ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
коммерциялық емес акционерлік қоғамы
IT-инжиниринг кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 ж.

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: Android платформасында «Электрондық кітапхана» мобильдік қосымшасын әзірлеу

Мамандығы: 5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету»

Орындаған: Керманалиева Ұ.А. Тобы: ВТк-15-1
Ғылыми жетекші: т.ғ.к., доцент Набиева Г.С.

Кеңесшілер:
Экономикалық бөлім: э.ғ.к., профессор _____ Ж.Г. Аренбаева
«24» 05 2019 ж.

Өміртіршілік қауіпсіздігі: т.ғ.д., аға оқытушы _____ Ш.Ш. Бекбасаров
«24» 05 2019 ж.

Есептеу техникасын қолдану: аға оқытушы _____ Ж.С. Айткулов
«28» 05 2019 ж.

Норма бақылаушы: аға оқытушы _____ К. Мукапил
«29» 05 2019 ж.

Сын-пікір беруші: ф.-м.ғ.к., доцент _____ Б.С. Каленова
«____» _____ 2019 ж.

Алматы 2019

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
коммерциялық емес акционерлік қоғамы

Басқару жүйелері және ақпараттық технологиялар институты

IT-инжиниринг кафедрасы

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

Дипломдық жобаны орындауға берілген
ТАПСЫРМА

Білім алушы Керманалиева Ұлдана Амантайқызы

Жобаның тақырыбы: Android платформасында «Электрондық кітапхана» мобильдік қосымшасын әзірлеу

2018 жылғы «26» қазан № 124 университет бұйрығымен бекітілген.

Аяқталған жобаны тапсыру мерзімі: «24» мамыр 2019 ж.

Дипломдық жобаның бастапқы мәліметтері (зерттеу (жоба) нәтижелерінің талап етілген параметрлері мен объектінің бастапқы мәліметтері): Ұсынылып отырған дипломдық жобада Android платформасында «Электрондық кітапхана» мобильдік қосымшасын әзірлеу. Жобаны орындау барысында Android Studio бағдарламасын қолданамын.

Дипломдық жобада қарастырылған мәселелер тізімі немесе дипломдық жобаның қысқаша мазмұны:

- талдау бөлімі;
- жобалау бөлімі;
- жүзеге асыру және тестілеу бөлімі;
- экономикалық бөлім;
- өміртіршілік қауіпсіздігі;
- А қосымшасы. Техникалық тапсырма;
- Ә қосымшасы. Программа листингі;
- Б қосымшасы. Ендіру актісі.

Графикалық материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс): 12 кесте, 34 сурет ұсынылған.

Ұсынылатын негізгі әдебиеттер:

1 Голощапов А. Google Android программирование для мобильных устройств. Санк-Петербург 2011-438 с.

2 Соснов А. Основы проектирование информационных систем. – М.: ДМК Пресс, 2002. – 1020 с.

3 Марк Шпеник, Оррин Следж. Microsoft SQL Server 2000 DBA Survival Guide. – М.: Вильямс, 2001. – 236 с.

4 Мамаев Е. MS SQL Server. Проектирование и реализация баз данных. Сертификационный экзамен. – СПб: BHV, 2004. – 416с.

5 Плю Р., Стефенс Р., Райан К. Освой самостоятельно SQL за 24 часа. – М.: Вильямс, 2000.

Дипломдық жобаның бөлімдеріне қатысты белгіленген кеңес берушілер

Бөлімдер	Кеңесшілер	Мерзімі	Қолы
Экономикалық бөлім	Аренбаева Ж.Г.	04.03.2019 – 24.05.2019	
Өміртіршілік қауіпсіздігі	Бекбасаров Ш.Ш.	04.03.2019 24.05.2019	
Программалық қамтама	Айткулов Ж.С.	02.04.2019- 28.05.2019	
Норма бақылау	Мукапил К.	04.04.2019 10.05.2019	

Дипломдық жобаны дайындау
КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Талдау бөлімі	29.10.18 – 21.12.18	орындалды
Жобалау бөлімі	03.01.19 – 15.02.19	орындалды
Жүзеге асыру және тестілеу бөлімі	18.02.19 – 12.04.19	орындалды

Тапсырманың берілген күні «29» қаңтар 2018 ж.

Кафедра меңгерушісі _____ Т.С. Картбаев

Жобаның ғылыми жетекшісі _____ Г.С. Набиева

Тапсырманы орындауға алған
білім алушы _____ Ұ.А. Керманалиева

Андатпа

Дипломдық жоба Android операциялық жүйесі негізінде «Электрондық кітапхана» мобильдік қосымша құрастыруға арналған. Интернет желісінсіз қазақ тіліндегі кітаптарды сапалы түрде қолданушыға түсінікті тілде оқи алуға мүмкіндік берілді. Электронды кітаптардың тізімі жасалды. Кітапхананы құрайтын әрбір кітап өңделді.

Жобаны дайындау барысында Android Studio интеграцияланған бағдарлама әзірлеу ортасы және Java бағдарламалау тілі пайдаланылды.

Өміртіршілік қауіпсіздігі мәселесі тексеріліп, ақпараттық жүйені әзірлеу барысында жарықтандыру жүйесін есептеу коэффициенті есептелінді. Сондай-ақ экономикалық бөлімінде жұмыстың орындалу құны, еңбек бағасы және жұмыстың тиімділігі анықталды.

Аннотация

Дипломный проект предназначен для разработки мобильного приложения «Электронная библиотека» на базе операционной системы Android. Книги без сети интернет на казахском языке получили возможность читать на понятном пользователю языке. Был создан список электронных книг. Каждая книга, которая составляет библиотеку, была обработана.

При подготовке проекта использовалась интегрированная среда разработки приложений Android Studio и язык программирования Java.

Была проверена проблема безопасности жизнедеятельности, в ходе разработки был рассчитан коэффициент расчета освещении. Также в экономическом отделе была определена стоимость работ, стоимость рабочей силы и эффективность работы.

Annotation

The graduation project is designed to develop a mobile application "Electronic Library" based on the Android operating system. Without the Internet, books in the Kazakh language were able to read in a language understandable to the user. A list of e-books has been created. Each book that makes up the library has been analyzed.

In preparing the project, the integrated application development environment Android Studio and the Java programming language were used.

The problem of life safety was checked, during the development the calculation coefficient for lighting was calculated. Also in the economic department was determined the cost of work, the cost of labor and work efficiency.

Мазмұны

Кіріспе

Бүгінгі таңда Android мобильдік бағдарлама әзірлеу, қарқынды дамып келе жатқан бағдарламалардың бірі болып табылады. Android операциялық жүйесі көптеген смартфондарда орнатылған. Қазіргі заманда өзекті мәселелердің бірі - ғаламтор бағдарламалары тез дамып жатыр. Соған байланысты ғаламтор екпінді көтеріле бастады және пайдаланушылардың сұранысы бойынша, бағдарламалар жаңартылып отыр. Платформалары арасында жетекшы орынды Android және IOS платформалары алады. Дипломдық жобаның мақсаты - Android операциялық жүйесі негізінде электрондық кітапхана мобильдік бағдарламасын құрастыру. Интернеттің адамдар арасында кең қолданысқа ие болуы және интернеттің жылдамдығының артуына байланысты қосымшаларда жұмыс жасау үлкен сұранысқа ие. Әрбір адамның қолында смарфондардың болуы интернет арқылы адамдарға қызмет көрсетуді өте ыңғайлы етеді. Университет аумағында студенттердің , оқытушылардың және қызметкерлердің сұраныс алмасуларын жеңілдету мақсатында дипломдық жұмысымды сұраныс жасау бағытында алдым.

Қазіргі таңда Android мобильдік қосымшаларын құру жақсы дамып келе жатқан бағдарламалардың бірі болып саналады. Android операциялық жүйесі бірнеше смартфондарда орнатылған. Оларға: Sony, Samsung, Lenovo және тағы да басқалары жатады және де ғаламтор – бағдарламаларының тез дамуы өзекті мәселеге айналып отыр. Соған байланысты ғаламтор пайдаланушыларының сұранысы бойынша, бағдарламалар жаңартылып жатыр. Осылардың ішінде Android және IOS платформалары жетекші орын алады.

Дипломдық жобаның мақсаты – Android операциялық жүйесі негізінде мобильдік бағдарлама құрастыру. Мысал ретінде «Электрондық кітапхана» атты жоба оқырмандарға арналған. Зерттеулер бойынша, осы тематикаға байланысты бағдарламалар, яғни қазақша кітаптар жиыны ғаламторда көп кездесе бермейді. Сол себепті ең басты мәселе ғаламтор желісі көмегінсіз кітап оқи алу болып табылады. Android – операциялық жүйесі планшеттерге, смартфондарға орнатылады. Бағдарламаларды өңдеудегі негізгі тілдердің бірі – Java өңдеу кітапханасы. Қосымшаны жасау үшін Android Studio ортасын қолданамыз, Android SDK құрылғысы SDK эмуляторы арқылы жүзеге асады.

Зерттелу жұмысының өзектілігі – Қазіргі кезде мобильді құрылғыларға арналған, көпшілік білетін операциялық жүйелердің бірі Google компаниясының өнімі Android ОЖ болып табылады.

Зерттелу жұмысының мақсаты мен міндеті – Бұл жұмыстың мақсаты адамдарға арналған Android операциялық жүйесіне арнап қосымша құру болып табылады. Android SDK құрылғысы қосымшаның өзінде жүзеге асырылатын болады.

1 Жобаны талдау

1.1 Android платформасы туралы

Жобаның негізгі мақсаты – Android операциялық жүйесі негізінде электрондық кітапхана мобильдік бағдарламасын құрастыру. Бұл тақырыпты таңдау себебім кітапханаларға барып қажетті мәліметтерді іздеу адамның уақытын алады, сондықтан электронды түрде кітаптарды қарау, керекті мәліметтерге тез әрі ыңғайлы қазіргі заманға сай қол жеткізу болып табылады. Оқытушылар немесе студенттер өзіне қажетті құжатқа онлайн сұраным жасап , смартфон немесе компьютер арқылы құжатының немесе сұралымының дайын болғандығы туралы хабарлама ала алады. Кітапханалардың келуімен кітапты кез-келген адамға оқуға болады. Бұл үшін кітапханаға келіп, орналасып , оқуға тура келеді. Кітапханаларда электронды технологияны дамыту арқылы веб-парақшалар пайда болды және оқырманға қол жетімді компьютерді табу арқылы интернеттен мағлұмат алуға және оқуға ыңғыйлы болды. Қазіргі уақытта адамдардың көпшілігі күнделікті ұялы телефонды пайдаланады. Және мобильдік қосымшаны пайдалану кез келген жерде ыңғайланып отырып электронды кітапхананы қолдану оңайырақ болады.

Ұялы телефондар бұрын-соңды болмаған нәрседен бас тартты және олардың функциясы өте жақсы бақыланды - бұл адамдар арасындағы қарым-қатынас құралы. Сонымен қатар, жақында пайда болды, бірақ біздің өмірімізде жақсы орнатылған смартфондар соншалықты функционалды, олар білмейтінін айту қиын: бұл ойыншы, камера, интернет ресурстарын пайдалану мүмкіндігі және тағы басқалар. Шындығында, барлық смартфондар компьютердің кішкентай көшірмесіне айналды, сіз әрқашан сізбен бірге жүре аласыз.

Бүгінгі күні күнделікті өмірде де, шетелге сапарларда да ыңғайлы смартфондар, коммуникаторлар, планшеттік компьютерлер және басқа да құрылғылардың түрлері Android ОС-на негізделген. Операциялық жүйенің таралуының себептері қандай?

Біріншіден, Android әртүрлі өндірушілердің көптеген құрылғыларын қолдайды, екіншіден, Android-де даму құралдарының қолжетімділігі жоғары. Android-ке арналған тегін платформаның дамуына арналған құралдар, мысалы, iPhone-ның (Apple-дан) дамуы кезінде көптеген бастапқы қаржы инвестицияларын талап етеді. Сондай-ақ, Android OS артықшылығы - үшінші тарап ресурстарымен жұмыс істеу үшін тегін кітапханалардың болуы (YandexMapKit, GoogleMapAPI және т.б.) ал WindowsPhoneMobile үшін мұндай кітапханалар кең таралған емес .

Ескі телефонның негізгі міндеттері – қоңырау шалу және қабылдау, SMS жазу. Бүгін бұл міндеттер интернет-ресурстармен толықтырылды, музыка тыңдау, суретке түсіру, ойындар мен қолданбаларды пайдалану.

Мобильді болашақ, ең алдымен, негізінен түрлі техниканы басқару, өз иесінің денсаулығын бақылау және де көптеген тағы да басқа функцияларды көруге мүмкіндік береді. Аталған функциялардың кейбірі қазіргі заманғы телефондарда - оларды құру саласындағы прогресс, ең алдымен, олардың ішінде қарапайым аймаққа қажет.

Кітапхана – оқырман мен кітап арасындағы негізгі алтын көпір. Бір ғажабы,осы заманғы ақпарат тасымалдаушы технология жүйесінде де кітапхана маңызды буын болып саналады.

Қазіргі кезде адамдардың өскелең талабын қанағаттандыру, рухани байлығы мен жалпы қабілетін дамыту және жоғары эстетикалық талғамын қалыптастыру міндеттерін іске асыруда кітапханалардың алатын орны ерекше. Кітапхана мәдениет саласы ретінде ақпараттық, білім беру және мәдени-ағарту қызметінің негізгі болып табылады. Кітапханаларда оқу феноменін тиімді пайдалану мен дамытудың сан ғасырлар бойы қалыптасқан бай тәжірибесі сақталған. Олар кітаптарда жинақталған адамзат жадының, ұлт жадының алтын қоры, ал кітапханашылар – сол теңдесіз қазынаның шырақшылары іспеттес. Кітапхана ғасырлар бойы келе жатқан шырағы бтік парасатталық пен білімділіктің киелі ордасы. Заманымыздың әр кезеңінде кітапханашы қызметі туралы ұғым жоғары болуы тиіс. Кітапханашы кітапханада тек кітап береді деген тар ұғым орнына, қоғамда кітапханашы мамандығының қажеттілігін мойындайтын оң пікірлер мен көзқарастар қалыптастыру керек.

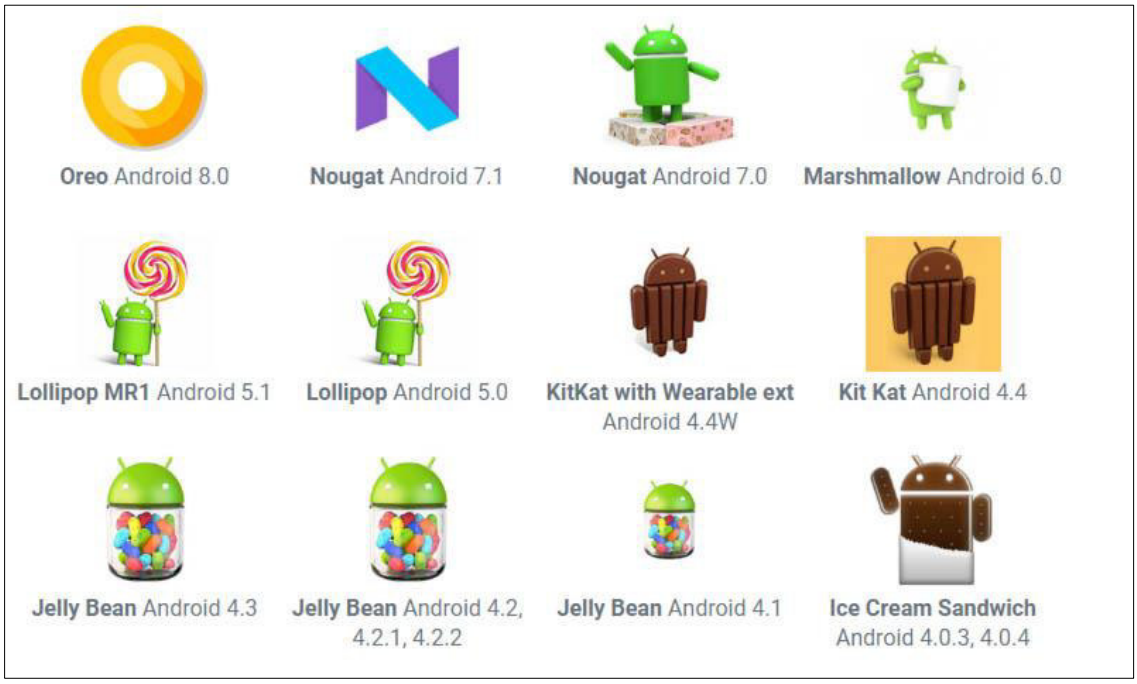
Бүгінде бүкіл дүниежүзілік мұраны іс жүзінде арнайы орын не жай бөлмей-ақ, үйге не кеңсеге сыйдыруға болады. Электронды кітапханаға Интернетке қосылған кез-келген компьютер не мобильді құрылғы арқылы қол жеткізе аласыз. Қажетті кітапты табу қиындық туғызбайды. Электронды кітапхананың міндетті элементтерінің бірі болып баспалдақты құрылымды электронды каталог саналады, ол кітапты қосу не жою шараларына байланысты автоматты түрде толықтырылады және жаңартылады. Сонымен қатар, электронды кітап қоймасында іздеу жүйесі қарастырылған, мұнда кітапты атауымен, авторымен, шығу мерзімі және басқа деректерге байланысты тауып алуға болады.

Android Lunix жүйесіндегі ең бірінші мобильдік жүйесі жасалған платформа және ол Люникс және бағдарлама жасаушылардың ішінде, көптеген қолданушылардың арасында мобильдік бағдарламаның атағын көтерген. Себебі, Linux файлдық қауіпсіздіктің саясат концепциясы кез-келген жүйе файлы үшін категория қолданушылардан тұрады деп пайымдайды: файлдың өз иесі немесе жасаушысы, қандай болса да қолданушылар тобы, оған көп жағдайда файл иесі және тағы басқалар кіреді. Осылайша бәрі рұқсат етілген қолданушы немесе файл иесі, тек ол ғана қол жеткізу құқығын өзгерту мүмкіндігіне ие болғандықтан, файлдық жүйе қауіпсіздігінің саясатын құрып, файл иесі үшін жеке құқығын анықтайды, қолданушы топтар үшін де, қалған жүйе қолданушылар үшін де. Заңды түрде Android платформасы 2003 жылдың 23 қыркүйегінде құрастырылды. Ең бірінші версиясы операциялық жүйенің

индексімен 1.0 Apple Pie атымен шықты. Android операциялық жүйесінің ядросы Linux-та жасалған және ол өзін ашық мобильбік платформа ретінде көрсетті.

Көптеген гаджет бағдарламасын жасаушылар «Жасыл роботты» тегін қолдана алады және жаңа Android операциялық жүйесін. Көлемді түрде бағдарламаны тарату үшін, Google компаниясы Open Handset Alliance организациясын негіздейді. Бұл организацияға тез арада көптеген үлкен дүниежүзілік смартфон өндірушілер қосылады, оларға жиынтықталады, тағыда мобильдік операторлар қосылады.

Android операциялық жүйесі барлық заманауи мобильді платформалардың арасында даусыз фаворит болып табылады. Ол пайдаланушыларға гаджетті жұқа күйге келтіруге мүмкіндік береді және өмірдің түрлі салаларында пайдалы Мобильді Ойын-сауық пен бағдарламалар әлеміне есік ашады. "Жасыл Робот" Олимпқа ұзақ және терең жол өтуге тура келді, ал әрбір жаңа нұсқа пайдаланушыларға мейірімді және жұмыста түсінікті болды.



1.1-сурет – Android нұсқалары

Үш айдан кейінгі Android 1.0 презентациясы, дүниежүзілік нарыққа бірінші рет смартфондық операциялық жүйені енгізді. Құрылғының аты – HTC Dream (T-mobile G1). Осыған дейін аппарат қомақты техникалық характеристикасын көрсетті. Оларға тоқталып өтсек: дисплейдің рұқсаты 3,2 дюйм, тактілік жиілігі 525 МГц процессор, жедел жадтың көлемі 192 мегабайт және камера матрицасының рұқсаты 3,2 МП.

Логотипқа қатысты жасалу тарихы өте қызықты. Бағдарламаның логотипін құрушы 2007 жылы ұзақ уақыт бойы лайық бейнесін таптай жүрді,

көптеген нұсқаларды қарастырды: фильмдерден, ойыншық дүкендерінен және интернет желісінен. Нәтижесіне орай, прототип ретінде туалет кабинкасындағы отырған адам алынған. Қарапайым көрініс басындағы екі антенасымен. Бұл мобильдік платформаны көрсетеді, ол ашық және жеңіл болуы қажет.

Смартфонның интерфейсі Gingerbread, кішкене созылған болып, ал Honeycomb қалып отырды. Ол компанияға аса табыс алып келмеді. Планшеттерге арнап көптеп бағдарламаларды қазір де шығарады, табысы аз болса да. Шынайы, күшті әсерді Samsung компаниясы өзінің көрінісінде IFA 2011 ұсынды. Ол Galaxy Note-ке қарап жобаланды. Бірінші 5 дюймдық смартфон, Корей компаниясы шығарған форм-фактор смартфон, өзінің компаниясына үлкен табыс алып келді. 2011 жылғы ең жақсы соңғы жаңалықтардың бірі, бұл күмәнсіз, екі Android версияның қосылуы болды: планшет пен смартофнның.Екі класс та құрылғының жаңартылған 4.0 Ice Cream Sandwich версиясында 19 жақсы жұмыс істеді. Бұл жүйе PC Magazine басылымның сыйлығын жеңіп алды, осы платформа көптеген өзгерістер алып келіп смартфонды толығымен өзгертті.

Тағы да айта кететін болсақ, 2012 жылы осы операциялық жүйе, жылдық көріністе бірінші орынға ие болды, ең жақсы платформа ретінде. Тағы бір ерекшілігі және жаңалығы, осы жылғы ICS екі архитектуралы процессорды қолданды Intel x86 және MIPS, олар ARM қосылған болып келді.

Android 4.0 бағдарламасын жасау уақытында, Google жұмыс берушісі бұл жасап жатқап платформаны бренд ретінде жарялады. Ол Galaxy Nexus моделі болып танылды. Атын әйгілі етіп тұрған бағдарлама жасаушы Samsung компаниясы болды. 2012 жылдың басындағы жаңалықтың бірі, бұл NVIDIA компаниясының төрт ядролы ARM процессоры Tegra 3 атымен әйгілі болды. Екіншіден Qualcomm компаниясы кейінрек осыған ұқсан чипсетті шығарды. Тағы да осы жылдың басында екі смартфон шығады деп жарияланды, ол HTC One X және Optimus 4X HD. Бірақ, екі смартфонның атағын Samsung компаниясының Galaxy S III аты смартфоны алды. Планшет дүкенге байланысты айта кетерлік жай, ең көп сатылған планшеттердің бірі болып ASUS компаниясының Nexus 7 атты моделі танылды. Оның бағасы 200 доллар және планшет жаңартылған Android 4,1 операциялық жүйесімен.

Android платформасы үшін қосымшаларды әзірлеу Android SDK жиынымен ұсынылған құрал-саймандар тобымен сабақтас. Сонымен бірге қосымшалар әзірлеу үшін құрал-сайманға Java SE (JDK) және интеграцияланған әзірлеулер ортасы керек болады. Соңғы үлгі ретінде Android IDE қолдану қабылданған, бірақ атап өтетін нәрсе, қосымшалардың әзірлеуді қарапайым мәтіндік редактормен және басқа да IDE көмегімен жүргізуге болады, сонымен қатар скриптердің және командалық жолдар қолдану арқылы құрал-саймандарды шақыруға болады. Бірақ Andrpod -те әзірлеу ыңғайлы әдіс болып табылады. ебебі, біріншіден бұл орта қажетті аспаптарға тікелей бйланыс жасай алады, ал – екіншіден, ол үшін арнайы Android Development Toolkit (ADT) плагині бар, Android IDE көмегімен

Android платформасына қосымшалар құруға ыңғайлы өтулермен қамтамасыз етеді.

1.2-сурет – Android 4.0 бағдарламасы

Project Butter технологиясы Google компаниясының тоқтап қалатын операциялық жүйесінен құтылып, өзінің жаңа операциялық жүйесін іске асырды. Бес жылдан кейін, яғни бұл 2012 жылдың соңына қарай Google компаниясы тағыда екі брендтық девайстарды шығарды, ол планшет Samsung Nexus 10 және смартфон LG Nexus 4. Айта кететін жай 2013 жылғы смартфондар үлкен дисплейға және өте тез жұмыс атқаратын процессорға ие болды.

1.2 Android операциялық жүйесіндегі бағдарлама жасау өзектілігі

Android негізінде бірегей, өзгеше операциялық жүйе. Жақсы нәтижеге жету үшін бағдарлама әзірлеуші операциялық жүйенің өзектілігін және өзгешілігін білуі қажет. Бағдарлама әзірлеу кезінде бірнеше қиындықтарды ескеру қажет. Оларға тоқталып өтейік:

– бағдарламаны қондыру кезінде, бағдарлама екі есе немесе төрт есе орын алады, яғни ол бағдарламаның түпнұсқалық орның қажет етеді;

– орнатылған флеш-картамен жұмыс істеу кезінде файлдың жылдамдығы он есе төмендейді, егер бос орын аз болса;

– әр процесс кезінде жедел жадтан 16Мб (кейде 24 Мб) қолданылады.

Android негізінде Linux операциялық жүйесінде жасалған. Бағдарламаның және ядроның арасында API қабаты және нативті кодтағы

кітапхана орналасқан. Бағдарлама виртуалды машинада Java (Dalvik Virtual Machine) орындалады.

Android — та көп теген бағдарламалар қосуға болады. Бірақ бір бағдарлама толығымен букіл экранды алады. Ағымдағы бағдарламаны қолданып отырып басқа бағдарламаны қосуға немесе жаңасын ашуға болады. Ол тарихын қоруге болатын браузерге ұқсайды.

Қолданушының интерфейсі әр экранда Activity класс кодындағы көрініст көрсетеді. Процесста әр түрлі Activity тұтынады немесе қолданылады. Процесстаң көрі Activity көбірек өмір сүреді, яғни ол дегеніміз Activity қолдануы көбірек. Activity жұмыс істеу кезінде тоқтатылып және жаңадан қосылып бүкіл қажетті информацияны сақтап қосылады.

Android арнаулы механизімді қолданып әрекеттің сипаттамалары бас Intent негізде жүргізіледі. Белгілі әрекетттер істелген кезде (қоңырау соғу,хабарлама жіберу, терезені көрсету), Intent шақырылады.

Тағы да Android демоға ұқсас Linux серверларын ұстайды, олардың атқару қызметі керекті әрекетті фондық режимде (мысалы, өленнің ойналуы). Бағдарламада деректерді айырбастау үшін Content providers (провайдері қолданылады). Айтылмыш жұмыс үшін деректер провайдерін пайдаланып, қолданушының құрылғысына орнату керек [1].

1.3 Android платформасына қосымша әзірлеу және әзірлеу құрал-саймандары

Кез келген платформаға программа жазу үшін бірінші құрал саймандарды әзірлеуден бастаймыз. Жұмыс үшін қажетті құрал-саймандар белгіленіп және оларды баптаған соң, енді кез-келген бір бағдарлама жазуға кірісеміз. Android-қа арналған аспаптарды қарамастан бұрын, осы кезеңге қажетті ортақ ұғымдармен танысып алайық.Алғашқы негізді ұғым бұл – Software Development Kit («devkit») немесе SDK [2].

Бұл аппараттық немесе программалық платформаны, операциялық немесе компъютерлік жүйелердің белгілі бір бағдарлама пакеті үшін – кейбір архитектуралар бойынша қосымша әзірлеуге мүмкіндік беретін, әзірлеу құралдарының жиыны ретінде түсіндіріледі. SDK – өзіне талқылау құралдары мен анықтамалық материалдарды қоса отырып, әзірлеушіге кең көлемді құрал- жабдықтар жиынын ұсынады. (мысалы кодтар, ескертулер және т.б). SDK көбінесе мақсатты платформа әзірлеушілерінен тікелей ұсынылады және онда оның ерекшеліктер есепке алынады. Сондай-ақ мұндай құрал-саймандар жағдайда тегін таралуы мүмкін. Бұл платформаға өзге әзірлеушілердің есебінен алуан түрлі қосымша атануға көмектеседі.

Сөз әзірлеу аспаптары туралы жүретін кезде, жиі кездестіруге болатын ұғым, IDE (Integrated development environment) ұғымы – интеграцияланған әзірлеулер ортасы. Ол бағдарламашының өнімді максималдау үшін және программалау процессі үшін ыңғайлы орталар шақырған программалық құралдардың жиынтығын ұсынады. Бұл күрделі программалық кешен өзіне

мәтіндік редактор, компилятор немесе интерпретатор, интеграцияланған талқылау және автоматтандыру үшін қажетті құралдар жиынтығын қоса алады. Бөтен программамен қамтамасыз ету үшін интергрияның мүмкіндіктері жиі қолданылады.Мысалы, жобалау құралдары немесе болжамдарды бақылау. Сондай-ақ IDE-де жасап жатқан бағдарламаның графикалық түріндегі интерфейсін визуалды редакциялау үшін тез құрастыруға арналған құрал- саймандар бар, сонымен бірге иерархия класындағы диаграммалар сияқты бағдарламалар, браузердің таптық, объекттердің инспектордың немесе қорлардың менеджердің, әзірлеулер интеграцияланған ортасы шақырған БЖ әзірлеу жылдамдық ыңғайлылық және жоғарылату қамтамасыз ету үшін бір программалық кешенге әр түрлі аспаптар топтастыру. Іс жүзінде құрал - саймандардың бір-бірімен байланыспағанын ескере отырып, программалау үшін қандай да бір IDE қолданбауға болады.

1.4 Android платформасында мобильді құрылғыларға қосымшалар құру мүмкіндіктері

Android жүйесі архитектурасының ерекшелігі. Android қосымша жасаудың алдында, жүйенің архитектурасымен және оның негізгі ерекшеліктерімен танысу қажет.

Android жүйесі – операциялық жүйеден, аралық қабаттың бағдарламалық қамтамасыз етуінен (middleware), сонымен бірге негізгі қолданбалы қосымшалардан құралған мобильді құрылғыларға арналған программалық стек. Android архитектурасын төрт деңгейге бөлу қабылданған:

- ядро деңгейі;
- орындалу ортасының кітапханасы деңгейі;
- қосымшалар каркасының деңгейі;
- қосымшалар деңгейі [3].

Android ядросы Linux операциялық жүйенің 2.6–шы нұсқасына негізделген, бірақ Android жүйесі таза күйіндегі Linux–жүйе емес, ол бірқатар ерекшеліктер мен өз жады бөлу механизмдері, процестер арасындағы қатынас және т.б. Android жүйесіне тән ядроның қосымша кеңейтілмелеріне ие. Ядро жабдық пен программалық стектің қалған бөлігі аралысындағы абстракция қабаты болып табылады. Бұл деңгейде процестерді басқару, жадты үлестіру және файл жүйесін басқару тәрізді қызметтер орналасады. Ядро деңгейінің негізгі құрамдас бөліктері:- процессорараліқ өзара әрекеттесу драйвері (IPC Driver); -қоректенуді басқару драйвері (Android Power Management); -мобильді құрылғыны құрамына кіретін жабдықты басқару драйверлі.

Ядро деңгейінен «жоғары», аралық қабат бағдарламалық қамтамасыз етуі ретінде қосымшалар үшін ең маңызды негізгі функционалды қамтамасыз етуі үшін кітапханалар жиынтығы (Libraries) орналасқан. Яғни осы деңгей жоғары жатқан деңгейлерге жүзеге асырылған алгоритмдерді беру, файлдық форматтарды қолдау, ақпаратты кодтау және кері кодтауды жүзеге асыру

(мысалы, мультимедиа кодектері), сурет салу және т.б. үшін жауап береді.Кітапхана C/C++ тілінде жазылған және нақты аппараттық қамтамасыз етуге арналып компиляция жасаған, сондықтан алдын ала орнатылған күйде жеткізіледі.Кейбірін қарастырып өтейік:

Surface Manager – Android жүйесіндегі терезелердің композитті менеджері Linux–тың Compiz менеджеріне ұқсас, бірақ ықшамдалған. Бұл жүйеге терезелердің мөлдірлігі және бір қалыпты өту әсерлерін жасауға мүмкіндік береді;

Media Framework – PacketVideo OpenCORE негізінде жүзеге асырылған кітапхана.Олардың көмегімен жүйе аудио және видео контентті жазу және ойнатуды, сонымен қатар статикалық бейнелерді көрсетуді жүзеге асыра алады; Көптеген кең таралған форматтарды қолдайды: MPEG4, H.264, MP3, AAC, AMR, JPG және PNG[4].

SQLite – Android жүйесінде деректер қорымен жұмыс жасауға арналған жеңіл және өнімді реляциялық ДҚБЖ; — FreeType – бит карталарымен жұмыс жасауға, сонымен бірге шрифттардың растеризациясы және олармен операцияларды жүзеге асыруға арналған кітапхана.Шрифттарды және мәтінді бейнелеу үшін арналған жоғары сапалы қозғағыш;

LibWebCore – Google Chrome және Apple Safari браузерлері пайдаланылатын белгілі WebKit браузер қозғағышының кітапханалары; SSL - аттас криптографиялық хаттаманы қолдау үшін арналған кітапхана. Libc- C тілінің стандартты кітапханасы, атап айтқанда оның Linux негізіндегі құрылғыларда жұмыс жасауға бағыттылған BSD нұсқасы. Bionic атауына ие. Бұл деңгейде Android Runtime орындалу ортасы орналасқан.Оны маңызды құраушылары: ядро кітапханалар жиынтығы және Dalvik виртуалды машинасы. Әр Android қосымшасы өз Dalvik виртуалды машинасында іске қосылады.Осының арқасында барлық жұмыс үстіндегі процестер операциялық жүйеден және бір–бірінен оқшауланған. Жалпы Android Runtime архитектурасы программаға виртуалдық машина ортасы шеңберінде ғана жұмыс жасауға мүмкіндік береді. Осылай операциялық жүйе ядросын оның басқа бөліктерінен келуі мүмкін зияннан қорғаныс жүзеге асырылады. Сондықтан қате код немесе зиянды программа Android–ты және оның базасындағы құрылғыларды бүлдіре алмайды. Мұндай қорғаныс Android Runtime–ның маңызды функциясы болып табылады. Бұл деңгейден жоғары кейде қосымшалар каркасы деп аталатын Application Framework деңгейі орналасады. Қосымшалар каркасы деңгейі арқылы өндірушілер төмен деңгейлерде жататын жүйенің құрамдас бөліктері беретін API–ға рұқсат алады.Одан басқа, фреймворк архитектурасының арқасында кез–келген қосымша басқа қосымшалардың рұқсат етілген жүзеге асырылған мүмкіндіктеріне қол жеткізе алады. Әр қосымшаның негізі және фреймворк бөлігі болып табылатын сервистер мен жүйелердің негізгі жиынына кіреді:

– қосымшаларының визуалды компоненттерін (мысалы, тізім, мәтіндік алаң, кестелер, батырма немесе тіпті кіріктірме web–браузер) жасау үшін қолданылатын көріністердің (Views) бай және кеңейтілетін тізімі;

– бір қосымшалар басқа қосымшалар өз жұмысына пайдалану үшін ашқан мәліметтерді басқарушы контент – провайдерлер (Content Providers);

– функционалсыз ресурстарға (жол мәліметтерге, кесте, файл және т.б.) қол жетімділікті қамтамасыз ететін ресурс менеджер (Resource Manager);

– әрбір қосымшаға өз мәлімдемелерін қалып–күй жолында бейнелеуге мүмкіндік беретін хабарламалар менеджері (Notification Manager);

– қосымшалардың өмірлік циклын басқаратын, әрекеттермен жұмысы туралы мәліметтерді сақтайтын, сонымен бірге әрекеттер арасында навигация жүйесін қамтамасыз ететін әрекеттер менеджері (Activity Manager);

– қосымшаларға құрылғының ағымдағы географиялық орын туралы жаңартылған мәліметтерді мерзімді алуға мүмкіндік беретін орналасу менеджері (Location Manager). Қорыта келгенде, Application Framework арқасында Android қосымшасы қосалқы функционалды қолдану мүмкіндігіне ие. Яғни қосымшалар және операциялық жүйе құрамдас бөліктерін бірнеше рет пайдалану қағидасы жүзеге асырылған.Android программалық стегінің жоғарғы қабатында қосымшалар деңгейі (Applications) орналасқан. Мұнда Android ОЖ- не алдын ала орнатылған қосымшалардың негізгі жиыны жатады.Мысалы, оған SMS жіберу қосымшасы, карталар, күнтізбе, браузер, пошталық клиент және т.б. бағдарламалар кіреді.Интегралданған қосымшалардың тізімі құрылғы түріне және Android нұсқасына байланысты өзгеруі мүмкін.Және бұл қосымшалар деңгейіне негізгі жиыннан басқа Android қосымшаларының бәрі жатады.

Мобильді құрылғыларға арналған Android операциялық жүйесінің алғашқы шығарылымынан біраз уақыттан кейін, артынша танымалдылығының айтарлықтай артуынан кейін жасаушыларға арналған құрал–жабдықтар шығарылды.

Android SDK – Android операциялық жүйесі үшін қосымшаларды жасау ортасы. Бағдарлама мобильді құрылғы камерасын, акселерометр, компас, GPS мәліметтерді, Bluetooth, Wi–Fi, EDGE және 3G бойынша рұқсатты қолданатын Android–қосымшаларын жасау мен тестілеуге мүмкіндік береді. Оған қоса мультимедиялық контентпен (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG және GIF форматтарындағы аудио, бейнелер, суреттер), SQLite мәліметтер базасымен, WebKit қозғалтқышындағы біріктірілген браузермен, Dalvik виртуалды машинасымен, GSM телефониямен және тағы басқаларымен жұмыс жасауды қолдайды. Оған қоса Android SDK қолданушылары қосымша орнатылған эмулятор көмегімен өздері жасап шығарған қосымшаларды тестілей алады.

1.5 Android мобильді жүйесінің архитектурасы

Мобильді қосымшаның дұрыс архитектурасын құру бүкіл жоба үшін аса маңызды. Код сапасына, жүйенің ауқымдылығына, сервердің API-ін және сенімді технологияларды таңдауға тиісті көңіл бөле отырып, болашақта көптеген проблемаларды болдырмауға болады.

Программистер салған архитектура қосымшаларды тұрақты қолдау мен "жөндеу" қажет пе, әлде бәрі қажет пе (және жұмыс істеуді жалғастыру) қалай жұмыс істейтініне тікелей әсер етеді. Мұнда алып келетін басты практикалық кеңестердің бірі-уақытты тиімді жоспарлау.

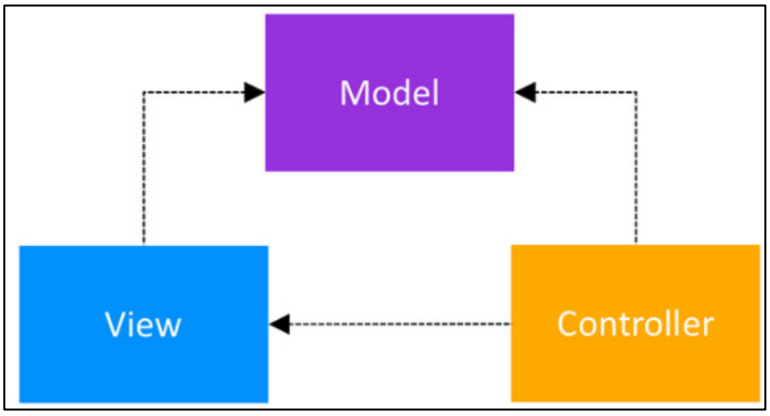
Model-View-Controller (MVC) үлгісі

Бұл үлгі ерте кезде алынған Android-қосымшалар архитектурасының алғашқы итерациясы болып табылады. Бұл код 3 түрлі қабатқа бөлінеді:

Model - деректер қабаты. Бизнес-логиканы өңдеуге және желімен және деректер базасымен өзара іс-қимыл жасауға жауап береді.

View - пайдаланушы интерфейсінің деңгейі (UI). Бұл модельден деректерді қарапайым визуализациялау.

Controller - логикалық деңгей, пайдаланушының мінез-құлқы туралы хабарлама алады және қажет болған жағдайда модельді жаңартады.



1.3-сурет – Мобильді қосымшаның MVC схемасы

Бұнда біз контроллер мен көрініс модельге байланысты екенін көреміз. Контроллер деректерді жаңартады. Ұсыным деректерді алады. Бірақ модель бөлінген және пайдаланушы интерфейсіне қарамастан сыналуы мүмкін.

Олардың бірі-әрекеттер мен фрагменттер контроллер ретінде әрекет еткенде. Олар деректерді өңдеу және көзқарастарды жаңарту үшін жауап береді. Бұл сәулет тәсілінің мәселесі-әрекеттер мен үзінділер тестілеу үшін өте үлкен және өте қиын болуы мүмкін.

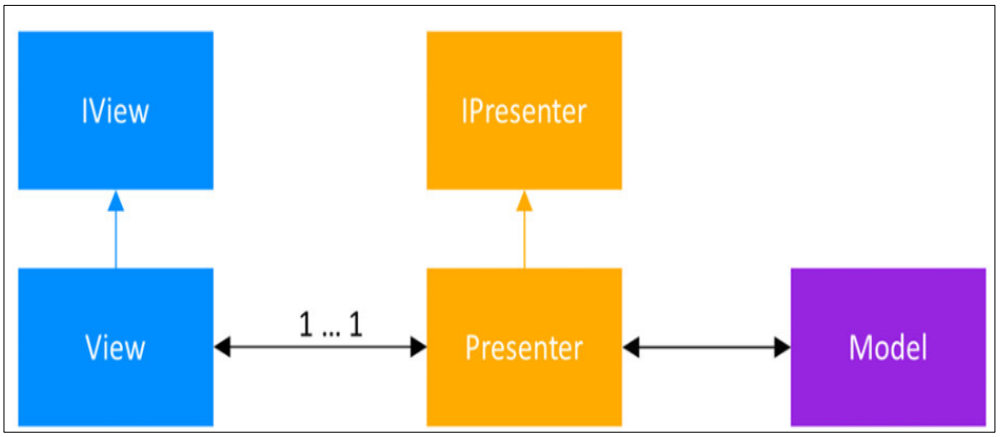
Логикалық (және дұрыс) көрінетін басқа тәсіл - бұл әрекеттер мен фрагменттер MVC әлеміндегі көріністер болуы тиіс жерде. Контроллерлер Android класын кеңейтпейтін немесе пайдаланбайтын жеке сыныптар болуы тиіс. Модельдермен бірдей.

Model View-Presenter (MVP) – үлгісі. Жұмыс істемеген бірінші тәсілден кейін Android әзірлеушілері ары қарай жылжып, ең танымал сәулет паттерлерінің бірі — MVP қолдануға тырысты. Бұл үлгі архитектураны таңдаудың екінші итерациясы болып табылады. Бұл үлгі кеңінен қолданыла бастады және әлі де ұсынылады. Android әзірлеуді бастағандар үшін, бұл білу оңай. Оның қарап көрейік 3 жеке қабаттар рөлдері :

Model – деректер қабаты. Бизнес-логиканы өңдеуге және желімен және деректер базасымен өзара іс-қимыл жасауға жауап береді.

View – пайдаланушы интерфейсінің деңгейі (UI). Бұл модельден деректерді қарапайым визуализациялау.

Presenter – модельден деректерді шығарып алады, пайдаланушы интерфейсінің логикасын қолданады және көріністің күйін басқарады, не көрсетуді шешеді және көріністен пайдаланушы хабарламаларына жауап береді. Бұл мәні бойынша MVC контроллері, ол мүлдем көріністе байланысты емес, тек интерфейс.



1.4-сурет – Мобильді қосымшаның Model View-Presenter (MVP) үлгісі

MVP схемасы көрініс пен жүргізуші тығыз байланысты екенін көрсетеді. Олар бір-біріне сілтеме жасауы керек. Олардың байланысы келісім-шарт интерфейсінің сыныбында анықталады.

View-ViewModel (MVVM) - model үлгісі. MVVM үлгісі үшінші қадам. Ол Android архитектурасының компоненттерін шығару арқылы Android командасы ұсынған архитектураның үлгісі болды. Міне, сондықтан біз осы модельді зерттеуге назар. Сондай-ақ, мен оны "My Crypto Coins"қолданбасы үшін пайдаланамын. Бұрынырақ сияқты, жеке код қабаттарына қарайық:

Модель-деректер көзі. ViewModel деректерді алу және сақтау үшін модельмен жұмыс істейді.

View-ViewModel пайдаланушылардың әрекеттері туралы хабарлайды.

ViewModel-көріністе қатысты деректер ағындарын ұсынады.

MVP үлгісімен салыстырғанда айырмашылық MVVM ViewModel-де презентатормен қалай орын алатындай көріністе сілтемелер жоқ. MVVM ViewModel әр түрлі көріністерді байланыстыра алатын оқиғалар ағынын ұсынады. Екінші жағынан, MVP жағдайында жүргізуші көрсету туралы ұсынысқа тікелей хабарлайды. MVVM сызбасын көрейік:

1.5-сурет- View-ViewModel (MVVM) үлгісі

1.6 Android Архитектурасының компоненттері

Егер Сіз Android бағдарламасының өмірлік циклімен таныс болсаңыз, сіз әдетте конфигурацияны өзгерту кезінде пайда болатын деректер ағыны мен сақтау және тұрақтылық бар барлық проблемаларды болдырмайтын қосымшаны жасау қандай бас ауыруы мүмкін екенін білесіз.

2017 жылы Android командасы біз жеткілікті күресті шешті. Олар өзіне жауапкершілік алып, Android архитектурасының компоненттерінің құрылымын ұсынды. Бұл, сайып келгенде, кодты қиындатпай немесе тіпті оған хактарды қолданбай барлық осы мәселелерді шешуге мүмкіндік береді.

Android Architecture Components-бұл сенімді, тексерілген және қолдаулы қосымшаларды жасауға көмектесетін кітапханалар жиынтығы. Қазіргі уақытта блогта осы пост жазу кезінде, ол келесі компоненттерден тұрады:

– деректерді байланыстыру-пайдаланушы интерфейсінің элементтеріне бақыланатын деректерді декларативті байланыстыру;

– өмірлік циклдар-белсенділік пен фрагменттердің өмірлік циклдарын басқару;

– LiveData-деректер базасы өзгергенде ұсынымдарды хабарлау;

– навигация-қолданбада шарлау үшін қажетті бәрін өңдеу;

– подкачка-деректер көзінен сұрау бойынша ақпаратты біртіндеп жүктеу;

– нөмір-SQLite деректер базасына еркін қол жеткізу;

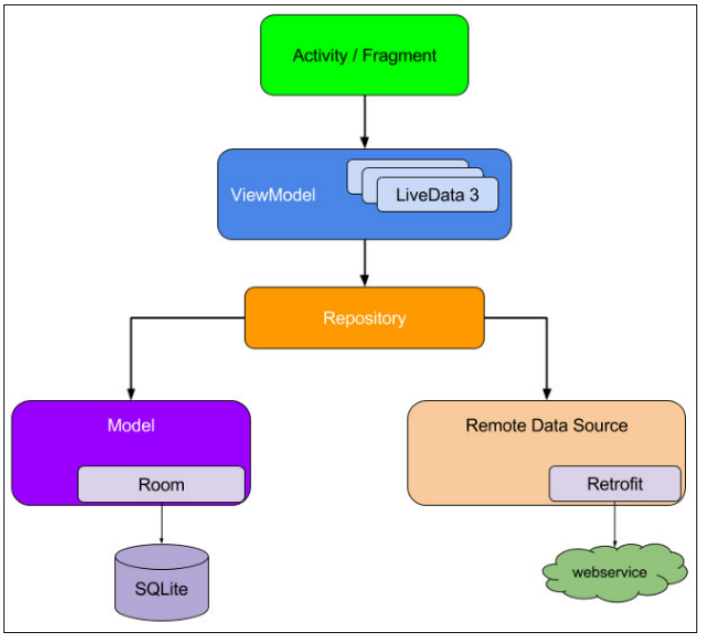
– ViewModel-өмірлік циклді ескере отырып, пайдаланушы интерфейсімен байланысты деректерді басқару;

– WorkManager-фондық тапсырмаларды басқару.

Біз осы диаграмма арқылы Crypto Coins қолданбасында MVVM архитектура үлгісін іске асырамыз.

Бұл ұсынылатын Google архитектурасы. Ол барлық модульдер бір-бірімен өзара әрекеттесуі тиіс екенін көрсетеді. Бұдан әрі біз жобамыздағы пайдаланылатын Android архитектурасының нақты компоненттерін ғана қамтитын боламыз.

Қосымшалардың архитектурасы бойынша нұсқаулық. Бұл нұсқаулық сенімді, сапалы қосымшаларды жасау үшін үздік тәжірибе мен ұсынылған архитектураны қамтиды.



1.6-сурет – Google архитектурасы

Бұл бет Android Framework базалық танысуды көздейді. Егер Сіз Android қолданбаларын әзірлеу үшін жаңа болсаңыз, жұмысты бастау және осы нұсқаулықта айтылған тұжырымдамалар туралы көбірек білу үшін біздің әзірлеушілер нұсқаулықтарымен танысамыз.

1.7 Мобильді қосымшаларды пайдаланушылар

Көп жағдайда қосымшалардың жұмыс үстелінен немесе іске қосу бағдарламасынан бірыңғай кіру нүктесі бар, содан кейін бірыңғай монолитті процесс ретінде іске қосылады. Екінші жағынан, Android бағдарламалары әлдеқайда күрделі құрылымы бар. Android бағдарламасына арналған типтік бағдарлама бірнеше компоненттерден тұрады, соның ішінде Әрекеттер, фрагменттер, қызметтер, контент-провайдерлер және кең тарату қабылдағыштары.

Қосымшаның осы компоненттерінің көпшілігі app Манифестінде жарияланады. Содан кейін Android OS осы файлды пайдаланып, сіздің қолданбаңызды құрылғының жалпы пайдаланушы интерфейсіне қалай біріктіруге болады. Android үшін дұрыс жазылған қолданба бірнеше компоненттерден тұрады және пайдаланушылар қысқа уақыт ішінде бірнеше қосымшалармен жиі өзара әрекеттеседі деп ескере отырып, қосымшалар әр түрлі пайдаланушы жұмыс процестері мен тапсырмаларына бейімделуі тиіс.

Мысалы, әлеуметтік желілер үшін сүйікті қолданбада фотосуреттерді бөліскен кезде

а) қолданба камера ниетін іске қосады. Содан кейін Android ОЖ сұрау өңдеу үшін Камера қолданбасын іске қосады. Қазіргі уақытта пайдаланушы

әлеуметтік желілерге арналған қосымшадан кетті, бірақ олардың тәжірибесі әлі жіксіз;

б) камера қолданбасы басқа ниеттерді тудыруы мүмкін, мысалы, тағы бір қолданбаны іске қосуға болатын файлды таңдауды іске қосу;

в) пайдаланушы әлеуметтік желі бағдарламасына оралып, фотосуреттермен бөліседі.

Кез келген уақытта процесс кезінде пайдаланушы телефон қоңырауымен немесе хабарламамен үзілуі мүмкін. Осы үзілуден кейін пайдаланушы осы фотосуреттермен алмасу процесіне қайта орала алады және оны жаңарта алады деп күтеді. Бұл жүктеу қолданбасының мінез-құлқы мобильді құрылғыларда таралған, сондықтан сіздің қолданба осы ағындарды дұрыс өңдеуге тиіс.

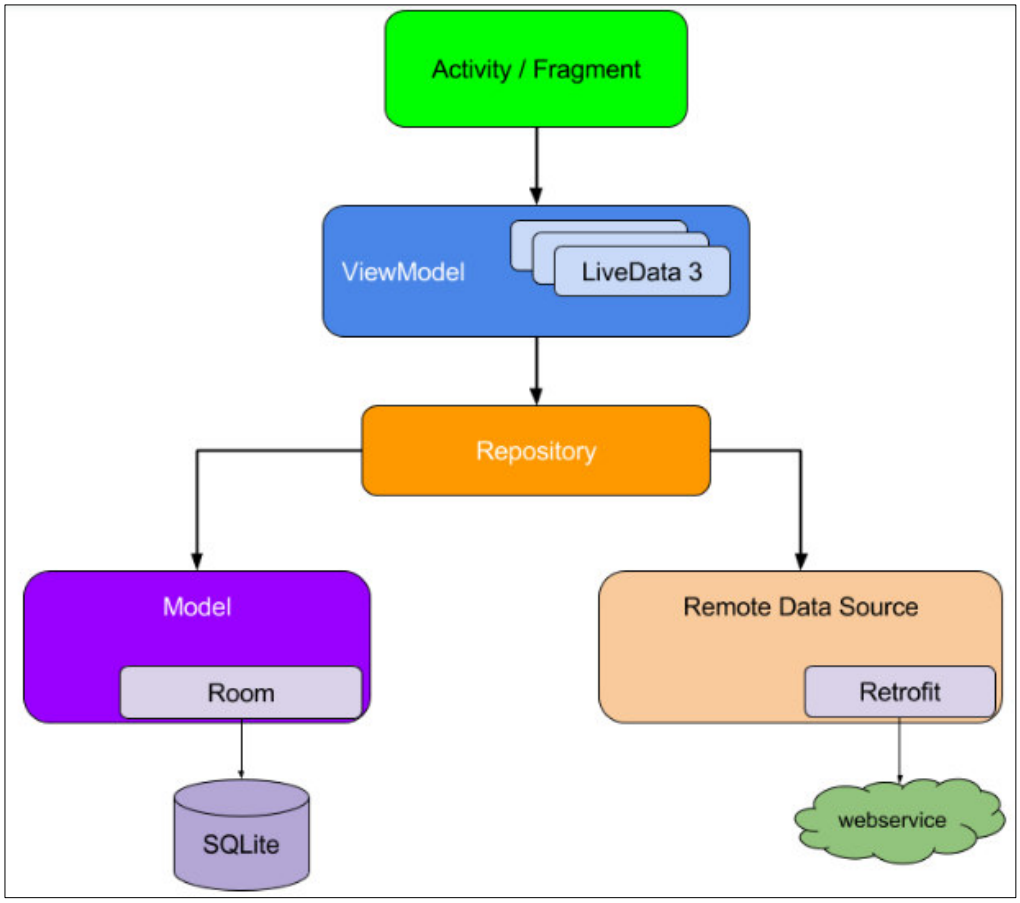
Ұялы құрылғылар ресурстарда да шектеулі екенін есте сақтаңыз, сондықтан кез келген уақытта Операциялық жүйе жаңа орын босату үшін кейбір қосымшалар процестерін өлтіруі мүмкін.

Бұл ортаның шарттарын ескере отырып, қолданба компоненттері жеке іске қосылуы мүмкін, ал Операциялық жүйе немесе пайдаланушы оларды кез келген уақытта жоя алады. Бұл оқиғалар сіздің бақылауыңызға жатпайтындықтан, бағдарлама құрамдастарында деректер немесе бағдарлама күйі сақталмауыңыз керек, ал бағдарлама құрамдастары бір-біріне байланысты болмауы керек.

Алдымен келесі диаграмманы қарастырайық, онда барлық модульдер қосымшаны әзірлегеннен кейін бір-бірімен өзара әрекеттесуі тиіс.

Әрбір компонент тек бір деңгейге төмен компонентке байланысты екенін ескеріңіз. Мысалы, әрекеттер мен фрагменттер тек көрініс моделіне байланысты. Репозиторий-бірнеше басқа сыныптарға тәуелді жалғыз класс; Бұл мысалда репозиторий деректердің тұрақты моделіне және ішкі деректер көзіне байланысты.

Бұл дизайн дәйекті және жағымды пайдаланушы интерфейсін жасайды. Соңғы жабылғаннан кейін бірнеше минуттан кейін немесе бірнеше күннен кейін пайдаланушы бағдарламаға қайта оралғанына қарамастан, ол қолданба жергілікті сақталатыны туралы пайдаланушының ақпаратын бірден көреді. Егер бұл деректер ескірген болса, бағдарлама репозиториясы модулі фондық режимде деректерді жаңартуды бастайды.



1.7-сурет – Компоненттер диаграммасы

2 Жобалау бөлімі

2.1 Унифицирленген модельдеу тілі. UML

Модельдеудің бірыңғайланған тілі (Unified Modeling Language — UML) – бұл бағдарламалық жүйелерді ерекшелендіру, бұрыштама қою, конструкциялау және құжаттамалау, сондай-ақ модельдер бизнесі мен өзге де бағдарламалық емес жүйелердің тілі болып табылады. UML бұдан бұрын да үлкен және күрделі жүйелерді модельдеу кезінде ойдағыдай қолданылып жүрген инженерлік әдіс-тәсілдердің бірлестігін көрсетеді.

UML-дің құрамдас бөлігі болып OCL табылады (Object Constraint Language — объектілерді шектеу тілі). UML-ды өңдеу 1994 жылғы қазан айында басталды, бұл кезде Rational Software Corporation-нан шыққан Гради Буч (Grady Booch) және Джим Рамбег (Jim Rumbaugh), OMT (Object Modeling Technique — объектілік модельдеу техникасы) әдістемесін бірыңғайландыру бойынша жұмыстарды бастаған болатын. 1995 жылғы қазан айында бірыңғайландыру әдісінің алдын-ала шамаланған болжамы ұсынылды. 1995 жылғы экономиялық құлдырау кезінде Иве Иакобсон (Ivar Jacobson) және оның Objectory компаниясы Rational-мен бірікті. Бірлесу қорытындысы болып OOSE (Object-Oriented Software Engineering) әдісімен бірыңғайландыру әдісінің қосылуы табылды.

Модельдеудің әмбебап тілін құру кезінде Гради Буч, Джим Рамбег және Иве Иакобсон өздеріне төмендегі мақсаттарды қойды:

− ОБ әдістемесін (тек қана БҚ ғана пайдаланбастан) пайдалана отырып, модельдеу жүйесін қамтамасыз ету;

− тілдің анық тұжырымдамасын жасау;

− күрделі жүйеде туындайтын көлем мәселесін шешу;

− адам ғана пайдаланып қоймайтын, сондай-ақ машина пайдалана алатын модельдеу тілін жасау [2].

UML кез келген жүйені модельдеу үшін жарамды: кәсіпорын масштабындағы ақпараттық жүйелерден таралған Web – қосымшаларға дейін және нақты уақыттың кіріктірілген жүйелері үшін де жарамды болады. Бұл өте мәнерлі тіл, ол жүйені жасау мен кейінгі жолды кеңінен ашып көрсетуге қатысы бар барлық көзқарас жағынан қарап шығуға мүмкіндік береді. Мәнерлі мүмкіндіктерінің молдығына қарамастан, бұл тіл түсіну және пайдалану үшін өте оңай. UML-ды зерттеуді оның концептуалды моделінен бастаған жөн, оның үш негізгі элементі болады: базалық құрылыс блоктары, осы блоктардың өзара үйлесімділігін анықтаушы ережелер және тілдің кейбір жалпы механизмдері.

Өзінің артықшылықтарына қарамастан UML — бұл тек тіл ғана; ол тек бағдарламалық қамтамасыз ету процесін құраушылардың бірі ғана. UML модельденетін нақтылыққа байланысты болмаса да, модельдеу процесі итерактивті және қадамдық болған жағдайда қолданған ыңғайлы. Ал жүйенің өзінің анық көрсетілген архитектурасы бар [3].

UML – бұл көрнекілеу, спецификациялау, құрастыру және бағдарламалық жүйелердің артефактілерін құжаттау тілі.

Тіл, өзінің құрамындағы сөздерді құрастыруға және мағыналы құрылымдар алуға мүмкіндігін беретін, сөздік пен ережелерден тұрады. Модельдеу тілінде сөздік пен ережелер жүйелердің концептуалды және физикалық түсінігіне бағдарланады. UML модельдеу тілі бағдарламалық қамтамасыз ету «сызбасын» құрастыру үшін стандартты құрал болып табылады.

UML тілінің сөздігі мен ережесі жақсы анықталған модельді қалай құрып және қалай оқуды түсіндіреді, бірақ та қандай жағдайда қандай модельдерді құру керектігі туралы хабарламайды.

Бағдарламашылардың көпшілігінің көзқарасы бойынша жобаны жүзеге асырудағы ойлану ол үшін код жазуға эквивалентті болып келеді. Сіз ойлайсыз – сіз код жасап жатырмын деп. Шын мәнісінде, кейбір заттар бағдарламалаудың қандай да тіліндегі тікелей кодтауда жақсы көрсетіледі, өйткені бағдарлама мәтіні – бұл алгоритмдер мен өрнектерді жазу үшін ең қарапайым және қысқа жол.

Бірақ та осындай жағдайда бағдарламашы формалды болмаса да модельдеумен айналысады. Ол идеяларын тақтаға немесе қағазға жаза алады делік. Алайда мұндай тәсіл жақсылық әкелмейді. Біріншіден, концептуальды модельге байланысты пікір алмасу пікірталасқа қатысушылардың барлығы бір тілде ғана сөйлеген кезде ғана мүмкін болады. Компаниялар жобаны жасау кезінде өзінің тілін жасауына тура келеді, оны жаңадан келген адамның түсінуі оңай емес, әрине. Екіншіден, бағдарламалаудың мәтіндік тілінің шегінен шығатын модельсіз бағдарламалық жүйелердің белгілі бір аспектілері туралы түсінік алуға болмайды. Өйткені, кластардың иерархиясында олардың неге арналғанын, әрине, әрбір кластың кодын мұқият үйренген кезде түсінуге болады, бірақ бүкіл құрылымды бірден және тұтас қабылдау мүмкін емес. Үшіншіден, егер автор өз ойындағы моделін айқын түрде көрсетпесе, онда ол жұмыстан ауысқанда оның ойы да жоғалады [4].

UML-ды пайдалану үшінші мәселені шешуге мүмкіндік береді: бұл модель тілдесуді оңайлатады.

Жүйенің кейбір ерекшеліктерін бәрінен бұрын мәтін түрінде модельдеуге болады, екінші біреулерін – графикалық модельдеуге болады. Шын мәнісінде барлық қызықты жүйелерде бір ғана бағдарламалау тілінің көмегімен беру мүмкін емес құрылымдар бар болады. Мұндайда, UML – екінші белгіленген мәселелерді шешуге мүмкіндік беретін графикалық тіл.

UML – бұл жай ғана графикалық символдар жиыны емес. Олардың әрқайсысының артында жақсы анықталған семантика тұр. Бұл бір дайындаушы жазған модельді екінші біреу бірмәнді түсіндіруі мүмкін, тіпті аспаптық бағдарламамен интерпретациялануы мүмкін. UML талдау, жобалау және жүзеге асыруға қатысты барлық мәнді шешімдердің өзгешіліктерін анықтауға мүмкіндік береді, ол бағдарламалық қамтамасыз ету жүйесін жасау мен кеңінен тарату процесінде де қабылданылуы тиіс.

UML-дың қолдану саласы бағдарламалық қамтамасыз етуді модельдеумен шектелмейді. Мәселен, оның мәнерлігі заң жүйелеріндегі құжаттар айналымын модельдеуге мүмкіндік береді және ауруханаларда науқастарға қызмет көрсету жүйесінің жұмыс істеуі мен құрылымын модельдеуді, аппараттық құралдарды жобалауды жүзеге асырады.

Сонымен, UML жасауда мынадай басты мақсаттарды атауға болады:

− пайдаланушыларға мағыналы модельдерді жасауға және олармен байланысуға мүмкіндік беретін, модельдеуді көрсетудің мәнерлі тілін қолдануға дайын етіп ұсыну;

− базалық тұжырымдаманы кеңейту үшін кеңейту және мамандандыру механизмдерін қарастыру;

− бағдарламаландырудың нақты тілдерінен және жасау процестерінен тәуелсіз болуын қамтамасыз ету;

− модельдендірудің осы тілін түсіну үшін формальды негізін қамтамасыз ету [5].

Қорыта келе, *UML*-дың мынадай мүмкіндіктерін атауғаболады.

− UML объектілі-бағдарланған болып келеді, соның нәтижесінде талдау және жобалау нәтижелерін сипаттау әдістері семантикалық жағынан қазіргі заманғы ОО-тілдерде бағдарламалау әдістеріне жақын болады;

− UML жүйе сипатының түрлі аспектілері мен барлық мүмкін болатын іс жүзіндегі көзқарастарының жүйесін сипаттауға мүмкіндік береді;

− UML диаграммасы молельді оқу үшін оның синтаксисімен жедел танысу үшін өте қарапайым болып келеді;

− UML дербес мәтіндік және графикалық стереотиптерді кеңейтуге және ендіруге мүмкіндік береді, бұл оның тек бағдарламалық инженерияда ғана емес басқа да қолданысына ықпал етеді;

− UML кеңінен таралып және жедел қарқынмен дамып келеді.

Бучтің, Рамбег және Иакобсонның әрекет жасауы 1996 жылдың қазан айында UML болжамында құжаттарды жасаумен аяқталды.1996 жылдың ішінде Rational бірлесіп қызмет атқарушылардың UML консорциумын құрады. Консорциум DEC, HP, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI және Unisys-тан тұрады. Олардың ынтымақтастығының нәтижесі 1997 жылдың қаңтар айында 1.0 болжамындағы UML ерекшелігін құру болып табылады.

1997 жылдың қаңтар айында бірлесіп қызмет атқарушыларға Object Time, Platinum Technology, Ptech, Taskon&Reich; Technologies, Softeam қосылып, 1997 жылдың 1 қыркүйегінде 1.1 болжамындағы UML ерекшелігі жарияланды.

Ақпараттандыру жүйесін дамытуда 1990шы жылдар объектілік технологиялардың қалыптаса бастауы кезеңі болып танылды. Объектілік БҚБЖ нарығы қалыптасып, белсенді дами бастады. Деректер базасының жүйесін бағдарламалық қамтамасыз ету нарығында басымдылықпен келе жатқан Oracle, Informix және IBM компанияларының деректер базаларының объектілік-реляциялық серверлерінің заманауи үлгілерінің шығуына

байланысты, 1996-1997 жылдары реляциялық ортадан объектілік ортаға көшіп-қонуының процесі өтеді.

Аталған процесстер, өз кезегінде талдаудың объектілік технологияларын және жүйелерді жобалаудың дамуын ынталандырды. Коммерциялық бағдарламалық өнімдерді іске асыратын түрлі әдістердің айтарлықтай саны пайда болды.

Көптеген компаниялар үшін бағдарламалық қамтамасыз етудің стратегиялық маңызы өсіп отырғандығына байланысты, индустрия бағдарламалық қамтамасыз етудің өндірісін автоматтандыру әдісін, оның сапасын көтеру, сондай-ақ оны нарыққа шығарудағы құны мен шығару уақытын төмендету әдістерін іздестіру үстінде.Бұл әдістер құрауыш технологияларға, көрсетушілік бағдарламаландыру, үлгілерді (pattern) және инструменттік (құрал-жабдық) ортаны (framework) пайдалануға негізделеді.

UML – бұл бағдарламалық жүйелердің артефактілерін көрсету, спецификациялау, конструкциялау және құжаттандыру тілі.

UML – бірыңғай модельдендіру тілі. Оны құруға индустрияның қандай да бір дәрежесінде болмасын барлық салалары қатысты болғандықтан, "UML - бұл бағдарламалық жүйелердің артефактілерін көрсету, спецификациялау, конструкциялау және құжаттандыру, сондай-ақ бизнес-процесстердің және бағдарламалық емес жүйелердің тілі. UML «міндеттерінің» тізбесіне басты назарды аудара кетейік. Спецификацциялау, көрсету, конструкциялау және құжаттандыру - бұлардың барлығы да жоғары деңгейлі жобалауға тікелей қатысты болып отыр. Ал жоғары деңгейлі «құрал-саймандар» қолданудың бір-бірегей аспектісі болуы мүмкін емес, сондықтан, UML анықтамасын келесідегі көпаспектілі интерпретациямен толықтыруға болады:

− UML әдіс есебінде жүйелердің қозғалысын тану үшін пайдаланылады;

− UML тіл есебінде пәндік аймақ туралы білімді «есептен шығару» үшін пай даланылады;

− UML модельдендіру тілі ретінде жүйелердің байланысу заңдылықтарын түсіну үшін (және мүмкін формальдауда) пайдаланылады;

− UML бірыңғайландыру түрі ретінде құрастырушылардың қызметін үйлестіру үшін пайдаланылады.

Көптеген программистердің көзқарасы көрсетіп отырғандай, жобаны іске асыру мәселесі бойынша ой-толғаулар шамамен оған код жазу үшін балама болып отыр. Кейбір заттар бағдарламаландырудың қандай да бір болмасын кодында тікелей алғанда өте жақсы мәнерленеді, себебі программаның мәтіні – бұл алгоритмдерді және өрнектерді жазу үшін өте қысқа және қарапайым жол.

Бірақ мұндай жағдайлардың өзінде программист бейресми болса да мод ельдендірумен айналысады. Ол өз ойының жобасын тақтада немесе майлықта жазды делік. Бірақ мұндай жақындасу жағымсыз лаң туғызуы мүмкін. Біріншіден, концепциялық модельге қатысты сылтаулар бойынша ой алмасуларға қатысатын пікірталастың қатысушылары бір тілде сөйлесетін

кезде ғана мүмкін болады. Жобаларда жасау кезінде компаниялар сөздерінің тілін жаңалық есебінде ашуы тиіс болатындығы ереже сияқты болып кеткен.

Бағдарламалық құралдарды шығаратын компания, орындалатын кодпен қатар басқа да артефактілерді шығарады, соның ішінде келесідегілерді:

- жүйеге қойылатын талаптар;
- архитектура;
- жоба;
- бастапқы код;
- жобалау жоспарлары;
- тесттер;
- прототип;
- болжам нұсқамалары, және т.б.

Өңдеу жөнінде қабылданған әдістемеге байланыссыз бір жұмысты орындау басқаларға қарағанда ресмиірек жүргізіледі. Айтылған артефактілер - бұл жобаның құрамалы бөлігінің жеткізушілері ғана емес, олар басқару үшін, нәтижені бағалау үшін, сондай-ақ жүйені жасау және оған өрбіту жасап болған соңғы уақытта ұжым мүшелері арасындағы қарым-қатынас құралы есебінде қажет.

UML жүйелік архитектураны және барлық оның бөлшектерін құжаттандыру жөніндегі проблеманы шешуге мүмкіндік береді, Жүйеге қойылатын талаптарды тұжырымдау және тесттерді анықтау үшін тілді ұсынады, және соңында жобаны жоспарлау және болжамдарды басқару кезеңінде жұмыстарды модельдендіру үшін құралдарды ұсынады.

UML тілі біріншіден бағдарламалық жүйелерді жасауға арналған. Оны пайдалану келесідегі салаларда әсіресе тиімді болмақ:

- кәсіпорын көлеміндегі ақпараттық жүйеде;
- банктік және қаржы қызметтерінде;
- телекоммуникацияда;
- транспортта;
- қорғаныс өнеркәсібінде, авиацияда және космонавтикада;
- бөлшектеп сату саудасында;
- медициналық электроникада;
- ғылымда;
- үлестірілген Web-жүйелерде.

UML қолдану аясы бағдарламалық қамтамасыз етуді модельдендірумен шектелмейді. Оның мәнерлілігі заң жүйесіндегі құжаттандыру айналымын, ауруханалардағы емделушілерге қызмет көрсету жүйесінің құрылымын және қызмет көрсетуін, аппарат құралдарына жобалау жасауды модельдендіруге мүмкіндік береді.

UML жасауда басты болып келесідегі мақсаттар танылған:

- пайдаланушыларға мағыналы модельдерді жасауға және олармен алмасуға мүмкіндік беретін, модельдендіруді көрсетушіліктің мәнерлі тілін қолдануға дайын етіп ұсыну;

− базалық тұжырымдаманы кеңейту үшін кеңейту және мамандандыру механизмдерін қарастыру;

− бағдарламаландырудың нақты тілдерінен және жасау процестерінен тәуелсіз болуын қамтамасыз ету;

− модельдендірудің осы тілін түсіну үшін формальды негізін қамтамасыз ету.

UML патенттелген құрал болып табылмайды және барлығы үшін ашық. Ол өзі негізделіп жасалған әдістерді пайдаланудың тәжірибе жүзінде расталған ғылыми қоғамдастықтары мен тұтынушыларының қажеттілігін қанағаттандыру үшін тағайындалған. Методология бойынша көптеген мамандар, ұйымдар және инструменталды құрал-жабдықтарды тасымалдаушылар бұл тілді пайдаланамыз деп міндеттеме алған. UML Бучтың, OMT, OOSE әдістемелерінде және басқа да озық әдістерінде пайдаланылуы сияқты, сондай-ақ UML және кең қауымдастық серіктестерінің ұсыныстарын енгізетіндіктен, семантиканың және нотацияның негізінде құрылғандықтан бұл тілді мойындап, қадірлеу кең, жаратынды түрде болуы тиіс.

UML атауындағы «бірыңғайлау» эпитетінің екі аспектісі бар. Біріншіден, ол модельдендірудің ертеректегі әдістерінің тілдері арасындағы маңызды емес көптеген ерекшеліктерді іс жүзінде жоққа шығарады. Екіншіден, мүмкін аса ерекше шығар, ол жүйелердің көптеген түрлі түрлерінің болашағын, жасау фазаларын (талаптарды талдау, жобалау және іске асыру) және ішкі тұжырымдарды бірыңғайластырады (олармен байланысты бағдарламалық қамтамасыз етуді емес, бизнесті).

Дегенмен, UML нақты тілді анықтаса да, бұл модельдендірудің тұжырымдарын болашақта жетілдіру үшін тосқауыл бола алмайды. UML жасау кезінде көптеген озық әдістер назарға алынған болатын, бірақ UML келешектегі болжамына өзге де әдістер өз ықпалын тигізетін болады.

Сонымен қатар, UML негізінде жаңа перспективалық (болашақ) әдістер анықталуы мүмкін. UML оның ұйытқысын қайта анықтамастан ак кеңейтілуі мүмкін.

UML оның ағымдағы түрінде көптеген инструменталды құрал-жабдықтардың, соның ішінде көрсетушілік модельдендірудің, ұқсатқыштық модельдендірудің, сондай-ақ құру ортасының негізі болып табылады.

Объектілік технологияларды дамытуда 1989 жылы құрылған Object Management Group (OMG) консорциумының құрылуы басты звено болып отыр, оның мақсаты - интероперальды біртекті емес бөлінген объектілік орталарды құру үшін индустриалды стандарттарды жасау болып табылады. OMG қабылдау, 1991 жылдан бастап, CORBA индустриалды стандартының және оның инфрақұрылымына байланысты стандарттардың бірқатар болжамдары, сондай-ақ тәжірибе жүзінде CORBA технологиясын белсенді түрде енгізу бұл стандарттардың объектілік талдау мен жобалау технологияларына, модельдендіру тілінің стандарттарын өндіру қажеттілігін сезінуге, CORBA архитектурасына сүйенетін жүйелерді құруға қолдау

көрсететін бұл тектес технологиялардың негізі болып табылатын, объектілік-бағдарланған инструменталды құрал-жабдықтардың технологиясын пайдаланатын интероперабельділікті қамтамасыз ететін және осы саладағы көптеген ұжымдар жинақтаған тәжірибелермен жинақталса да құрастырушылар мен пайдаланушылардың қызығушылығын тудыра алмады.

2.2 Прецеденттер диаграмасы

Прецеденттер-бұл талаптарды (әсіресе функционалдық) ұғыну мен тұжырымдаудың кеңінен қолданылатын механизмі. Олар жобаның көптеген аспектілеріне әсер етеді, соның ішінде ООА/П.

Прецеденттердің сипаттамасы-жүйенің не істеу керектігін көрсететін талаптарды ұғыну мен тұжырымдаудың тамаша әдісі. Өз кезегінде, талаптар-бұл прецеденттердің барлық жиынтығы, яғни жүйенің қызмет ету моделі және оның ортасы.

Прецеденттер-бұл барлық мүдделі тұлғалар үшін талаптарды тұжырымдау кезеңін жеңілдету механизмі. Негізінде, прецеденттерді сипаттау қиын емес, бірақ жүйеден не қажет екенін анықтау және оны қажетті деңгейде егжей-тегжейлі сипаттау жеткілікті қиын.

2.1-кесте – UML диаграмма түрлері

Құрылымдық диаграммалар	StructureDiagrams: *(Ағылшын тілінде)*	Структурные диаграммы: (*Орыс тілінде)*
Кластар диаграммасы	Class diagram	Диаграмма классов
Компоненттер диаграммасы	Componentdiagram	Диаграмма компонентов
Композит/ құрама құрылымдар: Кооперация диаграммасы (UML2.0) немесе келісімді үйлестіруші диаграмма	Composite structure diagram: Collaboration (UML2.0)	Композитной/составной структуры: Диаграмма кооперации (UML2.0) или диаграмма сотрудничества
Таратылған диаграмма	Deploymentdiagram	Диаграмма развёртывания
Объектілер диаграммасы	Object diagram	Диаграмма объектов
Пакеттер диаграммасы	Package diagram	Диаграмма пакетов
Профильдер диаграммасы (UML2.2)	Profile diagram(UML2.2)	Диаграмма профилей(UML2.2)
Өзгеру диаграммалары	BehaviorDiagrams:	Диаграммы поведения:
Қызмет диаграммасы	Activity diagram	Диаграмма деятельности
Күй ауысу диаграммасы	State Machinediagram	Диаграмма состояний
Прецеденттер диаграммасы	Use case diagram	Диаграмма прецедентов (вар-тов)

2.1-кестенің жалғасы

Құрылымдық диаграммалар	StructureDiagrams: *(Ағылшын тілінде)*	Структурные диаграммы: (*Орыс тілінде)*
Өзара әрекеттесу диаграммасы: - Коммуникация диаграммасы (UML2.0) / кооперация диаграммасы (UML1.x) - Өзара әрекеттесуді шолу диаграммасы (UML2.0) - Реттілік диаграммасы - Синхронизация диаграммасы (UML2.0)	Interaction Diagrams: - Communication diagram (UML2.0) / Collaboration (UML1.x) - Interaction overview diagram (UML2.0) - Sequence diagram - Timing diagram(UML2.0)	Диаграммы взаимодействия: - Диаграмма коммуникации (UML2.0) / Диаграмма кооперации (UML1.x) - Диаграмма обзора взаимодействия (UML2.0) - Диаграмма последовательности - Диаграмма синхронизации (UML2.0)

Орындаушы (actor)- мінез-құлыққа ие мән. Исп. жүйеге қатысты сыртқы болып табылады, бірақ прецедент сипаттаған процеске қатысады. Мысалы, адамдар орындайтын рөлдер: клиенттің рөлі.

Орындаушылардың:

− прецедентпен байланыс (communication);

− жалпылау (generalization).

Сценарий (scenario) - бұл орындаушылар мен жүйе арасындағы әрекеттердің немесе өзара әрекеттесудің арнайы реті. Ол кейде прецеденттің данасы деп аталады. Негізінде, прецедент (use case) — Бұл өзара байланысты табысты және сәтсіз сценарийлер жиынтығы, бір тапсырманы шешу үшін Орындаушы жүйені пайдалануды сипаттайтын. Прецеденттерді сипаттау үшін прецеденттер диаграммасы (Use case diagram) қолданылады.

Кез келген (соның ішінде бағдарламалық) жүйелер өз жұмысы барысында адамдар қолданатын және/немесе басқа жүйелермен өзара іс-қимыл жасайтындарын ескере отырып жобаланады.

Жүйе өз жұмысының процесінде өзара әрекеттесетін мәндер векторлар деп аталады, әрі әрбір сектор жүйе өзін қатаң белгілі, болжалды түрде ұстанады деп күтеді.

Эктордың қатаң анықтамасын береміз. Бұл үшін UML Zicom Mentor бойынша керемет көрнекі сөздікті қолданамыз:

Актор (actor) - бұл прецеденттермен немесе мәндермен (жүйе, кіші жүйе немесе сынып) өзара іс-қимыл кезінде орындалатын логикалық байланысты рөлдердің жиынтығы. Акторлар болуы мүмкін:

− адам немесе басқа жүйе;

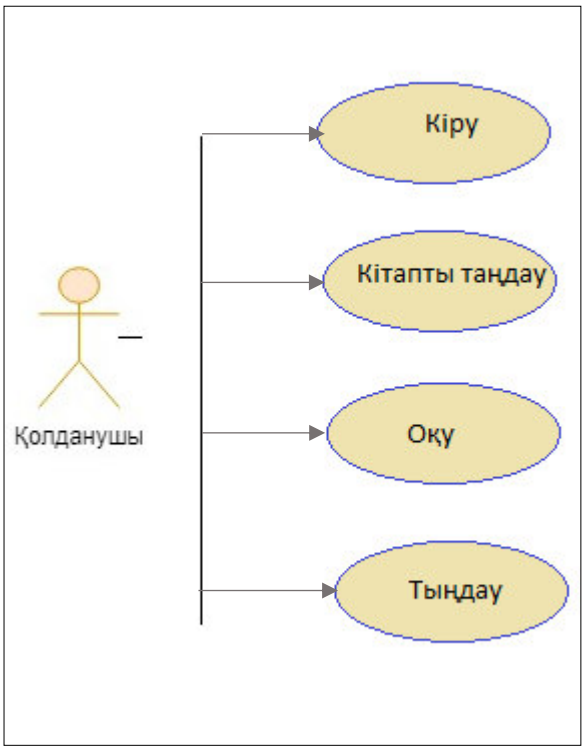
− кіші жүйе немесе сынып.

Экторлар-бұл нысаннан тыс нәрсе және онымен өзара іс - қимыл. Графикалық эктор, біз балалық шағында сурет салған сияқты, өз отбасы

мүшелерін бейнелейтін "адам" немесе суретте көрсетілгендей тиісті стереотипі бар сынып символымен бейнеленеді. Екі көрініс формасы бірдей мағынаға ие және диаграммаларда қолданылуы мүмкін. ""Стереотиптелген" форма жүйелік факторларды ұсыну үшін немесе эктордың қасиеттері бар және оларды көрсету қажет болған жағдайларда жиі қолданылады.

Мұқият оқырман бірден сұрақ қоя алады: неге актер емес, эктор? Келісемін, "эктор" сөзі орыс адамның есту қабілетін сәл кеседі. Біз дәл осылай айтудың себебі қарапайым-эктор сөзі action, бұл аудармада әрекет білдіреді. "Эктор" сөзінің сөзбе - сөз аудармасы - әрекет етуші тұлға-пайдалану үшін тым ұзақ және ыңғайсыз. Сондықтан да біз бұдан әрі де осылай айтатын боламыз.

Төменде жобаның прецеденттер диаграммасы көрсетілген:



2.1-сурет – Жобаның прецеденттер диаграммасы

2.3 Тізбек диаграммасы

Тізбектіліктің диаграммасы (sequence diagram) – кезектілік елестету тәсілі, объектілердің әрекеттестік операцияларының уақыттары-да, сценариймен қарастырылғандай, тізбек хабарлардың объектілер функцияларының жүзеге асыру барысында ауысады.

Күйдің және қызметтің диаграммасының қарайтын болсақ, осы диаграммалар және жүйе үшін тәртібінің серпінділікінің мамандама үшін пайдаланылады, уақыт ара ашық көрініс ала оған қатысты емес. Тәртіптің уақыттың аспектісі байыпты мағынаны ілеспе үдерістің модельдеуін білдіреді,

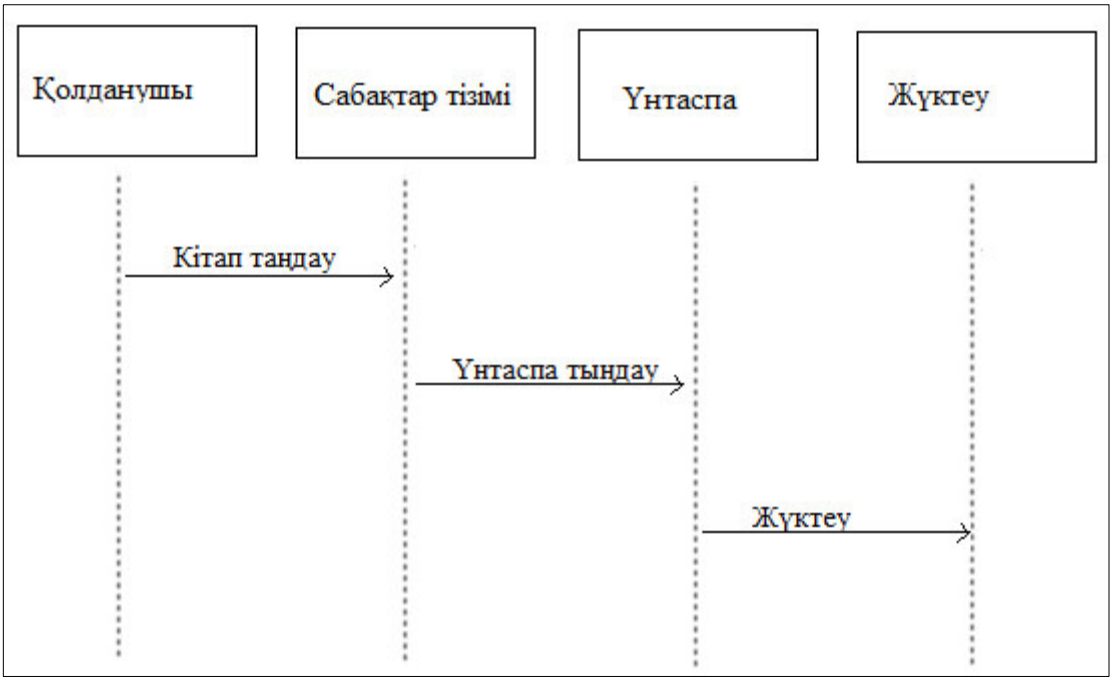
нысанның әрекеттестіктерін суреттейді. Нысанның әрекеттестігінің модельдеуі үшін UML тілінде тізбектіліктің диаграммалары пайдаланылады.

Тізбектіліктің диаграммасында ғана ауызекі әрекеттестікте қатысты нысандар бейнеленеді. Бұлақты кезбен тізбектіліктің диаграммалары үшін нысанның әрекеттестігінің серпінділігі уақытта болып табылады.

UML тізбектіліктің диаграммасында сияқты екі өлшем береді. Бірінші солдан оңға қарай түрде тік сызықтардың, бас-басы нешіншіден әрекеттестікте қатынасушының жеке нысанның өмірінің сызығын бейнелейді. Шеткі сол жақтағы диаграммада әрекеттестіктің айғайшысы болып табылатын нысан бейнеленеді. Оң сырттың ауызекі біріншімен әрекеттесу нысаны бейнеленеді. Ақырында, барлық нысандар тізбектіліктің диаграммасында біреудің кезектілік немесе нысанның белсенділігінің дәрежесімен әрекеттестікте дос доспен анықталу тәртібін тәрбиелейді.

Жазу-сызу бас-басы нысан тіктөртбұрышпен бейнеленеді және арадан бастап өмірдің өзінің сызығы орналасады. Тіктөртбұрыштың нысанның ішінде аты және сыныптың атының бөлігі қоснүктемен жазылады.

Пайдаланушы жасайтын бірінші әрекет-бұл жүйеге кіру, содан кейін кітаптар тізіміне кіру арқылы керекті кітапты оқи алады. Оқып біткеннен кейін үнтаспа тыңдауға болады. Оқу, тыңдау бөлімінен кейін кітапты сақтауға болады. Жүйеден шығу жолымен жүйедегі барлық әрекеттерді аяқтау. Төменде жобаның тізбек диаграммасы көрсетілген:



2.2-сурет – Жобаның тізбек диаграммасы

2.4 Класстар диаграммасы

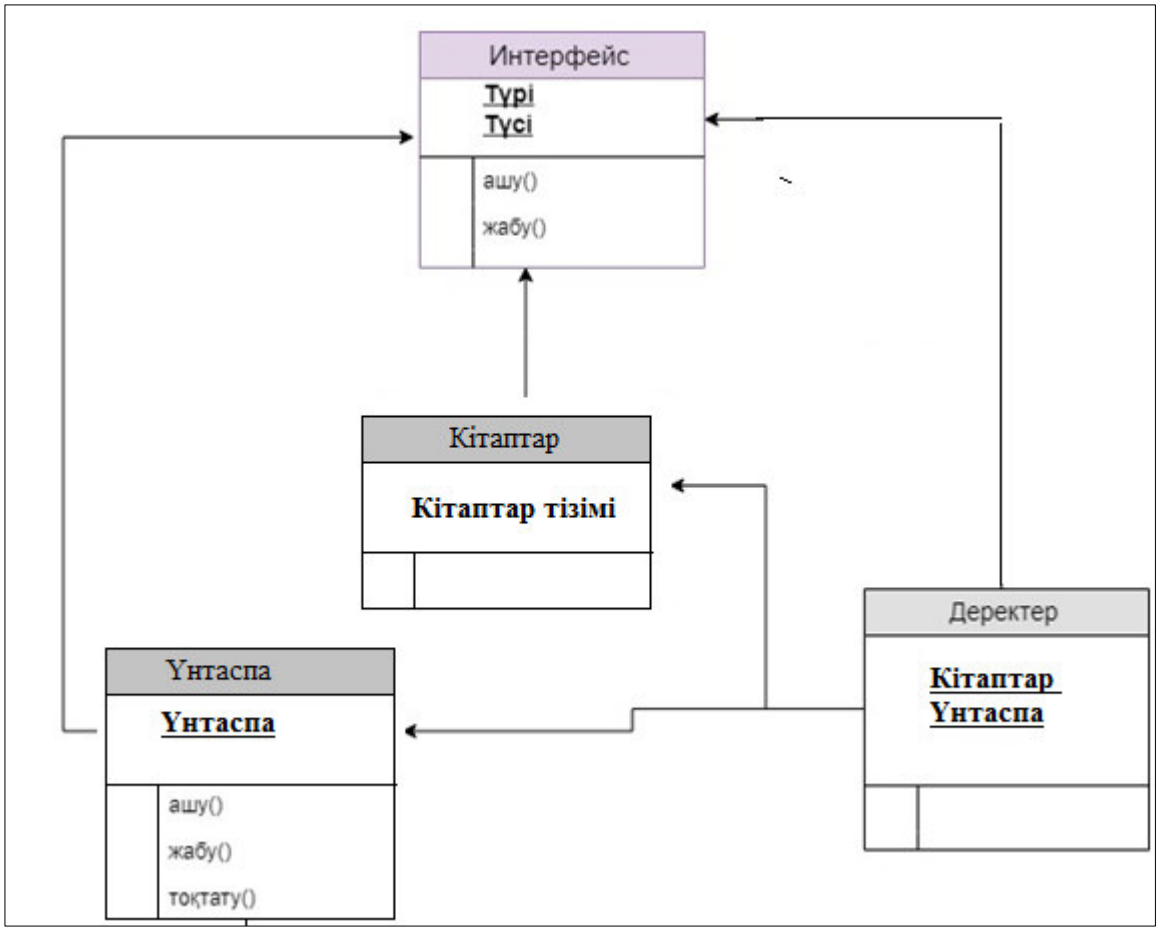
Класстар диаграммасы – жүйедегі класстардың статикалық құрылымын модельдеу үшін және класстар арасындағы байланысты кәрсету үшін жасалады. Класстар диаграммасы объектіге бағдарланған ұстанымдағы негізгі диаграмма болып табылады. Класстар диаграммасының қызметі: жүйедегі объектілердің типін анықтау және олардың арасындағы байланысты кәрсету болып келеді. Байланыстың статикалық екі түрі қолданылады: ассосация және подтиптер (тума типтер). Бұлардан басқа класстар диаграммасының элементтеріне атрибуттар, операциялар және объектілер арасындағы шектеулер жатады. Класстар диаграммасын жобалаудан бұрын, ол диаграмманың қандай мақсатта қолданылатынын анықтап алу керек.

Класстар диаграммасын жобалаушы үш түрлі мақсатта қолдануы мүмкін:

− концептуалдық аспект – мұнда класстар даграммасы зерттелетін пәндік облыстағы негізі ұғымдарды анықтайды. Бұл ұғымдар болашақта құрылатын класстарға сәйкес болу керек, бірақ іс жүзінде ол барлық уақытта бірдей орындалмайды. Сондықтан концептуалдық модель болашақ ақпараттық жүйемен әлсіз байланыста болады және ол программалау тіліне тәуелсіз болады;

− спецификациялық аспект – мұнда құрылатын диаграмма ақпараттық жүйенің (программалық жабдықтың) интерфейсі деңгейінде жасалады. Класстың өзінің ішкі құрылымы қарасытырылмайды;

− жүзеге асыру аспектісі (реализация) – мұнда класстар диаграммасы ақпараттық жүйеге (программалық жабдыққа) қатысатын класстарды ішкі құрылымдарымен қоса анықтайды. Бұл аспекті программистер үшін негізгі диаграмма болып табылады.



2.3-сурет – Жобаның класстар диаграммасы

2.5 Күй диаграммасы

Күй диаграммасы (state machine diagrams) – бұл жүйенің мінез-құлқын сипаттаудың белгілі технологиясы. Сол немесе басқа түрде күй диаграммасы 1960 жылдан бастап бар, және объектілі-бағытталған бағдарламалау кезінде олар жүйенің мінез-құлқын көрсету үшін қолданылған. Объектілі-бағытталған тәсілдерде сіз бір объектінің өмір бойы мінез-құлқын көрсету үшін жалғыз сыныптың күй диаграммасын сызамыз.

Жай-күй диаграммасы-бұл мәні бойынша стандартталған шартты белгілері бар автоматтар теориясынан күй диаграммасы[15] [16], ол компьютерлік бағдарламалардан бизнес-процестерге дейін көптеген жүйелерді анықтай алады. Мынадай шартты белгілер пайдаланылады:

Бастапқы жағдайды білдіретін шеңбер.

− соңғы жағдайды білдіретін (егер бар болса) шағын шеңбері бар шеңбер;

− жай-күйін көрсететін дөңгелектелген тіктөртбұрыш. Тіктөртбұрыштың ұшында күй атауы бар. Ортасында көлденең сызық болуы мүмкін, оның астында осы күйде болатын белсенділік жазылады;

− ауысуды білдіретін көрсеткі. Өту шақыратын оқиғаның атауы (егер бар болса) көрсеткінің жанында белгіленеді. Қорғау өрнегі "/" алдында қосылуы мүмкін және шаршы жақшаларға ("/"атауы[күзетіп_өтіру]) жасалуы мүмкін, бұл өрнек өтпелі орын алуы үшін шынайы болуы тиіс. Егер ауысу кезінде қандай да бір әрекет жүргізілсе, онда ол " / " кейін қосылады ([күзетілетіні_лау]/әрекет атауы);

− кіріс сызықтары көп немесе бір шығыс сызықтары бар немесе бір кіріс сызықтары және көптеген шығыс сызықтары бар қалың көлденең сызық. Бұл тиісінше біріктіру мен тармақтауды білдіреді.

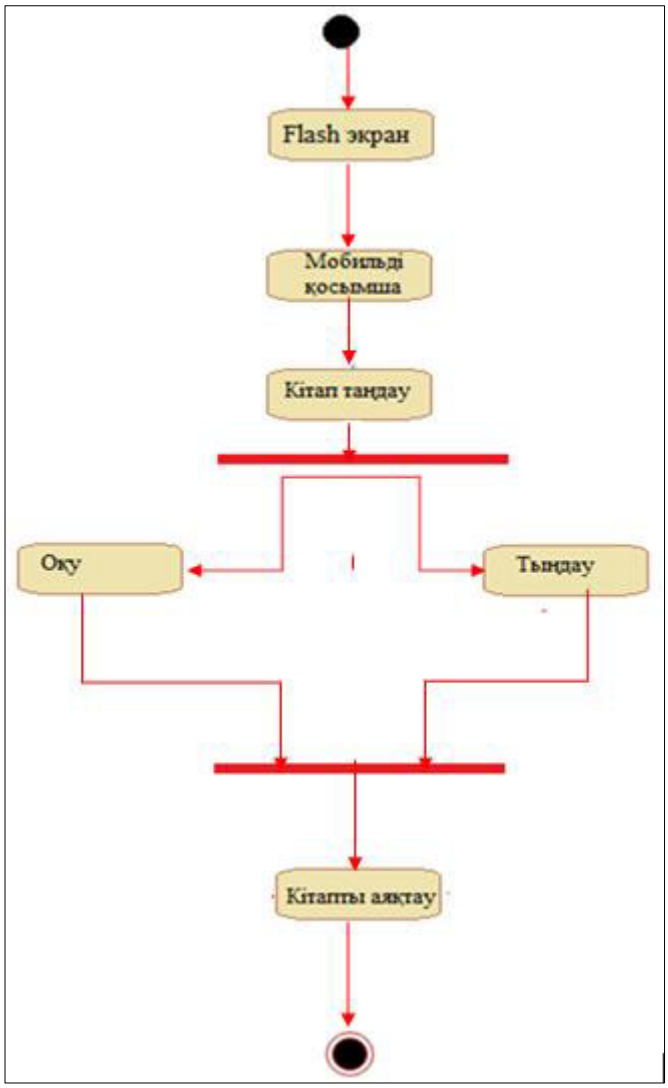
Объектілер мінез-құлқымен және жай-күйімен сипатталады. Мысалы, адам жаңа туған, сәби, бала, жасөспірім немесе ересек болуы мүмкін. Басқаша айтқанда, Нысандар бірдеңе жасайды және бірдеңе "біледі". Күй диаграммалары күрделі объектілердің қалай жұмыс істейтінін түсіндіру үшін қолданылады. "Күй" ұғымының мағынасы интуитивті түсінікті болғанымен, оның анықтамасын классиктер мен Zicom Mentor беретін түрде келтіреміз:

Жағдай (state)-белгілі бір қызметті орындайтын немесе қандай да бір оқиғаларды күтетін объектінің өмірлік циклiндегі жағдай. Объектінің жай-күйі оның кейбір атрибуттарының мәндерімен және басқа объектілермен байланыстардың болуымен немесе болмауымен анықталады.

Күй диаграммасы нысан бір күйден екінші күйге қалай ауысатынын көрсетеді. Әлбетте, күй диаграммалары жүйенің динамикалық аспектілерін модельдеу үшін қызмет етеді (тізбектер диаграммалары, кооперациялар, прецеденттер және, әрі қарай, қызмет диаграммалары сияқты). Жиі күй диаграммасы Автоматты көрсетеді деп естуге болады, бірақ бұл туралы кейінірек біраз сөйлесеміз. Күй диаграммасы объектінің өмірлік циклін моделдеу кезінде пайдалы (оның жеке түрі сияқты - біз одан әрі айтатын қызмет диаграммасы).

Басқа диаграммалардан жай - күй диаграммасы белгілі бір сыныптың бір данасының ғана жай-күйінің өзгеру процесін сипатталуымен ерекшеленеді, бұл ретте реактивті объектінің, яғни оның мінез-құлқы сыртқы оқиғаларға реакциясымен сипатталатын объектінің. Өмірлік цикл ұғымы реактивті объектілерге қолданылады, олардың қазіргі жағдайы (және мінез-құлқы) олардың бұрынғы жай-күйіне байланысты. Бірақ күй диаграммалары жеке нысанның динамикасын сипаттау үшін ғана емес, маңызды. Олар тікелей және кері жобалау арқылы орындалатын жүйелерді құрастыру үшін пайдаланылуы мүмкін. Және олар шын мәнінде осындай сапада қолданылады, UNIMOD, FLORA және т. б. сияқты қолданыстағы "UML орындалатын" нұсқаларын еске түсіреміз. Бірақ күй диаграммаларындағы белгілер туралы сөйлесейік. Дөңгелектелген тіктөртбұрыштар объектінің өмірлік циклі ішінде өтетін жағдайларды білдіреді. Көрсеткілермен объектінің диаграммасында сипатталатын әдістердің орындалуынан туындаған жағдайлар арасындағы өтпелер көрсетіледі. Сондай-ақ жалған күйлердің екі түрі бар: бастапқы, онда объект құрылғаннан кейін бірден орналасқан (тұтас шеңбермен белгіленеді) және егер оған көшкен болса, объектіні тастап кете алмайтын соңғы

(шеңбермен қоршалған шеңбермен белгіленеді). Төменде осы жобаның күй диаграммасы келтірілген:



2.4-сурет – Жобаның күй диаграммасы

2.6 Кооперация диаграммасы

Кооперация диаграммасы (Collaboration diagram) – хабарламаны қабылдайтын және жіберетін негізгі назары объектің құрылымдық ұйымдастырылуына жіберілетін өзара әрекет ету диаграмасы. Кооперация бұл кластар, интерфейстер және басқа да элементтер қоғамдастығы, олар кооперативтік мінез-құлықты қамтамасыз ету үшін бірлесіп жұмыс істейді.

Кооперация диаграмасында немесе кооперативті диаграмада объект экземплярлары пиктограмма түрінде көрсетілген. Тізбектелу диаграмасында сияқты, мұнда тілшелер алмасуы қолданудың бір нұсқасы асында болатын хабарламаны білдіреді. Олардың уақытша тізбектелуі, алайда хабарламаларды нөмірею жолымен белгіленеді.

Хабарламарды нөмірлеу тізбектеудіжоғарыдан төменге сызықтардың орналасу жағдайына қарағанда қабылдауды күрделі етеді. Басқа жағынан, бұндай кеңістіктік қабылдау кебір басқа моенттерді оңай беунелеуге мүмкіндік береді, мысалы объектілер, компоненттерді жабатын немесе басқа ақпаратты көрсетуге болады.

UML-да нөмірлеудің оңдық жүйесі қолданылады, өткені осы жағдайда қандай операция қандай процедураны шақаратыны түсінікті, алайда олардың жалпы түзбектелуін көру қиынғы соғады. Қолданылатын нөмірлеу жүйесіне тәуелсіз диаграмада осындай тізбектелу диаграмасы сияқты басқарушы ақпаратты орналастыруға болады.

Кооперация диаграммасының құрылымдық элементтері:

− класстар;
− нысандар;
− байланыс;
− хабарламалар;
− рөлдер.

Сонымен, біз тізбектер диаграммалары сияқты кооперация диаграммалары модельделетін жүйенің динамикалық аспектілерін сипаттау үшін арналғанын айттық. Әдетте олар үшін қолданылады:

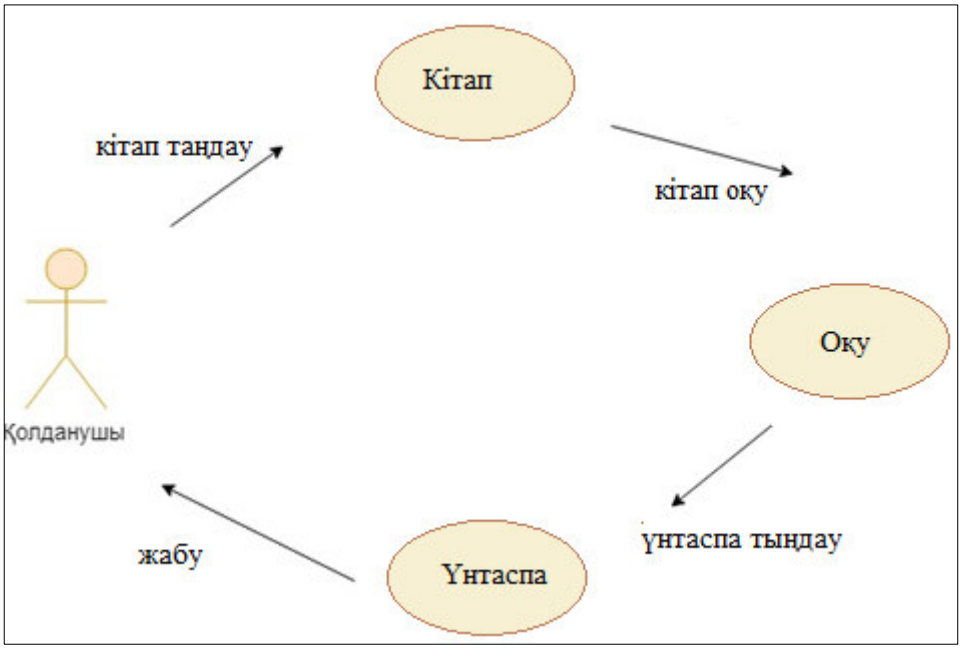
− кітап оқу ұзақтығынан "нақты ортадағы өзара іс-қимыл жасайтын нысандар жиынтығын көрсету";

− жүйенің динамикалық аспектілерін зерттеу нәтижелеріне негізделе отырып, сыныптар арасында функционалдылықты бөлу;

− күрделі операцияларды орындау логикасын сипаттау, әсіресе бір объект бірнеше объектілермен өзара әрекеттескен жағдайларда;

− жүйенің ішіндегі объектілермен орындалатын рөлдерді, сондай-ақ осы рөлдерді орындай отырып, олар тартылатын объектілер арасындағы қатынастарды зерттеу.

Кооперация диаграммалары туралы айтатын болсақ, мұндай диаграммалардың екі "деңгейі" – даналар деңгейі (мысалы, Instance-Level) және спецификация деңгейі (Specification-Level) жиі айтылады. Айырмашылық неде? Жауап қарапайым: даналардың деңгейі объектілер (сынып даналары) арасындағы өзара әрекеттесуді көрсетеді; мұндай диаграмма әдетте объектілі-бағытталған жүйенің ішкі құрылғысын зерттеу үшін жасалады. Спецификация деңгейі жүйеде негізгі кластармен орындалатын рөлдерді зерттеу үшін қолданылады. Кез келген жағдайда, өзара іс-қимыл диаграммасы процесті көрсетпейді. Ол хабар жіберу және қабылдау арқылы жүзеге асырылатын объектілер арасындағы өзара іс-қимылды көрсетеді. Бұл ретте хабарламалардың дәл тізбегі тізбектер диаграммасында сияқты жақсы көрінбейді, сондықтан сіз үшін хабарламаларды жіберу және қабылдау тәртібін көрсету маңызды болса, тізбектер диаграммасын пайдаланыңыз. Сонымен жобаның кооперация диаграммасы төменде көрсетілген:



2.5-сурет – Жобаның кооперация диаграммасы

2 Жүзеге асыру және тестілеу бөлімі

3.1 Android Studio мүмкіндіктері

Android Studio-Android қосымшаларын әзірлеу үшін ресми түрде Google IDE қолдауы тиіс. IntelliJ IDEA, Android Studio Apache 2.0 лицензиясымен қол жетімді. 2.1.1 өзекті тұрақты нұсқасы келесі функцияларды қамтиды:

− барлық Android құрылғыларына арналған қолданбаларды әзірлеуге болатын бірыңғай орта;

− Android TV және Android Wear қосымшаларын жасау мүмкіндігі;

− үлгілердің негізінде жұмыс істейтін Android ортақ макеттері мен компоненттерін жасау үшін" шеберлер";

− пайдаланушы интерфейсінің компоненттерін сүйретуге мүмкіндік беретін және бірнеше экрандарда макеттерді алдын ала қарау мүмкіндігін қамтитын функционалдық макеттер редакторы;

− Android және жылдам түзетулер үшін Рефакторинг;

− Gradle негізінде дамуды қолдау;

− өнімділікті арттыру үшін Lint құралдары, юзабилити, нұсқалардың үйлесімділігіне байланысты проблемаларды жою және т. б.;

− ProGuard интеграциясы және бағдарламаларға жазылу мүмкіндігі;

− жылдам және көп функциялы эмулятор;

− Instant Run жаңа ҚХА файлын жасаусыз іске қосылған қолданбаға өзгерістер енгізу үшін (Application PacKage Zip);

− Google Cloud Messaging және App Engine кіріктірілген Google бұлтты платформасын қолдау;

− C++ және NDK;

− плагиндер арқылы Android Studio мүмкіндіктерін кеңейту.

Android Studio жүктеу. Google Windows, Mac OS X және Linux үшін Android Studio ұсынады. Бұл бағдарламалық жасақтаманы бағдарламаның ресми бетінен жүктеп алуға болады. Android Studio орнату алдында Операциялық жүйе және компьютер келесі талаптарға сәйкес келетініне көз жеткізіңіз:

Windows ОЖ:

− Microsoft Windows 7/8/10 (32 биттік немесе 64 биттік нұсқасы);

− Гбайт жедел жады, 8 Гбайт жедел жады ұсынылады;

− ГБ дискідегі бос орын, ұсынылады 4 ГБ (IDE үшін 500 МБ + SDK Android және эмулятор бейнесі үшін 1,5 ГБ);

− 800 пиксельге 1280 экранның ең аз ажыратымдылығы;

− JDK 8;

− жедел эмулятор үшін: 64 биттік операциялық жүйе және Intel VT-x, Intel EM64T (Intel 64) және Execute Disable (XD) функцияларын қолдайтын Intel процессоры.

Mac OS:

− Mac OS X.5 немесе одан жоғары, 10.11.4 дейін (El Capitan);

− Гбайт жедел жады, 8 Гбайт жедел жады ұсынылады;

− ГБ дискідегі бос орын, ұсынылады 4 ГБ (IDE үшін 500 МБ + Android SDK және эмулятор бейнесі үшін 1,5 ГБ);

− Android Studio үшін экранның минималды ажыратымдылығы 1280 үшін 800 пиксель болуы керек;

− JDK 6.

Linux ОЖ:

− Gnome немесе KDE бар стационарлық компьютер: Ubuntu 12.04 сыналған, Precise Pangolin (64-биттік бөлу, 32-биттік бағдарламаларды қолдайтын);

− 32 биттік қосымшаларды іске қосуға мүмкіндік беретін 64 биттік бөлу;

− GNU с кітапхана (glibc) 2.11 нұсқасы немесе жаңа;

− Гбайт жедел жады, 8 Гбайт жедел жады ұсынылады;

− ГБ дискідегі бос орын, ұсынылады 4 ГБ (IDE үшін 500 МБ + Android SDK және эмулятор бейнесі үшін 1,5 ГБ);

− 800 пиксельге 1280 экранның ең аз ажыратымдылығы;

− JDK 8;

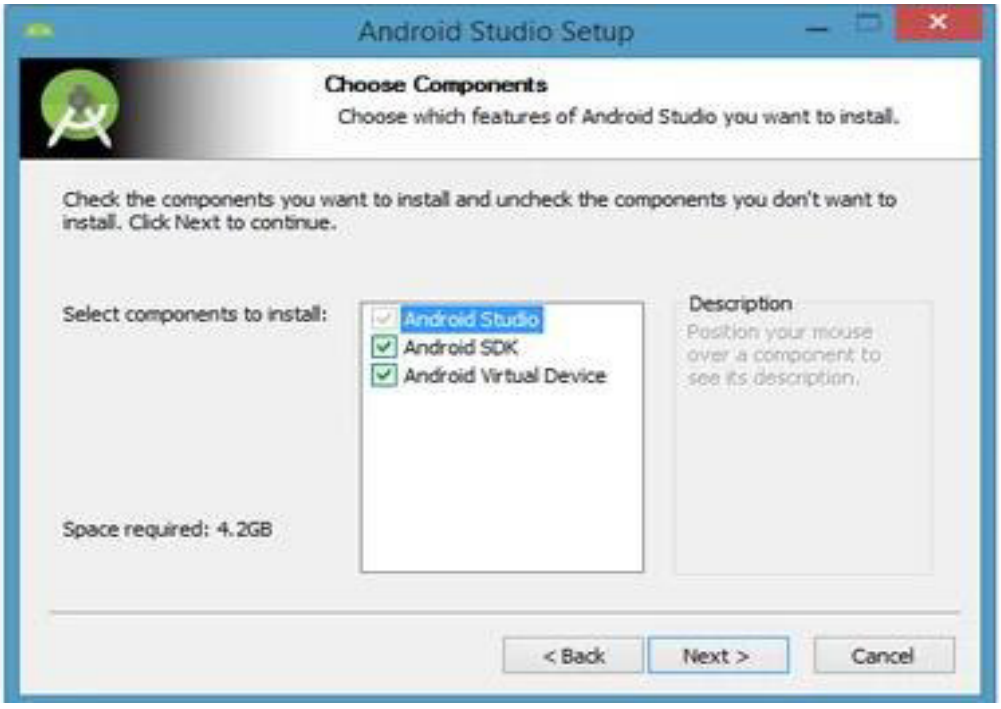
− жедел эмулятор үшін: Intel VT-x, Intel EM64T (Intel 64) және Execute Disable (XD) функцияларын қолдайтын Intel процессоры немесе AMD (AMD-V) виртуалдау технологиясын қолдайтын AMD процессоры.

Операциялық жүйе үйлесімді және Android Studio орнату және баптау үшін жарамды екеніне көз жеткіземіз, тиісті дистрибутивті жүктеп аламыз.

64 биттік Windows 8.1 Android Studio орнату. Мен android-studio-bundle-143.2821654-windows.exe процесін орнатып бастау үшін іске қостым. Орнатушы 1-суретте көрсетілген Android Studio орнату тілқатысу терезесін шығарады:

3.1-сурет – Android Studio Орнату

"Келесі" түймесін басып, Мен Android SDK (орнату жиынтығына кіремін) және Android Virtual Device (AVD):

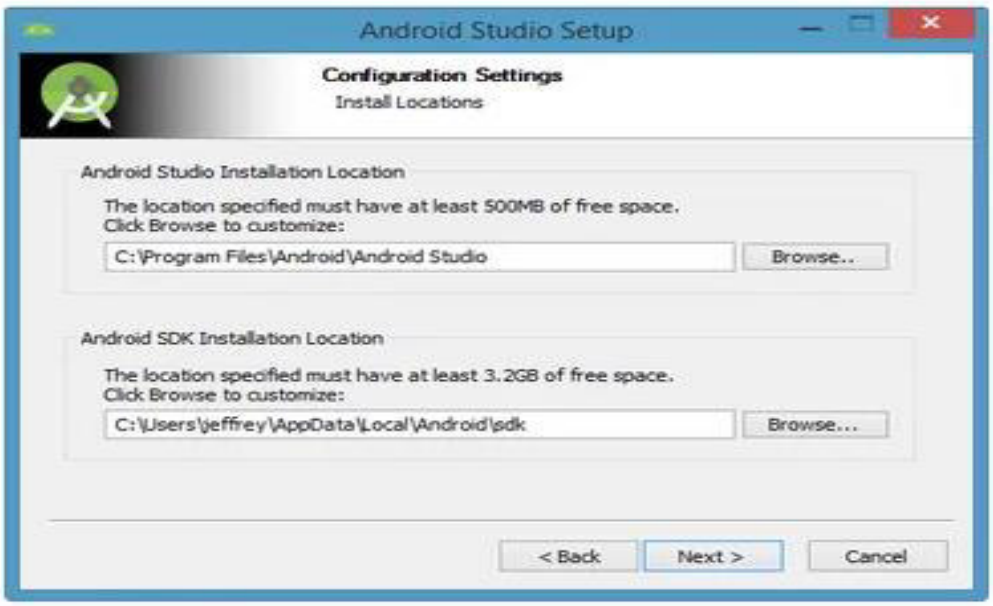


3.2-сурет – Android SDK және AVD орнату керек пе?

Мен әдепкі параметрлерді сақтауды шештім. "Бұдан әрі" батырмасын басқаннан кейін сіз лицензиялық келісімнің диалогтық терезесіне түсеміз. Орнатуды жалғастыру үшін оның шарттарын аламыз:

3.3-сурет – Лицензиялық келісім шарттары

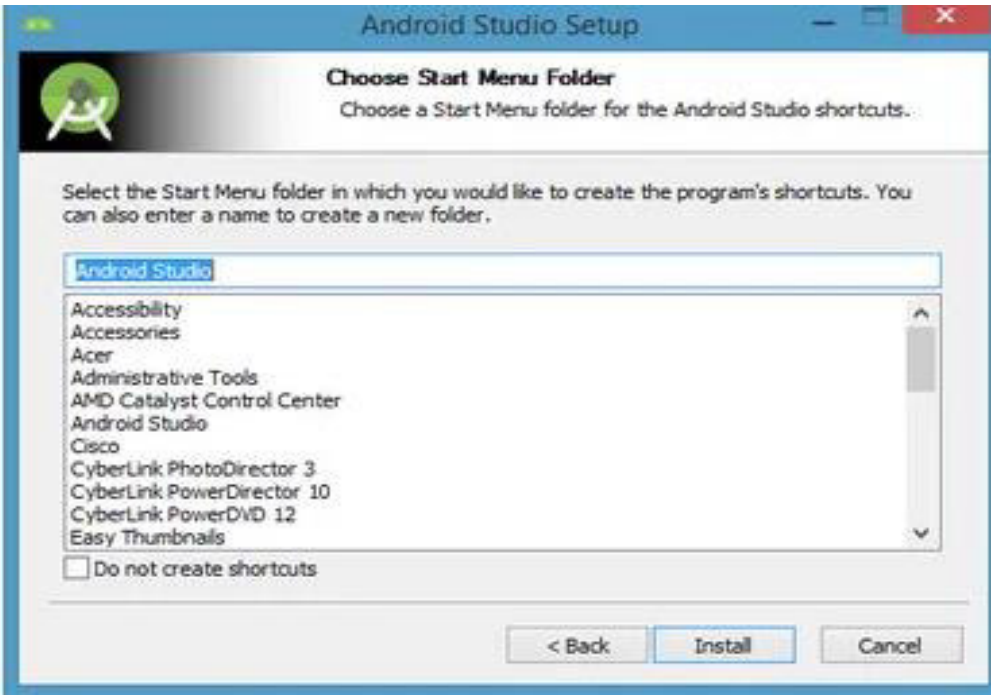
Келесі диалогтық терезеде Android Studio және Android SDK орнату қалтасын өзгерту ұсынылады:



3.4-сурет – Android Studio және Android SDK орнату орнын көрсету

Орнату орнын өзгертеміз немесе әдепкі параметрлерді пайдаланамыз және "Келесі"түймешігін басамыз.

Әдепкі бойынша орнатушы бағдарламаны іске қосу үшін белгі жасайды. Орнатуды бастау үшін Орнату түймешігін басамыз:



3.5-сурет – Android Studio үшін таңбашалар жасау

Пайда болған диалогтық терезеде Android Studio және Android SDK орнату барысы көрсетіледі. "Егжей-тегжейлі көрсету" түймешігін басып, орнату барысы туралы толық ақпаратты көре аласыз.

Диалогтық терезеде орнатудың аяқталғаны туралы хабар көрсетіледі:

3.6-сурет – Android Studio іске қосу

Орнатуды аяқтау және Android Studio сабаққа кірісу үшін "Дайын" түймешігін басамыз.

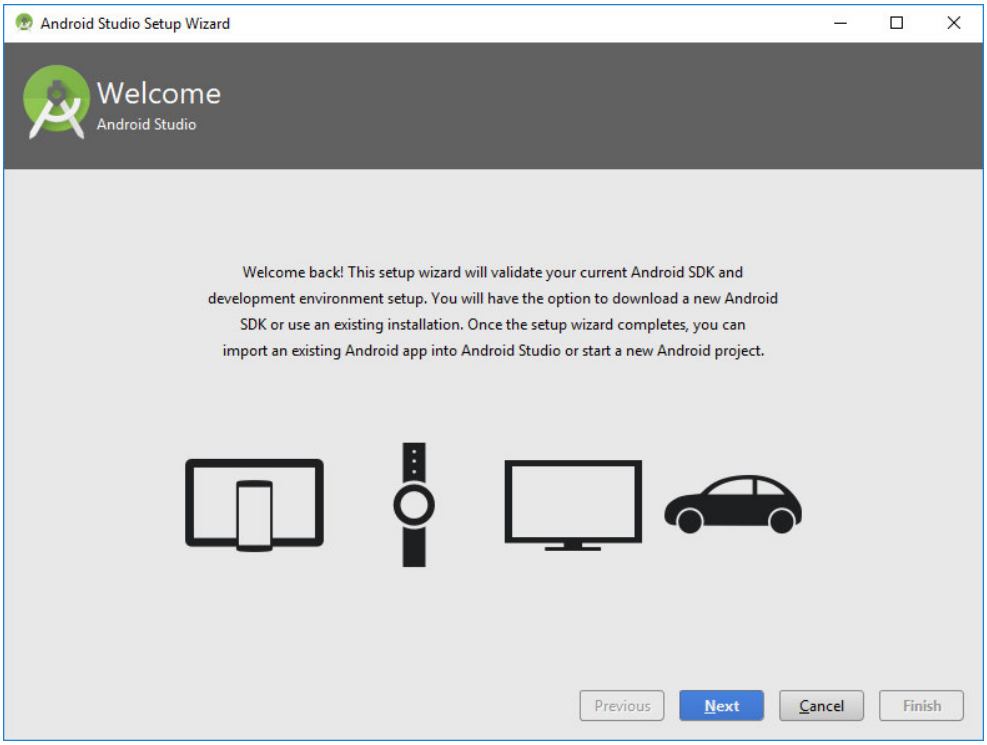
Android Studio Іске Қосу. Android Studio өз іске қосу кезінде келесі заставканы шығарады:

3.7-сурет – Android Studio экраны

Бірінші рет іске қосылғанда, диалог терезелерінде бірнеше теңшелім параметрлерін орнату ұсынылады. Бірінші тілқатысу терезесінде Android Studio нұсқасынан параметрлерді импорттауға назар аударамыз:

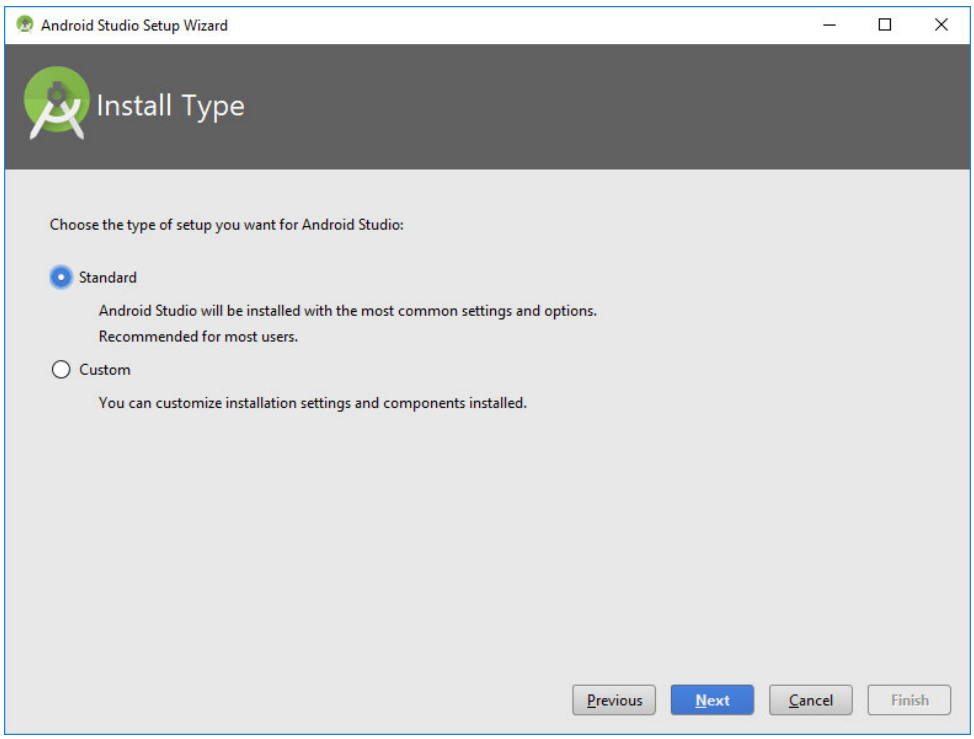
3.8-сурет – Импорт параметрлері

Қабылдауға болады, әдепкі "OK" батырмасын басамыз. Осыдан кейін Android Studio "орнату шебері" диалог терезесін шығарады»:



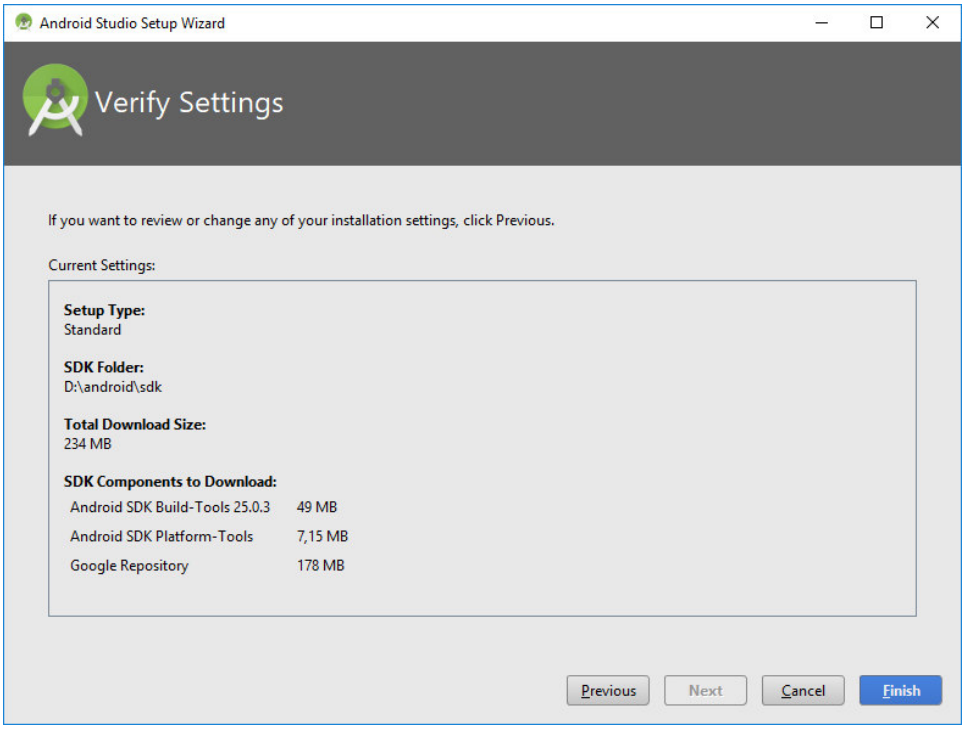
3.9-сурет – Android SDK параметрлерін және даму ортасын тексеру

"Әрі қарай" батырмасын басқаннан кейін "орнату шебері" SDK компоненттерін орнату түрін таңдауды ұсынады. Қазіргі уақытта стандартты конфигурацияны пайдалануды ұсынамын:



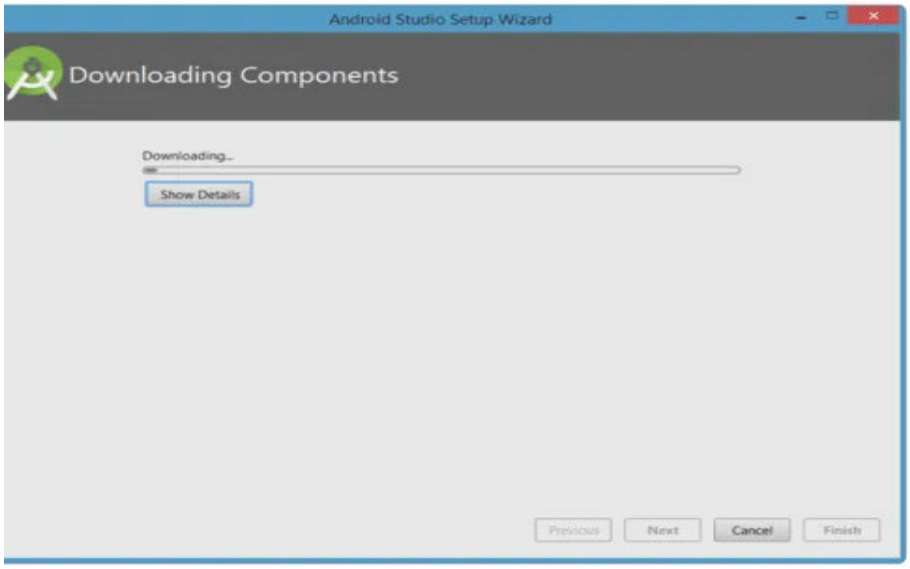
3.10-сурет – Орнату түрін таңдау

"Келесі" түймешігін басамыз және таңдалған параметрлерді растаймыз. Одан кейін жалғастыру үшін "Дайын" түймешігін басамыз:



3.11-сурет – Параметрлерді көру

Орнату шебері " қажетті компоненттерді жүктейді және тарқатады. Жүктелетін архивтер мен олардың мазмұндары туралы толық ақпаратты көргіңіз келсе, "мәліметтерді көрсету" түймешігін басамыз:



3.12-сурет – Android Studio компоненттерін жүктеу және тарату

Android Studio орнату алдында жүйенің параметрлерін тағы да тексеріңіз. Егер сіздің компьютер Intel процессоры негізінде жиналмаса, компоненттер толығымен жүктеліп, пакеттелген соң жағымсыз тосын пайда болуы мүмкін:

3.13-сурет – Intel базасында аппараттық жеделдету мүмкін емес

Проблеманы шешудің мүмкін нұсқалары-баяу эмулятор немесе Android құрылғысын әзірлеу үшін пайдалану.

Орнату шеберінің жұмысын аяқтау үшін "Дайын" түймесін басамыз. Содан кейін диалог терезесін көремін "Android Studio қош келдіңіз»:

3.14-сурет – "Android Studio қош келдіңіз»

3.2 Java тілінің басқа тілдерден артықшылықтары

Қазір Java — Android-қосымшаларында және үлкен көлемді мәліметтермен (big data) жұмыс жасауда қолданылатын басты тіл. Әрине, бағдарламалау тарихы Java-мен аяқталмайды. Бірақ жаңа тілдердің көпшілігі тар шеңберлі мәселелер аясында жобаланады, сондықтан оларды өзара салыстыру қиын. Мысалға, Swift тілін Apple компаниясы арнайы iOS пен Mac қосымшалары үшін құрастырған, ал Julia тілі үлкен көлемді есептеулермен жұмыс істеу кезіндегі жоғары өнімділікті сақтау мақсатында пайда болған.

Java шынында да желілік ортада құнды құрал болып табылады. Бірақ ол тек онымен шектеліп қалмайды. Java тілі әртүрлі бағдарлама жасау үшін қолданылатын универсалды тіл. Оған қоса Java платформаға тәуелсіз проектер жасау үшін қолданылады. Бағдарлаудың қарапайымдылығы және қауіпсіздік амалдары тез және түсінікті код жазуға мүмкіндік береді. Басқа бағдарлау тілдерінде кеңінен тараған қателер Java бағдарламаларында болмайды. Десекте, барлық ішкі әдіс – амалдарына қарамастан Java бағдарлау тілі қарапайым, үйренуге оңай тіл.

Артықшылықтары:

- архитектураға тәуелсіз;
- интернет бағдарламалар үшін мінсіз;

− бір рет жазылған код басқа жерлерде орындалады;
− көп ағымды;
− бағдарламалар желі арқылы тапсырманы орындай алады;
− динамикалы;
− код қажет болған жағдайда ғана қолданылады;
− ООП-ның бар болуы;
− Java бағдарлау тілі қазіргі ООП негізіне сәйкес бағдарламалар жазу үшін қолданылады;
− жад басқарылмалы;
− жад автоматты түрде бөлінеді;
− “Garbage collector” қолданылмайтын жадты өшіріп отырады;
− сенімді;
− қатты типтелген;
− көрсеткіштер жоқ (С бағдарлау тілін өткенде түсінесіздер;
− қарапайым;
− бір-бірімен өте жақсы қиысқан тілдің элементтері оның қолданысын жеңілдетеді.

Java бағдарлау тілі – ол JDK (Java Development Kit). Яғни, бағдарлау тілі бір-бірімен топтасқан көптеген кішкентай бөлшектерден, класстардан,тұрады. Демек, Java-да жазылған бағдарламалар, негізінде, осы класстардың жиынтығын, пакеттерді, қолданады.

Сонымен, Java-да бағдарлау кезінде қолданылады:
− Java-ның страндартты пакеттеріндегі класстар;
− өзіміз жаңадан жазған класстар;
− басқалардың көпшілік үшін жазған класстары.

Java-да бағдарламаның жазу процессі:

Жазу: java файлын жазу арқылы өзіміздің классты анықтаймыз.
Компиляция: жазылған .java файлды Java компиляторына жіберіп .class файлды аламыз.

Орындау: пайда болған .class файлды Java интерпретаторына кодты орындау үшін жібереміз.

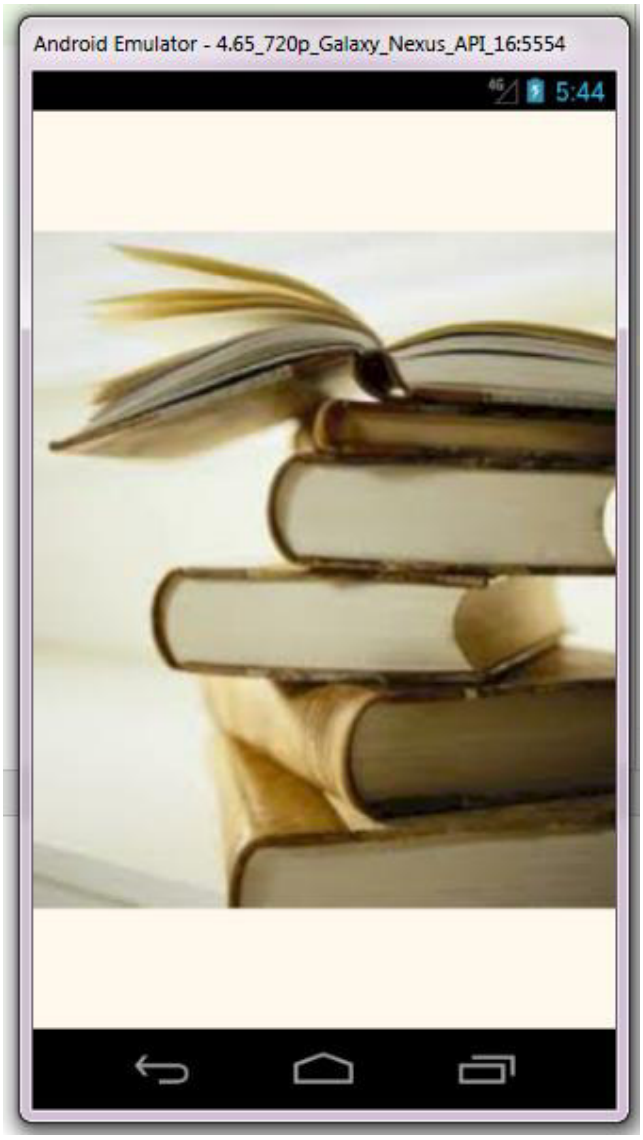
Java компиляторы – жазылған кодты орындау үшін дайындайды және байт-коды бар .class файлды шығарады. Егер компиляция кезінде қателер болса, онда кодтағы қателерді түзетіп қайтадан компиляциядан өткізу керек. Байт-код – Java виртуалды машинасында орындалатын оңтайландырылған нұсқаулардың жиынтығы. Яғни, JVM байт-кодтың интерпретаторы. Бұл әдіс басында әдеттегіден тыс болып көрінуі мүмкін, өйткені қазіргі кездегі бағдарлау тілдерінің компиляторлары байт-кодты шығармай бірден орындалатын кодты шығарады (.exe). Бірақ бұндай әдістің өзіндік артықшылықтары бар және жылдамдық ерекшеліктері зор емес. Java интерпретаторы (JVM – Java Virtual Machine) – .class файлды оқып оны компьютерге түсінікті тілге аударады, яғни бағдарламаны орындайды.

3.3 «Электрондық кітапхана» мобильдік бағдарламасын құру және оны іске асыру

«Электрондық кітапхана»- бұл қолданушыға ыңғайлы болу үшін жасалған кітаптардың онлайн мобильді қосымшасы. Оның көмегімен оқырмандар өздері кез-келген уақытта ғаламтор желісінің көмегінсіз керекті кітаптарды оқи алады, оқу үшін қызықты материалдармен өте қолжетімді баға бойынша оқи алады.

«Электрондық кітапхана» мобильді қосымшасының өзектілігі кез-келген уақытта қолжетімділігі және ыңғайлылығы болғандықтан, бірінші ретте мобильді қосымшаның дизайн жағын қарадым. Мобильді қосымша бейнесі ыңғайлы түрде, түсінікті түрде жасалған.

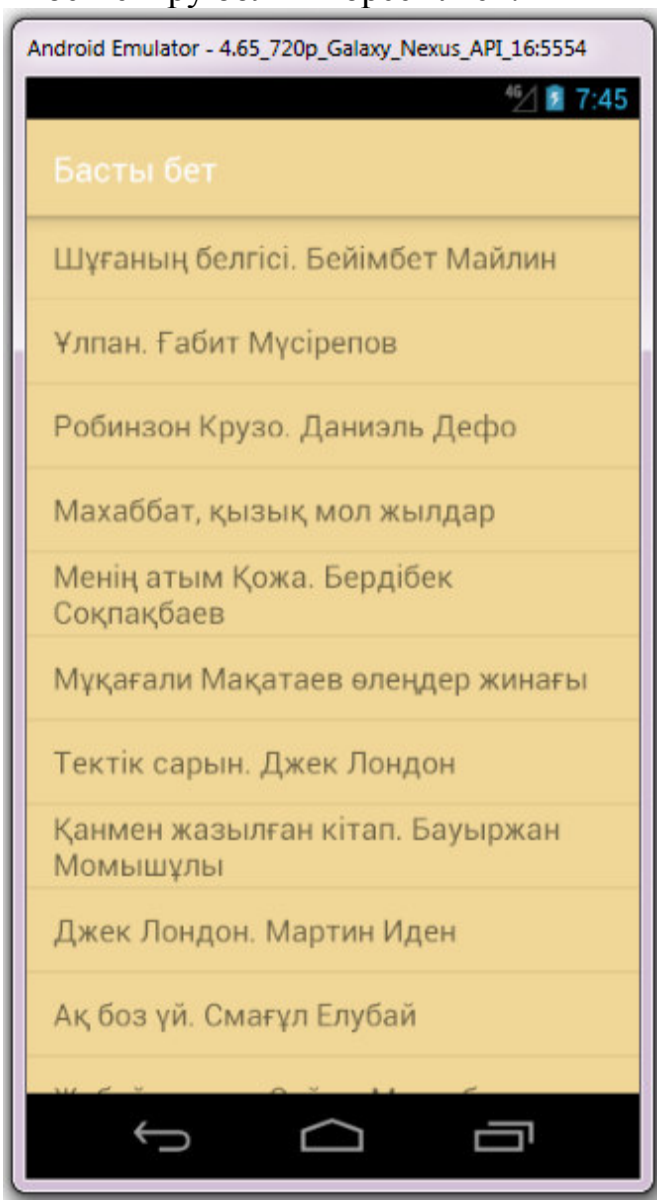
Электрондық кітапхана мобильді қосымшасын Android Studio платформасында жасадым. Бағдарламалау тілі ретінде Java тілін пайдаландым. Төменде мобильді қосымшаның Flash экран бөлімі көрсетілген.



3.15-сурет – Splashscreen activity

Мен бұл қосымшада 3 activity қолдандым: Splashscreen, MainActivity, DetailActivity. Splashscreen – алғашқы activity болып табылады. Бұл экрандағы «Кітапхана» қосымшасының өзектілігі суреттелген бетті setContentView және ImageView элементтері арқылы жасадым.

Келесіде басты бетте кіру бөлімі көрсетілген.



3.16-сурет – Кітаптар тізімі

Басты бет – Main Activity болып табылады. Бұл жерге кітаптар тізімін енгізілдім. Кітаптар тізімі Listview, ArrayAdapter идентификаторлар арқылы жасадым.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

```
        // ListView идентификаторын шақырамыз
        ListView listView = findViewById(R.id.listView);
        //ListView массивін орнықтырамыз
        listView.setAdapter(
                new ArrayAdapter<>(this, android.R.layout.simple_list_item_1,
titles));
        listView.setTextFilterEnabled(true);
```

Ары карай setOnItemClickListener функциясы көмегімен тізімнің шертулерін өңдедім.

```
        //ListView элементіндегі шертулерді өңдейміз
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> a, View v, int position, long id)
{
                Intent intent = new Intent();
                intent.setClass(MainActivity.this, DetailActivity.class);
                intent.putExtra("title", position);
                //екінші белсенділікті шакырамыз
                startActivity(intent);
            }
        });
```

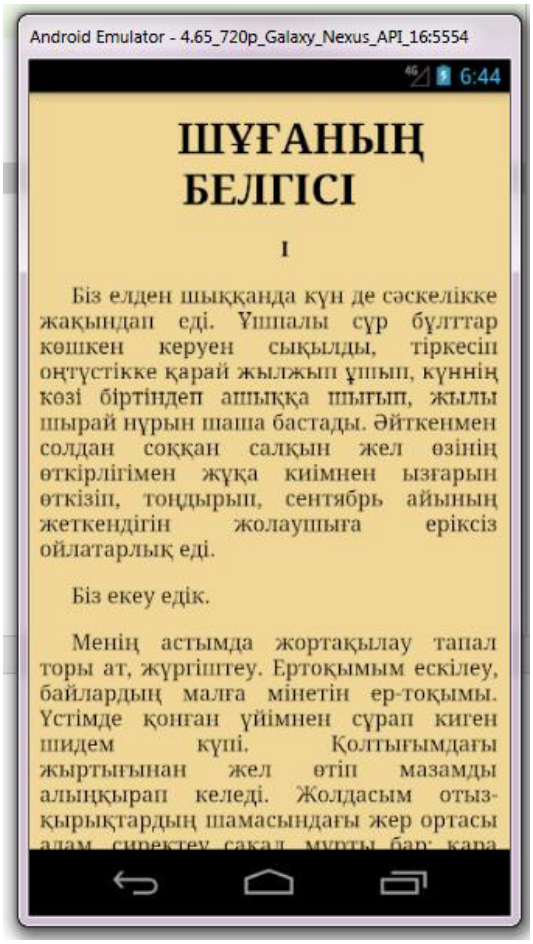
Экрандағы орналасу ретін LinearLayout контейнеріндегі Listview константалары арқылы жүзеге асырдым.

```
        <LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
            xmlns:tools=http://schemas.android.com/tools
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            tools:context=".MainActivity">
            <ListView
                android:id="@+id/listView"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"/>
        </LinearLayout>
```

Келесі activity - DetailActivity. Бұл белсенділікте ашылған кітапты оқу үшін, WebView компонентін қолдандым.

```
            {    super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_detail);
                Objects.requireNonNull(getSupportActionBar()).hide();
                WebView webView = findViewById(R.id.webView);
                Intent intent = getIntent();
                String resName = "n" + intent.getIntExtra("title", 0);
                Log.i("name", resName);
                Context context = getBaseContext();
```

```
String text = readRawTextFile(context,
getResources().getIdentifier(resName,
"raw", "kz.kermanalievauldana.kitaphana"));
webView.loadDataWithBaseURL(null, text, "text/html", "utf-8", null);}
```



3.17-сурет – Оқу терезесі

Қосымша ғаламтор желісіз оқуға арналғандықтан, кітаптар файлы жобаның ресурсына яғни космышаның ішіне орналастырылдым, және еш сырткы браузерсіз ашылуы ушін WebView компоненті қолданылды.

InputStreamReader, StringBuilder компоненттер арқылы, жобаның raw қапшығындағы файлдарды экранға текст қылып шығарылды.

```
private String readRawTextFile(Context context, int resId)
{    InputStream inputStream =
context.getResources().openRawResource(resId);
InputStreamReader inputReader = new InputStreamReader(inputStream);
BufferedReader buffReader = new BufferedReader(inputReader);
String line;
StringBuilder builder = new StringBuilder();
try {
    while (( line = buffReader.readLine()) != null) {
```

```
        builder.append(line);
        builder.append("\n");  }
} catch (IOException e) {
    return null; }
return builder.toString();}
```

4 Экономикалық бөлім

4.1 Жұмыстың экономикалық сипаттамасы және тиімділігі

Бұл дипломдық жұмыстың мақсаты Алматы энергетика және байланыс университеті үшін Android платформасыда «Электронды кітапхана» мобильді қосымшасының жүйесін құру. Жүйені құру үшін Android Studio 5.0 ортасы қолданылады.

Электронды кітапхана жүйесінің тиімділігі бұл шығындар мен уақытты үнемдеу болып табылады. Еңбекке ақы төлеу шығындарын төмендету қаржылық шығындарды азайтады, бұл өнімділік пен үнемдеудің жалпы өсуіне әкеледі.

Бұл бөлімде берілген жобаның экономикалық жүзеге асыруын құрайтын уақытша, еңбектік және қаржылық шығындарын көретін боламыз.

Осы бөлімнің негізгі мақсаты - зерттеудің шығындарының шамасын анықтау, осы ғылыми жобада техникалық проблеманы шешуде алынған негізгі және онымен байланысты нәтижелерді әлеуметтік өндірісте пайдаланудың экономикалық тиімділігін анықтауға арналған шығындарды есептеу.

4.2 Мобильді қосымшаны әзірлеудің еңбек сыйымдылығы

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

Мобильді қосымшаны әзірлеу кезеңдері	Жұмыс түрлері	Әзірлеудің еңбек сыйымдылығы, адам× сағ.
1 кезең	Пәндік сала туралы деректерді жинау	12
2 кезең	Деректерді өңдеу	7
3 кезең	Деректер қорының құрылымын құру	14
4 кезең	Алгоритмді әзірлеу	47
5 кезең	Бағдарламаны жазу	64
6 кезең	Бағдарламаны жөндеу	102
7 кезең	Бағдарламамен жұмыс істеу бойынша басшылық дайындау	110
8 кезең	Анықтамалық жүйені әзірлеу	16
Нәтиже	Бағдарламалық өнімді орындаудың еңбек сыйымдылығы	372

4.3 Мобильді қосышаны әзірлеуге арналған шығындарды есептеу

Мобильді қосышаны әзірлеуге арналған шығындарды анықтау тиісті сметаны жасау жолымен жүргізіледі, ол мынадай баптарды қамтиды:

− материалдық шығындар;

- еңбекке ақы төлеу шығындары;
- әлеуметтік салық;
- негізгі қорлардың амортизациясы;
- өзге шығындар.

4.2-кесте – Материалдық ресурстарға шығындар

Материалдық ресурстың атауы	Бірлік өлшем	Саны	Бірлік бағасы, тг	Сомасы, тг
Ноутбук *HP*	Штук	1	230000	230000
Смартфон Samsung A7	Штук	1	80000	80000
Нәтиже	Материалдық ресурстарға шығындар жиыны			310000

Алматы қаласы бойынша 2019 жылы заңды тұлғалар үшін Тариф ҚҚС есебімен 1 кВт/с үшін 18,32 теңгені құрайды ("АлматыЭнергоСбыт"ЖШС ресми сайтында ұсынылған деректерге сәйкес). Шығындар 3-кестеде көрсетілген.

Электр энергиясына жұмсалатын шығындардың жалпы сомасы (ЗЭ) (4.1) формула бойынша есептеледі:

$$З_э = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц \qquad (4.1)$$

мұндағы, M_i - I электр жабдығының паспорттық қуаты, кВт;

Кі - і электр жабдығының қуатын пайдалану коэффициенті (Кі=0.9 қабылданады);

T_i - І-ші жабдықтың барлық әзірлеу кезеңіндегі жұмыс уақыты;

Ц - электр энергиясының бағасы, тг / кВт×сағ;

I - электр жабдығының түрі;

n - электр жабдықтарының саны.

$$З_э = 0.9 \cdot 0.8 \cdot 372 \cdot 18.32 = 4906,8$$

$$З_э = 0.3 \cdot 0.7 \cdot 372 \cdot 18.32 = 1431,1$$

$$З_э = 1,970 \cdot 0.8 \cdot 323 \cdot 18.32 = 9325,76$$

4.3-кесте – Электр энергиясына арналған шығындар

Жабдықтың атауы	Паспорттық қуаты, кВт	Қуатты пайдалану коэффициенті	Мобильді қосымшаны әзірлеуге арналған жабдықтың жұмыс уақыты, адам	Электр бағасы қуаты, тг / кВт сағ	Сомасы, тг
Ноутбук *HP*	0.9	0.8	372	18,32	4906,8
Освещение	0,3	0,7	372	18,32	1431,1
Смартфон Samsung A7	1,970	0,8	323	18,32	9325,76
НӘТИЖЕ электр энергиясына арналған шығындар					15663,66

4.4 Мобильді қосымшаның еңбекақы төлеу шығындары

Инженер-әзірлеушінің орташа жалақысы 2019 жылы 180 000 тг, ал бағдарламашы 150 000 тг (Алматы қаласы үшін) құрайды.

Қызметкердің бір айдағы жұмыс сағаттары (4.2) формула бойынша анықталады:

$$Ч_м = N_м \cdot Ч_{рд} \quad (4.2)$$

мұндағы, Чм - бір айдағы қызметкердің жұмыс сағаты;
Nм - бір айдағы жұмыс күндерінің саны;
Чрд - күніне жұмыс сағаттарының саны.

$$Ч_м = 21 \cdot 8 = 168$$

Қызметкердің сағаттық бағасы (4.3) формула бойынша есептелуі мүмкін:

$$ЧС = \frac{ЗП_i}{ФРВ_i} \quad (4.3)$$

Инженер-разработчик (Әзірлеуші-инженер):

$$ЧС_i = \frac{180000}{168} = 1071{,}43 \text{ тг}$$

Программист:

$$\text{ЧС}_i = \frac{150000}{168} = 892{,}86 \text{ тг}$$

мұндағы, ЗПі - і қызметкердің айлық жалақысы, тг;

Фрв - і қызметкердің жұмыс уақытының айлық қоры, сағ.

Мобильді қосымшаы әзірлеудің еңбек сыйымдылығын анықтау үшін 4-кестеден алынған деректер пайдаланылады.

Мобильді қосыша әзірлеудің еңбек сыйымдылығы әзірлеуші инженер 255 адамға тең. (пәндік сала туралы деректерді жинау, деректерді өңдеу, Деректер базасының құрылымын құру, алгоритм әзірлеу, бағдарлама жазу, бағдарламаны жөндеу, бағдарламамен жұмыс істеу бойынша басшылықтарды дайындау, анықтамалық жүйені әзірлеу).

$$\text{Т}_1 = 12 + 7 + 14 + 47 + 64 + 102 + 16 = 262$$

Программисттің мобильді қосымшаны әзірлеу еңбек сыйымдылығы 219 адамға тең. (тапсырма қою, бағдарлама модульдерін әзірлеу, жүйені тестілеу).

$$\text{Т}_1 = 7 + 102 + 110 = 219$$

Еңбекақы төлеуге жұмсалатын шығындардың жалпы сомасы (ЗТР) (4.4) формула бойынша анықталады:

$$\text{З}_\text{э} = \sum_{i=1}^{n} \text{ЧС}_i \times T_i \qquad (4.4)$$

мұндағы, ЧСі - і қызметкердің сағаттық ставкасы, тг;

Ті - ПП әзірлеудің еңбек сыйымдылығы, адам×сағ;

І - қызметкердің санаты;

n - ПҚ әзірлеумен айналысатын қызметкерлердің саны.

Инженер-разработчик (Әзірлеуші-инженер):

$$\text{З}_\text{тр} = 1071{,}43 \cdot 262 = 280714{,}66 \text{ тг}$$

Программист:

$$\text{З}_\text{тр} = 892{,}86 \cdot 219 = 195536{,}34 \text{ тг}$$

Жалпы сомасы:

$$З_{тр.о} = 280714,66 + 195536,34 = 476251 \text{ тг}$$

4.4-кесте – Еңбекақы төлеу шығындары

Квалификация	Мобильді қосымша әзірлеудің еңбек сыйымдылығы, адам×сағ	Сағаттық ставка, тг / сағ	Сомасы, тг
Инженер-разработчик	262	1071,43	280714,66
Программист	219	892,86	195536,34
Нәтиже	Еңбекақы төлеу шығындары		476251,00

Қосымша еңбекақы:

$$З_{доп} = З_{тр} \cdot 10\%$$

$$З_{доп} = 476251 \cdot 0,1 = 47625,1 \text{ тг}$$

Еңбекақы қоры:

$$Ф_{зп} = З_{тр.о} + З_{доп}$$

$$Ф_{зп} = 476251 + 47625,1 = 523876,1 \text{ тг}$$

Әлеуметтік салықты есептеу:

$$Н_с = (Ф_{зп} - ОПВ) \cdot 11\% \quad (4.5)$$

мұндағы, МЗЖ – міндетті зейнетақы жарнасы– 10% от $Ф_{ЗП}$.

$$Н_с = \big(523876,1 - (523876,1 \cdot 0.1)\big) \cdot 0.11 = 51863,73 \text{ тг}$$

Негізгі қорлардың амортизациясын есептеу:

Амортизациялық аударымдардың жалпы сомасы (4.6) формула бойынша анықталады:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times H_{Ai} \times T_{НИРi}}{100 \times T_{Эфi}}, \quad (4.6)$$

мұндағы, Фі - і ОФ құны, тг;

І - ОФ амортизациясының жылдық нормасы, %;

ТНИРі - ПП әзірлеудің барлық кезеңіндегі і-ОФ жұмыс уақыты, сағ.;

ТЭфі - і-ОФ бір жылдағы жұмыс уақытының тиімді қоры, сағ/жыл;

I - ОФ түрі;

n – ОФ мөлшері,

ОФ амортизация нормасының жылдық есебі.

Құрылғылар:

$$\mathrm{H_{Ai}} = \frac{100}{\mathrm{T_{Ni}}} \tag{4.7}$$

$$\mathrm{H}_{Ai} = \frac{100}{4} = 25$$

мұндағы, Тп - і ОФ пайдаланудың ықтимал мерзімі, жыл;

Қосымшаларды әзірлеу үшін БҚ жұмыс уақытын анықтау үшін 1-кестедегі деректер пайдаланылады.

Мобильді қосымшаны әзірлеу үшін Android Studio бойынша жұмыс уақыты 313 сағатты құрайды.

(алгоритм әзірлеу, бағдарламаны жазу, бағдарламаны жөндеу, бағдарламамен жұмыс істеу бойынша басшылықтарды дайындау).

$$\mathrm{T_1} = 47 + 64 + 102 + 110 = 323$$

Смартфон:

$$\mathrm{З_{АМ}} = \frac{230000 \cdot 25 \cdot 323}{100 \cdot 1920} = 9673{,}17 \text{ тг}$$

Ноутбук:

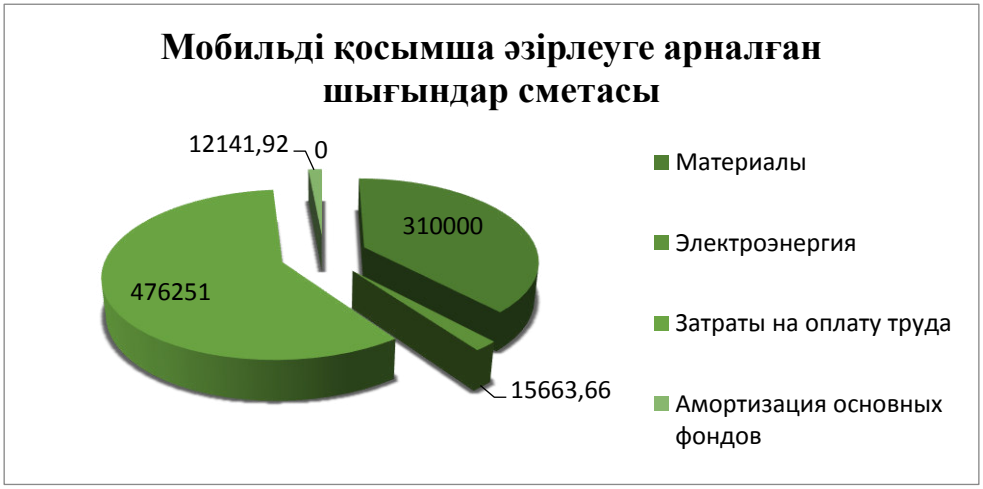
$$\mathrm{З_{АМ}} = \frac{80000 \cdot 25 \cdot 372}{100 \cdot 1920} = 2468{,}75$$

4.5-кесте – Негізгі қорлардың амортизациясы

Жабдық және БҚ атауы	Жабдықтармен БҚ құны, тг	Жылдық амортизация нормасы, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Ноутбук HP	230000	25	1920	372	2468,75
Смартфон Samsung A7	80000	25	1920	323	9673,17
НӘТИЖЕ					12141,92

4.6-кесте – Мобильді қосымша әзірлеуге арналған шығындар сметасы

Шығындар баптары	Сомасы, тг
Материалдық шығындар: - материалдар - электр энергиясы	 310000 15663,66
Шығындар баптары	Сомасы, тг
Еңбекке ақы төлеуге арналған шығындар.	476251
Негізгі қорлардың амортизациясы. Басқа шығындар.	12141,92 0
НӘТИЖЕ	814056,58



4.1-сурет – Шығындар диаграммасы

4.5 Мобильді қосымшаның ықтимал (шарттық) бағасын анықтау

Шарттық бағаны есептеу ($Ц_Д$):

$$Ц_Д = З_{НИР} \cdot \left(1 + \frac{Р}{100}\right), \quad (4.8)$$

мұндағы, $З_{нир}$ – әзірлеуге арналған шығындар ПП 1.6-кестеде көрсетілген, тг;

Р – ПП табыстылығының орташа деңгейі - 20%.

$$Ц_д = 814056{,}58 \cdot (1 + 0.2) = 814056{,}58 + 162811{,}32 = 976867{,}90 тг$$

2019 жылға ҚҚС ставкасы 12% мөлшерінде белгіленген.
ҚҚС есебімен өткізу бағасы мынадай формула бойынша есептеледі:

Цр=Цд + Цд * НДС.

$$Ц_р = 976867{,}896 + 976867{,}896 \cdot 0.12 = 1\ 094\ 092{,}04 тг$$

4.6 Мобильді қосымшаның экономикалық бөлімі бойынша қорытынды

Мобильді қосымшаның экономикалық бөлімі бойынша қорытындылай келе, осы бөлімде әзірленген бағдарламалық өнімді енгізудің экономикалық тиімділігі мен өзектілігі тұрғысынан талдау жүргізілді. Алдымен, бағдарламалық өнімді жасау бойынша жұмысты орындау мерзімі мен еңбек сыйымдылығы анықталды. Келесі қадам әзірленген мобильді қосымшаны іске асыру және енгізу үшін қажетті шығындарды есептеу болды. Соңғы кезең осы бағдарламалық өнімді енгізуден экономикалық тиімділікті есептеу болды. Жүргізілген есептеулер мен көрсеткіштер туралы айтуға мүмкіндік береді орындылығын және экономикалық тиімділік тұрғысынан алғанда, уақытша шығындарды енгізу, бағдарламалық өнім.

Осылайша, әзірленген бағдарламалық өнімнің өзіндік құны 814056,58 теңгеге тең,

Рентабельділік= 1 094 092,04 теңге

Мобильді қосымшаның рентабельділігі= 162811,32 теңге

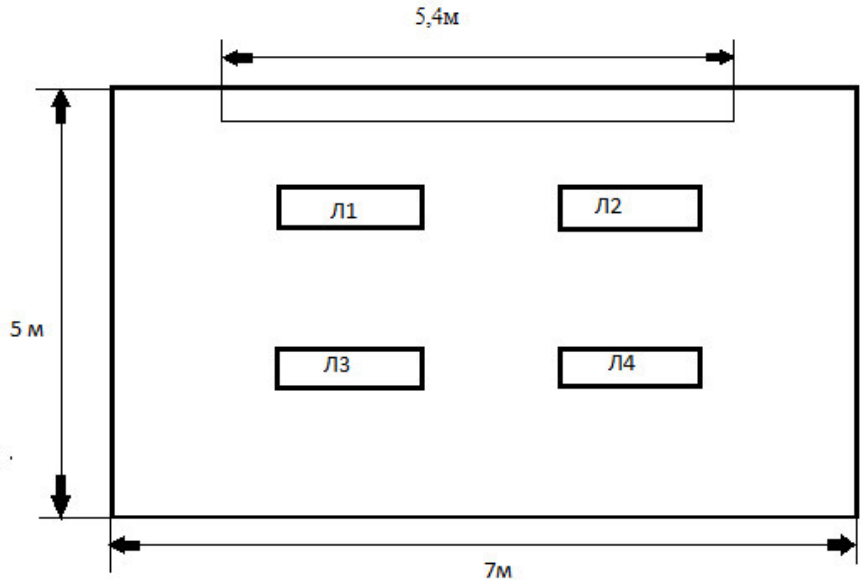
5 Өміртіршілік қауіпсіздігі

Дипломдық жобаның тақырыбы Android платформасында Электронды кітапхана мобильді қосымшасын әзірлеу. Бұл бөлмедегі вентиляция және Шудан оқшаулау жұмыс орнының талаптарына сәйкес келеді, себебі шулы құралдар жоқ және қазіргі кондиционер қажетті температураны, ауаның циркуляциясын және ылғалдылығын сақтайды. Ал жарықтандыру жақсы болды. Сондықтан дипломдық жобаның осы бөлімінде бөлме жарықтандырылуын есептеу және екі программист үшін қолайлы жағдай жасау туралы шешім қабылданды.

5.1 Жарықтандыруды есептеу

Программист орналасқан бұл жұмыс кабинеті ұзындығы 7 м, ені 5 м және биіктігі 3 м.

Көру жұмыстарының разряды – III, терезенің биіктігі – 1,8 м, терезенің басталу биіктігі 0,6 м. Жұмыс кабинетінің схемасы 5.1-суретте көрсетілген.



5.1-сурет – Жұмыс кабинетінің схемасы

Бұл бөлмеде бір терезе бар: ені 3 м және биіктігі 1,8 м.Жарық ойықтарының ауданы S=5,4 м2 жеткілікті ме, бөлменің қалыпты жарықтандырылуы үшін тексереміз. Есептеу (5.1) формула бойынша бүйірлік жарықтандыру кезінде жарық ойықтарының ауданын алдын ала анықтаудан тұрады:

$$100\frac{S_0}{S_n} = \frac{e_N K_з \eta_0}{\tau_0 r_1} K_{зд} \qquad (5.1)$$

мұндағы, S_0 – бүйірлік жарықтандыру кезіндегі жарық ойықтарының ауданы, м2;

S_n – үй-жай еденінің ауданы, м^2;

Е – ТЖК нормаланатын мәні;

$К_з$=1,2 – жарық өткізетін материалдың тік орналасуы кезіндегі қор коэффициенті 3.11-кесте бойынша қабылданады;

τ_0 – Жарық өткізудің жалпы коэффициенті;

r_1 – бөлме бетінен шағылысқан жарықтың және ғимаратқа жанасатын төсеніш қабаттың арқасында бүйірлік жарықтандыру кезінде ТЖК көтерілуін ескеретін коэффициент;

$К_{зд}$=1 – ескеретін коэффициент, терезелерді қараңғылау ғимараттарын қабылдайды 3.8-кесте бойынша [14].

Еден ауданы (5.2) формула бойынша анықталады:

$$S_n = L * B \tag{5.2}$$

Осылайша, (5.2) формула бойынша еден ауданы тең:

$$L = 7 \text{ м},$$
$$B = 5 \text{ м},$$
$$S_n = 7 * 5 = 35 \text{ м}^2$$

Табиғи жарықтандыру коэффициентінің нормаланған мәні (5.3) формула бойынша есептеледі:

$$e_N = e_н * m_N \tag{5.3}$$

мұндағы, N – табиғи жарықпен қамтамасыз ету жөніндегі әкімшілік-аумақтық аудан тобының нөмірі;

e_N = 1,2 – коэффициент мәні табиғи жарықтандыру бойынша 23.05.2011 байланысты сипаттамалары көру жұмыстары осы үй-жайда жүйесі және табиғи жарықтандыру, ТЖК мәні кезінде бүйірлік табиғи жарықтандыру 5.12-кесте бойынша [14];

m_N = 0,65 – жарықтық климаттың коэффициенті, кесте бойынша СниП түріне байланысты жарық ойықтарының, олардың бағдарлау бойынша тараптарға көкжиектің нөмірі тобының әкімшілік аудан, жарықтық климаттың коэффициенті сыртқы қабырғасында оңтүстіктен 5.1-кесте бойынша [14];

(5.3) формула бойынша әртүрлі аудандарда орналасқан ғимараттар үшін ТЖК e_N нормаланған мәндері тең:

$$e_N = e_н * m_N = 1.2 * 0.65 = 0.78\ \%$$

Бір жақты жарықтандыру кезіндегі үй-жайдың тереңдігі:

$$l = B - 1 = 5 - 1 = 4 \text{ м}$$

Жарықтылық сипаттамасының мәнін алу үшін, төмендегі қатынастар есептелген:

$$\frac{L}{l} = \frac{7}{4} = 1{,}75$$

$$\frac{l}{h_{\text{расч}}} = \frac{l}{(h_{\text{ок}} + h_{\text{нок}}) - h_{\text{рп}}} = \frac{4}{(1{,}8 + 0{,}6) - 0{,}8} = 2{,}5$$

Жарық сипаттамасының мәні 3.2 кесте бойынша қабылданады. Алынған мәндер кестедегі мәндерге сәйкес келмегендіктен, интерполяцияны өткіземіз:

$$x = 9{,}5 + \frac{(10{,}5 - 9{,}5) * (2{,}5 - 2)}{3 - 2} = 10$$

$$\eta_0 = 10$$

Жарық өткізудің жалпы коэффициенті τ_(0) (5.4) формула бойынша анықталады:

$$\tau_0 = \tau_1 * \tau_2 * \tau_3 * \tau_4 \qquad (5.4)$$

мұндағы, τ_1– 5.3 12-кесте бойынша анықталатын материалдың жарық өткізу коэффициенті. Терезелік табақ әйнектері үшін: дара,τ_1= 0.8;

τ_2 – терезе түптеуіндегі жарықтың жоғалуын ескеретін коэффициент. Бір терезе жақтаулары үшін $\tau_2 = 0.75$. [14];

τ_3 – тіреу конструкцияларындағы жарықтың жоғалуын ескеретін коэффициент, бүйірлік жарықтандыру кезінде 1-ге тең (5.5-кесте);

$\tau_4 = 1$ – күннен қорғайтын құрылғылардағы жарықтың жоғалуын ескеретін коэффициент [14].

(5.4) формуласына белгілі мәндерді қойып, τ_0 табамыз:

$$\tau_0 = \tau_1 * \tau_2 * \tau_3 * \tau_4 = 0.8 * 0.75 * 1 * 1 = 0.6$$

Үй-жай тереңдігінің шартты жұмыс бетінің деңгейінен терезенің үстіне дейінгі биіктікке қатынасы:

$$\frac{l}{h_{\text{расч}}} = \frac{4}{1{,}6} = 2{,}5$$

Есептік нүкте мен сыртқы қабырға арасындағы қашықтықтың үй-жайдың тереңдігіне қатынасы:

$$\frac{l}{B} = \frac{4}{5} = 0.8$$

Үй-жай ұзындығының оның тереңдігіне қатынасы:

$$\frac{L}{l} = \frac{7}{5} = 1{,}4$$

Төбенің, қабырғалардың және еденнің орташа өлшенген коэффициенті мынадай формула бойынша анықталады:

$$\rho = \frac{\rho_{\text{пот}} + \rho_{\text{ст}} + \rho_{\text{пол}}}{3} \qquad (5.5)$$

Осылайша, (5.5) формула бойынша:

$$\rho = \frac{\rho_{\text{пот}} + \rho_{\text{ст}} + \rho_{\text{пол}}}{3} = \frac{50 + 30 + 10}{3} = 30\% = 0.3$$

Алынған мәндер r_1 мәндерімен кестедегі мәндерге сәйкес келмегендіктен, интерполяцияны өткіземіз.

5.1-кесте - Коэффициенттің мәні r_1

Үй-жай тереңдігінің терезенің үстіңгі бетінің шартты жұмыс бетінің деңгейінен биіктікке қатынасы	Есептеу нүктесінің сыртқы қабырғадан үй-жайдың тереңдігіне қатынасы	Бүйірлік жарықтандыру кезінде r1 мәні		
		Төбенің, қабырғалардың және еденнің орташа өлшенген коэффициенті		
		0,3		
		Үй-жай ұзындығының оның тереңдігіне қатынасы		
		0,5	1	2 және одан көп
2,5-тен жоғары 3,5-ке дейін	0,1	1	1	1
	0,2	1	1,05	1,05
	0,4	1	1	1,1
	0,6	1,6	1,8	1,6
	0,8	1,9	1,7	1,4
	1	2,6	2,2	1,7

$$x = 1.7 + \frac{(1{,}4 - 1{,}7) * (1.4 - 1)}{2 - 1} = 1.58$$

$$r_1 = 1.58$$

Барлық табылған мәндерді орналастыра отырып, (5.1) формуладан бүйірлік жарықтандыру кезінде жарық ойықтарының ауданын S_0 табамыз:

$$S_0 = \frac{S_n e_N K_з \eta_0 K_{зд}}{100 \tau_0 r_1} = \frac{35 * 0.78 * 1.2 * 10 * 1}{100 * 0.6 * 1.58} = 3.455 \text{ м}^2 \approx 3 \text{ м}^2$$

Қорытынды: бұл кабинетте Жарық ойығының ауданы тексерілді, бөлмеде ол жеткілікті. Алайда кешкі және қысқы кезеңде қосымша жарық көздері қажет.

Жасанды жарықтандыру үй-жайда жүзеге асырылады пайдалана отырып, 4 люминесцентті шамдарды 65 Вт жарық ағынымен 3570 лм есептелген мөлшерден жалпы жарықтандыру (түрі шам – ПВЛМ). Ол үшін қажетті шамдарды пайдалану коэффициенті әдісімен есептейміз, ол көру жұмыстарының III–ші разрядын қамтамасыз ету үшін жеткілікті болып табылады. Үшін диспетчерлік пункттің қолданамыз газоразрядную люминесцентті шам ЛД–65, қуаты 65 Вт, жарық ағынымен 3570 лм, диаметрі 38 мм және ұзындығы-бабына штырьками 1515 мм [15]. (5.7) формула бойынша анықталады:

$$N_Л = \frac{S_{помещ} K_з Z E_Н}{Ф_Л n \eta} \qquad (5.7)$$

мұндағы, Z – ең аз жарықтану коэффициенті (орташа және ең аз жарықтану қатынасы). Есептерде z коэффициенті 1,1÷1,2 шегінде қабылданады;

n = 2 – амдағы шамдардың саны;

Фл - бөлменің индексіне және шағылысу коэффициенттеріне байланысты пайдалану коэффициенті.

$К_з$-қор коэффициенті. шаң бөлінбейтін үй-жайлар үшін $К_з$= 1,3 [15].

Жарықтандыру қондырғысын пайдалану кезінде жұмыс беттеріндегі жарықтандыру жарық көздерінің жарық ағынын азайту, шамдар мен жарықтандыру арматурасының ластануы, сондай-ақ жарықтандырылатын үй-жайдың қабырғалары мен төбесінің ластануы есебінен төмендейді. Сондықтан жарықтандыру қондырғысының қуатын анықтау кезінде қор коэффициенті енгізіледі. Қор коэффициенті ауаның шаңмен, түтінмен, найзамен және т. б. ластану дәрежесіне байланысты.

Жарықтандырудың бірқалыпты еместігін сипаттайтын Z коэффициенті көптеген айнымалы функциялар болып табылады және ең үлкен дәрежеде шамдардың арасындағы қашықтықтың шамдардың ілінуінің есептік биіктігіне (L/h) қатынасына байланысты болады, оны ұлғайтумен ұсынылған Z мәнінен

жоғары күрт өседі. Λ ұсынылған мәндерден аспайтын жағдайда қабылдауға болады:

Z = 1,15 - қыздыру шамдары және ДРЛ үшін;

Z = 1,1 - люминесцентті шамдар үшін [15].

Жарықтану қондырғысын пайдалану коэффициенті - бұл жұмыс бетіне түсетін жарық ағынының, жіберілетін жарық ағынына қатынасы.

Пайдалану коэффициенті шамның түріне, үй-жайдың геометриялық өлшемдеріне және беттердің шағылысу коэффициенттеріне байланысты [15].

Қолдану коэффициентін анықтау үшін бөлменің индексін анықтау қажет [14]:

$$i = \frac{S_{\text{п}}}{(L+B)h_{\text{расч}}} = \frac{A*B}{(L+B)h_{\text{расч}}} \quad (5.8)$$

мұндағы, S, L, B - үй-жайдың ауданы, ұзындығы және ені.

Барлық мәндерді (5.8) формулаға қойып аламыз:

$$i = \frac{A*B}{(L+B)h_{\text{расч}}} = \frac{7*5}{(7+5)*2} = 1.4$$

Д [15] арқылы шам тобын анықтаймыз – 1.

Көрсетілген әдіс бойынша есептеу кезінде бір шамның қажетті жарық ағыны мынадай формула бойынша анықталады:

$$\text{Фл} = \frac{E_{min} \cdot S \cdot Z \cdot K_3}{N \cdot \mu \cdot n}$$

мұндағы, Emin - ең аз нормаланған жарықтандыру, лк;

k - қор коэффициенті (люминесцентті шамдар үшін k=1,5);

S - жарықтандырылатын алаң, м2;

Z - ең аз жарықтандыру коэффициенті (жарықтандырудың біркелкі емес коэффициенті);

N - шамдардың саны;

μ - Шамдағы шамдардың саны; 10248

n - бірлік үлесіндегі жарық ағынын пайдалану коэффициенті.

Жасанды жарықтандыру жеткілікті ме тексеру үшін Emin есептеу керек 300 лк көп болуы керек.

$$E_{min} = \frac{\Phi_{\text{л}} * N * n * \eta_3}{S * Z * K} = \frac{3570 * 4 * 2 * 0.42}{35 * 1.1 * 1.3} = 240$$

Есеп айырысу бойынша Emin аз бізге қажет. Сондықтан бұл мәселені шешу керек. Мен көксәйек ескі люменсценғтные 65Вт шам жарық ағынымен

3570 лм жаңа люменсцентные шамдар бойынша 80 вт жарық ағынымен 4070 және олардың санын арттыру.

(5.7) формула бойынша жарықтандыруды жасау үшін шамдардың санын анықтаймыз, 4070 лм тең жарық ағынының мәнін пайдалана отырып, 300 лк жарықтандыруды құру үшін шамдардың санын анықтаймыз.

$$N = \frac{S_{\text{помещ}} K_з Z E_Н}{Ф_Л n \eta} = \frac{35*1,5*1,3*300}{4070*2*0,42} = 5,99 \approx 6 \text{ шам}$$

Бұл жағдайда, жарық ағыны мен жарық ағыны Ф = 4070 лм болатын 6 жарық шамдары бар 12 шам қажет. (5.7) формула бойынша жарықтандыруды жасау үшін шамдардың санын анықтаймыз, 4070 лм тең жарық ағынының мәнін пайдалана отырып, 300 лк жарықтандыруды жасау үшін шамдардың санын анықтаймыз.

Содан кейін тағы да тексердім E_{min}:

$$E_{min} = \frac{Ф_л * N * n * \eta_з}{S * Z * K} = \frac{4070*6*2*0.42}{35*1.5*1.3} = 300$$

Нәтижесінде біз бұл бөлмені жайлы жарықтандыру үшін жеткілікті болды.Шамдардың қуатын арттыру керек, яғни қуаты 65 Вт артық шамды таңдау керек. Осылайша, жалпы жарықтандыру шамдарында жарық ағыны 4070 лм болатын қуаты 80 Вт болатын люминесцентті ЛД шамдары таңдалған.

Жұмыс орнының үстінде шамдарды ілу биіктігін анықтаймыз [15].

$$h_{\text{расч}} = H_{\text{пом}} - (H_{\text{св}} + H_{\text{р.п.}}), \quad (5.9)$$

мұндағы, $H_{\text{св}} = 0$ – шам шамдардың шамдардың шамының биіктігі, м;

$H_{\text{р.п.}} = 0,8$ – еден үстіндегі жұмыс бетінің қашықтығы, м;

$H_{\text{пом}} = 3$ – үй-жайдың биіктігі, м.

Сонда шамдарды ілу биіктігі

$$h_{\text{расч}} = H_{\text{пом}} - \left(H_{\text{св}} + H_{\text{р.п.}}\right) = 3 - (0 + 1) = 2 \text{ м}$$

Шамдардың арасындағы қажетті қашықтықты мына формула бойынша анықтаймыз:

$$L = \lambda * h, \quad (5.9)$$

мұнда L - көрші шамдардың немесе люминесцентті шамдардың қатарлары арасындағы қашықтық;

h - жұмыс бетінен шамды ілу биіктігі.

Люминесцентті шамдары бар шамдарды пайдаланған кезде λ=0,45÷2.4 [15].

Осылайша, шамдар арасындағы қажетті қашықтық

$$L = \lambda * h = 1{,}15 * 2 = 2{,}3 \text{ м}$$

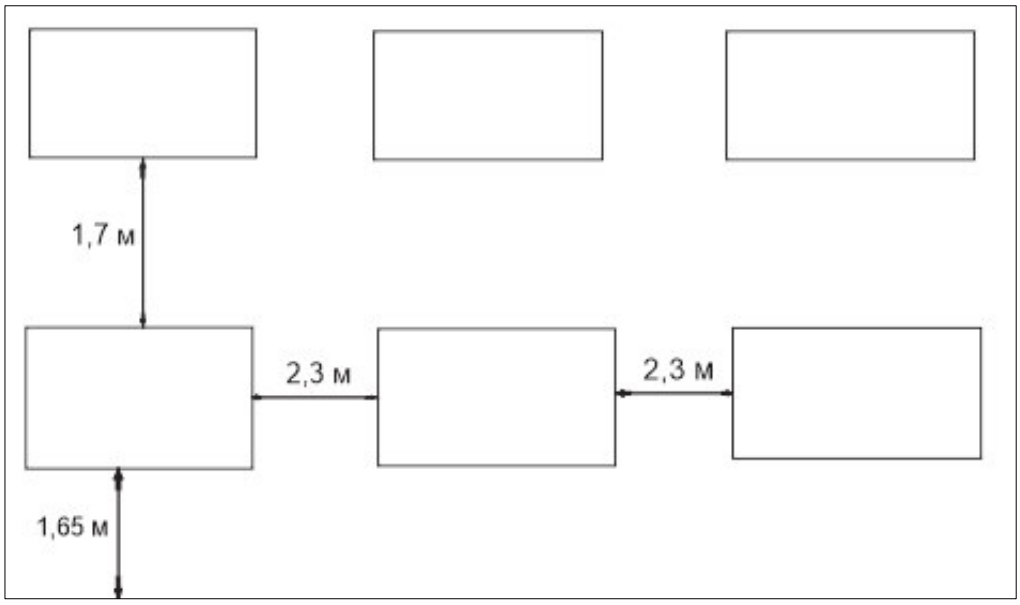
Шамдардың қатарлары арасындағы қашықтық:

$$Lb=\lambda*hp=1{,}7\cdot 1 = 1{,}7 \text{ м}$$

Шам мен қабырға арасындағы қашықтық:

$$la=lb=Lb/3 + [1{,}07;1{,}08] =1{,}7/3+1{,}08 =1{,}65 \text{ м}.$$

Жарық шамдарын және жарық ойықтарын орналастыру схемасы 5.2-суретте көрсетілген.



5.2-сурет – Бөлмедегі шамдар мен жарық ойықтарының орналасуы

5.2 Өміртіршілік қауіпсіздігі бөлімі бойынша қорытынды

Жүргізілген есептеулер нәтижесімен жасанды жарықтандыруды қайта құру жүргізілді. Жарық ағыны 3570 лм болатын 65Вт-дан ескі люминесцентті шамдарды жаңа люминесцентті шамдарға 80вт-дан жарық ағыны 4070 лм болатын жаңа люминесцентті шамдарға ауыстыру және шамдардың санын арттыру шешімі қабылданды.

Қорытынды

Дипломдық жобаның мақсаты – Android операциялық жүйесі негізінде мобильдік бағдарлама құрастыру. Мысал ретінде «Электрондық кітапхана» атты жоба оқырмандарға арналған. Зерттеулер бойынша, осы тематикаға байланысты бағдарламалар, яғни қазақша кітаптар жиыны ғаламторда көп кездесе бермейді. Сол себепті ең басты мәселе ғаламтор желісі көмегінсіз кітап оқи алу болып табылады. Android — операциялық жүйесі планшеттарға, смартфондарға орнатылады. Бағдарламаларды өңдеудегі негізгі тілдердің бірі — Java өңдеу кітапханасы. Қосымшаны жасау үшін Android Studio ортасын қолданамыз, Android SDK құрылғысы SDK эмуляторы арқылы жүзеге асады.

Дипломдық жобаны орындау барысында, заманауи мобильді технологиялардың адам өмірінің әртүрі салаларына қарқынды түрде еніп жатқаны және мобильді қосымшаларды қолдану айтарлықтай тиімді екені белгілі болды.

Нәтижесінде, интернет желісінсіз қазақ тіліндегі кітаптарды сапалы түрде қолданушыға түсінікті тілде оқи алу мүмкіндігі болып табылады.

Экономикалық бөлім бойынша, әзірленген бағдарламалық өнімнің өзіндік құны 814 056,58 теңгеге тең болып шықты.

Өміртіршілік қауіпсіздік бөлімі бойынша жүргізілген есептеулер нәтижесімен жасанды жарықтандыруды қайта құру жүргізілді. Жарық ағыны 3570 лм болатын 65Вт-дан ескі люминесцентті шамдарды жаңа люминесцентті шамдарға 80вт-дан жарық ағыны 4070 лм болатын жаңа люминесцентті шамдарға ауыстыру және шамдардың санын арттыру шешімі қабылданды.

Қорытындылай келе, «Электрондық кітапхана» оқырмандарға үйден шықпай-ақ кез келген уақытта кітап оқуға арналған. Келешекте жоба көптеген оқырмандарға көмектеседі деп сенемін.

Әдебиеттер тізімі

1 Мамаев Е. MS SQL Server. Проектирование и реализация баз данных. Сертификационный экзамен. – СПб: BHV, 2004. – 416с.

2 Голощапов А. Google Android программирование для мобильных устройств. Санк-Петербург 2011-438 с.

3 Соснов А. Основы проектирование информационных систем. – М.: ДМК Пресс, 2002. – 1020 с.

4 Марк Шпеник, Оррин Следж. Microsoft SQL Server 2000 DBA Survival Guide. – М.: Вильямс, 2001. – 236 с.

5 Отчет InMobi «Мобильные разработчики 2018»: возраст, доходы, монетизация, популярные языки»

6 Отчет Vision Mobile «Developer Economics. State of the developer nation Q1 2017» Дата обращения: 20.05.2018 г.

7 Отчет International Data Corporation «4G Smartphones to Surpass 1 Billion Mark in Shipments for 2016 as Emerging Markets Play Catch Up, According to IDC» Дата обращения: 19.05.2018 г.

8 Отчет International Data Corporation «Smartphone Volumes Expected to Rebound in 2017 with a Five-Year Growth Rate of 3.8%, Driving Annual Shipments to 1.53 Billion by 2021, According to IDC» Дата обращения: 19.05.2017 г.

9 Официальный сайт разработчиков приложений для Android. Дата обращения: 20.05.2017 г.

10 Профильный интернет-ресурс AndroidAuthority. Дата обращения: 20.05.2017 г.

11 Профильный интернет-ресурс AndroidAuthority. Статья «From Android Market to Google Play: a brief history of the Play Store»

12 Абдимуратов Ж.С., Дюсебаев М.К., Еңбекті қорғау және тіршілік қауіпсіздігінің негіздері. Дәрістер жинағы. - Алматы:АЭжБИ, 2007. - 35б.

13 Дэрси Л. Android за 24 часа. Программирование приложений под операционную систему Google/ Дэрси Л., Кондер Ш. --М.: Рид Групп, 2011. - 464 с. - (Профессиональные компьютерные книги).

14 Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. - СПб.: Питер, 2015. -1120 с.: ил. - (Серия «Классика computer science»).

15 URL адрес: https://startandroid.ru/ru/uroki/vse-uroki-spiskom/12-urok-3-sozdanie-avd-pervoe-prilozhenie-struktura-android-proekta.html