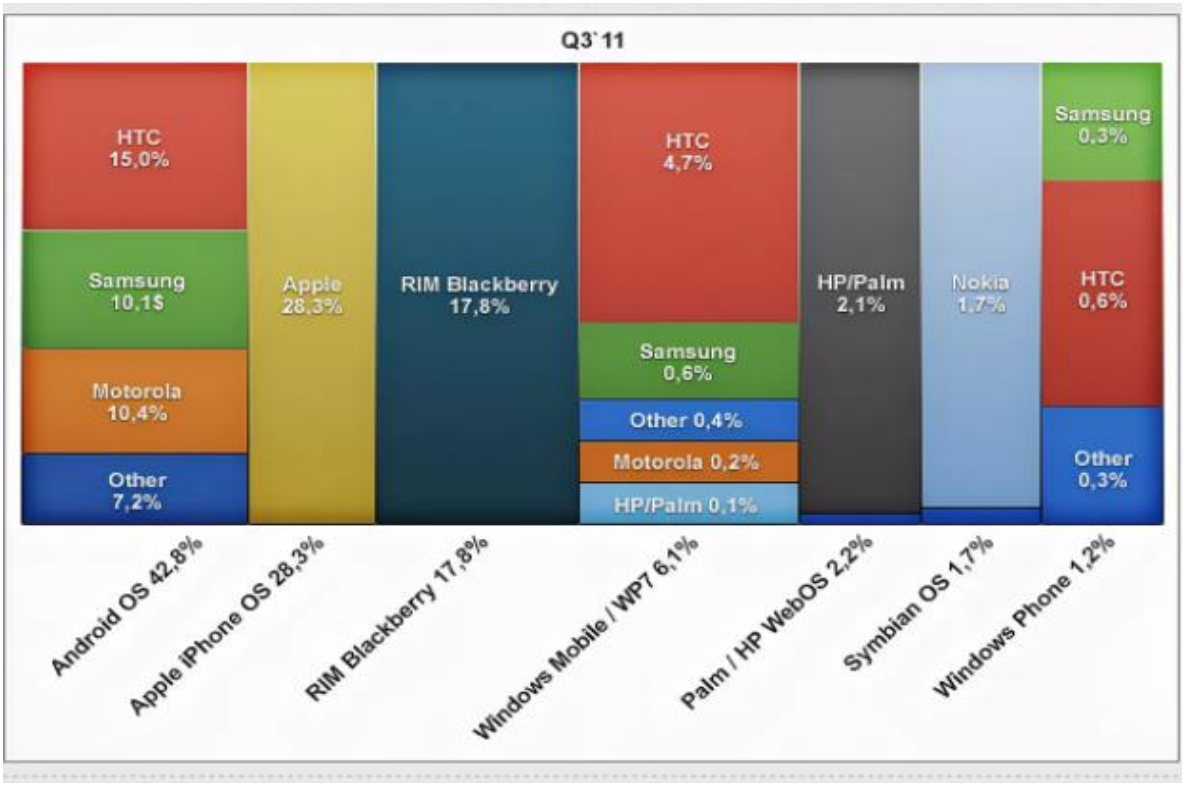
– Android – смартфондарға арналған портативті (желілік) операциялық жүйе, Linux ядросына негізделген планшеттік компьютерлер, электронды кітаптар, Сандық ойнатқыштар, сағаттар және нетбуктар. Бастапқыда Android Inc.компаниясы жасаған., содан кейін Google сатып алды. Кейіннен Google қазір платформаны қолдау және одан әрі дамытумен айналысатын Open Handset Alliance (OHA) альянсын құруды бастады. Android сізге Google әзірлеген кітапханалар арқылы құрылғыны басқаратын Java негізіндегі қосымшалар жасауға мүмкіндік береді. Android Native Development Kit жүйеге C және басқа тілдерде жазылған қосымшалар кітапханалары мен компоненттерін пайдалануға мүмкіндік береді;

– IOS ОЖ (2010 жылдың 24 маусымына дейін iPhone OS) – бұл американдық Apple компаниясы әзірлеген және шығарған мобильді операциялық жүйе. Ол 2007 жылы шығарылды; бастапқыда iPhone және IPod Touch үшін, кейінірек – IPad және Apple TV сияқты құрылғылар үшін. Windows Phone және Google Android-тен айырмашылығы, тек Apple шығарған құрылғылар үшін қол жетімді;

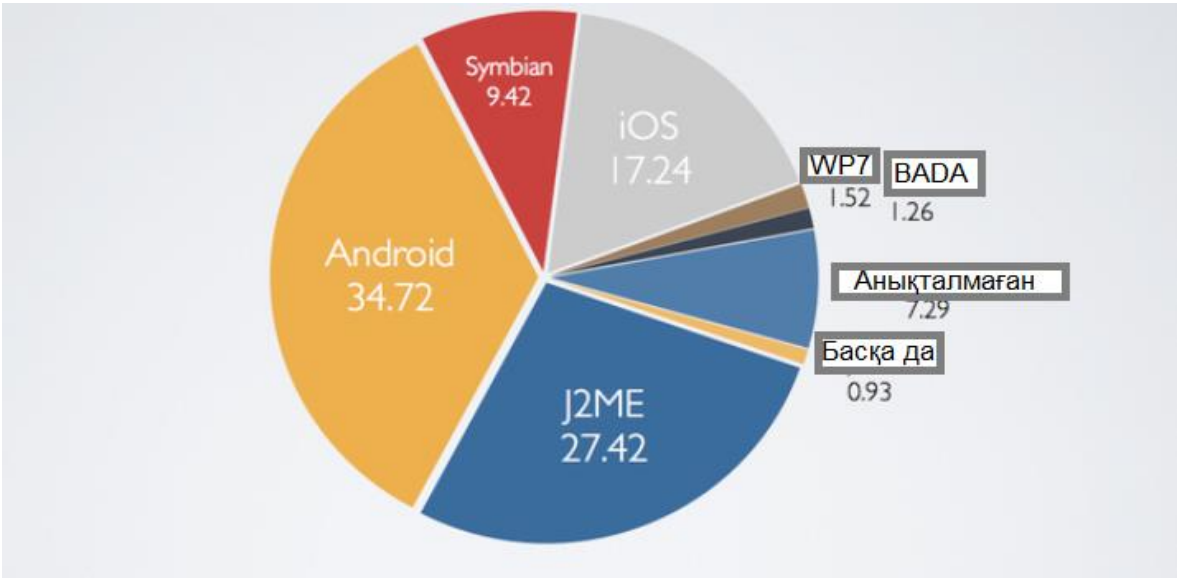
– Windows Phone 8 – Microsoft Windows телефон операциялық жүйесінің екінші буыны. Шығу 29 қазанда болды және Metro (немесе Modern UI) деп аталатын интерфейстің прототипі болып табылады. Windows Phone 8 Microsoft операциялық жүйелерінде қолданылатын жаңа Windows NT архитектурасын қолданады. Windows жұмыс істейтін құрылғылар [18].

– Phone 7. X Windows Phone 8 жаңартуын орындай алмайды, ал Windows Phone 8 үшін жасалған жаңа бағдарламалар Windows Phone 7-де жұмыс істей алмайды. Қазіргі уақытта Android экспоненциалды түрде дамуда.

Жыл сайын осы операциялық жүйені пайдаланушылар саны үнемі өсіп келеді. Canalys компаниясының соңғы есебіне сәйкес, жоғары технологиялық индустрияның жетекші талдаушысы, Android операциялық жүйесі мобильді құрылғылардың әлемдік нарығының 69,2% құрайды. Әрине, бұл факт, әсіресе Android үшін мобильді қосымшалар жасау үшін көптеген әзірлеушілердің назарын аударады. Мүмкін, бүгінгі күні бұл ең танымал және қызықты жүйе. Әзірлеушілер пайдаланушыларға бірегей мүмкіндік береді. Еркін бағдарламалық жасақтама жиынтығын орнату арқылы сіз жүйе үшін бағдарламалар құрып, оларды мамандандырылған интернет-дүкенде сата аласыз [21].



1.2-сурет – Ұялы телефондар өндірісінің нарығы



1.3-сурет – Интернет желісі бойынша қолданылатын платформалар, %

1.4 Құрылғыларды мобильді қосымшалар үшін қауіпсіздігі

1980 жылдары негізгі кадрларға алғашқы шабуылдардан бастап барлық компьютерлік жүйелердің қауіпсіздігі ерекше өзекті болды. Моррис құрты пайда болғаннан бері мұндай жүйелер көп өзгергенімен, қауіптер мен шабуылдар толығымен жойылмады. Керісінше, зиянды бағдарламалық кодты

және басқа киберқауіптерді қолданатын шабуылдар саны экспоненциалды түрде артады. Компьютерлік индустрия мобильді платформаларға көшкен сайын, шабуылдардың мақсаты мобильді жүйелер мен қосымшалар болып келеді. Sophos компаниясының 2013 жылғы зерттеуі көрсеткендей, кибершабуылдардың негізгі мақсаты қазір Android болып табылады, ал F-secure компаниясының есебінде алдыңғы екі жылда зиянды мобильді бағдарламалардың саны бірнеше жүзден 50 мыңға дейін өсті. Мобильді құрылғылардың әмбебап таралуы мен танымалдылығы мобильді қосымшалардың қауіпсіздігін қамтамасыз ету мәселесінің өте өзекті болуына әкеледі. Мұндай құрылғыларда жеке ақпараттың үлкен көлемі сақталғандықтан, олар қаржылық пайда алуға ұмтылатын шабуылдаушылардың шабуылдары үшін тартымды мақсаттар болып табылады. Алайда, Symantec мәліметтері бойынша, ересек пайдаланушылардың тек 57%-ы мобильді нысандардың қауіпсіздігін қамтамасыз ету құралдары бар екенін біледі. Осыған ұқсас есептер шабуылдардың жартысына жуығы жеке басын ұрлауға және пайдаланушылардың мінез-құлқын бақылауға бағытталғанын көрсетеді. Жағдай ұялы құрылғылардың салыстырмалы түрде төмен есептеу қуаты мен шектеулі пайдаланушы интерфейсі болуымен күрделене түседі, бұл зиянкестерге зиянды бағдарламалардың белсенділігін пайдаланушылардан жасыруға мүмкіндік береді [9].

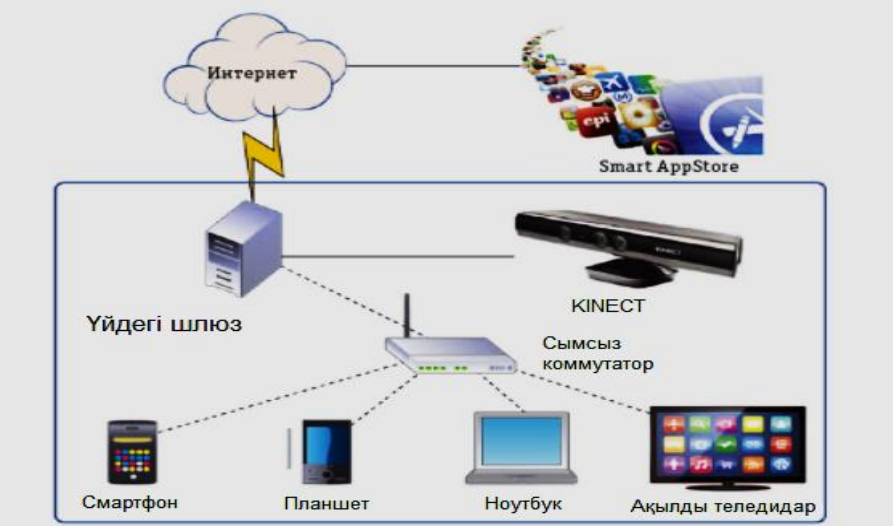
Мобильді платформалардың қауіпсіздігін қамтамасыз ету үшін зерттеушілер әртүрлі стратегияларды қолдануға тырысады. Кейбір тәсілдер негізінен зиянды қосымшаларды талдау, анықтау және бағалауды қолданады. Жүйелердің қауіпсіздігін қамтамасыз етудің дәстүрлі тәсілдеріндегідей, талдау статикалық және динамикалық әдістерді немесе олардың комбинациясын қолдануға негізделуі мүмкін. Бірқатар зерттеулер деректердің қауіпсіздігін жақсарту үшін жобалық шешімдерді іздеуге бағытталған, мысалы, деректерге қол жеткізу құқығын бақылау немесе қосымшаларды орындау ортасының оқшауланuын қамтамасыз ету.

Алайда, мобильді қосымшалардың қауіпсіздігі саласында сіз қосымшалар мен мәліметтермен шектеле алмайсыз – мобильді платформалар әртүрлі жаңа ортада қолданылады және пайдаланушыларға дербес компьютерлер үшін мүмкін емес тәсілдермен әсер етеді. Ұялы ботнеттердің пайда болуы тән мысал бола алады. Мобильді құрылғыларды пайдаланудан "киюге болатын" құрылғыларды ("ақылды" сағаттар, көзілдіріктер және т.б.) қолдануға көшу одан да көп алаңдаушылық тудырады. Мобильді қосымшалардың қауіпсіздігін қамтамасыз ету бағыты Ақпараттық қауіпсіздік қоғамдастығы үшін ерекше қызығушылық тудырады, себебі оның жаңалығы мен өзектілігі ғана емес, сонымен қатар мобильді қауіпсіздік Жүйелік қауіпсіздік идеяларына қатты әсер етуі мүмкін.

Тақырыптық таңдау бес үлкен мақаланы қамтиды. Олардың алғашқыларын Гильермо Суарес-Тангил (Guillermo Suarez-Tangil), Хуан Тапиадор (Juan Tapiador), Флавио Ломбарди (Flavio Lombardi), Роберто Ди

Пиетро (Roberto Di Pietro) жазды. Ол "ақаулардың дифференциалды талдауы арқылы зиянды бағдарламаларды залалсыздандыру" (Thwarting Obfuscated Malware via Differential Fault Analysis) деп аталады. Мақалада қосымшалар пакеттеріндегі зиянды компоненттерді анықтауға мүмкіндік беретін Alterdroid құралы сипатталған [5].

Alterdroid алдымен алдын-ала анықталған модельдерге негізделген статистикалық талдауды қолдана отырып, күдікті компоненттерді таңдайды. Содан кейін кейбір қате кездейсоқ таңдалған күдікті компонентке енгізіледі, содан кейін бүкіл бағдарлама қайта жасалады. Android контейнерлерінде қосымшаның өзгертілген және бастапқы нұсқалары іске қосылады, зиянды мінез-құлық дифференциалды талдауды қолдана отырып, олардың орындалу жолдарын салыстыру арқылы анықталады.

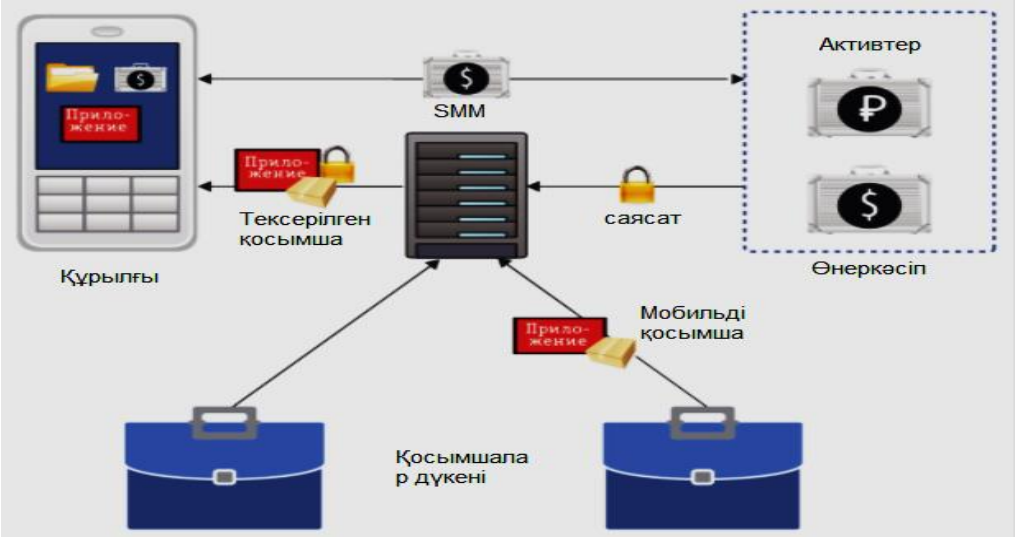


1.4-сурет – Мобильді құрылғылардың экосистемасы

Пайдаланушылар үй шлюзімен байланысқан веб-камераны (мысалы, Kinect-те орнатылған) пайдаланып қолданбалар дүкенімен өзара әрекеттеседі. Жүйе бөлмедегі пайдаланушыларды таниды және белсенді пайдаланушылардың тізімін немесе пайдаланушы топтарының профильдерін жасайды [3].

Мобильді құрылғыларды пайдалану шекараларының кеңеюі әртүрлі ұйымдардың өз қызметкерлеріне кеңседе өз құрылғыларында жұмыс істеуге мүмкіндік беруіне әкелді. Нәтижесінде ыңғайлылық пен қауіпсіздік арасындағы тепе-теңдікке қол жеткізуге байланысты жаңа проблемалар туындайды. Alessandro Armando (Alessandro Armando), Gabriele Costa, Luca Verderame және Alessio Merlo (Alessio Merlo) "BYOD парадигмасын қолдану кезінде қауіпсіздікті қамтамасыз ету" мақаласында ("өзіңіздің құрылғыңызды ұрлау" Paradigm) BYOD жағдайында мобильді құрылғыларды қолдану кезінде туындайтын қауіпсіздік мәселелерін талқылайды. BYOD саясатын анықтау мен қолдануды қолдайтын "метамаркет" қауіпсіздік архитектурасы (secure metamarket, SMM) ұсынылады Авторлар жасаған прототиптің Android

ортасында орындалуы сипатталған және оның негізінде жүргізілген эксперименттердің нәтижелері келтірілген. "Firefox OS және Tizen мобильді платформаларындағы қауіпсіздік" (Security in the Firefox OS and Tizen Mobile Platforms) тақырыптық топтамасының соңғы мақаласын Ольга Гадяцкая (Olga Gadyatskaya), Фабио Массаччи (Fabio Massacci) және Юрий Жавнерович (Yury Zhauniarovich) ұсынды. Авторлар екі дамып келе жатқан мобильді операциялық жүйені (Firefox OS және Tizen) ұсынады және оларда қолданылатын қауіпсіздік тәсілдерін талқылайды. Бұл жүйелер мен дәстүрлі мобильді ОЖ арасындағы маңызды айырмашылық олардың веб-қосымшалармен алғашқы интеграциясы болып табылады. Дегенмен, Android сияқты қарастырылған екі ОС Linux-қа негізделген және мақалада екеуі мен Android арасындағы негізгі айырмашылықтар қарастырылады [2].



1.5-сурет – SMM архитектурасын қолданатын BYOD сценарийі

SMM архитектурасын қолданатын BYOD сценарийі жеке мобильді құрылғыда орнатылған қосымшалардың ұйымның қауіпсіздік саясатына сәйкес келуін қамтамасыз етеді. Өнімділікке арналған қосымшалар: олар электрондық поштаны жіберу, жұмыс барысын бақылау, қонақүйлерді брондау және тағы басқалар сияқты әртүрлі тапсырмаларды жеңілдету арқылы бизнестің тиімділігін арттыруға бағытталған.

1.5 Құрылғылардың мобильді қосымшалар үшін қолдануға тиімділігі мен оның артық-кемшіліктерін салыстыру

iOS – бұл Apple жасаған мобильді операциялық жүйе. iOS – бұл iPhone операциялық жүйесінің қысқа нысаны. IOS-тың алғашқы нұсқасы 2007 жылы шығарылды. Қазіргі тұрақты iOS нұсқасы 13, ал iOS 14 бета нұсқасы осы күндері шығарылды. IOS 14-тің тұрақты нұсқасы шығаруға жақын. iOS iPhone

құрылғылары үшін қолданылады, ал iWatch және iPad-та сәйкесінше watchOS және iPadOS деп аталатын әртүрлі операциялық жүйелер бар.

iOS – бұл Android-тен кейінгі екінші танымал мобильді операциялық жүйе. iOS 2019 жылы 54 миллиард доллар табыс әкелді, ал android app store 2019 жылы 39 миллиард доллар табыс әкелді. Apple компаниясы дамыған елдердің пайдаланушыларының көпшілігіне ие. Apple app store дүкенінде 2 миллионнан астам қосымшалар бар.

IOS операциялық жүйесінің артықшылықтары:

Бұл операциялық жүйе өте жиі сынға ұшырайды және бәрі де оған қуанбайды. Бұған ерекше себептер бар. iOS жабық платформа. Бір қарағанда, бұл жерде ештеңе жоқ, өйткені дұрыс конфигурацияланған жүйе жоғары жылдамдыққа, еске түсіруге және қосымшаларды іске қосуда қиындықтар мен ақаулардың болмауына кепілдік береді. Бірақ егер сіз екінші жағынан қарасаңыз, пайдаланушы гаджетті "өзі үшін"конфигурациялай алмайды.[13].

– iOS қолданушыларына мобильді құрылғыны оңай пайдалануға көмектесетін қарапайым интерфейске ие. Егер сізде ескі iPhone болса және жаңа iPhone құрылғысына ауысқыңыз келсе, пайдаланушы интерфейсінде (пайдаланушы интерфейсі) ешқандай өзгеріс болмайды;

– Android-де әр түрлі экран өлшемдері бар құрылғылардың кең спектрі бар және қолданба жасаушыларға әр түрлі экран өлшемдеріне арналған қосымшаны жасау өте қиын болады. Бірақ iOS-та бірнеше құрылғылар бар және әзірлеушілер қосымшаларды оңай жасай алады;

– Apple iPhone телефонды пайдаланған кезде аз жылу шығарады. IPhone-дағы қосымшалар батареяны тиімдірек пайдаланады және осылайша аз жылу шығарады;

iOS операциялық жүйесінің кемшіліктері:

– iPhone негізгі экранда ұқсас белгішелерге ие және бұл белгішелер алдыңғы нұсқалармен бірдей дизайнға ие. Негізгі экран iOS-ты жаңа нұсқаға жаңартқан кезде бірдей көрінеді, яғни сіз негізгі экранда бірдей белгіше стилін көресіз;

– IOS құрылғылары өте қарапайым және сіз оларды компьютер ретінде пайдалана алмайсыз. Бірақ егер сіз мұны Android құрылғыларымен салыстыратын болсаңыз, онда сіз өз компьютеріңізді құрылғыларда да жасай аласыз;

– қолданбалар қымбат және олар үшін Виджет қолдауы жоқ. Егер сіз Android қосымшаларын салыстыратын болсаңыз, онда Android қосымшаларының көпшілігін ақысыз пайдалануға болады;

Android – бұл Google жасаған мобильді операциялық жүйе. Қазіргі уақытта Android – бұл нарықта өте танымал ОС. Google play Store-да Android құрылғысына жүктеу және орнату үшін 2 миллионнан астам қосымшалар қол жетімді. Android ОЖ-ны бүкіл әлемде шамамен 2 миллиард адам қолданады және қазіргі уақытта ең көп таралған ОЖ болып табылады. Сағат, автомобиль аудио ойнатқышы, Android TV, компьютерлер, планшеттер және

смартфондарды қоса алғанда, Android қолданатын құрылғылардың үлкен таңдауы бар.

Windows Phone артықшылықтары:

– Windows phone әлемдегі ең жеке смартфон деп аталады, өйткені сіз бұл телефонды оңай өзгерте аласыз, оның Microsoft өнімдерімен интеграциясы көптеген пайдаланушыларға, әсіресе Windows өнімдерімен көп жұмыс істейтіндерге, Windows Phone құрылғылардың барлық деңгейлерінде жақсы жұмыс істейді және сапасы жақсы;

– Windows Phone-да Cortana, Live Tiles, Offline Maps & Bangla Keyboard сияқты көптеген ерекше мүмкіндіктер бар, Cortana – бұл Windows Phone 8.1-ге қосылған жеке көмекші, сондықтан күнделікті әрекеттерді орындау оңай;

– Windows Phone керемет Lumia камерасымен бірге келеді 720, 820, 920, 925, 1520, etc – бұл жақсы сапалы камера телефоны, сондықтан сіз Windows phone-ны iPhone немесе Samsung Galaxy смартфондарымен салыстыра аласыз, Live Tiles – бұл басқа смартфондардың операциялық жүйелерінде ешқашан кездеспейтін Windows phone-ның ерекше ерекшелігі, ол динамикалық түрде өзгеруі мүмкін, оны көру өте жағымды және ешқашан баяуламайды;

– Windows phone iPhone сияқты офлайн карталармен бірге келеді, сіз картаны желіге қосылмай пайдалана аласыз, олар негізінен Android смартфондарында жоқ, Windows телефондарында Windows 8.1-дегі алғашқы ресми Bangla пернетақтасы болып табылатын тұрақты Bangla пернетақтасы бар, барлық дерлік тілдер Windows phone-да қол жетімді;

Мобильді қосымшаларды әзірлеу – бұл мобильді құрылғыда жұмыс істейтін бағдарламалық жасақтаманы құру процесі. Мұндай қиын міндет уақытты, дағдыларды және жеткілікті бюджетті қажет етеді. Дегенмен, тәжірибелі топпен де құнды нәрсе жасау әрдайым мүмкін емес, өйткені миллиондаған қосымшалар бар, бұл олардың арасында бөлектеуді қиындатады. Күн сайын әзірлеушілер жүздеген қосымшаларды жасайды: кейбіреулері танымал бола бастайды, табыс пен пайда әкеледі; басқалары жаппай таралмай, пайдаланушылардың тар шеңберімен шектеледі [19].

Бүгінде әрбір екінші адам смартфонға күніне 5 сағат жұмсайды. Қазіргі үрдіс сіздің бизнесіңізді құруға және интернетке деген сүйіспеншілікті тиімді түрде монетизациялауға мүмкіндік береді. Жарнамадан басқа, мобильді қосымша компанияның имиджін жасайды, мақсатты аудитория туралы ақпарат береді және сатып алушылардың мінез-құлық факторларын бағалайды.

Қарапайым мобильді қосымшалар ДК негізіндегі қосымшаларды алып, оларды мобильді құрылғыға өткізеді. Мобильді қосымшалар неғұрлым сенімді бола бастағанда, бұл әдіс біршама жетіспейді. Неғұрлым күрделі тәсіл оның шектеулері мен артықшылықтарын қолдана отырып, мобильді орта үшін арнайы дамуды қамтиды. Мысалы, орынға негізделген функцияларды қолданатын қосымшалар бастапқыда мобильді құрылғыларды ескере отырып, нөлден жасалады, егер пайдаланушы компьютердегідей орынға байланбаған болса. Қосымшалар екі кең санатқа бөлінеді: жергілікті қосымшалар және веб-қосымшалар. Жергілікті қосымшалар жақсы өнімділікке ие және UI (UI)

жақсырақ конфигурацияланған, және оларды шығарар алдында, әдетте, әзірлеу мен сапаны қамтамасыз етудің әлдеқайда қатаң процедурасынан өту керек.

1.2-кесте – IOS, Android, Windows Phone операциялық жүйелерін салыстыру

Критерийлар	iOS	Android	Windows phone
Жасау уақыты	Тілді меңгеру: Swift – жылдам; Objective-C – баяу. Әзірлеу уақыты: орташа	Тілді меңгеру: жылдам. Даму уақыты: жоғары.	Тілді меңгеру: жылдам. Әзірлеу уақыты: орташа.
Мамандардың болуы	Objective-C тілі: мамандардың тар шеңбері. Swift тілі: көптеген мамандар	Ол Java тілін қолданады, сондықтан көптеген мамандар бағдарламалай алады	Негізінде C# тілі қолданылады, сондықтан көптеген мамандар бағдарламалай алады
Әзірлеу және жөндеу ыңғайлылығы	Әзірлеу құралдары толық дамыған	Әзірлеу құралдары толық дамыған	Әзірлеу құралдары толық дамыған
ОЖ жұмыс жылдамдығы	Пайдаланушының барлық әрекеттерін жүктеу уақыты: жылдам. Пайдаланушының әрекеттеріне жүйенің жауабы: жылдам. Күрделі сұраныстарды өңдеу мүмкіндігі мен уақыты: жылдам, кідірістер болуы мүмкін.	Пайдаланушының барлық әрекеттерін жүктеу уақыты: жылдам. Пайдаланушының әрекеттеріне жүйенің жауабы: жылдам. Күрделі сұраныстарды өңдеу мүмкіндігі мен уақыты: жылдам, кідірістер болуы мүмкін.	Пайдаланушының барлық әрекеттерін жүктеу уақыты: жылдам. Пайдаланушының әрекеттеріне жүйенің жауабы: жылдам. Күрделі сұраныстарды өңдеу мүмкіндігі мен уақыты: жылдам, кідірістер болуы мүмкін.
Usability	Шектеулердің болуы. Пайдаланушы интерфейсі интуитивті	Шектеулердің болуы. Пайдаланушы интерфейсі интуитивті	Шектеулердің болуы. Пайдаланушы интерфейсі интуитивті

Бұл дипломдық жобадағы программалық жабдықтау клиент-серверлік технологияларға негізделген. Веб-қосымшалар HTML5 немесе CSS-те қолданылады және олар браузер арқылы іске қосылатындықтан, құрылғының минималды жадын қажет етеді. Пайдаланушы белгілі бір веб-параққа қайта бағытталады және барлық ақпарат сервердегі дерекқорда сақталады. Веб-қосымшаларды пайдалану үшін тұрақты байланыс қажет [7].

Қазіргі уақытта қосымшалардың бірнеше түрлері бар:

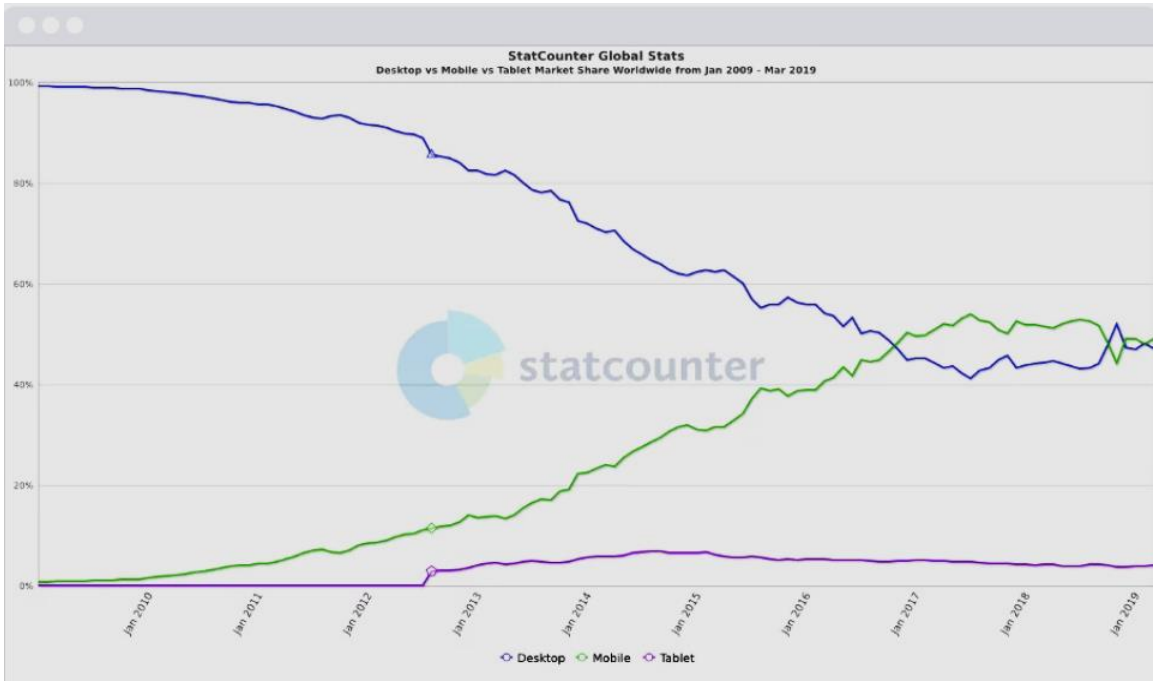
– Ойын қосымшалары: компьютерлік бейне ойындарға балама, олар ең танымал қосымшалардың бірі. Олар барлық қосымшаларды жүктеудің үштен бір бөлігін және тұтынушылық шығындардың төрттен үш бөлігін құрайды.

– Өнімділікке арналған қосымшалар: олар электрондық поштаны жіберу, жұмыс барысын бақылау, қонақүйлерді брондау және тағы басқалар сияқты әртүрлі тапсырмаларды жеңілдету арқылы бизнестің тиімділігін арттыруға бағытталған.

1.6 Мобильді қосымшаларды жобалауға арналған өзіндік сақталу керек ережелер

Әлемде ноутбуктер мен жұмыс үстелдерінен гөрі ұялы телефондар көп, олар интернет-трафикті көбірек жасайды. Төменде, салыстырмалы диаграммада, пост-дербес компьютер дәуірі басталған кезде бетбұрыс айқын көрінеді.

Бағдарлама дизайнеріне қандай білім мен дағдылар қажет, мобильді қосымша қалай жасалады, оны жасау кезінде не ескеру керек. Бұл мобильді қосымшалардың адам өміріндегі алатын орны ерекше екенін көптеген тұтынушылар біледі [10].



1.6-сурет – Компьютер мен ұялы телефондардың қолданысының көрінісі

2016 және 2017 жылдардың басында ұялы телефондар алға шықты және жақын арада бұл үрдіс өзгеруі екіталай. Графикте тек Интернет-трафик ұсынылған, бірақ сонымен қатар жергілікті қосымшалар бар. Сондықтан ұялы телефон сандық әлемнің патшасы деп айта аламыз.

Қосымшалармен жұмыс тіпті жаңа пайдаланушыларға таныс және интуитивті болуы керек, Сондықтан Google және Apple әзірлеушілерді жергілікті қосымшалардың интерфейстерін біріктіруге шақырады. Және ұсынымдар береді.

Android үшін Google Material Design, ал iOS үшін Human Interface Guidelines жазылған. Бұл беттерді бетбелгілерге қосуды ұмытпаңыз, сіз жұмыс кезінде оларға жиі ораласыз. Оларды оқып шығыңыз, атау, дизайн және ыңғайлылық үлгілерімен танысыңыз. Бұл құжаттар үнемі жаңартылып отыратындығын ұмытпаңыз, сондықтан өзгерістер туралы хабардар болу үшін оларды мезгіл-мезгіл қарап отырыңыз [6].

1.7 Мобильді қосымшалардың форматына талдау

Мобильді жарнаманы орналастыру тез әрі оңай болғанымен, мобильді жарнамадағы жарнаманың қазіргі форматтарын білмесеңіз, ең жақсы жарнамалық желіні таңдау көңіліңізді қалдыруы мүмкін. Әр жарнама форматының артықшылықтары мен кемшіліктерін түсіну сізге дұрыс жарнама форматын таңдауға көмектеседі.

Мобильді жарнаманың жетекші форматтарын қарастырайық.

Баннер – жарнаманың қуатты түрі. Ол интернетте жақсы жұмыс істесе де, мобильді қосымшалар үшін керемет нәтиже бермейді. Жақында жиналған AppFlood мәліметтері көрсеткендей, ол ең төменгі eCPM және CTR ұсынады. Алайда, баннер брендтің көрінуі мен жарнама берушілердің тәжірибесін арттыруға тырысатын жарнама берушілер үшін әлі де дұрыс жұмыс істейді.

Артықшылықтары:

– үлкен көлем;
– әр скриншотта қол жетімді;
– жаһандық із;
– қол жетімді формат.

Кемшілігі:

– төмен eCPM және CTR;
– қайталанатын жарнама.

Бұл мобильді қосымшаның жарнама берушілеріне пайдаланушылар оларды жүктеуге шешім қабылдағанға дейін қысқа алдын-ала қарауды қамтамасыз етуге мүмкіндік беретін интерактивті жарнамалар.

Сіз ойнатылатын жарнамаларды Interstitial Audience Network, Facebook жаңалықтар арнасы және басқалардан таба аласыз [5].

Артықшылықтары:

– өнімнің сипаттамалары нақты уақыт режимінде көрсетілуі мүмкін;
– ол кез-келген өнім немесе бренд үшін жұмыс істейді;
– бренд мазмұнымен тамаша өзара әрекеттесу.

Кемшіліктері:

– техникалық қиындық
– жоғары баға және өндіріс мерзімі

Бейне жарнама 15-тен 30 секундқа дейін созылады және қосымшаны жасаушылар үшін үлкен табыс көзі болып табылады. Сонымен қатар, бейне жарнама форматын қолдану қарапайым және оны Adcolony және iron Source сияқты танымал жарнама желілері қолдайды.

Бейне in-app үшін негізгі форматқа айналғаннан бері кейбір өзгерістерге ұшырады. Негізгі түрлендірулердің бірі – пайдаланушылар уақытының 90%-дан астамын мобильді құрылғыларды тік жылжытуға жұмсайды. Сонымен қатар, тік бейнелер ландшафтқа қарағанда 20 пайызға жақсы түрлендіріледі.

Артықшылықтары:

– жоғары CTR;
– пайдаланушылар бейнені көргенді ұнатады
– назар аудару.

Кемшіліктері:

– өткізілмеген бейне сізді ашуландыруы мүмкін.

Мобильді қосымша үшін қандай жарнама форматтарын пайдалану керек?

Бұл жарнама мен қосымшаның форматына байланысты. Жоғарыда көрсетілгендей, жарнаманың әр форматының өзіндік ерекшеліктері бар. Кейбір жарнама форматтары әлеуметтік қосымшалармен, ал басқалары ойын форматтарымен жақсы жұмыс істейді [16].

Әрбір бағдарлама ерекше, сондықтан сіз ең жақсы жарнама форматын және жарнама желісін таңдағаныңызды білгіңіз келсе, көрсетілген жарнама форматтары сіздің бағдарламаңыздың пайдаланушысына кедергі жасамайтынына, әртүрлі жарнама форматтарын зерттеп, бірнеше жарнама форматтарын және бір жарнама желісін пайдаланатындығына көз жеткізуіңіз керек.

2 Жобалау бөлімі

2.1 Мобильді қосымшаны құрудың негіздемесі және этаптары

Сіздің қосымшаңызды құрудың пайдасына қатысты барлық дәлелдерге қарамастан, жеңіл-желпі алға ұмтылудан тартынбауымыз қажет. Apple App және Google Play Store-да 1,5 миллионнан астам қосымшаның көмегімен мобильді қосымшаларды әзірлеу процесінде жүріп, сіздің бағдарламаңыздың маркетингтік мақсаттарыңызға да, нарықтық тауашаңызға да қалай сәйкес келетінін түсіну маңызды. Мобильді қосымшаларды әзірлеудің өмірлік циклі – бұл мобильді құрылғы тұрғысынан бағдарламалық жасақтаманы әзірлеудің тұрақты өмірлік циклінің (SDLC) көрінісі.

Қазіргі уақытта мобильді қосымшаны құру зымыран туралы ғылым емес. Алайда, сәтті мобильді қосымшаны құру – бұл алдын-ала жоспарлауды қамтитын процесс. Мобильді қосымшаны құру IDE-ді ашу, бірнеше нәрсені біріктіру, жылдам тестілеу және оны қолданбалар дүкеніне жіберу сияқты қарапайым болуы мүмкін, мұның бәрі жарты күн ішінде жасалады. Немесе сіз мұны қатаң алдын-ала жобалау, көптеген құрылғыларда сапаны тексеру, ыңғайлылықты тестілеу, бета-тестілеудің толық өмірлік циклі, содан кейін бірнеше түрлі жолмен орналастыруды қамтитын өте күрделі процесс жасай аласыз. Сіз таңдаған жол сіздің көзқарасыңызға сәйкес келеді. Төменде айтылғандарды ескере отырып, біз қосымшаларды әзірлеудің өмірлік циклын, сондай-ақ осы жолдағы мақсаттар мен міндеттерді қарастырамыз.

Мобильді қосымшаларды әзірлеудің әр жобасы нақты мақсаттардан басталуы керек.

Сіз қандай мақсаттарға қол жеткізуге болатынына көз жеткізіуіңіз маңызды. Сіз қандай мәселелерді шешуге тырысасыз?

Сіздің мақсаттарыңыз, айта келгенде, сіздің дамуыңыздың бүкіл процесін анықтайды. Егер олар басынан бастап нақты анықталмаса, онда сіз өзіңіздің жеке қосымшаңызды жасай бастағаннан кейін жолда адасып кету оңай.

Тек мақсат қоймаңыз – өлшенетін мақсаттар қойыңыз. Бағдарламаны әзірлеу іс жүзінде шексіз болғандықтан, сіздің мақсатыңыз үшін маңызды емес функцияларды, қоңыраулар мен хаттарды қосуға құмар болуы мүмкін.

Бағдарламаны әзірлеуде мақсаттарыңызға жеткеніңізге көз жеткізу үшін мақсат пен тапсырманың арасындағы айырмашылықты түсіну керек. Мақсаттар – сіз қол жеткізгіңіз келетін түпкілікті нәтиже. Мақсаттар – бұл мақсаттарға жету үшін жасайтын қадамдар [22].

Осы мақсаттарды "ақылды" етіп бөліңіз:

– спецификалық;
– өлшенетін;
– қол жетімді;
– шынайы;
– уақтылы.

Бұл адамдарға мақсаттарды ақшалай мағынада анықтау үшін жиі кездесетін қателік, мысалы, " көп ақша табу". Бірақ бұл бекер емес. Бұл сізді планетадағы кез-келген басқа бизнестен ерекшелендірмейді. Оның орнына үлкен мақсат сіздің мақсатыңыз бен миссияңызға бағытталуы керек.

Әр мақсаттың нақты ТНК болуы керек (тиімділіктің негізгі көрсеткіштері).

Табысқа жету үшін алдын-ала анықталған өлшемдеріңіз бар екеніне көз жеткізіңіз. Тиімділіктің негізгі көрсеткіштері сандық болған кезде жақсы жұмыс істейді.

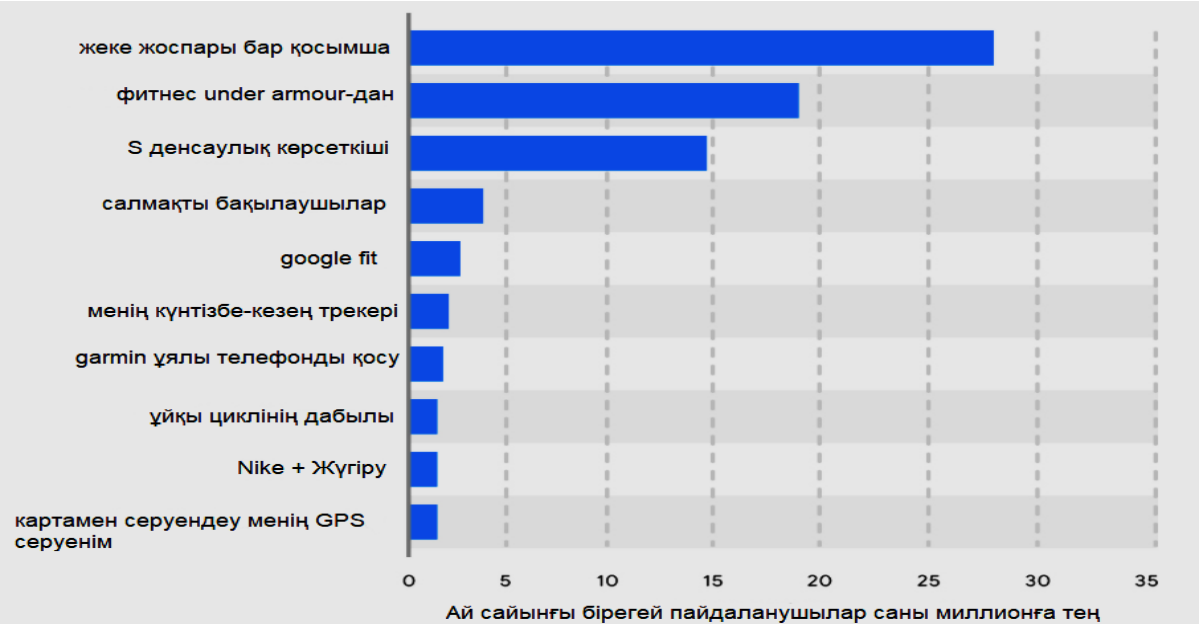
Мысалы, егер сіз электрондық коммерцияға арналған қосымшаны жасасаңыз, мақсаттарыңыздың бірі мобильді құрылғылардағы сатып алудан бас тарту ставкаларын төмендету болуы мүмкін. Осы мақсатқа жеткеніңізге көз жеткізу үшін сіз өзіңіздің бағдарламаңыз іске қосылған кезде оларды сандармен салыстыру үшін ағымдағы секіру жылдамдығын білуіңіз керек.

Маркетингтік зерттеулер жүргізу

Мақсаттарыңызды анықтағаннан кейін, сіздің бағдарламаңызда нарық қажеттілігі бар екеніне көз жеткізуіңіз керек. Әр қосымшаның идеясы теорияда жақсы естіледі, бірақ жалғастырмас бұрын идеяңызды тексеру керек.

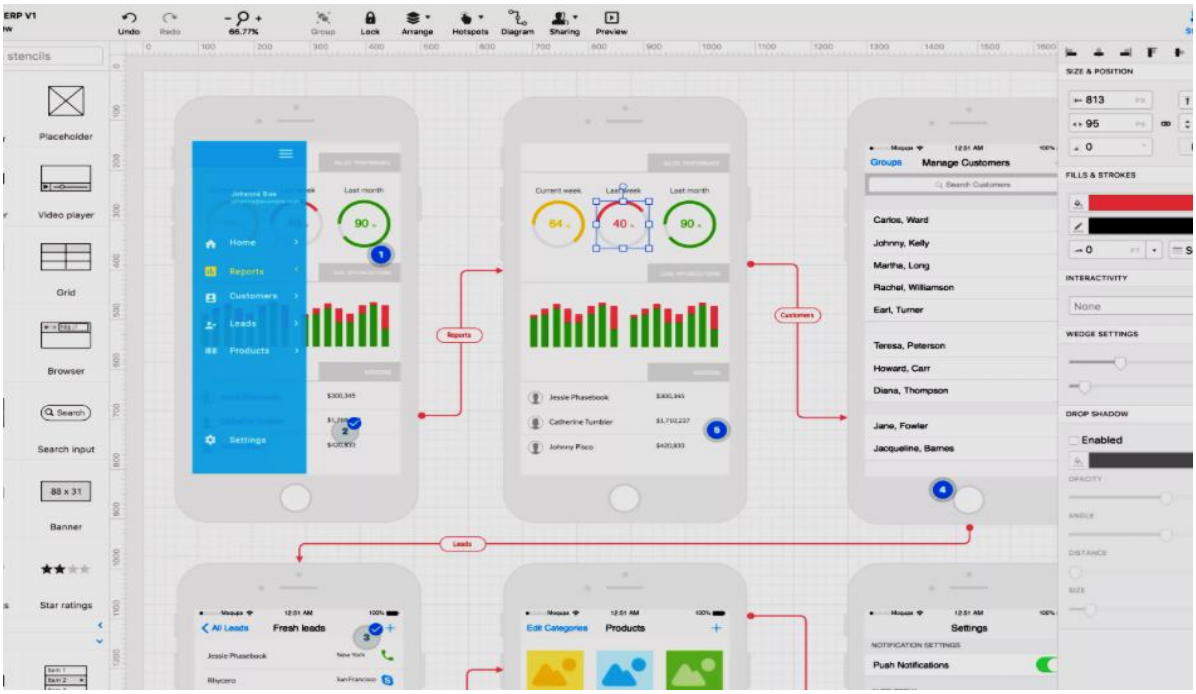
Даму басталғанға дейін дәл қазір маркетингтік зерттеу жүргізу әлдеқайда жақсы. Әйтпесе, сіз нарыққа қажет емес қолданба идеясына қымбат уақытты, ресурстар мен ақшаны жұмсауыңыз мүмкін.

Егер бағдарлама сіздің қолданыстағы бизнесіңізге арналған болса, тұтынушыларыңызбен не қалайтынын түсіну үшін сөйлесіңіз. Сіздің клиенттеріңіз белгілі бір мүмкіндіктерді алғысы келеді деп болжау оңай, бірақ нақты білудің жалғыз жолы – олар туралы көбірек білуге уақыт бөлу [14].



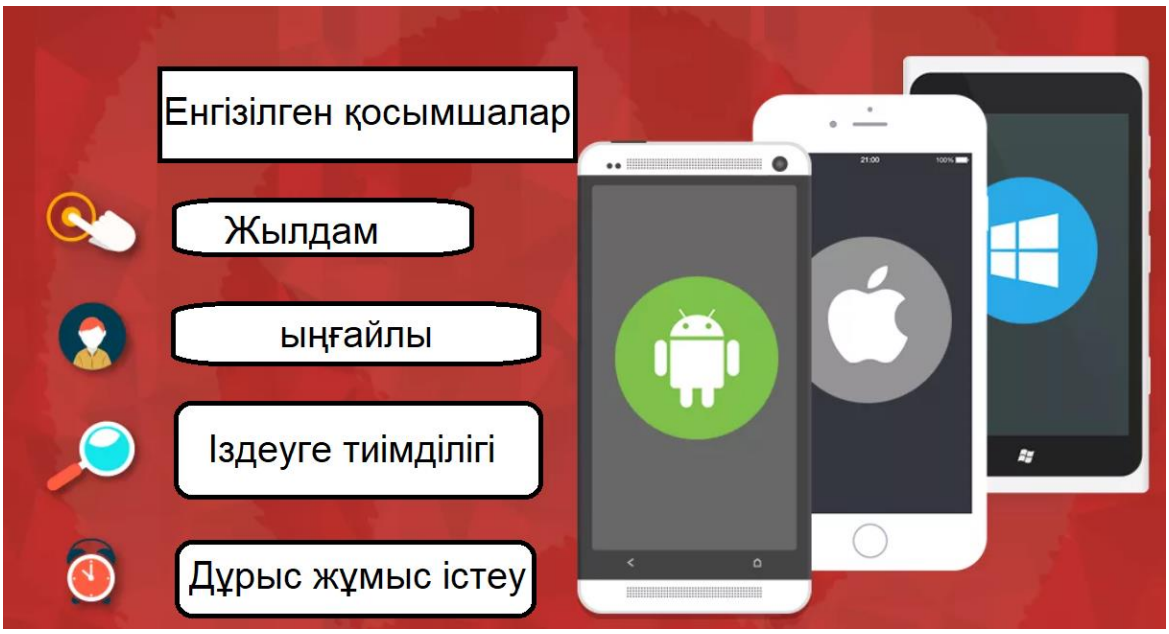
2.1-сурет – Мобильді қосымшаның нарық бойынша анализ келтіру

Егер сіз бұрын мобильді қосымшаны жасамаған болсаңыз, онда сіз раманың не екендігі туралы ойлануыңыз мүмкін. Бақытымызға орай, оны жасау өте оңай.Жақтау – бұл сіздің мобильді қосымшаңыздың өрескел орналасуы. Бұл тым ресми болмауы керек. Сіз қағаз парағында, тақтада, майлықта жақтау жасай аласыз немесе жақтауды жасау үшін сандық құралды қолдана аласыз. Жақтаудың мақсаты – қосымшаның негізгі компоненттерін суреттеу және схемаларды белгілеу. Жақтаудағы қолданбалы дизайн элементтері туралы алаңдамаңыз. Бұл құрал түпкілікті қосымшаның шынымен қалай көрінетіні туралы нақты көзқарас болуға арналмаған. Оның орнына, сіздің жақтауыңыздың бағыты құрылым мен ағынға бағытталуы керек.



2.2-сурет – Жақтаулардың платформаларын құру реті

Пайдаланушының саяхатын картаға салыңыз. Егер олар белгілі бір батырманы басса не болатынын көрсетіңіз. Келесі бет қандай болады? Пайдаланушы А және В опцияларын басқанда экранда не болады? Міне, сым жақтауының маңыздылығын білдіретін ұқсастық. Сіз үй салғыңыз келді делік. Сіз бірінші күні жерді сындырмайсыз ба? Оның орнына сіз қағазға жоспар құрдыңыз. Содан кейін сіз бұл жоспарларды инженерге немесе сәулетшіге апарып, өзіңіз қалаған нәрсені түсіндіре аласыз. Олар бұл жоспарларды алып, оларды сызбаларға сызбалар жасау үшін пайдаланады. Бетон құюды немесе қабырғаларды салуды бастамас бұрын, қағазға осы өзгерістерді енгізу әлдеқайда оңай. Дәл осындай тұжырымдаманы мобильді қосымшаларды әзірлеуге де қолдануға болады. Кез-келген құрылысты бастамас бұрын қағазға өзгеріс енгізу әлдеқайда оңай. Сондай-ақ, жақтау сіздің командаңызға қосымшаның қалай жұмыс істейтіні және жұмыс істеуі туралы көзқарас тұрғысынан бір бетке өтуге көмектеседі [12].



2.3-сурет – Бағдарламаны әзірлеудегі енгізілген қосымшаларға сүйену

Гибридтік даму жергілікті маршрутқа экономикалық тиімді балама болып табылады. Гибридті жинақтардың көпшілігі веб-технология болып табылатын Javascript - ке сүйенеді. Бұл бағдарламалау тілі бағдарламаны бір рет жасауға және оны бірнеше платформаларға орналастыруға мүмкіндік береді. Javascript-те қалай жазуды білетін адам қосымшаны бір рет кодтап, оны iOS және Android-ке орналастыра алады. Гибридті қосымшаның икемділігі әлі де жоғары, Ол не істей алады. Оны жасау үшін сізге төмен деңгейлі кодтауға мамандандырылған әзірлеуші қажет емес, бірақ ол әлі де веб-әзірлеу дағдыларына ие болуы керек. Жалпы, гибридті даму жергілікті дамумен салыстырғанда уақыт пен ақшаны үнемдейді. Алайда, гибридті қосымшаның өнімділігі жергілікті қосымшаның өнімділігінен сәл төмен болады деп күтуге болады. Бақытымызға орай, көптеген адамдар үшін өнімділіктің аздап төмендеуі қосымшаның сәттілігіне әсер етпейді. Бүгінгі интернет жылдамдығында өнімділік айырмашылығы күрт төмендейді [15].

2.1.1 Мобильді қосымшалар және қолданбаларды әзірлеу

Қолданбалар - бұл іскери шешімдер мен қызметтердің нақты іске асырылуы, олар қолда бар кезде қол жетімді - олар пайдаланушыларды шабыттандыруы керек және сонымен бірге мәліметтер мен ресурстардың қауіпсіздігіне зиян келтірмей өнімділікті арттырады. Біз сіздің компанияңыз үшін интеллектуалды және тиімділігі жоғары қосымшаларды жасаймыз!

Қосымша браузерге негізделген қосымшалар жаңа емес. Оларды кез-келген клиентте үнемі қолдануға болады, оларды басқаруға оңай және модульдік кеңейтуге болады. Жұмыс әлемінде смартфондар мен планшеттер сияқты мобильді құрылғылардың көбірек қолданылуымен веб-қосымшалар

көптеген бизнес-процестердің өзегі болып табылады. Қосымшалар ретінде олар ұялы байланыста қолдануға арналған және мысалы, кез-келген жерде, кез-келген уақытта жазуды, салыстыруды немесе пайдалануды жеңілдетеді.

2.2 Мобильді қосымшаны бағдарламалау және деректер базасына шолу жасау

Қысқаша мазмұны – көптеген қол жетімді ұсыныстармен мобильді қосымшаны жасаушыларға қажетті қосымшаның дұрыс дерекқорын таңдау оңай емес. Бұл блог әзірлеушілерге масштабталатын және жылдам мобильді қосымшаларды құруға көмектесетін жалпы өлшемдер мен кейбір прецеденттерге негізделген критерийлерден тұрады Сіз жаңа мобильді қосымшаны жасайсыз ба немесе қолданыстағы бағдарламаға көбірек мүмкіндіктер қосқыңыз келе ме, жоқ па, бүгінгі таңда қол жетімді барлық нұсқаларды ескере отырып, дұрыс дерекқорды таңдау өте қиын болуы мүмкін. Сіздің қазіргі дерекқорыңыз миллиондаған пайдаланушыларды өңдеуге және жиі жаңартуға жеткіліксіз болуы мүмкін [8].

Crisp хабар алмасу платформасы өзінің бастапқы платформасын Firebase-пен мәліметтер базасы ретінде құрды. Бірақ олар тез арада келесі мәселелерге тап болды:

– күрделі сұраныстарды өңдеудегі қиындықтар;
– деректерді офлайн режимде сақтау;
– веб-масштабтағы объект қатынастарын басқару.

Crisp үшін олардың пайдаланушылары өз клиенттерімен тиімді қарым-қатынас жасай алуы өте маңызды болды. Бірақ клиенттер көбінесе бағдарлама арқылы жіберілген маңызды хабарламаларды жіберіп алды. Осылайша, атыс базасы бұл жерде жарамсыз болып шықты.

Кейінірек олар деректерді сақтау үшін SQL дерекқорына және өнімділігі мен масштабталуы үшін хабарламаларды сақтау үшін MongoDB-ге көшті. Crisp – бұл мәліметтер базасы қосымшаның сапасы мен өнімділігін күрт нашарлататын жалғыз оқиға емес.

Үзілістерге тап болған Uber өзінің мәліметтер базасын PostgreSQL-ден MySQL-ге ауыстыруға мәжбүр болды. Uber – бұл таңдауды дерекқорды репликациялаудың тиімсіз функцияларына және PostgreSQL-дің бір нұсқасынан екіншісіне тұрақты көшіп-қонуға тап болды.

Деректер қауіпсіздігі

Синхрондалған және орталықтандырылмаған сақтауды пайдалану кезінде деректердің қауіпсіз қол жетімділігін, берілуін және сақталуын қамтамасыз ету маңызды. Мұны толығымен қамту үшін Сіз аутентификацияға, демалу деректеріне, қозғалыс деректеріне және оқу/жазу қол жетімділігіне жүгінуіңіз керек [9].

2.2.1 Мобильді веб-қосымша және жергілікті қолданба

Негізінде кез-келген мобильді веб-қосымша – бұл мүмкіндіктерге бай жауап беретін веб-сайт. Ол сіздің мобильді құрылғыңыздың шолғышында бар, оны іздеу жүйесі арқылы оңай табуға болады және оны құрылғыға орнатудың қажеті жоқ (бірақ кейбір қосымшалар болуы мүмкін, кейінірек көресіз). Браузерде веб-қосымшалардың болуы олардың Интернетке қосылусыз жұмыс істеуіне немесе мобильді құрылғының мүмкіндіктерін толық пайдалануға мүмкіндік бермейді, бірақ бұл веб-қосымшалардың прогрессивті дамуымен біртіндеп өзгеріп отырады.

Сонымен, жергілікті мобильді қосымша – бұл мақсатты мобильді ОЖ үшін ОЖ-мен үйлесімді тілдерде жазылған және мобильді құрылғының қоймасында орнатылған бағдарламалық жасақтама. Жергілікті қолданбаларға браузер қажет емес және олар оффлайн режимінде жұмыс істей алады. Мобильді веб-қосымшадан айырмашылығы, жергілікті қолданба ең жақсы өнімділікті ұсынады, өйткені барлық ОЖ-ны және аппараттық функцияларды пайдалануға болады.

Мобильді веб-қосымшаның жергілікті қосымшадан айқын артықшылығы – қолданушылар оны кез-келген веб-сайт сияқты Интернеттен оңай таба алады. Сіздің брендіңіз танымал болса да немесе сіздің маркетингіңіз ойластырылған және белсенді болса да, қолданбалар дүкенінде орналасқан сіздің отандық қолданбаңыз ешқашан веб-бағдарламамен бірдей танымалдылыққа ие болмайды.

Дегенмен, дүкендер өздерінің жаңа мазмұнының ашылуын жақсартуға тырысады. Пайдаланушыларға соңғы қолданбалардың белгішелерін басу үшін ұпайлар берілетін кейбір акциялар өткізіледі және қосымшаға / тақырыпқа ақысыз жүктеме сыйақы ретінде беріледі. Алайда, бұл науқанның тиімділігі зерттеліп жатыр, ал оңтайландырылған іздеу жүйесінің экспозициясы сәттілікке кепілдік береді.

Барлық қолданбалар бірдей емес:

–Жергілікті қолданбалар белгілі бір амалдық жүйеге, сондықтан iOS, Android немесе Windows Phone сияқты белгілі бір типтегі құрылғыларға арналған. Сіздің артықшылығыңыз: сіз өте жылдамсыз және камера мен сенсор сияқты құрылғының барлық функцияларын қолданасыз. Олар сыртқы түрі мен құрылымы бойынша амалдық жүйенің қалған бөлігіне сәйкес келетіндіктен, оларды пайдалану оңай және таныс.

–Веб-қосымшалар – бұл қосымшаға ұқсас мобильді веб-парақтар. Олар құрылғыға орнатылмаған, бірақ Интернеттен жүктелген. Бұл орнату қажеттілігін жояды және құрылғыда деректер сақталмайды, сондықтан оны пайдалану өте оңай. Дегенмен, сіз тек құрылғының функцияларын өте шектеулі мөлшерде қолдана аласыз және тек белсенді Интернет қосылымымен жұмыс жасай аласыз. Сіздің үлкен артықшылығыңыз: сіз барлық құрылғыларда жұмыс жасай аласыз және өте қауіпсізсіз. Олар, мысалы, Titanium / Appcelerator,

Phonegapp / Cordova, Xamarin, Ionic немесе SAP Fiori немесе Salesforce (SaaS) сияқты дайын өнімдер сияқты жинақталатын құрылымдармен әзірленген.

–Гибридті қосымшалар жергілікті және веб-қосымшалардың сипаттамаларын біріктіреді және оффлайн веб-қосымшалар болып табылады, демек, олар белсенді интернет байланысынсыз жұмыс істей алады және камера мен GPS сияқты құрылғы мүмкіндіктерін толық пайдалана алады. Сіздің пайдаңыз: олар максималды икемділікті ұсынады, өйткені бұл қосымшалар тек бір рет жасалған және барлық мақсатты платформаларға сәйкес келеді. Жергілікті қолданбалармен салыстырғанда қосымшалар сәл баяу жұмыс істейді және құрылғының әдеттегі көрінісінен азды-көпті ерекшеленеді.

Қолданбалардың барлық түрлерімен сіз қосымшаны және сақталған деректерді, мысалы, шифрланған жадты, қосымшаны контактілермен құлыптауды немесе қолданба жұмыс істейтін өзіңіздің қауіпсіз контейнерді пайдалана отырып, қалған деректер жадынан және құрылғының қосылыстарынан қорғай аласыз.

Жоспарланған бағдарламаға, сондай-ақ қол жетімді инфрақұрылымға және құрылғылардың әртүрлілігіне байланысты, әр жағдайда, қосымшалардың қай түрлерінің мағынасы бар екенін өлшеу қажет. Қажетті мүмкіндіктер жиынтығы, сайып келгенде, қол жетімді стандартты қосымшалардың талаптарға сай келетіндігін немесе өзіңіздің жеке дизайныңызды жасаудың қажет екендігін анықтайды.

2.2.2 Веб-сайтқа қарсы веб-қосымша

Барлығы жұмыс үстелінен басталды, сондықтан оны бірінші орынға қояйық: мобильді қосымшаны қашан Safari, Google Chrome немесе кез-келген басқа жұмыс үстелінің шолғышында ашуға болады? Бұл ең қиын мәселе, өйткені айырмашылық өте аз және веб-сайтта да, веб-қосымшада да нақты көрсеткіштер жоқ.

Дегенмен, көбінесе веб-сайт сізге пассивті ақпарат беретін интернет-ресурс ретінде қаралады, ал кез-келген белсенді пайдаланушының өзара әрекеттестігі (электрондық коммерция, чат модулі, жеке кесте және т.б.) веб-қосымшаны ерекшелендіреді. Мысалы, жұмыс үстелі Википедия веб-сайт болса, жұмыс үстелі Twitter сөзсіз веб-қосымша болып табылады.

Сұр аумақ та өте үлкен, себебі мазмұн пайдаланушының тәжірибесін, сөйлесу жазбаларын, тіркелгі профильдерін және жеке бақылау тақталарын қолдайды. Нью-Йорк Таймс парағына қараңызшы: ол әлі күнге дейін жаңалықтар сайты болғанымен, параметрлер түймесі «жағымды» сезімді тудырады.

Бұл көптеген басқа ақпараттық веб-сайттарға қатысты, өйткені олардың барлығы келушілерді қызықтыруға тырысады және оларға өзара әрекеттесу интерфейсін ұсынады. Екі жылдан кейін Интернет көбіне веб-сайттармен емес, веб-қосымшалармен толығады деп айтуға толық негіз бар.

2.2.3 Мобильді веб-қосымша және мобильді веб-сайт

Түсінікті болу үшін айқын бір нәрсені түсіндірейік. Біз мобильді құрылғылар туралы айтқан кезде де, мобильді веб-сайт пен мобильді веб-қосымшаның арасындағы бұл айырмашылық өзгермейді. Басқаша айтқанда, жауап беретін (мобильді) веб-сайт іс жүзінде интерактивті мүмкіндіктерге ие емес және мобильді құрылғыларда ыңғайлы түрде қарауға болатын мәтіндік немесе мультимедиялық ақпараты бар беттерден тұрады. Сонымен бірге мобильді веб-қосымша тек мобильді құрылғыларға оңтайландырылған парақтарды көрсетуге ғана емес, сонымен қатар мобильді құрылғыларға оңтайландырылған пайдаланушы тәжірибесінің мүмкіндіктеріне арналған.

Ақырында, смартфонда бұрмаланған веб-парақтар (үлкейтілген немесе «бұзылған» орналасуы керек) мобильді құрылғыларға оңтайландырылмаған, сондықтан тек жұмыс үстеліне бағытталған веб-сайттар ретінде жұмыс істейді. Уақыт өте келе азаяды деп үміттенемін.

2.2.4 Мобильді веб-қосымша және гибридті қосымша

«Гибрид» терминінің өзі мобильді веб-қолданба мен жергілікті қосымшаның арасында болатын нәрсені сипаттауға арналған. Веб-қосымша сияқты, бұл түрі браузерде бар (әдеттегіде болмаса да, бірақ орнатылған WebView-те ОЖ-да) және құрылғыға толық және еркін қол жетімділікке ие емес. Жергілікті қолданба сияқты, ол құрылғының қоймасына орнатылуы керек және міндетті түрде интернет байланысының жұмыс істеуі қажет емес.

Сіз мобильді веб-қосымшаның орнына мобильді гибридті қосымшаны дамыта аласыз, негізінен сіз пайдаланушылардан оны өз құрылғыларына орнатуды, оффлайн режимінде жұмыс істеуге және push-хабарландыру жібере алуды сұрай отырып, көбірек пайдаланушыларға қол жеткізгіңіз келсе. Алайда Прогрессивті Веб-қосымшалардың мүмкіндіктері көп ұзамай жаттығуларды толықтай басуы мүмкін.

Жергілікті, веб немесе гибридті: қайсысын таңдау керек? Мен атап өткім келгендей, осы қосымшалардың әрқайсысының артықшылығы мен кемшілігі бар. Осы жерде түйіндейік.

Құрылғының ерекшеліктері:

– Веб-қосымшалар кейбір мүмкіндіктерді пайдалана алатын болса да, жергілікті қолданбалар (және жергілікті mashup компоненттері) құрылғының барлық атрибуттарына, соның ішінде GPS, камера, қимылдар мен хабарландыруларға қол жеткізе алады.

– Автономды жұмыс – егер сіздің қосымшаңыз байланыссыз жұмыс істеуі қажет болса, жергілікті бағдарлама жақсы болады. Браузерді кэштеу HTML5-те қол жетімді, бірақ ол түпнұсқаға ауысқанда алатыннан гөрі шектеулі.

– Анықталуы – веб-қосымшалар табуға болатын сыйлық алады. Мазмұнды қосымшадан гөрі желіден табу әлдеқайда оңай: адамдарда сұрақ

туындаған кезде немесе ақпарат қажет болғанда, олар іздеу жүйесіне кіріп, сұранысын енгізіп, іздеу нәтижелерінен парақ таңдайды. Олар қолданбалар дүкеніне бармайды, қосымшаны іздейді, жүктемейді, содан кейін қосымшадан жауап іздеуге тырыспайды. Қолданбалар дүкендерінен қосымшаларды іздей алатын қосымшалардың жанкүйерлері болғанымен, көптеген қолданушылар қолданбаларды орнатуды және күтіп ұстауды ұнатпайды (және олардың құрылғысында бос орынды ысыраптайды) және қолданбаны оны жиі қолдануды жоспарлаған жағдайда ғана орнатады.

– Жылдамдық – жергілікті қосымшалар жылдамдық бәсекесінде жеңіске жетеді. 2012 жылы Марк Цукерберг Facebook-тің ең үлкен қателігі отандыққа емес, ұялы интернетке бәс қою деп мәлімдеді. Осы уақытқа дейін Facebook қосымшасы HTML ядросы бар масх болды; 2012 жылы ол нағыз отандық қосымшамен ауыстырылды. Жауаптылық - бұл ыңғайлылықтың кілті.

– Орнату – жергілікті немесе гибридті қосымшаны орнату пайдаланушылар үшін қиындық тудырады: олар өзара әрекеттесу шығындарын негіздеу үшін шынымен ынталандырылуы керек. Веб-қосымшаны «орнату» басты экранда бетбелгі құруды қамтиды; бұл процесс қосымшалар дүкенінен жаңа бағдарламаны жүктеуге қарағанда оңайырақ, бірақ қолданушыларға онша таныс емес, өйткені адамдар мобильді құрылғыларда бетбелгілерді жиі қолданбайды.

– Сервис – жергілікті қолданбаны жүргізу тек қолданушылар үшін ғана емес, сонымен қатар әзірлеушілер үшін де қиынға соғуы мүмкін (әсіресе, егер олар әр түрлі платформаларда бір ақпараттың бірнеше нұсқасымен жұмыс істеуге мәжбүр болса): өзгерістер жаңа нұсқаға оралып, қолданба дүкеніне орналастырылуы керек. ... Екінші жағынан, веб-қосымшаны немесе mashup-ті ұстау веб-парақты жүргізу сияқты оңай, және сіз мұны жиі немесе қажет болған жағдайда жасай аласыз.

– Платформаның тәуелсіздігі – әр түрлі браузерлер HTML5-тің әр түрлі нұсқаларын қолдай алатын болса да, егер платформаның тәуелсіздігі маңызды болса, сіз бұған жергілікті қолданбаларға қарағанда веб пен масхап арқылы жетуіңіз мүмкін. Бұрын талқыланғандай, гибридті немесе веб-қосымшаларды құру кезінде кодтың кем дегенде бір бөлігін қайта пайдалануға болады.

Мазмұн бойынша шектеулер, мақұлдау процесі және төлемдер. Сіздің мазмұныңыз бен дизайныңызға ережелер енгізетін үшінші тараппен жұмыс уақытты да, қымбатты да алады. Жергілікті және гибридті қосымшалар қосымшалар дүкендері қабылдаған мақұлдау мен мазмұнды шектеу процестерінен өтуі керек, ал Интернет барлық адамдар үшін ақысыз. Таңқаларлық емес, алғашқы веб-қосымшалар Playboy сияқты басылымдарда пайда болды, олар Apple-дің мазмұнына қатысты цензурадан аулақ болғысы келді. IOS қосымшасында жазылым сатып алу жазылым бағасының 30% Apple-ге тиесілі дегенді білдіреді, бұл баспагерлердің бюджеттеріне айтарлықтай әсер етеді.

Әзірлеу құны. Гибридті және веб-қосымшаларды жасау арзанырақ болуы мүмкін, өйткені бұған дейінгі веб-тәжірибеге негізделген дағдылар қажет. NN /

g клиенттері көбінесе ана тіліне көшу әлдеқайда қымбат деп санайды, өйткені ол арнайы мамандандырылған қабілетті қажет етеді. Екінші жағынан, HTML5 салыстырмалы түрде жаңа болып табылады, және оны жақсы білу, сондай-ақ мобильді веб пен машупты дамыту туралы жақсы түсінік - бұл да өте жақсы дағдылар.

Пайдаланушы интерфейсі. Сонымен, егер сіздің басымдықтарыңыздың бірі амалдық жүйеге және осы платформадағы басқа қосымшалардың көпшілігіне сәйкес келетін пайдаланушы тәжірибесін қамтамасыз ету болса, онда жергілікті бағдарламалар сіз үшін. Бұл сізде мобильді пайдаланушының веб немесе mashup қосымшасында жақсы тәжірибе ұсына алмайтыныңызды білдірмейді - бұл графика мен көрнекіліктер қолданушылар бұрыннан үйреніп алған суреттермен бірдей болмайтынын және солай болатынын білдіреді. мобильді технологиялардың артықшылықтарын пайдалану және ұялы байланыс шектеулерін азайту қиынырақ.

3 Жобаны жүзеге асыру және тестілеу бөлімі

3.1 Программалық құралдар сипаттамалары

Бұл мобильді қосымша негізгі 4 активити, 7 фрагмент және 2 алерттен тұрады. Қолданылған мәліметтер базасы онлайн SQL базасы мен локалды SQLite базасы, ал базаларда сақталатын объекттер саны 4. Негізгі активитилер MainActivity, LogInActivity және RegistrationActivity, ал төртінші SplashActivity мобильді қосымшаны бастау үшін арналған.

SplashActivity өңдегішке уақытты орнатамыз және өңдеушіні шақырамыз ().postDelayed, ол белгіленген уақыттан кейін runnable-дан run әдісін шақырады және бағдарламаның негізгі экранына қайта бағытталады.

Өңдегіштер – бұл негізінен пайдаланушы интерфейсінің ағынымен өзара әрекеттесуге мүмкіндік беретін фондық ағындар (пайдаланушы интерфейсін жаңарту).

Өңдеушілер – бұл Android өңдегіш класының ішкі сыныптары және оны ағын қажет болған кезде орындалатын нысанды көрсету арқылы немесе өңдегішке хабарлама жіберген кезде шақырылатын өңдегіштің ішкі сыныбындағы handleMessage() қоңырау шалу әдісін қайта анықтау арқылы пайдалануға болады.

Өңдегішті қолданудың екі негізгі әдісі бар:

а) болашақта қандай да бір сәтте хабарламалар мен орындалатын файлдардың орындалуын жоспарлау;

б) кезекке қою – бұл өз ісіңізден басқа ағымда орындалуы керек әрекет.

Бұл активтиде маңызды операциялар орындалмайды, тек логотип шығарып, 0,5 секундтан соң мобильді қосымшаның жұмысын бастау үшін MainActivity ді ашып береді.

```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_splash);
    new Handler().postDelayed(this::startApplication,  delayMillis: 500);
}

private void startApplication() {
    startActivity(new Intent( packageContext: this, MainActivity.class));
    finish();
}
```

3.1-сурет – Жаңа активити бастау операциясы

Мобильді қосымшаларды пайдалану тәжірибесі басқалардан ерекшеленеді, өйткені пайдаланушының қосымшамен өзара әрекеттесуі әрқашан бастапқы экраннан басталмайды. Мысалы, егер пайдаланушы Gmail қосымшасын іске қосса, олар электрондық пошталардың тізімін көреді. Алайда, егер ол Gmail-ді іске қосатын әлеуметтік медиа қосымшаларын қолданса, пайдаланушы тікелей электрондық поштаны жіберу экранына кіре алады.

Әрекет класы осы парадигманы жеңілдетуге арналған. Бір қолданба басқасын шақырған кезде, қоңырау шалушы бағдарлама қосымшаның өзі емес, басқа қосымшаның әрекетін бастайды. Осылайша, әрекет қосымшаның қолданушымен өзара әрекеттесуі үшін кіру нүктесі ретінде қызмет етеді.

Әр түрлі іс-шаралар бар, бірақ олардың барлығы қандай да бір жолмен негізгі белсенділік класының мұрагерлері болып табылады. Мысалы, егер қолдау кітапханасы пайдаланылса, жаңа жобаны жасау кезінде Android Studio AppCompatAcitivty-ден мұра болатын MainActivity класын жасайды. Егер сіз мұрагерлік иерархиясын қарасаңыз, нәтижесінде сіз осы иерархияның басында әрекет класы болатындығын көре аласыз.

Әрекет – бұл бағдарлама өзінің UI-ді сызатын терезе. Бұл терезе әдетте экранды толығымен толтырады, бірақ экраннан аз болуы мүмкін немесе басқа терезелердің үстінде болуы мүмкін. Әдетте, бір әрекет – бұл қолданбадағы бір экран. Мысалы, бір әрекет параметрлер терезесін, ал екіншісі фотосуреттерді таңдау терезесін іске асыра алады.

Көптеген қосымшаларда бірнеше экрандар бар, олардан бірнеше іс-шаралар бар деп қорытынды жасауға болады. Әдетте, іс-шаралардың бірі негізгі ретінде анықталады және қолданушы қосымшаны іске қосқан кезде пайда болатын алғашқы экран болып табылады. Содан кейін әр әрекет кез-келген әрекетті орындау үшін басқа әрекетті бастай алады. Gmail мысалында негізгі әрекет электрондық пошталардың тізімі бар экран екенін көруге болады. Осы әрекеттен сіз жаңа хаттарды немесе кез-келген басқа хаттарды жасауға және жіберуге жауап беретін әрекетті бастай аласыз.

Әрекеттер қолданбада жақсы пайдаланушы интерфейсін ұйымдастыру үшін бірге жұмыс істесе де, әр әрекет басқалармен тығыз байланысты: әдетте қолданбада іс-шаралар арасында минималды байланыс бар. Сонымен қатар, әрекеттер көбінесе басқа қосымшалардың әрекеттерін бастайды. Мысалы, веб-шолғыш қызметі әлеуметтік желі қосымшасының қызметін іске қосуы мүмкін.

Қосымшаға жаңа әрекетті қалай қосу керектігін мысалмен қарастырайық. Android Studio-да жаңа бос жоба жасаңыз. Әдепкі бойынша, бағдарлама MainActivity деп аталатын бір негізгі әрекетті жасайды. Бұл жағдайда жобада келесі әрекеттер орын алады.

Сол атаудағы MainActivity класы құрылды.java. Бұл сыныпта қызметтің барлық жұмысы жүзеге асырылады, оны құру кезінде oncreate() әдісі қайта анықталған және белсенділік таңбаларын жүктеу бар.

MainActivity кезектегі екінші активити болып саналады. MainActivity 7 фрагменттен тұрады, және активитиде фрагменттерді ауыстыруға арналған батырмалар бөлімі көрсетілген. Ол төменгі батырмалар түрінің 2 түрі бар.

Біріншісі мобильді қосымшаларға тіркелген қолданушыларға, ал екіншісі тіркелмеген қолданушыларға арналған. Мобильді қосымшада тіркелген немесе тіркелмегенін Асинхрондық тапсырмада локалды файлдан тексереміз.

```
private class CheckIsAuthorized extends AsyncTask<Context, Void, Boolean> {

    @Override
    protected Boolean doInBackground(Context... contexts) {
        try {
            ObjectInputStream inputStream = new ObjectInputStream(
                    contexts[0].openFileInput( name: "login.dat"));
            user = (User) inputStream.readObject();
            inputStream.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return user != null;
    }

    @Override
    protected void onPostExecute(Boolean aBoolean) {
        if (aBoolean) {
            nonAuthorizedBarLinear.setVisibility(View.INVISIBLE);
            authorizedBarLinear.setVisibility(View.VISIBLE);
        } else {
            authorizedBarLinear.setVisibility(View.INVISIBLE);
            nonAuthorizedBarLinear.setVisibility(View.VISIBLE);
        }
    }
}
```

3.2-сурет – Қолданушы тіркелген немесе тіркелмегенін тексеру

```
private void initNonAuthorized() {
    ((LinearLayout) findViewById(R.id.search_job_nauth__linear)).setOnClickListener
            (v -> changeContentFragment(1));
    ((LinearLayout) findViewById(R.id.favourites_job_nauth__linear)).setOnClickListener
            (v -> changeContentFragment(2));
    ((LinearLayout) findViewById(R.id.log_in_nauth__linear)).setOnClickListener
            (v -> startLogInActivity());
}

private void initAuthorized() {
    ((LinearLayout) findViewById(R.id.search_job_auth__linear)).setOnClickListener
            (v -> changeContentFragment(1));
    ((LinearLayout) findViewById(R.id.favourites_job_auth__linear)).setOnClickListener
            (v -> changeContentFragment(2));
    ((LinearLayout) findViewById(R.id.inquiries_auth__linear)).setOnClickListener
            (v -> changeContentFragment(3));
    ((LinearLayout) findViewById(R.id.account_auth__linear)).setOnClickListener
            (v -> changeContentFragment(4));
}
```

3.3-сурет – Батырмаларды басқандағы операциялардың орындалуын тексеру

3.2 Қосымшада барлық қолданылатын Fragment-терге талдау

Фрагменттің ішінде іздеу интерфейсін құрудың бірнеше себептері бар.

Іздеу мүмкіндігі бар интерфейсті құру кезінде Сіз Android Манифестінде әдепкі бойынша "іздеу мүмкіндігі бар әрекетті" көрсетуіңіз керек. Өздеріңіз

білетіндей, Фрагмент ата-аналық белсенділіксіз өмір сүре алмайды, сондықтан бұл бөлу мүмкін емес.

Пайдаланушы іздеу тілқатысу терезесінде немесе виджетте іздегенде, жүйе іздеу әрекетін іске қосады және оған ACTION_SEARCH әрекеті ниетіне сәйкес іздеу сұрауын жібереді. Іздеу әрекеті қосымша ниет сұрауынан сұрау шығарады, содан кейін деректеріңізді іздейді және нәтижелерді ұсынады.

Іздеу нәтижелерін беруге жауап беретін негізгі Ішкі жүйе фрагментті емес, әрекетті күтеді; осылайша, әрекетке мүлдем тәуелсіз іздеу интерфейсін іске асыру мүмкін емес, өйткені бұл Негізгі жүйенің өзін өзгертуді қажет етеді. Егер маған сенбесеңіз, SearchableInfo класының бастапқы кодын тексеріңіз:).

Дегенмен, сипаттаған нәрсеге қол жеткізу өте қиын болар еді. Мысалы, Android-ді қабылдау үшін іздеу әрекетін іске асыруды қарастырғыңыз келуі мүмкін.intent.action.Іздеу мақсаты және (мысалы, нәтижелерді тізім ретінде дереу көрсетудің орнына) іздеу сұрауын үзінділеріңізге жібереді. Мысалы, іздеу мүмкіндігі бар келесі әрекетті қарастырыңыз:

```
@SuppressLint({"CommitTransaction", "ResourceType"})
private void changeContentFragment(final int content) {
    Fragment fragment;
    Bundle bundle = new Bundle();
    switch (content) {
        case 1:
            fragment = new SearchJobFragment();
            break;
        case 2:
            fragment = new FavouriteFragment();
            break;
        case 3: {
            fragment = new RequestsFragment();
            bundle.putInt(Messages.USAGE.name(), 1);
            break;
        }
        default:
            fragment = new AccountFragment();
    }
    if (content != 1) {
        bundle.putSerializable(Messages.USER.name(), user);
        fragment.setArguments(bundle);
    }
    FragmentManager fragmentManager = getFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.main__fragment, fragment);
    fragmentTransaction.commit();
}
```

3.4-сурет – Әр бір батырмаға арналған фрагменттер

Ал бұл метод, сол әр бір батырмаға арналған фрагменттерді ауыстыруға жауап береді. Негізгі фрагменттер ретінді 4 фрагмент қолданылады. Олар: SearchFragment, FavouriteFragment, RequestsFragment, AccountFragment.

```
@SuppressLint("StaticFieldLeak")
private class GetRecommendationsTask extends AsyncTask<Void, Void, ArrayList<Job>> {
    @Override
    protected ArrayList<Job> doInBackground(Void... voids) {
        ArrayList<Job> jobs = dbManager.getRecommendations();
        for (int i = 0; i < jobs.size(); i++) {
            if (user != null) {
                if (jobs.get(i).getOwnerId() == user.getId()) {
                    jobs.remove(i);
                    i--;
                    continue;
                }
            }
            jobs.get(i).setFavourite(dbManager.isFavourite(jobs.get(i).getId(), getActivity()));
        }
        return jobs;
    }

    @Override
    protected void onPostExecute(ArrayList<Job> jobs) {
        if (jobs.isEmpty()) {
            emptyListTxtView.setVisibility(View.VISIBLE);
        } else {
            jobsRecycler.setVisibility(View.VISIBLE);
            jobsRecycler.setAdapter(new JobAdapter(getActivity(), jobs, user));
        }
        loadProgressBar.setVisibility(View.INVISIBLE);
    }
}
```

3.5-сурет – Қолданушыларға ұсынылатын жұмыстар тізімін шығару

Бұл скриншотта қолданушыларға ұсынылатын жұмыстар тізімін шығаруға арналған асинхрондық тапсырманы көре аламыз. Фонда орындалатын тапсырмада, мәліметтер базасынын ұсынылатын жұмыстар тізімін алып, олардың арасынан қолданушы өзі ұсынған жұмыстарды жоямыз. Ал посттан кейінгі тапсырмада, сол тізімді қолданушыға көрсетеміз.

```
@SuppressLint("StaticFieldLeak")
private class AutoCompleteTask extends AsyncTask<String, Void, ArrayList<Job>> {

    @Override
    protected ArrayList<Job> doInBackground(String... strings) {
        ArrayList<Job> jobs = strings[0].replaceAll( regex: " ",  replacement: "").length() == 0 ?
                dbManager.getRecommendations() : dbManager.autoComplete(strings[0]);
        if (user != null) {
            for (int i = 0; i < jobs.size(); i++) {
                if (jobs.get(i).getOwnerId() == user.getId()) {
                    jobs.remove(i);
                    i--;
                }
            }
        }
        return jobs;
    }

    @Override
    protected void onPostExecute(ArrayList<Job> jobs) { setSearchResult(jobs); }
}
```

3.6-сурет – Базадан жұмыстарді іздеу асинхрондық тапсырмасы

JobFragment жұмыс туралы толық ақпарат көрсетуге, және жұмысқа қолданушы туралы ақпарат жіберу үшін арналған, егер қолданушыға ол жұмыс сай келетін болса әрине.

```
@SuppressLint("StaticFieldLeak")
class SendResumeTask extends AsyncTask<Void, Void, Void> {

    @Override
    protected Void doInBackground(Void... voids) {
        dbManager.sendResume(new Candidate( id: -1, job.getOwnerId(), user, job,
                resumeEdtTxt.getText().toString()));
        return null;
    }

    @Override
    protected void onPostExecute(Void aVoid) {
        closeDialog();
        Toast.makeText(getActivity(), "Your data has been sent!",
                Toast.LENGTH_SHORT).show();
        back();
    }
}
```

3.7-сурет – Асинхрондық тапсырмада қолданушы туралы ақпаратты жіберу

3.7-суретте көрсетілгендей асинхрондық тапсырмада қолданушы туралы ақпаратты жіберу операциясы орындалғанын көре аламыз. Тапсырманы жіберу барысында жұмысқа LoadingAlert қосылады, операция біткеннен кейін LoadingAlert ті жабамыз. Жіберілген ақпарат, мәлеметтер базасында сақталады. Сіз бүктеп, суреттің жалпы көрінісін ала алатын жұмбақты елестетіп көріңіз. Немесе егер сіз сайттарды, атап айтқанда PHP-ді дамытқан болсаңыз, онда сіз сайттың жалпы бөліктерін, мысалы, мәзірді include көмегімен бөлек блок ретінде қосу мүмкіндігімен таныссыз. Өйткені анықтау емес, түсінікті, шештім деп айтуға өз сөзімен соң.

Фрагмент (Фрагмент) – бұл іс жүзінде қосымшаның әртүрлі бөліктеріне қосыла алатын Activity сияқты. Бірақ бір әрекетте бірнеше фрагмент болуы мүмкін.

FavouriteFragment кезектегі екінші басты фрагмент. Бұл фрагмент тек қолданушылардың сақталған жұмыстарын көрсету үшін арналған.

Мұндай дизайн әдетке айналады және жоба жаңа activity-мен күрделене түседі. Нәтижесінде, тіпті ең аз пайдалы бағдарлама барлық жедел жадты жоятын activity стекімен толып кетеді, ал Google Play-де тастар мен шешімдер әзірлеушіге ұшады.

UI – мен жұмыс істеу оңай және жылдам болуы үшін Google фрагментті (фрагмент) – белсенділік пен бағдарламаның визуалды компоненттері арасындағы сынып қабатын жасады. Бір жағынан, бұл пайдаланушыға көрсетілетін кез-келген көрініс нысандарына арналған контейнер. Екінші

жағынан, әрекет етудің жалғасы, одан фрагмент өмірлік циклдегі өзгерістер туралы барлық ақпаратты алады.

Бүгін мен сіздермен әйгілі "құпиямен" бөліскім келеді: әрекет жаппай қолдануға арналмаған. Керісінше, қосымшада бұл тек төтенше жағдайларда қолданылатын құрал. Жан-жақты генерация жаңа Activity жасайды елеулі проблемалар, олар жұмысқа қосымшалар болжамсыз. Егер сіздің құрылғыңызда бәрі тұрақты болса да, әлемде Android құрылғыларының саны мүмкін емес, олардың көпшілігінде сіздің қосымшаңыз құлап кетеді.

Мұның бәрі, өйткені Android ОЖ сіздің белсенділігіңізді тірі қалдыруға уәде бермейді. Есіңде болсын, бұл компоненттер бір-бірінен тәуелсіз, ерекше өмірлік циклі бар. Егер әрекет onPause күйіне ауысса және бұл жиі орын алса, ол Шредингердің мысығына айналады: сіз оның тірі немесе жоқ екенін алдын-ала біле алмайсыз.

Мәзір мен басқа да ұсақ-түйектерді шығару үшін қосымша әрекеттерді қолдана отырып, сіз қолданба ішіндегі барлық логикалық байланыстарға қауіп төндіресіз. Әрекет стекке жиналады, ал ОЖ оларды бір-бірден немесе топпен жүктей бастайды. Пайдаланушы алдыңғы терезеге оралғысы келгенде, әрекет жойылуы мүмкін және пайдаланушы бағдарламадан шығады.

Логиканы сақтау проблемаларынан басқа, ООР кодты қолдау тәртібі де бар: Activity-ге тығыз байланған интерфейстерді одан әрі дамыту мүмкін емес. Масштабтау, кейбір элементтерді басқаларына тез ауыстыру, анимация – мұның бәрін қосымшаның жаңа нұсқаларында орындау өте қиын болады.

Бірыңғай экожүйеге сәннің арқасында барлық мобильді қосымшалар өте ұқсас болды. Сіз тек жолды сезініп, аздап жаттығуыңыз керек, содан кейін сапа тез өсе бастайды. Сонымен, бастайық!

```
@SuppressLint("StaticFieldLeak")
private class GetFavouriteTask extends AsyncTask<Void, Void, ArrayList<Job>> {
    @Override
    protected ArrayList<Job> doInBackground(Void... voids) {
        ArrayList<Job> jobs = dbManager.getFavourite(getActivity());
        for (int i = 0; i < jobs.size(); i++) {
            if (user != null) {
                if (jobs.get(i).getOwnerId() == user.getId()) {
                    jobs.remove(i);
                    i--;
                }
            }
        }
        return jobs;
    }

    @Override
    protected void onPostExecute(ArrayList<Job> jobs) {
        if (jobs.isEmpty()) {
            emptyListTxtView.setVisibility(View.VISIBLE);
        } else {
            jobsRecycler.setVisibility(View.VISIBLE);
            jobsRecycler.setAdapter(new JobAdapter(getActivity(), jobs, user));
        }
        loadProgressBar.setVisibility(View.INVISIBLE);
    }
}
```

3.8-сурет – Локалды базадан барлық сақталған жұмыстардың тізімі

RequestsFragment кезектегі үшінші басты фрагмент, және фрагмент мобильді қосымшадағы ең үлкен фрагмент болып саналады. Бұл фрагментте жоғарғы батырмалар бөлімі мен тізім бар. Батырмалар қолданушы өзі туралы жіберген жұмыстан келген жауаптар, қолданушының жұмысына жіберілген қолданушылар және қолданушы өзі салған жұмыстардың тізімін көрсетуге арналған, және фрагмент осындай үш бөлімнен тұрады. Әр бір батырманы басқан сәтте жаңа бөлімдер басталып отырады.

```
feedbackTitle.setOnClickListener(v -> startFeedback());
candidatesTitle.setOnClickListener(v -> startCandidates());
vacanciesTitle.setOnClickListener(v -> startVacancies());
```

3.9-сурет – Жаңа бармалар басу арқалы жаңа бөлімге көшу

ToAnswerFragment қолданушылардың салған жұмыстарына кандидаттарға жауап жазуға арналған. Бұл фрагментте қолданушы кандидат туралы ақпаратты көре алады, және жауап жазып, қабылданды немесе қабылданбады статусын белгілеп жауап жібере алады.

```
@SuppressLint("StaticFieldLeak")
class SendAnswerTask extends AsyncTask<Void, Void, Void> {

    @Override
    protected Void doInBackground(Void... voids) {
        dbManager.sendAnswer(new FeedBack( id: -1, candidate.getUser().getId(),
                job, status, answerEdtTxt.getText().toString()), candidate.getId());
        return null;
    }

    @Override
    protected void onPostExecute(Void aVoid) { closeDialog(); }
}
```

3.10-сурет – Жауап қайтару бөлімі

Жауап жіберу тапсырмасы асинхрондық тапсырма. Скриншотта базаға жауапты сақтау операциясы көрсетілген. Операция бастамастан бұрын LoadingAlert іске қосылып, операция біткеннен кейін жабылады.

JobEditorFragment мобильді қосымшада екі мақсатта қолданылады. Қолданушы жариялайтын жұмыс туралы ақпарат енгізу мақсатында және жұмыс туралы ақпаратты жаңарту мақсатында.

AccountFragment қолданушы өзі туралы ақпаратты жаңарту үшін қолданылады. Фрагмент басталған сәтте қолданушы турал ақпарат мәтен

өрістеріне жазылады. Және сақтау батырмасын басқанда, жаңа ақпарат базада сақталады.

LogInActivity қолданушы мобильді қосымшаға кіру үшін қолданылады. Егер қолданушыда аккаунт жоқ болса, қолданушыға тіркелуге арналған жаңа активити басталады.

RegistrationActivity қолданушы тіркелуге арналған аквтивити. Қолданушы мәтін өрістерін толтырып өзі турал ақпарат жазып, тіркелу батырмасын басқанда, тіркелу операциясы басталды.

3.3 JobAdapter, LoadingAlert, RequestAdapter, AnswerMessageAlert бағдарламалық өңделуі

Бұл адаптер RecyclerView ге, яғни тізімге элементтерді шығару үшін қолданылады. Тізімнен жұмысты таңдап оны басқан сәтте жаңа фрагмент басталды.

```
private void openProfile(Job job) {
    if (user != null) {
        Fragment fragment = new JobFragment();
        Bundle bundle = new Bundle();
        bundle.putSerializable(Messages.JOB.name(), job);
        bundle.putSerializable(Messages.USER.name(), user);
        fragment.setArguments(bundle);
        FragmentManager fragmentManager = activity.getFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.main__fragment, fragment);
        fragmentTransaction.commit();
    } else {
        activity.startActivity(new Intent(activity, LogInActivity.class));
        activity.finish();
        activity.overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_left);
    }
}
```

3.11-сурет – Жұмысты қабылдап алу кезеңі

Бұл метод таңдалған жұмысты қабылдап алып бізге сол жұмыс туралы ақпарат көрсететін JobFragment ті ашып береді. Ал егер қолданушы тіркелмеген болса, тіркелуге арналған жаңа активитиді бастайды. Жаңа фрагментті бастау кезінді жұмысты bundle ге салып, фрагментті бастауға арналған аргументтер тізіміне саламыз. Бұл әдіс арқылы, таңдалған жұмысты сол жұмыс туралы ақпарат көріге арналған фрагментке жібереміз.

StartDialog алертті ашу үшін, және closeDialog алертті жабу үшін қолданылады. Ал startTask асинхрондық тапсырмаларды бастау үшін қолданылған. Бұл метод полиморфизм принципі арқылы өзгертілетін болады, және анонимді класста LoadingAlert ті кеңейту арқылы startTask арқылы басталатын жаңа асинхрондық тапсырманы қоса аламыз. Асинхрондық тапсырма біткеннен кейін мендетті түрде алертті жабамыз.

```
holder.jobLinear.setOnLongClickListener(v -> {
    BottomSheetDialog bottomSheetDialog = new BottomSheetDialog(activity,
            R.style.bottomSheetDialogTheme);
    View bottomView = LayoutInflater.from(activity.getApplicationContext()).
            inflate(R.layout.bottom_sheet__v,
                    activity.findViewById(R.id.bottom_sheet_job__container));
    TextView open = bottomView.findViewById(R.id.bottom_sheet_first__txt_v),
            favourite = bottomView.findViewById(R.id.bottom_sheet_second__txt_v);
    open.setText("Open profile");
    open.setOnClickListener(v1 -> {
        openProfile(job);
        bottomSheetDialog.cancel();
    });
    favourite.setText(!jobs.get(position).isFavourite() ?
            "Save to favorites":
            "Remove from favorites");
    favourite.setOnClickListener(v1 -> {
        if (jobs.get(position).isFavourite()) {
            dbManager.removeFromFavourite(job.getId(), activity);
        } else {
            dbManager.saveToFavourite(job.getId(), activity);
        }
        jobs.get(position).setFavourite(!jobs.get(position).isFavourite());
        bottomSheetDialog.cancel();
    });
    bottomSheetDialog.setContentView(bottomView);
    bottomSheetDialog.show();
    return true;
});
```

3.12-сурет – Жұмысты басып тұру арқылы келесі парақшаға өту

```
public class LoadingAlert {

    private final Activity activity;
    private AlertDialog alertDialog;

    public LoadingAlert(Activity activity) { this.activity = activity; }

    @SuppressLint("InflateParams")
    public void startDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(activity);
        LayoutInflater inflater = activity.getLayoutInflater();
        builder.setView(inflater.inflate(R.layout.loading__alert,  root: null));
        builder.setCancelable(false);
        alertDialog = builder.create();
        alertDialog.show();
        alertDialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
        startTask();
    }

    public void closeDialog() {
        alertDialog.dismiss();
    }

    public void startTask() {

    }
}
```

3.13-сурет – Күту alert

Бұл alert 3 түрлі методтан тұрады, startDialog, closeDialog, startTask.

RequestAdapter мобильді қосымшада үш түрлі мақсатта RequestFragment те қолданылады. Адаптер шығаруға арналған тізім мен қолданылу мақсатына қарай өзгереді.

Бірінші адаптер жауаптар тізімін шығару мақсатында қолданылса, статусты көрсетеміз. Ол статус қабылданды немесе қабылданбады болады. Ал тізімдегі бір жауапты таңдап басқан сәтте жауапты үлкейтіп көрсетуге арналған

алерт ашылады және сол алерттегі жою батырмасын басса асинхрондық тапсырма басталып базадан жауапты жою операциясы орындалады. Операцияны төмендегі скриншоттан көре аласыздар.

```
AnswerMessageAlert answerMessageAlert = new AnswerMessageAlert(activity) {
    @Override
    public void deleteMessage() {
        closeDialog();
        LoadingAlert loadingAlert = new LoadingAlert(activity) {

            @SuppressLint("StaticFieldLeak")
            class DeleteTask extends AsyncTask<Void, Void, Void> {

                @Override
                protected Void doInBackground(Void... voids) {
                    dbManager.deleteFeedback(feedBacks.get(position).getId());
                    return null;
                }

                @Override
                protected void onPostExecute(Void aVoid) {
                    closeDialog();
                    fragment.startFeedback();
                }
            }

            @Override
            public void startTask() { new DeleteTask().execute(); }
        };
        loadingAlert.startDialog();
    }
};
holder.contentLinear.setOnClickListener(v ->
        answerMessageAlert.startDialog(feedBack.getAnswer()));
```

3.14-сурет – Статус тексеру адаптері

```
Candidate candidate = candidates.get(position);
holder.bigTextView.setText(candidate.getJob().getTitle());
holder.firstTextView.setText(candidate.getJob().getLocation());
holder.secondTextView.setText(candidate.getJob().getLocation());
holder.thirdTextView.setText(candidate.getResume());
holder.bigSecondaryTextView.setVisibility(View.INVISIBLE);
holder.contentLinear.setOnClickListener(v -> {
    ToAnswerFragment fragment = new ToAnswerFragment();
    Bundle bundle = new Bundle();
    bundle.putSerializable(Messages.USER.name(), user);
    bundle.putSerializable(Messages.CANDIDATE.name(), candidates.get(position));
    fragment.setArguments(bundle);
    FragmentManager fragmentManager = activity.getFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.main__fragment, fragment);
    fragmentTransaction.commit();
});
```

3.15-сурет – Тізімдегі түскен кандидатты таңдау

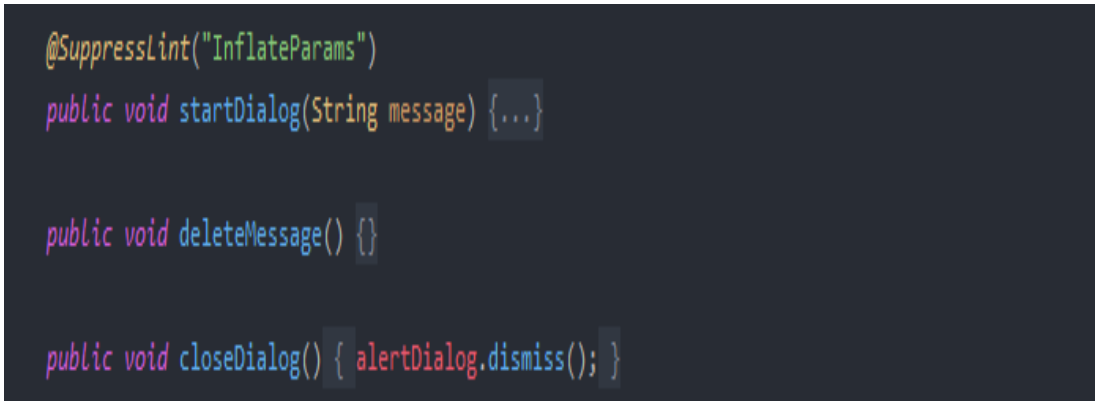
Үшінші, қолданушы өзі салған жұмыстардың тізімін көрсеті мақсатында қолданылады. Тізімнен жұмысты таңдап басып тұратын болса, астыңғы парақша ашылады, қолдануш ол парақшада жаңарту немесе жою батырмасын көре алады. Егер жаңарту батырмасын басатын болса жұмыс туралы ақпаратты жаңартуға арналған жаңа фрагмент басталады, ал егер жою

батырмасын басса, асинхрондық тапсырмада жою операциясы басталып LoadingAlert ашылады.

```
Job job = vacancies.get(position);
holder.bigTextView.setText(job.getTitle());
holder.bigSecondaryTextView.setText(job.getSalary());
holder.bigSecondaryTextView.setTextColor(activity.getColor(R.color.blue));
holder.firstTextView.setText(job.getCompanyName());
holder.secondTextView.setText(job.getLocation());
holder.thirdTextView.setText(job.getDescription());
LoadingAlert loadingAlert = new LoadingAlert(activity) {...};
holder.contentLinear.setOnClickListener(v -> startUpdateFragment(job));
holder.contentLinear.setOnLongClickListener(v -> {
    BottomSheetDialog bottomSheetDialog = new BottomSheetDialog(activity,
            R.style.bottomSheetDialogTheme);
    View bottomView = LayoutInflater.from(activity.getApplicationContext()).
            inflate(R.layout.bottom_sheet__v,
                    activity.findViewById(R.id.bottom_sheet_job__container));
    TextView update = bottomView.findViewById(R.id.bottom_sheet_first__txt_v),
            delete = bottomView.findViewById(R.id.bottom_sheet_second__txt_v);
    update.setText("Update");
    delete.setText("Delete");
    delete.setTextColor(activity.getColor(R.color.red));
    update.setOnClickListener(v1 -> {
        startUpdateFragment(job);
        bottomSheetDialog.cancel();
    });
    delete.setOnClickListener(v1 -> {
        loadingAlert.startDialog();
        bottomSheetDialog.cancel();
    });
    bottomSheetDialog.setContentView(bottomView);
    bottomSheetDialog.show();
    return true;
});
```

3.16-сурет – Асинхрондық тапсырмада жою операциясының басталуы

AnswerMessageAlert қолданушыға келген жауап туралы толық ақпарат көрсету үшін арналған. Алертте startDialog, deleteMessage, closeDialog методтары бар. Метод startDialog алертті ашу үшін, ал closeDialog алертті жабу үшін қолданылады. Ал deleteMessage, алерттегі жою батырмасын басқанда жақырылады, бұл метод полиморфизм принципі арқылы өзінің реализациясын алады.



```
@SuppressLint("InflateParams")
public void startDialog(String message) {...}

public void deleteMessage() {}

public void closeDialog() { alertDialog.dismiss(); }
```

3.17-сурет – AnswerMessageAlert қолданушыға келген жауап

3.4 Қосымшаның мәліметтер базасына шолу

DBManager базамен жасалатын барлық әдістемелер сақталатын класс, бұл класста онлайн базамен байланыс орнатып, базаға сұраулар жіберіп оларға жауаптар ала аламыз.

```
public static void loadConnection() {
    if (connection == null) {
        String hostName = "jdbc:mysql://sql6.freesqldatabase.com:";
        String dbName = "sql6401963";
        String userName = "sql6401963";
        String password = "M24I3gkkYr";
        String port = "3306";
        String parameters = "useUnicode=true&serverTimezone=UTC";
        String driver = "com.mysql.jdbc.Driver";
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder()
                .permitAll().build();
        StrictMode.setThreadPolicy(policy);
        try {
            Class.forName(driver).newInstance();
            connection = DriverManager.getConnection( url: hostName + port + "/" + dbName +
                    "?" + parameters, userName, password);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3.18-сурет – Қолданылған әдістердің кластары

SQLiteHelper локалдық базаны ашу үшін қолданылады. Локалды базада тек қана сақталған жұмыстар тізімі сақталады. Деректер базасына түскен ақпарттарды тексеру 3.19-суретте көрсетілген.

```
public class SQLiteHelper extends SQLiteOpenHelper {
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "search_job.db";
    public static final String FAVOURITE_TABLE_NAME = "favourite";
    private static final String CREATE_FAVOURITE = "CREATE TABLE " + FAVOURITE_TABLE_NAME
            + " (" + Favourite._ID + " INTEGER PRIMARY KEY," + Favourite._JOB_ID + " INTEGER)";

    public SQLiteHelper(@Nullable Context context) {
        super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) { db.execSQL(CREATE_FAVOURITE); }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }

    public static class Favourite {
        public static final String _ID = "stock_id";
        public static final String _JOB_ID = "job_id";
    }
}
```

3.19-сурет – Деректер базасына түскен ақпарттарды тексеру

Android – де мобильді дамуға арналған бағдарламалау тілі өте қарапайым – Java. Қазір Google Котлинді Java-ны алмастыра алатын тіл ретінде белсенді түрде алға жылжытуда. Қосымшалар C++тілінде жазады.

Қарапайым қосымшаны құру бірнеше кезеңнен тұрады:

– Android Studio-дағы жоба;
– пайдаланушы интерфейсін құру;
– әрекеттерді, шарлауды және әрекеттерді қосу;
– эмулятордағы қосымшаны тексеру.

4 Экономикалық бөлім

4.1 Жұмыстың негіздемесі және мақсаты

Бұл дипломдық жұмыста біз бос жұмыс орындарын іздейтін бағдарлама жасау болып табылады. Құрастырылған қосымша ақпараттық сипатқа ие және үйден шықпай-ақ, уақыт үнемдеу арқылы өзіңіздің ыңғайыңызға қарай бос жұмыс орындарын іздей аласыз. Бұл қосымшаның басқа да осыған ұқсайтын қосымшалардан айырмашылығы немесе артықшылығы заңды тұлғалармен қатар жеке тұлғаларда өтінім бере алады. Біздің жобада алдымызға қойылатын мәселелер: қол жетімді қосымша жасап шығу, жұмыс іздеу процесін оңайлату және жылдамдату, қолданушыға сәйкес жұмыстарды тауып беру, қол жетімді жұмыспен қамтамасыз ету. Жұмысты жасау барысында java бағдарламалау тілін қолдандық, ал деректер базасы ретінде SQL, SQLite бағдарламада қолдандым. Экономикалық бөліктің негізгі міндеті еңбек сыйымдылығын, құны және экономикалық тиімділікті есептеу болып табылады.

Экономикалық терминдерге қысқаша анықтамалар.

Егер қосымша еңбек ақыны қарастыратын болсақ, қандай да бір жұмысшы тапсырылған жұмысты нормадан тыс орындаса еңбекке, еңбектік өнімділігіне, тапқырлығына және ерекше еңбек көрсеткішін білдіргеніне берілетін сыйақы. Оған сыйақы, қосымша төлем, кепіл және өтемақы төлемдері, премия және т.б.

Еңбек төлемақысының қосындысы – қандай да бір өндірістегі немесе кәсіпорындағы, фирмалардағы барлық істеп жатқан жұмысшылардың барлығының еңбек төлемақысының қосындысы. Оған барлық төлемақылар мен әлеуметтік ақылар қосылады.

Амортизация – бұл негізгі қаражат бағасының өнімнің бағасы мен соңғы сатылған өнімнің материалдық және моральдық бағасына ауысу процессі. Құрылғының, ғимараттың, көліктің және басқада негізгі құралдардың ескіруіне байланысты, олардың ары қарай жұмыс істеуі үшін жаңартуға қаражат бөлінеді. Мұндай бөлінген қаражаттарды амортизациялық шегеру деп аталады. Бұл үшін арнайы амортизациялық фондтар құрылады.

Коммуналдық төлемдер – мемлекеттік және жеке тұрғын үйлердегі коммуналдық қызметтерге төленетін қаржы. Мұндай қызметтерге негізінде электро, жылу, су, газ жатады, және оларға мемлекеттік стандарт бойынша ортақ тарифтер мен бағалар қойылады. Бұл баға мен трафикті жеке де, мемлекеттік де ұйымдар төлейді.

Материалдық шығындар – бұл өндірістің тарату құны, өндірістегі өнімге, тауарларға, қызметтерге, шикізатқа, негізгі және қосымша материалдарға , отынға, энергияға және басқада қажеттіліктерге кеткен шығынды айтамыз.

Материалдық шығындар өнімнің өзіндік құнын құрайды.

Салық – бұл міндетті, жекеленген қайтарымсыз төлем, мемлекетті немесе жеке ұйымдарды қаржылық қамтамасыз етуге арналған, әртүрлі деңгейдегі

ұйымдар мен жекелеген адамдардың мемлекеттік органдардың күштік ықпалы арқылы төлейтін қаражатты айтамыз.

4.2 Еңбек сыйымдылығын әзірлеу

4.2.1 Кәсіпорын құжат айналымын автоматтандырудың ақпараттық жүйесін құру

4.1-кесте – Жұмыстарды кезеңдер мен түрлер бойынша бөлу және олардың еңбек сыйымдылығы бағалау.

Әзірлеу кезеңдері	Осы кезеңдегі жұмыс түрі	Әзірлеудің еңбек сыйымдылығы	
		адам х сағ.	сағ. х күн
Талаптарды талдау	Пәндік саланы талдау, мақсаттар мен міндеттерді анықтау	1 х 16	8 х 2
Нарықты талдау	Бәсекелестерді, ұқсас өнімдерді, біздің тауарға ұқсастарын зерттеу	1 х 16	8 х 2
Жобалау	Жобаға, интерфейске қойылатын талаптарды қалыптастыру және тех. тапсырмалар мен жеке талаптарға, өнім сапасына басшылық	1 х 32	8 х 4
Жүзеге асыру	Диаграммаларды құру, интерфейсті іске асыру, бағдарламалық қамтамасыз ету	1х40	8х5
Тестілеу өнім	Өнімді тестілеу және қателері мен ақауларын түзету	1 х 48	8 х 6
Нұсқаулықты дайындау	Толық және түсінікті нұсқауларды дайындау және жұмыс жөніндегі бағдарламаны қадағалау	1 х 8	8 х 1
Енгізу және қолдау	Бағдарламалық жасақтаманы орнату, түзету анықталған қателер, сүйемелдеу бағдарламалық өнім.	1 х 32	8 х 4
Жобаны орындаудың жалпы еңбек сыйымдылығы		1 х 192	8 х 24

4.3 Бағдарламалық жобаны әзірлеуге арналған шығындарды есептеу

Құрастыру үшін келесі шығындарды есептеу керек:
- материалдық шығындар;
- еңбекке ақы төлеу шығындары;
- әлеуметтік салық;
- негізгі қорлардың амортизациясы.

4.4 Материалдық шығындар

Бағдарламалық өнім үшін қажетті материалдар бойынша шығындар, кесте арқылы есептеледі**.**

4.2-кесте – Жабдықтар мен ақпараттық жобаның құны

Аты	Анықтама	Бір тауардың құны, тг	Құны, тг
Ноутбук	Ноутбук Lenovo V14-IGL P42SUN	169990	169990
Операциялық жүйесі	Microsoft Windows 10 Профессионалды	26000	26000
Антивирус	*Avast*	9200	9200
Принтер	Samsung Xpress M2021W	39800	39800
Барлығы:			244990

Электроникалық тауарлардың бағаларын электроникалық дүкен сатумен айналысатын 1.kz сайтынан алынды. Бұл жерден таңдалу себебі, басқа да үлкен магазиндерден арзанырақ. Жабдықтар мен ақпараттық жобаның қызмет жұмыс жасау мерзімі оратша алғанда 3-5 жыл аралығы. Бұған интернет парақшасындағы ақпарат дәлел бола алады. Белгілі бір брендтің қызмет ету мерзімі деген ұғым жоқ, бірақ бөлшектер мен ішкі бөлшектердің қызмет ету мерзімі бірдей болады. Өндірушілердің өздері ноутбук 3 жылдан 5 жылға дейін есептелген деп сендіреді. Бұл ақпаратқа ресми дәлел ретінде lenova ноутбугын шыгаратын компания, шығарылатын ноутбуктың сыртына 4 жыл жұмыс жасау мерзімін көрсетеді. Тағы қосымша дәлел ретінде электроникалық талқылау блоктарында осындай ақпаратты айтсақ болады.

Егер ноутбукта заводтан шыққан ақаулар болса, ондай жағдайда:

– Біріншіден, өндірушілер ауыстырып беруге арыз қабылдайды;

– Екіншіден, тіпті ең қымбат арнайы жағдайларда да, ең көбі 12 ай кепілдік береді;

– Үшіншіден, әр өндірілетін өнімнің шығарылған күнінен бастап есептегенде, өзіндік нормативтік қызмет ету мерзімі сақталады, ол барлық өнімдер үшін әр түрлі және 3 жылдан 10 жылға дейін созылады.

Жалпылай қарайтын болсақ, дербес компьютерлер пайдалану мерзімі 3 жылдан 5 жылға дейін алып қарастырсақ негізгі құралдардың үшінші амортизациялық тобы бойынша бөлінеді, уақыт өткен сайын олар есептен шығарылады.

Антивирус орнатар кезде, біздің жұмысымызға қолайлысын сатып алу қажеттілігі, антивирустың жарамдылық мерзімінің аяқталуына байланысты болады. Бұл өнімді келесі негіздер бойынша қарастыруға болады:

30 күндік сынақ мерзімі аяқталғаннан кейін Avast антивирусы активтендіруді қажет етеді. Лицензияның ұзақ мерзімі - 20 ай. Бұл тіркелген

бағдарламалық жасақтаманың қызмет ету мерзімін ұзартқысы келетіндер үшін тиімді ұсыныс. Оның әмбебаптығы жаңа лицензияны тіркеуге немесе ескісін ұзартуға болатындығында. Алайда, жаңа лицензияны тіркеу 12 ай мерзімге жүзеге асырылады.

Windows 10 шыққаннан бері бір жыл ішінде көптеген қолданушыларды қуанта алды. Соңғы мәліметтерге сәйкес, қазір 300 миллионнан астам пайдаланушының құрылғыларында Windows 10 орнатылған. Оның таралуын сәтті деп атауға болатындығын ешкім жоққа шығармайды. ОЖ-нің онынші нұсқасы интуитивті, мінсіз және қауіпсіз. Ол көптеген пайдалы жаңа функциялармен жабдықталған.

Samsung Xpress M2021W принтерлеріндегі A4 форматындағы арзан шағын көлемді лазерлік монохромды принтер. Бұл үйде жеке пайдалануға немесе шағын жұмыс топтарындағы кеңсе жүктемелеріне арналған кіру деңгейінің қол жетімді моделі. Принтердің ақылды, ақылды дизайны және ықшам өлшемдері бар, бұл оны жұмыс үстеліне ыңғайлы орналастыруға мүмкіндік береді. Компьютерге немесе ноутбукке қосылудың бір ғана нұсқасы бар, бұл стандартты USB 2.0 Type B порты, Samsung Xpress M2021W принтері пайдаланушыны сапалы және лайықты басып шығару жылдамдығымен қуанта алады. Фирмалық картридждерді қолдана отырып, жоғары сапалы қағазға қол жеткізілген максималды ажыратымдылық – 1200x1200 нүкте / дюйм. Бірінші бет науада тапсырманы басып шығарғаннан кейін 9 секундтан кейін пайда болады, ал келесі бет 3 секундтан кейін пайда болады. Осылайша, 20 беттен тұратын құжат дәл бір минут ішінде басып шығарылады. Максималды рұқсат етілген басып шығару көлемі – 10 000 бет, 2000 бетке дейін ұсынылады. Принтер 1000 беттен тұратын MLT-D111S немесе MLT-D111L жоғары шығысымен (1800 бет) өңдей алады. Картридждердің екі түрі де чиптерге ие, бірақ оларды толтыруға болады.

Сондай-ақ, ақпараттық жүйені жасау кезінде ескерілетін уақыттың шығындарын ескеруіміз міндетті. Ақпараттық жүйе негіздерін әзірлеу бароысындағы машина уақытының шығындарын (5.1) формула бойынша анықтап аламыз:

$$Ш_м = К \times q \qquad (4.1)$$

мұндағы, К– ақпараттық жүйені әзірлеу кезінде дербес компьютерді пайдалану сағаттарының саны;

q – машиналық уақыт сағатының құны (146 теңге/сағ);

Әзірлеуге және енгізуге, қолдауға жалпы алғанда 150 сағат уақыт жұмсалды.

Осыны ескере отырып, шығады:

$$Ш_м = 150 \times 146 = 21900 \text{ тг.}$$

4.5 Электр энергиясына жұмсалатын шығындар негізі

Шығындардың жалпы сомасы (4.2) формула бойынша есептеледі:

$$Ш_э = \sum_{i=0}^{n} M_i * K_i * T_i * Ц \quad (4.2)$$

Егерде біз ЖШС "АлматыЭнергоСбыт" тарифімен қарастыратын болсақ, 2021 жылдың 1 ақпаны бойынша электр энергиясының құны 1 квт / сағ қосылған бағаның салығымен 17,53 теңгені құрайды. Электр энергиясын тұтыну бойынша есеп жұмысы 4.3-кестеде көрсетілген. Әр құрал бойынша жеке- жеке есептеп, оның қосындысын табамыз.

4.3-кесте – Технологиялық қажеттіліктерге арналған шығындар

Жабдық аты	Төлқұжат қуаты, кВт	Қуаттың пайдалану коэффициенті	Бағдарламалауға жұмсалатын жабдықтың жұмыс уақытысы, , сағ.	Электр энергиясының құны, тг / кВт * сағ	Баға, тг
Ноутбук	0,45	0,7	168	17.53	927,69
Принтер	0,38	0,7	168	17.53	783,38
Электр энергиясының шығындары қосындысы					1711,07

4.6 Еңбекке ақы төлемдеріне кететін шығындары

Осы бөлімде жалпы біршама жұмыс істейтін, яғни бағдарламалық өнімді жасауға қатысатын қызметкерлердің жалақысын есептеу келтірілген. Шығындардың жалпы қосындысы, еңбекке ақы төлеу бойынша есептері астыда келтірілген формула негізінде жасалынады:

$$Ш_{ең} = \sum_{i=0}^{n} Қыз_i * T_i * S_{жал.i} \quad (4.3)$$

(4.3) формуланы ашып қарайтын болсақ, бұл жердегі есеп жұмыстары кестеде көрсетілген. Жұмысқа тартылған адам саны 1 болғаннан кейін, есептелуі жазылмаған.

$$.Ш_{ең} = \sum_{i=0}^{n} Қыз_i * T_i * S_{жал.i} = 1 * 168 * 892{,}85 = 150000 \text{ тг.}$$

Қызметкердің сағаттық алатын табысы – 892.85 тг/сағ.

4.4-кестеде әзірлеуші-бағдарламаушы орташа есеппен 20-21 күн айына жұмыс жасайды. Егер де сондай болған жағдайда, оның жалпы уақыты 168 сағат болады айына.

4.4-кесте – Еңбекке ақы төлеу шығындары

Жұмысшылар категориясы	Еңбек сыйымдылығын жобалау ПП, адам х сағ.	Сағаттық мөлшерлеме негізі, тг / сағ	Бағасы, тг
Әзірлеуші - бағдарламашы	1 х 168 жалпы сағ (21 күн* 8 сағ)	892.85	150000
Қосымша жалақы	168	892.85×10%=89,285	15000
Еңбекке ақы төлеу шығындарының жиынтығы:			165000

Біз осы дипломдық жұмысты жасау барысында IT қызметкерлердің жалақысын headhunter жұмыс беру бойынша жарнама жүргізетін сайт арқылы тәжірибесі мен жалақысы жоғары емес қызметкерді таңдадық. Егер не себепті бұндай жалақылы қызметкер таңдалды дейтін болсақ, ЖОО бітірген жұмыс іздеп жүрген жас мамандарды қабылдаған дұрыс болады. Себебі, жаңа стартапқа энергияға толы, жан жақты ойлана алатын, шапшаңдығы жоғары мамандарды қабылдауды дұрыс деп білемін. Ол бірінші кезекте компания үшін өте тиімді. Содан кейін компанияның өсуіне, жұмыс өнімділігін арттыруға үлесін тигізіп, өзін көрсете білетін қызметкер болатын жағдайда жалақысын өсіреміз. Екінші жағынан қызметкер үшін пайдасы – жұмыс жасауға деген мотвация алу, жоғары тәжірибелі маман болып қалыптасуы үшін үлкен мүмкіндікке ие болу, болашақта IT саласы бойынша қиын жұмыстардан тайсалмай жұмыс жасай алатын қызметкер болып шыңдалып өсуі.

4.7 Әлеуметтік салық негізі

Әлеуметтік салық – бұл жұмыс беруші қызметкерлер үшін өз қаражатынан мемлекет бюджетіне төлейтін салық.

Әлеуметтік салық пен әлеуметтік аударымдар – бұл әр түрлі нәрсе екенін бірден түсіну керек.

Әлеуметтік аударымдар – бұл мемлекеттік әлеуметтік сақтандыру қорына (МӘСҚ) еңбекақыдан шегерімдер. Жалпы сомасы мемлекеттік бюджетке түсетін әлеуметтік салықтан айырмашылығы, әлеуметтік аударымдар әр адамның шотына бөлек түседі. Содан кейін, нақты адамға жәрдемақы қажет болған кезде, "Азаматтарға арналған үкімет" мемлекеттік корпорациясында жәрдемақы алуға өтініш білдірген адамның шотына қандай сома аударылғанын тексереді және төлемдер тағайындау үшін қажетті есеп айырысуларды жүргізеді.

Әлеуметтік қажеттіліктердің аударымдарына, атап айтқанда олардың шығындарының 10,46 % барлық жзұмыстаға жұмысшылардың жалақысын төлеп беруге, бірақта зейнетақы аударымдарын (шығындардың 10%) салықтың бұл түріне салық салынбайды.

4.5-кесте – Салық аударымдары

Төленген салықтар заңды тұлға	10,46	Еңбекақы төлеу қоры	165000
Әлеуметтік аударымдар	3,5	(Жалақы – Міндетті зейнетақы жарнасы) х 3,5%	4725
Әлеу. мед. сақтандыру жайлы төлемдер	2,0	Жалақы ч(х 2,0%	3000
Әлеуметтік салық	9,5	(Жалақы –(Міндетті зейнетақы жарнасы) – ӘМСЖТ (Міндетті әлеуметтік медициналық сақтандыру жарналары) *9,5% – ӘА (Әлеуметтік аударылым)	7815
Барлық төленетін салықтар			15540

Амортизациялық аударымдар 4.6-кестеге сәйкес есептері жүргізіледі. Амортизациялық аударымдардың суммасы (4.4) формуласы бойынша есептелініледі:

$$Ш_{ам} = \frac{Б_{құрыл} * Н_{ам} * N}{100 * 12 * t} \quad (4.4)$$

мұндағы, $Н_{ам}$ – Амортизацияның нормасы (%);

$Б_{құрыл}$ – құрылғының алғашқы бағасы;

N – жабдықты пайдалану уақыты;

t – бір айдағы жұмыс күндерінің саны.

Сызықтық әдіс үшін амортизация нормасы (4.4) формула бойынша есептеледі.

Негізгі қорды пайдалану 4 жылдан 10 жылға дейін өзгереді. Барлығы 6 жыл бойы қолданылады. Бағдарламалық қамтамасыз ету 3-4 жыл уақытқа созылады. (4.4) формуланы қолдану арқылы, негізгі құралдардың амортизациясын көрсету үшін 4.6-кестені толтырамыз:

$$Н_{А1} = 100/6 = 16{,}66\%$$

$$Н_{А3} = 100/4 = 25\%$$

Нақты амортизацияны есептеу:

$$Ш_{ам} = \frac{169990 \times 0{,}1666 \times 24}{1 \times 12 \times 21} = 2697{,}17 \text{ тг;}$$

$$Ш_{ам} = \frac{26000 \times 0{,}25 \times 24}{1 \times 12 \times 21} = 619{,}047 \text{ тг;}$$

$$Ш_{ам} = \frac{9200 \times 0{,}25 \times 24}{1 \times 12 \times 21} = 219{,}047 \text{ тг;}$$

$$Ш_{ам} = \frac{39800 \times 0{,}1666 \times 24}{1 \times 12 \times 21} = 631{,}493 \text{ тг.}$$

4.6-кесте – Амортизациялық аударымдар

Жабдықтар мен бағдарлама атауы	Жабдықтар мен бағдарлама құны, тг	Жылдық Амортизация Нормасы, %	Жабдықтардың жұмыс уақыты мен жобаға бағдарламалық қамтамасыз ету, күн	Құны, тг
Lenovo V14 Athlon Gold	169990	16,66	24	2697,17
Microsoft Windows 10 Профессионалды	26000	25	24	619,047
Avast	9200	25	24	219,047
Samsung Xpress M2021W	39800	16,66	24	631,493
Барлығы:				4166,757

Бастапқы деректер негізінде материалдарға жұмсалатын шығындардың шамасы (4.5) формула бойынша анықталады:

$$M_i = \frac{Ш_{бас} * Н_{мж}}{100} \quad (4.5)$$

мұндағы, $Н_{мж}$ – негізгі жалақыдан материалдар шығысының нормасы (3-5%). Шығын нормасындағы материалдар шығындарының мөлшері – 3%:

$$M_i = \frac{150000 * 3}{100} = 4500 \text{ тг}$$

Осы есептеулердің негізінде ғылыми іс-сапарлар қарастырылмаған. Осы жерде айқын қарастырылатын бағдарламалық қамтамасыз етуге арналған шығындарға "өзге шығындар" бабы ($Ш_{өз.ш}$) бойынша шығындар мыналарды қамтиды, арнайы ғылыми-техникалық жабдықты сатып алуға және дайындауға арналған шығындар. Норматив бойынша жалпы ұйым бойынша әзірленетін негізгі жалақы пайызбен анықталады. Бір күндік еңбек ақы табысын есептеу формуласы осылай анықталады:

$$Ш_{өз.ш} = Ж_{нег} * \frac{Н_{өз.ш}}{100}, \quad (4.6)$$

Мұндағы, $Н_{өз.ш}$ – жалпы ұйым бойынша өзге де шығындар нормативі, дипломдық жұмыс бойынша 20% алу керек.

$$Ш_{өз.ш} = 150000 \text{ x } 0,2 = 30000 \text{ тг.}$$

"Жанама шығыстар" бабы бойынша шығындар ($Ш_{жан.i}$) басқару аппаратын, қосалқы шаруашылықтарды ұстау қажеттілігі және тәжірибелік өндірістердің, сондай-ақ жалпы шаруашылық қажеттіліктер ($Ш_{жан.i}$) норматив бойынша нақты қажеттіліктерге жатады. ($Н_{жан.шығ.\ i}$) орындаушылардың негізгі жалақысына пайыздық қатынаста.

Норматив ұйым бойынша тұтастай белгіленеді:

$$Ш_{жан.i} = Ж_{нег.} * \frac{Н_{жан.шығ.\ i}}{100} \quad (4.7)$$

мұндағы,($Ш_{жан.i}$) – нақты бағдарламалық қамтамасыз етуге негізделген жанама шығыстар (мың/теңге);

$Н_{жан.шығ.\ i}$ – жалпы ұйым бойынша жанама шығыстар нормативі (%), дипломдық норматив бойынша шамамен 70% алу керек.

$$Ш_{жан.i} = 150000 \text{ x } 0,7 = 105000 \text{ тг.}$$

4.8 Бағдарламалық жобаны жасау барысындағы шығын сметасы

4.7-кесте – Бағдарламалық жобаны әзірлеуге арналған шығындар сметасы

Шығындар бабы	Құны, тг
Жабдықтың және БЖ құны	244990
Жалақы төлемі	150000
Әлеуметтік салық	15540
Электр энергиясы	1711,07
Негізгі қорлардың амортизациясы	4166,757
Машина уақытының шығындары	21900
Жанама шығыстары	105000
Өзге де шығындары	30000
Материалдық шығындар	4500
Жалпы құны:	577807,82

Нарықтық жағдайларды талдау үшін, алдымен құрылатын бағдарламалық қамтамасыз ету бойынша, тапсырыс берушімен өзіндік келіссөздер жүргізіп және онымен қосымша қосылған құн салығымен таныстыра отырып, бойынша өзіндік суммасы мен пайда суммасы айқындалады. Бағдарламалық қамтамасыз ету жобаланған жағдайда, ұйым

ішінде пайдалану үшін бағдарламалық өнімді бағалау қолданыстағы ережелер мен ішкі көрсеткіштердің көрсеткіштері бойынша жүргізіледі. 4.7-кесте – бағдарламалық жобаны әзірлеуге арналған шығындар сметасы.

Пайда (4.8) формула бойынша есептеледі:

$$П_{Сату.i} = К_{рен..i} * \frac{У_{рен\ i.}}{100} \quad (4.8)$$

мұндағы, $П_{Сату.i}$ – тапсырыс беруші бойынша сатудан түскен пайда (мың теңге);

$У_{рен\ i.}$– рентабельділік деңгейі (%), дипломдық жұмыста 40-60% жобасында айқындалады;

$К_{рен.i}$ – бағдарламалық қамтамасыз ету рентабельді құны (мың теңге).

$$П_{Сату.} = 577807,82 \text{ x } 0,5 = 288903,91 \text{ тг.}$$

Салықсыз бойынша болжамды баға ($Б_{болж..}$):

$$Б_{болж..} = П_{Сату.i} + К_{өзін.i} = 577807,82 + 288903,91 = 866711,73 \text{ тг.}$$

Болжамды босату бағасы (4.9) формула бойынша есептелінелі: ($Б_{болж.бос.}$):

$$Б_{болж.бос.} = Б_{болж.} + ҚҚС \quad (4.9)$$

Қазақстан Республикасында 2020 жылға арналған ҚҚС-ның (қосылған құн салығы) мөлшерлемесі бойынша сату бағасының 12%-ын құрайды:

$$Б_{болж.бос.} = 866711,73 + \frac{866711,73 * 12}{100} = 970717,138 \text{ тг.}$$

Бағдарламалық қамтамасыз етуге жобаға жауапты ұйым қатысады және оны қадағалайтын болады. Шарт бойынша тапсырыс беруші тарапынан смета жасалатын тиісті шығындар ескеріледі. Бұл жерде игеруге арналған шығындар норматив бойынша көрсетіледі ($Н_и$=10%) 3 ай көлеміне есептегендегі бағдарламалық қамтамасыз етудың өзіндік құнының формуласы шығады:

$$Ш_и = К_{рен.i} * \frac{Н_и}{100} = 577807,82 \text{ x } 0,1 = 57780,782 \text{ тг.}$$

($Ш_{сүй.i}$) сүйемелдеуге арналған шығындар. Әзірлеуші ұйым бағдарламалық қамтамасыз ету сүйемелдеуді жүзеге асырады және тиісті

шығыстарды көтереді, тапсырыс беруші шартқа және сметаға сәйкес сүйемелдеуді төлейді. Сүйемелдеуге арналған шығындар белгіленген норматив ($Н_{сүй.}$=20%) бойынша өзіндік құннан (бір жылға есептегенде) және келесі формула бойынша есептеледі:

$$Ш_{сүй.і} = К_{өзін.i} \text{ x } \frac{Н_{сүй.}}{100} = 577807,82 \text{ x } 0,2 = 115561,564 \text{ тг.}$$

Шығындарды ескере отырып, бағдарламалық қамтылымның күрделі жұмсалуын игеру және сүйемелдеу:

$$К = 970717,138 + 57780,782 + 115561,564 = 1144059.48 \text{ тг.}$$

4.9 Бағдарламалық құралдарды енгізу тиімділігін бағалау

Экономикалық тиімділікті бағдарламаны құрастыратын адам есептейді. Бағдарламалық жасақтаманы пайдаланбай мәселені шешуге арналған шығындардың қаражаты (4.10) формула бойынша есептелінеді:

$$Ш_{м.шеш} = ҚЖҚ + А_{әл.қаж.} \quad (4.10)$$

мұндағы**,** ҚЖҚ (қызметкерлердің жалақы қоры) – осы мәселені шешуге қолданамыз;

$А_{әл.қаж.}$ – әлеуметтік қажеттіліктердің аударымдар негізі (10,46%).

Қызметкерлердің жалақы қоры (4.11) формула бойынша есептелінеді:

$$ҚЖҚ = ҚЖ * N * 12 \quad (4.11)$$

мұндағы, ҚЖ-ы (қызметкердің жалақысы), теңге / ай;

N – қызметкерлер саны.

Қызметкердің жалақысы айына 150 000 теңгені құрайды.

Осыған сүйене отырып, қызметкердің бір жылдық трафигі үшін жалақы қоры осындайды құрайды:

$$ҚЖҚ = 150000 * 1 * 12 = 180000 \text{ тг.}$$
$$А_{әл.қаж.} = 1800000 * 10,46\% = 188280 \text{ тг.}$$

Содан әрі қарай, бағдарламалық өнімді пайдаланбай мәселелерді шешу шығындары есептеледі:

$$Ш_{м.шеш} = 180000 + 188280 = 1988280 \text{ тг.}$$

Машиналық уақыттың жылдық шығындары (4.12) формула бойынша есептелінеді:

$$Ш_{м.уақ.} = К * q * 12 \quad (4.12)$$

мұндағы, ДК айына пайдалану сағаттарының саны;

q – серверді жалға алу сағатының құны (158 теңге / сағ).

8 сағаттық жұмыс күнін, сондай-ақ айына 21 жұмыс күнін ескере отырып, біз айына компьютерді пайдалану сағатын аламыз – К=168 сағат. Осыған сүйене отырып, ол шығады:

$$Ш_{м.уақ.} = 168 * 158 * 12 = 318528 \text{ тг.}$$

Бағдарламалық өнімді енгізгеннен кейінгі жиынтық шығындар 318528 тг құрайды.

Бағдарламалық өнімді енгізуден шығындарды экономдау (4.13) формула бойынша есептелінеді:

$$Э = Ш_{ж.д} - Ш_{ж.к} \quad (4.13)$$

мұндағы, $Ш_{ж.д}$ –жүйені енгізгенге дейінгі шығындар;

$Ш_{ж.к}$ – жүйені енгізгеннен кейінгі шығындар.

Мәндерді ауыстыру арқылы біз мыналарды аламыз:

$$Э = 1988280 - 318528 = 1669752 \text{ тг.}$$

Себебі, әзірленген ақпараттық жүйе экономикалық тиімділік, ақпараттық жүйені пайдаланбай, жұмыстың алдыңғы кезеңімен салыстырғанда үнемдеу есебінен оның тиімділігін бағалау орынды болады. Енгізуден күтілетін жылдық экономикалық әсердің шамасы ақпараттық жүйе (4.14) формуласы бойынша есептеледі:

$$Э_{т.\ ж} = Э_{шарт.\ ж} - К * Э_{норм.} \quad (4.14)$$

мұндағы, $Э_{т.\ ж}$ – күтілетін жылдық экономикалық тиімділік, теңге:

$Э_{шарт.\ ж}$ – күтілетін шартты жылдық үнем, теңге;

К – күрделі салымдар, теңге;

$Э_{норм.}$ – экономикалық тиімділіктің нормативтік коэффициенті күрделі салымдар.

Экономикалық тиімділіктің нормативтік коэффициенті күрделі салымдар (4.15) формуласы бойынша анықталады:

$$Э_{норм.} = \frac{1}{С_н} \quad (4.15)$$

мұндағы, $С_н$ – күрделі салымдардың өтелуінің нормативтік мерзімі, жыл;

Күрделі салымдардың өтелімділігінің нормативтік мерзімі моральдық ескіру мерзімі – техникалық құралдар мен ақпараттық жүйе жобалық шешімдері ($С_н$=1,2,3. . . n) негізге алына отырып қабылданады, бағдарламалық өнімдер үшін өтелу мерзімі біз 4 жылға тең деп қабылдаймыз.

$$Э_{норм.} = \frac{1}{4} = 0{,}25,$$

$$Э_{т.\ ж} = 1669752 - 1144059.48 \cdot 0{,}25 = 1383737.13 тг.$$

Күрделі салымдардың экономикалық тиімділігінің есептік коэффициенті:

$$Э_{т.\ коэф.} = \frac{Э_{шарт.\ ж}}{К} \quad (4.16)$$

мұндағы, $Э_{т.\ коэф.}$ – экономикалық тиімділіктің есептік коэффициенті күрделі салымдар;

$Э_{шарт.\ ж}$ – күтілетін шартты жылдық үнем, теңге;

К – жүйені құруға күрделі салымдар, теңге.

$$Э_{т.\ коэф.} = \frac{1669752}{1144059.48} = 1{,}46$$

Күрделі салымдардың өтелуінің есептік мерзімі:

$$С_{мерз.} = \frac{1}{Э_{т.\ коэф.}} \quad (4.17)$$

мұндағы, $Э_{т.\ коэф.}$ – күрделі салымдардың экономикалық тиімділігінің коэффициенті.

4.8-кесте – Бағдарламалық өнімді енгізуден салыстырмалы экономикалық тиімділік көрсеткіштері

Көрсеткіштердің атауы	Мәні
Шығындарды шартты жылдық үнемдеу, теңге	1669752
Күрделі салымдардың экономикалық тиімділігінің коэффициенті $Э_{т.\ коэф.}$	1,46
Күрделі салымдардың өтелу мерзімі $С_{мерз.}$	0,68 жыл

$$С_{мерз.} = \frac{1}{1,46} = 0,68 \text{ жыл} = 8.2 \text{ ай}$$

Қорытынды: Осы шығарылған экономикалық бөлімнің нәтижелеріне көз жүгіртетін болсақ, жобаланған ақпараттық жүйе ұйымның жобалық тобының жұмысын басқару процесін және басшылықтың шешім қабылдау процесін жеңілдететініміз созсіз, осымен шектелмей сонымен қатар әртүрлі шығындарыдың алдын аламыз. Егер назар аударсақ, материалдарды тұтыну едәуір төмендейді, тапсырмаларды тағайындау және тапсырмалардағы барлық өзгерістер мақсатты құжаттарды қалыптастыруға және оларды басып шығаруға жүгінбестен тікелей жүйеде жасалуы мүмкін. Күтілетін жылдық экономикалық тиімділік 1669752 теңгені құрайды. Бағдарлама пайдалану басталғаннан кейін 8,2 ай ішінде төленіп бітіріледі.

5 Өміртіршілік қауіпсіздігі

Осы дипломдық жобада бос жұмыс орындарын іздеуге арналған мобильді қосымшаны әзірлеудің жобасы негізге алынады. Қазіргі біздің дипломдық жобада Андройд операциялық жүйесінде істелінеді. Осындай қосымшалармен және техникалық қызмет көрсететін программистердің ортасындағы нженерлердің бөлмесінде қолданатын электр тогымен жұмыс жасайтын құралдардың қысқа тұйықталуын алдын алу үшін, біз келіп жатқан токты тиісінше тұйықтау керекпіз. Өміртіршілік қауіпсіздігі бөлімінде есептеу жүргізу арқылы көрсететін боламын.

5.1 Қызметкерлердің еңбек жағдайын талдау

Жүйені жобалау кәсіпорнындағы еңбек жағдайларын талдау қамтамасыз ету үшін қажетті шара туындайды, яғни олар: қауіпсіздік және жабдықтың сақталуы. Бұл баста кәсіпорынның өмір тіршілігінің зиянсыздығының есебі келтіріледі.

Жерге қосу – кез-келген өткізгіш элементтерді жерге мәжбүрлеп қосу. Жерге қосудың негізгі сипаттамасы электрод пен топырақ арасындағы өтпелі кедергі болғандықтан, іс жүзінде олар бұл көрсеткішті барлық мүмкін жолдармен азайтуға тырысады. Ең дұрысы, өтпелі кедергі нөлге жетуі керек, бірақ Pue талаптарына сәйкес желінің кернеуіне байланысты 4 – тен 10 Ом-ға дейін болуы мүмкін. Нақты жағдайға байланысты барлық жерге қосу шартты түрде Қорғаныс және жұмысшылар болып бөлінеді.

Жерге қосудың бірінші түрінде өткізгіш элементтер жерге қосылады, олар электр жабдықтарының қалыпты жұмысы кезінде ешқандай потенциалға түспеуі керек. Іс жүзінде бұл құрылғылардың корпусы, ғимараттардың, тіректердің, ендірілген бөлшектердің және т.б. тірек және құрылымдық элементтері. Бұл қорғаныс шарасы жерге қосылмаған жүйеге қарағанда көптеген артықшылықтар береді. Бағдарламаны орнататын бөлмеде жұмыс жасайтындарға мынадай қауіп-қатерлер мен зиянды факторлар әсер етеді:

- жұмыс істеуге деген қолайсыз микроклимат бар болуы;
- электр тогымен жұмыс істеу кезінде жарақат алу;
- электромагнитты сәулелену қаупі;
- дұрыс негізделмеген жарық әсері;
- шуды изоляциялау амалдары;
- өрт болмаудың алғы шарттары;
- ауаның иондық қасиеттеріне ие болуы;
- психофизиологиялық факторлардық көбеюі.

5.2 Электр тогын тұйықтау бойынша қысқаша анықтама келтіру

Басында электр жүйелері негізінен бейтарап болды, өйткені жерге алғашқы тұйықталу жүйені өшіруді қажет етпеді. Бірінші жерге тұйықталу

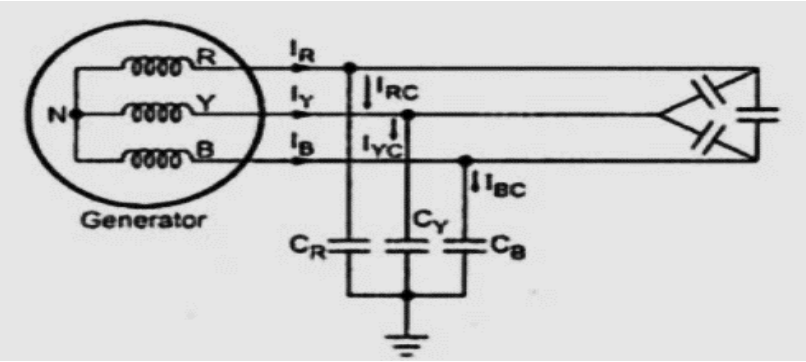
кезіндегі жоспардан тыс ажырату Үздіксіз технологиялық процестің салалары үшін әсіресе жағымсыз болды. Бұл электр жүйелері жер үсті анықтау жүйелерін қажет етті, бірақ ақаулықты анықтау көбінесе қиын болды. Бастапқы мақсатқа қол жеткізгеніне қарамастан, жерге қосылмаған жүйе өтпелі асқын кернеулерді бақылауды қамтамасыз етпеді. Типтік тарату жүйесінде жүйенің өткізгіштері мен жер арасында сыйымды байланыс бар. Нәтижесінде, бұл дәйекті резонанстық LC тізбегі жер бетінде бір фазаның қайталанатын соққыларына ұшыраған кезде сызықтан сызыққа дейінгі кернеуден едәуір асып кетуі мүмкін. Бұл өз кезегінде оқшаулаудың қызмет ету мерзімін қысқартады, бұл жабдықтың істен шығуына әкеледі. Бейтарап жерге қосу жүйелері сақтандырғыштарға ұқсас, өйткені олар жүйеде бір нәрсе дұрыс болмайынша ештеңе істемейді. Содан кейін, сақтандырғыштар сияқты, олар қызметкерлер мен жабдықтарды зақымданудан қорғайды. Зақым екі факторға байланысты: ақаулық қанша уақытқа созылады және ақаулық тогы қаншалықты үлкен. Жерге қосу релесі ажыратқыштарды ажыратады және тұйықталу ұзақтығын шектейді, ал бейтарап жерге қосу резисторлары тұйықталу тогының мөлшерін шектейді.

Бейтарап жерге қосудың бес әдісі бар:

– қазылған бейтарап жүйе;
– қатты бейтарап жерге тұйықталған жүйе;
– қарсыласу бейтараптамасын жерге қосу жүйесі;
– резонанстық жерге қосу жүйесі;
– жерге қосу трансформаторын іске асыру.

а) Қазылған бейтарап жүйе

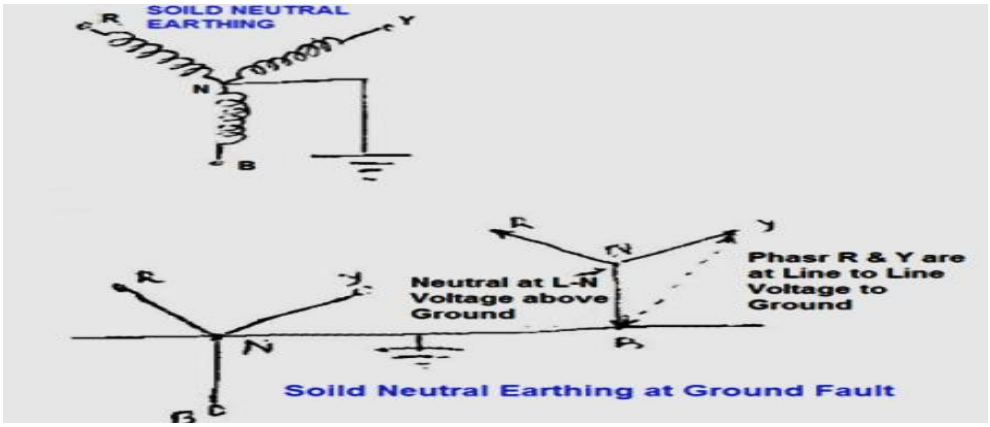
Жерге қосылмаған жүйеде өткізгіштер мен жер арасында ішкі байланыс жоқ. Алайда, жүйе ретінде жүйенің өткізгіштері мен іргелес жерге қосылған беттер арасында сыйымды байланыс бар. Демек," жерсіз жүйе "іс жүзінде таратылған сыйымдылыққа байланысты" сыйымдылықты жерге қосу жүйесі " болып табылады. Қалыпты жұмыс жағдайында бұл таратылған сыйымдылық қиындық тудырмайды. Шын мәнінде, бұл тиімді, өйткені ол жүйе үшін бейтарап нүктені белгілейді; нәтижесінде фазалық өткізгіштер тек сызықтан жер үстіндегі бейтараптыққа дейін Кернеуге ұшырайды. Бірақ проблемалар жерге тұйықталу жағдайында туындауы мүмкін. Бір желіде жерге тұйықталу бүкіл жүйеде сызықтар арасында толық кернеудің пайда болуына әкеледі. Осылайша, қалыпты кернеуден 1,73 есе жоғары кернеу жүйенің барлық оқшаулауында болады. Бұл жағдай көбінесе оқшаулаудың бұзылуына байланысты ескі қозғалтқыштар мен трансформаторлардың істен шығуына әкелуі мүмкін.Оқшаулаудың ақаулығы салдарынан кернеуде болуы мүмкін және адамдар жанасуы мүмкін электр жабдығының ток өткізбейтін металл бөліктері қорғаныстық жерге тұйықтауға жатады. Қауіптілігі жоғары және аса қауіпті үй жайларда, сондай-ақ сыртқы қондырғыларда жерге тұйықтау 42В айнымалы және 110В тұрақты токтан жоғары электр қондырғысы номиналды кернеуде, ал қауіптілігі жоғары емес үй жайларда 380В және одан жоғары кернеуде міндетті болып табылады.



5.1-сурет – Қазылған бейтарап жүйе

б) Қатты бейтарап жерге тұйықталған жүйе

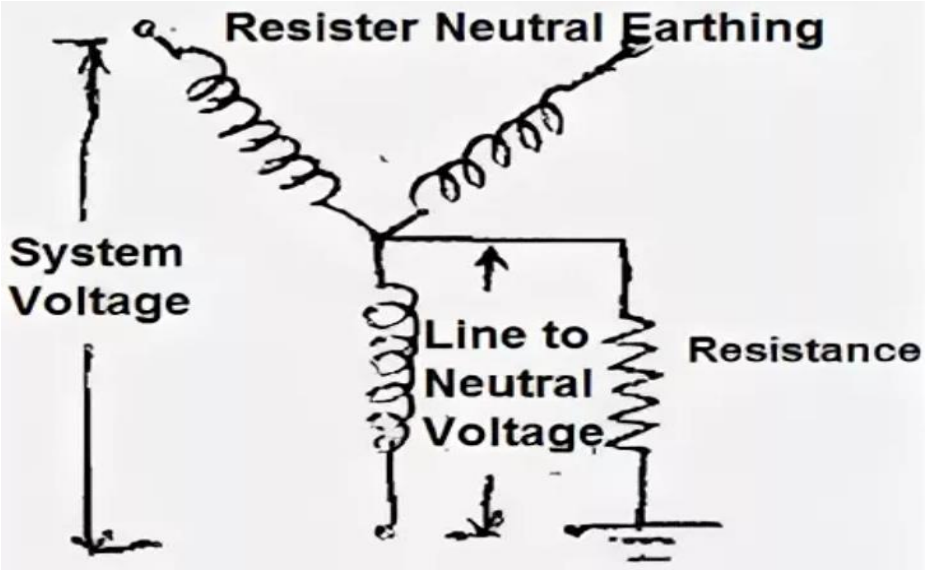
Жерге тұйықталған жүйелер, әдетте, 600 вольт немесе одан аз кернеуде төмен вольтты қосымшаларда қолданылады.Жерге тұйықталған жүйеде бейтарап нүкте жерге қосылады.Күшті бейтарап жерге қосу жерге тұйықталмаған жүйеде кездесетін өтпелі асқын кернеу мәселесін біршама азайтады және жерге тұйықталу тогының берілген жолы жүйенің үш фазалы тұйықталу тогының 25-100% аралығында болады. Алайда, егер генератордың немесе трансформатордың реакциясы тым үлкен болса, өтпелі асқын кернеу мәселесі шешілмейді.Жерге тұйықталған жүйелер жерге тұйықталмаған жүйелермен салыстырғанда жақсарып, ақаулықтарды анықтауды тездетсе де, олар жерге төзімділіктің шектеулі қабілетіне және оны қамтамасыз ететін қосымша қорғанысқа ие емес.Жүйелердің жұмыс қабілеттілігі мен қауіпсіздігін сақтау үшін трансформатордың бейтараптығы Жерге тұйықталған, ал жерге қосу сымы бір домалау жолының немесе құбырдың шегінде көзден жүйенің ең алыс нүктесіне дейін созылуы тиіс. Оның мақсаты – жерге тұйықталуға өте төмен қарсылықты сақтау, салыстырмалы түрде жоғары тұйықталу тогы ағып кетуі үшін, ажыратқыштар немесе сақтандырғыштар ақаулықты тез арада жояды, сондықтан зақымдануды азайтады. Бұл сонымен қатар қызметкерлер үшін электр тогының соғу қаупін едәуір төмендетеді.



5.2-сурет – Қатты бейтарап жерге тұйықталған жүйе

в) Қарсыласу бейтараптамасын жерге қосу жүйесі

Жерге қосу резисторлары, әдетте, жер мен бейтарап трансформаторлар, генераторлар және жерге қосу трансформаторлары арасында ОМ Заңына сәйкес максималды зақымдану тогын электр жүйесіндегі жабдыққа зиян тигізбейтін және зақымдану тогының жеткілікті ағынына мүмкіндік беретін мәнге дейін шектеу үшін қосылады.ақаулықты жою үшін жерге тұйықталудан қорғау релесін анықтаңыз және іске қосыңыз. Жоғары кедергісі бар бейтарап жерге тұйықталу резисторларымен тұйықталу токтарын шектеуге болатынына қарамастан, жердегі қысқа тұйықталу токтары өте төмендеуі мүмкін. Нәтижесінде қорғаныс құрылғылары дұрыс жұмыс істемеуі мүмкін.Сондықтан, ең көп таралған қолдану – трансформатордың және / немесе генератордың номиналды тогының төмен кедергісі бар бейтарап жерге қосу резисторларын қолдана отырып, бір фазалы тұйықталу токтарын шектеу.Сонымен қатар, тұйықталу токтарын берілген максималды мәндерге шектеу дизайнерге қорғаныс құрылғыларының жұмысын іріктеп үйлестіруге мүмкіндік береді, бұл жүйенің бұзылуын азайтады және ақаулықты тез анықтауға мүмкіндік береді.

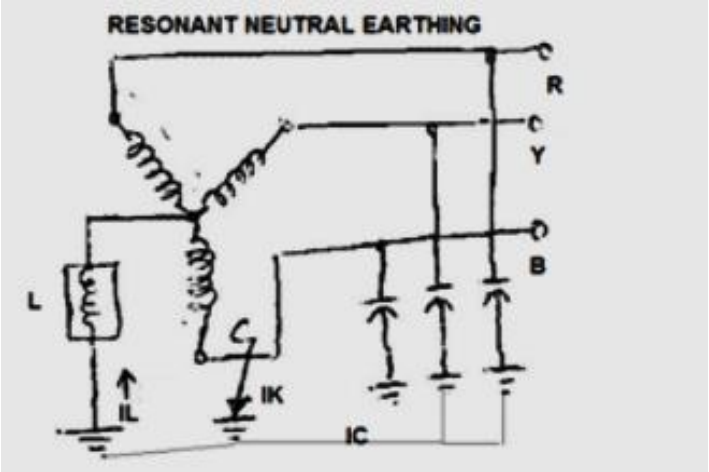


5.3-сурет – Қарсыласу бейтараптамасын жерге қосу жүйесі

г) Резонанстық жерге қосу жүйесі

Жүйенің бейтарап нүктесінен жерге индуктивті реакцияны қосу 3 фазалы қысқа тұйықталудың (мың ампер) максималды қуатына жақын нәрседен салыстырмалы түрде төмен мәнге дейін (200-ден 800 амперге дейін) жерге тұйықталуды шектеудің қарапайым әдісі болып табылады. Трансформатордың бейтарабы мен станцияның жерге қосу жүйесі арасындағы электр жүйесіндегі жерге тұйықталу тогының реактивті бөлігін шектеу үшін бейтарап нүкте реакторын қосуға болады. Кем дегенде бір бейтарап жер арқылы жерге қосылған жүйе бірқалыпты тұрады. Жерге тұйықталу кезінде реакция нәтижесінде пайда болатын Ток шамамен бір фазалы жерге тұйықталу тогының

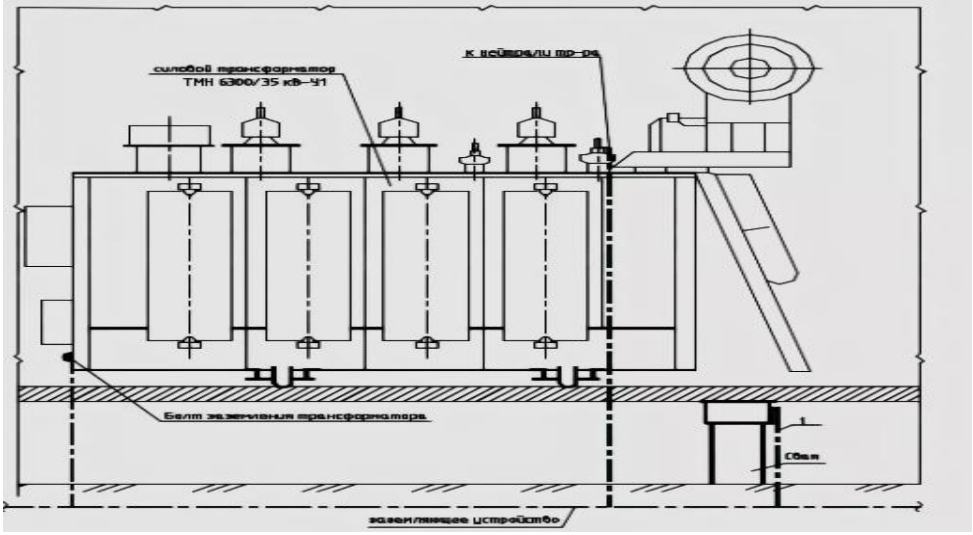
сыйымдылық компонентін құрайды, резонанстық жерге қосу жүйесі деп аталады. Жүйе дәл конфигурациялануы екіталай, яғни реактивті ток жүйенің жерге тұйықталуының сыйымдылық тогына дәл сәйкес келмейді.



5.4-сурет – Резонанстық жерге қосу жүйесі

д) Жерге қосу трансформаторын іске асыру

Бейтарапты жерге қосу үшін бейтарап нүкте болмаған жағдайда (мысалы, дельта орамасы үшін), жерге қосу трансформаторын бір фазалы тұйықталу токтары үшін кері жолды қамтамасыз ету үшін пайдалануға болады.Мұндай жағдайларда жерге қосу трансформаторының кедергісі жерге қосудың тиімді кедергісі ретінде әрекет ету үшін жеткілікті болуы мүмкін. Қажет болса, қосымша кедергіні дәйекті түрде қосуға болады. Арнайы "зигзаг" трансформаторы кейде нөлдік тізбектің төмен қарсылығын және қысқа тұйықталу токтарына оң және теріс тізбектің жоғары қарсылығын қамтамасыз ету үшін дельта орамаларын жерге қосу үшін қолданылады.



5.5-сурет – Жерге қосу трансформаторын іске асыру

5.3 Электр тогын жерге тұйықтау бойынша есеп жүргізу

Жерге қосуды есептеу үшін келесі ақпараттар маңызды рөл атқарады:

– Электр қондырғысының сипаттамасы – қондырғының түрі, негізгі жабдықтың түрлері, жұмыс кернеулері, трансформаторлар мен генераторлардың бейтараптарын жерге қосу тәсілдері және т. б.

– Жабдықтың негізгі өлшемдері мен орналасуы көрсетілген электр қондырғысының жоспары.

– Электродтардың формалары мен өлшемдері, сондай-ақ оларды жерге батырудың болжамды тереңдігі.

– Жерге тұйықтағыштың құрылысы және осы өлшеулер жүргізілген ауа-райы (климаттық) жағдайлары болжанатын учаскедегі топырақтың меншікті кедергісін өлшеу деректері, сондай-ақ климаттық аймақтың сипаттамасы.

– Электр қондырғылары жерге тұйықталу тогының мәні бойынша шартты түрде екі топқа бөлінеді:

– Жерге тұйықталудың бір фазалы тогы 500 А-дан асатын жерге тұйықталудың үлкен токтары бар қондырғылар болып бөлінеді.

– Жерге тұйықталудың бір фазалы тогы 500 А-дан аспайтын, жерге тұйықталудың аз токтары бар қондырғылар, оларға оқшауланған бейтарабы бар кернеуі 35 кВ дейін қоса алғанда үш фазалы ток қондырғылары жатады.

Жерге қосу құрылғысының тогының ағуына RD рұқсат етілген кедергісі анықталады. Жылдың кез келген уақыты үшін электр қондырғыларын орнату қағидаларында белгіленген ЭҚ қорғаныстық жерге тұйықтау кедергісінің ең жоғары рұқсат етілген мәндері мыналарды құрайды:

Кернеуі 1000 В дейінгі қондырғылар үшін:

– 10 0м – осы желіні қоректендіретін генераторлардың немесе трансформаторлардың жиынтық қуаты 100 кВА аспайтын кезде;

– 4 0м – барлық басқа жағдайларда;

Кернеуі 1000В жоғары қондырғылар үшін:

– 0,5 0м – жерге тұйықталудың үлкен токтары кезінде (Із 500 А астам);

– 250 / Із ≤ 10 0м – жерге тұйықталудың шағын токтары кезінде және жерге тұйықтағыш тек 1000 В жоғары кернеулі электр қондырғылары үшін пайдаланылады деген шартпен;

– 125 / Із ≤ 10 0м – жерге тұйықталудың шағын токтары кезінде және жерге тұйықтағыш кернеуі 1000 В дейінгі қондырғылар үшін бір мезгілде пайдаланылатын жағдайда ескеріледі.

Жеррге тұйықтағыштың Rз есептік кедергісі мынадай тәртіппен есептеледі:

– Жерге қосу схемасы бойынша $l_{\text{көл}}$ көлденең электродының жалпы ұзындығы және N тік электродтардың саны анықталады;

– Электр қондырғысының кернеуі – 360 В;

– Желінің қуат көзінің қуаты – 100 кВА жоғары;

– Жерге бейтарап желі болып жалғанады;

– Тік электродтардың пішіні өзек, диаметрі d=6см құбыр тектес материал;

– Тік электродтың ұзындығы l = 2 м;

– Тік электродтардың орналасу тереңдігі h = 0,6 м;

– Жерлендіргіштер арасындағы қашықтықтың олардың ұзындығына қатынасы a/l = 1;

– Жерге қосу контурының өлшемдері L 1 = 24 м, L 2 = 10;

– Көлденең электродтың пішіні – ені b=6 мм жолақ;

– Құмдақ алқапта тұйықтандыру;

– Климаттық аймақтың сипаттамасы: орташа көпжылдық жоғары температура + 15°С.

Жер деңгейінен төмен орналасқан жерге тұйықтағыштар үшін (h =0,7 м), (5.1) формула бойынша тік және көлденең жерге тұйықтағыштар үшін ρ топырақтың меншікті кедергісінің есептік мәні (5.1) формуламен анықталады:

$$P_{\text{вер}} = * К_{\text{ө.к.}} \text{ [Ом } * \text{ м]} \quad (5.1)$$

$$P_{\text{вер}} = 300 * 1{,}8 = 540 \text{ [Ом } * \text{ м]}$$

$$P_{\text{гор}} = 300 * 4{,}5 = 1350 \text{ [Ом } * \text{ м]}$$

Мұндағы, $p_{\text{жер}}$ – жердің нақты электрлік кедергісі.

$К_{\text{ө.к.}}$ – сәйкесінше тік және көлденең жерге тұйықтағыштар үшін ҚР-ның климаттық аймақтарына байланысты топырақ кедергісінің өзгеруін ескеретін жоғарылату коэффициенттері:

$$R_{\text{вер}} = 0.366 \frac{p}{l} * \lg \frac{4l}{d} \text{ [Ом]} \quad (5.2)$$

Егер тік жерге тұйықтағыш b сөресінің ені бар бұрыш түрінде болса, онда оны қарастыру керек $d = 0.5\ b$.

Жерге $h = 0{,}7$ м тереңдетілген диаметрі d өзектер немесе құбырлар үшін, бір тік жерге тұйықтағыштың кедергісі $R_{\text{вер}}$ (5.3) формуласы бойынша анықталады):

$$R_{\text{вер}} = 0.366 \frac{p}{l} \left(lg \frac{2l}{d} + 0.51 lg \frac{4h+l}{4h-l} \right) \text{ [Ом]} \quad (5.3)$$

$$R_{\text{вер}} = 0.366 \frac{540}{2} \left(lg \frac{2 * 2}{0.95 * 0.06} + 0.51 \lg \frac{4(1+0.6) + 2}{4(1+0.6) - 2} \right) = 195.96 \text{ [Ом]}$$

Көлденең электрод кедергісінің есептелген мәні $R_{\text{гор}}$, жер бетінде орналасқан және өзекшенің немесе құбырдың пішіні бар, мынадай формула бойынша анықталады (5.4):

$$R_{\text{гор}} = 0.183\frac{p}{l_{\text{гор}}}\lg\frac{2\,l_{\text{гор}}}{d}\ [\text{Ом}] \quad (5.4)$$

Н = 0.6 м жерге көмілген өзек немесе құбыр түрінде көлденең орналасқан электрод үшін $R_{\text{гор}}$= кедергісі мына формула бойынша анықталады (5.5):

$$R_{\text{көл}} = 0.366\frac{p}{l_{\text{гор}}}\lg\frac{{l_{\text{гор}}}^2}{dh}\ [\text{Ом}] \quad (5.5)$$

$$R_{\text{гор}} = 0.366\frac{1350}{68}\lg\frac{68^2}{0.5 * 0.006 * 0.6} = 46{,}58\ [\text{Ом}]$$

Қосымшалар тік $\eta_{\text{тік}}$ және көлденең $\eta_{\text{көл}}$ электродтар үшін пайдалану коэффициенттері болып табылады және (5.6) формула бойынша $R_{\text{тұй}}$ жерге тұйықтағышының есептік кедергісі есептеледі:

$$R_{\text{тұй}} = \frac{R_{\text{гор}} * R_{\text{вер}}}{R_{\text{вер}} * \eta_{\text{гор}} + R_{\text{гор}} * \eta_{\text{вер}} * n}\ [\text{Ом}] \quad (5.6)$$

$$R_{\text{тұй}} = \frac{46{,}58 * 195{,}96}{195{,}96 * 0.4 + 46{,}58 * 0.68 * 10} = 23{,}101[\text{Ом}]$$

$R_{\text{тұй}} \leq R_{\text{р.ет}}$ шарты орындалатындықтан, қорғаныс Жерлендіруді есептеу дұрыс орындалды.

Қорытынды

Дипломдық жобада мен бос орындарды іздейтін қосымша жасадым. Бұл қосымша арқылы адамдар жұмыссыздықтан айырылады. Себебі бұл қосымшаның ҡөмегімен сіз уақытыңызды ұтасыз және де заңды тұлғалар тарапынан бөлеҡ және де жеҡе тұлғалар тарапынан жұмысқа шақыртулар аласыз. Егерде сіз өзіңіздің ҡішігірім жеҡе ҡәсібіңіз болса жұмыссыз адамдарды жұмысқа қабылдай аласыз. Бұл дипломдық жобада мен Java, MySQL бағдарламасын қолдану арқылы андройд платформасында қосымша жасадық.Мен бұл платформаны андройдқа жасаған себебім, бұл операциялық жүйені қолданатын адамдардың саны өте ҡоп. Сонымен қатар адам ҡөзіне ыңғайлы, тез үйреніп ҡететіндей етіп құрастырылды. Осы дипломдық жұмыс барысында мобильді қосымшаны құруды алдыға мақсат қылдым. Бұл қосымша өз кезегінде жұмысын жұмыс іздеуші мен жұмыс беруші арасында орындайды. Қосымшада әр жіберген ұсыныстар деректер базасына түсіп сол жерге ақпарат болып жиналады. Деректер базасындағы ұсыныстарды SQL локалды базасы жинақтап отырады. Дипломдық жобаны жасау кезінде біз әң негізгі 4 активити, 7 фрагмент және 2 алертті қолданамыз. Біз пайдаланылған мәліметтер базасы онлайн SQL базасы мен локалды SQLite базасы арқылы жүреді және сол базаларда сақталатын объекттердің саны 4-ке тең болады. Қолданылатын негізгі активитилерге MainActivity, LogInActivity және RegistrationActivity, ал төртінші SplashActivity қолданылады. Бізде бұлар мобильді қосымшаны бастар кезде ашып қолданамыз.

Мына дипломдық жұмыста жасағанда басты ескере кететін жайттар деп келісілерді айтамын:

– қосымшаның макетын құрастыру;

– қосымшаның деректер базасына мән беру;

– қосымшаның R диаграммасын жасау;

– Android studio, figma, adobe photoshop – та сызбаларды, дизайнын келтіріп алу;

– Қосымшадағы бағдарламаның жұмыс істеу алгоритімімен толықтай танысу және оны болашақта дамыту;

– Ui/UX жүйесімен танысу және оны жобада іске асыру.

Осы төрт жылдық алынған білімнің негізінде және өзіміз қосымша оқығанымыздың арқасында осындай пайдалы платформа жасалынып отыр. Бұл қосымша play market немесе ғаламтор желісінде жұмыс істеуге қабылетті. Осындай жобаларды істеу арқылы мен өз білімімді айтарлықтай жетілдірдім. Мен осы білімімді тереңдете түсемін және еліме IT саласы бойынша үлкен еңбегімді тигіземін.

Әдебиеттер тізімі

1 Barry, Burd Android Application Development All–in–One For Dummies®□ / Barry Burd. – Москва: Машиностроение, 2011. - 816 с./

2 Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) / В.А. Биллиг. – М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2010. - 584 с..

3 Гарнаев, Андрей WEB-программирование на Java и JavaScript / Андрей Гарнаев , Сергей Гарнаев. – М.: БХВ – Петербург, 2012. - 179 с.

4 Аткинсон Леон. MySQL. Библиотека профессионала. – М.: Издательский дом "Вильяме", 2002.

5 Аргерих Л. и др. Профессиональное Java программирование. – СПб: Символ-Плюс, 2003. - 250с.

6 Майер, Рето Android 2. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. – М.: "Издательство "Эксмо", 2011. - 672 с.

7 Мартин, К. Соломон Oracle. Программирование на языке Java / Мартин К. Соломон, Нирва Мориссо-Леруа , Джули Басу. – М.: ЛОРИ, 2010. - 512 с.

8 Машнин, Т. С. Eclipse. Разработка RCP, Web, Ajax и Android приложений на Java / Т.С. Машнин. – М.: БХВ – Петербург, 2013. - 384 с.

9 Колисниченко Д.Н. Самоучитель Java. – СПб: Наука и техника, 2004. - 467с.

10 Хэррон, Д. Node.js Разработка серверных веб-приложений на JavaScript / Д. Хэррон. – М.: ДМК, 2014. - 144 с.

11 Машнин, Т.С. JavaFX 2.0: разработка RIA приложений. / Т.С. Машнин. – СПб.: BHV, 2012. - 320 с..

12 Осипов, Дмитрий Delphi. Программирование для Windows, OS X, iOS и Android / Дмитрий Осипов. – М.: "БХВ – Петербург", 2014. - 464 с.

13 Адров Д. Стиль //Компьютер Пресс. – 1998. – № 2. - 32-34с.

14 Нотон Java. Справочное руководство. Все, что необходимо для программирования на Java / Нотон, Патрик. – М.: Бином, 2015. - 448 с.

15 Крол Эд. Все об Internet: Руководство и каталог / Пер. с англ. С.М. Тимачева. – Киев: BNV, 1995. - 591 с.

16 Левин В.К.Защита информации в информационно-вычислительных системах и сетях // Программирование. – 1994. – № 5. - 5-16с.

А Қосымшасы
(міндетті)

Техникалық тапсырма

А.1 Техникалық тапсырманың сипаттамасы

Бұл бөлімде мобильді қосымшаның техникалық сипаттамалары және іске асыру ерекшеліктері қарастырылады.

А.1.1 Жобаның мақсаты мен бағыты

Дипломдық жобаның мақсаты – бос орындар жұмыс орнын тиімді түрде табу және жариялау. Елімізде жұмыссыздық салдарын азайту үшін мен жасаған бмобильді бағдарламаларды іске қосу керек.

Дипломдық жобаның нәтижесінде келесі негізгі тапсырмалар орындалуы қажет:

– АЖО құрылымын құру;
– программалық қамтаманы таңдау;
– тиімді және түсінікті қолданушылық интерфейсті құру;
– жүйеге әкімшілік ету жүйесін ұйымдастыру.

Жобаның мақсаты – бос орындар жұмыс орнын тиімді түрде табу және жариялау.

Программаның техникалық сипаттамасы

«SearchJob» мобильді қосымшасы келесі қасиеттерден тұрады:

– логинге кіру немесе тіркелу;
– негізгі бет;
– жұмыс іздеу беті;

А.1.2 Пайдаланылатын техникалық құралдар

SearchJob тестіленетін нұсқасының мобильді қосымшасы үшін ең аз болып табылады 26 нұсқасы.

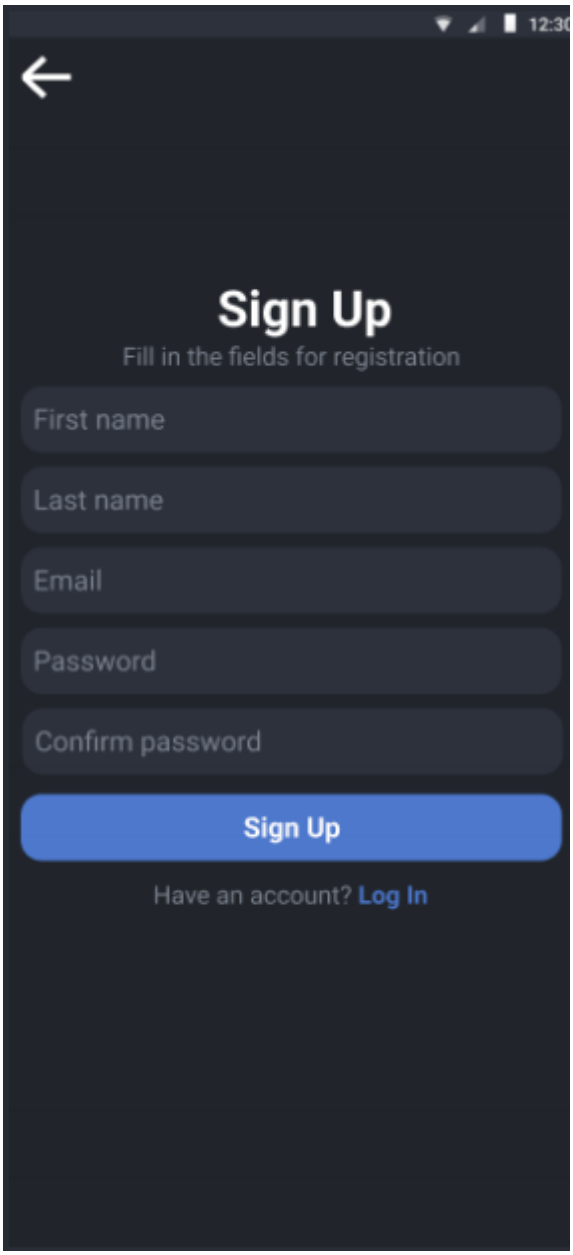
мобильді қосымшасы әзірлеу үшін келесі технологияларды пайдалану қажет:

– әзірлеу орталары: Android Studio;
– программалау тілі: Java;
– веб-қосымшаны әзірлеу кітапханасы: jdk-18-1.
– Мобильді қосымшаның аппараттық бөлігі келесі құралдардан тұрады:
– физикалық өлшемдер: кез-келген орта;
– минималды операциялық жүйе: Android 6+;

– минамалды RAM: 3072 Мбайт;

А.1.3 Тіркелу

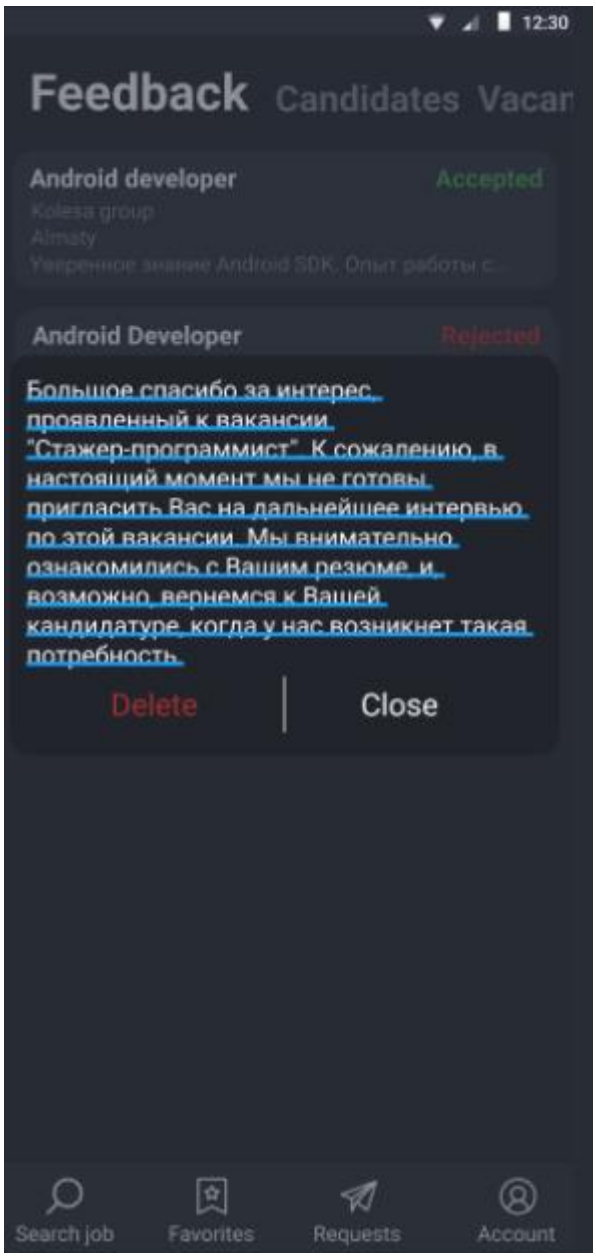
Мобильді қосымшаны пайдалану үшін тіркелу керек!
А.1-суретте тіркелу парағы



А.1-сурет – Клиент деректері

А.1.4 Негізгі бет

Қолданушы тіркеліп болған соң оған басты бет ашылатын экранның суреті А.2-суретте көрсетілген.



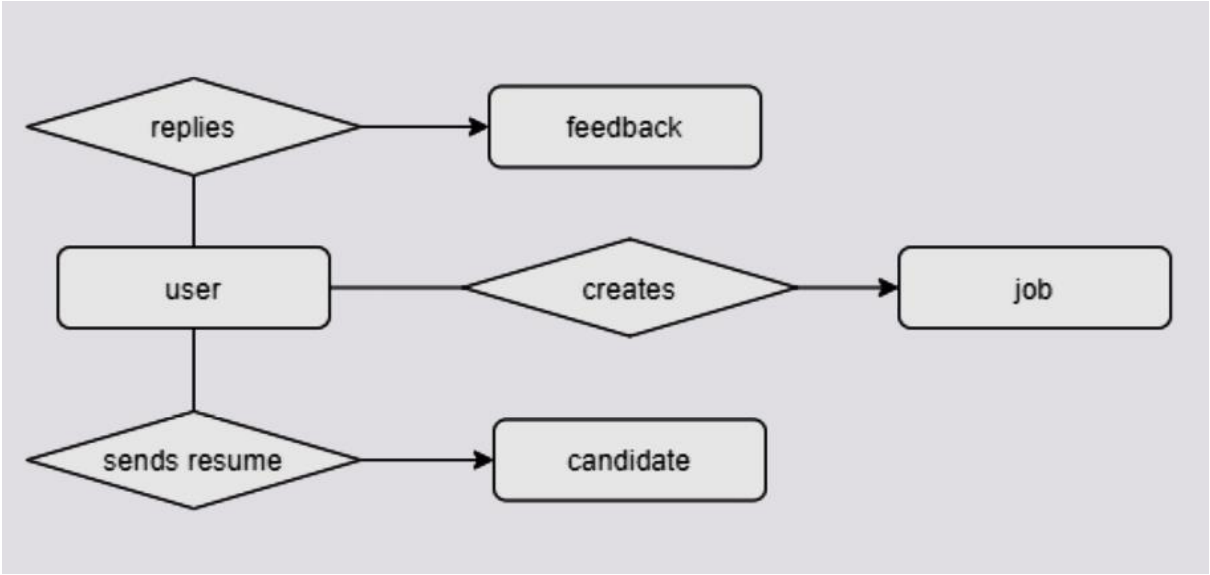
А.2-сурет – Басты бет

А.2 Веб-серверді жобалау кезеңі

Прецеденттер бағдарламаға қойылатын функционалдық талаптарды түсіну үшін құнды құрал болып табылады. Прецеденттердің бірінші нұсқасы жобаны орындаудың ерте сатысында жасалуы тиіс. Прецеденттердің егжей-

тегжейлі нұсқалары осы прецедентті іске асырар алдында тікелей пайда болуы тиіс.

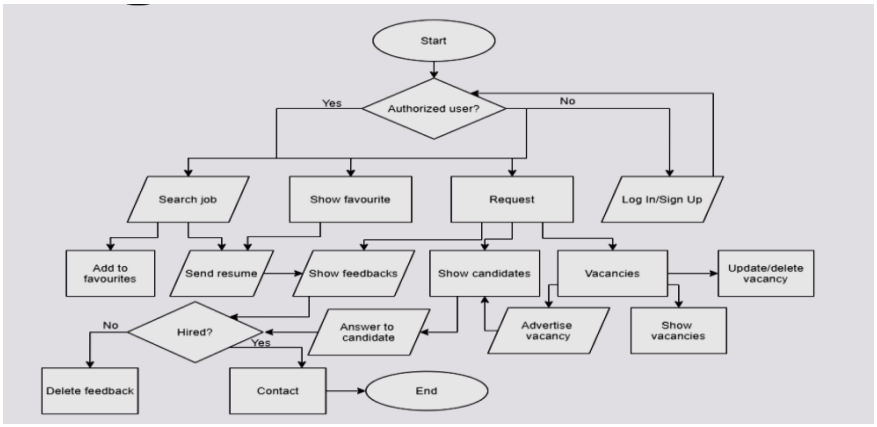
А.3-суретте пайдаланушы үшін прецеденттер диаграммасы көрсетілген. Яғни пайдаланушы келесі әрекеттерді пайдалана алады:



А.3-сурет – Пайдаланушы үшін прецеденттер диаграммасы

А.3-сурет бойынша тіркеуші кез-келген веб-беттен пациент туралы мәліметтерді енгізе алады.

Кезектілік (Sequence) диаграммаларын бір прецедент шеңберінде бірнеше объектілердің әрекетін қарау қажет болған кезде қолдану керек. Кезектілік диаграммалары объектілердің өзара әрекеттесуін ұсыну үшін жақсы, бірақ әрекетті дәл анықтау үшін өте қолайлы емес. А.3-суретте пайдаланушы үшін кезектілік диаграммасы көрсетілген. Яғни пайдаланушы прецедентті әрекеттерді пайдалана алады:

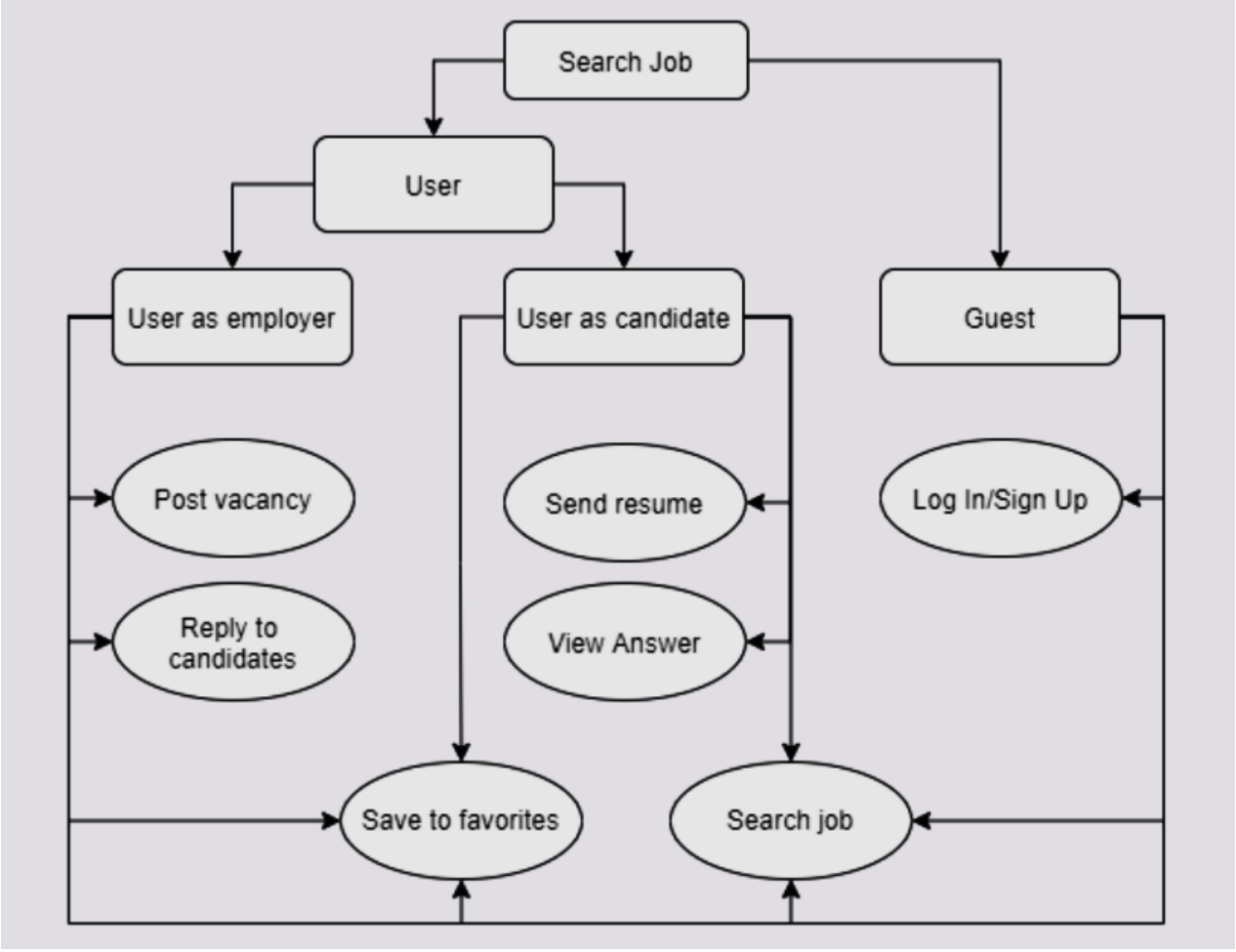


А.4-сурет – Пайдаланушы үшін тізбектелген диаграммасы

UML тілінде физикалық мәндерді ұсыну үшін арнайы компонент қолданылады. Компонент кейбір интерфейстер жиынтығын жүзеге асырады және модельдің физикалық көрінісі элементтерін жалпы белгілеу үшін қызмет етеді.

Навигация жүйесі – бұл қандай да бір тәртіппен орналасқан бөлімдерге немесе сайттарға сілтейтін сілтемелерден тұратын Web-парақтың көрінісі.

Навигациялық жүйе жалпы екі бөлімнен тұрады. Клиенттік бөлімнің навигациялық сұлбасы және басқару бөлімінің навигациялық сұлбасынан тұрады. Тіркеуші бөліміне навигациялық сұлба А.5-суретте веб-беттердің негізгі компоненттері көрсетілген:

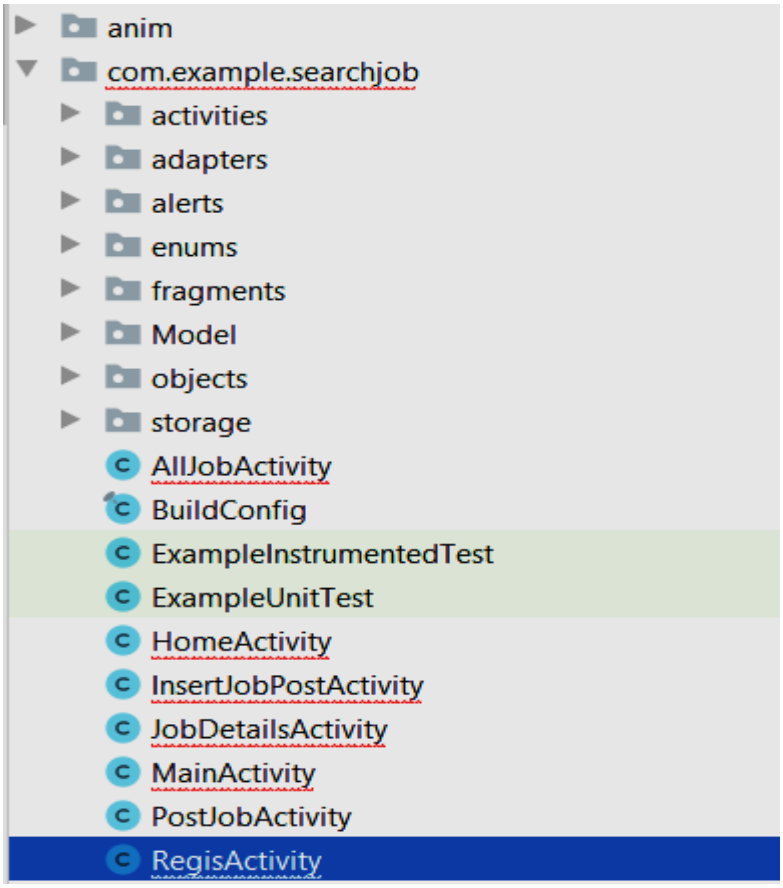


А.5-сурет – IS диаграммасы

А.5-сурет бойынша мобильді қосымшаның 3 компоненттері бар, солардың ішінде 2 компонент модульдерден тұрады (олар: Image Target және AR Camera). Барлық компоненттер Vuforia пакетінің құрылымдық директориясында орналасқан.

Программа құрылымы бэкенд бойынша 12 кластан тұрады. Оның 5 класы негізгі кластар және 4 интерфейстен тұрады.Интерфейстер кластармен қарым қатынасты жүзеге асырады. Config пакетінде кластармен фрондендтен келген сұраныстарды жұмыс істетеді.Пакетте конфигурациялық кластар орналасқан, А.6-суретте көрсетілген. Controller пакетінде жүйені басқарушы кластар және оны жүзеге асыратын анотациялар орналасқан.

Фронтенд бөлімінің файлдары templates пакетінде орналысқан және .ftl кеңейтілімінде сақталаған.Ол freemarker шаблондаушысының кеңейтілімі. А.6-суретте көрсетілген құрылым бойынша parts пакетінің ішінде 4 негізгі шаблондар орналасқан.Ал оның сыртындағы 7 файлда макросталған веб-файлдары орналасқан.А.6-суретте жобаның пакеті көрсетілген:



А.6-сурет – Жобаның пакеті

Фронтенд бөлімінде login файлында қолданушының логин пароль енгізу жолдары қосылған.Яғни ол файлдарды бэкенд бөлімінде Android Studio контроллері керекті файлдан келетін сұранысты қажетінше өңдеп осы URL ден келген бетке қайта жібереді, А.6-суретте көрсетілген.

Ә қосымшасы
(міндетті)

Программа листингі

MainActivity.java

```
package com.example.searchjob;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
public class MainActivity extends AppCompatActivity {
   private EditText email, password;
   private Button btnLogin, btnRegis;
   //Firebase
   private ProgressDialog mDialog;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      mAuth = FirebaseAuth.getInstance();
      if (mAuth.getCurrentUser() != null){
         Intent intent = new Intent(MainActivity.this, HomeActivity.class);
         startActivity(intent);
         //finish();
      }
      mDialog = new ProgressDialog(this);
      LoginFunction();
   }
   private void LoginFunction(){
      email = findViewById(R.id.email_login);
      password = findViewById(R.id.login_password);
      btnLogin = findViewById(R.id.btn_login);
      btnRegis = findViewById(R.id.btn_reg);
      btnLogin.setOnClickListener(new View.OnClickListener(){
         @Override
         public void onClick(View view){
            String mEmail = email.getText().toString().trim();
            String pass = password.getText().toString().trim();

            if (TextUtils.isEmpty(mEmail)){
```

```
            email.setError("Write email...");
            return;
        }
        if(TextUtils.isEmpty(pass)){
            password.setError("Write password...");
            //return;
        }
        mDialog.setMessage("Processing...");
        mDialog.show();
        mAuth.signInWithEmailAndPassword(mEmail, pass).addOnCompleteListener(new
OnCompleteListener<AuthResult>(){
            @Override
            public void onComplete(@NonNull Task<AuthResult> task){
                if(task.isSuccessful()){
                    Toast.makeText(getApplicationContext(), "Succesfull!",
Toast.LENGTH_SHORT).show();
                    //Intent intent = new Intent(getApplicationContext(), HomeActivity.class);
                    startActivity(new Intent(MainActivity.this, HomeActivity.class));
                    finish();
                    mDialog.dismiss();
                }else{
                    Toast.makeText(getApplicationContext(), "Failed to login...",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
});
btnRegis.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view){
        //Intent intent = new Intent(getApplicationContext(), RegisActivity.class);
        startActivity(new Intent(MainActivity.this, RegisActivity.class));
        finish();
    }
});
}
}
```

JobDeyailsActivity.java

```
package com.example.searchjob;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;
public class JobDetailsActivity extends AppCompatActivity {
    private Toolbar toolbar;
    //TextView
    private TextView mTitle;
```

```
    private TextView mDate;
    private TextView mDesc;
    private TextView mSkills;
    private TextView mSalary;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_job_details);
        toolbar = findViewById(R.id.toolbar_job_details);
        getSupportActionBar().setHomeButtonEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mTitle = findViewById(R.id.job_details_title);
        mDate = findViewById(R.id.job_details_date);
        mDesc = findViewById(R.id.job_details_desc);
        mSkills = findViewById(R.id.job_details_skills);
        mSalary = findViewById(R.id.job_details_salary);
        Intent intent = getIntent();
        String title = intent.getStringExtra("title");
        String date = intent.getStringExtra("date");
        String desc = intent.getStringExtra("description");
        String skills = intent.getStringExtra("skills");
        String salary = intent.getStringExtra("salary");
        mTitle.setText(title);
        mDate.setText(date);
        mDesc.setText(desc);
        mSkills.setText(skills);
        mSalary.setText(salary);
    }
}
InsertJobActivity.java
package com.example.searchjob;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.example.searchjob.Model.Data;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import java.text.DateFormat;
import java.util.Date;
public class InsertJobPostActivity extends AppCompatActivity {
    private Toolbar toolbar;
```

```
private EditText job_titles, job_description, job_skills, job_salarys;
private Button button_post_job;
//Firebase
private FirebaseAuth mAuth;
private DatabaseReference mJobPost;
private DatabaseReference mPublicDatabase;
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_insert_job_post);

   toolbar = findViewById(R.id.insert_job_toolbar);
   //setSupportActionBar(toolbar);
   //getSupportActionBar().setTitle("Post Job");
   getSupportActionBar().setHomeButtonEnabled(true);
   getSupportActionBar().setDisplayHomeAsUpEnabled(true);
   mAuth = FirebaseAuth.getInstance();
   FirebaseUser mUser = mAuth.getCurrentUser();
   String uId = mUser.getUid();
   mJobPost = FirebaseDatabase.getInstance().getReference().child("Job Post").child(uId);
   mPublicDatabase = FirebaseDatabase.getInstance().getReference().child("Public database");
   InsertJob();
}
private void InsertJob() {
   job_titles = findViewById(R.id.job_title);
   job_description = findViewById(R.id.job_desc);
   job_skills = findViewById(R.id.job_skill);
   job_salarys = findViewById(R.id.job_salary);
   //Button
   button_post_job = findViewById(R.id.btn_job_post);
   button_post_job.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
         String title = job_titles.getText().toString().trim();
         String description = job_description.getText().toString().trim();
         String skills = job_skills.getText().toString().trim();
         String salary = job_salarys.getText().toString().trim();
         if (TextUtils.isEmpty(title)) {
            job_titles.setError("Failed title...");
            return;
         }
         if (TextUtils.isEmpty(description)) {
            job_description.setError("Failed description...");
            return;
         }
         if (TextUtils.isEmpty(skills)) {
            job_skills.setError("Failed skills...");
            return;
         }
```

```
            if (TextUtils.isEmpty(salary)) {
                job_salarys.setError("Failed salary...");
                return;
            }
            String id = mJobPost.push().getKey();
            String date = DateFormat.getDateInstance().format(new Date());
            Data data = new Data(title, description, skills, salary, id, date);
            mJobPost.child(id).setValue(data);
            mPublicDatabase.child(id).setValue(data);
            mPublicDatabase.child(id).setValue(data);
            Toast.makeText(getApplicationContext(), "Successfull",
Toast.LENGTH_SHORT).show();
            //Intent intent = new Intent(getApplicationContext(), PostJobActivity.class);
            startActivity(new Intent(InsertJobPostActivity.this, PostJobActivity.class));
            finish();
        }
    });
  }
}
HomeActivity.java
package com.example.searchjob;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import com.google.firebase.auth.FirebaseAuth;
public class HomeActivity extends AppCompatActivity {
  private Button btn_alljob;
  private Button btn_postjob;
  //Toolbar
  private Toolbar toolbar;
  //Database
  private FirebaseAuth mAuth;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);
    //Database
    mAuth = FirebaseAuth.getInstance();
    //Toolbar
    toolbar = findViewById(R.id.toolbar_home);
    //setSupportActionBar(toolbar);
    //getSupportActionBar().setTitle("Post Job");
```

```
        btn_alljob = findViewById(R.id.btn_all_job);
        btn_postjob =  findViewById(R.id.btn_post_job);
        btn_alljob.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                //Intent intent = new Intent(getApplicationContext(), AllJobActivity.class);
                startActivity(new Intent(HomeActivity.this, AllJobActivity.class));
                finish();
            }
        });
        btn_postjob.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                //Intent intent = new Intent(getApplicationContext(), PostJobActivity.class);
                startActivity(new Intent(HomeActivity.this, PostJobActivity.class));
                finish();
            }
        });
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.mainmenu, menu);
        return super.onCreateOptionsMenu(menu);
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()){
            case R.id.logout:
                //Intent intent = new Intent(HomeActivity.this, MainActivity.class);
                startActivity(new Intent(getApplicationContext(), MainActivity.class));
                //finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
AllJobActivity.java
package com.example.searchjob;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.example.searchjob.Model.Data;
import com.firebase.ui.database.FirebaseRecyclerAdapter;
import com.google.firebase.database.DatabaseReference;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.TextView;
import com.google.firebase.database.FirebaseDatabase;
public class AllJobActivity extends AppCompatActivity {
   private Toolbar toolbar;
   //Recycler
   private RecyclerView recyclerview;
   //Firebase
   private DatabaseReference mAllJobpost;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_all_job);
      toolbar = findViewById(R.id.all_job_posts);
      /*setSupportActionBar(toolbar);
      getSupportActionBar().setTitle("All job Post");*/
      getSupportActionBar().setHomeButtonEnabled(true);
      getSupportActionBar().setDisplayHomeAsUpEnabled(true);
      /Database
      mAllJobpost = FirebaseDatabase.getInstance().getReference().child("Public database");
      mAllJobpost.keepSynced(true);
      recyclerview = findViewById(R.id.recycler_all_job);
      LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);
      linearLayoutManager.setStackFromEnd(true);
      linearLayoutManager.setReverseLayout(true);
      recyclerview.setHasFixedSize(true);
      recyclerview.setLayoutManager(linearLayoutManager);
   }
   @Override
   protected void onStart() {
      super.onStart();
      FirebaseRecyclerAdapter<Data, AllJobPostViewHolder>adapter = new
FirebaseRecyclerAdapter<Data, AllJobPostViewHolder>
            (
                  Data.class,
                  R.layout.alljobpost,
                  AllJobPostViewHolder.class,
                  mAllJobpost
            ) {
         @Override
         protected void populateViewHolder(AllJobPostViewHolder allJobPostViewHolder, Data
model, int position) {

            allJobPostViewHolder.setJobTitle(model.getTitle());
            allJobPostViewHolder.setJobDate(model.getData());
            allJobPostViewHolder.setJobDesc(model.getDesc());
            allJobPostViewHolder.setJobSkills(model.getSkills());
            allJobPostViewHolder.setJobSalary(model.getSalary());
            allJobPostViewHolder.myview.setOnClickListener(new View.OnClickListener() {
               @Override
```

```
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(),JobDetailsActivity.class);
                intent.putExtra("title", model.getTitle());
                intent.putExtra("date", model.getData());
                intent.putExtra("description", model.getDesc());
                intent.putExtra("skills", model.getSkills());
                intent.putExtra("salary", model.getSalary());
                startActivity(intent);
                finish();
            }
        });
    }
  };
  recyclerview.setAdapter(adapter);
 }
 public static class AllJobPostViewHolder extends RecyclerView.ViewHolder{
  View myview;
  public AllJobPostViewHolder(View itemView) {
   super(itemView);
   myview = itemView;
  }
  public void setJobTitle(String title){
   TextView mTitle = myview.findViewById(R.id.all_job_title);
   mTitle.setText(title);
  }
  public void setJobDate(String data){
   TextView mDate = myview.findViewById(R.id.all_job_date);
   mDate.setText(data);
  }
  public void setJobDesc(String desc){
   TextView mDesc = myview.findViewById(R.id.all_job_desc);
   mDesc.setText(desc);
  }
  public void setJobSkills(String skill){
   TextView mSkill = myview.findViewById(R.id.all_job_skill);
   mSkill.setText(skill);
  }
  public void setJobSalary(String salary){
   TextView mSkill = myview.findViewById(R.id.all_job_salary);
   mSkill.setText(salary);
  }
 }
}
```

RegisActivity.java

```
package com.example.searchjob;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.text.AutoText;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
public class RegisActivity extends AppCompatActivity {
    private EditText emailReg, passReg;
    private Button btnReg, btnLogin;
    //Firebase auth
    private FirebaseAuth mAuth;
    private ProgressDialog mDialog;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_regis);
        mAuth = FirebaseAuth.getInstance();
        mDialog = new ProgressDialog(this);
        registration();
    }
    private void registration(){
        emailReg = findViewById(R.id.email_registration);
        passReg = findViewById(R.id.registration_password);
        btnReg = findViewById(R.id.btn_regis);
        btnLogin = findViewById(R.id.btn_log);
        btnReg.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                String email = emailReg.getText().toString().trim();
                String pass = passReg.getText().toString().trim();
                if(TextUtils.isEmpty(email)){
                    emailReg.setError("Please write email!");
                    return;
                }
                if(TextUtils.isEmpty(pass)){
                    passReg.setError("Please write password!");
                    return;
                }
                mDialog.setMessage("Processing...");
                mDialog.show();
                mAuth.createUserWithEmailAndPassword(email, pass).addOnCompleteListener(new
OnCompleteListener<AuthResult>(){
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task){
```

```
                if (task.isSuccessful()){
                    Toast.makeText(getApplicationContext(), "Succesfull!",
Toast.LENGTH_SHORT).show();
                    //Intent intent = new Intent(getApplicationContext(), HomeActivity.class);
                    startActivity(new Intent(RegisActivity.this, HomeActivity.class));
                    finish();
                    //finish();
                    mDialog.dismiss();
                }else{
                    Toast.makeText(getApplicationContext(), "Failed...",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
  });
  btnLogin.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view){
      //Intent intent = new Intent(getApplicationContext(), MainActivity.class);
      startActivity(new Intent(RegisActivity.this, MainActivity.class));
      finish();
    }
  });
 }
}
```