Қазақстан Республикасы Білім және Ғылым Министрлігі

Л. Н. Гумилев атындағы Еуразия ұлттық университеті

Сарсекенов Бексултан

Жобаны басқаруға арналған көп қолданушылы қосымша әзірлеу

ДИПЛОМДЫҚ ЖҰМЫС

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

Нұр-Сұлтан 2022

Қазақстан Республикасы Білім және Ғылым Министрлігі

Л. Н. Гумилев атындағы Еуразия ұлттық университеті

«Қорғауға жіберілді»

Компьютерлік және

программалық инженерия

кафедрасының меңгерушісі

т.ғ.к., доцент м.а.

Дюсекеев К.А. \_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_20\_\_\_ж.

**ДИПЛОМДЫҚ ЖҰМЫС**

**Тақырыбы: «Жобаны басқаруға арналған көп қолданушылы қосымша әзірлеу»**

5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандығы бойынша

|  |  |
| --- | --- |
| Орындады: | Сарсекенов Б.Б |
|  |  |
| Ғылыми жетекшісі: PhD, доцент м.а. | Жартыбаева М.Г. |

Нұр-Сұлтан 2022

Л. Н. Гумилев атындағы Еуразия ұлттық университеті

Ақпараттық технологиялар факультеті

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

«Компьютерлік және программалық инженерия» кафедрасы

**Бекітемін**

Компьютерлік және

программалық инженерия

кафедрасының меңгерушісі

т.ғ.к., доцент м.а.

Дюсекеев К.А.\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_20\_\_\_ж.

**Диплом жұмысын орындауға**

**ТАПСЫРМА**

В4-70400-03 тобының, «5В070400 - Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандығының, күндізгі бөлім студенті Сарсекенов Бексултан.

1. Дипломдық жұмыс тақырыбы: «Жобаны басқаруға арналған көп қолданушылы қосымша әзірлеу» университет бойынша «14» қантар 2022ж. № 47-п ректорының бұйрығымен бекітілген.
2. Студенттің орындалған дипломдық жұмысты тапсыру уақыты «23»мамыр 2021 ж. // Бекжан Ағай
3. Жұмыс істеу үшін бастапқы мәліметтер (заңдар, әдебиеттер, зертханалық өндіріс туралы мәліметтер): Android Developers Dashboard, Unity 3d бағдарламасы. VisualStudio, ASP.NET Xamarin.Forms .NET Entity Framework MSSQL Server , Xamarin.Essentials, Xamarin Community Toolkit … //
4. Дипломдық жұмыста қарастырылатын сұрақтар тізімі:

* жобаларды басқару құрудың алғышарттары;
* жобаларды басқару жүйесін құруға қажет құралдарға шолу және деректерді жинау;
* қосымшаны жобалау;
* жобаны басқаруға арналған көп қолданушылы қосымшаны қолданысқа енгізу.

1. Графикалық құжаттар тізімі (сызбалар, кестелер, диаграммалар және т.б.):

* <https://docs.microsoft.com> сайтындағы бағдарламалық қамтамалар мен құралдардың құжаттамаларындағы сызбалар мен кескіндер ;
* Жобаны басқаруға арналған көп қолданушылы қосымшаның функционалдық және архитектруалық сұлбалары,UML диаграммалары;
* Мәліметтер базасының құрылымдық сұлбалары, кестелер;
* Жобаны басқаруға арналған көп қолданушылы қосымшаның интерфейсі;
* SWOT талдау;

1. Ұсынылатын негізгі әдебиеттер тізімі:

* Макеффри М. Unreal Engine VR для разработчиков (Мировой компьютерный бестселлер) – 2019;
* Виртуальная реальность в Unity. / Пер. с англ. Рагимов Р. Н. – М.: ДМК Пресс, 2016. – 316 с.: ил.;
* Android Developers Dashboard [Электрондық ресурс]. Қолжетімділік режимі: *https://developer.android.com/about/dashboards/index.html* (Сұраныс күні: 08.03.2021).
* Android Development Tool [Электрондық ресурс]. Қолжетімділік режимі: *https://developer.android.com* (Сұраныс күні: 18.02.2021).
* Django documentation [Электрондық ресурс]. Қолжетімділік режимі: *https://docs.djangoproject.com* (Сұраныс күні: 19.02.2021).

1. Жұмыс бойынша кеңестер (оларға қатысты жұмыс бөлімдерін көрсете отырып)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер, бөлімнің, тараудың атауы | Ғылыми жетекші, консультант | Тапсырма алу мерзімі | Тапсырманы берді (қолы) | Тапсырманы қабылдады (қолы) |
| Мазмұны | Жартыбаева М.Г. | 22.11.2021-10.01.2022 |  |  |
| Кіріспе | Жартыбаева М.Г. | 22.11.2021-10.01.2022 |  |  |
| Әдебиеттер  тізімі | Жартыбаева М.Г. | 22.11.2021-10.01.2022 |  |  |
| 1 тарау | Жартыбаева М.Г. | 11.01.2022-04.02.2022 |  |  |
| 2 тарау | Жартыбаева М.Г. | 04.02.2022-05.03.2022 |  |  |
| 3 тарау | Жартыбаева М.Г. | 05.03.2022-08.04.2022 |  |  |
| 4 тарау | Жартыбаева М.Г. | 09.04.2022-24.04.2022 |  |  |
| Қорытынды | Жартыбаева М.Г. | 24.04.2022-25.04.2022 |  |  |

8. Дипломдық жұмысты орындау кестесі//Бекжан Ағай

|  |  |  |  |
| --- | --- | --- | --- |
| № | Жұмыс кезеңдері | Жұмыстың кезеңдерінің орындалу мерзімі | Ескерту |
| 1 | Дипломдық жұмыстың тақырыбын бекіту | 22.11.2021 |  |
| 2 | Дипломдық жұмысты даярлау үшін материал жинау | 22.11.2021 |  |
| 3 | Дипломдық жұмыстың аналитикалық бөлімін дайындау (1 тарау) | 11.01.2022 | Практика алдында |
| 4 | Дипломдық жұмыстың жобалық бөлімін дайындау (2 тарау) | 04.02.2022 | Практика кезінде |
| 5 | Дипломдық жұмыстың экономикалық бөлімін дайындау (3 тарау) | 05.03.2022 | Практика кезінде |
| 6 | Дипломдық жұмыстың еңбекті жане қоршаған орта бөлімін дайындау (4 тарау) | 24.04.2022 | Практика кезінде |
| 7 | Дипломдық жұмыстың толық мәтінінің алдынғы нұсқасын аяқтау | 24.04.2022 | Практикадан кейінгі бірінші аптада |
| 8 | Дипломдық жұмысты алдын-ала қорғауға тапсыру | 25.04.2022 | Диплом алдындағы практика |
| 9 | Дипломдық жұмысты пікірлеме алуға тапсыру |  |  |
| 10 | Дипломдық жұмыстың соңғы нұсқасын ғылыми жетекшінің пікірімен тапсыру |  |  |
| 11 | Дипломдық жұмысты қорғау | 02.06.2021 |  |

Тапсырманың берілген күні «22» желтоқсан 2020 ж.//Бекжан

Ғылыми жетекшісі

PhD, доцент м.а. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жартыбаева М.Г.

Тапсырманы қабылдады студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сарсекенов Б.

МАЗМҰНЫ

[Кіріспе 7](#_Toc103959167)

[1 Графикалық мобильді қосымшаларды әзірлеу алғышарттары 9](#_Toc103959168)

[**1.1 Жоба дегеніміз не ?** 9](#_Toc103959169)

[**1.2 Жобаны басқару кезендері** 13](#_Toc103959170)

[**1.3 Жобаның мақсаты мен тапсырмаларын аныктау** 15](#_Toc103959171)

[**1.4 Жоба тобы және қаржылық бөлігі** 18](#_Toc103959172)

[2 Жобаларды басқаруға арналған қосымшаны әзірлеуде қолданылған құралдарға шолу 20](#_Toc103959173)

[**2.1 ASP.NET платформасына шолу** 20](#_Toc103959174)

[**2.2 ASP.NET Identity** 22](#_Toc103959175)

[**2.3 ASP.NET Web – API 2 Individual Accounts шаблоны** 26](#_Toc103959176)

[**2.4 Xamarin.Forms** 28](#_Toc103959177)

[**2.5 MVVM паттрені** 33](#_Toc103959178)

[3 Жобаны бағдарламалау 35](#_Toc103959179)

[**3.1 Жобаларды басқаруға арналған көпқолданушылы қосымша үшін сервер құру** 35](#_Toc103959180)

[**3.2 Жобаны басқаруға арналған көпқолданушылы қосымшаның клиент қосымшасы** 39](#_Toc103959181)

[**3.3 Жобаны басқаруға арналған көпқолданушылы қосымшаның архитектурасына шолу.** 44](#_Toc103959182)

[**4 Жобаны қолданысқа енгізу** 47](#_Toc103959183)

[Қорытынды 48](#_Toc103959184)

[Пайдаланылған әдебиеттер тізімі 50](#_Toc103959185)

# Кіріспе

Ақпараттық ресурстар, жүйелер мен технологиялар адамның қазіргі өмірінің ажырамас, қарқынды дамып келе жатқан элементтері. Бұған шетелдік және отандық компаниялардың статистикасы мен зерттеулері дәлел. 1997 жылы ұялы байланыс нарығында web технологиясы компьютерге қосылу үшін кабельді пайдаланбай, тікелей интернеттен ұялы телефондарға бағдарламалар орнатуға мүмкіндік берді. Осы уақыттан бастап қоғамды "жұмылдыру" процесі басталды. Мобильді қосымшаларды жасаудағы сапалы серпіліс 2000 жылдардың басында үлкен сенсорлық экрандары бар мобильді құрылғылардың өндірісі болды, бұл жаңа деңгейдегі Мобильді қосымшаларды құруға мүмкіндік берді. 2010 жылы мобильді құрылғылар Мобильді қосымшаларды әзірлеу үшін заманауи ақпараттық технологияларды қолдануға мүмкіндік беретін қуатты процессорлармен жабдықталған. Көптеген бағдарламашылар кәсіби қызметтің жаңа бағыттарын игере бастады және нарықтық тенденцияларға бейімделе бастады

Мобильді қосымшаларды әзірлеу қазіргі уақытта ақпараттық технологиялар саласындағы ең танымал міндеттердің бірі болып табылады. Мобильді даму тұтынушылардың қалауын көрсететін қосымшаларды құруға бағытталған; шешу алгоритмі алдын-ала белгісіз мәселелерді шешіңіз.Енді қосымшалар бірнеше көздерден алынған ақпаратқа аналитикалық талдау жасай алады және қолданушыға шешім қабылдауға, процестерді бақылауға және басқа да маңызды мәселелерді өзі үшін ең аз уақыт пен аналитикалық шығындармен шешуге көмектеседі. Бұл өз кезегінде бизнес-процестерді оңтайландыруға, шешімдердің өнімділігі мен тиімділігін арттыруға ықпал етеді.

Мобильді қосымша - белгілі бір платформада орнатылған, белгілі бір функционалдылыққа ие, белгілі бір әрекеттерді орындауға мүмкіндік беретін телефон үшін арнайы бағдарлама. Бұл қолданушыға әртүрлі ақпаратпен өзара әрекеттесуге көмектесетін адаптердің бір түрі.

Егер бұрын тұтынушылардың бизнеспен онлайн-өзара әрекеттесуінің негізгі арнасы веб-сайттар болса, бүгінде оларды мобильді қосымшалар тез ығыстыруда. Бұл таңқаларлық емес, өйткені пайдаланушы аудиториясы тұрақты компьютерлерден смартфондар мен планшеттерге жүйелі түрде ағып келеді. Осыған байланысты кәсіпкерлер мобильді ортада өкілдік құру туралы қамқорлық жасауы керек. IOS, Android және Windows Phone қосымшалары-бұл бизнесті алға жылжытудың перспективалы арнасы, аудиторияның жаңа сегментін тартудың тиімді әдісі, қазіргі тұтынушыларға қамқорлық көрсету және олармен өзара әрекеттесу мүмкіндіктерін кеңейту.

Жергілікті қосымшалар белгілі бір платформаға арналған бағдарламалау тілдерінде жазылады және операциялық жүйеге енеді, тез және дұрыс жұмыс істейді, басқа мобильді қосымшалардың функционалдығы мен жылдамдығы жағынан артықшылығы бар. Олар интерфейсті және бағдарламаның жалпы әрекетін осы платформа үшін ең табиғи түрде жүзеге асыруға мүмкіндік береді. Қосымшалар сонымен қатар құрылғылардың аппараттық құралдарына қол жеткізе алады: камера, микрофон, акселерометр, телефон кітапшасы және т.б., ресурстарды үнемді пайдаланады, Интернет байланысы ажыратылған кезде толық немесе ішінара жұмыс істейді. Әрине, олар әзірлеушінің даму ортасында арнайы білімі болуы керек, сонымен қатар әр платформаның өзінің бағдарламалау тіліне сәйкес келетіндігіне байланысты қосымшаны жазу үшін жоғары шығындар мен үлкен шығындарға ие.

# 1 Графикалық мобильді қосымшаларды әзірлеу алғышарттары

## **1.1 Жоба дегеніміз не ?**

Жоба – бұл күрделі, қайталанбайтын, уақыты, бюджеті, ресурстары шектеулі бір реттік әрекет және нақты нұсқаулары бар , тұтынушының қажет-тіліктеріне бейімделген жүзеге асыру шарасы болып табылады.

Көптеген ұйымдастырушылық іс-шаралар сияқты, жобаның негізгі мақсаты тапсырыс берушінің қажеттіліктерін қанағаттандыру болып табылады. Осы іргелі ұқсастықтан басқа, жобаның сипаттамалары оны басқа ұйымдастырушылық әрекеттерден ажыратуға көмектеседі.

Жобаның негізгі сипаттамалары келесідей:

- Қойылған мақсаты.

- Бастау және аяқталу нүктесі бар анықталған орындалу ұзақтығы.

- Әдетте жобаға бірнеше департаменттер мен түрлі мамандар тартылады.

- Әдетте бұрын-соңды жасалмаған жаңа нәрсемен айналысады.

- Жұмыстың уақытына, құнына және сапасына қойылатын арнайы талаптар болады.

Біріншіден, жобалардың 1 қаңтарға дейін 12 қабатты тұрғын үй кешенін салу немесе теңшелетін бағдарламалық пакеттің 2.0 нұсқасын мүмкіндігінше тезірек шығару сияқты нақты мақсаты бар. Мұндай біртұтас, анықталған мақсат қызметкерлердің күні-түні бірдей тапсырмаларды орындайтын ұйымның күнделікті өмірінде жиі болмайды.

Екіншіден, нақты анықталған мақсат болғандықтан, жобалардың нақты анықталған соңғы нүктесі болады, бұл дәстүрлі жұмыс орындарындағы тұрақты жауапкершіліктерге мүлдем ұқсамайды. Көптеген жағдайларда жұмысшылар бір жұмыста қалмай, бір жобадан екіншісіне ауысады. Парсы шығанағы жағалауында тұщыландыру қондырғыларын салған соң, инженерге Малайзияда мұнай өңдеу зауытын салу міндеті жүктелуі мүмкін.

Үшіншіден, функционалдық мамандану принципі бойынша бөлінетін ұйымдастыру жұмыстарының көптеген түрлерінен айырмашылығы, жобаларға әртүрлі мамандардың күш-жігерін біріктіру және біріктіру тән.

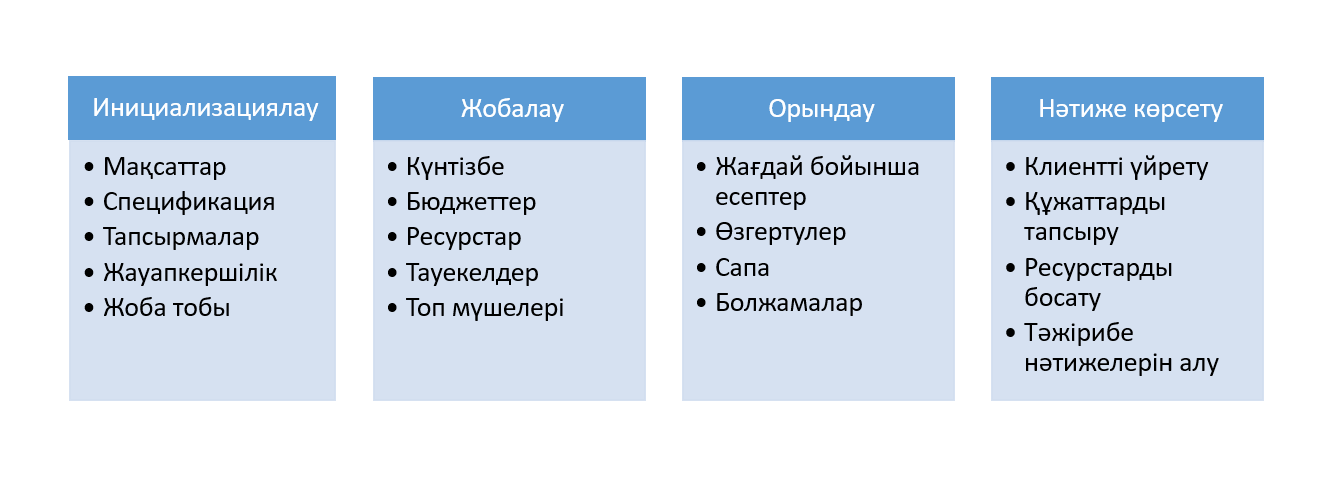
Мысалы, жоба қатысушыларының, инженерлер, қаржы немесе маркетинг мамандары немесе сапаны бақылау мамандары арнайы менеджерлердің жетекшілікпен бөлек кеңселерде жұмыс істеудің орнына, олар жоба менеджерінің басшылығымен жобада жұмыс істейді.

Төртіншіден, жоба ешқашан бірдей болмайды және әрқашан ерекше элементтерді қамтиды. Мұндай ерекше элементтер орын ала ма, жоқ па деген сұрақ емес, бірегейлік дәрежесі қандай болады. Адамды айға ұшыру сияқты, бұрын-соңды болмаған нәрсені жүзеге асыру жаңа мәселелерді шешуді және мүлдем жаңа технологиялардың пайда болуын талап ететіні анық. Екінші жағынан, тіпті белгілі және дамыған әдістер мен оларды қолданатын негізгі құрылыс жобалары да тапсырыс берушіге белгілі бір дәрежеде бейімделуді қажет етеді, бұл оларды ерекше етеді.

Сонымен жобаларды уақыт шектеулерін, шығындар мен жұмысты орындауға қойылатын арнайы талаптар байланыстырады. Жобаларды не орындалғаны, қанша тұратыны және оған қанша уақыт жұмсалғаны бойынша бағаланады.

Жоғарыда аталған үш шектеу көптеген басқа жұмыстарға қарағанда көбірек жауапкершілікті жүктейді. Дәл осы үш фактор жобаны басқарудың негізгі функцияларының біріне баса назар аударады, атап айтқанда: уақытты, шығындарды және жұмысты теңестіру, сонымен бірге тұтынушылардың қанағаттануын барынша арттыру.

**Жобаның өмірлік циклі**

Жобалық жұмыстың бірегей сипатын көрсетудің тағы бір тәсілі - жобаның өмірлік циклін сипаттау. Кейбір жоба менеджерлері жобаның өмірлік циклін жобаны басқару үшін ірге тасы ретінде пайдалануды пайдалы деп санайды. Өмірлік цикл жобалардың шектеулі қызмет ету мерзіміне ие екендігін және күш-жігер деңгейінде болжамды өзгерістер болуын және жобаның өмір сүруіне назар аударуын білдіреді. Жобаны басқару әдебиетінде өмірлік циклдің бірнеше үлгілері бар. Олардың көпшілігі белгілі бір салаға немесе жоба түріне қолданылады. Мысалы, бағдарламалық қамтамасыз етуді әзірлеудің жаңа жобасы 5 кезеңнен тұруы мүмкін: тапсырманы анықтау (жұмыстың мәнін анықтау), жобалау, кодтау (бағдарламалау), интеграциялау/тестілеу және пайдалану. **Сурет 1** Жобаның өмірлік циклы.

Жобаның өмірлік циклі, әдетте, дәйекті түрде төрт кезеңнен өтеді: анықтау, жоспарлау, орындау және нәтижелерді клиентке жеткізу (жеткізу). Барлығы жоба бекітілген сәттен басталады.

Жоба баяу дамиды, бірте-бірте күш-жігер шарықтау шегіне жетеді, содан кейін жоба тапсырыс берушіге тапсырылған кезде жойылады. «Анықтау» кезеңінде жоба нақтыланады (спецификациялар анықталады); жобаның мақсаттары белгіленеді, командалар құрылады; міндеттері айқындалады. «Жоспарлау» кезеңінде күш-жігер деңгейі артады, жоба нені көздейтінін, оны қашан аяқтау керектігін, одан кімге пайда әкелетінін, жұмыс сапасының қандай деңгейін сақтау керектігін және нені сақтау керектігін анықтау үшін жоспарлар әзірленеді және жоба бюджеті болады.

«Орындау» кезеңінде жобалау жұмыстарының негізгі бөлігі - физикалық және психикалық түрде орындалады. Материалдық өнім жасалады (мысалы, көпір, есеп, есептеу бағдарламасы). Бақылау мақсатында жобаның жұмыс кестесіне, сметаға сәйкестігін, бекітілген техникалық шарттарға сәйкестігін тексереді. Осы параметрлердің әрқайсысы үшін қандай болжамдар бар? Қандай өзгерістер енгізу керек? «Жеткізу» кезеңінде әдетте екі нәрсе орындалады: олар дайын өнім жобасын тұтынушыға жеткізеді және жоба ресурстарын қайта бөледі. Жобалық өнімді жеткізу тапсырыс берушіні оқытуды және оған қажетті құжаттарды беруді қамтуы мүмкін. Көшіру әдетте басқа жобаларға жабдықты (және/немесе материалдарды) босатуды және жоба тобы мүшелері үшін жаңа тапсырмаларды жасауды білдіреді.

Тәжірибеде жобаның өмірлік циклін кейбір жоба топтары жалпы жобаны орындау кезінде негізгі тапсырмаларды орындау уақытын анықтау үшін пайдаланады. Мысалы, жобалау тобы негізгі күш-жігерін анықтау кезеңінде жоспарлауы мүмкін, ал сапа тобы өзінің негізгі күш-жігерін жобаның өмірлік циклінің соңғы кезеңдеріне жұмсайды. Көптеген ұйымдарда бір уақытта жүзеге асырылатын жобалар портфолиосы болғандықтан, олардың әрқайсысы өмірлік циклінің әртүрлі кезеңдерінде ұйымда да, жоба деңгейінде де мұқият жоспарлау және басқару маңызды.

**Жоба Менеджері**

Менеджерлер, әдетте, алдын ала белгіленген мақсаттарға жету үшін адам ресурстарын және басқа ресурстарды қалай тиімді пайдалану керектігі туралы шешім қабылдайды, содан кейін бұл шешімдер іс жүзінде жүзеге асырылады. Белгілі бір мағынада жоба менеджерлері басқа менеджерлер сияқты функцияларды орындайды. Яғни, олар жоспарлар мен кестелер жасайды, жұмысшыларды ынталандырады және бақылайды. Әртүрлі типтегі менеджерлер әртүрлі мәселелерді шешетіндіктен ғана бар. Мысалы, маркетинг менеджері өнімдерді немесе қызметтерді таратумен айналысады; өндіріс менеджері ресурстарды дайын өнімге айналдыруға бағытталған; қаржы менеджері ұйымның жұмыс істеуі үшін жеткілікті қаражаттың болуын қамтамасыз етеді. Жоба менеджерінің рөлі ерекше, ол уақытша, қайталанбайтын әрекеттерді басқарады және көбінесе ұйымның ресми құрылымына тәуелсіз жұмыс істейді.

Жоба менеджерлері ресурстарды нақты жобаны уақытында, бюджетте және спецификацияларды бұзбай аяқтайтындай етіп басқара алуы керек. Олар ұйым мен тұтынушы арасындағы байланыстырушы буын болып табылады және тұтынушының күтуін не істеуге болатынын және не істеу керек екенін біріктіре білуі керек (тұтынушының күтуін парасаттылықпен байланыстыра білу).

Олар көбінесе функционалдық бөлімдерімен құрылымдық байланысты уақытша мүшелерден тұратын жоба командасының мүшелерін басқарады, үйлестіреді және біріктіреді. Жоба менеджерлері барлық жұмысқа жауапты, көбінесе өте аз өкілеттіктері бар. Олар уақыт, шығындар және жоба талаптары арасында ақылға қонымды теңгерім жасай алуы керек. Сонымен қатар, функционалдық бөлімдердегі әріптестерінен айырмашылығы, жоба менеджерлері көбінесе олардан шешім қабылдау талап етілетін салаларда үстірт білімге ие. Осыған байланысты олар қажетті міндеттерді шешуге және қажетті шешімдерді қабылдауға қажетті уақытта қажетті адамдарды ынталандыру арқылы жобаны жүзеге асыруға жетекшілік етуге мәжбүр. Әрине, жобалық менеджмент ерекше және қызықты мамандық.

*Жобаны басқару маныздылығы*

Жобаны басқару енді кейбір ерекше жағдайларда ғана қолданылмайды, керісінше, бизнесті жүргізудің стандартты тәсіліне тезірек айналуда. Кәдімгі компаниялардағы жұмыстардың үлес салмағы жобалар ретінде жүзеге асырылады. Болашақта ұйымдарды дамытудың стратегиялық бағдарында жобалардың маңыздылығы мен рөлі артады деп күтілуде.

Жобаны басқару әлдеқашан белгілі бір қажеттілікке бағытталған процесс болудан қалды. Жыл сайын бұл процесс әмбебап болып келеді. Қызметі жобалық менеджментпен тікелей байланысты кез келген ұйым үшін оның қызметтеріне сұраныс тұрақты өсуде. Болашақ жобаларда. Олардың бизнес стратегиясын әзірлеудегі рөлі барған сайын маңызды бола түсуде. Мұның бірнеше себептері бар:

*Өмірлік циклді қысқарту*

Жобалық менеджмент қажеттілігінің жылдан жылға артып келе жатқан негізгі себептерінің бірі заманауи өнімдердің өмірлік циклінің қысқаруы болып табылады. Егер жоғары технологиялық салаға қатысты өнімді алсақ, онда олардың өмір сүру мерзімі бір жылдан үш жылға дейін созылады. Бірнеше жыл бұрын мұндай өнімдердің өмірлік циклі 15 жылға жетті. Жобаны жүзеге асырудың алты айға созылуы тұрақты түрде 33%-ға пайданың жоғалуына әкеледі. Жылдамдық табыстың негізі болды, сондықтан ірі корпорациялар көп салалы жобалық топтарға сенеді. Бұл өнімнің жаңа түрлерін әзірлеу және нарыққа шығару процесін жеделдетуге мүмкіндік береді.

*Техникалық прогресс*

Қазіргі әлем тұрақты технологиялық прогресспен байланысты. Ғылым мен техниканың жетістіктері жобаларда көрініс тауып, оларды күрделірек етеді. Егер осыдан 30 жыл бұрын жол салу өте қарапайым процесс болса, қазір инновациялық технологияларға, құрал-жабдықтарға, құжаттамаға және пайдаланылған стандарттарға сүйене отырып, олай деп айту мүмкін емес. Сондықтан жаңа жобалардың күрделілігіне байланысты оларды басқару оларды тиімді жүзеге асырудың маңызды құрамдас бөлігі болып табылады.

*Қоршаған ортаға көніл бөлу*Ғаламшардағы климаттың өзгеруі, жаһандық жылыну қаупі, бір қызығы, жобаны басқару қызметіне де із қалдырды. Енді бірде-бір компания қоршаған ортаға зиян келтірсе, пайда табу үшін жұмыс істей алмайды. Соңғы өнімді пайдаланудан қоршаған ортаға кері әсерді азайту әдістерін әзірлеу бойынша жұмыс жобаның барлық кезеңдерінде жүргізілуі керек.

*Жоба топтары*

Заманауи компаниялар өз персоналының қысқаруымен сипатталады. Бұл процесс көптеген фирмалардың өмір сүруі үшін өте маңызды. Жоба командалары қазір кейбір қызметкерлерді ауыстырып, олардың міндеттерін мойнына алуда. Қысқарту фирмаларды жобаларға қалай қарайтынын қайта қарауға мәжбүр етеді.

*Клиент қажеттіліктері*

Дизайн қызметіндегі үнемі өсіп келе жатқан бәсекелестік тұтынушылардың қанағаттанушылығына тәуелділікке әкелді. Көптен бері тұтынушыны жаппай тұтынуға арналған өніммен қанағаттандыру мүмкін емес. Тұтынушы өз қалауы мен қажеттіліктеріне сәйкес келетін өнімді қалайды. Мұндай тілектер тапсырыс беруші мен мердігердің жақындасуының катализаторы болып табылады. Өзгерістер өнімдердің жеке тапсырыс бойынша шығарыла бастауына әкелді. Егер бірнеше жыл бұрын гольф клубын сатып алу үшін сізге қолайлы және қаржылық мүмкіндіктеріңізге сәйкес келетін клуб қажет болса, қазір бұл процесс қиын болуы мүмкін. Ойын стиліне байланысты қысқа адамға және ұзын адамға таяқшаны таңдауға болады, тәжірибелі ойыншылар мен жаңадан бастаушыларға және т.б. Бұл жағдайда жобаны басқару өте маңызды. Ол тұтынушының сұранысын қанағаттандыратын және қарым-қатынасты нығайтуға бағытталған өнімдерді шығаруға көмектеседі.

*Жоғары жылдамдық*

Қазіргі әлемді сипаттайтын және кез келген компанияның қызметі тәуелді болатын өзгерістер жылдамдығы бірден бірнеше жобаны бір уақытта басқару қажеттілігін тудырады. Жоба бөлімінің басшысы үшін ең қиын міндет - оған қол жетімді ресурстарды бөлу. Көптеген компаниялар кішігірім жобалар ресурстардың аз ғана бөлігін тұтынады деген пікірге байланысты шағын жобаларды жүзеге асырумен байланысты қаншалықты үлкен проблемалар туындауы мүмкін екенін түсінбейді. Нәтижесінде мұндай жобаларды бақылау жоғалады, бұл жоба командасының айтарлықтай қаржылық шығындары мен уақытын жоғалтуына әкеледі. Бірнеше жобаны бір уақытта жүзеге асырумен айналысатын инжинирингтік фирмада бір сұрақ туындайды – бұл жобаларды параллель басқаруға қолдау көрсете алатын ортаны сауатты құру.

## **1.2 Жобаны басқару кезендері**

Жобаны оңтайландыруға және командаңызға жақсы ұйымдастырылған жоспар бойынша жұмыс істеуге мүмкіндік беретін жобаны басқарудың бес негізгі кезеңі бар. Оларға бастау, жоспарлау, орындау, өнімділікті бағалау және жабу жатады.

**Жобаны бастау**

Жобаны басқару моделінің бастау кезеңінде жоба кең штрихтармен сипатталады. Бұл кезеңде жоба демеушілері мен мүдделі тараптар анықталып, бастапқы зерттеулер басталады. Сонымен қатар, коммуникация жоспары барлық қатысушыларға оңай жеткізілуі үшін жобаны жазбаша түрде құжаттау ұсынылады. Көптеген командалар жобаны бастапқы кездесуден немесе техникалық-экономикалық негіздемеден бастайды. Жобаны қалай бастау керектігі туралы шешім қызметкерлердің қалаған қарым-қатынас стиліне негізделуі керек.

Жобаның бастапқы идеясын сипаттаудан басқа, жобаның күтілетін нәтижелеріне байланысты пайдалар, шығындар және тәуекелдер де сипатталуы керек. Ұйым табысты қалай өлшейтініне байланысты қосымша көрсеткіштерді де қарастыру қажет болуы мүмкін.

Жобаны алдын ала бағалаудан кейін бизнес-кейс немесе кішірек жобалар үшін жоба жарғысы жасалады. Бұл құралдар жобаңызды егжей-тегжейлі сипаттауға және ұсынуға көмектеседі, соның ішінде жоба мақсаттары, бюджет және уақыт кестесі сияқты мәліметтер.

**Жобаны жоспарлау**

Жобаны жоспарлау кезеңінде жобаны басқару шеңберінде жобаның жол картасының көмегімен нақты мақсаттар тұжырымдалады. Мақсаттарды жоспарлаудың көптеген жолдары бар, бірақ SMART мақсаттары, Clear мақсаттар және Мақсаттар мен негізгі нәтижелер (OKR) бастауға көмектесетін жобаны жоспарлаудың үш стратегиясы.

SMART Goals – нақты, өлшенетін, қол жеткізуге болатын, нақты және уақытпен шектелген дегенді білдіретін аббревиатура. Көптеген командалар бұл әдісті қызметкерлер арасындағы байланысты жақсарту, нақты жол картасын жасау және бақыланатын көрсеткіштерді құру үшін пайдаланады.

CLEAR Goals – бұл бірлескен, шектеулі, эмоционалды, бағаланатын және нақтыланатын дегенді білдіретін аббревиатура. Көптеген командалар бұл әдісті таңдайды, себебі ол практикалық іске асыру үшін шынайырақ және ынтымақтастыққа бағытталған.

Мақсаттар мен негізгі нәтижелер (OKR) әдістемесі басқа екі әдістен түбегейлі ерекшеленеді. Олар бүкіл ұйымда ашықтықты құра отырып, компания деңгейінде мақсаттарды ашуға мүмкіндік береді. Жобадан бастап, содан кейін мақсатты анықтаудың орнына, OKR мақсатпен басталады, содан кейін оның айналасында жобалар жасайды.

**Жобаның орындалуы**

Орындау кезеңінде сіздің команда сіздің мақсаттарыңызға жетуге назар аударады. Алғашқы екі кезеңде жиналған ақпарат негізінде жоба ынтымақтастық пен өзара әрекеттесу арқылы уақытында құрылады және іске қосылады.

Орындаушыларға тапсырмалар беруден басталуы керек. Жобаны басқару құралын пайдаланған жағдайда, бұл тапсырмаларды жобаның жұмыс үрдісіне қосуға болады, осылайша топ ресурстарға жылдам қол жеткізіп, бір жерде байланыса алады. Уақыт кестесінің бағдарламалық құралы және басқа жобаны салыстыру құралдары қызметкерлерге жобаның әрбір кезеңін визуализациялауға көмектеседі.

Уақыт шкаласы мен ауқымы күн сайын өзгеретіндіктен, жоба кестесін қажетінше жаңарту қажет және олар аяқталғаннан кейін тәуелділерді жабуды керек. Егер сізде шұғыл мерзімдер болса, команданың жұмысын басымдықтар бойынша жоспарлауды ұмытпаңыз. Жобаның жоспарға сәйкес жүріп жатқанына және тапсырмалардың орындалып жатқанына көз жеткізу үшін орындаушылармен үнемі тексеріп тұру пайдалы болады.

Кестелер өзгеруі мүмкін, бірақ кез келген жағдайда жұмыстың жиналуын болдырмау үшін бастапқы мерзімдерге мүмкіндігінше жақын болу маңызды. Басқаша айтқанда, жобаның бастапқы шеңберінен тым алыс кетпеңіз.

**Жобаның тиімділігін бағалау**

Жоба өнімділігін өлшеу бірнеше себептерге байланысты маңызды. Жобаны жетілдіру мүмкіндігі басты мүмкіндіктердің бірі болып табылады, бірақ бұл маңызды, өйткені қызметкерлер табыстардан да, сәтсіздіктерден де сабақ ала алады. Мақсат қою бағдарламалық құралы арқылы стратегиялық мақсаттарды қою және оларға қол жеткізу үшін пайдалануға болатын көптеген әртүрлі негізгі тиімділік көрсеткіштері (KPI) бар. KPI таңдау сіз жұмыс істеп жатқан жобаның көлемі мен түрлерімен анықталады.

**Жобаның жабылуы**

Бұл кезең әртүрлі компаниялар мен командаларда ең көп өзгереді. Кейбір адамдар қажырлы еңбектің нәтижесін атап өтуді ұнатады, ал басқалары келесі үлкен нәрсеге бірден секіруді ұнатады. Жобаны аяқтаудың дұрыс немесе бұрыс жолы жоқ және сізге және сіздің қызметкерлеріңізге ең жақсысы сізге байланысты.

## **1.3 Жобаның мақсаты мен тапсырмаларын аныктау**

**Жобаны анықтау**

Тапсырыс берушінің және негізгі мүдделі тараптардың қажеттіліктерін қанағаттандырудың ең жақсы тәсілдерінің бірі таңдаулы ақпаратты қажет ететін жобаны жоспарлау мен бақылаудың біріктірілген жүйесін пайдалану болып табылады. Бір шағын жобада жұмыс істейтін жоба менеджерлері ресми жоспарлау жүйесі мен ақпарат болмаған кезде тапсырмаларды жоспарлап, жоспарлай алады. Дегенмен, жоба менеджері бірнеше шағын немесе бір үлкен және күрделі жобаны басқаруы керек жағдайларда, жоба менеджері бұдан былай егжей-тегжейлерді өңдей алмайтын шектеуге тез жетеді.

Ұсынылған әдіс жоба схемасын жасаудың бір түрі болып табылады, сондықтан жұмыс процесін құрылымдау деп аталады. Жоспарды жобалаудағы бастапқы қадамдар барлық тапсырмалардың анықталуын және барлық қатысушының олардан не қажет екенін түсінуін қамтамасыз етеді. Жоспар мен оның егжей-тегжейлері нақтыланғаннан кейін желіні жоспарлау және ресурстарды бөлу үшін біріктірілген ақпараттық жүйені әзірлеуге болады. Сол негізгі ақпарат кейінірек жобаның орындалу барысын бақылау үшін пайдаланылады.

**Техникалық тапсырманы әзірлеу**

Техникалық тапсырманы (ТҚ) әзірлеу жоба жоспарын әзірлеу кезеңін белгілейді Техникалық тапсырма – жобаңыздың түпкі нәтижесінің немесе мақсатының анықтамасы — тұтынушыға арналған өнім немесе қызмет. Мұндағы басты мақсат – соңғы пайдаланушы үшін жұмыстың аралық нәтижелерін барынша нақты анықтау және жоба жоспарларын шоғырландыру (бір бүтінге). Техникалық тапсырманы әзірлеу түбегейлі маңызды болғанымен, жақсы менеджменті бар ірі корпорациялардың жоба менеджерлері бұл кезеңге үстірт қарайды. Зерттеулер көрсеткендей, техникалық тапсырманың нашар әзірленуі жобаның табысты болуына ең көп тараған кедергі болып табылады.

Техникалық тапсырма жоба менеджері мен клиенттің басшылығымен әзірленуі керек. Жоба менеджері тапсырыс берушімен жобаның әрбір кезеңіндегі мақсаттарды, жұмыстың аралық нәтижелерін, техникалық талаптарды және т.б.

Tехникалық тапсырма дұрыстығына көз жеткізу үшін келесі бақылау тізімін пайдалануға болады:

ТК бойынша сұрақтар тізімі;

1. Жобаның мақсаттары.

2. Жұмыстың аралық нәтижелері.

3. Бақылау пункттері.

4. Техникалық талаптар.

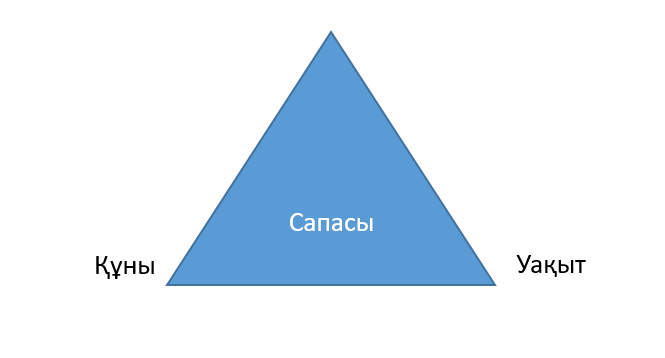
5. Шектеулер мен ерекшеліктер.

6. Тапсырыс берушімен бірлесіп жұмыстың орындалуын тексеру.

1. Жобаның мақсаттары. Техникалық тапсырманы анықтаудағы бірінші қадам клиенттің қажеттіліктерін қанағаттандыру үшін негізгі мақсаттарды анықтау болып табылады. Мысалы, нарықты терең талдау нәтижесінде компьютерлік бағдарламалық қамтамасыз етумен айналысатын компания ағылшын тілінен орыс тіліне автоматты түрде аудара алатын бағдарлама жасауды шешеді. Жоба құны 1,5 миллион доллардан аспайтын үш жыл ішінде аяқталуы керек.
2. Жұмыстың аралық нәтижелері. Келесі қадам жобаның бүкіл өмірлік цикліндегі жұмыстың аралық нәтижелерін анықтау болып табылады. Мәселен, жобаны әзірлеудің ең ерте кезеңіндегі жұмыстың аралық нәтижесі спецификациялар тізімі болуы мүмкін. Келесі қадам үлгілік тестілеу болуы мүмкін. Соңғы қадам соңғы тестілеу және бекітілген бағдарлама болуы мүмкін. Аралық жұмыс қадамдары әдетте уақытты, мөлшерді және/немесе шығындар сметасын қамтиды.
3. Басқару нүктелері. Белгілі бір кезең – жобамен жұмыс істеу процесінде белгілі бір уақытта орын алатын маңызды оқиға. Маңызды кезең диаграммасы жұмыстың негізгі сегменттерін ғана көрсетеді; ол жобаға қажетті уақыттың, құнның және ресурстардың бірінші, шамамен болжамын көрсетеді.
4. Техникалық талаптар. Әдетте, өнім немесе қызмет жақсы жұмыс істеуі үшін техникалық талаптарға сай болуы керек. Мысалы, ДК спецификациясы адаптерсіз 120 В айнымалы токта немесе 240 В тұрақты токта жұмыс істей алады.
5. Шектеулер мен алып тастаулар. Техникалық тапсырманың шекаралары нақты анықталуы керек. Мұны орындамау бос үміттерге және ресурстар мен уақыттың босқа кетуіне әкеледі.
6. Тапсырыс берушімен бірлесіп жұмыстың орындалуын тексеру. Жобаның Техникалық тапсырма сұрақтарының бақылау парағы тапсырыс берушімен жұмысты бірлескен тексерумен аяқталады. Негізгі мәселе – тұтынушының күтілетін нәтижемен түсінуі және келісімі.

**Приоритеттерді орнату**

Дәстүрлі түрде жобаның сапасы мен жалпы табыстылығы тапсырыс берушінің немесе жоғары басшылықтың жобаның құнына (сметасына), уақытына (кесте) және орындалуына (Техникалық тапсырма) қатысты күтулерінің қалай орындалатынына немесе асып кетуіне байланысты деп есептеледі.



**Сурет 2** Жоба басқару арақатынасы .

Бұл критерийлер арасындағы байланыс жағдайға байланысты өзгереді. Мысалы, жобаны тез орындау немесе жоғары шығындарды болдырмау үшін кейде тиімділік (жұмыстың) мен ТҚ (жобаның) арасында ымыраға келу қажет; жиі, жоба неғұрлым ұзағырақ болса, соғұрлым оның құны артады. Дегенмен, бұл әрқашан бола бермейді. Кейде жобаның құнын арзанырақ, тиімділігі төмен жұмыс күшін немесе жобаны ұзартатын жабдықты пайдалану арқылы азайтуға болады.

Жоба менеджерінің негізгі міндеттерінің бірі - уақыт, шығындар және өнімділік арасындағы байланысты басқару. Бұған қол жеткізу үшін жоба менеджерлері жобаның басымдылықтарының сипатын анықтауы және түсінуі керек. Әрбір критерийдің салыстырмалы маңыздылығын анықтау үшін жоба менеджері барлық басымдықтарды жоба демеушісімен және басқарудың жоғарғы деңгейімен бейтарап талқылауы қажет. Бұл үшін талап етілетін әдіс - критерийлердің қайсысын шектеу керек, қайсысын күшейту керек және қабылдануы керек екенін анықтайтын жоба матрицасын құру:

Ұстау. Бастапқы параметр белгіленеді және бекітіледі.Жоба белгіленген мерзімге, бюджетке сәйкес келуі және техникалық сипаттамалар мен масштабқа сәйкес келуі керек.

Жақсарту. Масштабты анықтау кезінде критерийлердің қайсысын оңтайландыру керек? Уақыт пен шығын тұрғысынан оңтайландыру әдетте шығындарды азайту немесе жұмыс уақытын қысқарту мүмкіндіктерін пайдалануды білдіреді. Тиісінше, өнімділік тұрғысынан жақсарту әдетте жобаға құндылық қосуды білдіреді.

Келісу. Критерийлердің қайсысы бастапқы параметрді қанағаттандыра алады? Коэффициенттерді өзгерту қажет болғанда, кестеден ауытқуға, жобаның көлемін немесе орындалуын қысқартуға немесе бюджеттен асып кетуге жол беруге болады ма?

Изображение выглядит как стол

Автоматически созданное описание

**Сурет 3** Приоритеттер матрицасы.

## **1.4 Жоба тобы және қаржылық бөлігі**

Жоба тобы – жоба жұмысын орындауға қатысатын және олардың орындалуына жоба жетекшісінің алдында жауапты жеке мамандарды, топтарды және/немесе ұйымдарды біріктіретін уақытша ұйымдық құрылым.

Жобалық топ жобаны жүзеге асыру мерзіміне мақсатты түрде құрылады. Оның құрамына ішкі және сыртқы орындаушылар мен кеңесшілер кіруі мүмкін.

Жалпы ұйымдастыру функциясында мынадай біршама дербес, бірақ өзара байланысты қызметтерді бөліп көрсетуге болады:

1. Басқару жүйесін ұйымдастыру:
2. Басқару объектісі мен субъектісі арасындағы байланыстар мен қатынастарды ұйымдастыру;
3. Басқару субъектісінің ішіндегі байланыстарды ұйымдастыру.

Басшының ең басты міндеті өзіне бағынышты қызметкерлерінің күш-жігерін алға қойылған міндеттерді шешуге және оларды мейлінше жақсы атқаруға қабілетті ұйым құруға бағыттау болып табылады. Осы тұста төменгі анықтамаларға көңіл бөлуіміз қажет.

*Өкілеттіліктер***-** ұйым ресурстарын пайдалануға және қызметкерлердің күш-жігерін нақтылы бір мәселелерді шешуге бағыттауда берілетін шектеулі мүмкіндік немесе құқық.

*Делегирлеу***-** мақсаттардың орындалуына толық жауап беретін адамға өкілдік пен шешімдерді жүзеге асыруды жүктеу.

*Жауапкершілік-* алда тұрған мақсаттарды орындау жөніндегі міндеттеме және олардың қанағаттанарлық шешіміне жауап беру.

Делегирлеу тек қана өкілеттіліктер қабылданған кезде ғана жүзеге асады. Сондықтан жауапкершіліктің өзі делегирленуі мүмкін емес. өкілеттіліктерді беру процессі екі тұжырымдамаға жүгінеді.классикалық тұжырымдама бойынша, ұйым арасында өкілеттіліктер жоғарыдан төменгі буынға беріледі және бұл басқару деңгейін анықтайды. Қарама-қарсы немесе өкілеттілікті қабылдау тұжырымдамасы кейбір жағдайда орындаушы ұйымдағы өз міндеттерін толық ұсынған кезде басшысының берген өкілеттілігін қабылдамай тастайды деген ұғымға негізделеді. Дегенмен, кез-келген жағдайда өкілеттіліктер шектеулі болып келеді.

Жоба құнын есептеу түрлі тәсілдермен жүргізілуі мүмкін. Мысалы, басқа жобаларға талдаулар жүргізіп, ұқсас есептеулер арқылы. Немесе, жобаның негізгі параметрлерінің бағалары арқылы, мысалы, құрылыс материалдары және жүргізілген жұмыстарға төленетін төлемдер.

Жоба құнын «жоғарыдан-төменге» принципі негізінде, жобаның жалпы құнынан бөлек тапсырмалардың құнын анықтау арқылы, жүргізілуі мүмкін. Немесе «төменнен-жоғарыға» принципімен, жеке тапсырмалар құнын анықтап, олардың қосындысы негізінде жобаның жалпы құнын алу арқылы, жүргізілуі мүмкін.

Еңбек шығындары – бұл адамдар, құрылғылар және құрал-жабдықтар. Еңбек шығындарының ерекшелігі тапсырмалардың аяқталуымен олар тауысылмайды және басқа тапсырмаларды орындауда қолданылуы мүмкін.

Материалдық ресурстар – бұл материалдар мен шикізаттар. Олар тапсырмаларды орындау барысында толық пайдаланылады және басқа тапсырмаларды орындауға қолданыла алмайды, себебі таусылып қалады.

Жеке ресурстар – бұл жеке адамдар немесе құрылғылар

 Топтық (біріктірілген) ресурстар – бұл бірдей немесе өзара алмастырыла алатын ресурстар. Мысалы, редакторлар немесе секретарлар. Біріктірілген ресурстарға кіретін барлық жеке ресурстар бірдей шығынға ие болады және осы топ үшін анықталған бір ресурстар күнтізбесін қолдануы керек. Олар үшін жеке тағайындаулар және жеке демалыс күндерін жасауға болмайды.

Фиксацияланған шығындар – бұл тапсырмалармен тікелей байланысқан шығындар. Мысалы: құрылыс жүргізуге рұқсат алуға кететін шығындар.

 Еңбек ресурстарының шығындары - тапсырмаға тағайындалған әрбір ресурстың шығындары. Олар жұмыс мерзімімен немесе ресурсты пайдалануға кететін уақыт шығынына және тағайындалумен байланысты.

Жалпы шығындар – ресурстар мен фиксацияланатын шығындар қосындысы

Еңбек ресурстарының шығындары төмендегі формула арқылы есептеледі.

Ресурстар шығындары - уақыт ресурстар бірлігінің шығындары \* тапсырмаларды орындауға кететін ресурстардың уақыт мөлшері.

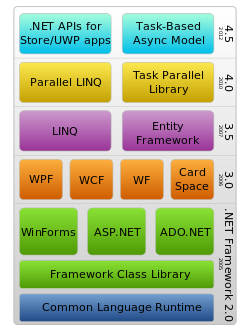
# 2 Жобаларды басқаруға арналған қосымшаны әзірлеуде қолданылған құралдарға шолу

## **2.1 ASP.NET платформасына шолу**

.NET Framework — 2002 жылы Microsoft шығарған бағдарламалық платформа. Платформаның негізі әртүрлі бағдарламалау тілдеріне қолайлы Common Language Runtime (CLR) болып табылады. CLR функционалдығы осы құрылымды пайдаланатын кез келген бағдарламалау тілінде қол жетімді. .NET Framework қазіргі уақытта .NET ретінде әзірленуде.

.NET Framework Microsoft корпорациясының Sun Microsystems (қазір Oracle иелігінде) барған сайын танымал Java платформасына жауабы болып табылады деп есептеледі.

.NET Framework Microsoft корпорациясының меншікті технологиясы болып табылады және ресми түрде Windows отбасының операциялық жүйелерінде жұмыс істеуге арналған, бірақ .NET Framework бағдарламаларын кейбір басқа операциялық жүйелерде іске қосуға мүмкіндік беретін тәуелсіз жобалар (ең алдымен Mono және Portable.NET) бар.



**Сурет 4** .NET технологиялар стекі.

ASP.NET (.NET үшін Active Server Pages) — веб-қызметтерді, бағдарламалық құрал инфрақұрылымын, бағдарламалау үлгісін қамтитын веб-қосымшаларды әзірлеу платформасы. ASP.NET .NET Framework бөлігі болып табылады және ескі Microsoft ASP технологиясының эволюциясы болып табылады.

ASP.NET сыртқы түрі бойынша ескі ASP технологиясына көптеген ұқсастықтарды сақтайды, бұл әзірлеушілерге ASP.NET жүйесіне көшуді салыстырмалы түрде жеңілдетеді. Сонымен қатар, ASP.NET ішкі жағынан ASP-тен өте ерекшеленеді, себебі ол .NET платформасына негізделген және сондықтан осы платформа ұсынған барлық жаңа мүмкіндіктерді пайдаланады.

ASP.NET - бұл HTML, CSS және JavaScript көмегімен тамаша веб-сайттар мен веб-қосымшаларды құруға арналған тегін веб-құрастыру платформасы. Сондай-ақ, Web API интерфейстерін жасауға және Web Sockets сияқты нақты уақыттағы технологияларды пайдалануға болады.

ASP.NET веб-қосымшаларды жасау үшін үш шаблон ұсынады: Web Forms, ASP.NET MVC және ASP.NET веб-беттері. Барлық үш шаблон тұрақты және жетілген және олардың кез келгенімен тамаша веб-қосымшалар жасауға болады. Қандай шаблонды таңдасаңыз да, сіз ASP.NET-тің барлық артықшылықтары мен мүмкіндіктерін барлық жерде аласыз.

Әрбір шаблон әртүрлі бағдарламалау стиліне бағытталған. Сіз таңдаған бағдарламалау активтеріңіздің (білім, дағдылар және әзірлеу тәжірибесі), жасап жатқан қолданба түріне және сізге ыңғайлы әзірлеу тәсіліне байланысты.

**Web Forms**

ASP.NET Web Forms көмегімен сіз таныс апарып тастау, оқиғаға негізделген үлгіні пайдаланып динамикалық веб-сайттарды құра аласыз. Дизайн беті және жүздеген басқару элементтері мен құрамдас бөліктері деректерге қол жеткізу мүмкіндігі бар күрделі, қуатты UI басқаратын сайттарды жылдам құруға мүмкіндік береді.

**MVC**

ASP.NET MVC сізге динамикалық веб-сайттарды құрудың қуатты, үлгілерге негізделген әдісін ұсынады, ол тапсырмаларды таза бөлуге мүмкіндік береді және жағымды, икемді бағдарламалау үшін белгілеуді толық бақылауға мүмкіндік береді. ASP.NET MVC ең соңғы веб-стандартты пайдаланатын күрделі қолданбаларды жасау үшін жылдам, TDD қолайлы әзірлеуге мүмкіндік беретін көптеген мүмкіндіктерді қамтиды.

**ASP.NET веб-беттері**

ASP.NET веб-беттері және Razor синтаксисі динамикалық веб-мазмұнды жасау үшін сервер кодын HTML-мен біріктірудің жылдам, қолжетімді және жеңілдетілген жолын қамтамасыз етеді. Дерекқорларға қосылу, бейнелер қосу, әлеуметтік желі сайттарына сілтеме жасау және соңғы веб-стандартты орындайтын әдемі сайттарды жасауға көмектесетін көптеген кеңейтілген мүмкіндіктерді қоданануға болады.

Барлық үш ASP.NET шаблон .NET Framework негізінде жасалған және .NET және ASP.NET негізгі функцияларын ортақ пайдаланады. Мысалы, барлық үш платформа мүшелікке негізделген кіру қауіпсіздік үлгісін ұсынады және үшеуі де ASP.NET негізгі функционалдығының бөлігі болып табылатын сұрауларды басқару, сеанстарды өңдеу және т.б. үшін бірдей мүмкіндіктерді пайдаланады.

Сондай-ақ, үш платформа толығымен тәуелсіз емес және біреуін таңдау екіншісін пайдалануды болдырмайды. Платформалар бір веб-бағдарламада бірге өмір сүре алатындықтан, әртүрлі платформалармен жазылған қолданбалардың бөлек құрамдастарын көру сирек емес. Мысалы, қолданбаның клиент бөліктерін түзетуді оңтайландыру үшін MVC жүйесінде әзірлеуге болады, ал деректерге қол жеткізу және әкімшілік бөліктер деректерді басқару элементтері мен деректерге оңай қол жеткізу мүмкіндіктерін пайдалану үшін веб-пішіндерінде әзірленеді.

**Web API**

ASP.NET Web API құрылымы браузерлер мен мобильді құрылғыларды қоса алғанда, клиенттердің кең ауқымы үшін HTTP қызметтерін жасауды жеңілдетеді. ASP.NET Web API — .NET Framework негізіндегі REST қолданбаларын құруға арналған тамаша платформа.

## **2.2 ASP.NET Identity**

ASP.NET мүшелік жүйесі ASP.NET 2.0 нұсқасымен 2005 жылы енгізілді, содан бері веб-қосымшалардың аутентификация мен авторизацияны өңдеу тәсілдерінде көптеген өзгерістер болды. ASP.NET Identity - бұл вебке, телефонға немесе планшетке арналған заманауи қолданбаларды құрастырған кезде мүшелік жүйе қандай болуы керек екеніне жаңа көзқарас.

ASP.NET мүшелігі 2005 жылы жиі кездесетін, пішіндердің аутентификациясын және пайдаланушы аттары, құпия сөздер және профиль деректері үшін SQL серверінің дерекқорын қамтитын торап мүшелігі талаптарын шешуге арналған. Бүгінгі күні веб-қосымшалар үшін деректерді сақтау опцияларының анағұрлым кең ауқымы бар және көптеген әзірлеушілер өз сайттарына аутентификация және авторизация функционалдығы үшін әлеуметтік сәйкестендіру провайдерлерін пайдалануға мүмкіндік бергісі келеді. ASP.NET Мүшелік дизайнының шектеулері бұл өтуді қиындатады:

- Дерекқор схемасы SQL Server үшін жасалған және оны өзгерту мүмкін емес. Профиль ақпаратын қосуға болады, бірақ қосымша деректер басқа кестеге жинақталған, бұл Profile Provider API арқылы басқа кез келген әдіспен қол жеткізуді қиындатады.

- Провайдер жүйесі қосалқы деректер қоймасын өзгертуге мүмкіндік береді, бірақ жүйе реляциялық дерекқорға сәйкес келетін болжамдардың айналасында жасалған. Сіз Azure сақтау кестелері сияқты реляциялық емес сақтау механизмінде мүшелік ақпаратты сақтау үшін провайдерді жаза аласыз, бірақ содан кейін көп кодты және NoSQL дерекқорларына қолданылмайтын әдістер үшін көптеген ерекше жағдайларды жазу арқылы реляциялық дизайнды айналып өтуіңіз керек.

- Жүйеге кіру/шығу функциясы пішіндердің аутентификациясына негізделгендіктен, мүшелік жүйесі OWIN пайдалана алмайды.

**ASP.NET қарапайым мүшелігі**

ASP.NET қарапайым мүшелігі ASP.NET веб-беттеріне арналған мүшелік жүйе ретінде әзірленді. Ол WebMatrix және Visual Studio 2010 SP1 бірге шығарылды. Қарапайым мүшеліктің мақсаты Web Pages қолданбасына мүшелік функционалдылықты қосуды жеңілдету болды.

Қарапайым мүшелік пайдаланушы профилі ақпаратын теңшеуді жеңілдетеді, бірақ ол әлі де ASP.NET мүшелігімен басқа мәселелерді бөліседі және оның кейбір шектеулері бар:

- Реляциялық емес ортада мүшелік жүйе деректерін сақтау қиын болды.

- Сіз оны OWIN арқылы пайдалана алмайсыз.

- Ол бар ASP.NET мүшелік провайдерлерімен жақсы жұмыс істемейді және кеңейтілмейді.

**ASP.NET Identity**

ASP.NET жүйесіндегі мүшелік тарихы жылдар бойы дамып келе жатқандықтан, ASP.NET командасы тұтынушылардың пікірлерінен көп нәрсені үйренді.

Пайдаланушылар өз қолданбаңызда тіркелген пайдаланушы аты мен құпия сөзді енгізу арқылы жүйеге кіреді деген болжам енді жарамсыз. Интернет әлеуметтік сипатқа ие болды. Пайдаланушылар Facebook, Twitter және басқа әлеуметтік веб-сайттар сияқты әлеуметтік арналар арқылы нақты уақытта бір-бірімен өзара әрекеттеседі. Әзірлеушілер пайдаланушылардың өздерінің веб-сайттарында бай тәжірибесі болуы үшін өздерінің әлеуметтік идентификацияларымен жүйеге кіру мүмкіндігін қалайды. Заманауи мүшелік жүйесі Facebook, Twitter және басқалар сияқты аутентификация провайдерлеріне қайта бағыттауға негізделген жүйеге кіру мүмкіндігін қосуы керек.

Веб-әзірлеу дамыған сайын веб-әзірлеу үлгілері де өзгерді. Қолданбалық кодты бірлік тестілеу қолданбаларды әзірлеушілер үшін басты мәселе болды. 2008 жылы ASP.NET модель-көрініс-контроллер (MVC) үлгісіне негізделген жаңа құрылымды қосты, ішінара әзірлеушілерге бірлік тексерілетін ASP.NET қолданбаларын құруға көмектесу үшін. Қолданбалардың логикасын бірлікте сынағысы келетін әзірлеушілер де мүшелік жүйемен мұны істеу мүмкіндігін алғысы келді.

Веб-қосымшаларды әзірлеудегі осы өзгерістерді ескере отырып, ASP.NET Identity келесі мақсаттармен әзірленді:

- Бір ASP.NET Identity жүйесі

- ASP.NET идентификациясын ASP.NET MVC, веб пішіндері, веб-беттері, Web API және SignalR сияқты барлық ASP.NET шеңберлерімен пайдалануға болады.

- ASP.NET идентификациясын веб, телефон, дүкен немесе гибридті қолданбаларды құрастырған кезде пайдалануға болады.

- Пайдаланушы туралы профиль деректерін қосудың қарапайымдылығы

- Сіз пайдаланушы және профиль ақпаратының схемасын басқара аласыз. Мысалы, пайдаланушылар қолданбаңызда тіркелгіні тіркеген кезде енгізген туған күндерін сақтау үшін жүйені оңай қосуға болады.

- Тұрақтылықты бақылау

- Әдепкі бойынша, ASP.NET сәйкестендіру жүйесі дерекқордағы барлық пайдаланушы ақпаратын сақтайды. ASP.NET Identity бағдарламасы барлық тұрақтылық механизмін жүзеге асыру үшін Entity Framework Code First пайдаланады.

- Сіз дерекқор схемасын басқаратындықтан, кесте атауларын өзгерту немесе бастапқы кілттердің деректер түрін өзгерту сияқты жалпы тапсырмаларды орындау оңай.

- SharePoint, Azure Storage Table Service, NoSQL дерекқорлары сияқты әртүрлі сақтау жүйелерін оңай қосуға болады.

- Рөлдер басқарушысы

- Рөлдер бойынша қолданбаның бөліктеріне кіруді шектеуге мүмкіндік беретін рөлдер провайдері бар. Сіз «Әкімші» сияқты рөлдерді оңай жасай аласыз және пайдаланушыларды рөлдерге қоса аласыз.

- Мәлімдемелерге негізделген

- ASP.NET идентификациясы шағымдарға негізделген аутентификацияны қолдайды, мұнда пайдаланушының жеке куәлігі шағымдар жинағы ретінде ұсынылған. Шағымдар әзірлеушілерге рұқсат етілген рөлдерге қарағанда пайдаланушының жеке басын сипаттауда әлдеқайда мәнерлі болуға мүмкіндік береді. Рөл мүшелігі тек логикалық мән (мүше немесе мүше емес) болса да, шағым пайдаланушының жеке басы мен мүшелігі туралы мәліметтерді қамтуы мүмкін.

- Әлеуметтік медиа тіркелгі провайдерлері

- Қолданбаға Microsoft тіркелгісі, Facebook, Twitter, Google және басқа пайдаланушылар сияқты әлеуметтік логиндерді оңай қосуға және пайдаланушыға қатысты деректерді сақтауға болады.

- OWIN интеграциясы

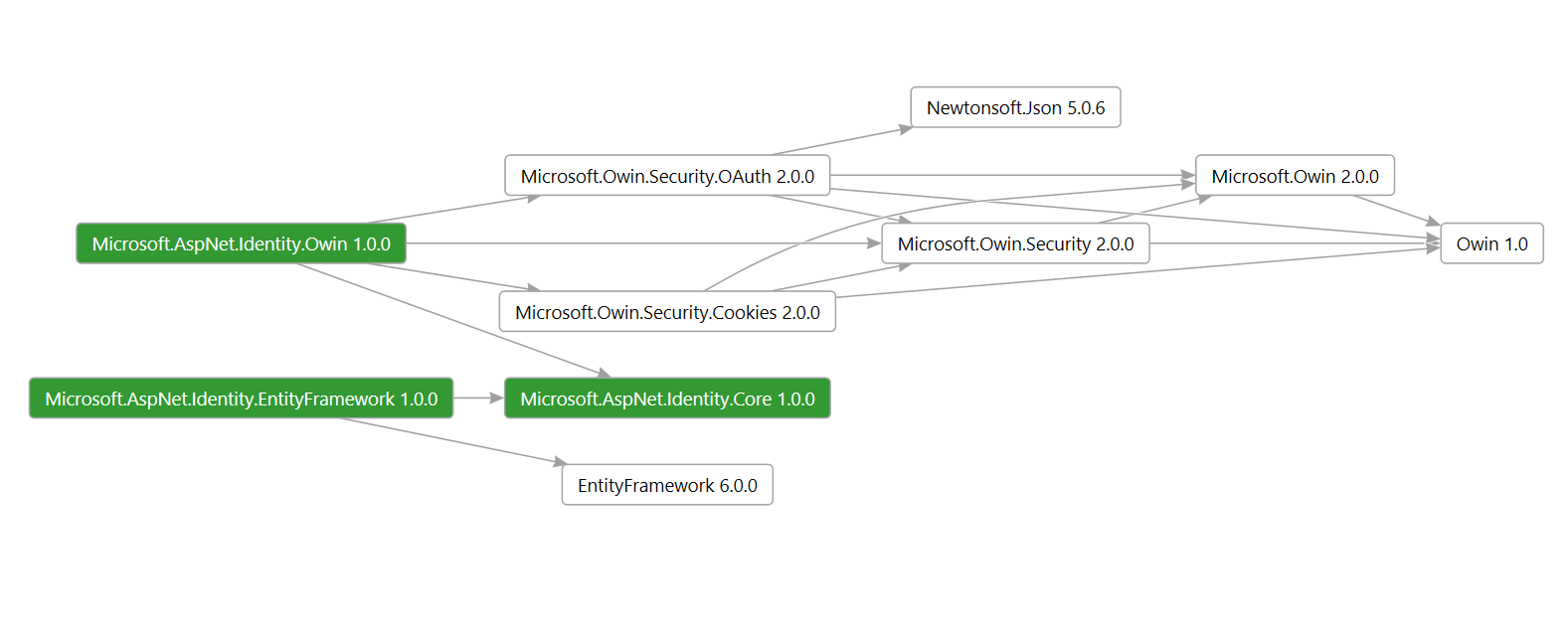
- ASP.NET аутентификациясы енді кез келген OWIN негізіндегі хостта пайдаланылуы мүмкін OWIN аралық бағдарламалық құралына негізделген. ASP.NET Identity жүйесі System.Web-ге тәуелді емес. Бұл кез келген орналастырылған OWIN қолданбасында пайдалануға болатын толық үйлесімді OWIN платформасы.

- ASP.NET Identity пайдаланушыларды веб-сайтқа кіру немесе одан шығу үшін OWIN аутентификациясын пайдаланады. Бұл cookie файлын жасау үшін FormsAuthentication пайдаланудың орнына қолданба мұны істеу үшін OWIN CookieAuthentication пайдаланады дегенді білдіреді.

- NuGet бумасы

- ASP.NET идентификациясы Visual Studio 2017 бағдарламасымен жеткізілетін ASP.NET MVC, веб пішіндері және Web API үлгілерінде орнатылған NuGet бумасы ретінде таратылады. Бұл NuGet бумасын NuGet галереясынан жүктеп алуға болады.

- ASP.NET идентификациясын NuGet бумасы ретінде шығару ASP.NET тобына жаңа мүмкіндіктер мен қателерді түзетулерді қайталауға және оларды әзірлеушілерге икемді түрде қамтамасыз етуге мүмкіндік береді.



**Сурет 5** ASP.NET Identity 2.1 құлымдық моделі.

## **2.3 ASP.NET Web – API 2 Individual Accounts шаблоны**

Visual Studio 2013 бағдарламасында Web API жобасының үлгісі аутентификация үшін үш опцияны береді:

* Individual Accounts. Қолданба мүшелік дерекқорды пайдаланады.
* **Organizational accounts**. Пайдаланушылар Azure Active Directory, Office 365 немесе жергілікті Active Directory тіркелгі деректерімен жүйеге кіреді.
* Windows authentication. Бұл опция интранет қолданбаларына арналған және Windows аутентификация IIS модулін пайдаланады.

Жеке тіркелгілер пайдаланушыға жүйеге кірудің екі жолын ұсынады:

* Жергілікті логин. Пайдаланушы логин мен парольді енгізе отырып, сайтта тіркеледі. Қолданба құпия сөз хэшін мүшелік дерекқорында сақтайды. Пайдаланушы жүйеге кіргенде, ASP.NET Identity жүйесі құпия сөзді тексереді.
* Әлеуметтік логин. Пайдаланушы Facebook, Microsoft немесе Google сияқты сыртқы қызмет арқылы жүйеге кіреді. Қолданба әлі де мүшелік дерекқорда пайдаланушы үшін жазба жасайды, бірақ ешқандай тіркелгі деректерін сақтамайды. Пайдаланушы сыртқы қызметке кіру арқылы аутентификацияланады.

**Жергілікті кіру тіркелгі деректерінің ағыны**

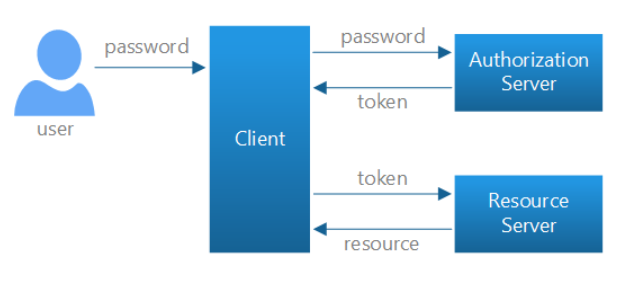
Жергілікті кіру үшін Web API OAuth2 ішінде анықталған ресурс иесінің құпия сөз ағынын пайдаланады.

1) Пайдаланушы клиентке атау мен құпия сөзді енгізеді.

2) Клиент бұл тіркелгі деректерін авторизация серверіне жібереді.

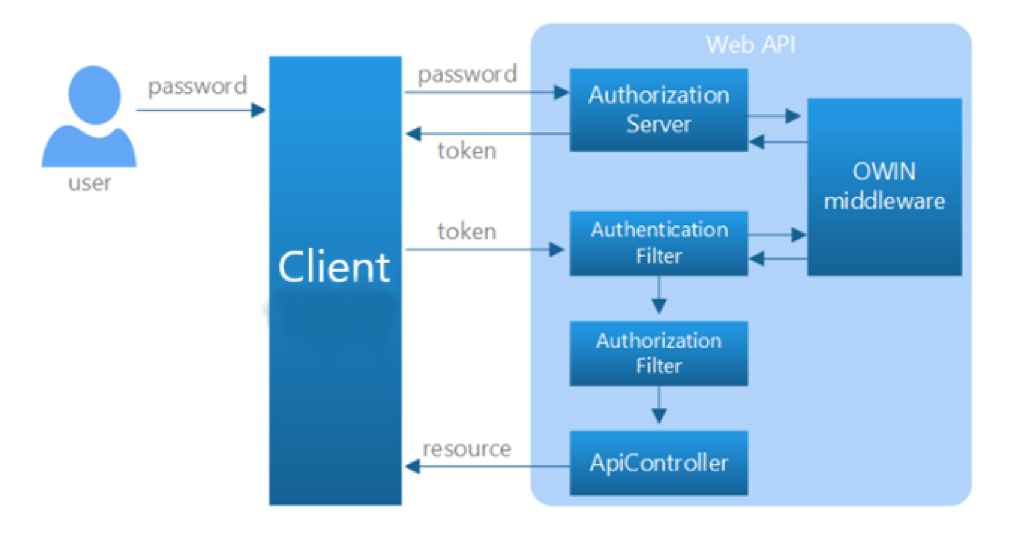
3) Авторизация сервері тіркелгі деректерін аутентификациялайды және кіру таңбалауышын қайтарады.

4) Қорғалған ресурсқа қол жеткізу үшін клиент HTTP сұрауының Авторизация тақырыбына кіру таңбалауышын қосады.



**Сурет 6** OAuth2 авторизация схемасы.

Web API жобасы үлгісінде Жеке тіркелгілерді таңдаған кезде, жоба пайдаланушы тіркелгі деректерін тексеретін және таңбалауыштарды шығаратын авторизация серверін қамтиды. Келесі диаграмма Web API құрамдастары тұрғысынан бірдей тіркелгі деректерінің ағынын көрсетеді.



**Сурет 7** WebApi тіркелгі ағыны.

Бұл сценарийде Web API контроллері ресурс серверлері ретінде әрекет етеді. Authentication filter access token – ді тексереді және [Authorize] атрибуты ресурсты қорғау үшін пайдаланылады. Контроллерде немесе әрекетте [Authorize] атрибуты болса, сол контроллерге немесе әрекетке барлық сұраулар аутентификациялануы керек. Әйтпесе, авторизациядан бас тартылады және Web API 401 (Рұқсат етілмеген) қатесін қайтарады.

Авторизация сервері және аутентификация сүзгісі екеуі де OAuth2 мәліметтерін өңдейтін OWIN аралық бағдарламалық құрал құрамдасына шақырады.

**Individual Accounts жобасының үлгісін түсіну**

Cұраныстар AccountController классымен өңделеді. Ішкі түрде AccountController мүшелік дерекқорды басқару үшін ASP.NET Identity пайдаланады.

Бұл үлгі қолданады:

* OAuth2 авторизация сервері.
* Пайдаланушы тіркелгілерін басқаруға арналған Web API соңғы нүктесі
* Пайдаланушы тіркелгілерін сақтауға арналған EF үлгісі.

Міне, осы мүмкіндіктерді жүзеге асыратын негізгі қолданбалы класстар:

* AccountController. Пайдаланушы тіркелгілерін басқаруға арналған Web API соңғы нүктесін қамтамасыз етеді. Класстағы басқа әдістер құпия сөзді қалпына келтіруді, әлеуметтік жүйеге кіруді және басқа функцияларды қолдайды.
* ApplicationUser, /Models/IdentityModels.cs ішінде анықталған. Бұл сынып мүшелік дерекқорындағы пайдаланушы тіркелгілеріне арналған EF үлгісі болып табылады.
* ApplicationUserManager, /App\_Start/IdentityConfig.cs ішінде анықталған Бұл класс UserManager ішінен алынады және жаңа пайдаланушы жасау, құпия сөздерді тексеру және т.б. сияқты пайдаланушы тіркелгілеріндегі әрекеттерді орындайды және дерекқордағы өзгерістерді автоматты түрде сақтайды.
* ApplicationOAuthProvider. Бұл нысан OWIN аралық бағдарламалық құралына қосылады және аралық бағдарламалық құрал көтерген оқиғаларды өңдейді.

## **2.4 Xamarin.Forms**

Xamarin.Forms ашық бастапқы UI құрылымы болып табылады. Xamarin.Forms әзірлеушілерге Xamarin.Android, Xamarin.iOS және Windows қолданбаларын бір ортақ код базасынан құруға мүмкіндік береді.

Xamarin.Forms C# тіліндегі кодпен XAML арқылы пайдаланушы интерфейстерін жасауға мүмкіндік береді. Бұл интерфейстер әр платформада өнімді жергілікті басқару элементтері ретінде көрсетіледі.

Xamarin.Forms келесі мақсаттары бар әзірлеушілерге арналған:

* UI макетін және дизайнын платформалар арқылы бөлісу.
* Платформаларда кодты, сынақты және бизнес логикасын бөлісу.
* Visual Studio көмегімен C# тілінде кросс-платформалық қолданбаларды жазу.

Xamarin.Forms – Android, iOS және Windows 10 үшін кросс-платформалық қолданбаларды жасауға бағытталған платформа. Неліктен бұл платформаны пайдалану керек, ол қандай артықшылықтар әкеледі? Мобильді қосымшалардың айтарлықтай бөлігі бірнеше платформалар үшін, мысалы, Android және iOS үшін жасалғаны туралы белгілі бір статистика бар. Дегенмен, әзірлеушілер сөзсіз келесі қиындықтарға тап болады:

* графикалық интерфейсті құру тәсілдерінің айырмашылығы тура немесе басқаша бағдарламалауға әсер етеді. Әзірлеушілер қолданбаны белгілі бір платформадағы интерфейске қойылатын талаптарға бейімдеуге мәжбүр
* әртүрлі API интерфейстері - бағдарламалау интерфейстеріндегі және белгілі бір функцияларды іске асырудағы айырмашылық бағдарламашыдан осы ерекше мүмкіндіктерді ескеруді талап етеді.
* дамыту үшін әртүрлі платформалар. Мысалы, iOS жүйесіне арналған қосымшаларды жасау үшін бізге сәйкес орта - Mac OS X және XCode сияқты бірқатар арнайы құралдар қажет. Ал бағдарламалау тілі ретінде Objective-C немесе Swift таңдалады. Android үшін біз әртүрлі орталарды пайдалана аламыз - Android Studio, Eclipse және т.б. Бірақ мұнда Java немесе Kotlin қосымшалардың басым көпшілігі үшін пайдаланылады.

Ал Windows жүйесіне арналған қолданбаларды жасау үшін Visual Studio қолданылады, ал тілдер ретінде C #, F #, VB.NET, C ++ қолданылады.

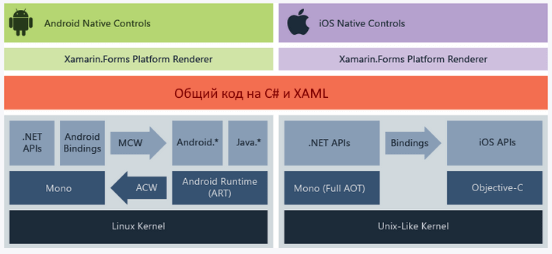
Платформалардың, әзірлеу құралдарының және бағдарламалау тілдерінің мұндай ауқымы қосымшаларды әзірлеу уақытына, сайып келгенде, әзірлеуге бөлінген қаражатқа оң әсер ете алмайды. Барлық платформаларға бірден қолданбаларды жасауды жеңіл әрі қарапайым ететін бір құралдың болуы өте тиімді болар еді. Xamarin платформасы дәл солай.

Xamarin бір мезгілде барлық үш платформа үшін - Android, iOS, UWP үшін C # және .NET көмегімен бір қолданба логикасын жасауға мүмкіндік береді.

Xamarin.Forms қолданудың артықшылықтары:

* Әзірлеу процесі барлық платформалар үшін бір кодты жасайды
* Xamarin әрбір платформаның жергілікті API интерфейстеріне тікелей қатынасты қамтамасыз етеді
* Қолданбаларды құру кезінде біз .NET платформасын және C# бағдарламалау тілін (сонымен қатар F#) пайдалана аламыз, ол айтарлықтай өнімді және сонымен бірге түсінікті және үйренуге және пайдалануға оңай.
* Xamarin Forms бірнеше платформаларды қолдайды. Негізгі платформалар: Android, iOS, UWP, Tizen. Қауымдастық әзірлеген қосымша платформалар: MacOS, WPF, GTK#.

Xamarin жұмысын осылай көрсетуге болады:



**Сурет 8** Xamarin құрылымдық схемасы.

Xamarin .NET Framework бағдарламасының ашық бастапқы іске асырылуын қамтамасыз ететін моно жақтаудың үстінде жұмыс істейді. Моно әртүрлі платформаларда жұмыс істей алады - Linix, MacOS және т.б.

Әрбір жеке платформа деңгейінде Xamarin бірқатар ішкі платформаларға сүйенеді. Соның ішінде:

* Xamarin.Android - Android ОЖ-де қолданбаларды жасауға арналған кітапханалар
* Xamarin.iOS - iOS қолданбаларын құруға арналған кітапханалар

Бұл ішкі платформалар үлкен рөл атқарады - олар арқылы қолданбалар Android немесе iOS жүйесімен жұмыс істейтін құрылғылардағы қолданба интерфейстеріне сұраулар жібере алады. Қысқаша айтқанда, бұл осылай көрінеді.

Xamarin.Android көмегімен Xamarin пайдаланатын C# коды аралық тілге (IL) құрастырылады, ол содан кейін қолданба іске қосылған кезде жергілікті жинаққа құрастырылады. Xamarin қолданбалары моно орындалу уақытында іске қосылады. Код Android API интерфейсіне тікелей қол жеткізе алмайды. Ол үшін Android Runtime Virtual Machine (ART) қамтамасыз ететін Android.\* және Java.\* аттар кеңістігінің функционалдық мүмкіндіктеріне жүгіну керек. Басқарылатын шақырылатын орауыштардың (MCW) арнайы қабаты басқарылатын кодтық қоңырауларды жергілікті қоңырауларға аударуға және Android.\* және Java.\* аттар кеңістігінің функционалдық мүмкіндіктеріне қол жеткізуге мүмкіндік береді.

Керісінше, Android Runtime (ART) Xamarin коды бар қолданбаны шақырғанда, барлық қоңыраулар Android Callable Wrappers (ACW) арқылы өтеді.

Xamarin.iOS қолданбалары, JIT компиляциясын пайдаланатын Xamarin.Android қолданбасынан айырмашылығы, жергілікті ARM кодына C# кодының AOT (уақытынан бұрын) компиляциясын пайдаланады. Xamarin Objective-C код шақыруларын C# кодына аудару үшін аралық селекторлар деңгейін және C# кодын Objective-C тіліне аудару үшін Тіркеушілер деңгейін пайдаланады. Нәтижесінде, селекторлар мен тіркеушілер қабаттары жалпы жоғарыдағы суретте «байланыстырулар» деп белгіленген және шын мәнінде Objective-C кодын C# кодымен әрекеттесуге мүмкіндік беретін аралық қабатты білдіреді.

Нәтижесінде, осы платформалардың арқасында біз Android үшін бөлек, iOS үшін бөлек қолданбаларды жасай аламыз, бірақ Xamarin-тің ең маңызды ерекшелігі кросс-платформалық қосымшаларды жасау мүмкіндігі болып табылады, яғни барлық платформалар үшін бір логика. Бұл мүмкіндік Xamarin.Forms технологиясы арқылы ұсынылған және ол Xamarin.Android және Xamarin.iOS деңгейінен жоғары деңгейде жұмыс істейді. Яғни, Xamarin.Forms көмегімен біз визуалды интерфейсті бір рет анықтай аламыз, оған C # тілінде бір рет логиканы тіркей аламыз және мұның бәрі Android, iOS және Windows жүйелерінде жұмыс істейді. Xamarin.Forms содан кейін Xamarin.Forms көрнекі құрамдастарын платформаға тән GUI-ге аудару үшін XAML/C# басқару элементтерін жергілікті басқару элементтеріне байланыстыру үшін рендерлерді, арнайы нысандарды пайдаланады.

Xamarin.Forms платформалар арқылы UI элементтерін жасау үшін дәйекті API ұсынады. Бұл API XAML немесе C# ішінде іске асырылуы мүмкін және Model-View-ViewModel (MVVM) сияқты үлгілер үшін деректерді байланыстыруды қолдайды.

Орындалу уақытында Xamarin.Forms платформалық рендерерлерді Xamarin.Android, Xamarin.iOS және UWP жүйелеріндегі кросс-платформа UI элементтерін жергілікті басқару элементтеріне түрлендіру үшін пайдаланады. Бұл әзірлеушілерге платформалар арқылы кодты ортақ пайдаланудың артықшылықтарын түсіне отырып, жергілікті көріністі, сезімді және өнімділікті алуға мүмкіндік береді.

Xamarin.Forms қолданбалары әдетте ортақ .NET Standard кітапханасынан және жеке платформа жобаларынан тұрады. Ортақ кітапхана XAML немесе C# көріністерін және қызметтер, үлгілер немесе басқа код сияқты кез келген бизнес логикасын қамтиды. Платформа жобалары кез келген платформаға тән логиканы немесе қолданба талап ететін бумаларды қамтиды.

**Қосымша функционалдылық**

Xamarin.Forms қолданбаларында әртүрлі функционалдылықты қосатын кітапханалардың үлкен экожүйесі бар. Бұл бөлім осы қосымша функциялардың кейбірін сипаттайды.

**Xamarin.Essentials**

Xamarin.Essentials - жергілікті құрылғы мүмкіндіктері үшін платформааралық API интерфейстерін қамтамасыз ететін кітапхана. Xamarin сияқты, Xamarin.Essentials жергілікті утилиталарға қол жеткізу процесін жеңілдететін абстракция болып табылады. Xamarin.Essentials ұсынған утилиталардың кейбір мысалдары мыналарды қамтиды:

* Құрылғы ақпараты
* Файлдық жүйе
* Акселерометр
* Телефон тергіш
* Сөйлеген сөйлемдерді текстке аударуға
* Экранды блоктау

**Shell**

Xamarin.Forms Shell қолданбалардың көпшілігі талап ететін негізгі мүмкіндіктерді қамтамасыз ету арқылы мобильді қолданбаларды әзірлеудің күрделілігін азайтады. Shell ұсынған мүмкіндіктердің кейбір мысалдары мыналарды қамтиды:

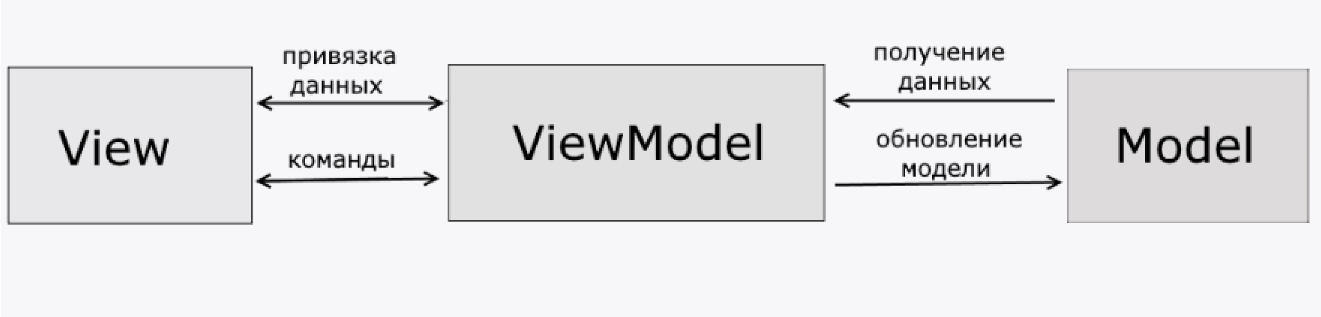
* Жалпы навигация тәжірибесі
* URI негізіндегі навигация схемасы
* Біріктірілген іздеу өңдеушісі

## **2.5 MVVM паттрені**

MVVM (Model-View-ViewModel) үлгісі қолданбалы логиканы көрнекі бөліктен (көрініс) бөлуге мүмкіндік береді. Бұл үлгі архитектуралық болып табылады, яғни қолданбаның жалпы архитектурасын көрсетеді.

Бұл үлгіні Джон Госсман 2005 жылы Presentational Model үлгісінің модификациясы ретінде енгізді және бастапқыда WPF жүйесінде қолданбаларды әзірлеуге бағытталған. Қазір бұл үлгі WPF шегінен шығып, әртүрлі технологияларда, соның ішінде Android, iOS үшін әзірлеуде қолданылғанылады.

MVVM үш құрамдас бөліктен тұрады: Үлгі (Model), Көрініс үлгісі (ViewModel) және Көрініс (View).



**Сурет 9 MVVM** паттернің схемасы.

**Model**

Модель қолданбада пайдаланылатын деректерді сипаттайды. Модельдерде үлгі сипаттарын тексеруге арналған логика сияқты осы деректерге тікелей қатысты логика болуы мүмкін. Сонымен қатар, модельде деректерді көрсетуге және визуалды басқару элементтерімен әрекеттесуге қатысты ешқандай логика болмауы керек.

Модельдің INotifyPropertyChanged немесе INotifyCollectionChanged интерфейстерін іске асыруы сирек емес, бұл жүйеге үлгі сипатының өзгерістері туралы хабарлауға мүмкіндік береді. Бұл көрініспен байланыстыруды жеңілдетеді, дегенмен модель мен көрініс арасында тікелей өзара әрекеттесу қайтадан болмайды.

**View**

Көрініс немесе көрсетілім пайдаланушы қолданбамен әрекеттесетін көрнекі интерфейсті анықтайды. Xamarin.Forms ішінде көрініс түймелер, мәтін өрістері және басқа көрнекі элементтер түріндегі интерфейсті анықтайтын xaml коды болып табылады.

Xamarin.Forms ішіндегі терезе (Контент парағы) xaml тіліндегі интерфейсті де, оған байланысты C# кодын да қамтуы мүмкін болса да, ең дұрысы C# кодында InitializeComponent әдісін шақыратын және терезенің инициализациясын орындайтын конструктордан басқа логика болмауы керек. Қолданбаның барлық негізгі логикасы ViewModel компонентінде шығарылады.

Дегенмен, кейде байланысты код файлында ViewModel ішіндегі MVVM үлгісінде іске асыру қиын кейбір логика болуы мүмкін.

Көрініс оқиғаларды сирек ерекшеліктермен өңдемейді, бірақ әрекеттерді негізінен пәрмендер арқылы орындайды.

**ViewModel**

ViewModel немесе ViewModel үлгіні және көріністі деректерді байланыстыру механизмі арқылы байланыстырады. Үлгідегі сипат мәндері өзгерсе, модель INotifyPropertyChanged интерфейсін жүзеге асырған кезде, үлгі мен көрініс тікелей қатысты болмаса да, көріністе көрсетілетін деректер автоматты түрде өзгереді.

ViewModel де үлгіден деректерді алу логикасын қамтиды, содан кейін ол көрініске жіберіледі. Сондай-ақ VewModel үлгідегі деректерді жаңарту логикасын анықтайды.

Көрініс элементтері, яғни түймелер сияқты көрнекі құрамдас бөліктер оқиғаларды пайдаланбағандықтан, көрініс ViewModel бағдарламасымен пәрмендер арқылы әрекеттеседі.

Мысалы, пайдаланушы мәтіндік өріске енгізілген деректерді сақтағысы келеді. Ол түймені басады және сол арқылы ViewModel-ге команда жібереді. ViewModel қазірдің өзінде тасымалданған деректерді алады және сәйкесінше үлгіні жаңартады.

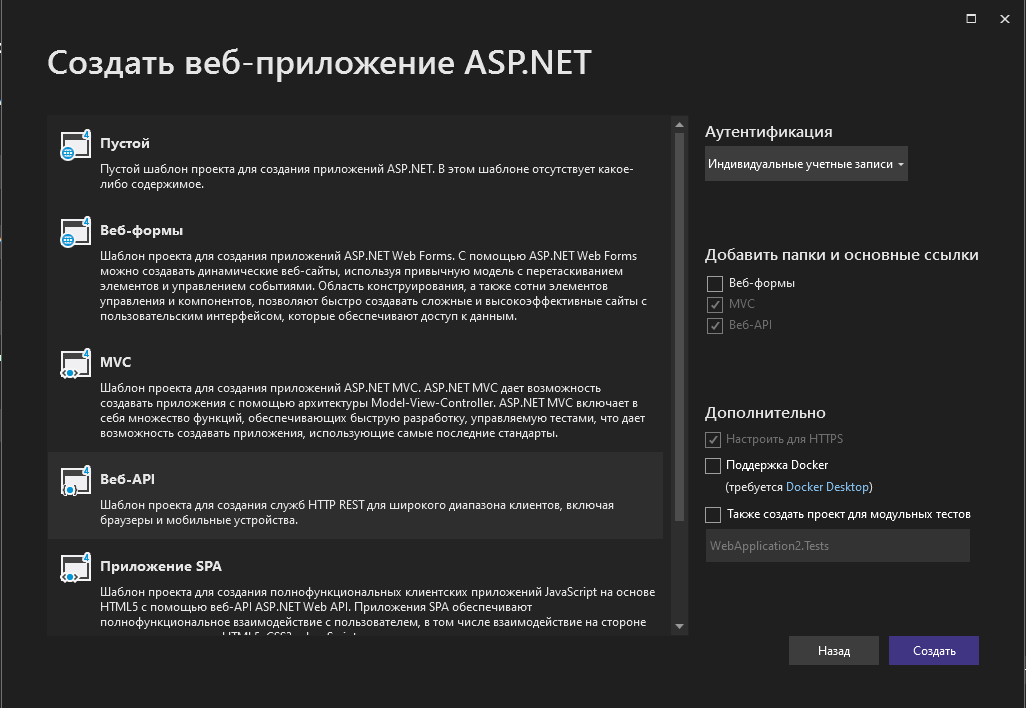
MVVM үлгісін қолдану нәтижесі қолданбаны әзірлеу және сынау, сондай-ақ әрі қарай өзгерту және қолдау оңайырақ үш құрамдас бөлікке функционалдық бөлу болып табылады.

# 3 Жобаны бағдарламалау

## **3.1 Жобаларды басқаруға арналған көпқолданушылы қосымша үшін сервер құру**

**Сервер жобасын құру**

Ен алдымен біз Visual Studio бағдарламасында ASP.NET Web Application (.NET Framework) проектін ашып, Individual Accounts шаблонын қолдану үшін аутентификацияны тандау элементінде “Индивидуальные учетные записи” тандаймыз:



**Сурет 10** Сервер жобасын құру терезесі.

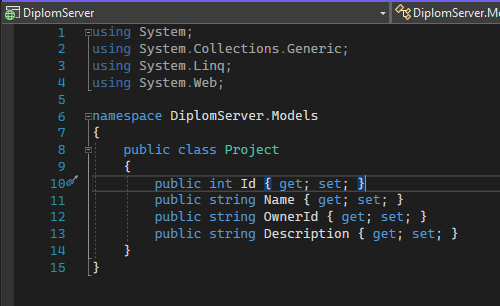
Жоба құрылғанынан кейін біздің жобада ASP.NET Identity- мен жұмыс жасайтын арнайы Account Controller контроллер класы құрылады. Бұл контроллер-классының функцияларын кейін біз Пайдаланушыны тіркеу, Тіркелген пайдалнушыларды алу, Пайдаланушы құпиясөзін өзгерту және т.б. істер үшін қолданамыз.

Ақпаратты әрбір пайдаланушы ала алмау үшін және жеке пайдаланушымен жұмыс жасау үшін біз Authorize атрибутын қолданамыз. Сол арқылы біз қай ақпараттың қай пайдаланушыға жататының білеміз. Контроллерге Authorize атрибутын бергенде оған тек access token – і бар пайдаланушы ғана қолдана алады. Access token алу үшін пайдаланушы тіркелген кездегі өзінің электронды поштасын және құпиясөзін жібереді, ал жауап ретінде сервер ApplicationUser классының экземплярын жібереді, және оның ішінде Access token болады.

Авторизацияланған сұраныс жіберу үшін , Http сұраныстың header – інде authorization параметріне access token – ді жіберуіміз керек, сонда сервердегі Identity компоненті пайдаланушыны танып оның сұранысын орындайды.

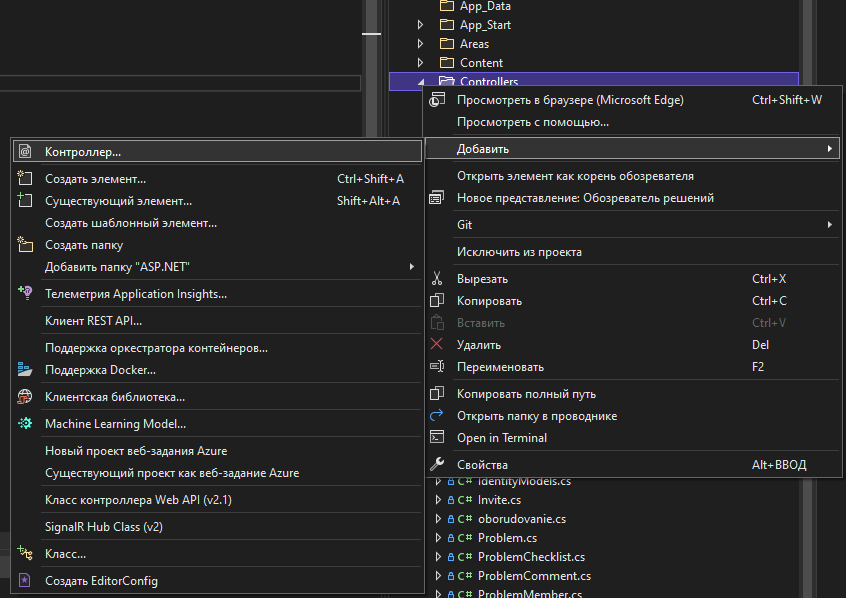
**Контроллерлер құру**

Жобамыздың жұмысы үшін және Дерекқор базасымен жұмыс жасау үшін өзімізге керекті контроллерлерді және фунцияларды құруымыз керек. Ол үшін біздің жұмыс жасайтын ақпараттымыздың моделін құруымыз керек:



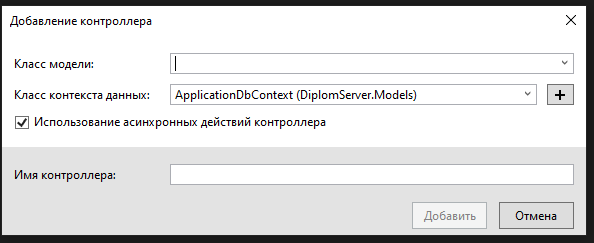
**Сурет 11** Project моделі.

Бұл класс біздің болашақ жобамыздың атын, жобаны құрған пайдаланушының идентификаторын және жоба сипаттамасын сақтауға арналған. Бұл модельмен жұмыс жасайтын контроллерді жасау үшін біз жоба контроллерлерінің папкасына басып контексті меню ішінде Контроллер қосу опциясын тандаймыз:



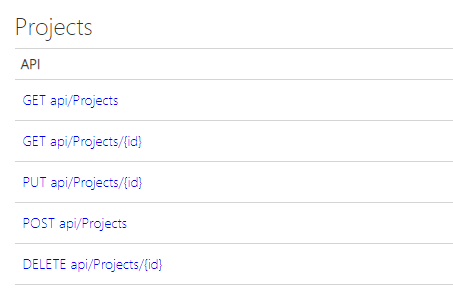
**Сурет 12** Контроллерлер папкасының контесттік менюі.

Осыдан кейін бізге Конроллер моделін тандау диалогі шығады:



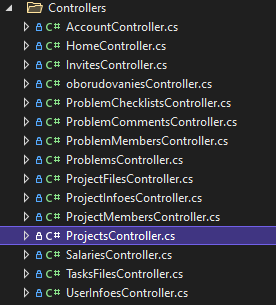
**Сурет 13** Контроллер құру диалогі

Контроллер моделін мен атын тандаған сон , Visual Studio бізге қарапайым Дерекқормен жұмыс жасайтын CRUD (Create,Read,Update,Delete) операцияларды жүргізетін HTTP GET,POST,PUT,DELETE пен шақырылатын функцияларды құрады:



**Сурет 14** Project контроллерінің әдістері.

Контроллер api/Projects маршруты бойынша шақырылады.Осындай әдіспен қалған контроллерлерді құрамыз:

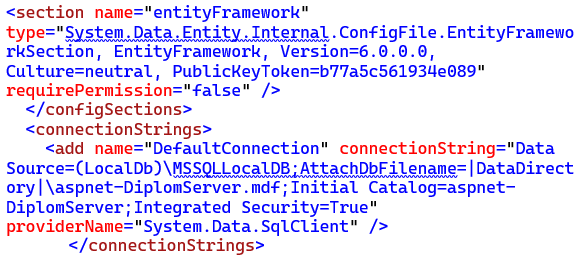


**Сурет 15** Сервер қосымшасының контроллерлері.

Бұл контроллерлер арқылы біз деректер базасына өзгерістерді клиенттік қосымша арқылы жасаймыз. Қауіпсіздік және деректерқорына дұрыс мәліметтер түсу үшін контроллерлерге [Authorize] атрибутын міндетті түрде қосамыз.

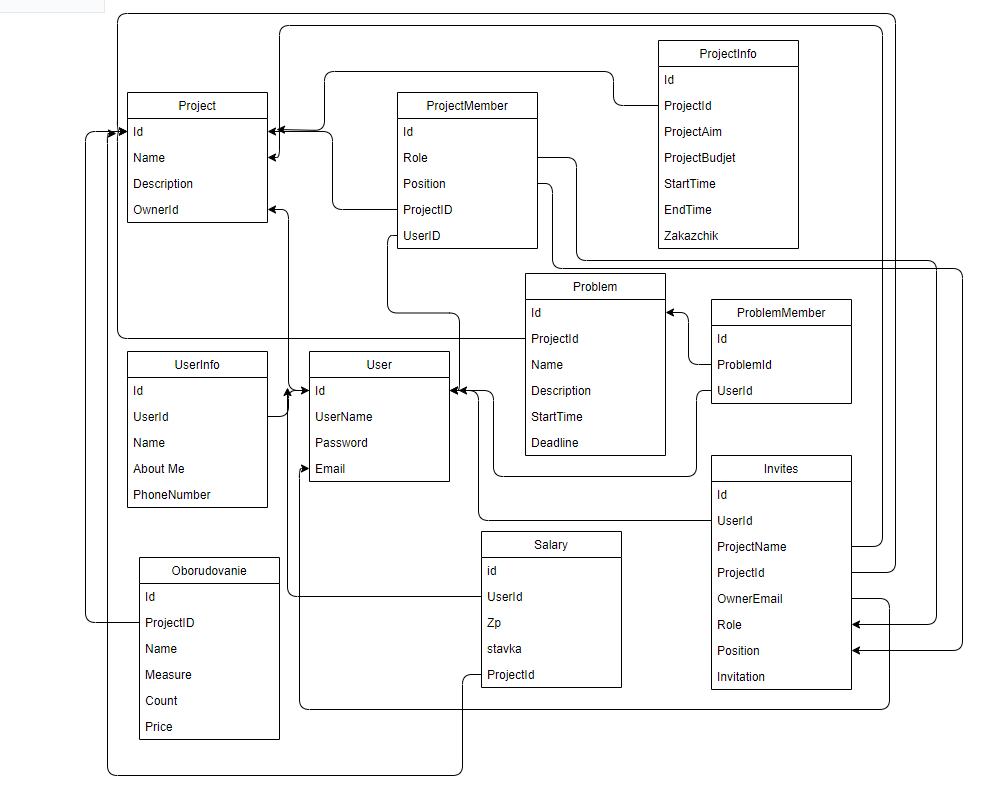
**Деректер қоры**

Деректер қорын серверге қосу үшін біз WebConfig.xml файлында деректер қорына қосылу жолын қосамыз:



**Сурет 16** Деректер қорын қосу.

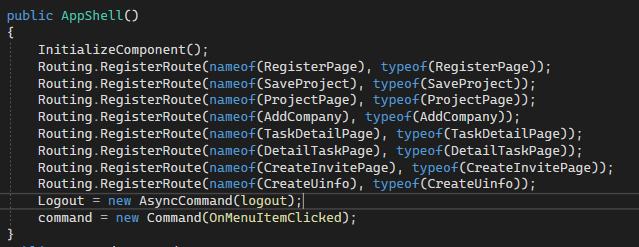
Деректер базасын қосқаннан кейін серверді қосқаннан кейін деректер қорын қолдануға болады. Серверді жасау барысында мен деректердің арасындағы байланысты осындай түрде жүзеге асырдым:



**Сурет 17** Деректер Қорының құрылымы.

## **3.2 Жобаны басқаруға арналған көпқолданушылы қосымшаның клиент қосымшасы**

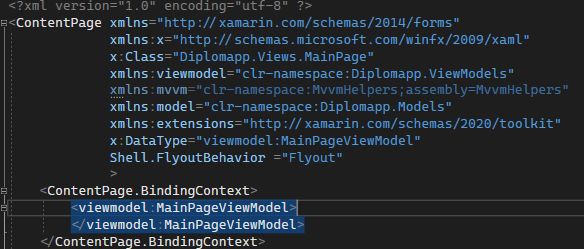
Жобаның клиентін Xamarin.forms фреймворгінде жасадым. Жоба клиенті Shell Navigation шаблоны4 0олдана отырып жасалған. Бұл шаблон арқасында қосымшаның навигациясын жасау женіл болды.Shell Navigation шаблонында , беттер арасында навигация GotoAsync() әдісі арқылы жүргізіледі. Бұл әдіс бізге керекті бетке дейін марштрутты аргімент ретінде қабылдайды. Ал маршрутты қолданғаннан бұрын AppShell.cs классында тіркеу қажет. Ол осылай жасалады:



**Сурет 18** AppShell классындағы тіркелген маршруттар.

Xamarin.Forms фреймворгінде Shell класы бұл барлық интерфейстің аталық классы болып табылады. Бұл класс өз ішінде Content ретінде ContentPage классын қабылдайды. Яғни қосымша беттері ContentPage классының экземлярлары болып табылады.

ContentPage бетімен жұмысты біз Xaml не C# тілдерімен жасай аламыз. Бірақ MVVM паттернің қолдану үшін біз әр бет үшін тағы бір қосымша класс ашамыз, бұл класс MVVM – дегі ViewModel қызметін атқарады. Бұл классқа біз интерфейс бетін байланыстыруымыз қажет, оны жасау үшін ContentPage-тің BindingContext қасиетіне құрылған ViewModel классының аттар кеністігін қосамыз:

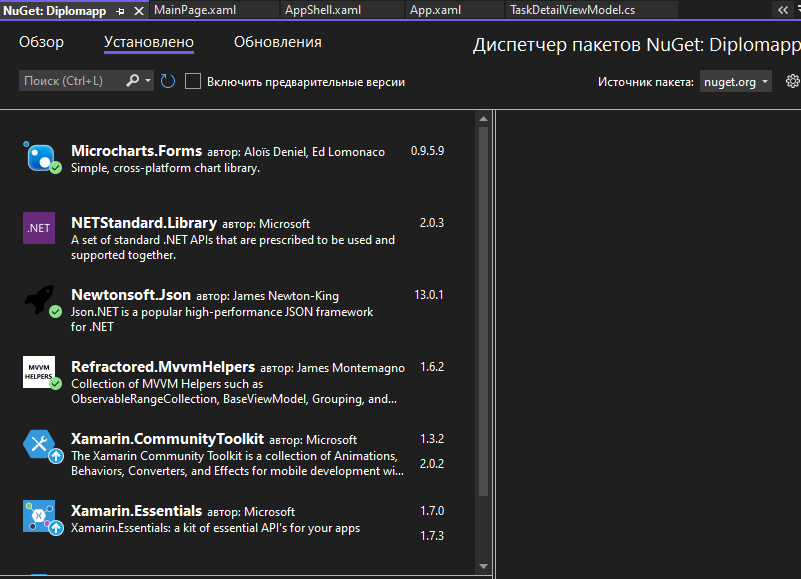


**Сурет 19** View мен ViewModel арасындағы байланыс орнату.

Суретте көрініп тұрғандай аттар кенітігін қосу үшін біз xmlns қасиетін қолданамыз, xmlns-тен кейінгі жазба қосылған аттар кеңістігінің кілттік сөзі болып табылады, кейін бұл аттар кеңістігін кілттік сөз арқылы шақырта аламыз.

Интерфейстегі басқару элементтер ContentPage.Content параметріне салынады, және ол үшін арнайы элементтер контейнерлері қолданылады,менің қосымшамда бұл көбінесе Stacklayout. Ол интерфейс элементтерін бірінен сон бірін орналастырады. MVVM паттерні бойынша біз керекті элементтердің параметрлеріне ViewModel классының қасиеттерін бере аламыз.Қасиет ретінде біз мәтінді, команданы , тіпті логикалық мәндерді бере аламыз, бұл өте ыңғайлы тәсіл болып табылады.

MVVM паттернің қолдануын женілдету үшін біз жобаға қосымша MVVM Helpers және XamarinCommunity Toolkit кітапханалар бумаларын қосамыз, оны істеу өте ыңғайлы NuGet packages Manager Visual Studio – ға қосылған утилита арқылы жасаймыз:



**Сурет 20** NuGet бумалардың менеджері.

Сонымер қатар Http сұраныстармен жұмысты жасау үшін Newtonsoft.Json және Xamarin платформасының қосымша мүмкіндіктерін қолдану үшін Xamarin.Essentials бумаларын оранатамыз.

Класстар мен беттерді құрғаннан кейін бетті толтыруды бастаймыз.Басты бетімізде пайдаланушының Жобалары болу қажет , мен оларды тізім ретінде көрсетемін , ол үшін ListView қолданамыз. Бұл интерфейс элементі тізім ретінде қолданылады, Тізім элементтерін қолмен жазбау үшін оның ItemSource параметрі бар, бұл параметр байланысушы элемент ретінде List классыны4 экземлярын қабылдайды, яғни , біз Project типті серверден алынған ақпаратты List классына қосып , осы тізімді ListView.ItemSource параметріне берсек онда бізде Project типті тізім шығады. Бұл тізім шаблоның тағы өзгертуге болады, ол үшін ListView.ItemTemplate параметрімен жұмыс жасаймыз.

Мысал ретінде MainPage бетіндегі проекттер тізімінің коды:

<ListView HasUnevenRows="True"

CachingStrategy="RecycleElement"

ItemsSource = "{Binding ProjectGroups}"

BackgroundColor="Transparent"

IsGroupingEnabled="True"

SeparatorVisibility="None"

IsPullToRefreshEnabled="True"

ItemTapped ="ListView\_ItemTapped"

ItemSelected="ListView\_ItemSelected"

RefreshCommand="{Binding command}"

>

<ListView.GroupHeaderTemplate>

<DataTemplate x:DataType = "{x:Null}">

<ViewCell>

<StackLayout Orientation="Horizontal" >

<Label Text="{Binding Key}" Margin="15,0,0,0"/>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.GroupHeaderTemplate>

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell x:DataType="model:Project" >

<ViewCell.ContextActions>

<MenuItem Text = " favorite " />

<MenuItem Text = " Delete " />

</ViewCell.ContextActions>

<StackLayout>

<Frame CornerRadius="20" BorderColor="Gray" Margin="10,5,10,5" >

<StackLayout>

<Label Text="{Binding Name}"/>

<Label Margin="10,10,0,0" Text="{Binding Description}"/>

</StackLayout>

</Frame>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

<ListView.Footer>

<StackLayout>

<Button Command = "{Binding command}" Text="Обновить"/>

</StackLayout>

</ListView.Footer>

</ListView>

Бұл жерде аталған параметрлерден басқа біз ListView –дің топтауға арналған функциясын қолданамыз, GroupHeader , footer және тағы басқа параметрлерді қолданамыз.

Негізінен Xaml тілінде интерфейс құру осындай түрде болады. Ал бұл интерфейсті ақпаратпен толтыру және интерфейсті қолдану логикасын біз ViewModel класстарында жасаймыз.

ViewModel класстарында барлық керекті қасиеттер мен командалар жазылады. Серверге сұраныс жіберетін командаларды қарастырайық:

Изображение выглядит как текст

Автоматически созданное описание

**Сурет 21** Login командасының коді.

Бұл команда пайдаланушыны серверге авторизациялайды, және Access token – ді сақтайды. Бұл команданың алгоритмін қарастырайық:

1. HttpRequestMessage обьектін құрады;
2. Бұл обьекттің content қасиетіне KeyValuePair түріндегі тізімді салады;
3. Серверге жіберілетін сұранысқа /Token адресін қосып сұранысқа жауапты алады;
4. Серверден келген жауаптан Content шығарады;
5. Алынған мәтінді Json форматынан token-ді шығарады;
6. Token – ді сақтайды және MainPage бетіне көшеді.

Алынған token арқылы біз серверге авторизацияланған сұраныстар жібере аламыз.Сервермен жұмыс осындай түрде жасалады.

Серверден алынған ақаратты арнайы контейнерлерде сақтаумыз керек , ол үшін ObservableRangeCollection<T> классы өте ынғайлы , ол класстың ішінде MVVM паттернімен жұмыс жасайтын INotifyCollectionChanged параметрі анықталған.

Таза Xamarin.Forms MVVM паттернімен толықтай жұмыс жасауға бейімделмеген болғандықтан мен кейбір оқиғаларды командаларға аударамын деп шештім, оны жасау үшін Xamarin Community Toolkit бумасынан EventToCommandBehavior -ді қолданамыз:

<TabbedPage.Behaviors>

<xtc:EventToCommandBehavior EventName="Appearing"

Command="{Binding initialize}"/>

</TabbedPage.Behaviors>

Оған қоса интерфейс элементтерімен жұмыс жасағанда элементтердің параметрлерін алу мәселесін шешеу керек. Онсыз ViewModel классының мағынасы жоқ болады, өйткені параметрлер ContentPage - тің C# файлында ғана болады. Бірақ бұл мәселені де Xamarin Community Toolkit арқылы шешуге болады. Ол үшін біз EventArgsConverter – ді қолданамыз:

<ContentPage.Resources>

<ResourceDictionary>

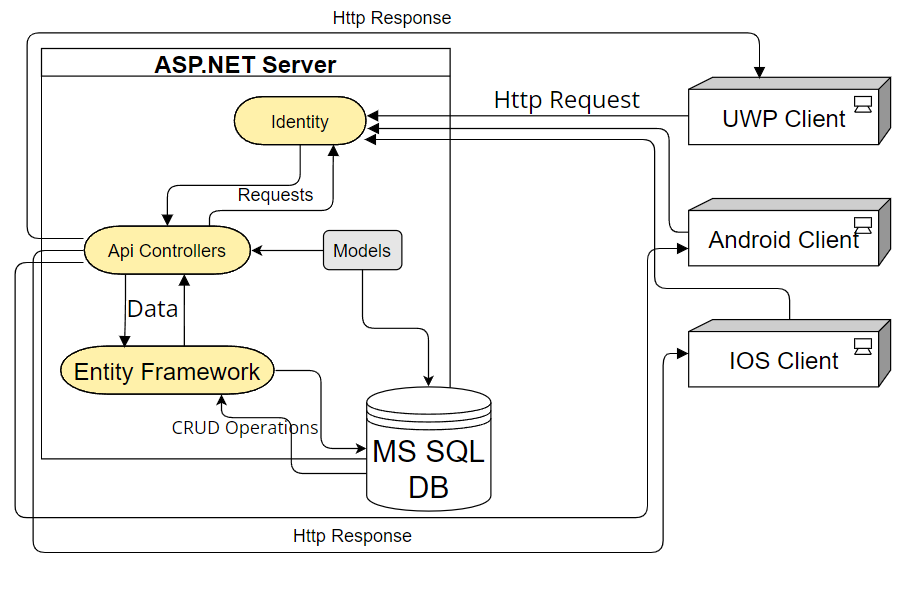
<xtc:ItemSelectedEventArgsConverter x:Key="ItemSelectedEventArgsConverter" />

</ResourceDictionary>

</ContentPage.Resources>

## **3.3 Жобаны басқаруға арналған көпқолданушылы қосымшаның архитектурасына шолу.**

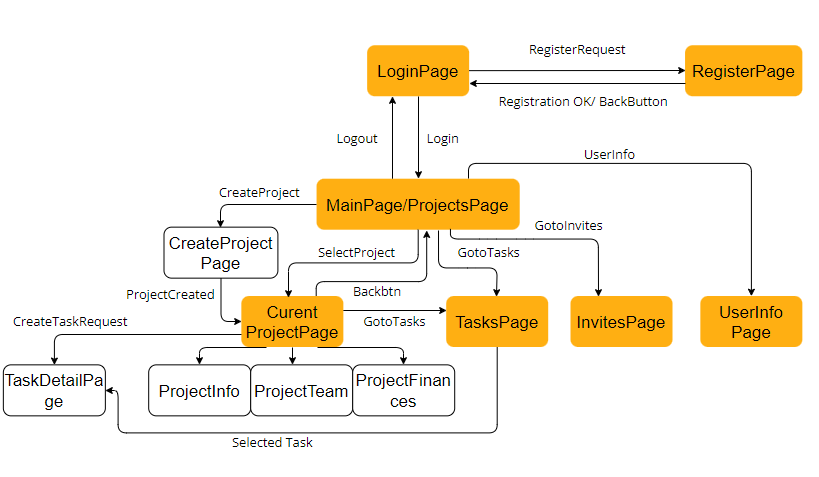
Жоба бойынша мен екі қосымшаны әзірлеу керек болған, яғни, біріншісі- сервер,екіншісі-клиент қосымша болуы керек еді. Сондықтан менің жүйемнің архитектурасы осындай болды:



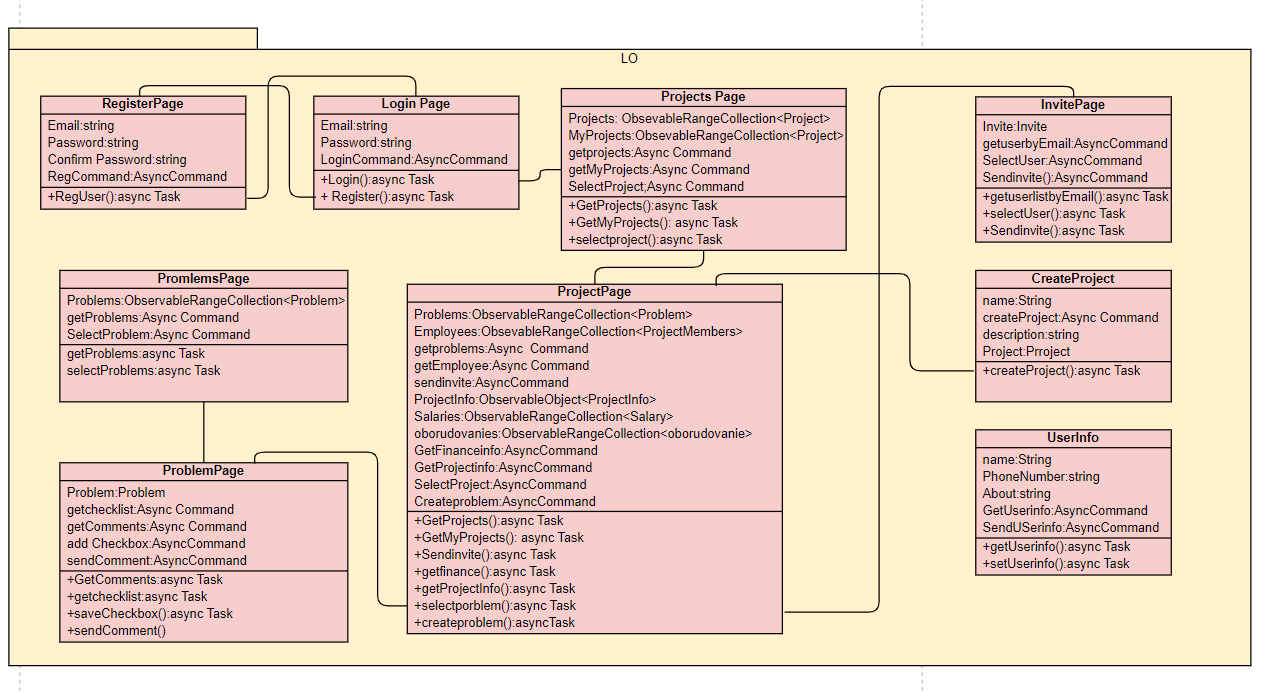
**Сурет 22** Жоба жүйесінің архитектруралық сызбасы.

Суретте көрініп тұрғандай клиент пен сервер арасындағы ақпарат алмасу Http протоколы арқылы жүзеге асады. Серверге келген сұраныстарды контроллерлер өңдейді , ал контроллерге дейін пайдаланушыны Identity компоненті тексереді, және пайдаланушы туралы ақпаратты контроллерге береді.

Клиент қосымшасының интерфейсі осындай навигациялық стек ағашы бойынша өзгертіледі.

 **Сурет 23** Навигациялық стек ағашы.

Клиенттік қосымшаның ViewModel негізгі класстары (RegisterPage, LoginPage, ProjectsPage ,InvitePage ,CreateProjectPage ) интерфейске ақпаратты дайындау,өндеу және жіберумен айналысады. Осы класстардың диаграммасы:



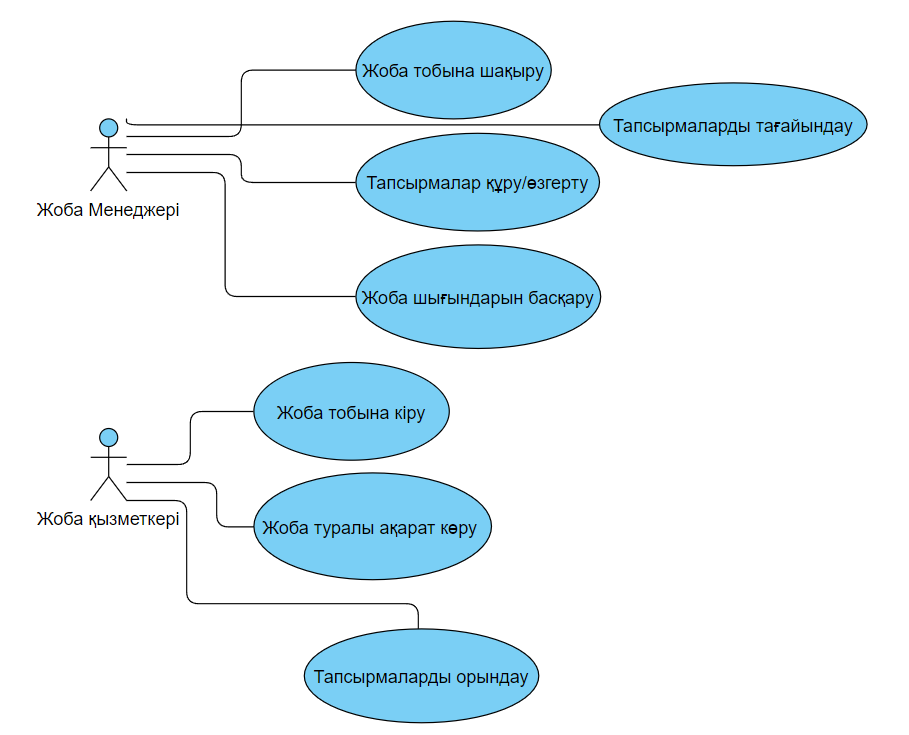
**Сурет 24** Класстардың UML диаграммасы

Осы схемада көрініп тұрғандықтай, Класстардағы ақпаратты жинаушы ретінде ObservableRengeCollection классының экземплярлары қоладнылады, бұл класстың тиімді жағы, ол интерфейсті қайта салу оқиғасын бір рет шақырады, ал бұл біздің интерфейсіміздің тез жұмыс жасауына көмектеседі.

Класстардағы әр әдіс async кілттік сөзімен бірге жазылған, яғни бұл әдістер ассинхронды түрде жүреді, ал ол біздің интерфейстің ойланып тұрып қалуына жол бермейді. Әр осындай әдіс AsyncCommand типті командада шақырылады, ал біз бұл командаларды MVVM паттерні бойынша интерфейсте шақырамыз.

Жобадағы пайдаланушылардың екі рөлі анықталған, рөлдер бойынша пайдаланушылардың жобадағы ақпаратқа және онымен жұмыс жасау функцияларына әртүрлі рұқсаттар беріледі.

Пайдаланушылардың UseCase диаграммасы:



**Сурет 25** Use Case диаграммасы.

Жоба менеджері рөлі жобаны құрғаннан кейін жобаны құрған пайдаланушыға автоматты түрде беріледі.Ал жоба қызметкері бұл жерде жобаға қосылған кезде жобадағы атқаратын қызмет атауымен бірге беріледі.

## **4 Жобаны қолданысқа енгізу**

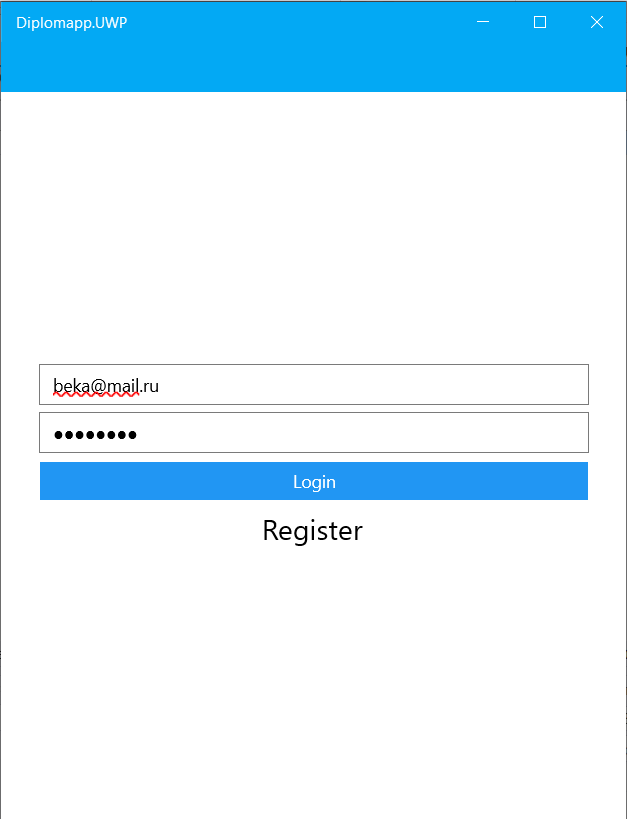
Қосымшамен жұмыс жасау үшін алдымен пайдаланушыны тіркеу керек. Қосымшада тіркелу үшін еmail , құпиясөзді және құпиясөзді расстау жолдарын толтырып Register батырмасын басу қажет.

Изображение выглядит как текст

Автоматически созданное описание

**Сурет 26** Қосымшаға тіркелу беті.

Тіркелгеннен кейін қосымша кіру бетін ашады:



**­Сурет 27** Қосымшаға кіру беті.

Бұл бетте біз тіркелген аккаунттын поштасын және құпиясөзін жазып login батырмасы арқылы кіреміз.

Кірген кезде бізді жобалар тізімінің беті қарсы алады:

Изображение выглядит как текст

Автоматически созданное описание

**Сурет 28** Қосымшаның басты беті.

Бұл бетте біз құрылған жобаға кіре аламыз , қатысушы ретінде кірген жобаларға кіре аламыз , және жаңа жоба құра аламыз. Жаңа жоба құру батырмасын бассақ, жаңа жобаны құру беті ашылады:

Изображение выглядит как текст

Автоматически созданное описание

**Сурет 29** Жаңа жоба құру беті.

Бұл бетке жаңа жоба туралы ақпарат енгізгеннен кейін сақтау батырмасын басудан кейін , жоба туралы қосымша ақпарат қосу беті ашылады:

# Қорытынды

Қазіргі уақытта интерактивтік бағдарламалар өте көп таралған. Оның себебі – цифрландыру. Ақпаратты басқаша, интерактивті түрде беру біздің заманымызда ең маңызды істердің арасында. Интерактивті кітаптар қазіргі заманда көп таралған бағдарлама. Егер сіз кітап оқуды ұнататын болсаңыз, бірақ қарапайым кітаптарға уақыт жетіспейтін болса, онда Android үшін интерактивті кітаптар дәл сізге қажет. Жалпы алғанда, бұл бағдарламалар классикалық мобильді "оқырмандар" болып табылады, бірақ әлдеқайда жетілдірілген және жетілдірілген функционалдығы бар. Осындай бағдарламалардың көмегімен смартфонның экранында сүйікті кітабыңызды оқу процесі нағыз рахатқа айналады, оны тастау қиын. Интерактивті кітаптар Тек ересектер үшін ғана емес, балалар үшін де болуы мүмкін. Тірі кейіпкерлері бар ертегіден гөрі балаға не тартымды болуы мүмкін. Қазіргі уақытта көптеген интерактивті кітаптар бар және олардың ауқымы үнемі кеңеюде. Бұл қосымшалардың ерекшелігі-жүйенің өте төмен талаптары, сондықтан оларды техникалық тұрғыдан ескірген Android смартфондарына да орнатуға болады.

Интерактивтік көбінесе тарихи бағдарламаларға қолданылады. Мысалы: Ұшақтар, қару, музыкалық аспап мұражай бағдарламалары

Қазіргі білім берудің басты міндеті-оқушыға іргелі білім беру ғана емес, сонымен бірге оған одан әрі әлеуметтік бейімделу үшін барлық қажетті жағдайларды қамтамасыз ету, өзін-өзі тәрбиелеуге бейімділікті дамыту.

Қазіргі мұғалім үшін басты міндеттердің бірі – оқу процесін студенттер үшін қызықты, динамикалық және заманауи ету. Бұл ретте педагогтарға интерактивті технологиялар көмекке келді. Ақпараттық және коммуникациялық технологиялар (АКТ) білім берудің барлық салаларына біртіндеп еніп келеді. Бұған қоғамды жаһандық ақпараттандыру, мектептер мен ЖОО-ларда жаңа компьютерлік техника мен қазіргі заманғы бағдарламалық қамтамасыз етуді тарату, білім беруді ақпараттандыруға бағытталған мемлекеттік және халықаралық бағдарламаларды құру ықпал етеді.

Біздің негізгі мақсат жоғарыда айтылған ақпараттың барлығын қорыта отырып жаңа заманауи технологиямен ежелгі тарихымызбен халқымыздың рухани байлығын одан әрі дамыту және заман талабымен ұштастыру. Әр бір қазақ баласы өз тарихын біліп қайдан шыққанын білуге итермелейтін кішкене ғана зат жасау біздің арманымыз.

Келешек ұрпаққа тура осы жолмен тарихымызды қалдырмасақ қазіргі кезде мұражайларда ескіріп бара жатқан кітаптарымызды оқуға ешкімнің де құлқы жоқ. Ол кітаптарымызды оқығанның өзінде ұстап көру мүмкіншілігі өте аз. Біз өзіміздің бағдарламамыз арқылы сол бұйымдарды оқып, 3д форматта көзімен көріп, ұстап сезіну мүмкіндігін беру.

Еліміздің алдыға өркендеп дамуы әрине өте керемет, бірақ тура сол өрккендеу бағытында тарихымыз мен салт-дәстүрімізді ұмытпайық. Себебі өзінің өткенін білмейтін халықтың жарқын болашағына мен кәміл сене алмаймын. Әр адам қолынан келгенше осы жолда еңбек етсе барлығы жақсы болады.

Бағдарламаның ішінде 3 тілдік формат болғандықтан біздің елге қызығатын әр-бір туристке бұл баға жетпес ақпарат болады, және рухани жаңғыру кезеңімен тұспа тұс келгендіктен, онымен қоса Абай атамыздың 175 жылдығына бағдарламаның ішіне «Көзімнің қарасы» - өлеңін фонға қойдық.

Еліміздің болашағы бізде, яғни жастарда. Барлығы өз қолымызда, бірге шарықтайық, бірге дамытайық елімізді…

# Пайдаланылған әдебиеттер тізімі

1. Ешин С.С. өнімділік критерийі негізінде Android платформасына арналған ORM кітапханаларын салыстырмалы талдау. // Информатика: мәселелер, әдіснамалар, технологиялар XV халықаралық ғылыми-әдістемелік конференцияның материалдары(12-13 ақпан 2015, Воронеж). Воронеж: Воронеж мемлекеттік университетінің баспа-полиграфиялық орталығы, 2015. – Б.276-280.

2. Кирпичников а.п., Ляшева С. А., Шлеймович М. П., Еремеев Д. Е. мобильдік құрылғыларға арналған қосымшалар арқылы деректер базасымен пайдаланушылардың өзара іс-қимылының автоматтандырылған жүйесі. // Қазан технологиялық университетінің хабаршысы, 2015. – Т.18. – № 3. – Б.235-238.

3. Малышкина Е.А. тұтынушыға тиімді әсер ететін фактор ретінде интернет-бизнестегі маркетингтік құралдарды жетілдіру. // Әлеуметтік-экономикалық құбылыстар мен процестер. - Тамбов атындағы мемлекеттік университеті. Г. Р. Державина, 2012. – Б.136-146.

4. Google Play–дегі "Афиша-Мейрамханалар"қосымшасы [Электрондық ресурс]. Қолжетімділік режимі: *https://play.google.com/store/apps/details?id=ru.afisha.restaurants* (Сұраныс күні: 16.01.2021).

5. Google Play қолданбасы "кестелер" [Электрондық ресурс]. Қолжетімділік режимі: *https://play.google.com/store/apps/details?id=com.saas.app* (Сұраныс күні: 17.01.2021).

6. Google Play қолданбасы "ТоМесто – ең жақсы мейрамханалар". [Электрондық ресурс]. Қолжетімділік режимі: *https://play.google.com/store/apps/details?id=ru.tomesto.tomesto* (Сұраныс күні: 19.01.2021).

7. Google Play қолданбасы "EatOut" [Электрондық ресурс]. Қолжетімділік режимі: *https://play.google.com/store/apps/details?id=ru.yell.rest* (Сұраныс күні: 19.01.2021).

8. Google Play қолданбасы "ерін" [Электрондық ресурс]. Қолжетімділік режимі: *https://play.google.com/store/apps/details?id=ru.lipe.android.activity* (Сұраныс күні: 20.01.2021).

9. Google Play қолданбасы "RestOn – кестелерді брондау" [Электрондық ресурс]. Қолжетімділік режимі: *https://play.google.com/store/apps/details?id=com.reston.cs.company.android* (Сұраныс күні: 22.01.2021).

10. Google Play қолданбасы "Toptable" [Электрондық ресурс]. Қолжетімділік режимі: *https://play.google.com/store/apps/details?id=com.safaroff.toptable* (Сұраныс күні: 25.01.2021).

11. Харди Б., Филлипс Б. Android үшін бағдарламалау. – СПб.: Петр, 2014. – 592 б.

12. Шевченко Д. деректер базасы құрылымының нұсқалық көші-қоны: негізгі тәсілдер [Электрондық ресурс]. Қолжетімділік режимі: *https://habrahabr.ru/post/121265* (Сұраныс күні: 26.02.2021).

13. Android Developers Dashboard [Электрондық ресурс]. Қолжетімділік режимі: *https://developer.android.com/about/dashboards/index.html* (Сұраныс күні: 08.03.2021).

14. Android Development Tool [Электрондық ресурс]. Қолжетімділік режимі: *https://developer.android.com* (Сұраныс күні: 18.02.2021).

15. Django documentation [Электрондық ресурс]. Қолжетімділік режимі: *https://docs.djangoproject.com* (Сұраныс күні: 19.02.2021).

16. Firebase Cloud Messaging. [Электрондық ресурс]. Қолжетімділік режимі: *https://firebase.google.com/docs/cloud-messaging* (Сұраныс күні: 20.02.2021).

17. Unity в действии. Мультиплатформенная разработка на C#. - М.: Питер, 2018. - **608** б.

18. Арстанова, Л.Г. Занятия и развлечения со старшими дошкольниками. Разработки занятий, бесед, игр и развлечений на нравственные темы / Л.Г. Арстанова. - М.: Учитель, 2017. - **324** б.

19. Архангельская, М.Д. Бизнес этикет, или игра по правилам / М.Д. Архангельская. - М.: Эксмо, **2015**. - 160 б.

20. Вакуленко, Ю.А. Веселая грамматика. Разработки занятий, задания, игры / Ю.А. Вакуленко. - М.: Учитель, 2017. - **780** б.

21. Джейсон, Финкэнон Flash-реклама. Разработка микросайтов, рекламных игр и фирменных приложений с помощью Adobe Flash / Финкэнон Джейсон. - М.: Рид Групп, 2012. - **945** б.

22. Джейсон, Финкэнон Flash-реклама. Разработка микросайтов,рекламных игр и фирменных приложений с пом / Финкэнон Джейсон. - М.: РИД ГРУПП ООО Москва, 2012. - 288 б.

23. Любанова, Т.П. Бизнес-план: опыт, проблемы. Содержание бизнес-плана, пример разработки / Т.П. Любанова, Л.В. Мясоедова, Т.А. Грамотенко, и др.. - М.: Приор, **2012**. - **204** б.

24. Паласиос, Хорхе Unity 5.x. Программирование искусственного интеллекта в играх / Хорхе Паласиос. - М.: ДМК Пресс, 2016. - **849** б.

25. Платов, В. Я. Деловые игры. Разработка, организация, проведение. Учебник / В.Я. Платов. - М.: Профиздат, **2010**. - 192 б.

26. Таран, В.А. Играть на бирже просто?! / В.А. Таран. - М.: СПб: Питер, **2016**. - 272 б.

27. Тарп, Ван; Бартон Д.Р. Биржевые стратегии. Игры без риска / Тарп, Ван; Д.Р. Бартон, С. Сьюггеруд. - М.: СПб: Питер, **2010**. - 400 б.

28. Финни, К. 3D-игры. Все о разработке (+ CD-ROM) / К. Финни. - М.: Бином. Лаборатория знаний, **2011**. - 976 б.

29. Финни, К. 3D-игры: Все о разработке (+ CD-ROM) / К. Финни. - М.: Бином. Лаборатория знаний, 2015. - **133** б.

30. Хорхе, Паласиос Unity 5.x. Программирование искусственного интеллекта в играх. Руководство / Паласиос Хорхе. - М.: ДМК Пресс, 2017. - **427** б.

31. Шабельникова, Е.Ю. Английский язык. Обучение детей 5-6 лет. Разработка занятий, лингвострановедческий материал, мероприятия, игры / Е.Ю. Шабельникова. - М.: Учитель, 2015. - **557** б.

32. Язев, Ю. Волшебство момента вращения. Искусство разработки игр на движке Torque 2D, включает описание версий 3.2 и 3.3 / Ю. Язев. - М.: Солон-Пресс, 2016. - 448 б.