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 «MyBrain» мобильді қосымшасын әзірлеу

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

Орындаған: Чакеева А.С. Тобы: ВТк-15-1
Ғылыми жетекші: аға оқытушы Мукапил К

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

Өміртіршілік қауіпсіздігі: т.ғ.д., аға оқытушы _____ Ш.Ш. Бекбасаров
«21» 05 2019 ж.

Есептеу техникасын қолдану: аға оқытушы _____ Ж.С. Айткулов
«21» 05 2019 ж.

Норма бақылаушы: аға оқытушы _____ К. Мукапил
«20» 05 2019 ж.

Сын-пікір беруші: т.ғ.к., асс. профессор _____ Ф.Н. Абдолдина
«21» 05 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 Чакеева Ақбөпе Саматқызы

Жобаның тақырыбы: Android платформасында «MyBrain» мобильді 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 «MyBrain» мобильді қосымшасын әзірлеу. Жобаны орындау барысында Android Studio мен Java тілін 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): 7 кесте, 34 сурет ұсынылған.

Ұсынылатын негізгі әдебиеттер:

1 Жаринов, К.В. Основы веб-мастеринга: учебник / К.В. Жаринов. – СПб.: БХВ-Петербург, 2003. – 352 с.

2 Мэрдок, К. JavaScript. Наглядный курс создания динамических Webстраниц: учебник / К. Мэрдок. – М.: Вильямс, 2001. – 288 с.

3 Хомоненко, А. Д. Базы данных: учебник / А. Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев. – СПб.: КОРОНА-Век, 2009. – 736 с.

4 Ржеуцкая, С.Ю. Базы данных. Язык SQL: учеб.пособие / С.Ю. Ржеуцкая – Вологда: ВоГТУ, 2010. – 159с.

5 Холл, М. Программирование для Web. Библиотека профессионала: учебник / М. Холл, Л. Браун. – М.: Вильямс, 2002. – 1264 с.

Дипломдық жобаның бөлімдеріне қатысты белгіленген кеңес берушілер

Бөлімдер	Кеңесшілер	Мерзімі	Қолы
Экономикалық бөлім	Аренбаева Ж.Г.	04.03.2019 – 13.05.2019	
Өміртіршілік қауіпсіздігі	Бекбасаров Ш.Ш.	20.05.2019 02.05.2019	
Программалық қамтама	Айткулов Ж.С.	29.04.2019 19.05.2019	
Норма бақылау	Мукапил К.	04.04.2019 – 10.05.2019	

Дипломдық жобаны орындау
КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Талдау бөлімі	29.10.18 – 28.12.18	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» қазан 2018 ж.

Кафедра меңгерушісі _____ Т.С. Картбаев

Жобаның ғылыми жетекшісі _____ К. Мукапил

Тапсырманы орындауға алған
білім алушы _____ А.С. Чакеева

Аңдатпа

Осы дипломдық жобада «MyBrain» үшін ағылшын тілін онлайн оқыту жүйесі ұсынылған. Жобаның бірінші бөлімінде қазіргі заманның қоғамдағы ағылшын тілінің өзектілігін және бәсекелестерін талдау, салыстырмалы талдау және біздің жобаның басқа жүйелерден айырмашылығы көрсетілген.

Екінші бөлімінде жүйені жобалау үшін пайдаланылған uml диаграммалары туралы ақпарат бар.

Үшінші бөлімде, Android мобильді қосымшасын жасау үшін программалау тілдері және диаграммадағы жүйенің сипаттамасы мен жобаны практикалық іске асыруды көрсетеді.

Бесінші бөлімде жобаның тиімділігі экономикалық негіздеме қарастырылады. Яғни, жобаның қандай мөлшерде шығын алып келетіні көрсетілді.

Алтыншы бөлімде жұмысқа арналған бөлмеде жарықтың қандай жағдайда жеткіліктіі болатынын зерттеп, есептеу арқылы жазылды.

Аннотация

В данном дипломном проекте представлена система онлайн обучения английскому языку для» MyBrain". В первой части проекта отражен анализ актуальности английского языка и конкурентов в современном обществе, сравнительный анализ и отличие нашего проекта от других систем.

Во второй части содержится информация о диаграммах uml, используемых для проектирования системы.

В третьем разделе, языки программирования для создания мобильного приложения Android и описание системы на диаграмме и показывает практическую реализацию проекта.

В пятой части рассматривается экономическое обоснование эффективности проекта. То есть было показано, в каком размере будет нанесен ущерб проекту.

В шестом разделе записано, изучив, при каких условиях свет достаточен в помещении для работы.

Annotation

This diploma project presents a system of online English language training for" My Brain". The first part of the project reflects the analysis of the relevance of the English language and competitors in modern society, comparative analysis and the difference of our project from other systems.

The second part contains information about UML diagrams used for system design.

In the third section, the programming languages for creating the Android mobile application and the description of the system are shown in the diagram and shows the practical implementation of the project.

The fifth part discusses the feasibility of the project. That is, it was shown how much damage will be caused to the project.

In the sixth section, recorded, studied, under what condition

Мазмұны

Кіріспе

Осы жобада «MyBrain» үшін ағылшын тілін онлайн оқыту жүйесі ұсынылған. «MyBrain» - бұл пайдаланушыларға ыңғайлы болу үшін жасалған ағылшын тілі курстарының онлайн-порталы. Оның көмегімен клиенттер өздері үшін ыңғайлы кез келген уақытта, оқу үшін қызықты материалдармен өте қолжетімді баға бойынша оқи алады.

Жобаның бірінші бөлімінде қазіргі заманның қоғамдағы ағылшын тілінің өзектілігін және бәсекелестерін талдау, салыстырмалы талдау және жобаның басқа жүйелермен айырмашылығы көрсетілген.

Екінші тарауда жүйені жобалау үшін пайдаланылған uml диаграммалары туралы ақпарат бар.

Ал, жүзеге асыру және тестілеу бөлімінде мобильді қосымшаны жасау үшін пайдаланған технологиялар туралы ақпарат көрсетілген. Танымал платформалардың талдауы, мобильді қосымшаны жасау үшін программалау тілдері. Жобаны практикалық іске асыру да осы бөлімде көрсетілген.

1 Талдау бөлімі

1.1 Жобаның өзектілігі

Біздің өміріміз ең маңызды өлшем - оқытудан басталады. Ең басынан бастап біз үйреніп, дем алып, тамақ жеп, сөйлей бастаймыз. Біздің өмірімізде әрбір адамның ең маңызды рөлі оның білімі болып табылады. Біз ақпаратты әртүрлі жолдармен, соның ішінде өмірлік сабақтар, мектеп пен университетте алған біліміміз арқылы аламыз, бірақ бұл барлық адамдар үшін жеткіліксіз. Біз әр түрлі қосымша курстарға қатысамыз және жаңа білімге көп уақыт жұмсаймыз.

Бүгінгі күні әрбір адам осы курстарға қатыса алады. Адамдар оларға жету үшін жолға жеткілікті уақыт жұмсай алмай жатады.

Ең маңызды білімнің бірі – тіл білу. Біз өмір сүріп жатқан тілдің көмегімен, үйренеміз, сөйлесеміз. Біз әлеммен және адамдармен тіл арқылы танысамыз. Біз неғұрлым көп тілдерді білсек, соғұрлым: саяхат, оқу, өмір сүруге талпынамыз.

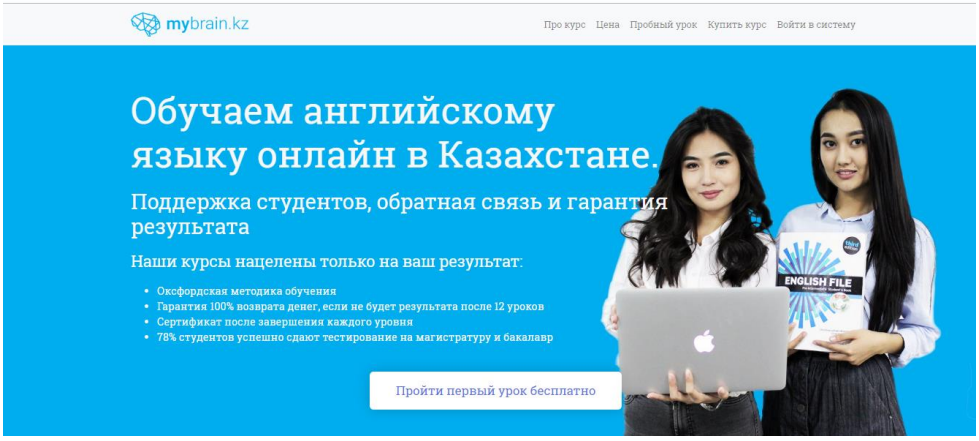
Бүгінгі күні ең танымал тілдердің бірі - ағылшын. Қазіргі әлемдегі оның маңыздылығы қарапайым сөздермен сипаттау оңай емес. Бірнеше жыл бұрын бұл тіл шет тілдерінің бірі болса, бүгін ол халықаралық тіл болып табылады. Ерте жастан бастап Ағылшын тілін білу резюмедегі ең маңызды аспектілердің бірі болып саналатыны құпия емес. Әрқайсысы ағылшын тілін кем дегенде ең төменгі деңгейде білуге және түсінуге тырысады. Барлығымыз ағылшын тіліндегі әндерді тыңдайтын болсақ, бізде саяхат жасау деген армандар бар, бірақ ең бастысы - ағылшын тілінде сөйлеу.

Технология ғасырында әрбір адам жолда уақыт өткізуге мүмкіндік алады. Бұл мәселенің шешімі қашықтан оқыту болып табылады.

Онлайн оқыту әртүрлі типтерден және деңгейден тұрады. Олардың барлығы ортақ мақсаттарға ие. Ағылшын тілін онлайн оқытудың артықшылықтары өте көп. Олардың бірі кез-келген уақытта қол жетімділік. Тек интернетке кіру мүмкіндігі бар құрылғы қажет. Қазіргі кезде, бос уақыт табу оңай емес, бірақ онлайн режимінде оқытудың ерекшелігі, кез-келген уақытта сабаққа қол жетімділік. Бұл сабақты бірнеше күнен немесе айдан кейінде қайталауға көмектеседі. Көптеген онлайн курстар сізге ақша үнемдеуге көмектеседі, себебі, компания курстарына немесе оқу орындарына қарағанда әлдеқайда арзан. Сондай-ақ, онлайн-оқытудың негізгі материалы - сабақты меңгеруге және тыңдаушыларды көбірек тартуға көмектесетін мультимедия. Жоғарыда аталғандардың бәрі онлайн оқытудың негізгі артықшылықтары болып табылады.

«MyBrain» - бұл пайдаланушыларға ыңғайлы болу үшін жасалған ағылшын тілі курстарының онлайн-порталы. Оның көмегімен клиенттер өздері үшін ыңғайлы кез келген уақытта, оқу үшін қызықты материалдармен өте қолжетімді баға бойынша оқи алады. Жоба «MyBrain» жауапкершілігі

шектеулі серіктестігінің идеясы ағылшын тілін онлайн түрде оқытудың өзектілігіне негізделген. Порталдың негізгі беті 1.1-суретте келтірілген.



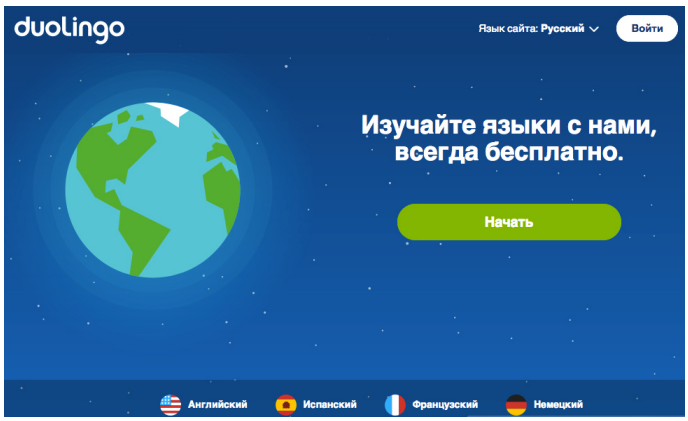
1.1-сурет – «MyBrain» білім алу жүйесінің бет-бейнесі

Ал, менің дипломдық жұмысым, осы порталдың мобильдік қосымшасын құрастыру. Мобильдік қосымшаның порталдан біршама өзгешеліктері болады. Мысалға, бірнеше сатылық тесттер.

1.2 Ұқсас жобаларды шолу. Қазіргі заманғы аналогтар

Көптеген ұқсас жобалар әзірленді, мысалы, ағылшын тілін үйренуге арналған веб-страницы, мақсаттары бірдей тілді онлайн үйренуге арналған қосымшалар. Осы бөлімде ағылшын тілін үйренуге арналған порталдар туралы мәліметтер келтіріледі.

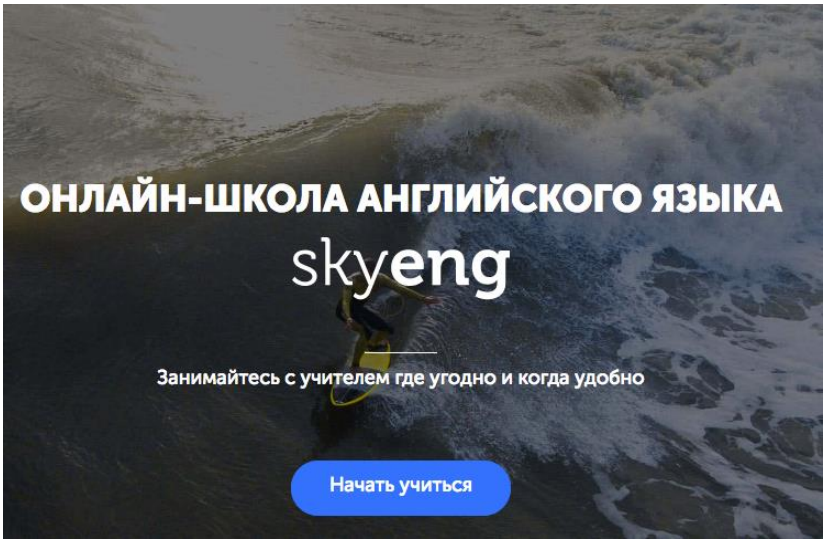
Duolingo - бұл тіл үйрену мен краудсорсингқа арналған тегін платформа. Пайдаланушылар веб-сайты, тескттіжәне басқа құжаттарды, тапсырымаларды орындаумен бір мезгілде аудара алады. 2017 жылдан бастап орыс тілін меңгергендерге , ағылшын, неміс, француз және испан тілдерін меңгеруге болады, ал итальяндық және швед тілі әлі дайындық барысында. Төменде порталдың негізгі беті 1.2-суретте көрсетілген.



1.2-сурет – «Duolingo» білім алу жүйесінің бет-бейнесі

Дуолинго жетістігі, статистикалық талдаудан негізделген, компоненттер бойынша іздестірілуде. Нью-Йорк университеті мен Оңтүстік Каролина Университетінің профессорлары жүргізген зерттеу, бұл Дуолингода 34 сағаттан аспайтын оқу, американдық университеттегі 130 сағат болатын оқу-әдістемелік семинарларға тең деген тұжырымға келді. Бірақ, зерттеу сөйлеу тәжірибесіне аса қатты мән бермеді. 196-дан 108 оқып жүргендер Duolingo-дағы оқуларын 2 аптаға жетпей тастап кетті. [1]

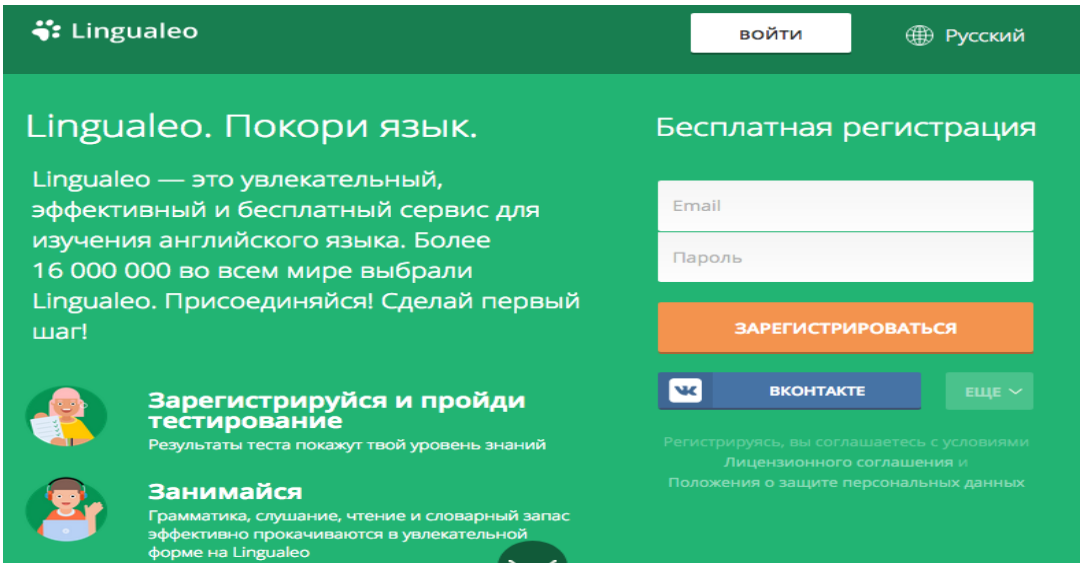
Skyeng - 2012 жылы құрылған Ресейдегі үлкен ағылшын онлайн - мектебінің бірі. Мұғаліммен сабақ жеке интерактивті платформада 24 сағат бойы жүреді және мобильді қосымшамен толықтырылған. Бұл мектепте ағылшын тілін үйрену кіріспе сабақтан басталады, сабақ барысында методист оқушының білім деңгейін, мақсаты мен қызығушылығын анықтайды. Осы мәліметтердің арқасында система әр оқушыға сәйкес келетін мұғалімді тағайындайды. Оқу аптаның әр күнінде, әр түрлі уақытта жүреді. Әр оқушыда, мобильді қосымшамен байланысқан жеке сөздігі болады. Ол арқылы сабақ уақытында өткен сөздерді енгізіп қоюға болады. Барлық мәліметтер оқушының жеке кабинетінде тұрады. [2]Келесі суретте оқу жүйесінің беткі страницасы көрсетілген.



1.3-сурет – «Skyeng» білім алу жүйесінің бет-бейнесі

LinguaLeo-бұл орыс, бразилиялық португал және түрік тілдерінің тасымалдаушылары үшін ағылшын тілін үйрену бойынша қызметтерді ұсынатын freemium онлайн-платформасы. 2015 жылдың желтоқсан айындағы жағдай бойынша бүкіл әлемде 13 миллионнан астам адам ағылшын тілін үйрену үшін оның онлайн-сервисін пайдаланған. Lingualeo интернетте, Android, iOS, Windows Phone және браузерді кеңейту ретінде қол жетімді. 2010 жылдың қарашасында компания Ангел инвесторларынан қаржыландыру түрінде $ 200 000 тартты, ал 2012 жылдың маусымында LinguaLeo Runa $ 3 млн алды.

Lingualeo ағылшын тілін оқытуды тиімді ету үшін әрбір пайдаланушыны оқыту бағдарламасын жекелендіреді. Біріншіден, Lingualeo пайдаланушыларға тілдік дағдыларын анықтау үшін орналастыру тестін ұсынады. Содан кейін сервис жеке оқыту бағдарламасын әзірлейді, ол пайдаланушының дағдысын, мақсатын және қалауын ескереді. Төменде 1.4 - суретте білім беру жүйесінің сәлемдесу беті көрсетілген.



1.4-сурет – «Lingualeo» білім алу жүйесінің бет-бейнесі

Lingualeo оқытудың жылдам дамуына арналған оқу материалдарын ұсынады. Пайдаланушылар грамматиканы үйренуге, сөздік қорын арттыруға және оқу мен тыңдау түсінігін жақсартуға мүмкіндік беретін жаттығуларды орындайды. Олар сондай-ақ бағдарлама мониторингі құралдарының көмегімен оқыту үрдісін қадағалай алады.

Сервистің анықтаушы ерекшелігі - ол әрбір пайдаланушыға өзіне ұнайтын нақты контентті таңдауға мүмкіндік береді. Тегін Lingualeo есептік жазбасы оқушыларға контент пен сервис құралдарының көп бөлігіне қол жеткізуге мүмкіндік береді. Коллекцияға жаңалықтар, ойын-сауық және іскерлік мақалалар, Ted әңгімелері, танымал әндер, бейнероликтер, тарих пен әзілдер қоса алғанда, 200 000-нан астам жеке Оқу материалдары кіреді.

Мұндай тәсіл шетел тілін үйрену кезінде туындайтын негізгі мәселені шешеді - мотивацияның болмауы. Қызмет оқушылардың қызығушылығын арттыру, көңілді және қатысуды барынша арттыру және оларды оқуды жалғастыруға шабыттандыру үшін ойын принциптерін қолданады. Лев Лев сервистің басты кейіпкері және тілдік Джунгли бойынша жеке гид болып табылады. Бұл лев фрикадельки жейді, және фрикадельки алу үшін пайдаланушы мәтіндерді оқып, бейнелерді көріп, тілдік квестерді орындап, оқу кезінде өз білімдерін бекітуі керек [5].

1.3. Android операциялық жүйесі

Android SCRUM революциялық әдіснамасы бойынша әзірлеудің арқасында ең революциялық Операциялық жүйе болды. 2014 жылдың екінші тоқсанында сатылған смартфондардың 86 % - ында Android операциялық жүйесі орнатылды. 2017 жылдың мамыр айында әзірлеушілерге арналған конференцияда Google Android тарихында 2 млрд-тан астам Android құрылғысы іске қосылғанын жариялады.

2005 жылдың 11 шілдесінде Google корпорациясы Android, Inc компаниясын сатып алды.[8] [9] 130 миллион доллар үшін. 2007 жылдың 5 қарашасында компания Open Handset Alliance (OHA) құру туралы ресми түрде жариялады және Android ашық мобильді платформасын жариялады[10], ал 2007 жылдың 12 қарашасында альянс Android "Early Look" SDK әзірлеушілеріне арналған пакеттің бірінші нұсқасын және Android эмуляторын ұсынды[8][11].

2008 жылдың 23 қыркүйегінде операциялық жүйенің бірінші нұсқасы, сонымен қатар SDK 1.0, Release 1 әзірлеушісінің бірінші толық пакеті ресми түрде шықты. Платформаның бірінші нұсқасы шыққан сәттен бастап жүйенің бірнеше жаңартулары болды. Бұл жаңартулар анықталған қателерді түзетуге және жүйеге жаңа функцияны қосуға қатысты.

2009 жылы платформаның төрт жаңаруы ұсынылды. Мысалы, ақпан айында 1.1 нұсқасы шықты, бірақ әр түрлі қателерді түзете отырып[16]. Сәуір айында және қыркүйекте тағы екі жаңарту шықты — 1.5 "cupcake"(капкейк) және 1.6 "donut"(пончик). "Cupcake" жаңартуы маңызды өзгерістерді енгізді: виртуалды пернетақта, бейне ойнату және жазу, браузер және басқалар[19]. "Donut" - де алғаш рет түрлі рұқсаттар мен экран тығыздығы мен CDMA желілерінің қолдауы пайда болды. Сол жылдың қазан айында Android 2.0 "Eclair"операциялық жүйесінің нұсқасы шықты[11] (эклер) Google бірнеше аккаунттарын қолдайтын, HTML5 тілінің браузеріне және басқа да жаңалықтарға қолдау көрсететін, сондай-ақ "Eclair" (эклер)(2) нұсқасы бойынша аздаған жаңартудан кейін "тірі тұсқағаздар" пайда болды және құлыптау экраны түрді өзгертті[11].

2010 жылдың ортасында Google "Froyo"(мұздатылған йогурт) атауымен 2.2 Android нұсқасын ұсынды, ал 2010 жылдың соңында — Android 2.3 "Gingerbread"(зімбір пряник). "Froyo" жаңартылғаннан кейін смартфонды кіру нүктесі ретінде пайдалануға, смартфонды сандық немесе әріптік-сандық парольмен дәстүрлі бұғаттауды және басқа да өзгерістерді пайдалануға болады, ал "Gingerbread" (зімбір пряник) жаңартуы көшіру мен кірістіру функциясын толық бақылауға, бағдарламаларды басқару мен бақылауды жақсартуға, құрылғыдағы бірнеше камераларды қолдауға және т. б. әкелді.

2011 жылдың 22 ақпанында Android 3.0 "Honeycomb" (бал ұясы) интернет-планшетіне бағытталған платформасы ресми түрде ұсынылды. Бұл нұсқаның бастапқы коды Google компаниясы оны смартфондарға тасымалдау қаупіне байланысты ашылмады.

Android 4.0" Ice Cream Sandwich " (балмұздақ сэндвич), 2011 жылдың 19 қазанында шыққан— планшеттерге және смартфондарға арналған алғашқы әмбебап платформа. Сондай-ақ, жаңарту Android 4.4.4 KitKat-ға қолданылған жаңа "Holo" интерфейсін әкелді .

2012 жылдың маусым айында 4.1 реттік нөмірі бар "Jelly Bean"(желейный боб) атты жаңарту шықты, ол сол жылдың қазан айының соңында аздаған жаңарту салдарынан 4.2-ке және 2013 жылдың шілде айында жаңартылғаннан кейін 4.3-ке ауысты.

2013 жылдың 31 қазанында Google Android 4.4 операциялық жүйесінің келесі нұсқасын ұсынды, ол Nestlé өндірушісімен келісім бойынша шоколад батончигі "KitKat" деп аталды[32]. Алғаш рет KitKat Nexus 5-те пайда болды; бұл Android нұсқасы 512 МБ RAM және 800x480 пикс ажыратымдылығы бар экраны бар құрылғылардың кең жиынтығында жұмыс істеу үшін оңтайландырылған. ұсынылатын минимум ретінде. Сондай-ақ, сынақ опциясы ретінде, әзірлеушінің параметрлерінде ART виртуалды машинасы қол жетімді болды.

25 Маусым 2014 Google Nexus смартфондарын және басқа да кейбір смартфондарды әзірлеушілер, пайдаланушылар үшін қолжетімді Android L ұсынды[33].

2014 жылдың 15 қазанында ресми түрде Android 5.0 Lollipop (леденец) жарияланды. Жүйенің негізгі жаңартулары-Жаңа Material Design дизайны және ART виртуалды машинасына толық көшу. Сондай-ақ, Егер Android құрылғысында құпия сөз немесе графикалық кілт орнатылған болса және жақын жерде Android Wear бар құрылғы иесінің сағаты болса, құрылғы автоматты түрде құлыпталады.

9 желтоқсан 2014 Google орына ресми ортасын әзірлеу, негізделген Eclipse (adt-bundle), Android Studio[6][5].

2015 жылы Android Wear (кейінірек Wear OS) қолданылатын құрылғыларына арналған Операциялық жүйе жарияланды. Сондай-ақ, Google IO-да Android Auto (автомобильдер үшін) және Android TV (теледидарлар үшін) нұсқалары ұсынылды, осылайша Android тек ұялы құрылғыларға арналған Операциялық жүйе болудан қалды.

29 мамыр 2015 Google — дің айтуынша, жаңа операциялық жүйенің басты мақсаты-смартфонмен қарым-қатынастың пайдаланушылық тәжірибесін жақсарту, өзара әрекеттестікті интуитивті және оңай ету[36]. 17 Тамыз 2015 Бұл нұсқа Android 6.0 Marshmallow деп аталды.

2016 жылдың 19 мамырында Google I/O әзірлеушілері конференциясында Android N ұсынылды. Операциялық жүйенің соңғы нұсқасын тарату 22 Тамыз 2016 жылы басталды. Қолдау: Huawei Nexus 6P, LG Nexus 5X, Motorola Nexus 6, HTC Nexus 9, ASUS Nexus Player, Google Pixel C және General Mobile 4G.[9]

2017 жылдың наурыз айында әзірлеушілер үшін Android O ұсынылды. Пайдаланушы нұсқасы 21 Тамыз 2017 жылы 8.0.0 Oreo ретінде шығарылды. 5 желтоқсан 2017 Android O 8.1 тұрақты жинау шығарылды.

2018 жылдың 7 наурызында Google Android P DP1 шығарды, ал бірінші бета 2018 жылдың 8 мамырында шықты, бірақ тек Pixel үшін ғана емес, Sony Xperia XZ2, Nokia 7 Plus, Xiaomi Mi MIX 2S, Vivo X21, Oppo R15 Pro, Essential Phone, OnePlus 6. Android 9 Pie нұсқасының таратылуы 2018 жылдың 6 тамызында басталды.

2019 жылы 13 наурызда Google барлық ұрпақ Pixel смартфондарына қол жетімді Android Q Beta 1 ОЖ ашық бета-сынағын бастады. Барлығы компания 6 бета нұсқаларын шығарады. 3 сәуір 2019 жыл Android Q Beta 2, Pixel смартфондарынан басқа GSI-бейнелер түрінде пайда болды. Олардың көмегімен, Project Treble бағдарламасын қолдайтын кез келген құрылғының әзірлеушілері ОЖ-ның осы нұсқасын өз бағдарламаларымен танысу және тестілеу үшін қоя алады. Сондай-ақ, Google iPhone X сияқты қимылдарды сынауды бастады, бағдарлама өзіне кірмей, қалқымалы терезелерде мессенджерлер жауаптар, перде медиа ойнатуды басқару және жаңа дыбыс басқару.

Android-бұл түрлі мобильді құрылғыларға арналған ең танымал және ең перспективалы операциялық жүйелердің бірі. Бұл смартфонның немесе планшеттің әрбір пайдаланушысына оны толығымен өз қажеттіліктеріне теңшеуге мүмкіндік береді. Бұл операциялық жүйені әзірлеу алыс 2003 жылы басталды, бірақ ол Google компаниясы сатып алғаннан кейін 2 жылдан кейін ғана белгілі болды. Android тарихындағы өзгерістер 2008 жылдың күзінде болды. Ол кезде Google компаниясы Android басқаруындағы алғашқы смартфон болған T-Mobile G1 көрсетті. Дәл сол сәтте көптеген әлемдік өндірушілер перспективалық операциялық жүйеге назар аударды.

1.5-сурет – Ең алғашқы Android операциялық жүйесі.

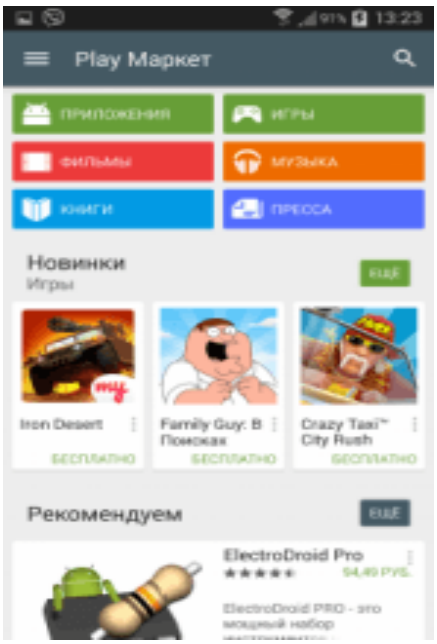
Қарастырылып отырған OS әрқашан өз әзірлеушілері ашық кодпен жүйе ретінде жайғасты. Бұл кез келген тілек білдірушіге Android-гаджеттер мүмкіндіктерін кеңейту үшін өз қосымшаларын, ойындарын және басқа да қосымшаларды жасауға мүмкіндік береді. Әзірлеушілер бастапқыда Операциялық жүйе ең "Бюджеттік" темірде барынша тез жұмыс істеуі үшін бәрін ойластырды. Бұл сөзсіз артықшылығы болып табылады, өйткені енді ең қарапайым қаржылық мүмкіндіктері бар адамдар да заманауи смартфондардың барлық негізгі артықшылықтарын пайдалана алады.

1.6-сурет – Android 5.0 операциялық жүйесі.

Жүйенің ашықтығын ұялы электроника өндірушілері де пайдаланады, мысалы, HTC компаниясынан Sense өзіндік пайдаланушы интерфейстерін шығару. Бұл түрлі өндірушілердің Android гаджеттер бір-біріне ұқсамайтын етеді. Классикалық және "таза" Android әуесқойлары Nexus құрылғыларына назар аудару керек. Олар дәстүрлі түрде бірінші болып жаңартуларды алады.

Басқа құрылғыларды жаңарту мерзімі әдетте ОЖ жаңа нұсқаларының ерекшеліктеріне сәйкес фирмалық қабықшаларды пысықтау қажеттігіне байланысты созылады.

Ойындар мен қосымшалардың басты көзі-Play Маркет. Осы дүкеннің каталогтарында тегін және ақылы контенттің үлкен таңдауы бар, оның ассортименті белсенді түрде кеңейіп келеді. "Android" ойындары мен қосымшаларының сапасы үнемі өсіп келеді. Әр түрлі қолданбаларды Google өзі де шығарады.



1.7-сурет – Play Маркет

Жалпы, Android құрылғысында жасырын талаптар бойынша заманауи смартфондар мен планшеттер болуы тиіс барлық функциялар бар. Бұдан басқа, функционал қосымшалар, виджеттер немесе сыртқы тігістер арқылы оңай кеңейтіледі.

Android смартфонын сатып алғысы келетіндерге көптеген функциялар мен бағдарламалар интернетпен жұмыс істеуге бағытталғанын есте сақтау қажет. Wi-Fi рұқсаты болмаса, интернет пайдаланушыларына тиімді тарифті қосу немесе желіге шығуды талап ететін кейбір мүмкіндіктерді өшіру ұсынылады.

Сондай-ақ, Android құрылғысының шектеулі дербестігін, әсіресе алдыңғы ұрпақ өкілдерін атап өтпеуге болмайды. Смартфонды немесе планшетті белсенді пайдаланған кезде, күнделікті, ал кейбір жағдайларда тәулігіне 2 реттен зарядтау керек. Өндірушілер осы кемшілікті түзетумен белсенді жұмыс істейді және соңғы уақытта нарықта едәуір ұлғайған автономды қызықты құрылғылар пайда бола бастады. Әзірлеушілер де шет қалмайды. Әрбір жаңа нұсқамен Операциялық жүйе" жейді " аз заряд.

Келтірілген ақпараттың негізінде қарастырылатын операциялық жүйенің негізгі артықшылықтары мен кемшіліктерін бөліп көрсетуге болады. Сондықтан, Android артықшылықтары:

– бағдарламалық кодтың ашықтығы, бұл кез келген бағдарламалар мен ойындарды әзірлеуге мүмкіндік береді;

– құрылғының "темірге" талап етілмейді;

– кез келген талғамға арналған қосымшалар мен қызықты ойындардың үлкен ассортименті;

– смартфонның функционалы туралы заманауи ұғымдарға толық сәйкес келу;

– пайдаланушы үшін әрекет еркіндігі. Қаласаңыз, жүйе және оның интерфейсі иесінің қажеттіліктеріне оңай теңшеледі.

– операциялық жүйенің танымалдығы қазіргі заманғы өндірушілердің басым көпшілігі арасында. Бұл пайдаланушыға кез келген бюджетпен функционалды және заманауи смартфонды сатып алуға мүмкіндік береді.

Android операциялық жүйесінің кемшіліктеріде бар. Олар:

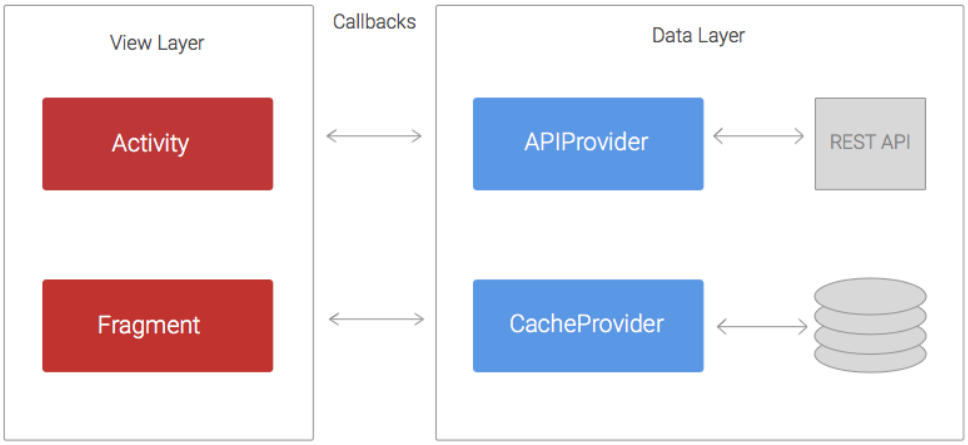
– бағдарламалық кодтың ашықтығы. Иә, оны бір мезгілде артықшылықтарға да, кемшіліктерге де жатқызуға болады. Бұл кемшілік ұялы құрылғыларды өндірушілер өз қабықтарын жасауды қалайды себебі болып табылады. Осыған байланысты ОЖ жаңартудың ресми шығуы мен оны әртүрлі құрылғыларға алу арасында елеулі уақытша кідірістер пайда болады.

– салыстырмалы төмен автономдық. Соңғы уақытта бұл бағытта алға үлкен қадам жасалды, сондықтан жақында "Android смартфондары ең спорттық, өйткені олардың күніне 2 рет жаттығу" қызықты тарихқа айналатыны туралы әзілдер болды.

– платформа үнемі жетілдіріліп, дамып келеді. Бүгінде ол операциялық жүйелер нарығындағы көшбасшылардың үштігінде өзін сенімді сезінеді. Көптеген сарапшылар Болашақ дәл осы Android үшін қателеспеді деп бірнеше рет атап өтті.

1.4 Android мобильді жүйесінің архитектурасы

2012 жылдары жобалардың құрылымы өте қарапайым болды. Желімен жұмыс істеуге арналған кітапханалар жоқ, AsyncTask арқылы істелінді. Төменде берілген диаграмма сол шешімдердің үлгі архитектурасын көрсетеді:



1.8-сурет – Шешімдердің үлгі архитектурасы

Код екі деңгейге бөлінді: REST API арқылы, сондай-ақ әртүрлі Жергілікті сақтау орындары арқылы алынған деректерді алу/сақтау үшін

жауап беретін деректер деңгейі (data layer) және деректерді өңдеу және көрсету үшін жауап беретін көрініс деңгейі (view layer).

API Provider белсенді және фрагменттерге REST API-мен өзара әрекеттесуге мүмкіндік беретін әдістерді ұсынады. Бұл әдістер фондық ағында сұрау жасау үшін URLConnection және AsyncTask қолданады,содан кейін нәтижелерді кері қоңырау функциясы арқылы іске қосады. Сол сияқты жұмыс істейді және CacheProvider: SharedPreferences немесе SQLite деректерді алатын әдістер бар және нәтиже қайтаратын кері қоңырау функциялары бар.

Мұндай тәсілдің басты мәселесі-бұл көрініс деңгейі тым көп жауапкершілік. Қолданба блогтан постылар тізімін жүктеуге, оларды SQLite-те араластыруға, содан кейін ListView-де көрсетуге тиіс қарапайым сценарийді ұсынайық. Activity мынаны жасауы керек:

– API Provider әдісін шақыру#load Posts(Callback);

– берілген callback ' е-де onSuccess() әдісін шақыруды күтіңіз, содан кейін CacheProvider#savePosts(Callback) шақыру;

– берілген callback ' е-де onSuccess() әдісін шақыруды күту, содан кейін ListView-де деректерді көрсету;

– APIProvider және CacheProvider пайда болуы мүмкін екі ықтимал қателерді бөлек өңдеңіз.

Және бұл қарапайым мысал. Нақты өмірде API деректерді біздің көрініс деңгейін күтетін түрде емес, сонымен қатар Activity олармен жұмыс істей алмастан бұрын деректерді өзгертуі және/немесе сүзуі тиіс болатындай болуы мүмкін. Немесе, мысалы, loadPosts () бір жерден алу керек аргументті қабылдайды (мысалы, Play Services SDK арқылы сұрайтын электрондық пошта мекенжайы). SDK адресін кері шақыру функциясы арқылы асинхронды қайтарады, яғни қазір бізде кері шақыру функциясының үш деңгейі бар. Егер біз көп қиындықтарды бұруды жалғастыратын болсақ, нәтижесінде callback hell деп аталатын нәрсе аламыз.

Активити және фрагменттерді қолдану қиын болады.

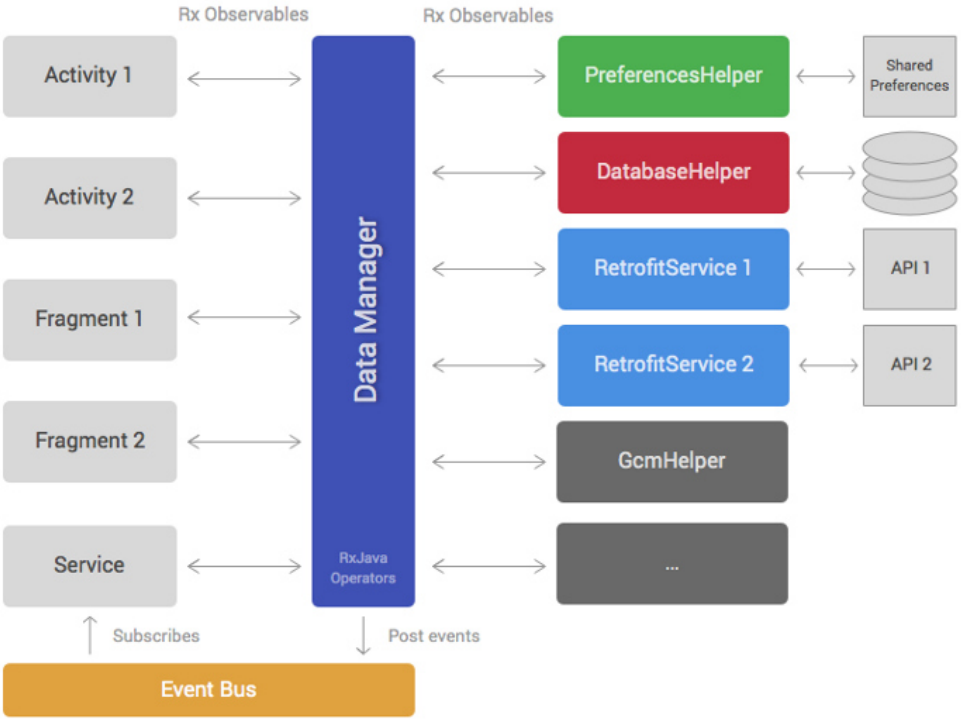
Салыным деңгейі тым көп код жаңа функционалдың қосылуын немесе өзгерістер енгізуді күрделендіруге әкеп соқтырады және түсіну үшін қол жетімсіз болады.

Юнит-тестілеу қиындайды (егер мүлдем мүмкін болмаса), өйткені көптеген логиктер юнит-тестілеуге өте көп емес активитерде немесе фрагменттерде болады.

Жоғарыда сипатталған тәсіл 2014 жылға дейін пайдаланылды. Осы уақыт ішінде біз сипатталған проблемалардан ауырсыну мен қиындықты жеңілдететін бірнеше өзгерістер енгіздік. Мысалы, біз бірнеше қосымша сыныптарды қосып, активит пен фрагменттерді түсіру үшін Логиканың бір бөлігін шығарды, сондай-ақ біз APIProvider volley пайдалана бастады. Осы өзгерістерге қарамастан, код әлі де тест қиын болды, және callback-hell мезгіл сол жерде, содан кейін онда серпіліс болды.

Біз RxJavaбірнеше сынақ жобаларында сынап көрдінді, және кері шақырудың салынған функциялары мәселесін шешу табылған сияқты екенін түсіндірілді. Егер қысқа болса, RxJava асинхронды ағындар арқылы сіздің деректеріңізді басқаруға мүмкіндік береді. Бұл жағдайда streams сияқты ағындарды білдіреді, threads-орындау ағындарымен шатастырмаңыз) және түрлендіру, сүзу, немесе сізге қажет сияқты деректерді біріктіру үшін ағындарға қолдануға болатын көптеген операторларды ұсынады.

Екі жылдың ішінде жинаған барлық бүрлерді назарға ала отырып, жаңа қосымшаның архитектурасын ойластыра басталды және келесіге келді:



1.9-сурет – Мобильді қосымшаның архитектурасы

Код сол бұрынғыдай екі деңгейге бөлінген: деректер деңгейі DataManager және көмекшілер сыныптарының жиынтығын қамтиды, ұсыну деңгейі Activity, Fragment, ViewGroup және т.б. сияқты Android SDK сыныптарынан тұрады.

Көмекші сыныптар (диаграммадағы үшінші баған) жауапкершіліктің өте шектеулі салалары болады және оларды дәйекті түрде іске асырады. Мысалы, көптеген жобалар REST API қол жеткізу кластары бар, дБ-дан деректерді оқу немесе бөгде өндірушілердің SDK-мен өзара іс-қимыл. Әр түрлі қосымшаларда әртүрлі көмекші сыныптар жиынтығы болады, бірақ ең жиі қолданылатын болады:

– PreferencesHelper: SharedPreferences деректерімен жұмыс істейді;
– DatabaseHelper: SQLite жұмыс істейді;
– REST API-ге өтініштерді орындайтын Retrofit сервистері.

Volleyорнына Retrofit пайдалана бастады, өйткені ол RxJava жұмыс қолдайды. Көптеген көпшілік көмекшілер әдістері RxJava Observables қайтарады.

Data Manager-Жаңа архитектураның орталық бөлігі. Ол көмекшілерден алынған деректерді біріктіру, сүзу және түрлендіру үшін RxJava операторларын кеңінен қолданады. Data Manager міндеті активті және фрагменттерді деректерді "тарау" жұмыстарынан босату болып табылады — ол өзінің ішінде барлық қажетті трансформацияларды шығарады және көрсетуге дайын деректерді сыртқа береді.

Төменде код DataManager әдісі қалай көрінуі мүмкін көрсетеді. Ол келесідей жұмыс істейді:

– тізімді Retrofit арқылы жүктейді;

– DatabaseHelper арқылы жергілікті деректер базасына деректерді кэштейді;

– бүгінгі күні жарияланғанын таңдап, постыларды сүзеді, өйткені көріністің деңгейі тек оларды көрсету керек.

```
public Observable<Post> loadTodayPosts()
{ return mRetrofitService.loadPosts()
.concatMap(new Func1<List<Post>, Observable<Post>>() {
@Override
public Observable<Post> call(List<Post> apiPosts) {
return mDatabaseHelper.savePosts(apiPosts);
}
})
.filter(new Func1<Post, Boolean>() {
@Override
public Boolean call(Post post) {
return isToday(post.date);
}
});
}
```

Көрініс деңгейіндегі компоненттер тек осы әдісті тудырып, қайтарылған Observable-ге жазылатын болады. Жазылым аяқталысымен, алынған Observable қайтарылған посттар оларды RecyclerView немесе ұқсас нәрсе көрсету үшін Adapter-ге қосылуы мүмкін.

Бұл архитектураның соңғы элементі-eventbus. Event bus деректер деңгейінде болатын оқиғалар туралы хабарларды іске қосуға мүмкіндік береді, ал көрініс деңгейіндегі компоненттер осы хабарламаларға қол қойылуы мүмкін. Мысалы, signOut әдісі () DataManager тиісті Observable өз жұмысын аяқтағанын хабарлайтын хабарды іске қоса алады, содан кейін осы оқиғаға қол қойылған активити пайдаланушы жүйеден шыққанын көрсету үшін өз интерфейсін қайта суреттей алады.

Бұл тәсіл не себепті жақсы?

– Observables және RxJava операторлары бізді кері қоңырау функциясынан құтқарады;

– Data Manager бұрын көрініс деңгейінде орындалған жұмысты өзіне алады, осылайша іске қосу және фрагменттерді жүктейді;

– код бөлігін DataManager-ге жылжыту және көмекшілер сыныптар юнит-тестілеу активити және фрагменттерді оңай етеді;

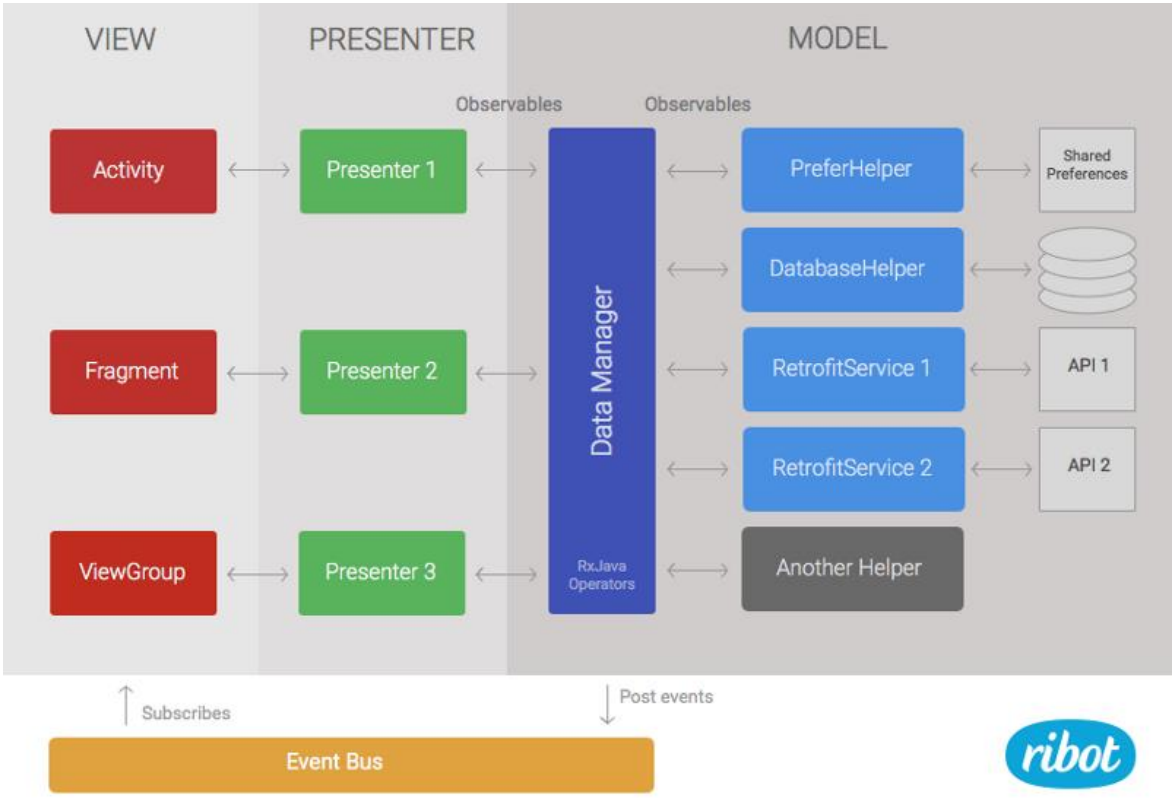
– нақты жауапкершілікті бөлу және datamanager деректер деңгейімен өзара іс-қимыл жасаудың жалғыз нүктесі ретінде таңдау тестілеу неғұрлым Достық барлық сәулет етеді. Сынып-көмекшілер, немесе DataManager, арнайы қақпақтарға оңай ауыстыруға болады.

Ал қандай проблемалар қалды?

– үлкен және күрделі жобаларда DataManager тым көп болуы мүмкін және оны қолдау қиын;

– біз көріну деңгейінің компоненттерін (активити және фрагменттер сияқты) оңай жасадық, олар әлі де RxJava жазылымдарын басқару, қателерді талдау, және т.б. сияқты логика бар.

2014 жылдары Android-қоғамдастығында MVP, немесе MVVM сияқты жеке сәулет үлгілері танымал бола бастады. Сынақ жобасы, сондай-ақ жеке мақалада осы үлгілерді зерттеуден кейін, біз MVP жобаларымыздың архитектурасына маңызды өзгерістер әкелуі мүмкін деп тапты. Біз кодты екі деңгейге бөлгендіктен (деректер мен көріністер), MVP енгізу табиғи көрінді. Біз жай ғана жаңа presenter ' s деңгейін қосу керек болды, және оған көріністерден код бөлігін көшіру.



1.10-сурет – Мобильді қосымшаның архитектурасы

Деректер деңгейі өзгеріссіз қалады, бірақ қазір ол MVP-дан тиісті деңгейдің атауына сәйкес үлгі деп аталады.

Presenter ' ы деректерді үлгіден жүктеуге және деректер жүктелген кезде ұсыну деңгейінде тиісті әдістерді шақыруға жауап береді. Presenter ' ы Datamanager қайтарылатын Observables-ке қол қойылады. Демек, олар жазылымдар мен жоспарлаушылар сияқты мәндермен жұмыс істеу керек. Сонымен қатар, олар пайда болған қателерді талдай алады немесе қажет болса, деректер ағынына қосымша операторлар қолдана алады. Мысалы, егер біз кейбір деректерді сүзу керек болса, және бұл сүзгі еш жерде пайдаланылмайды, бұл сүзгіні DataManager емес, presenter ' а деңгейіне шығару мағынасы бар.

Төменде presenter а деңгейінде болуы мүмкін әдістердің бірі ұсынылған. Мұнда Datamanager әдісімен қайтарылатын Observable жазылымы бар.loadTodayPosts (), біз алдыңғы кодта анықтады.

```
public void loadTodayPosts() {
mMvpView.showProgressIndicator(true);
mSubscription = mDataManager.loadTodayPosts().toList()
.observeOn(AndroidSchedulers.mainThread())
.subscribeOn(Schedulers.io())
.subscribe(new Subscriber<List<Post>>() {
@Override
public void onCompleted() {
MvpView.showProgressIndicator(false);
}
@Override public void onError(Throwable e)
{
mMvpView.showProgressIndicator(false); mMvpView.showError();
}
@Override public void onNext(List<Post> postsList) {
mMvpView.showPosts(postsList);
}
});
}
```

mMvpView-бұл presenter жұмыс істейтін көрініс деңгейінің компоненті. Әдетте бұл Activity, Fragment немесе ViewGroup болады.

Алдыңғы архитектура сияқты, көрініс деңгейі Android SDK стандартты компоненттерден тұрады. Айырмашылық қазір бұл компоненттерге Observables тікелей қол қойылмайды. Оның орнына, олар Mapview интерфейсін имплементациялайды және showError() немесе showProgressIndicator () сияқты түсініксіз және түсінікті әдістердің тізімін ұсынады. Ұсыну деңгейінің компоненттері сондай-ақ пайдаланушымен өзара іс-қимылды өңдеуге (мысалы, басу оқиғалары) және presenter ' Е-ге сәйкес әдістерді шақыруға жауап береді. Мысалы, біз постылар тізімін жүктейтін Түйме болса, біздің Activity Onclicklistener ' е presenter әдісін шақыруы керек.loadTodayPosts().

Бұл тәсіл не себепті жақсы?

– белсенді және фрагменттер одан да оңай болады, өйткені олардың жұмысы енді пайдаланушы интерфейсін суреттеуге/жаңартуға және пайдаланушымен өзара іс-қимыл жасау оқиғаларын өңдеуге түседі. Осылайша, оларды қолдау оңай болады.

– Presenter ' ларға арналған юнит-тестерді жазу өте оңай көріністің деңгейін жасыру керек. Бұрын бұл код ұсыну деңгейінің бір бөлігі болды және оның юнит-тестілеуін өткізу мүмкін болмады. Сәулет одан да көп тестіленеді;

– егер DataManager тым тітіркендіргіш болса, біз әрқашан кодтың бөлігін presenter ' s-ке көшіре аламыз.

Ал қандай проблемалар қалды?

– Data Manager кодтың саны көп болған жағдайда да тым ашулы (қиын) болуы мүмкін.

1.5 Android мобильді қосышаны құру процесі

Мобильді қосымша (ағылш. "Mobile app") — смартфондарда, планшеттерде және басқа да мобильді құрылғыларда жұмыс істеуге арналған бағдарламалық қамтамасыз ету. Көптеген мобильді бағдарламалар құрылғыда алдын ала орнатылған немесе App Store, BlackBerry App World, Google Play, 1mobile market, Windows Phone Store, Яндекс сияқты онлайн дүкендерден жүктеп алуға болады.store және басқалар, тегін немесе ақы.

Бастапқыда мобильді қосымшалар электрондық поштаны жылдам тексеру үшін пайдаланылды, бірақ олардың жоғары сұранысы олардың мақсаттарының кеңеюіне және ұялы телефондар мен GPS ойындары сияқты басқа да салаларда, қарым-қатынас, бейне көру және интернетті пайдалану сияқты басқа да салаларда әкелді.

Мобильді қосымшалар нарығы бүгін өте дамыған[қалай?] және ұдайы өсуде.

App Annie аналитикалық компаниясының есебіне сәйкес, 2017 жылы мобильді қосымшалардың әлемдік нарығы 28,6% – ға өсіп, $166 млрд.жетеді деп күтілуде, оның $65 млрд. – қосымшалар мен жазылымдарды сатып алу шығындары, ал $101 млрд. - жарнама берушілер мен қосымшаларды әзірлеушілердің жылжыту шығындары.

App Annie нарықтарын талдау жөніндегі Директор Самир Синха 2016 жылы пайдаланушылар $52 млрд, жарнама берушілер – $77 млрд. қосымшаларға жұмсайды. Табыстың өсуі әлеуметтік желілер (бірінші кезекте Facebook), ағындық бейне (YouTube және т.б.) тегін тұғырнамалары және ұялы ойындарға енгізілген жарнаманың арқасында болды. Ең танымал формат — бейнетіркеуіш. Ірі брендтердің жарнамасы барлық орналастырудың 12,5% құрайды.

Бұл термин 2007 жылдан бері өте танымал болды және 2010 жылы американдық диалектикалық қоғамның "жыл сөздері" тізіміне енгізілді.

2011 жылы Дүниежүзілік денсаулық сақтау ұйымы mHealth мобильді медициналық технологияларының жарылысын және олардың әлеуетін бүкіл әлем бойынша денсаулық сақтау жүйесінің бейнесін тез өзгерту мүмкіндігін болжады. 2016 жылға қарай әлемде 260 мың медициналық мобильді қосымшалар болды. 2016 жылы Еуропалық комиссия жанында медициналық

мобильді қосымшалар мен құрылғыларды сараптау ережелерін әзірлеу бойынша жұмыс тобын құрды. Смартфон қосымшалардың көмегімен өкпе функциясының негізгі параметрлерін өлшеуге болады (бастапқы деректерді алу үшін микрофонға дем алу керек), жүрек жиілігін өлшеуге, микрофлюидтік қосымшалардың көмегімен қанға немесе ағзаның басқа сұйықтықтарына талдау жасауға болады. Оны офтальмоскоп ретінде қолдануға болады,ал саптаманың көмегімен отоскопқа айналдыруға болады.

XXI ғасыр-мобильді технологиялардың гүлдену дәуірі. Қазір тіпті ұялы құрылғысыз адамды көрсету қиын, ал адам-Бизнес одан да көп. Мобильді технологиялар бизнестің барлық салаларында бар. Мобильді экожүйелер күн сайын өзгеріп отырады және тұрақты эксперимент негізінде дамиды. Осы қосымшалардың әрқайсысында қосымшаның бағыты мен контекстіне байланысты нақты көрнекі стиль мен тона бар.

Мобильді қосымшаларды әзірлеу процесі неден тұрады?

– идея

Бұл, әрине, идея. Бастапқы кезеңде болашақ мобильді қосымшаның мағынасын мұқият ойластыру қажет және ол не үшін пайдаланылады. Содан кейін ол қандай платформада пайдаланылатынын анықтау қажет. Әдетте, қазір қосымшалар бірден екі iOS және Android платформасы үшін бір уақытта жазылады, бірақ егер бұл еш нәрсе болмаса, не асығыс емес болса, онда компания мамандары сұраныс/ұсыныс және бәсекелестік нарығын алдын ала талдай отырып, платформалардың бірінің пайдасына таңдау жасауға көмектеседі.

– техникалық тапсырма

Әзірлеудің алдында тапсырыс берушіден ТТ алу қажет. Егер ол жоқ болса, онда Тапсырыс берушіге толтыру үшін бриф беріледі. Бұл кезеңге ерекше көңіл бөлінеді, өйткені ТТ нәтиженің техникалық ерекшеліктеріне тікелей әсер етеді. Осы кезеңде осындай жұмыс түрлері орындалады:

Мобильді қосымша функционалының сипаттамасы жасалады;

Әзірлеу мерзімдері анықталады және келісіледі;

Қаржылық шығындар есептеледі және есеп айырысу тәртібінің моделі әзірленеді;

Тапсырыс берушімен шарт ресімделеді;

– прототиптеу

Пайдаланушы мобильді қосымшамен қалай жұмыс істейтінін түсіну үшін бағдарламаның әр түрлі экрандары арасында өзара іс-қимылдың графикалық картасы жасалады. Бұл кезеңде мобильді қосымшаның барлық функционалын пысықтау жүзеге асырылады. UI жобалау сатысында мамандар қосымшаның жұмыс істеу принципін, әр экранда функциялар мен түймелердің орналасуын анықтайды. Бұл кезеңде:

- Қосымшаның функционалы өңделеді;
- Қолданба экрандарының сызбалары әзірленеді;
- Қолданба экрандарын;
- байланысы және олар бойынша ауысу ойластырылады;

– мобильді қосымшаның дизайны

Болашақ қолданбаның барлық экрандарының дизайны жасалады және барлық пайдалану сценарийлері үшін әр түрлі күйлер суреттеледі. Дизайн тұжырымдамасы бекітілгеннен кейін ішкі түймелер мен белгішелер, сондай-ақ барлық басқа графикалық элементтер бейнеленеді. Әдетте, мобильді қосымшаның дизайнын суреттеу юзабилити-зерттеулерді жобаланған дизайн-концепция барынша қарапайым және ыңғайлы екеніне көз жеткізу үшін және пайдаланушыларға қойылған міндеттерді барынша тез шешуге көмектеседі.

– әзірлеу

Әзірлеушілерге ТТ және қосымша дизайнының макеттері беріледі және олар "жасайды"бастайды. Бағдарламашылар статикалық суретті интерактивті жұмыс моделіне айналдырады. Қолданбаның бірінші нұсқасы шығарылады.

– тестілеу

Мобильді қосымша мұқият тестілеуден өтеді, соның нәтижесінде тексеру кестесі құрылады, онда барлық қателер көрсетіледі және егжей-тегжейлі сипатталады. Қосымшаларды жобалау барысында нақты пайдаланудың барлық қателіктерін қарастыру мүмкін емес. Бұл кезеңде қателердің, кемшіліктердің және қосымшалардың кемшіліктерінің тізбесі қалыптасады және оларды жою мерзімдері анықталады. Содан кейін түзетілген қателері бар және қажет болған жағдайда өзгертілген функционалы бар қосымша шығарылады, бұл сынамалы тестілеуден кейін кестеде көрсетіледі.

– қайта тестілеу

Қолданба тест құрылғыларына орнатылады және Google Play немесе AppStore жүктелген сияқты дәл жұмыс істейді. Қолданба бағдарламалардың ресми каталогында пайда болардың алдында, қолданушылардың бағдарламаны орнату және қолдану барысында әзірлеудің қателіктеріне таппайтынына көз жеткізу қажет. Сондықтан осы кезеңде өнімнің логикасы, оның серверлік бөлігінің жұмысы тағы да тексеріледі, қосымша әр түрлі жағдайларда және операциялық жүйелердің әр түрлі нұсқаларында тестіленеді.

2 Жобалау бөлімі

2.1 Унифицирленген модельдеу тілі. UML

Модельдеудің біріздендірілген тілі (UML) ақпараттық жүйелердің (АЖ) "сызбаларын" құруға арналған стандартты құрал болып табылады. UML көмегімен осы жүйелердің элементтерін визуализациялауға, сипаттауға, құрастыруға және құжаттауға болады.

UML кез-келген жүйелерді модельдеу үшін жарамды: кәсіпорын ауқымындағы ақпараттық жүйелерден таратылған Web-қосымшаларға және тіпті нақты уақыттың енгізілген жүйелеріне дейін. Бұл жүйені әзірлеу мен одан әрі өрістетуге қатысы бар барлық көзқарастан қарауға мүмкіндік беретін өте мәнерлі тіл. Мәнерлі мүмкіндіктердің көптігіне қарамастан, бұл тіл түсіну және пайдалану үшін оңай. Ол үш негізгі элементті қамтиды: негізгі құрылыс блоктары, осы блоктардың өзара үйлесуі мүмкін екенін анықтайтын ережелер және тілдің кейбір жалпы механизмдері.

UML АЖ әзірлеу процесінің құрамдас бөлігі болып табылады. UML үлгілейтін шындыққа байланысты болмаса да, үлгілеудің процесі пайдалану прецеденттерін қарауға негізделген кезде оны қолдану жақсы, итеративті және қадамдық болып табылады,ал жүйенің өзі анық айқын архитектураға ие.

UML-бұл бағдарламалық жүйелердің элементтерін визуализациялау, спецификациялау, құрастыру және құжаттау үшін тіл. Тіл сөздіктер мен ережелерден тұрады, ол оған кіретін сөздерді біріктіріп, мағыналы құрылымдарды алуға мүмкіндік береді. Модельдеу тілінде сөздік және ЕРЕЖЕ жүйенің тұжырымдамалық және физикалық көрінісіне бағытталған. UML сияқты модельдеу тілі АЖ "сызбаларды" құрастырудың стандартты құралы болып табылады.

Кез келген күрделіліктегі АЖ құру кезінде болашақ жүйені модельдеу жүргізіледі, алайда көптеген жағдайларда бұл қандай да бір құжатты әзірлемей бейресми түрде жасалады. Алайда, мұндай тәсіл қиындықтарға ұшырайды. Біріншіден, тұжырымдамалық модель туралы пікір алмасу пікірталасқа қатысушылардың барлығы бір тілде сөйлегенде ғана мүмкін болады. Екіншіден, бағдарламалаудың мәтіндік тілінің шегінен шығатын модельсіз ақпараттық жүйелердің белгілі бір аспектілері туралы түсінік алуға болмайды. Үшіншіден, Егер код авторы өзі ойлаған модельдерді айқын түрде іске асырмаған болса, егер ол жұмыс орнын ауыстырса, бұл ақпарат мәңгілікке жоғалады. Ең жақсы жағдайда оны іске асыруға байланысты ішінара қайта құруға болады.

UML пайдалану үшінші мәселені шешуге мүмкіндік береді: айқын үлгі қарым-қатынасты жеңілдетеді.

Жүйенің кейбір ерекшеліктері мәтін түрінде үлгілеу жақсы, басқалары — графикалық. Шын мәнінде, барлық қызықты жүйелерде бір ғана

бағдарламалау тілі арқылы елестету мүмкін емес құрылымдар бар. UML-белгіленген мәселелердің екіншісін шешуге мүмкіндік беретін графикалық тіл.

UML-бұл тек графикалық таңбалар жиынтығы емес. Олардың әрқайсысында жақсы белгілі семантика бар. Бұл дегеніміз, бір әзірлеуші жазған модель басқаша — немесе тіпті аспаптық бағдарламамен бірдей түсіндірілуі мүмкін. Жоғарыда аталған проблемалардың бірі шешілуде.

Бұл контексте специфицирование дәл, мағынасыз және толық модельдерді құруды білдіреді. UML талдау, жобалау және іске асыруға қатысты барлық маңызды шешімдерді сипаттауға мүмкіндік береді, олар ақпараттық жүйені әзірлеу және өрістету процесінде қабылдануы тиіс.

UML визуалды жобалау тілі болып табылады, ал оның көмегімен жасалған модельдер тікелей АЖ бағдарламалау тілдеріне аударылуы мүмкін.

АЖ әзірлеу кезінде мынадай элементтер де құрылады:

- жүйеге қойылатын талаптар;
- сәулет сипаттамасы;
- жоба;
- бастапқы коды;
- жобалық жоспарлар;
- тесттер;
- прототиптер;
- болжамдар, және т. б.

Игерудің қабылданған әдістемесіне байланысты бір жұмыстарды орындау басқаларына қарағанда формальды түрде жүргізіледі. Аталған элементтер-бұл жобаның жай ғана жеткізілетін құрамдас бөліктері емес; олар басқару үшін, нәтижені бағалау үшін, сондай-ақ жүйені әзірлеу кезінде және оны өрістетгеннен кейін ұжым мүшелері арасындағы қарым-қатынас құралы ретінде қажет.

UML жүйелік архитектураны және оның барлық бөлшектерін құжаттау проблемасын шешуге мүмкіндік береді, жүйеге қойылатын талаптарды қалыптастыру және тестілерді анықтау үшін тілді ұсынады, ақырында жобаны жоспарлау және нұсқаларды басқару кезеңінде жұмыстарды модельдеу үшін құралдарды ұсынады.

UML қайда қолданылады

UML тілі ең алдымен ақпараттық жүйелерді әзірлеуге арналған. Оны пайдалану әсіресе келесі салаларда тиімді:

- кәсіпорын ауқымының ақпараттық жүйелері;
- банктік және қаржылық қызметтер;
- телекоммуникация;
- көлік;
- қорғаныс өнеркәсібі, авиация және ғарышкер;
- бөлшек сауда;
- медициналық электроника;
- ғылым;

– таратылған Web-жүйелер.

UML қолдану аясы бағдарламалық қамтамасыз етуді модельдеумен шектелмейді. Оның мәнерлілігі заң жүйесіндегі құжат айналымын, ауруханалардағы пациенттерге қызмет көрсету жүйесінің құрылымы мен жұмыс істеуін модельдеуге, аппараттық құралдарды жобалауды жүзеге асыруға мүмкіндік береді.

UML тұжырымдамалық моделі

UML түсіну үшін үш құрамдас бөлікті қамтитын оның тұжырымдамалық моделін меңгеру қажет: тілдің негізгі құрылыс блоктары, олардың комбинациясы ережелері және бүкіл тіл үшін кейбір ортақ механизмдер. Бұл элементтерді меңгере отырып, сіз UML модельдерін оқып, оларды өз бетінше жасай аласыз — басында, әрине, өте қиын емес. Тілмен жұмыс істеу тәжірибесін алу үшін Сіз оның мүмкіндіктерін пайдалануды үйренесіз.

UML құрылыс блоктары

UML тілінің сөздігі құрылыс блоктарының үш түрін қамтиды:

– мәні;
– қатынастар;
– диаграммалар.

Мәндер-модельдің негізгі элементтері болып табылатын абстракциялар. Қарым-қатынас әр түрлі мәндерді байланыстырады; диаграммалар қызығушылық танытатын мәндер жиынтығын топтастырады.

UML-де нысандардың төрт түрі бар:

– құрылымдық;
– мінез-құлық;
– топтастыру;
– аннотациялық.

Мәндер тілдің негізгі объектілі-бағытталған блоктары болып табылады. Олардың көмегімен дұрыс модельдер жасауға болады. Құрылымдық нысандардың жеті түрі бар:

– сынып (Class);
– интерфейс (Interface);
– кооперация (Collaboration);
– прецедент (Use case);
– белсенді сынып (Active class);
– компонент (Component);
– түйін (Node).

Бұл жеті базалық элементтер-сыныптар, интерфейстер, кооперациялар, прецеденттер, белсенді сыныптар, компоненттер және тораптар — UML моделіне енгізілуі мүмкін негізгі құрылымдық нысандар болып табылады. Сондай-ақ, осы нысандардың түрлері бар: актерлер, сигналдар, утилиттер (сынып түрлері), процестер мен жіптер (Белсенді сынып түрлері), қосымшалар, құжаттар, файлдар, кітапханалар, беттер мен кестелер (компоненттер түрлері).

Мінез-құлық мәні (Behavioral things) UML моделінің динамикалық құрамдас бөлігі болып табылады. Бұл тіл етістіктері: олар модельдің уақыт пен кеңістікте мінез-құлқын сипаттайды.

Мінез-құлық мәндерінің екі негізгі түрі бар:

– өзара әрекеттесу (Interaction);
– автомат (State machine).

Бұл екі өзара әрекеттесу элементі және автоматтар UML моделіне кіретін негізгі мінез-құлық мәні болып табылады. Олар семантикалық түрде әр түрлі құрылымдық элементтермен, бірінші кезекте — кластармен, кооперациялармен және объектілермен байланысты.

Топтастырушы мәндер UML моделінің ұйымдастырушы бөліктері болып табылады. Бұл модельді бөлуге болатын блоктар. Тек бір бастапқы топтастыру мәні бар, атап айтқанда пакет.

Сонымен қатар, UML-дің негізгі топтастырушы мәндері де бар, мысалы, қаңқалар (Frameworks), модельдер мен кіші жүйелер.

Аннотация мәні-UML моделінің түсіндірме бөліктері. Бұл модельдің кез келген элементіне Қосымша сипаттама, түсініктеме немесе ескерту. Аннотация элементтерінің тек бір негізгі түрі бар-ЕСКЕРТУ (Note). Ескерту-элементке немесе элемент тобына қосылған түсініктемелер немесе шектеулер үшін ғана таңба. Графикалық Ескертпе мәтіндік немесе графикалық түсініктемесі бар жиегі қисық тікбұрыш түрінде бейнеленеді.

UML тілінде қатынастардың төрт түрі анықталды:

– тәуелділік;
– қауымдастық;
– жалпылау;
– іске асыру.

Бұл қатынастар UML-дегі негізгі байланыстырушы құрылыс блоктары болып табылады және дұрыс модельдерді жасау үшін қолданылады.

Сонымен қатар, UML моделіне қосуға болатын қарым-қатынастардың негізгі түрлері болып табылады, мысалы, нақтылау (Refinement), трассалау (Trace), қосу және кеңейту (тәуелділік үшін).

UML-дегі Диаграмма-бұл көбінесе шыңдармен (мәндермен) және қырлармен (қатынастармен) байланысты бағаң түрінде бейнеленген элементтер жиынтығының графикалық көрінісі. Диаграммалар әртүрлі көзқараспен жүйені визуализациялау үшін сызылады. Диаграмма кейбір мағынада жүйенің проекцияларының бірі. Әдетте, неғұрлым тривиальды жағдайларды қоспағанда, диаграммалар жүйе құрылған элементтердің бүктелген көрінісін береді. Бір элемент барлық диаграммаларда немесе тек бірнеше (ең көп таралған нұсқа) болуы мүмкін немесе бірде-бір (өте сирек) болмауы мүмкін. Теориялық диаграммалар кез келген мәндер мен қатынастардың комбинацияларын қамтуы мүмкін. Іс жүзінде, алайда, ақпараттық жүйенің архитектурасын құрайтын ең көп қолданылатын бес түрге сәйкес келетін типтік комбинациялардың салыстырмалы аз саны қолданылады. Осылайша, UML-да диаграммалардың тоғыз түрін анықтайды:

– сынып диаграммалары;
– Нысандар диаграммалары;
– прецеденттер диаграммалары;
– тізбектер диаграммалары;
– кооперация диаграммалары;
– күй диаграммалары;
– әрекеттер диаграммалары;
– компоненттер диаграммалары;
– өрістету диаграммалары.

Мұнда UML-да қолданылатын диаграммалар тізімі толық емес. Аспаптық құралдар басқа диаграммаларды жасауға мүмкіндік береді, бірақ аталған тоғыз іс жүзінде жиі кездеседі.

UML тілінің ережелері

UML құрылыс блоктарын бір-бірімен еркін біріктіруге болмайды. Кез келген басқа тіл сияқты, UML жақсы ресімделген модель қалай көрінуі керек екенін анықтайтын ережелер жиынтығымен сипатталады, яғни семантикалық өзі келісілген және оған байланысты емес барлық модельдермен үйлесімде тұрған.

UML тілінде дұрыс және бір мәнді анықтауға мүмкіндік беретін семантикалық ережелер бар:

– мәндер, қатынастар және диаграммалар беруге болатын атаулар;
– іс-әрекет аймағы (аты кейбір мағынаға ие контекст);
– көріну (аттар көрінгенде және басқа элементтермен пайдаланылуы мүмкін);
– тұтастық (элементтер қалай дұрыс және бір-бірімен сәйкес келуі керек);
– орындау {кейбір динамикалық үлгіні орындау немесе имитациялау дегенді білдіреді).

Ақпараттық жүйелерді әзірлеу процесінде құрылатын модельдер уақыт өте келе эволюциялайды және әртүрлі уақытта жобаның әртүрлі қатысушыларымен бірдей қаралуы мүмкін. Осы себепті жақсы безендірілген модельдер ғана емес, сондай-ақ олар:

– құрамында жасырын элементтер бар (элементтер қатары қабылдауды жеңілдетуді көрсетпейді);
– толық емес (жеке элементтер жоқ);
– келісілмеген (модельдің тұтастығына кепілдік берілмейді).

Жүйенің барлық бөлшектері толық анықталмайынша, әзірлеу процесінде өте жақсы ресімделген модельдердің пайда болуы сөзсіз емес. UML тілінің ережелері — қажет болмаса да-модельмен жұмыс істеу барысында талдаудың, жобалаудың және іске асырудың аса маңызды мәселелерін шешуге итермелейді,нәтижесінде модель уақыт өте жақсы ресімделген болады.

UML тілінің жалпы механизмдері

Егер кейбір келісімдерді ұстанса, құрылыс жеңілдетіледі және тиімді жүргізіледі. Белгілі бір сәулеттік үлгілерге сәйкес ғимаратты викториан немесе француз стилінде ресімдеуге болады. Сол принцип UML-ге қатысты да қолданылады. Осы тілмен жұмыс төменде аталған жалпы тетіктерді дәйекті пайдалануды жеңілдетеді:

– ерекшеліктер (Specifications);
– қосымша (Adornments);
– қабылданған бөліктер (Common divisions);
– кеңейту механизмдері (Extensibility mechanisms).

Архитектурасы

Ақпараттық жүйелерді визуализациялау, спецификациялау, құрастыру және құжаттау үшін оларды әртүрлі көзқараспен қарау қажет. Жобаға қатысы бар барлық адамдар-соңғы пайдаланушылар, талдаушылар, әзірлеушілер, жүйелік интеграторлар, тестілеушілер, техникалық жазушылар мен жобалар менеджерлері — өз мүдделерін көздейді және әркім өмірдің әр түрлі кезеңдерінде құрылатын жүйеге қарайды. Жүйелік архитектура, мүмкін, барлық көру нүктелерін басқару үшін пайдаланылатын ең маңызды элемент болып табылады және сол арқылы оның өмірлік циклы бойы жүйенің итеративті және инкременттік әзірленуіне ықпал етеді.

Архитектура— бұл маңызды шешімдердің жиынтығы:

– бағдарламалық жүйені ұйымдастыру;
– жүйені құрайтын құрылымдық элементтерді және олардың интерфейстерін таңдау;
– басқа элементтермен кооперацияларда маманданған осы элементтердің мінез-құлқы;
– осы құрылымдық және мінез-құлық элементтерінен көп және одан да көп ірі кіші жүйелерді құру;
– жүйенің барлық ұйымдастырылуын бағыттайтын және анықтайтын архитектуралық стиль: статикалық және динамикалық элементтер, олардың интерфейстері, кооперациясы және оларды біріктіру тәсілі.

Бағдарламалық жүйенің архитектурасы тек құрылымдық және мінез-құлық аспектілерін ғана емес, бқ және пайдалануды, функционалдығын, өнімділігін, икемділігін, қайта қолдану мүмкіндігін, толықтығын, экономикалық және технологиялық шектеулер мен ымыраларды, сондай-ақ эстетикалық мәселелерді қамтиды.

Прецеденттер (Use case view) тұрғысынан көрініс соңғы пайдаланушылар, талдаушылар және тестілеушілер бақылайтын жүйенің мінез-құлқын сипаттайтын прецеденттерді қамтиды. Бұл түр бағдарламалық жүйенің шынайы ұйымдастырылуын емес, жүйелік архитектураның қалыптасуы тәуелді қозғаушы күштерді білдіреді. UML тілінде бұл түрдің статикалық аспектілері прецеденттер диаграммаларымен, ал динамикалық өзара әрекеттесу, күй және әрекеттер диаграммаларымен беріледі.

Жобалау тұрғысынан түр (Design view) есептер сөздігі мен шешім се қалыптастыратын класстарды, интерфейстерді және кооперацияларды

қамтиды. Бұл түр ең алдымен жүйеге қойылатын функционалдық талаптарды, яғни ол соңғы пайдаланушыларға ұсынуға тиіс қызметтерді қолдайды. UML тілінің көмегімен осы түрдің статикалық аспектілерін кластар мен нысандар диаграммаларымен, ал динамикалық өзара әрекеттесу, күй мен әрекеттер диаграммаларымен жіберуге болады.

Процестер тұрғысынан (Process view) түр параллелизм және синхронизация механизмдерін қалыптастыратын жіптер мен процестерді қамтиды. Бұл түр негізінен жүйенің өнімділігін, ауқымдылығын және өткізу қабілетін сипаттайды. UML - де оның статикалық және динамикалық аспектілері жобалау тұрғысынан түрге арналған диаграммалармен визуалданады, бірақ тиісті жіптер мен процестерді білдіретін белсенді класстарға ерекше назар аударылады.

Іске асыру тұрғысынан көрініс (Implementation view) соңғы бағдарламалық өнімді құрастыру және шығару үшін пайдаланылатын компоненттер мен файлдарды қамтиды. Бұл түр, ең алдымен, өзара әр түрлі біріктірілуі мүмкін тәуелсіз (кейбір дәрежеге дейін) компоненттер мен файлдардан жасалатын жүйе нұсқаларының конфигурациясын басқаруға арналған. UML тілінде бұл түрдің статикалық аспектілерін компоненттер диаграммаларының көмегімен, ал динамикалық — өзара әрекеттесу, күй және әрекеттер диаграммаларының көмегімен береді.

Тарату тұрғысынан көрініс (Deployment view) ол орындалатын жүйенің аппараттық құралдарының топологиясын қалыптастыратын тораптарды қамтиды. Бірінші кезекте ол физикалық жүйені құрайтын бөліктерді бөлумен, жеткізумен және орнатумен байланысты. Оның статикалық аспектілері өрістету диаграммаларымен, ал динамикалық-өзара іс-қимыл, жай-күй және іс-қимыл диаграммаларымен сипатталады.

Аталған түрлердің әрқайсысы дербес болып саналуы мүмкін, сондықтан жүйені әзірлеуге қатысы бар тұлғалар тек оларға тікелей қатысты сәулет аспектілерін зерттеуге шоғырлануы мүмкін. Бірақ бұл түрлердің бір-бірімен өзара әрекеттесуін ұмытуға болмайды. Мысалы, өрістету тұрғысынан түр тораптары іске асыру тұрғысынан түр үшін сипатталған компоненттерден тұрады, ал олар өз кезегінде жобалау мен процестер тұрғысынан түрлерден кластардың, интерфейстердің, кооперациялардың және белсенді кластардың физикалық іске асырылуы болып табылады. UML аталған бес түрдің әрқайсысын және олардың өзара әрекеттесуін көрсетуге мүмкіндік береді.

2.2 Прецендеттер диаграмасы

UML-да пайдалану нұсқаларының диаграммасы (use case diagram) - бұл субъект пен пайдалану нұсқаларының арасындағы қатынастарды көрсететін диаграмма және жүйені тұжырымдамалық деңгейде сипаттауға мүмкіндік беретін пайдалану нұсқаларының үлгісінің ажырамас бөлігі болып табылады.

Прецедент-модельдеуші жүйенің мүмкіндігі (оның функционалдық бөлігі), оның арқасында пайдаланушы нақты, өлшенетін және қажетті нәтиже ала алады. Прецедент жүйенің жеке сервисіне сәйкес келеді, оны пайдалану нұсқаларының бірін анықтайды және пайдаланушының жүйемен өзара іс-қимылының типтік тәсілін сипаттайды. Пайдалану параметрлері әдетте сыртқы жүйенің талаптарын көрсету үшін қолданылады. Пайдаланушының әрекеті: сабаққа қатысу, емтихан тапсыру, жүйеге кіру. Сонымен, бұл пайдаланушы жүйеге кіре алады, сабақ көруге және әр сабақтан кейін викториналдан өтуге болады.

"Жазылу" - бұл біздің жүйеде пайдаланушының жолындағы бірінші функция. Бұл функция пайдаланушы аутентификациясына және телефон нөміріне жауап береді. Бұл функция аты, тегі, электрондық поштасы, ағылшын тілі және телефон нөмірі қабылдайды, содан кейін жауап ретінде енгізілген телефон нөмірі бар Mobizon post сұрау жібереді, сондай-ақ пайдаланушының телефон нөміріне жіберілген қауіпсіздік кодын қайтарады. Осыдан кейін пайдаланушы осы кодты енгізеді, және ол Mobizon сәйкес келсе, тіркеу аяқталды.

Log in функциясы пайдаланушы кіруіне жауап береді. Бұл әдіс телефон нөмірі мен құпия сөзді қабылдайды. Осы деректерді қабылдағаннан кейін ол деректер қорына сұрау жібереді және енгізілген деректерді түзетемін, бұл функция true қайтарады.

Сабаққа қатысыңыз-жүйені тіркегеннен кейін пайдаланушының қол жетімді сабақтарының бар-жоғын тексереді. Егер олар болса, жүйе сабақ бейнесін, тест пен сөздерді ашады.

Біздің жүйеміздің өмірлік циклінде екі рет тест сынақтарын өткізу. Firslty ол тіркеу кезінде шақырады, жүйе пайдаланушыға тест өтуге ұсынады. Тест нәтижесі пайдаланушыға ағылшын тілінің деңгейін білуге көмектеседі. Тест өткеннен кейін екінші рет оқушының сабақты сәтті аяқтағанына көз жеткізу үшін әр деңгейден кейін ашылады.

Add lesson әдісі сабақтың бекітілген бейне файлын қабылдайды, дұрыс және дұрыс емес жауаптар мен сөздермен көпше таңдау ретінде тестілейді. Содан кейін бейне url-жолын, сұрақ массивін және сөз массивін кірістіретін деректер базасына сұраныс жасалады.

Ағын қосу және кесте орнату-бір-бірімен тығыз байланысты екі функция. Add stream функциясы ағын деңгейін орнатуға және курстың басталу күнін орнатуға жауап береді. Кестені теңшеу кезінде ағынның басқаруына жауап береді.

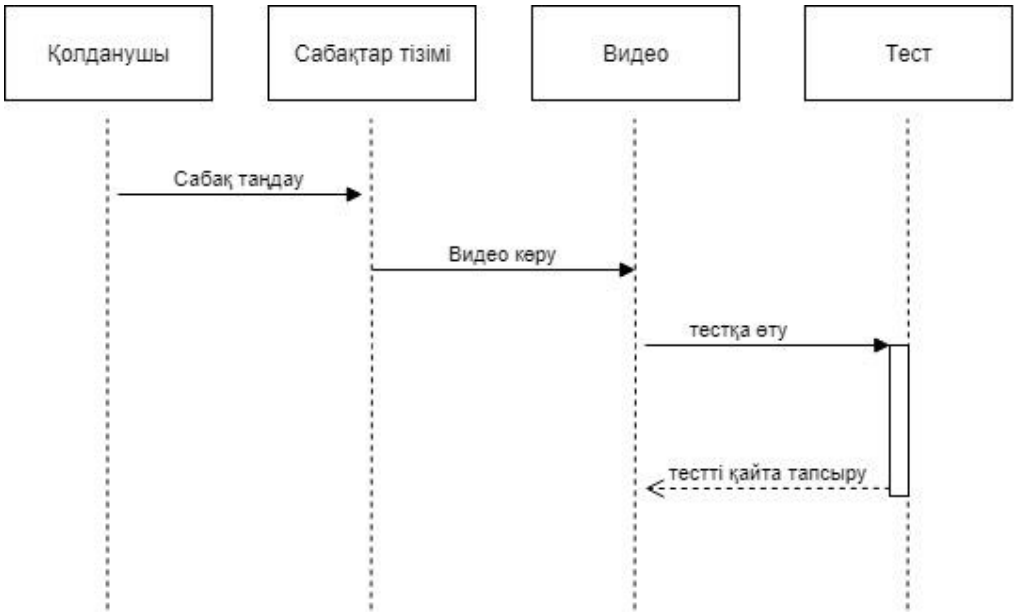
Төменде жобаның прецеденттер диаграммасы көрсетілген:

2.1-сурет – Жобаның прецеденттер диаграммасы

2.3 Тізбек диаграммасы

Тізбектер диаграммалары уақыт ағымында хабар алмасу терминдерінде сыныптар арасындағы өзара әрекеттесуді сипаттайды. Олар сондай-ақ оқиғалар диаграммалары деп аталады. Реттілік диаграммасы-әр түрлі сценарийлерді визуализациялау мен тексерудің жақсы тәсілі. Олар жүйе өзін қалай ұстайтынын болжауға және жаңа жүйені модельдеу процесінде класқа қажет болатын міндеттерді анықтауға көмектеседі[8]. пайдаланушыны тіркеудің толық процесі бейнеленеді. Біз осындай қызметтерді пайдаландық Mobizon.kz SMS-растау және CloudPayments төлем виджет үшін. Сонымен, алдымен пайдаланушы Тіркелу түймешігін басады және жүйе тіркелгі бар екенін тексереді. Егер бұл телефон нөмірі жоқ болса, жүйе Mobizon сұрау жасайды, ол SMS арқылы пайдаланушыға тексеру кодын жіберу керек және жауап ретінде бұл кодты жүйеге жіберу керек. Біздің көпшілігіміз іс-әрекет дәйектілігінің схемасы-бұл әртүрлі орындау сценарийлерін визуализациялау мен тексерудің жақсы тәсілі екенін білеміз, біз оны біздің дипломдық жобаға қосуға шешім қабылдадық. Пайдаланушының ағаш дайындау қызметінің

дәйекті іс-әрекеттері қысқаша көрсетілген. Пайдаланушы жасайтын бірінші әрекет-бұл жүйеге кіру, содан кейін сабақтар тізіміне кіру арқылы видео көреді. Видео біткеннен кейін тест тапсырады. Тестті қайта тапсыруға мүмкіндік бар. Тест бөлімінен кейін сөздер бөліміне өту арқылы сөз жаттауға болады. Жүйеден шығу жолымен жүйедегі барлық әрекеттерді аяқтау. Төменде жобаның тізбек диаграммасы көрсетілген:



2.2-сурет – Жобаның тізбек диаграммасы

2.4 Класстар диаграммасы

Кластар диаграммасы (class diagram) объектілі-бағытталған бағдарламалау кластарының терминологиясында жүйе моделінің статикалық құрылымын ұсыну үшін қызмет етеді. Класс диаграммасы, атап айтқанда, объектілер мен кіші жүйелер сияқты пәндік саланың жекелеген мәндерінің арасындағы Әртүрлі өзара байланыстарды көрсете алады,сондай-ақ олардың ішкі құрылымы мен қатынас түрлерін сипаттайды. Бұл диаграммада жүйенің жұмыс істеуінің уақытша аспектілері туралы ақпарат көрсетілмейді. Осы тұрғыдан алғанда, класс диаграммасы жобаланатын жүйенің тұжырымдамалық моделін одан әрі дамыту болып табылады.

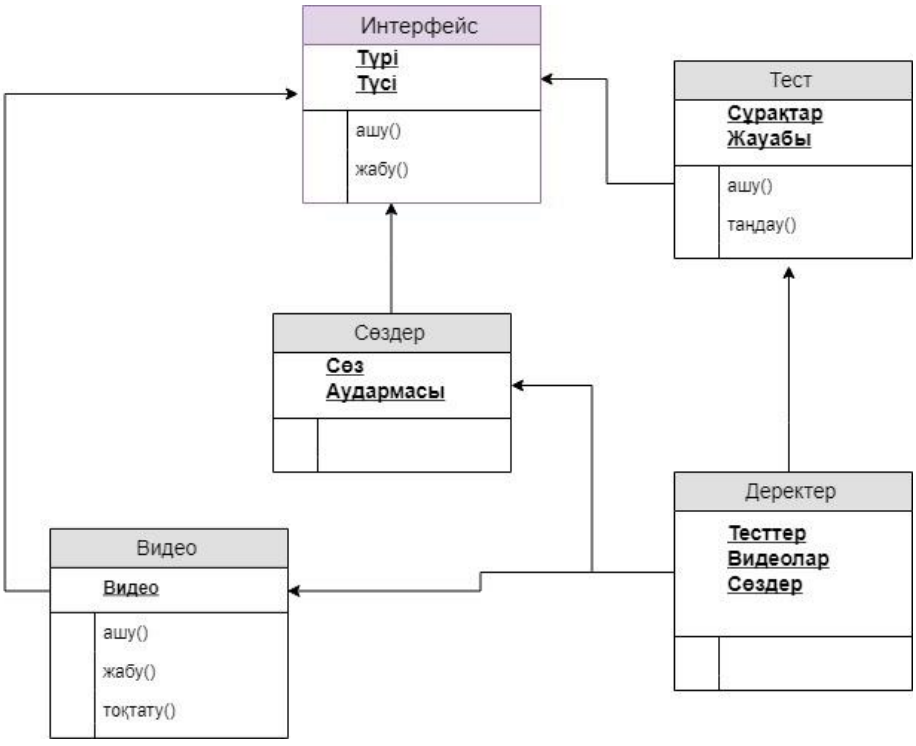
Класс диаграммасы құрылымдық қатынастардың әртүрлі типтерімен байланысты "классификатор" типті элементтері болып табылатын кейбір бағандар болып табылады. Сонымен қатар, бұл диаграммада интерфейстер, пакеттер, қарым-қатынастар және объектілер мен байланыс сияқты жеке даналар болуы мүмкін. Бұл Диаграмма туралы айтатын болсақ, жобаланатын жүйенің статикалық құрылымдық моделін білдіреді. Сондықтан кластардың диаграммасын жүйенің логикалық моделінің графикалық өзара байланыстары ұсынылған деп санауға болады, олар уақытты тәуелді емес немесе инвариантты.

Класс диаграммасы көптеген элементтерден тұрады, олар жиынтығында пәндік сала туралы декларативті білімді көрсетеді. Бұл білімдер UML тілінің кластары, интерфейстер және олардың құрамдастары арасындағы қатынастар сияқты базалық ұғымдарында түсіндіріледі. Сонымен қатар, бұл диаграмманың жеке компоненттері жүйенің жалпы моделін ұсыну үшін пакеттерді құруы мүмкін. Егер класс диаграммасы кейбір пакеттің бір бөлігі болып табылса, онда оның компоненттері басқа пакеттердегі элементтерге ықтимал сілтемелерді қоса алғанда, осы пакеттің элементтеріне сәйкес келуі тиіс.

Жалпы жағдайда статикалық құрылымдық модель пакеті кластардың бір немесе бірнеше диаграммалары түрінде ұсынылуы мүмкін. Жеке диаграммаларға кейбір көріністің декомпозициясы пәндік аймақтың құрылымдық өзара байланыстарының ыңғайлылығы және графикалық визуализациясы мақсатында орындалады. Бұл ретте диаграмма компоненттері статикалық семантикалық модель элементтеріне сәйкес келеді. Жүйенің моделі, өз кезегінде, UML тілінде сипатталатын сыныптардың ішкі құрылымымен келісілуі тиіс.

UML тіліндегі Класс (class) бірдей құрылымы, мінез-құлығы және басқа сыныптардан объектілермен қатынасы бар көптеген объектілерді белгілеу үшін қызмет етеді. Графикалық сынып тікбұрыш түрінде бейнеленеді, ол қосымша көлденең сызықтармен бөлімдер немесе секцияларға бөлінуі мүмкін. Бұл бөлімдерде класс атауы, атрибуттар (айнымалылар) және операциялар (әдістер) көрсетілуі мүмкін.

Төменде жобаның класстар диаграммасы көрсетілген:

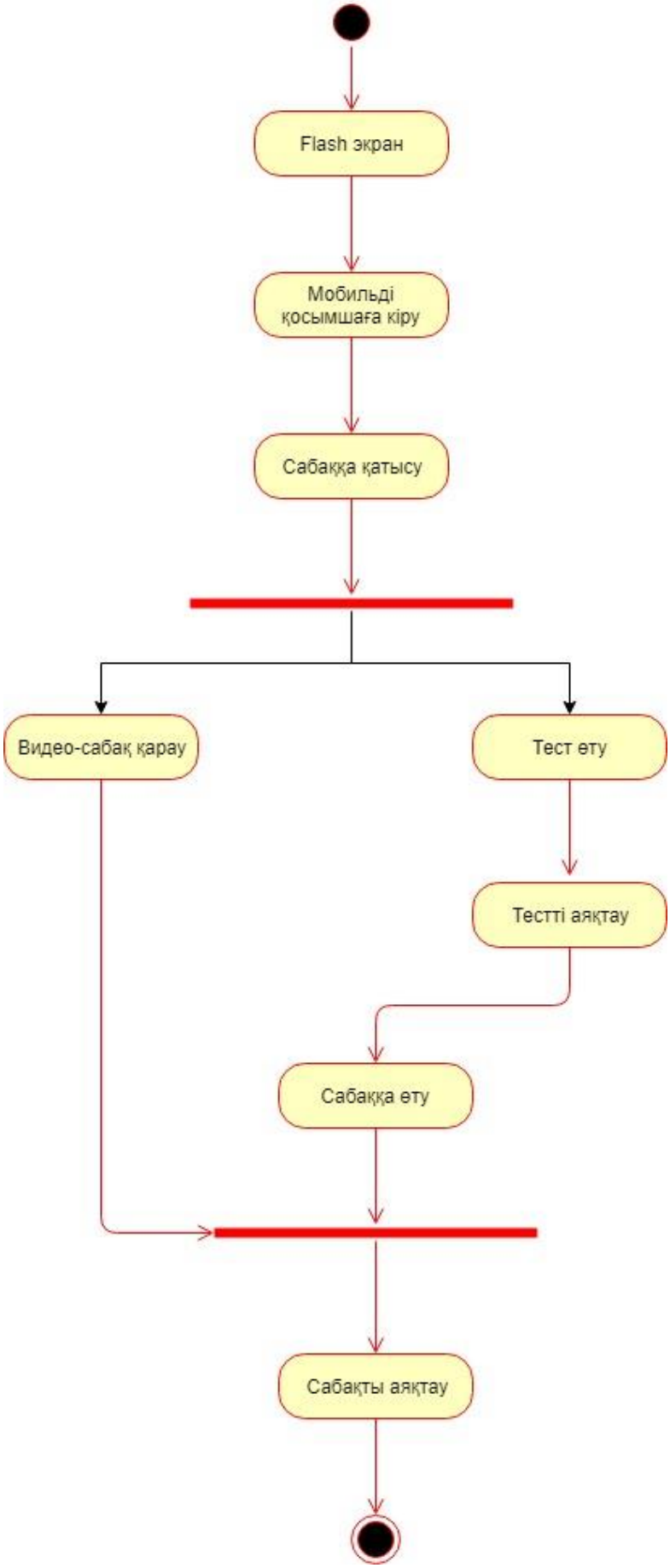


2.3-сурет – Жобаның класстар диаграммасы

2.5 Күй диаграммасы

UML-да белсенділік диаграммасы процедуралық жүйелік әрекеттердің орындалған жиынтығының графикалық көрінісі болып табылады және күй диаграммасының вариациясы ретінде қарастырылады. Іс-қимылдар диаграммалары параллель және шартты әрекеттерді, пайдалану нұсқаларын және жүйелік функцияларды егжей-тегжейлі деңгейде сипаттайды. Іс-әрекет диаграммасы іс-әрекет тізбектеріне және іс-әрекет бастамашылығының тиісті шарттарына назар аудара отырып, үлкен іс-әрекеттің дәйекті жұмыс ағынын модельдеу үшін қолданылады. Әрекет күйі жұмыс процесінің әрбір қадамының өнімділігіне жатады. Белсенділік диаграммасы көрсеткілермен біріктірілген фигуралармен көрсетілген. Бағыттамалар іс-әрекеттің басынан бастап аяқтауға дейін іске қосылады және орындалатын іс-әрекеттердің дәйекті тәртібін ұсынады. Қара шеңберлер жұмыс процесінің бастапқы күйін көрсетеді [9]. Қара шеңбер соңғы күйін көрсетеді. Дөңгелектелген тіктөртбұрыштар әрбір тіктөртбұрыштың ішіндегі мәтінмен сипатталатын орындалатын әрекеттерді білдіреді. Төменде келтірілген мысалда біздің жүйеде тіркелгі бар пайдаланушының әрекеттерін көре аласыз. Мұнда пайдаланушы енгізген жарамды құпия сөз немесе телефон нөмірі сияқты бірнеше опциялар бар, оларда белсенді курстар бар ма. Сонымен қатар, пайдаланушы жоқ болса, жүйе курс сатып алуды ұсынады.

Қысқаша, барлық деректер пайдаланушылар кейбір сұрауларды орындағанда қарастырылады және өңделетін сервермен және деректер базасымен өзара әрекеттесетін пайдаланушының қадамдары сипатталады. Егер бұл деректер базасындағы ағымдағы деректер (телефон нөмірі немесе электрондық пошта және пароль) болса және дұрыс жазылған болса, пайдаланушы берген сұрау табысқа ие болады және жүйе қатынауды тексеруді береді. Сәтті тексергеннен кейін пайдаланушы ағымдағы курстарды көре алады немесе жаңа курстарды таңдай алады, кестені, бағалауды, трансляция санын, кітапханаларды және т.б. біліңіз. Егер курс болмаса, пайдаланушы курстар ала алады, әйтпесе студент сабаққа қатысуы тиіс. Төменде осы жобаның күй диаграммасы келтірілген:



2.4-сурет – Жобаның күй диаграммасы

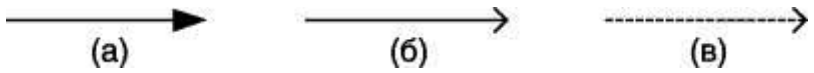
2.6 Кооперация диаграммасы

Кооперация диаграммасы қажетті мақсатқа жету немесе пайдаланудың кейбір нұсқасын іске асыру үшін өзара хабарламалармен алмасатын жекелеген объектілер деңгейіндегі жүйенің мінез-құлқын сипаттауға арналған

Кооперация — (collaboration) - модельделетін жүйенің жалпы контекстінде пайдаланудың жекелеген нұсқаларын іске асыру мақсатында бірлесіп өзара іс-қимыл жасайтын жекелеген кластардың көптеген объектілерінің ерекшелігі.

Кооперация ұғымы – UML тілінде іргелі болып табылады. Кооперацияның мақсаты-жекелеген пайдалану нұсқаларын немесе жүйедегі аса маңызды операцияларды іске асыру ерекшеліктерін мамандандыру. Кооперация осы кооперацияға қатысушылардың өзара іс-қимыл терминдеріндегі жүйенің мінез-құлқының құрылымын анықтайды.

UML тіліндегі хабарламалар, сондай — ақ, хабарламаны жіберуші және алушы-Нысандар ойнайтын рөлдерді сипаттайды. Кооперация диаграммасындағы хабарламалар тиісті байланыс немесе Қауымдастық рөлінің жанында қосымша бағыттамалармен бейнеленеді. Көрсеткі бағыты хабар алушыны көрсетеді. Хабар көрсеткісінің сыртқы түрі белгілі бір мағынаға ие. Кооперация диаграммаларында хабарламаларды белгілеу үшін үш нұсқалардың бірі пайдаланылуы мүмкін. (2.5-сурет)



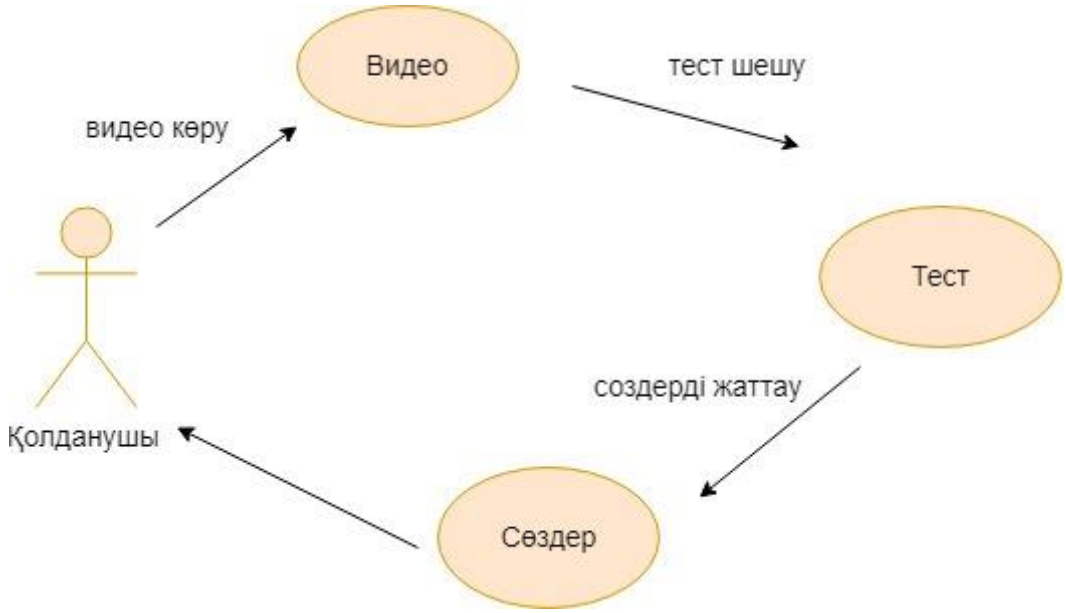
2.5-сурет – Кооперация диаграммасындағы әртүрлі хабарлар түрлерінің графикалық бейнесі

Кооперация диаграммасын құру кластар диаграммасын құрастырғаннан кейін бірден бастауға болады. Кооперация диаграммаларын әзірлеу кезінде алдымен объектілер мен олардың арасындағы байланыстар бейнеленеді. Одан әрі кооперация диаграммасына олардың тәртібі мен басқа да семантикалық ерекшеліктерін көрсете отырып, барлық хабарламалар салу қажет. Кооперация диаграммасы бұрын құрылған кластар диаграммасында анықталған объектілер мен байланыстарды ғана қамтуы мүмкін. Олай болмаған жағдайда, егер жоқ сыныптар негізінде құрылатын объектілерді кооперациялау диаграммасына қосу қажеттігі туындаса, онда сыныптар диаграммасы оған осы сыныптарды анық сипаттауды қосу арқылы модификациялануы тиіс.

Кооперация диаграммасын құру процесі кластар диаграммасын және кезектілік диаграммасын құру үдерістерімен келісілуі тиіс. Бірінші жағдайда, бұрын айтылғандай, оларды тудыратын сыныптар анықталған объектілерді ғана пайдалануды қадағалау қажет. Екінші жағдайда берілетін хабарламалардың кезектілігін келісу қажет. Әңгіме кооперация

диаграммасында және бірізділік диаграммасында бір өзара іс-қимылды модельдеу үшін хабарламаларды жүргізудің әртүрлі тәртібіне жол берілмейтіні туралы болып отыр.

Кооперация диаграммасы бір жағынан кластар диаграммасының статикалық моделінен дәйектілік, жай-күй және қызмет диаграммаларымен ұсынылатын мінез-құлық динамикалық моделдеріне тұжырымдамалық келісілген ауысуды қамтамасыз ететінін есте сақтау қажет. Сонымен жобаның кооперация диаграммасы төменде көрсетілген:



2.6-сурет – Жобаның кооперация диаграммасы

3 Жүзеге асыру және тестілеу бөлімі

3.1 Android Studio мүмкіндіктері

Android Studio-бұл 2013 жылдың 16 мамырында Google I/O конференциясында анонсталған Android платформасымен жұмыс істеу үшін біріктірілген даму ортасы (IDE).

IDE 2013 мамыр айында жарияланған 0.1 нұсқасынан бастап, содан кейін 2014 жылдың маусым айында шығарылған 0.8 нұсқасынан бастап бета-тестілеу сатысына өтті. 1.0 бірінші тұрақты нұсқасы 2014 жылдың желтоқсанында шығарылды, сонда Eclipse үшін Android Development Tools (ADT) плагинін қолдау тоқтатылды.

JetBrains компаниясынан IntelliJ IDEA бағдарламалық жасақтамасына негізделген Android Studio - Android қосымшаларын әзірлеудің ресми құралы. Бұл жұмыс ортасы Windows, OS X және Linux үшін қол жетімді. 17 мамыр 2017, Google I / O жыл сайынғы конференциясында, Google Java және C++қосу Android платформасына арналған ресми бағдарламалау тілі ретінде Android Studio-да қолданылатын Kotlin тілін қолдауды жариялады.

Ерекшеліктері:

Жаңа мүмкіндіктер әрбір жаңа Android Studio нұсқасымен пайда болады. Қазіргі уақытта келесі мүмкіндіктер бар:

– кеңейтілген макеттер редакторы: WYSIWYG, Drag-and-Drop көмегімен UI компоненттерімен жұмыс істеу қабілеті, экранның бірнеше конфигурациясында макетті алдын ала қарау функциясы;

– Gradle негізделген бағдарламаларды құрастыру;

– құрастырудың әртүрлі түрлері және бірнеше генерация .apk Файлдар

– код рефакторингі;

– статикалық код талдағышы (Lint), өнімділік, нұсқалардың үйлесімсіздігі және т.б. мәселелерін табуға мүмкіндік береді;

– кірістірілген ProGuard және қолданбаларға қол қою үшін утилитасы;

– негізгі макеттер мен Android компоненттерінің үлгілері;

– Android Wear және Android TV үшін бағдарламаларды дамыту[5];

– Google Cloud Messaging және App Engine сервистерімен интеграцияны қамтитын Google Cloud Platform қолдау;

– Android Studio 2.1 Android N Preview SDK қолдайды, бұл әзірлеушілер жаңа бағдарламалық платформаға арналған қосымшаны жасау бойынша жұмысты бастай алады;

– Android Studio 2.1 жаңа нұсқасы жаңартылған Jack компиляторымен жұмыс істей алады, сондай-ақ Java 8 қолдауы мен жетілдірілген Instant Run функциясын алды;

– Platform-tools 23.1.0 бастап Linux үшін тек 64 биттік;

– Android Studio 3.0-ке стандарт бойынша JetBrains IDE-ге негізделген Kotlin тілінің құралдары кіреді.

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

IDE жүктеуге және тегін пайдалануға болады. Онда UI құру үшін макеттер бар, әдетте қосымшамен жұмыс басталады. Studio бағдарламасында смартфондар мен планшеттерге арналған шешімдерді әзірлеу құралдары, сондай-ақ Android TV, Android Wear, Android Auto, Glass және қосымша контекстуалды модульдер үшін жаңа технологиялық шешімдер бар.

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

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

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

3.2 Android Studio және AVD орнату. Экран қасиеті мен компоненттері.

Бағдарламаны жазу үшін-әзірлеу ортасы қажет. Осы бөлімде даму ортасын қалай орнату және баптау керектігі көрсетіледі.

Сонымен, басқа JDK деп аталатын тиісті SDK жүктеу және орнату қажет (егер, әрине, ол орнатылмаған болса).

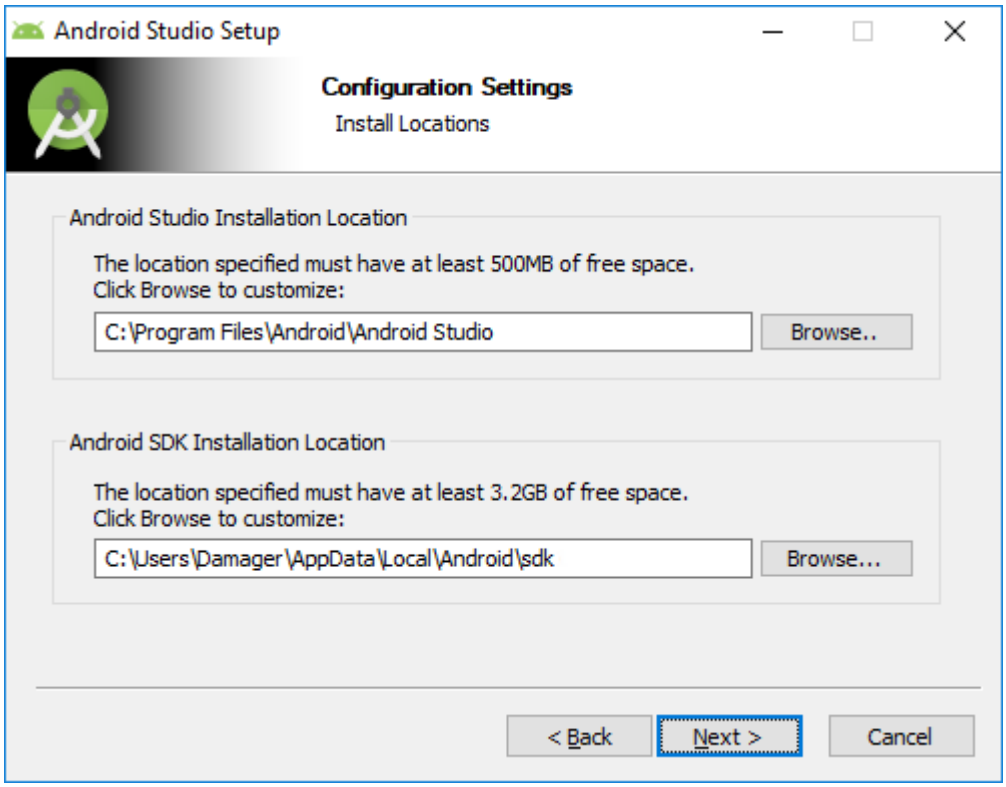
Орнатқаннан кейін компьютерді қайта жүктеуді (перезагрузить) ұсынамын.

Әзірлеу ортасында біз бағдарламаны жасап, дайын қосымшаны шығаруда алатын боламыз. Қазір бірнеше даму ортасы бар, Біз ұсынылған Google Android Studio Google таңдау

Бұл бетті ашып, Download Android Studio түймесін басыңыз. Егер сізге Арнайы нұсқа қажет болса, онда download Options түймесін басыңыз және онда сіздің нұсқаңызды таңдаңыз.

Файл 2 гигқа дейін салмақ болуы мүмкін екенін ескеріңіз.

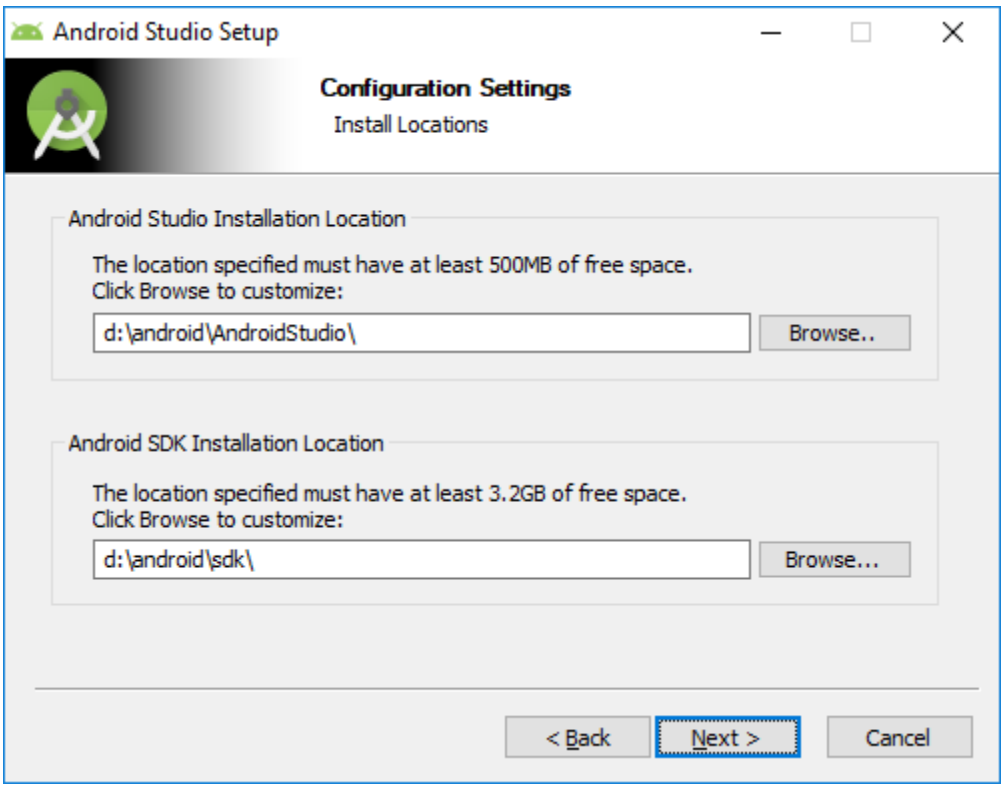
Сонымен, exe файлын жүктеді. Іске қосқалы отырмыз. Ол жол сұрамайынша Next басамыз.



3.1-сурет – Жол таңдау

Бізге екі жолды көрсету қажет. Бірінші жолы Android Studio орнату үшін қолданылады. Екіншісі-Android SDK орнату үшін.

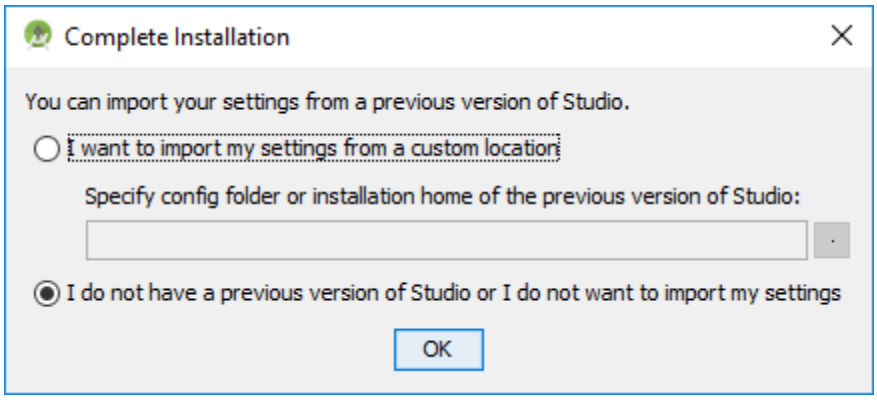
Оларды өзімізге ауыстырайық. Ол үшін android каталогын жасаймыз. Кез келген жағдайда оған жол бос орынсыз және орыс символдарсыз болатындай етіп жасаңыз. Мысалы - <диск атауы>:\android. Менде бұл болады d:\android. Және бұл жолды визардқа қоямыз.



3.2-сурет – SDK жолын көрсету

Орнату басталғанша Next бірнеше рет басамыз. Орнату аяқталған кезде Android Studio іске қосылады. Егер іске қосылмаған болса, онда оның таңбашасын іске қосу кезінде іздейміз.

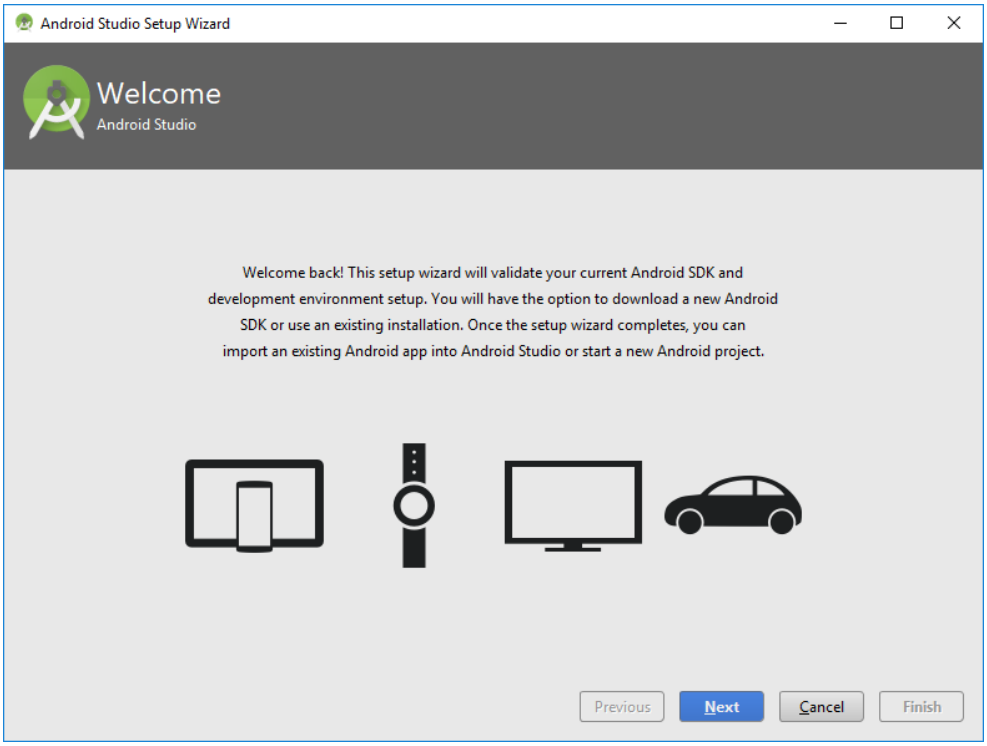
Бірінші кезекте ол алдыңғы версияның параметрлерін импорттау керек емес пе деп сұрайды.



3.3-сурет – Параметрлерді импорттау

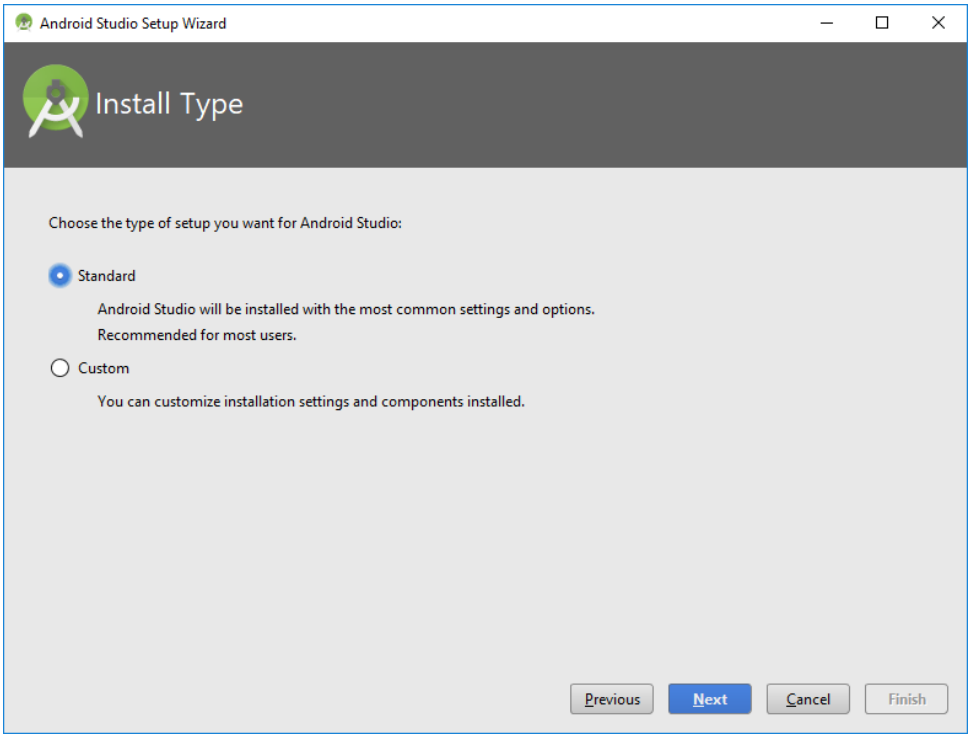
Таңдалған төменгі тармақты және Ок батырмасын басыңыз. Егерде ескі параметрлер жоқ болса.

Одан әрі орнату визардасы пайда болады.



3.4-сурет – Орнату терезесі

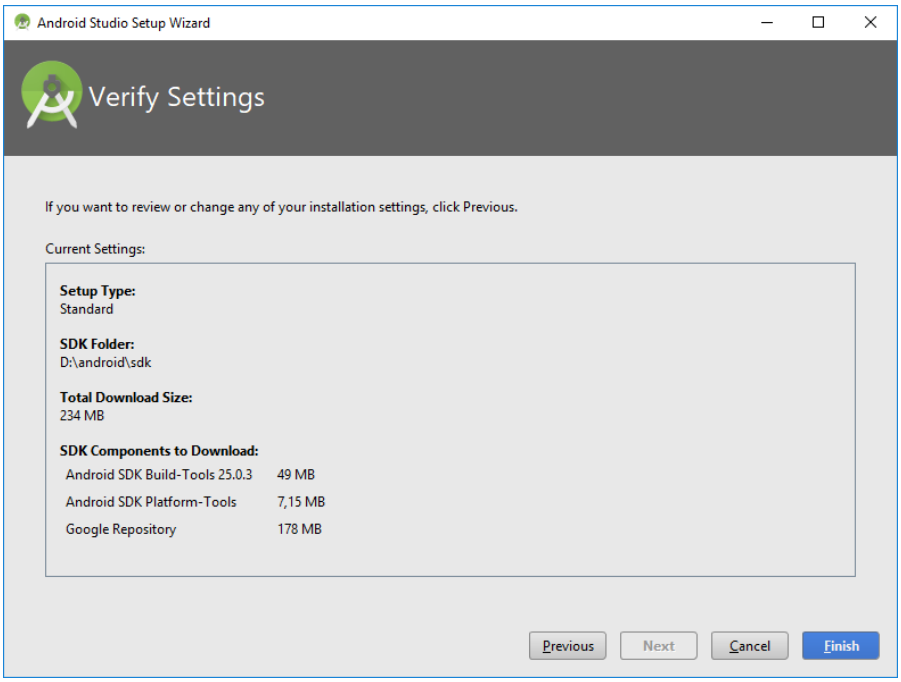
Next күтеміз.
Бұл жерде Standard режимін таңдаймыз.



3.4-сурет – Режим таңдау терезесі

Next таңдаймыз.

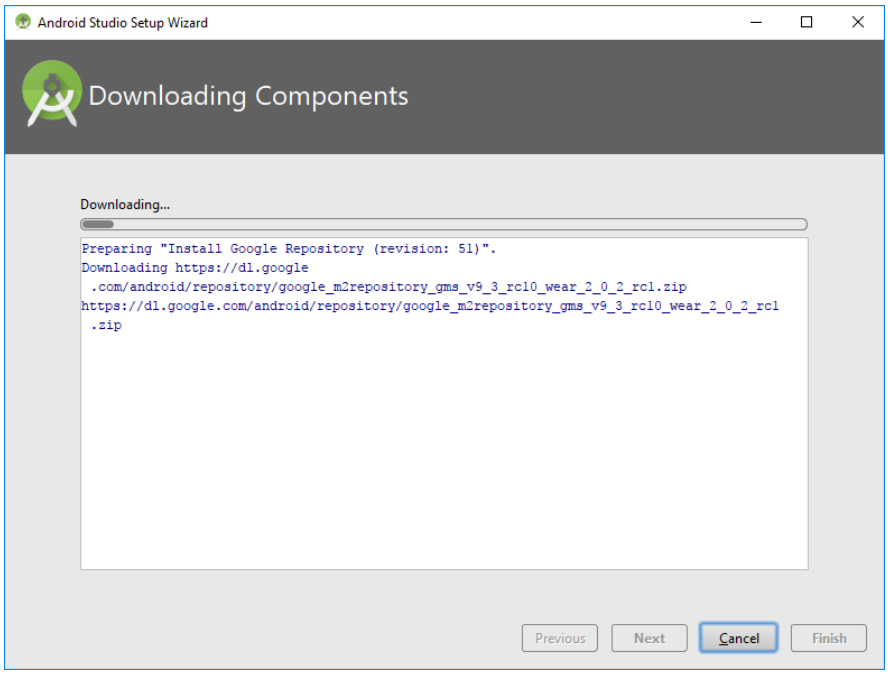
Visard орнатуды аяқтау үшін бірнеше компоненттерді жүктеу қажет екенін хабарлайды.



3.5-сурет – Компоненттерді жүктеу

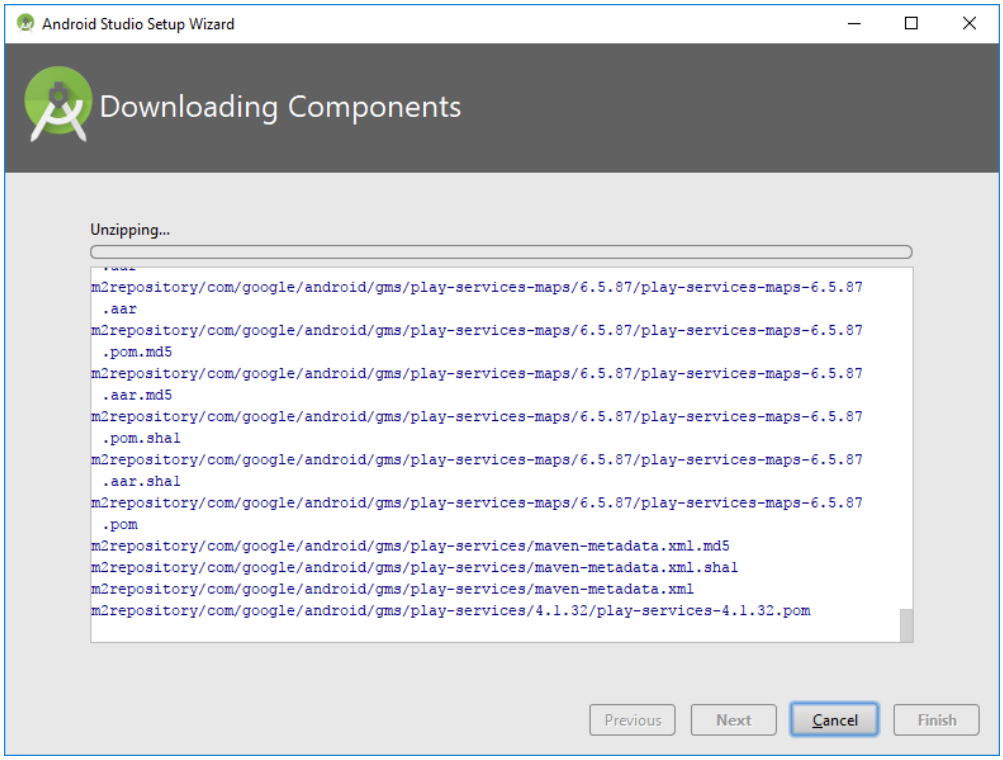
Next таңдаймыз.

Жүктеу процесі жүрді



3.6-сурет – Жүктеу процесі

Содан кейін тарқату процесі.



3.7-сурет – Тарқату процесі

Соңында, ол аяқтағанда, finish батырмасын басылғаннан кейін, Welcome экранды ашады.

3.3 Android SDK

Android SDK Android платформасына арналған мобильді қосымшаларды әзірлеуге көмектесетін әртүрлі кітапханаларды, құжаттаманы және құралдарды қамтиды.

– API Android SDK — API кітапхана Android үшін берілетін қосымшаларды әзірлеу;

– SDK құжаттамасы-әрбір пакетке және сыныпқа не енгізілген және қосымшаларды әзірлеу кезінде қалай пайдалануға болатын егжей-тегжейлі анықтамалық ақпаратты қамтиды;

– AVD— Android Virtual Device) - Android ұялы құрылғысының интерактивті эмуляторы. Эмуляторды пайдалана отырып, нақты Androidyctpойctva пайдаланбай қолданбаларды іске қосуға және сынауға болады;

– Development Tools-SDK жасалған бағдарламаларды құрастыруға және реттеуге мүмкіндік беретін әзірлемелер үшін бірнеше құрал-саймандарды қамтиды;

– Sample Code-Android SDK кейбір Android мүмкіндіктерін көрсететін типтік қолданбаларды және кодыңызда API жеке ерекшеліктерін қалай пайдалану керектігін көрсететін қарапайым бағдарламаларды ұсынады.

Android қолданбаларын әзірлеу алдында API өзгерісін басқару платформасының жалпы тәсілін түсіну пайдалы. Сондай-ақ, Android API Level (API деңгейінің идентификаторы) және оның бағдарлама Орнатылатын құрылғылармен үйлесімділігін қамтамасыз етудегі рөлін түсіну маңызды.

API деңгейі-Android платформасының API нұсқасын анықтайтын бүтін сан. Платформа қамтамасыз етеді құрылымының API, қолданбалар қолдана алады өзара іс-қимыл үшін жүйе Android. Android платформасының әрбір келесі нұсқасы API жаңартуларын қамтуы мүмкін.

API-құрылым жаңартулары жаңа API ерте API нұсқаларымен үйлесімді болып қалатындай етіп жасалған. Осылайша, API-дегі өзгерістердің көпшілігі жиынтық болып табылады және жаңа функционалдық мүмкіндіктерді енгізеді немесе алдыңғы түзетеді. API бөлігі үнемі жаңартылып отырғандықтан, ескірген API пайдалану ұсынылмайды, бірақ қолда бар қосымшалармен үйлесімділікке байланысты жойылмайды.

Android қолданбасын пайдаланатын API деңгейі әрбір Android-қосымшаның конфигурация файлында көрсетілген бүтін идентификатормен анықталады.

3.4 Java құрылымы және өзгешеліктері

Java-1991 жылдан бастап Sun Microsystems компаниясы әзірлеген және 1995 жылдың 23 мамырында ресми түрде шығарылған программалаудың объектілі-бағытталған тілі. Бастапқыда жаңа бағдарламалау тілі Oak (James Gosling) деп аталды және тұрмыстық электроника үшін әзірленді, бірақ кейін Java деп аталды және апплеттерді, қосымшаларды және серверлік бағдарламалық қамтамасыз етуді жазу үшін қолданыла бастады.

Java бағдарламалары виртуалды java-машинада (JVM) - байт-кодты өңдеу және интерпретатор ретінде жабдыққа беретін нұсқаулықпен орындалатын байт-кодқа трансляциялануы мүмкін,бірақ байт-код мәтінге қарағанда анағұрлым жылдам өңделеді.

Java тілі түрлі тұрмыстық аспаптарға арналған озық бағдарламалық жасақтаманы құру жобасының бір бөлігі ретінде пайда болды. Жобаны іске асыру C++ тілінде басталды, бірақ көп ұзамай көптеген мәселелер пайда болды, олармен күресудің ең жақсы құралы — бағдарламалау тілінің өзгеруі болды. Әрбір архитектураға жеке компиляциялауға тура келмейтін және әр түрлі операциялық жүйелер астындағы әртүрлі процессорларда пайдалануға болатын бағдарламаларды жасауға мүмкіндік беретін платформа-тәуелсіз бағдарламалау тілі қажет екені айқын болды.

Java тілі Internet желісіне интерактивті өнімдерді жасау үшін қажет болды. Шын мәнінде, Java құру кезінде қабылданған көптеген архитектуралық шешімдер C және C++ұқсас синтаксисті ұсыну ниетімен

түсіндіріледі. Java-да айнымалыларды жариялау, параметрлерді беру, операторлар үшін және кодты орындау ағынын басқару үшін бірдей келісімдер қолданылады. Java-да с++барлық жақсы қасиеттері қосылған.

Үш негізгі элемент Java тілі технологиясына біріктірілді

Java кең пайдалану үшін өзінің апплеттерін (applets) ұсынады — шағын, сенімді, динамикалық, платформаға тәуелді емес, Web беттеріне қосылатын белсенді желілік қосымшалар. Java апплеттері кез келген HTML құжаттары сияқты оңай тұтынушыларға теңшеуге және таратуға болады

Java қарапайым және таныс синтаксисті жұмыста сенімді және ыңғайлы әзірлеу ортасымен үйлестіре отырып, қосымшалардың объектілі-бағытталған әзірлемелерінің қуатын босатады. Бұл бағдарламашылардың кең ауқымына жаңа бағдарламалар мен жаңа апплеттерді жылдам жасауға мүмкіндік береді

Java ұсынады программистке бай жинағы-сынып үшін объектілерді анық абстракциялау көптеген жүйелік функцияларды пайдаланылатын жұмыс кезінде терезесі желісімен және енгізу-шығару. Бұл кластардың негізгі ерекшелігі олар жүйелік интерфейстердің кең спектрі үшін пайдаланылатын абстракциялардың платформаларынан тәуелсіз құруды қамтамасыз етеді

Қазіргі уақытта Java тілі ең кең тараған және танымал бағдарламалау тілдерінің бірі болып табылады. Тілдің бірінші нұсқасы 1996 жылы Oracle компаниясы жұтқан Sun Microsystems компаниясының жер қойнауында пайда болды. Java түрлі есептер үшін қолдануға болатын әмбебап бағдарламалау тілі ретінде ойластырылған. Қазіргі уақытта Java тілі үлкен жолды жүріп, көптеген түрлі нұсқаларды басып шығарды. Ағымдағы нұсқа-Бұл ақпан айында шыққан Java 12. Ал Java тек әмбебап тілден бүкіл платформа мен экожүйеге айналды,ол бірқатар міндеттерді біріктіретін түрлі технологияларды: десктопты қосымшаларды құрудан бастап ірі веб-порталдар мен сервистерді жазуға дейін. Сонымен қатар, Java тілі қарапайым ДК, планшеттер, смартфондар мен ұялы телефондар және тіпті тұрмыстық техника сияқты бірқатар құрылғылар үшін бағдарламалық жасақтаманы жасау үшін белсенді қолданылады. Android мобильді ОЖ-ның танымалдығын есте сақтау жеткілікті, ол үшін көптеген бағдарламалар Java-да жазылған.

Java тілінің негізгі ерекшелігі оның коды алдымен платформадан тәуелсіз арнайы байт-кодқа таратылуы болып табылады. Содан кейін бұл байт-кодты JVM (Java Virtual Machine) виртуалды машинасымен орындайды. Бұл тұрғыда Java стандартты интерпретацияланатын тілдерден PHP немесе Perl сияқты ерекшеленеді, олардың коды бірден интерпретатормен орындалады. Сонымен қатар Java с немесе С++сияқты таза компиляцияланатын тіл емес.

Мұндай архитектура кроссплаторлықты және бағдарламаларды Java -ға аппараттық көшіруді қамтамасыз етеді, соның арқасында мұндай

бағдарламалар қайта қалпына келтірусіз түрлі платформаларда-Windows, Linux, Mac OS және т.б. орындалуы мүмкін.

Java C/C++ және C#сияқты синтаксис тіл болып табылады. Сондықтан, егер сіз осы тілдердің бірімен таныс болсаңыз, Java меңгеру оңай болады.

Java тағы бір басты ерекшелігі-ол қоқысты автоматты түрде жинауды қолдайды. Ал бұл сізге c++ сияқты бұрын пайдаланылған объектілерден жадты қолмен босатудың қажеті жоқ дегенді білдіреді, өйткені қоқыс жинаушы бұл сіз үшін автоматты түрде жасайды.

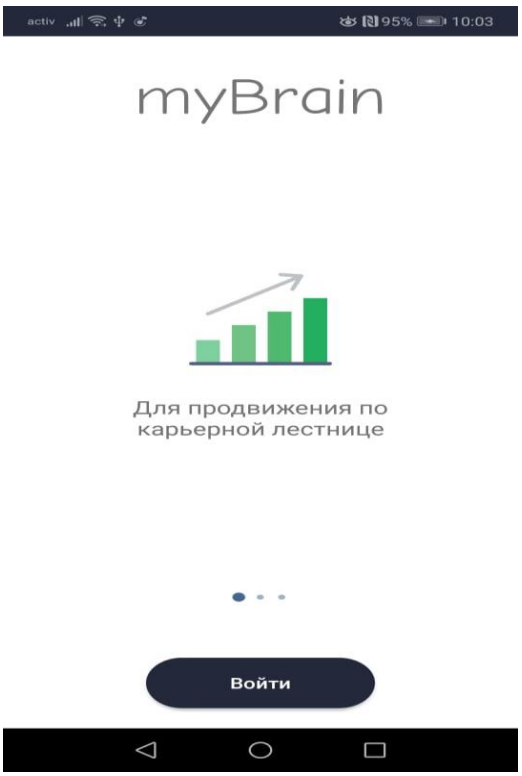
Java объектілі-бағытталған тіл болып табылады. Ол полиморфизм, мұра, статикалық типизацияны қолдайды. Объектілі-бағытталған тәсіл ірі, бірақ сонымен қатар икемді, масштабталатын және кеңейтілетін қосымшаларды құру бойынша міндеттерді шешуге мүмкіндік береді.

3.4 «MyBrain» мобильдік бағдарламасын құру және оны іске асыру

«MyBrain» - бұл пайдаланушыларға ыңғайлы болу үшін жасалған ағылшын тілі курстарының онлайн мобильді қосымшасы. Оның көмегімен клиенттер өздері үшін ыңғайлы кез-келген уақытта, оқу үшін қызықты материалдармен өте қолжетімді баға бойынша оқи алады.

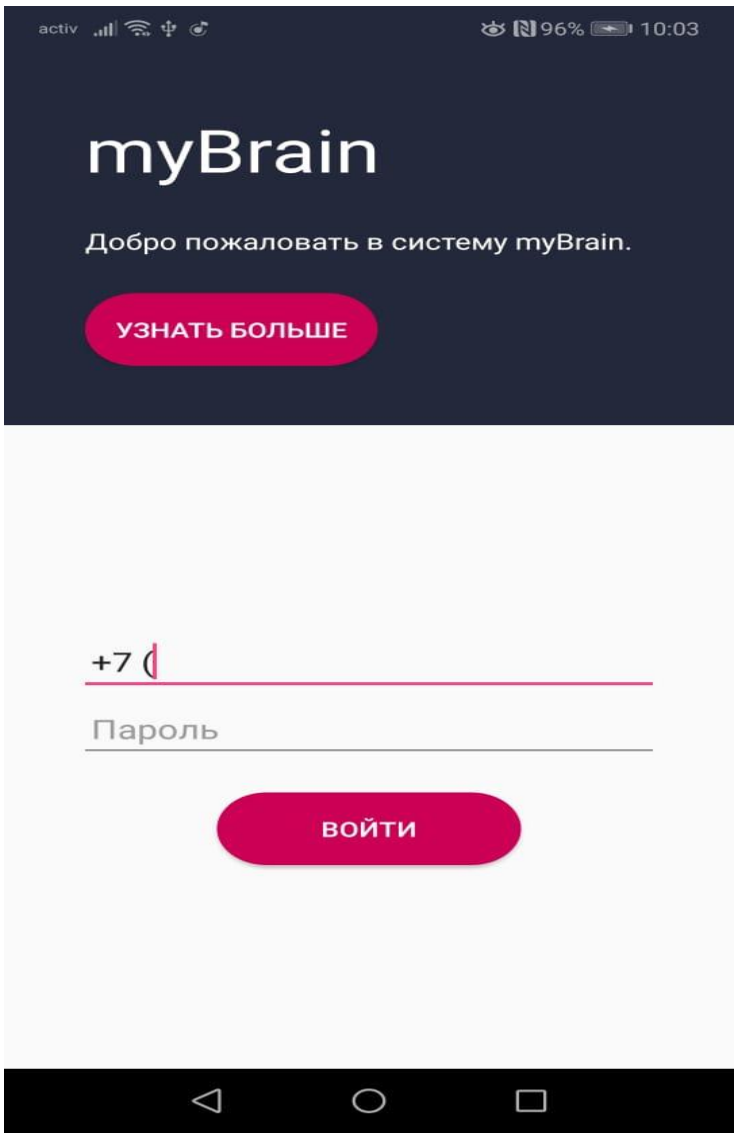
«MyBrain» мобильді қосымшасының өзектілігі кез-келген уақытта қолжетімділігі және ыңғайлылығы болғандықтан, бірінші ретте мобильді қосымшаның дизайн жағын қарадым. Мобильді қосымша бейнесі ыңғайлы түрде, түсінікті түрде жасалған.

MyBrain мобильді қосымшасын Android Studio платформасында жасадым. SQLite (Room) базасын қолдандым. Бағдарламалау тілі ретінде Java тілін пайдаландым. Төменде мобильді қосымшаның Flash экран бөлімі көрсетілген.



3.10-сурет – Flash Activity

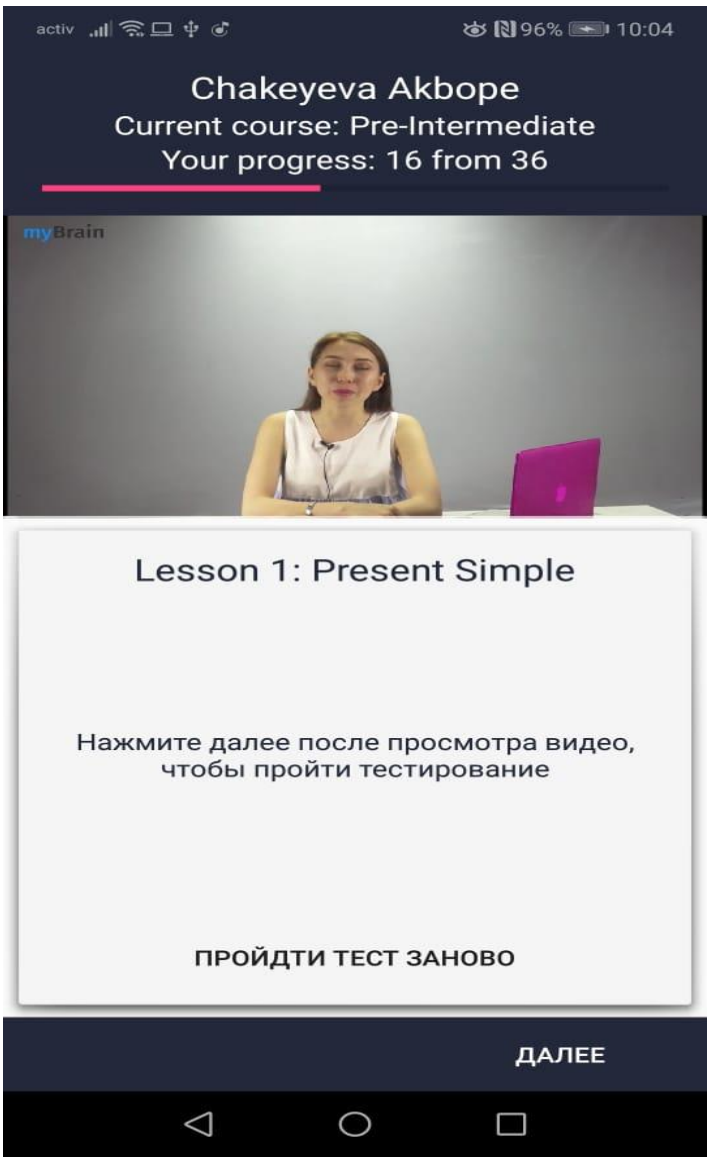
Flash Activity – алғашқы activity бболып табылады. Бұл экрандағы «MyBrain» қосымшасының өзектілігі суреттелген бетті Fragment-тердің ViewPager арқылы ауысуы түрінде жасадым. Яғни, ViewPager-дің көмегімен Flash Activity-дағы мәліметтер экранды жылжытқан реттеп ауысып тұрады. Осыдан кейін «Войти» батырмасын басатын болсақ, кіру бөліміне өтеміз. Бұл жерде телефон номер енгізетін жерде `com.redmadrobot:inputmask:2.3.0` арнайы кітапханасын қолдандым. Төменде кіру бөлімі көрсетілген.



3.11-сурет – Кіру бөлімі

Және де бұл жерде «Узнать больше» батырмасын басатын болсақ, «MyBrain» порталына өтетін болды. Сол жерден толық мәлімет көрсек болды. Осылай батырманы басқан кезде өтуі үшін, `browserIntent` функциясын қолдандым. Порталдың ссылкасын `Uri.parse(`**https://mybrain.kz/** арқылы бердім.

Келесі бет бізде видео-сабақ қарау бөлімі. Бұл жерде видео қосылуы үшін, **`cn.jzvd:jiaozivideoplayer:6.2.9`** кітапханасын қолдандым. Осы кітапхананы қолданғаннан кейін видео ыңғайлы түрде көрінеді. Мысалға, телефонды қалай қозғалсақ, сол бағытта видеода қозғалады. Соның арқасында видеоны үлкен режимде керек жерінде тоқтатып көрсе болады. Android мобильді қосымшасын жасау барысында ыңғайлы кітапханалар қолданған дұрыс, себебі, Android Studio-да кітіапханалардың көмегімен жететін артықшылықтар кездесе бермейді. Төменде осы видео-қарау беті көрсетілген.



3.12-сурет – Видео-сабақ қарау беті

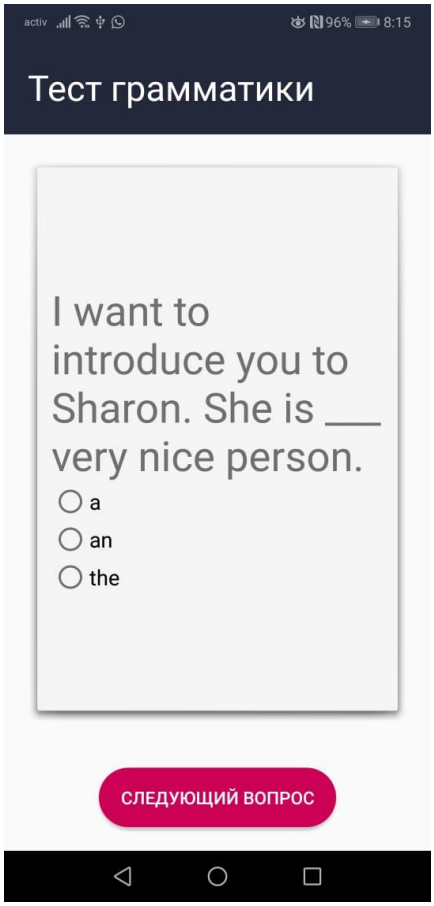
Видео-қарау бетінің төменгі жағында «далее» батырмасын басып тестілеу бөліміне өтеміз.

Бұл жерде, fragment size бойынша тест сұрақтары ауысып отырады. Яғни, базадан қанша сұрақ келсе сонша fragment беті ауысып тұрады. Fragment Activity-ге көмекші бет ретінде қолданылады. Егерде көлемі көп мобильді қосымша жасау керек болса Fragment қолданған дұрыс, себебі, Activity көлему ауыр болып есептеледі. Және де осылай әр тест әр бетте шығып тұруы үшін Activity қолданатын болсақ, қанша тест бар сонша Activity құру керек болады.

Ал Fragment қолданғанда тек бір Fragment құрамыз. Сол себепті, көлемі жеңіл болады. Соңғы сұраққа келгенде «Следующий вопрос» «Следующий этап» сөзіне ауысады. Бұл жерде батырма оуыспайды. Тек арнай кодпен жазып сөзді ауысады. Мысалы,

```
        grammarButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        viewPagerGrammar.setCurrentItem(getItem(+1), true);

        if (viewPagerGrammar.getCurrentItem()==fragments.size()-1){
            grammarButton.setText("Следующий этап");
            grammarButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {

                    Intent inte = new Intent(TestActivity.this,
WordsNewActivity.class);
                    startActivity(inte);
                }
            });
```

Төменде осы тест бөлімі келтірілген.

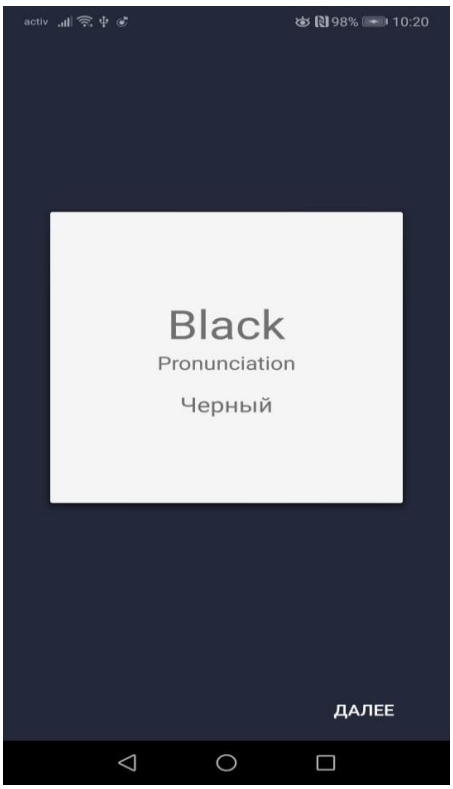


3.13-сурет – Сұрақтар беті

3.14-сурет – Сұрақтар беті

Келесі этап, сөздер бөлімі. Бұл жерде де fragment size бойынша сөздер ауысып отырады. Яғни, базадан қанша сөз келсе сонша fragment беті ауысып тұрады.

```
    public static WordsPageFragment newInstance(String word, String
description){
    WordsPageFragment fragment = new WordsPageFragment();

    Bundle args = new Bundle();

    args.putString(ARG_WORDS, word);
    args.putString(ARG_DESCRIPTION, description);
    fragment.setArguments(args);
    return fragment;
}


@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    if (getArguments() != null) {
        word = getArguments().getString(ARG_WORDS);
        description = getArguments().getString(ARG_DESCRIPTION);
    }
}
```



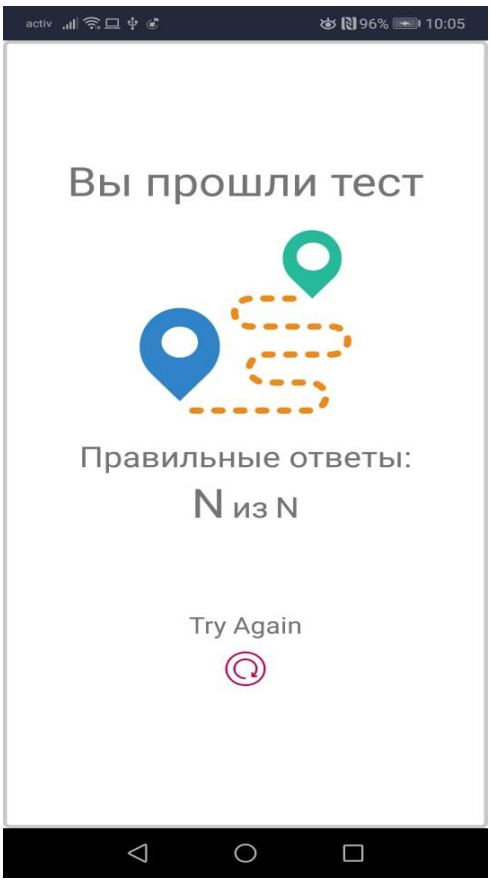
3.15-сурет – Сөздер беті

Келесі сөздер бетінен кейін, осы жаттауға берілген сөздердің бірі тест ретінде келеді. Келген тест дұрыс белгілен сөзде, correct сөзі шығады.

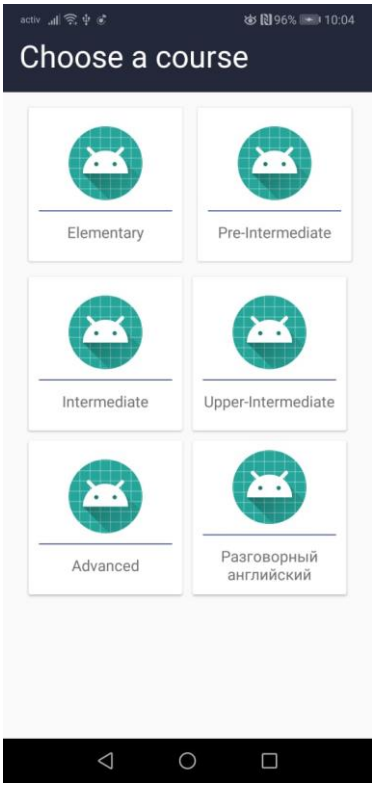
```
      button = (Button) view.findViewById(R.id.translateFirstButton);

button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        button.setText("Correct!");
        button.setTextColor(getResources().getColor(R.color.chromeGreen));

    }
});
```

Төменде соңғы Finish Activity келтірілген. Бұл жерден сабақты біткеннен кейін, нәтижені көріп қайтадан деңгей таңдау бөліміне өтсе болады.

```
      exit = (ImageButton) findViewById(R.id.exit);
exit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new
Intent(FinishActivity.this,CardViewPageActivity.class);
        startActivity(intent);
    }
});
```



3.16-сурет – Finish Activity

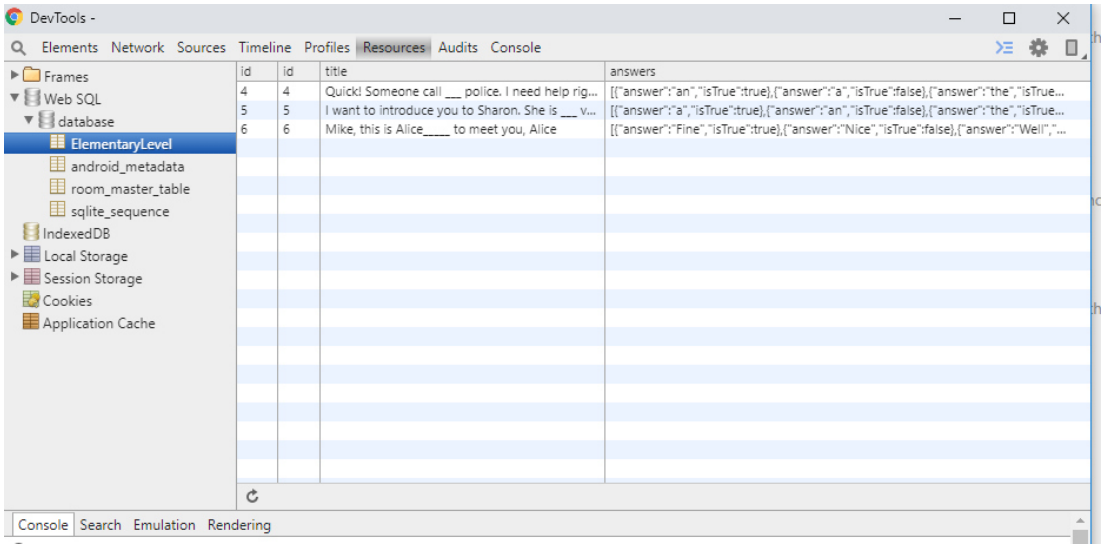


3.17-сурет – Деңгей таңдау беті.

Деңгей таңдау беті – CardViewPageActivity. Бұл жерде widget.CardView қолданамыз. Бұл дизайн болып табылады. Бұл жерде де com.android.support:cardview-v7:26.1.0 кітапханасын қолдану арқылы widget.CardView-тін пайдалана аламыз.

```
public class CardViewPageActivity extends AppCompatActivity {

    private CardView cardView;

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

        cardView = (CardView) findViewById(R.id.firstCardView);

        cardView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(CardViewPageActivity.this,
VideoPageActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

Келесі төменде Chrome Inspect терезесі көрсетілген.



3.18-сурет – Chrome Inspect

Chrome Inspect SQlite базасын көруге арналған терезе. Оның көмегімен базада қандай мәліметтер барын көре аламыз. Мен бұл жерде Room оберкасын қолдандым. Бұл жерде де, кітапхана қоладанамыз.

```
    implementation "android.arch.persistence.room:runtime:1.1.1"
implementation "android.arch.persistence.room:rxjava2:1.1.1"

annotationProcessor "android.arch.persistence.room:compiler:1.1.1"
```

```
implementation                          'io.reactivex.rxjava2:rxandroid:2.1.1'
implementation 'io.reactivex.rxjava2:rxjava:2.2.8'
```

Room үш негізгі компонентке ие: Entity, Dao және Database. Room-мен жұмыс істеу кезінде бізге SQL сұрау жазу қажет болады.

Room - бұл жаңа Android Architecture бөлігі, жергілікті қосымшалардың архитектурасын қолдайтын Google кітапханалары тобы. Room Realm, ORMLite, GreenDao және көптеген басқа балама ретінде ұсынылады.

Room-бұл Android-та орнатылған SQLite төменгі деңгейлі байламдарға арналған жоғары деңгейлі интерфейс. Ол SQLite API үстінен API құру арқылы компиляция кезінде өз жұмысының көп бөлігін орындайды, сондықтан сіз Cursor немесе ContentResolver-мен жұмыс істеудің қажеті жоқ.

4 Экономикалық бөлім

Бұл дипломдық жобада ағылшын тілін үйрену үшін "My Brain" мобильді қосымшасы әзірленді. Мобильді қосымша ағылшын тілін онлайн үйрену бойыншы операциялардың орындалуының еңбек сыйымдылығын азайтуға арналған.

Мобильді қосымшаның тиімділігі оның кез-келген уақытта, кез-келген жерде пайдалана алуында.

Еңбек шығындарын азайту қаржы шығындарын азайтуға мүмкіндік береді,бұл жалпы өнімділік пен үнемдеуді арттыруға әкеледі.

Экономикалық бөлімнің негізгі міндеті зерттеу жүргізу шығындарының шамасын, осы дипломдық жұмыста қойылған техникалық міндеттерді шешу кезінде алынатын негізгі және ілеспе нәтижелерді қоғамдық өндірісте пайдаланудан экономикалық тиімділікті анықтау үшін өзіндік құнын анықтау болып табылады. Қабылданған ғылыми-техникалық шешімнің тиімділігін бағалау барлық қажетті шығындар мен шығындарды ескеруі тиіс, ол үшін белгілі бір схема бойынша бірқатар қажетті есептеулер жүргізу талап етіледі.

4.1 Мобильді қосымшаны әзірлеудің еңбек сыйымдылығы

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	15
2 кезең	Деректерді өңдеу	5
3 кезең	Деректер қорының құрылымын құру	14
4 кезең	Алгоритмді әзірлеу	45
5 кезең	Бағдарламаны жазу	53
6 кезең	Бағдарламаны жөндеу	110
7 кезең	Бағдарламамен жұмыс істеу бойынша басшылық дайындау	105
8 кезең	Анықтамалық жүйені әзірлеу	18
НӘТИЖЕ Бағдарламалық өнімді орындаудың еңбек сыйымдылығы		365

4.2 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.

2-Кесте – Материалдық ресурстарға шығындар

Материалдық ресурстың атауы	Бірлік өлшем	Саны	Бірлік бағасы, тг	Сомасы, тг
Ноутбук Acer Aspire E5-575G	Штук	1	230000	230000
НӘТИЖЕ Материалдық ресурстарға шығындар жиыны				230000

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) мынадай формула бойынша есептеледі:

$$З_э = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц \quad (1.2)$$

мұндағы Мі-І электр жабдығының паспорттық қуаты, кВт;

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

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

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

І-электр жабдығының түрі;

n-электр жабдықтарының саны.

$$З_э = 0.9 \cdot 0.8 \cdot 365 \cdot 18.32 = 4814,5$$
$$З_э = 0.3 \cdot 0.7 \cdot 365 \cdot 18.32 = 1404,2$$

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 Acer Aspire E5-575G	0.9	0.8	365	18,32	4814,5
Жарықтандыру	0,3	0,7	365	18,32	1404,2
НӘТИЖЕ электр энергиясына арналған шығындар					6218,7

4.3 Мобильді қосымшаны әзірлеуге арналған шығындарды есептеу

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

Қызметкердің бір айдағы жұмыс сағаттары мынадай формула бойынша анықталады:

$$Ч_м = N_м \cdot Ч_{рд} ,$$

мұнда, 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 мына формула бойынша есептелуі мүмкін:

$$ЧС = \frac{ЗП_i}{ФРВ_i}, \quad (1.4)$$

Инженер-разработчик (Әзірлеуші-инженер):

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

Программист:

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

мұнда ЗПі-і қызметкердің айлық жалақысы, тг;

Фрв-і қызметкердің жұмыс уақытының айлық қоры, сағ.

Мобильді қосымшаы әзірлеудің еңбек сыйымдылығын анықтау үшін 1.1 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)

$$\text{Т}_1 = 15 + 5 + 14 + 45 + 53 + 105 + 18 = 255$$

Программисттің мобильді қосымшаны әзірлеу еңбек сыйымдылығы 220 адамға тең. (тапсырма қою, бағдарлама модульдерін әзірлеу, жүйені тестілеу).

$$\text{Т}_1 = 5 + 110 + 105 = 220$$

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

$$\text{З}_\text{э} = \sum_{i=1}^{n} \text{ЧС}_i \times T_i \text{ , (1.3)}$$

мұнда 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):

$$\text{З}_\text{тр} = 1071{,}43 \cdot 255 = 273214{,}65 \text{ тг}$$

Программист:

$$\text{З}_\text{тр} = 892{,}86 \cdot 220 = 196429{,}2 \text{ тг}$$

Жалпы сомасы:

$$З_{тр.о} = 273214,65 + 196429,2 = 469643,85 \text{ тг}$$

4-Кесте- Еңбекақы төлеу шығындары

Квалификация	Мобильді қосымша әзірлеудің еңбек сыйымдылығы, адам×сағ	Сағаттық ставка, тг / сағ	Сомасы, тг
Инженер-разработчик	255	1071,43	273214,65
Программист	220	892,86	196429,2
НӘТИЖЕ Еңбекақы төлеу шығындары			469643,85

Қосымша еңбекақы:

$$З_{доп} = З_{тр} \cdot 10\%$$
$$З_{доп} = 469643,85 \cdot 0,1 = 46964,385 \text{ тг}$$

Еңбекақы қоры:

$$Ф_{зп} = З_{тр.о} + З_{доп}$$
$$Ф_{зп} = 469643,85 + 46964,385 = 516608,235 \text{ тг}$$

Әлеуметтік салықты есептеу:

$$Н_с = (Ф_{зп} - ОПВ) \cdot 11\%$$

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

$$Н_с = \left(516608,235 - (516608,235 \cdot 0.1)\right) \cdot 0.11 = 51144,2153 \text{ тг}$$

Негізгі қорлардың амортизациясын есептеу:

Амортизациялық аударымдардың жалпы сомасы мынадай формула бойынша анықталады:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times H_{Ai} \times T_{НИРi}}{100 \times T_{Эфi}}, \quad (1.5)$$

мұнда 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;
І-ОФ түрі;n - количество ОФ.

Расчет годовой нормы амортизации ОФ

Құрылғылар:

$$H_{Ai} = \frac{100}{T_{Ni}} (1.6)$$

$$\text{Н}_{Ai} = \frac{100}{4} = 25$$

мұнда 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).

$$\text{Т}_1 = 45 + 53 + 110 + 105 = 313$$

Жабдық:

$$\text{З}_{\text{АМ}} = \frac{230000 \cdot 25 \cdot 365}{100 \cdot 1920} = 10930{,}99 \text{ тг}$$

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 Acer Aspire E5-575G	230000	25	1920	365	10930,99
ОС Windows 10	Тегін	-	1920	365	-
SQLite	Тегін	-	1920	313	-
НӘТИЖЕ					10930,99

6-Кесте - Мобильді қосымша әзірлеуге арналған шығындар сметасы

Шығындар баптары	Сомасы, тг

1. Материалдық шығындар, оның ішінде: - материалдар - электр энергиясы 2. Еңбекке ақы төлеуге арналған шығындар. 3. Негізгі қорлардың амортизациясы. 4. Басқа шығындар.	230000 6218,7 469643,85 10930,99 0
НӘТИЖЕ	716793,54

4.1-сурет – Шығындар диаграммасы

4.4 Мобильді қосымшаның ықтимал (шарттық) бағасын анықтау

Шарттық бағаны есептеу ($Ц_д$) :

$$Ц_д = З_{НИР} \cdot \left(1 + \frac{Р}{100}\right), \quad (1.7)$$

онда ЗНИР - әзірлеуге арналған шығындар ПП (кесте 1.6), тг;
Р - ПП табыстылығының орташа деңгейі - 20% .
Ц_д=702 704,13·(1+0.2)=843244,956 тг
2019 жылға ҚҚС ставкасы 12% мөлшерінде белгіленген.
ҚҚС есебімен өткізу бағасы мынадай формула бойынша есептеледі:

Цр=Цд + Цд * НДС.

$$Ц_р = 843244{,}956 + 843244{,}956 \cdot 0.12 = 944434{,}35 \text{ тг}$$

Мобильді қосымшаның экономикалық бөлімі бойынша қорытынды

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 702 704,13 теңгеге тең,

Рентабельділік= 944 434,35 теңге

Мобильді қосымшаның рентабельділігі= 140 540,826 теңге

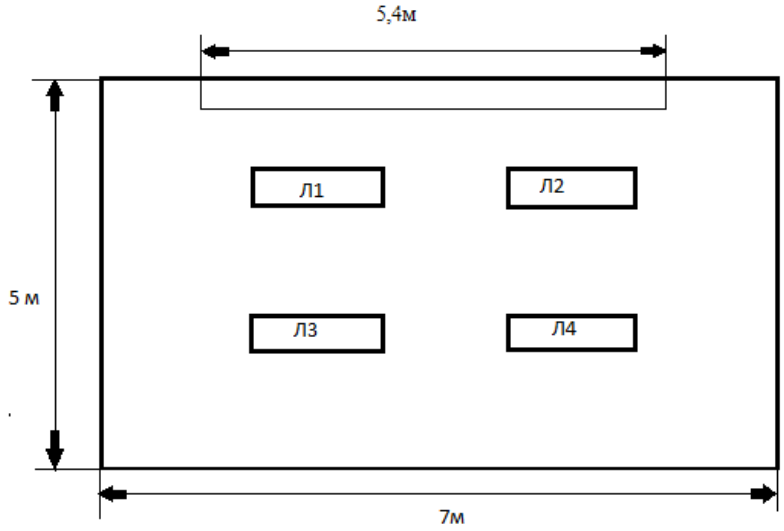
5 Өмір тіршілік қауіпсіздік бөлімі

Дипломдық жобаның тақырыбы Android платформасында my Brain 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 (1-сурет).



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 мынадай формула бойынша бүйірлік жарықтандыру кезінде жарық ойықтарының ауданын алдын ала анықтаудан тұрады.

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

мұнда S_0 – бүйірлік жарықтандыру кезіндегі Жарық ойықтарының ауданы, м2;

S_n-үй-жай еденінің ауданы, м2;

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 противостоящими ғимараттар қабылдайды 3.8-кесте бойынша [14].

Еден ауданы мынадай формула бойынша анықталады

$$S_n = L * B \qquad (5.2)$$

Осылайша, (3.2) 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 мына формула бойынша есептеледі

$$e_N = e_н * m_N \qquad (5.3)$$

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

e_н = 1,2 – коэффициент мәні табиғи жарықтандыру, выбираемое Қнже 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)мына формула бойынша анықталады

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

мұнда 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 $\tau_2 = 0.75$. (5.4-кесте) [14].

τ_3 – тіреу конструкцияларындағы жарықтың жоғалуын ескеретін коэффициент, бүйірлік жарықтандыру кезінде 1-ге тең (5.5-кесте) [14];

τ_4 = 1 – күннен қорғайтын құрылғылардағы жарықтың жоғалуын ескеретін коэффициент,(5.6-кесте) [14];

3.4 формуласына белгілі мәндерді қойып, τ_0 табамыз.

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

кесте бойынша жарықтану үшін r_1 коэффициентін Ол үшін біз табамыз:

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, формула бойынша (5.5)

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

Алынған мәндер r_1 мәндерімен кестедегі мәндерге сәйкес келмегендіктен, интерполяцияны өткіземіз

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
Свыше 2,5 до 3,5	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

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

сонда $r_1 = 1.58$

Барлық табылған мәндерді орналастыра отырып, (3.1) формуладан бүйірлік жарықтандыру кезінде жарық ойықтарының ауданын S_0табамыз

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

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]. Для определения количества светильников используем формулу [15]

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

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

n = 2-Шамдағы шамдардың саны;

бөлменің индексіне және шағылысу коэффициенттеріне байланысты пайдалану коэффициенті.

К_з-қор коэффициенті. Шаң бөлінбейтін Үй-жайлар үшін К_з= 1,3 (7-қосымша) [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]:

$$i = \frac{S_п}{(L+B)h_{расч}} = \frac{A*B}{(L+B)h_{расч}} \qquad (3.10)$$

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

Барлық мәндерді формулаға қойып (3.10), аламыз:

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

Д [15] арқылы шам тобын анықтаймыз – 1.

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

$$Ф_Л = \frac{E_{min} \cdot S \cdot Z \cdot K_з}{N \cdot \eta \cdot n} \quad (11)$$

мұнда 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;

N-Шамдағы шамдардың саны; 10248

h - бірлік үлесіндегі жарық ағынын пайдалану коэффициенті.

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

$$E_{min} = \frac{Ф_л * N * n * \eta_З}{S * Z * K} = \frac{3570 * 4 * 2 * 0.42}{35 * 1.1 * 1.3} = 240$$

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.

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

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

Бұл жағдайда, жарық ағыны мен жарық ағыны Ф = 4070 лм болатын 6 жарық шамдары бар 12 шам қажет.(3.9) 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]

$$h_{\text{расч}} = H_{\text{пом}} - (H_{\text{св}} + H_{\text{р.п.}}), \qquad (3.8)$$

мұнда н_св = 0 – шам шамдардың шамдардың шамының биіктігі, м;
Н_ (р. с.) = 0,8 – еден үстіндегі жұмыс бетінің қашықтығы, м;
Н_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:

$$\text{L} = \lambda * \text{h} \qquad (5.9)$$

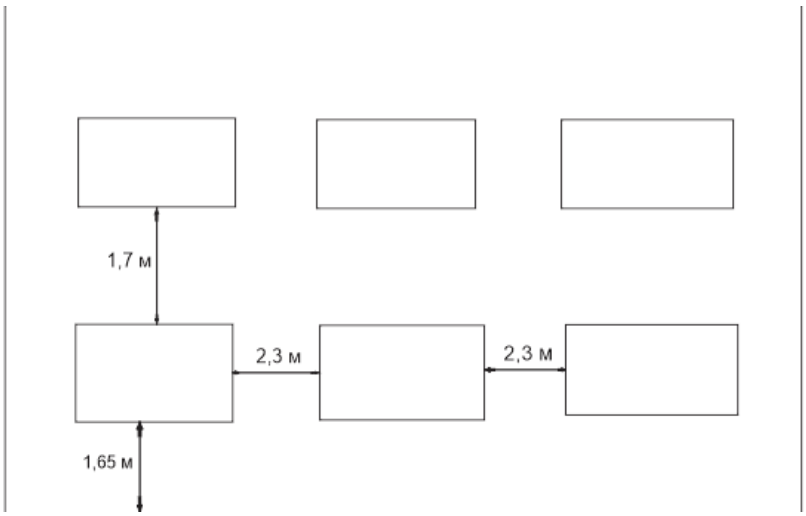
мұнда L-көрші шамдардың немесе люминесцентті шамдардың қатарлары арасындағы қашықтық;
h-жұмыс бетінен шамды ілу биіктігі.
Люминесцентті шамдары бар шамдарды пайдаланған кезде
λ=0,45÷2.4 [15].
Осылайша, шамдар арасындағы қажетті қашықтық

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

Шамдардың қатарлары арасындағы қашықтық:
Lb=λ*hp=1,7·1 = 1,7 м.
Шам мен қабырға арасындағы қашықтық:

la=lb=Lb/3 + [1,07;1,08] =1,7/3+1,08 =1,65 м.

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



Сурет 5.2-бөлмедегі шамдар мен жарық ойықтарының орналасуы

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.