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: Blockchain технологиясын пайдалана отырып, «Мұғалім студент көзімен» атты сауалнама жүргізетін бағдарламалық қамтама құру

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

Орындаған: Нуралиева А. Е. Тобы: ВТк-15-1
Ғылыми жетекші: PhD, доцент Картбаев Т.С.

Кеңесшілер:
Экономикалық бөлім: э.ғ.к., профессор _____ Ж.Г. Аренбаева
«10» 05 2019 ж.

Өміртіршілік қауіпсіздігі: т.ғ.д., аға оқытушы _____ Ш.Ш. Бекбасаров
«17» 05 2019 ж.

Есептеу техникасын қолдану: аға оқытушы _____ Ж.С. Айткулов
«15» 05 2019 ж.

Норма бақылаушы: аға оқытушы _____ К. Мукапил
«17» 05 2019 ж.

Сын-пікір беруші: PhD, acc. проф _____ Н. К. Мукажанов
«____» _____ 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 Нуралиева Асель Ерисовна

Жобаның тақырыбы: Blockchain технологиясын пайдалана отырып, «Мұғалім студент көзімен» атты сауалнама жүргізетін бағдарламалық қамтама құру

2019 жылғы «01» мамыр № 33 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 Blockchain технологиясын қолдана отырып сауалнама құрастыратын, жүргізетін және талдайтын бағдарламалық қамтама құру. Жобаны орындау барысында Ruby және JavaScript тілідерін, PostgreSQL деректер қорын қолданамын.

Дипломдық жобада қарастырылған мәселелер тізімі немесе дипломдық жобаның қысқаша мазмұны:

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

Графикалық материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс): 12 кесте, 31 сурет ұсынылған.

Ұсынылатын негізгі әдебиеттер:

1 Фернандес, Оби Путь Rails. Подробное руководство по созданию приложений в среде Ruby on Rails / Оби Фернандес. - М.: Символ-плюс, 2009. - 768 с.

2 Bruce, A Tate Ruby on Rails – Up and Running / Bruce A Tate. - Москва: Мир, 2006. - 182 с.

3 Jeremy, McPeak JavaScript 24–Hour Trainer / Jeremy McPeak. - Москва: Гостехиздат, 2010. - 360 с.

4 Чаффер, Джонатан Изучаем jQuery 1.3. Эффективная веб-разработка на JavaScript / Джонатан Чаффер , Карл Шведберг. - М.: Символ-плюс, 2010. - 448 с.

5 Дрешер Д. Основы блокчейна; ДМК Пресс - М., 2018. - 735 с.

Дипломдық жобаның бөлімдеріне қатысты белгіленген кеңес берушілер

Бөлімдер	Кеңесшілер	Мерзімі	Қолы
Экономикалық бөлім	Аренбаева Ж.Г.	04.03.2019 – 10.05.2019	
Өміртіршілік қауіпсіздігі	Бекбасаров Ш.Ш.	04.03.2019 – 14.05.2019	
Программалық қамтама	Айткулов Ж.С.	04.03.2019 – 02.05.2019	
Норма бақылау	Мукапил К.	04.04.19 – 10.05.2019	

Дипломдық жобаны дайындау
КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Талдау бөлімі	15.11.2018	орындалды
Жобалау бөлімі	23.01.2019	орындалды
Жүзеге асыру және тестілеу бөлімі	04.02.2019	орындалды

Тапсырманың берілген күні «3» 09 2018 ж.

Кафедра меңгерушісі _____ Т.С. Картбаев

Жобаның ғылыми жетекшісі _____ Т.С. Картбаев

Тапсырманы орындауға алған
білім алушы _____ А. Е. Нуралиева

Аңдатпа

Дипломдық жобаның негізгі міндеті блокчейн технологиясын қолдана отырып, бағдарламалық қамтама құру болып табылады, оның көмегімен анонимді сауалнамаларды құрастырып, өткізуге болады. Бағдарлама Ruby серверлік тілі, JavaScript бағдарламалау мультипарадигмендік тілі арқылы іске асырылды, сондай-ақ PostgreSQL деректер базасы құрылды. Сауалнамалардан алынған жауапты сақтау үшін, блокчейн технологиясының принциптері қолданылды. Осы технологиясының көмегімен қайсыбір өзгертуге және хакерлік шабуылдарға қарсы тұра алатын бағдарлама құрылды. Әзірленген жүйеде графикалық түрде оңай жаңа сауалнамаларды құруға болады, сондай-ақ алынған деректерге байланысты сұрыптау арқылы талдау өткізуге болады. Бұл дипломдық жобада бағдарламалық қамтама әзірлеудің оңтайлы құралдарын таңдау, сондай-ақ осы жүйені құру жұмыстары жүргізілді.

Аннотация

Основной задачей дипломного проекта является создание программного обеспечения с использованием технологии блокчейн, с помощью которого можно составить и проводить анонимные опросы. Программа реализована с помощью серверного языка Ruby, мультирадигментированного языка программирования JavaScript, а также СУБД PostgreSQL. Для сохранения ответа от анкетирования были использованы принципы технологии блокчейн. С помощью этой технологии была создана программа, которая способна противостоять каким-либо изменениям и хакерским атакам. В разработанной системе можно графически легко создавать новые анкеты, а также проводить анализ путем сортировки в зависимости от полученных данных. В данной дипломной работе был проведен выбор оптимальных средств разработки программного обеспечения для создание оптимальной системы.

Annotation

The main purpose of the thesis is to create software using blockchain technology, with which you can create and conduct anonymous surveys. The program is implemented using Ruby server language, multi-pigmented JavaScript programming language, and PostgreSQL DBMS. The principles of blockchain technology were used to save the response from the questionnaires. With this technology was created a program that is able to withstand any changes and hacker attacks. In the developed system, it is provided to create new questionnaires graphically easily, and also to carry out the analysis by sorting depending on the received data. In this thesis work, a selection of optimal means of development of software was carried out for the development of an optimal system.

Мазмұны

Кіріспе

Инновациялық заманда, күн сайын жаңа технологиялар мен жаңалықтар ашылып жатыр. Санының көптігіне қарамастан, олар бір мақсатты көздейді – ол адам өмірін жеңілдету және ыңғайлау. Бағалы құжаттар, ақша және басқа да меншік көздері пайда болғаннан кейін қоғамда бір-бірімен алмасу кезінде сенімсіздік мәселесі туды. Сондықтан да, адамдардың өзара әрекеттесу үшін үшінші жақ, яғни нотариус, банк және сот сияқты қызметтері қажет болды.

Қазіргі уақытта, блокчейн таратылған тізім технологиясы танымал әрі өзекті жаңалықтардың бірі болып табылады. Себебі, бұл технология үшінші жаққа жүгінбеуге, дәлірек айтқанда бағалы меншікпен алмасу кезінде олардың көмегінсіз іс-әрекеттерді орындауға мүмкіндік береді. Бағалы меншікті әркім әр қалай түсінеді, біреуге бұл ақша немесе жылжымайтын мүлік болса, ал басқаларына ол дерек болуы мүмкін.

Басқа технологияларға қарағанда, блокчейн технологиясының негізгі ерекшелігі ақпаратты сақтау принципінде болып табылады. Ақпаратты хешқа түрлендіріп, өзара байланысқан деректер тізбегінің макетін жасайды. Осының арқасында базада сақталған деректер қорын өзгертуге болмайды.

Дипломдық жұмыстың мақсаты айтылған технологияны пайдалана отырып, «Мұғалім студент көзімен» атты сауалнама жүргізетін бағдарлама қамтамасын құру болып табылады. Бұл бағдарлама арқылы графикалық түрде сауалнама құруға және өткізуге болады. Студенттен жауап алынғаннан кейін ақпаратты сұрыптауға және талдауға болады, тек өзгертуге, жоюға болмайды.

Бұл жұмыста сауалнамадан жіберілген жауаптардың тарихын жеке блокчейнде сақтау жүргізілді. Мұндай бағдарламалық қамтама құрудың бірқатар артықшылықтары бар:

– өзгертілмеген сауалнама қорытындыларын алынады;
– сауалнама құру жеңілділігі;
– сауалнаманы талдау визуальды құралдар арқылы жүргізіледі;
– базада тек қорытынды хштарын сақтағандықтан тез жұмыс істейді.

Осылайша, дипломдық жобаның тақырыбы біздің нарық аясында өте өзекті болып табылады.

Зерттеу нысаны - ЖОО-да міндетті сауалнама жүргізу жүйесі болып табылады.

Зерттеу пәні - сауалнама жасау, талдау және өткізу үшін арналған жүйе.

Дипломдық жобаның мақсаты - Blockchain технологиясын пайдалана отырып, «Мұғалім студент көзімен» атты сауалнама жүргізетін бағдарлама қамтамасы бағдарламалық қамтаманы құру.

Қойылған мақсатқа жету үшін келесі міндеттерді шешу қажет:

– блокчейн технологиясының теориялық негіздерін үйрену;
– бағдарламалық қамтамасыз етуді әзірлеу әдістерінің бірін таңдау, таңдауды негіздеу;
– frontend, backend және деректер базасын әзірлеуді қамтитын жүйені әзірлеу.

1 Талдау бөлімі

1.1 Пәндік аймақтың сипаттамасы

Пәндік аймақты талдау – бұл жүйелік талдау кезеңінің бірінші қадамы, одан кейін бағдарламалық жүйені әзірлеу басталады. Әзірлеушілер үйрену керек:

– тапсырыс берушілер айтқан тілді түсіну;
– олардың қызметінің мақсатын анықтау;
– олар шешетін міндеттердің жиынтығын анықтау;
– осы есептерді шешу кезінде мәндер жиынтығын анықтау.

Кез келген түрдегі ДБ жобалаудың бірінші кезеңі – концептуалды сұлба (ақпараттық құрылым) құрумен аяқталатын пәндік саланы талдау болып табылады.

Бұл кезеңде пайдаланушылардың сұраулары талданады, жобаланатын ДБ мазмұнын анықтайтын ақпараттық объектілер мен олардың сипаттамалары таңдалады. Өткізілген талдау негізінде пәндік аймақ құрылады.

Пәндік саланы талдау үш кезеңге бөлінеді:

– тұжырымдамалық талаптар мен ақпараттық қажеттіліктерді талдау;
– ақпараттық нысандарды және олардың арасындағы байланыстарды анықтау;
– пән аймағының концептуалды моделін құру және ДҚ тұжырымдамалық сұлбасын жобалау.

Пәндік аймақты талдаудың бірінші фазасы кезінде әзірлеуші осы ДҚ-ның болашақ пайдаланушыларының сұраныстарының тізімін алады.

Пәндік саланы талдаудың екінші фазасы ақпараттық объектілерді таңдаудан, әрбір объект үшін қажетті қасиеттерді бекітуден, объектілер арасындағы байланыстарды анықтаудан, ақпараттық объектілерге салынатын шектеулерді анықтаудан, олардың арасындағы байланыс типтерінен, ақпараттық объектілердің сипаттамасынан тұрады.

Пәндік саланы талдаудың қорытынды фазасы оның ақпараттық құрылымын немесе тұжырымдамалық моделін жобалаудан тұрады. Тұжырымдамалық модель деректерді талдау нәтижесінде анықталатын объектілерді және олардың өзара байланысын сипаттауды қамтиды.

Пәндік аймақ – басқаруды ұйымдасыру, соңында, автоматтандыру мақсатында зерттеуге келетін нақты әлемнің бөлігі. Пәндік аймақ көптеген бөліктерден тұрады, мысалы, кәсіпорын – цехтардан, бухгалериядан т. б. Осы пәндік аймақтың әр бір бөлігі, яғни бөлігі көптеген объекттер мен үрдістермен сипатталады, сонымен қатар бұл пәндік аймақ әр түрлі көзқарастары бар тұтынушылармен де сипатталады.

Концептуалды модель жүйені пайдаланушылардың ақпараттық мүдделерін ескере отырып, пәндік саланы құрылымдау үшін қолданылады. Ол пайдаланушының пәндік салаға көзқарасы болып табылады және тұрақты,

яғни өзгеріссіз болуы керек. Оны тек ұзартуға және қосымша деректерді қосуға болады.

Концептуалды схеманың кең таралған үлгілерінің бірі "мән – байланыс" моделі болып табылады, оның негізгі компоненттері мән мен байланыс болып табылады [1].

Бұл дипломдық жұмыстың мақсаты, яғни мен зерттегелі отырған пәндік аймақ – Blockchain технологиясын пайдалана отырып, «Мұғалім студент көзімен» атты сауалнама жүргізетін бағдарламалық қамтама құру. Ең алдымен, бұл тақырыптың өзектілігі және Университет өміріндегі маңызы қаншалықты екенін білу үшін, төмендегі сұрақтарға жауап іздейік:

– оқыту деңгейін бағалайтын сауалнамалар қандай мақсатпен құрылады?

Әрбір ЖОО-ның негізгі міндеттерінің бірі студенттерге жоғары сапалы білім беру, ал, бұны бағалаудың ең дұрыс және тиімді әдісі – ол сауалнама жүргізу. Қазіргі кезде, көптеген оқыту орталықтарында сауалнамалар арқылы керекті ақпарат жиналады, сонымен қатар бұл ақпарат іштен келген соң, оқытылып жүрген студенттердің мұғалім және жалпы білім беру деңгейі туралы шынайы ақпарат береді;

– сауалнамалар ЖОО оқыту саясатына қандай ықпал тигізеді?

Сауалнаманы өткізгеннен кейін, қорытындылар жиналып, анализ істелінеді. Нақты мұғалімнің алынған ақпаратқа байланысты портреті құрылады. Сауалнама анонимді жүргізілетіндіктен студент өзін еркін ұстап, қысымсыз әрбір сұраққа жауап береді. Сонымен қатар, мұндай әдіс жемқорлыққа қарсы бағытта да қолданылуы мүмкін, яғни сауалнамалар ЖОО-мен ортақ мақсатты орындауға көмектеседі;

– блокчейн технологиясынын сауалнамаға қандай қатысы бар?

Соңғы онжылдықта «блокчейн» термині әлемдегі инновациялық жаңалықтарда, форумдарда, мақалаларда қызу пікір-таластарды оятқан. Бұл технология деректерді сақтаудың жаңа принципімен байланысты. Деректер қорындағы ақпаратты өзгертуге және қайсыбір хакерлік шабуылдарға қарсы тұрады. Яғни, бір сақталынған дерек мұндай қорда әрдайым сақталулы тұрады. Блокчейн сауалнама жүргізуге өте қолайлы технология болып табылады. Себебі, деректер қорына түскен әрбір қорытынды ақпарат ешқандай өзгертуге шалдыға алмайды.

«Мұғалім студент көзімен» атты сауалнама жүргізетін бағдарламаның мақсаты:

– мұғалімдердің кәсіби шеберлігі және сапасы туралы студенттердің пікірін зерттеу;

– ЖОО-ның білім беру процесінің тиімділігін арттыру және жетілдіру;

– мұғалім өзін педагог ретінде жетілдіру, студенттермен "кері байланыс" негізінде педагогикалық қызметінің сапасын арттыру;

– студенттердің білім беруді жетілдіру мақсатында ұсыныстарын жинау;

– ЖОО басшылығына мұғалімдердің педагогикалық қызметі туралы ақпараттың толық көлемімен қамтамасыз ету.

1.2 Бағдарламалық құралдарға талдау

Қазіргі уақытта қолданушыларға мыңдаған бағдарламалық өнімдер ұсынылып, әрбір күн сайын жаңа өнімдер пайда болып жатыр. Бағдарлама құрған кезде frontend және backend құрылымдары құрылады. Frontend бөлімінде сайтты жүктеу, жылдам әрі дұрыс кодтың жасалуы, backend - сервердің кодын жазу болып табылады, яғни, бұл программа full-stack жұмысты талап етеді.

Бағдарламаның frontend бөлігін құру үшін жады және жылдамдығы жағынан тиімді көптеген бағдарламалық тілдер бар. Соның ішінде ең танымалдары HTML белгілеу тілі, CSS рәсімдеу тілі, Java Script бағдарламалау тілі, ал backend серверлік бағдарламалау тілдерін Ruby, PHP, Python т.б. қолданады. Деректер базасы ретінде SQL, PostgreSQL, MangoDB таңдалуы мүмкін.

Бағдарламалау тілін таңдау талаптары төмендегідей:

а) Соңғы өнімнің жұмыс жылдамдығы. Нарықтың сұранысына байланысты, физикалық жүйелерді моделдеу, экономикалық мәліметтердің үлкен көлемін есептеу, үш өлшемді графиканы шығару және т.б. орындау жылдамдығына талап етілетін бағдарламалар құрастырылады. Бұл мақсаттар үшін компиляцияланатын тілдер қолданған жөң. Себебі, құрастырғаннан кейін бағдарлама артық ештеңе талап етпейді және артық кідіріссіз орындалатын тек машиналық командаларды қамтиды. Мұндай бағдарламалардың жұмыс схемасы мынадай:

– бағдарлама бірден орындалады, сондықтан ол өзін-өзі жеткілікті және қосымша кітапханаларды талап етпейді;

– бағдарлама өз кодынан басқа машиналық коды бар кітапханалардың шақыруларын сақтайды, сондықтан, өз командаларын орындаудан басқа, бағдарлама кітапханалардан функцияларды шақырады;

– одан басқа, бағдарлама төмен деңгейдегі тілдерде жазылған және тез әдепкі бойынша жұмыс істейтін драйверлер қабаты арқылы жұмыс істей алады.

б) Бағдарламаны атқаратын жедел жадының көлемі. Бұл талап бағдарлама қосылатын жүйелер, мобильді платформалар, микроконтроллерлер үшін әзірленген кезде пайда болады. Бұл жағдайда, неғұрлым бағдарламалық тіл аз жад жұмсайды, соғұрлым жақсы. Орындау схемасында функционалдық блоктар аз болса, компьютер жадысының аз бөлігін қамтитын болады.

в) Бағдарлама компьютерге әлде адамға бағдарланған. Бағдарлама бірінші кезекте кіммен жұмыс істейтін болады? Адам немесе компьютер бар ма? Бірінші жағдайда бағдарлама дизайн және ыңғайлылық талаптарына жауап беретін қуатты графикалық бөлігі болуы тиіс. Графикалық бөлікті әзірлеу көбінесе көп уақытты талап етеді, өйткені бұл тіл күрделілігімен ерекшеленеді. Бұл жағдайда, өте жақсы болып C++/C#, Ruby, Python тілдері болып табылады. Себебі, олар бірыңғай, жоғары деңгейлі және

бағдарламалар орындау жылдамдығы тез. Алайда, егер графикалық интерфейсі өте күрделі емес болса, яғни бағдарлама мен пайдаланушы арасындағы байланыс минималды болса, онда таңдау жылдам тілдерге ASM, C түседі.

г) Кроссплатформалылығы. Бұл дегеніміз ең аз өзгерістермен әр түрлі платформаларда, түрлі оперативті жүйелерде бағдарламаның жұмыс істеу мүмкіндігі. Бұл салада әртүрлі кітапханалармен Java, C#, Flash, C++ тілдерін бөліп көрсетуге болады. Java ең негізгі бағдарлама тілі болып табылады. Өйткені, ол JVM (Java Virtual Machine) бар кез келген платформада жұмыс істеу керек деген шартпен жасалған. Java бағдарламалары ешқандай өзгерістерді талап етпейді, мысалы, jar файл Windows, Linux және Mac оперативті жүйелерінде жұмыс істей береді. Таза C++ тіліңде кроссплатформалы бағдарламаны жазу өте қиын, себебі, код шамадан тыс көп болғандықтан, орындау жылдамдығы жоғалады. Оның бір шешімі ретінде, "барлық платформаларға бір код" принципіне қол жеткізуге мүмкіндік беретін, Qt кроссплатформлы кітапханаларды жатқызуға болады. Бірақ әрбір платформаларға бағдарламаны бөлек жинау қажет. Соңғы, интерпретацияланатын, скриптік тілдерді қосуға болады. Олардың жұмыс істеуі үшін жүйеде бұл тілдің интерпретаторы болуы қажет. Олар әзірлеу тұрғысынан өте ыңғайлы, бірақ баяулығымен көрінді.

Бағдарламалық қамтамасыз етуді әзірлеу кезінде, көзделетін мақсаттардың бірі - ақпаратты сақтаудың бірыңғай орталығын құру болып табылады. Сондай-ақ, әзірленетін жүйеге қойылатын талаптардың бірі ыңғайлы, артық ақпаратпен жүктелмеген пайдаланушы интерфейсі. Бұдан басқа, ақпараттық жүйе қолданушының рөліне байланысты ұсынылатын ақпаратты шектеуді қамтамасыз етуі, қателерге және теріс кіріс деректерге төзімді болуы тиіс [2].

Frontend және Backend әзірлеу – бұл бағдарлама жасаудың процесі, яғни клиенттік және серверлік. Клиент бөлігі – frontend әзірлеу, оған CSS беттеу, макет және шаблондар жасау, сондай-ақ визуализацияға және анимацияға жауап беретін пайдаланушы интерфейсі мен арнайы скрипттерді жатқызуға болады. Серверлік бөлікке, бағдарламаның ядросын құру, платформаны, негізгі функционалды және әкімшілік бөлікті әзірлеу жатады. Frontend әзірлеушілер сайттың сыртқы түрін әзірлеу үшін пайдаланушы интерфейсі арқылы жұмыс істейді, тестілеу арқылы пайдаланушылардың әдеттерін және дизайнды ыңғайлы етіп өзгертуді зерттейді. Backend әзірлеушілер код архитектурасымен жұмыс істейді, қауіпсіздікті және басқаруды қамтамасыз ететін қосымшаларды әзірлейді. Frontend әзірлеу кезінде, үлгі түрінде сатылатын дизайнын жасауды қоспағанда, дербес жүзеге асырылмайды. Backend әзірлеу тәуелсіз сервис ретінде дербес жүзеге асырылуы мүмкін. Frontend командасының мақсаты барлық пайдаланушылар үшін қол жетімді интерфейс құру болып табылады және ұялы және дербес компьютерлері үшін барлық көріністерге жауап береді. Backend командасы жасалынған сыртқы интерфейс үшін қолданбаларды жасау және оны қолдау үшін қажет. Сонымен

қатар, олар сайттың ашылуына және барлық функциялардың дұрыс жұмыс істеуіне қамқорлық жасайды.

Серверлік бөлімді әзірлеу үшін, қазіргі уақыттағы ең танымал үш тілдер – PHP, Ruby және Python қарастырылды. Бұл тілдердің сипаттамасы және артықшылықтары мен кемшіліктері төменде көрсетілген.

а) Ruby-динамикалық, рефлективті, интерпретацияланатын жоғары деңгейлі бағдарламалау тілі. Ол операциялық жүйеден тәуелсіз көп дәлдікті іске асыру, күшті динамикалық типтеу, қоқыс жинаушы және басқа да көптеген мүмкіндіктерге ие. Синтаксис ерекшелігі бойынша ол Perl және Eiffel тілдеріне жақын, объектілі-бағытталған тәсіл — Smalltalk. Сондай-ақ, тілдің кейбір ерекшеліктері Python, Lisp, Dylan тілдерінен алыңған.

Артықшылықтары:

– кроссплатформалы және ашық бастапқы коды бар;

– гипермәтін белгілеу тіліне кіріктірілуі мүмкін;

– жоғары деңгейдегі бағдарламалау тілі (VHLL);

– жаңа әзірлеушіге тілді тез үйренуге мүмкіндік беретін қарапайым және түсінікті синтаксис;

– DB2, MySQL, Oracle және Sybase сияқты деректер базасына оңай қосылу мүмкіндігі;

– қарапайым API арқылы көп ағынды қосымшаларды жазу мүмкіндігі.

Кемшіліктері:

– басқа тілдермен салыстырғанда процессорлық уақыттың үлкен шығыны (CPU time);

– жаңартулары басқа тілдермен салыстырғанда баяу дамуы.

б) PHP немесе Personal Home Page Tools (жеке веб-беттерді жасау құралдары) – бұл веб-әзірлеу саласында кең таралған, интерпретацияланатын бағдарламалау тілі. Қазіргі уақытта кез келген хостинг PHP бағдарламалау тілін қолдайды. PHP – сценарийлердің серверлік тілі. HTML-мәтінге салынған PHP конструкциялары әр бетке кіргенде сервермен орындалады. Оларды өңдеу нәтижесі әдеттегі HTML мәтінімен бірге браузерге жіберіледі. PHP ашық бастапқы коды бар және тегін таратылады. PHP-бұл бағдарламалау тілі, PHP қолданудың негізгі саласы веб-әзірлеу саласы болып табылады.

Артықшылықтары:

– PHP лицензиясы бойынша тегін бағдарламалық қамтамасыз;

– қол жетімді кеңейтілімдер мен көптеген бастапқы кодтарды ұсынады;

– кез келген операциялық жүйеде немесе платформада жұмыс істейді;

– Microsoft ASP (Active Server Pages) сияқты бәсекелестерге жақсы балама;

– шектеулі орындау ортасында кодты орындауға рұқсат береді.

Кемшіліктері:

– дербес қосымшаларын әзірлеу үшін қолайлы емес;

– жаһандық конфигурация параметрлері тілдің семантикасын өзгертуі мүмкін;

– жалпы басқа бағдарламалау тілдерімен салыстырғанда аз қорғалған деп саналады.

в) Python - әзірлеушінің өнімділігін және кодтың оқылуын арттыруға бағытталған жалпы мақсаттағы жоғары деңгейлі бағдарламалау тілі. Python құрылымдық, объектілі-бағытталған, функционалдық, императивті және аспектілі-бағытталған бағдарламалауды қолдайды. Негізгі архитектуралық ерекшеліктер — динамикалық түрлендіру, жадыны автоматты басқару, ерекшеліктерді өңдеу механизмі, көп ағынды есептеулерді қолдау, деректердің жоғары деңгейлі құрылымы.

Артықшылықтары:

– жылдам динамикалық семантикалық қасиеттерді қамтамасыз ету;

– қажетті функцияларды тестілеу және импорттау арқылы қосымшаларды қарапайым түрде құру;

– бағдарламалаудың объектілі-бағытталған тәсілі.

Кемшіліктері:

– көп ядролы және көп процессорлы есептеу жүйелерімен жұмыс жасауының жеткіліксіздігі;

– деректер базасына қол жеткізудің шектелген деңгейі;

– коммерциялық қолдаудың болмауы.

Frontend бағдарламалық бөлігін құру мақсатында JavaScript, HTML, CSS тілдері қарастырылды.

а) JavaScript-бұл веб-беттердің интерактивтілігі үшін әзірленген клиенттік бағдарламалау тілі. Клиент ол сервер жағында емес, браузер жағында орындалатынын білдіреді. Бұл үлкен артықшылық, себебі оған арнайы орта қажет емес, кез келген заманауи браузерде JavaScript интерпретаторы бар, сондықтан JavaScript-пен жұмыс істеу үшін тек браузер жеткілікті. JavaScript-сценарий тілі, немесе скрипттер. Скрипт бағдарламалық код — іске қосар алдында алдын ала өңдеуді (мысалы, компиляция) талап етпейтін нұсқаулық жиынтық. JavaScript коды веб-бетті жүктеу кезінде браузер қозғалысымен түсіндіріледі. JavaScript - прототипті объектілі-бағытталған тіл. Ол бірнеше кірістірілген нысандарды қолдайды, сондай-ақ өз нысандарын жасауға немесе жоюға мүмкіндік береді.

Артықшылықтары:

– қолдану жылдамдығы салыстырмалы түрде тез. JavaScript сценарийі клиент тарапынан жазылған, веб-серверді қолдау үшін қолдау қажет емес. Ол сондай-ақ клиент жағында компиляция қажет емес, ол оған белгілі бір жылдамдық артықшылықтарын береді. Сценарий пайдаланушының компьютерінде орындалғандықтан, тапсырмаларға байланысты нәтижелер бірден орындалады.

– скрипттік тілінің қарапайымдылығы. JavaScript игеру және іске асыру оңай. Ол беттердегі әр түрлі объектілер үшін көптеген алдын ала орнатылған функцияларды қамтамасыз ететін DOM моделін пайдаланады, бұл пайдаланушы мақсатын шешу үшін сценарийді әзірлеу үшін оны оңай етеді.;

– бағдарламалық тілдің әмбебаптылығы. JavaScript басқа тілдермен жақсы жұмыс істейді және түрлі қосымшаларда қолдануға болады.

Кемшіліктері:

– қауіпсіздік. JavaScript веб-беттерге және клиенттік браузерлерге анық қосылған, ол пайдаланушы жүйесін пайдалана алады, сондықтан зиянды код клиенттік машинада іске қосылуы мүмкін;

– браузерді қолдау. JavaScript кейде әртүрлі браузерлермен түсіндіріледі. Әр түрлі құрастыру механизмдері JavaScript-ті әртүрлі түрінде көрсете алады, бұл функционалдылық пен интерфейс тұрғысынан келіспеуге әкеледі. JavaScript көп бөлігі DOM браузерлер элементтерімен манипуляцияларға байланысты;

– көптеген бәсекелестер. JavaScript - бұл машиналарда жұмыс істейтін сценарийлердің тіл және оның орнына бірдей сапада (мысалы, JQuery) жақсы және жеңіл тәсілмен жасайтын басқа технологиялар.

б) HTML-бұл Интернет желісінде өте кең таралған гипермәтіндік белгілеу тілі. HTML тілі браузерде көретін бет құрылымын анықтайды. Интернет желісіндегі әрбір сайт ақпаратты көрсету үшін HTML тілін пайдаланады. HTML html тэгтердің арқасында браузерде көрген бет құрылымын анықтайды, браузер "оқиды", оларды өңдейді, содан кейін экранда сізге теруді шығарады, бірақ HTML элементтері түрінде, кейбір HTML элементтерімен сіз тіпті тышқанның немесе пернетақтаның көмегімен өзара әрекеттесе аласыз.

Артықшылықтары:

– автор таңдаған қаріптер, кодтар, стильдер, мәтін түсі туралы ақпаратты сақтайды;

– түрлі компьютерлік платформаларда оқылады;

– көптеген бағдарламалар жұмыс істей алады.

Кемшіліктері:

– веб-беттер, қазірдің өзінде жүктелген және web-шолғышпен көрсетілген, өзгерту мүмкін емес;

– графиктер бөлек сақталады;

– жиі жеке файл емес, бүкіл файл тобын білдіреді;

– қарапайым мәтіндік файлдарға қарағанда дискіде көп орын алады.

в) CSS (Cascading Style Sheets ағылшын тілінен аударғанда — стильдердің каскадтық кестелері) — олар белгілеу тілі жазылған веб-беттің сыртқы түрін сипаттау үшін пайдаланылады. CSS стильдері веб-беттерді рәсімдеу үшін пайдаланылады және олар көбінесе HTML және XHTML пішіміне қолданылады, сонымен қатар XML пішіміндегі құжаттармен де қолданылуы мүмкін.

Артықшылықтары:

– пайдаланудың қарапайымдылығы. Стильдердің каскадтық кестесі (CSS) сіздің сайтыңыздың стилін тез өзгертуге мүмкіндік береді, өйткені барлық стильдер бір-біріне жазылады .css файл (кейде мұндай файлдар

бірнеше болады), оларды бір уақытта бірнеше бет үшін қайта пайдалануға болады.;

– сайт көлемінің айтарлықтай азаюы. Осылайша, біз мазмұнды (HTML коды) визуалды безендіруден (CSS) бөліп аламыз, бұл бізге HTML бет өлшемін бірнеше рет азайтуға мүмкіндік береді. Сондай-ақ, сіз сайт жүктелгеннен кейін браузер CSS файлын кэштейді және кейін стильдер барлық басқа беттер үшін пайдаланылады, яғни оларды қайта және қайта жүктеудің қажеті жоқ.;

– стилизацияның қосымша мүмкіндіктері. Мысалы, мәнерлер көмегімен бір мәтін блогы басқа мәтінді (мысалы, бұл мәзір үшін жарамды) сүртіп, мәзір тіркелген және сайт жылжыған кезде сол жерде қалатындай етіп жасауға болады;

– кестесіз макеттің құрылымы. Әлемде CSS жоқ болғанға дейін барлық веб-шеберлер веб-беттегі элементті оңай позициялау мүмкін болатындай беттеудің кестелік әдісін қолданды, бірақ бұл әдіс беттерді жүктеуді баяулатты.

Кемшіліктері:

– кроссбраузерлік, яғни әртүрлі браузерлерде әртүрлі көрініс көрсетеді.

Соңғы қадам, деректер қорын таңдау. Деректер қоры - бұл кез келген түрдегі деректердің логикалық қоймасы. Әрбір деректер базасы, ол деректерді өңдеудің белгілі бір құрылымын анықтайды. ДҚБЖ - бұл әр түрлі формадағы, өлшемдегі және үлгідегі деректер қорын басқаратын қосымшалар немесе кітапханалар.

ДҚБЖ деректерді сақтау және жұмыс істеу үшін құрылымдарды, яғни кестелерді пайдаланады. Әрбір баған, атрибут, өзінің ақпарат түрін қамтиды. Бірегей кілтке ие деректер базасындағы әрбір жазба кесте жолына жіберіледі және оның атрибуттары кесте бағандарында бейнеленеді. Енді, ең танымал үш ДҚБЖ талқыланады. Бұларға, өте қуатты SQLite, жиі қолданылатын MySQL және ең озық және икемді PostgreSQL қарастырылады.

а) SQLite - бұл қосымша қолданатын таңқаларлық кітапхана. Файлдық деректер қоры бола отырып, ол деректердің кез келген түрлерін оңай өңдеуге арналған тамаша құралдар жиынтығын ұсынады. Қолданба SQLite пайдаланғанда, олардың байланысы қандай да бір интерфейс емес, деректерді қамтитын файлдардың функционалдық және тікелей шақырулары көмегімен жүргізіледі, бұл операциялардың жылдамдығы мен өнімділігін арттырады.

Артықшылықтары:

– файлдық. Барлық деректер қоры бір файлда сақталады, бұл орын ауыстыруларды жеңілдетеді;

– стандартталған. SQLite SQL тілің қолданады;

– әзірлеу және тіпті тестілеу үшін ыңғайлы. Әзірлеу кезеңі кезінде көпшілікке келешекте үлкейе алатын шешім қажет және осы мақсатқа SQLite, өзінің бай функциялар жиынтығымен жеткілікті функционал ұсына алады.

Кемшіліктері:

– Пайдаланушы басқаруының болмауы: озық ДБ пайдаланушыларға артықшылықтарға сәйкес кестедегі байланыстарды басқару мүмкіндігін береді, бірақ SQLite-те мұндай функция жоқ.

б) MySQL - барлық ірі серверлік деректер қоры арасында ең танымал. Оны түсіну өте оңай, және ол туралы желіде көптеген ақпарат табуға болады. MySQL және SQL стандарттарын толығымен іске асыруға тырыспаса да, ол кең функционалды ұсынады.

Артықшылықтары:

– қарапайымдылығы. MySQL ДҚ-мен жұмысты бастауды жеңілдететін, визуалды құралдарды қоса алғанда, көптеген бөгде құралдар бар;

– көптеген функцияларжиынтығ. MySQL SQL тілінің функционалының көп бөлігін қолдайды;

– қауіпсіздік. MySQL-да көптеген қауіпсіздік функциялары бар;

– қуат және масштабталу мүмкіндігі. MySQL шын мәнінде үлкен деректер көлемімен жұмыс істей алады және масштабталатын қосымшалар үшін жақсы келеді;

– жылдамдығы. Кейбір стандарттармен елемеу MySQL жұмыс істеуге мүмкіндік береді.

Кемшіліктері:

– белгілі шектеулер. Анықтау бойынша, MySQL кез келген нәрсені жасай алмайды және онда белгілі бір функционалдылық шектеулері бар;

– сенімділік мәселелері. Кейбір операциялары кезінде басқа ДҚБЖ-ға қарағанда сенімді емес.

в) PostgreSQL - бұл ең озық ДҚБЖ, ең алдымен стандарттарға толық сәйкестікке және кеңейтілуге бағытталған. PostgreSQL, немесе Postgres, ANSI/ISO SQL стандарттарына толық сәйкес келуге тырысады. PostgreSQL басқа ДҚБЖ –дан объектілі-бағытталған функционалдылығымен артықшыланады. Соның ішінде ACID тұжырымдамасының толық қолдауына ие. Қуатты Postgres технологиясына негізделген бола отырып, бірнеше тапсырмаларды бір мезгілде өңдеу өте жақсы тәжірибеленеді.

Артықшылықтары:

– толық SQL-үйлесімділік;

– PostgreSQL сақталатын процедуралар есебінен бағдарламалық кеңейтуге болады;

– PostgreSQL-тек реляциялық емес, сонымен қатар объектке-бағытталған ДҚБЖ болып табылады.

Кемшіліктері:

– өнімділік. Қарапайым оқу операцияларында PostgreSQL қарсыластарына жол бере алады.

Жоғарыда қарастырылған бағдарламалық тілдер мен қосымшаларға байланысты, бағдарламалық қамтаманы құру үшін неғұрлым ыңғайлы деп: front-end үшін JavaScript бағдарламалау тілі, back-end бөлімі үшін Ruby серверлік тілі және ДҚБЖ PostgreSQL таңдалынды.

1.3 Блокчейн технологиясының түсінігі

Бүгінгі таңда блокчейн технологиясы өміріміздің көптеген салаларына белсенді ене бастады. Бұл технологиясының маңыздылығы Интернет желісі жасалуымен теңдес, қазіргі қоғамға еркін еніп, айналамыздағы өзгертуде.

Біздің өміріміз ақша, құпия деректер мен құжаттарға тығыз байланысты. Осыған байланысты біз ақша, құжаттар мен мәліметтерді беретін, оларды тексеріп және дұрыстығын куәландыратын көшірмелердің шынайылығын тексеретін түрлі делдалдармен қарым-қатынас жасаймыз. Сонымен қатар, біз бұл делдалдарға сенім артуға мәжбүрміз, алайда банктердің клиенттердің ақшасын жасыру, нотариустардың алаяқтық келісімшарттар жасау, мемлекеттік органдардың және коммерциялық ұйымдардың қызметкерлерінің өз мүддесі үшін зиянды әрекеттер жасау жағдайлар кездеседі. Мәмілелерді, келісімшарттарды жасасу, сондай-ақ оқиғаларды тіркеу кезінде біздің құқықтарымызды бұзудан қамтамасыз ететін блокчейн технологиясы.

Blockchain - кез-келген ақпарат туралы шынайы жазбалардың сенімді сақтау технологиясы. Мысалы, blockchain-де ақша аударымдарын сақтауға болады және криптовалютада blockchain кімге және қанша виртуалды ақша аударылғаны туралы ақпаратты жазуға пайдаланылады. Жалпы алғанда қағазға жазуға болатын барлық нәрселерді жазу сақтау мүмкіндіктері бар, тек бір айырмашылығы блокчейндағы жазбаларды ауыстыру немесе жалған жазу мүмкін емес [3].

Blockchain - әр деректер блогы алдыңғы бөлігімен байланысы бар блоктарының тізбегі. Блок жазбалардың жиынтығынан тұрады. Жаңа блоктар үнемі тізбектің соңына қосылады. Блокчейн технологиясы өте маңызды 3 принциптерге негізделген өте күрделі тізбек болып табылады:

– таратылу;
– ашықтық;
– қауіпсіздік.

Blockchain-нің барлық пайдаланушылары компьютерлер желісін қалыптастырады, олардың әрқайсысында деректер блогының көшірмесі болады [2].

Соның салдарынан блокчейнді өшіру немесе бұзу мүмкін емес, себебі бұл үшін барлық компьютерлердегі ақпаратты өшіру немесе өзгерту қажет. Кем дегенде бір пайдаланушы бар болса, онда блокчейн бар. Әрбір жаңа пайдаланушы бұл желіні кеңейтеді және нығайтады. Барлық компьютерлердің, яғни пайдаланушылардың құқығы бірдей, ол жерде ұйымдастырушылар, модераторлар, бақылаушылар және басқарушылар жоқ. Әркім өзіне жауап береді.

Блокчейндегі деректер мазмұны әрқашан ашық болады. Кез келген блокты оңай оқып, осы блоктағы барлық жазбаларды көре аласыз. Пайдаланушылар тізбекті көре алады және ақпаратты өзгертуді қадағалай алады. Осылайша, blockchain ішіндегі барлық деректер оңай тексерілуі мүмкін.

Blockchain сенімділігі мен қауіпсіздігін деректердің жарамдылығы мен дұрыстығын оңай тексеруге болатын криптографиялық кілттермен жүргізіледі. Шын мәнінде, кілт - тек сан болып табылады, бірақ өте үлкен сан. Мысалы, мынадай: 1173161954235709850086879078532699846656405640394575840079 13129639935.

Ол хэш функциясы деп аталатын арнайы алгоритм арқылы есептеледі. Дәл осы деректер жиынтығы үшін хэш функциясы бір кілт береді және оның екі өте маңызды қасиеттері бар:

– кілтпен сіз бастапқы деректер жиынтығын біле алмайсыз;

– бірдей кілт беретін басқа деректер жиынтығын табу мүмкін емес. Басқаша айтқанда, тек қана кілті бар, пайдаланушы ештеңе білмейді және ешқандай зиян келтіре алмайды. [5]

Атап өту керек кілттердің тағы бір маңызды ерекшелігі бар. Бастапқы деректерде ең аз өзгеріс болса да, 1.1-кестеде кілт толығымен өзгеретіндігін көре аламыз.

1.1-кесте – Бастапқы деректер және кілт

Бастапқы деректер	Кілт
кез-келген деректермен қарапайым жол.	1672738941421626349
кез-келген деректермен қарапайым жол!	7452308507701029857
кез-келген деректермен қарапайым жол?	3891821463731365823
кез-келген деректермен қарапайым жол	-2462786558714469272

Көріп отырғаныңыздай, бастапқы деректердегі бір ғана таңбаны өзгерту кілттің толық өзгерісін береді.

Барлық деректер blockchain желісінің пайдаланушыларының компьютерлерінде сақталады. Желінің барлық пайдаланушылары бірдей және жалпы айтқанда, олар кез келген нәрсені жасай алады, яғни басқа пайдаланушыларды алдауға тырысуы да мүмкін. Оларға ешкім тыйым сала алмайды, өйткені бәрі бірдей жағдайда, тең құқықтарға ие және өз міндеттерін бірдей орындап, тіпті бұза алады. Осылайша, blockchain желісінің пайдаланушысы делдалдарға, мысалы, банктерге, мемлекеттік органдарға, аудиторларға, бақылаушыларға, сақтандыру компанияларына немесе тіркеушілерге мұқтаж емес. Рұқсат сұрайтын ешкім жоқ болғандықтан, ешкімге ыңғайлы қосымша құқықтар мен мүмкіндіктер бермейді.

Blockchain желісіне кіру арқылы пайдаланушы деректермен алмасу үшін желідегі басқа компьютерлерге блоктар мен жазбалар арқылы қосылады. Бұл желі еш географиялық орынға байланысты емес, яғни Алматыдағы

пайдаланушы бір уақытта Мәскеу, Лондон, Нью-Йорк және Буэнос-Айрес қалаларындағы пайдаланушыларға қосыла алады.

Жаңа деректерді алғаннан кейін, әр пайдаланушы олардың дұрыстығын тексереді және түпнұсқалығына сенімді бола отырып, оларды үйде сақтайды, әрі қарай желі арқылы тасымалдайды. Мәселен, желісінде деректердің екі түрі болуы мүмкін – жалған және нақты. Жақсы қатысушылардың әрқайсысы жалған мәліметтерді тапқаннан кейін оларды одан әрі аудармайды. Нәтижесінде бұл деректер тек зиянкестердің арасында ғана жүреді, ал жақсы қатысушылар тек дұрыс деректермен алмасады.

Желідегі қатысушылар екі топқа бөлінеді: жаңа жазбаларды жасайтын қарапайым пайдаланушылар және блоктарды жасайтын майнерлер. Яғни, блокты құру өте үлкен ресурстарға негізделген және күрделі үдеріс болып табылады және сондықтан да бәрінің де оларды құруы мүмкін емес [6].

Жай пайдаланушылар желідегі жазбаларды жасайды және таратады, мысалы, «кілті бар В пайдаланушыға 300000 ақша аударылады» немесе «кілт Х-мен автомобильді қарызға алған адам». Көріп отырғаныңыздай, барлық жазбалар ашық, бірақ шифрланған. Автокөліктің кілтін білсеңіз, ол кепілдікке салынғанын біле аласыз, бірақ олар нақты түрде деректерді бермесе оның иесінің атын немесе ипотекалық банктің атауын білмейсіз. Бір адамның бірнеше кілті болуы мүмкін, сондықтан да автокөлік иесінің кілтін білсеңіз де, мейлі ол сол адам, сіз ол адамның ипотекалық несиенің бар екендігін, не жоқ екендігін біле алмайсыз. Әрі қарай, майнерлер жазбаларды жинайды, тексереді және блокта жазады, содан кейін осы блоктарды желі арқылы жібереді. Осыдан кейін, қарапайым пайдаланушылар блоктарды алады және оларды өз орындарында сақтап, басқа адамдарға жаңа жазбаларды дұрыс және сенімді түрде сақтайды. Жаңа жазба блокта болмаса, ол сенімді деп саналмайды. Желінің кез-келген мүшесі оны өзінің жеке сенімділігімен пайдалана алады. Сондықтан, әдетте, қатысушылар жаңа жазбаларды жібереді, ол майнерге жеткеннен кейін блокқа жазылады. Жазба блокта сақталған кезде ғана оның тексерілгеніне және дұрыс екеніне сенімді бола аласыз және оны жоюға болмайды. Блокчейн осылай жұмыс істейді.

Blockchain блоктары блок тақырыбынан және блок денесінен тұрады, ал блоктың корпусы жазбалардың тізімі ғана. Blockchain ішіндегі блоктарды кілттермен байланыстырады, себебі алдыңғы блоктың кілті әрбір блоктың тақырыбында сақталады. Бұл өте маңызды және бір мезгілде блоктық қауіпсіздікті және бұзылмауды қамтамасыз ететін техникалық шешім.

Біріншіден, әрбір блоктың кілті бүкіл блоктар мен алдыңғы блоктың кілтінің деректерінен есептеледі. Бұл дегеніміз, блок-дискілерде бұл блоктың ғана емес барлық алдыңғы блоктардың жазбалары кодталған. Кез-келген блоктағы тіпті маңызды емес деректердің өзгеруі оның кілтінің толық өзгеруіне әкеледі, бұл, өз кезегінде, барлық кейінгі блоктардың кілттерін өзгертуді талап етеді. Осылайша, барлық блокчейн мен кілттерді көре отырып, кез-келген деректердің дұрыстығын оңай тексере аласыз, атап айтқанда, реті дұрыс емес блоктың жоқтығына көз жеткізу үшін немесе тізбектің ортасына

жаңа блок салынса, блоктың кілті өзгертіледі. Жай сөзбен айтқанда, қиындықсыз және алданып қалмағаныңызға көз жеткізуге болады.

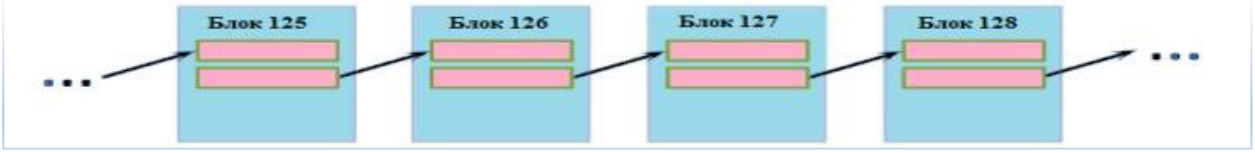
Екіншіден, блоктың кілті желі қауіпсіздігінің деңгейін белгілейтін маңызды қауіпсіздік ережелерін қанағаттандыруы керек, ол желінің өсуімен өзгеруі мүмкін. Мысалы, бірінші блоктардың кілттері он нөлден басталды – бұл жаңа блокты құру күрделілігін анықтайды.

Бірақ, деректерді тексеруден және таратудан басқа, ол жаңа блоктарды құру бойынша жұмыс істейді. Басқа желі мүшелерінен жаңа жазбаларды алғаннан кейін, майнер оларды біріктіріп, болашақ блоктың тақырыбын құрады және блоктың кілтін есептейді.

Алғашқы есептеулерден кейін, кілт осындай болды. «31173161954235709850086879078 5326998466564058918214637313658 2». Алайда, ережелерге сәйкес, кілт он нөлден басталуы керек. Кілтті өзгерту үшін бастапқы деректерді өзгертуіңіз керек. Мұны істеу үшін блоктың тақырыбында «nonce» деп аталатын арнайы мән беріледі. Алғашқы есептеуде ол 0 болып табылады. Сондықтан майнер мәнді 1-ге өзгертеді және қайтадан кілтін есептейді. Енді ол толық өзгерді «68790785326982311731610985008 1469846656405891373136582 9542357». Яғни тағы да 0 ден басталмады. Майнер «nonce» өрісін 2-ге өзгертеді де, кілтті қайта есептейді.

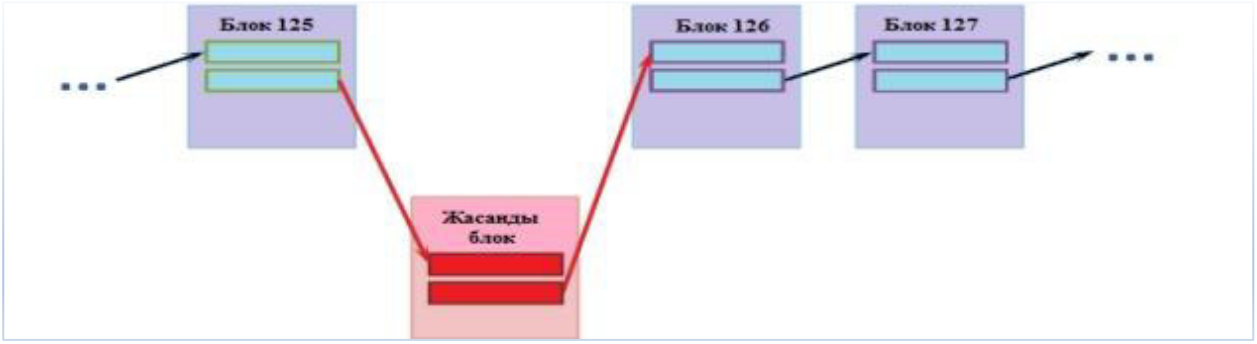
Майнерлер сәйкес кілт табу үшін миллиард және триллион қайта есептеулер жүргізеді. Сәйкес кілт табылған кезде майнер блокты сақтап, оны желідегі басқа қатысушыларға жібереді. Блок кілті бұзуға болмайтын кілтпен кодталған. Майнинг кілтінің өндірілуі бұл үдерістің прогрессінің жоқтығына бекітіледі. Бұл дегеніміз, алдын ала есептеу жүргізе алмайсыз, кілттерді сатып ала алмайсыз, кілттер қоймасын жасай алмайсыз – мұның бәрі мағынасыз, барлық қатысушылар тең. Әрбір майнерде сәйкесті кілтті алудың бір мүмкіндігі бар – бұл есептеу. Майнерлер осылайша жұмыс жасайды. Жасалған әрбір блок үшін олар ақы алады. Кілтті алғаш кім тапты, блокты жасап, жұмыс жасағаны үшін ақы алады. Содан кейін олар барлық жаңа блокты іздейді. Мәселен, күрделі есептеу әдісі, әрине, блокты құруды қиындатады, бірақ ол жалған блоктарды құруды мүмкін емес етеді.

Блокта тізбекті немесе өзгерістер жазбаларын жасау үшін жұмыс істемейді, өйткені кілттер өзгереді – бұл бірден анық болады. Егер шабуылдаушы жалған блок жасауға тырысса, не болады? Нақты мысал ретінде 1.2-суретте көрсетілген блоктардың тізбегін қарастыруға болады.



1.2-сурет – Блоктар тізбегі

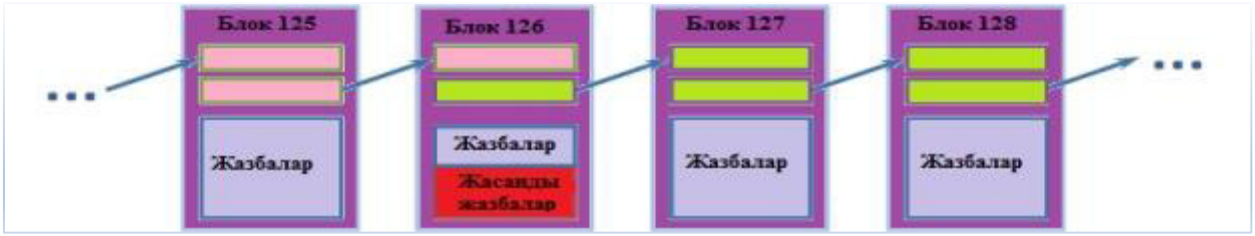
Зиянды мақсаттарына сүйене отырып, шабуылдаушы өзінің жасанды блоктарын екі бұрыннан бар блоктар арасында 1.3-суреттегідей қажетті жазбалармен кіргізіп, сол арқылы өзгерістерді қалыптастыруға шешім қабылдады делік.



1.3-сурет – Жасанды блок енгізу сызбасы

Оның блогының тақырыбында ол алдыңғы блоктың кілтін енгізеді және барлық қалған қатысушыларға жаңа блок туралы ақпаратты жібереді. Дегенмен, олар дереу жасандылықты анықтайды, себебі 126 блоктың кілті (және барлық кейінгі блоктар) да жасанды блоктың кілтін ескермейді Алдаудың жалғыз мүмкіндігі зиянкес кейінгі барлық блоктардың қайталануын жасау. Бірақ, бұл мүмкін емес, өйткені адал блоктардың жаңа блоктарын жасау жылдамдығы зиянкестердің жалған блоктарын жасау жылдамдығымен бірдей.

«Зиянкес» ешқашан «жақсы» болмайды. Тіпті, «51% шабуыл» деп аталатын мүмкіндікті қатаң теориялық тұрғыда мойындау керек («жақсы» адамдарға қарағанда «жаман» адамдар көп болғанда). Бірақ іс жүзінде оны жүзеге асыру мүмкін емес. 1.4-суретте көрсетілгендей жалған ақпараты бар блокты енгізген жағдайда кілттер байланысы жұмыс істемейді.



1.4-сурет – Жасанды жазба енгізу сызбасы

Жазбалар орналасқан блоктың корпусын өзгерткенде, бүкіл блоктың кілті өзгереді. Бірақ келесі блокта дұрыс кілт мәні сақталды. Сондықтан, кез-келген қатысушы оған жіберілген блок шынымен нақты емес, жалған болып табылатынын оңай тексере алады.

Енді жазбаларды жасау немесе blockchain жасанды блокты енгізу мүмкін емес екеніне көзіміз жетті. Бірақ барлық пайдаланушылар бірдей құқықтарға ие болғандықтан, пайдаланушыға «В пайдаланушыдан А пайдаланушы шотына ақша аударылсын» деген жазба енгізуге не кедергі келтіреді? Жазбалар қорғанысы блок қорғанысынан да күрделі ілінісу тізбегі арқылы қорғалған. Әрбір жазба 1.5-суреттегідей алдыңғы жазба сілтемесімен қатар, ашу ережесін қамтиды.

1.5-сурет – Жазба құрылымы

Жазба көзі - алдыңғы жазбаның нәтижесіне сілтеме. Ал нәтиже - келесі жазбаның болашақ көзі. Операцияның мазмұны формализацияланған түрде, осы операцияның мәні, мысалы, «100 биткоин беру» немесе «1000 теңге мөлшерінде несие беру» сипатталады. Тек құлыптан босату ережесін білетін адамдар жазбалар тізбегін кеңейтуге және сол арқылы блокчейндегі ақпаратты өзгертуге құқылы. 1.2-кестеде блокчейн технологиясына байланысты терминтерге түсінік берілген.

1.2-кесте – Анықтамалар, терминдер және қысқаша сөздер

Терминдер және қысқаша сөздер	Анықтама
Блокчейн	Әрбір келесі блок криптографиялық түрде байланысқан қаржы транзакцияларының блоктарының тізбегі.
Орталықсыздандырылған файлдық қойма (DFS)	Мәмілемен байланысты файлдарды сақтау және алмасу ортасы
Цифрлық қолтаңба (ЭЦҚ — электрондық цифрлық қолтаңба)	Шифрлау алгоритмімен ашық кілтті генерациялау коды, ол транзакцияға қоса беріледі және мазмұнның өзгермегендігін және жіберушіні сәйкестендіруді тексеру үшін қажет.

Қарапайым мысалда бұл керемет механизмді қарастырайық. Алдыңғы жазбада «Пайдаланушы құпия сөзі кілті 5008687907853269984665640589 1...» деген блокчейн шартымен пайдаланушыға ақша аударылады. Бұл ақшаны жұмсау үшін пайдаланушы А өзінің парольді бұғаттау ережесінде бір құпия сөзді көрсетіп, жаңа жазба жасайды және ереже келесідей көрінеді: «ABCDEF паролін кілтін есептеңіз».

Ережелер мен шарттарды сипаттау үшін қатысушылардың өзара әрекеттестігі үшін өте күрделі логика мен ережелерді орнатуға мүмкіндік беретін жеткілікті қуатты бағдарламалау тілі пайдаланылады. Бұған қоса, әрбір жазбада бірнеше дереккөздер мен нәтижелер болуы мүмкін, яғни бірнеше жазу жазбаларына жазбалар түрлендіре алады.

Дипломдық жобаны әзірлеу келесі кезеңдерден тұрады:

– қолданыстағы сауалнама жүйелерін талдау;

– бағдарламалық қамтаманы жобалау. Бұл негізгі кезең, оған немқұрайлы қарау сайттың алға жылжуы мен қолдауы кезіндегі қиындықтарға, сондай-ақ сайттың кейіннен толық қайта өңдеу қажеттілігіне әкелуі мүмкін;

– сұрақтарды дайындау. “Мұғалім студент көзімен” сауалнамасындағы сұрақтарды жипап, жаңа сауалнама құру;

– код жазу. Бағдарламалау, функционалдық бөлім жазу;

– тестілеу. Тестілеу мақсаты - әзірленген шешімнің техникалық тапсырмаға сәйкестігін тексеру және жүйедегі барлық тетіктерді ретке келтіру. Жүйе беттерінің қолжетімділігін және оның функционалының жұмыс қабілеттілігін тексеру жүргізіледі. Сайттың бағдарламалық кодындағы қателіктер оны іздеу арқылы индекстеудің қиындауына, веб-беттерді жүктеудің баяулауына және соның салдарынан - сайтқа кіру және конверсияның төмен болуына әкеп соқтыруы мүмкін.

1.4 Backend бөлімінің құралдарына талдау

«Мұғалім студент көзімен» атты сауалнама жүргізетін бағдарлама қамтамасын құру үшін, жоғарыда талдаудың нәтижесі бойынша backend үшін серверлік Ruby тілі, ал frontend бөлімі үшін JavaScript тілі таңдалды.

Ruby - тез және ыңғайлы объектілі-бағытталған бағдарламалау үшін жоғары деңгейдегі интерпретацияланатын тіл. Тіл операциялық жүйеден тәуелсіз, көп дәлдікті жүзеге асыру, қатаң динамикалық типтеу, "қоқыс жинаушы" және басқа да көптеген мүмкіндіктерге ие. Ruby синтаксис ерекшеліктері бойынша Перл және Эйфель тілдеріне, Smalltalk объектілі-бағытталған тәсілге жақын. Сондай-ақ, тілдің кейбір ерекшеліктері Python, Түлкі, Dylan және CLU алынған.

Синтаксистің қарапайымдылығы программистерге тестілеу (TDD) арқылы өнімдерді әзірлеуді жүргізуге, MVC іске асыруға және ORM арқасында деректер базасымен жұмыс істеуге мүмкіндік береді. Қазіргі заманғы Ruby бағдарламашысы Design Patterns саласындағы жобалардың үлгілерін және SaaS үшін бағдарламаларды жасауға мүмкіндік алады.

JavaScript - framework пайдалану AJAX байланысты жұмыс процесін жеңілдетеді.

Ruby-да Redmine багтрекинг жүйесі, Inkscape векторлық графикасына арналған редактор жазылған. Сонымен қатар, Metasploit үшінші нұсқасы (ақпараттық қауіпсіздікті зерттеу жобасы) толығымен Ruby-ке жазылды. Ruby NASA, Motorola сияқты көптеген ірі ұйымдарда қолданылады.

Ең танымал болып Ruby 2004 жылы шығарылған Ruby on Rails арқасында болды. Rails Twitter, GitHub, Diaspora және т.б. сияқты алыптарды құру кезінде қолданылды. Мысалы, деректер қоры үшін SQL сұраныстарын пайдаланбай деректерді алуға мүмкіндік беретін Active Record сұраныстарының интерфейсін пайдалануға болады [7].

Қазір Ruby үшін, Ruby on Ralis үшін стартаптар үшін тіл даңқы бекітілді. Rails-да түпнұсқалығы мен тиімділігімен ерекшеленетін және шаблонды лендингтер мен интернет-дүкендердің шеңберінен шығатын жобаларды қысқа мерзімде жүзеге асыру алынады. Әрине, сайтқа жүктемелердің өсуіне және талаптардың өзгеруіне байланысты кейбір жобалар белгілі бір ерекшелікті ескеретін және кейбір жағдайларға (мысалы, Java немесе Scala) сәйкес келетін басқа тілдерге жазылады. Алайда, Ruby және Rails дамуын жалғастырады, сондықтан жақын арада ешқандай ауысу қажет емес.

Ruby тілінің ерекшеліктері:

– Ruby ашық бастапқы коды бар және еркін қол жетімді тіл;

– Ruby-әмбебап, түсіндірілетін бағдарламалау тілі;

– Ruby-шынайы объектілі-бағытталған бағдарламалау тілі;

– Ruby - Python және PERL сияқты серверлік скрипт тілі;

– Ruby Common Gateway Interface (CGI) сценарийлерін жазу үшін пайдалануға болады;

– Ruby гипермәтіндік таңба (HTML) тіліне енгізілуі мүмкін;

– Ruby жаңа әзірлеушіге тез және оңай оқуға мүмкіндік беретін таза және қарапайым синтаксис бар;

– Ruby C ++ және Perl сияқты көптеген бағдарламалау тілдеріне ұқсас синтаксисі бар тіл;

– Ruby өте масштабты және Ruby жазылған үлкен бағдарламаларға оңай қолдау жүргізе алынады;

– Ruby Интернет және интернет қолданбаларын жасау үшін пайдалануға болады;

– Ruby Windows және POSIX ортасында орнатуға болады;

– Ruby TCL / Tk, GTK және OpenGL сияқты көптеген GUI құралдарын қолдайды;

– Ruby DB2, MySQL, Oracle және Sybase оңай қосуға болады;

– Ruby Ruby скриптерінде тікелей пайдалануға болатын кіріктірілген функциялардың бай жиынтығы бар.

Енді, Ruby-ды басқа серверлік тілдермен салыстыратын болсақ

– Ruby және Java;

– Ruby түсіндірілетін тіл;

– Ruby-да барлығы объект (Java-да int және Integer түрлері бар, бұл белгілі бір қолайсыздық тудырады);

– Ruby-тегі айнымалылар статикалық типтелген емес және хабарландыруды талап етпейді;

– Ruby модулдері (modules) <<mixins>> көмегімен Java тілінің интерфейстерін (interfaces) құрастыруға мүмкіндік береді.

Ruby және Perl:

– Ruby игеру әлдеқайда оңай, бағдарламалау оңай, ал жазылған бағдарламаларды сүйемелдеу оңай;

– Ruby префикстерінде (@ ,$,@@) айнымалы түрі емес, көріну аймағын (scope) сипаттайды;

– Ruby Perl тілінен тұрақты өрнектер, айнымалы $_ және тағы басқалар.

Ruby және Python:

– Ruby тіліндегі басқару конструкциялары мен әдістері end негізгі сөзімен аяқталады, ал Python бағдарламаның кезекті жолында жетекші олқылықтардың санын өзгерту аяқтау белгісі болып табылған кезде < < екі өлшемді> синтаксис деп аталады.;

– Ruby-да, Python тіліне қарағанда, түр мен сынып ұғымдары синонимдер болып табылады;

– Python мұраны қолдамайды және әдістерді бар түрлерге қосуға мүмкіндік бермейді;

– Ruby-да қолданылатын қоқыс жинау алгоритмі C тілінде әдістердің жүзеге асырылуын оңай жазуға мүмкіндік береді;

– C/C++ жазылған Ruby кеңейтімдері жаңа сыныптарды анықтауға мүмкіндік береді;

– жиі Ruby Python қарағанда тезірек.

– Ruby – түсіндірілетін тіл;

– Ruby-да барлығы объект (Java-да int және Integer түрлері бар, бұл белгілі бір қолайсыздық тудырады) [8].

Егер Ruby-ді басқа тілдермен салыстырсақ, онда бір міндеттерді шешу үшін код өлшемі айтарлықтай аз екенін көруге болады, бірақ бұл оның бірдей функционалдық тіл етіп жұмыс істеуіне кедергі келтірмейді. Бірнеше танымал тілдердің түрлі мүмкіндіктерін салыстыру үшін 1.3-кестеде көрсетілгендей белгілерді енгіп, 1.4-кестесінде келтіреміз салыстырмалы сипаттама жасаймыз.

1.3-кесте – салыстырмалы сипаттама үшін пайдаланылатын шартты белгілер

Белгі	Түсінігі
+	Мүмкіндік бар
-	Мүмкіндігі жоқ
+ / -	Толығымен қолдау мүмкіндігі жоқ
- /+	Мүмкіндігі өте шектеулі қолдау
?	Деректер жоқ

1.4-кесте – бағдарламалау тілдерінің функционалдық мүмкіндіктерінің салыстырмалы сипаттамасы

Мүмкіндігі	Ada	C	C++	C#	Prolog	Java	Perl	php	Delphi	Ruby
Рефлексивті	-	-	-	-/+	+	-/+	+/-	+	-/+	+
Декларативті	-	-	-	-/+	+	-	-/+	+	-	+
Бөлінген	+	+/-	+/-	-/+	+	+	-	-	-	-/+
Жадты қолмен басқару	+	+	+	+	?	-	-	-	+	-
Қоқыстарды жинау	-/+	-	-	+	?	+	+	+	-	+
Кортеждер	-	-	+/-	+/-	?	-	+	-	-	+
Еркін ұзындықтағы бүтін сандар	-	-	-	+	?	+	+	?	-	+
Динамикалық типтеу	-	-	-	+/-	-	-	-/+	+	-/+	+

Кестеден Ruby тілі Php мүмкіндіктеріне ұқсас екенін көреміз. Ruby, C++, Ada-дан айырмашылығы деректер ретінде өз кодымен жұмыс істей алады. Қарастырылып отырған тіл декларативті болып табылады, өйткені тапсырманың шешімін ғана емес, шешімді компьютерді анықтау керек. Тілдің қарама-қарсы түрі – бұл императивті.

Ruby бағдарламасын параллельдеуге мүмкіндігі шектеулі, бірақ ол Php, Perl, Delphi тілдерімен салыстырғанда бар, сондықтан олардан артықшылық береді. Кортеж – бұл ерікті түрдегі бірнеше атаусыз өрістер бар, аталмаған деректер түрі. Ruby әдісті (функцияны) қайтару мүмкіндігі бар.

Ruby тілінің мүмкіндіктері:

– Ada, Eiffel және Python тілдерінің әсерімен әзірленген қысқа және қарапайым синтаксис;

– ерекшеліктер өңдеуге мүмкіндік беру;

– шын мәнінде әдіс болып табылатын операторларды қайта анықтауға мүмкіндік беру;

– толық объектілі-бағытталған бағдарламалау тілі;

– көптеген тұқым қуалауды (multiple inheritance) қолдамайды, бірақ оның орнына модульдердің механизміне негізделген "қоспалар" концепциясы қолданылады;

– автоматты қоқыс жинаушы бар. Ол барлық Ruby нысандары үшін, соның ішінде сыртқы кітапханалар үшін жұмыс істейді;

– C тілінде Ruby үшін кеңейтулерді жасау өте қарапайым, өйткені қоқыс жинаушы күрделі емес және ыңғайлы API бар;

– айнымалы тоққа толық байланған тұйықталуларды қолдайды.

– айнымалыларды алдын ала хабарландыруды талап етпейді;

– Ruby-да тікелей тілде көптеген жобалау үлгілері іске асырылған, мысалы, "singleton" (жалғыз) бір нақты объектіге қажетті әдістерді қосу арқылы жүзеге асырылуы мүмкіндігі;

– операциялық жүйеден тәуелсіз көп дәлдікті қолдауы бар;

– көптеген платформаларға ауыстырыла алады. Ол Linux-да жасалған, бірақ Unix, DOS, Microsoft Windows (атап айтқанда Win32), Mac OS, BeOS, OS/2 және т. б. көптеген нұсқаларында жұмыс істейді [9].

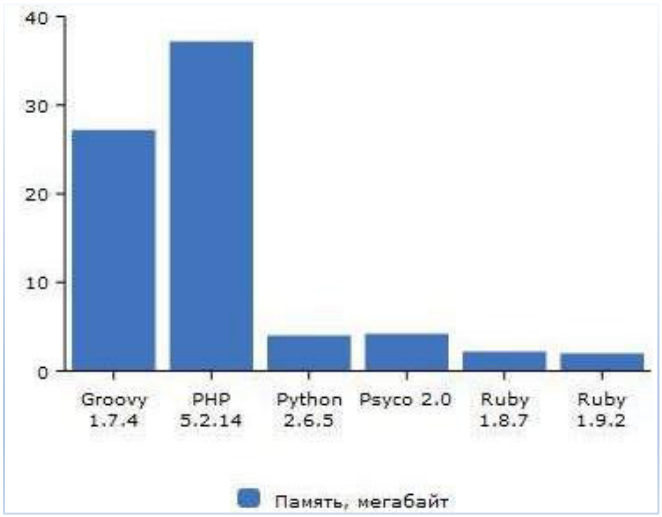
Ruby өз қоқыс жинаушысы болғандықтан, жадымен жұмыс басқа тілдерге қарағанда жоғары жылдамдықта болады.

Интерпретаторлардың көпшілігі бірден белгілі бір жад көлемін алды, және жұмыс барысында бұл көрсеткіш өзгерге алмайды. Жады бойынша PHP сұраулары желілік түрде өсті, ал Ruby толқын тәрізді қолданды да босатты. 1.5-кестеге жүргізілген тестілердің ең жоғарғы мәндері енгізілді және 1.6-суретте көрсетілді.

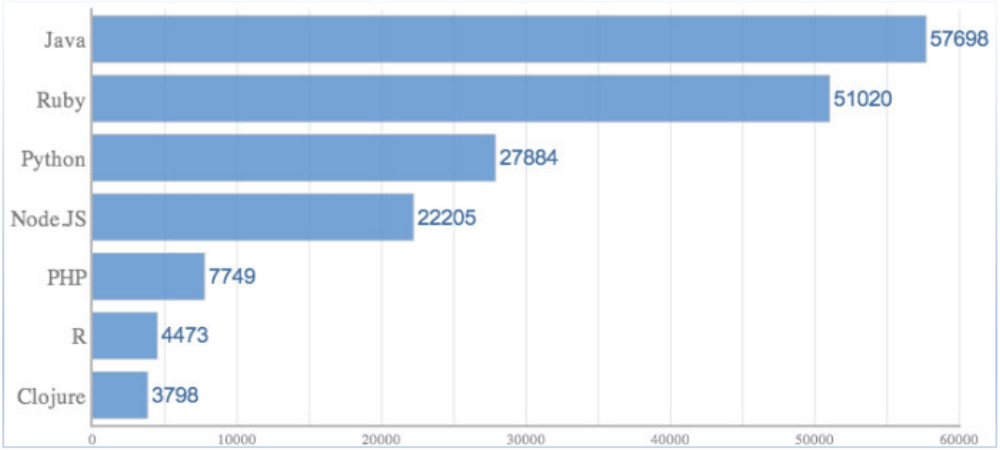
Бүкіл әлемдік статистикаға жүгінетін болсақ ең ірі ашық кодты, яғни open source, дайын жобалар туралы ақпарат жинайтын порталдың статистикасына жүгінетін болсақ, Ruby тілі үштікке кіріп тұр. Бұл барлық интернет желісі бойынша жиналған ақпаратты құрайды. 1.7-суретте бұл мағлұмат көрсетілген.

1.5-кесте – тесттерді өту кезінде жадты жүктеу

Тіл	Жады, мб
Groovy 1.7.4	27
PHP 5.2.14	37
Python 2.6.5	3.8
Python 2.6.5 + Psyco 2.0	4
Ruby 1.8.7	2
Ruby 1.9.2	1,8



1.6-сурет – Тестілеу кезінде жадымен жұмыс істеудің салыстырмалы диаграммасы



1.7-сурет – Әлем бойынша 2018 жылғы бағдарламалық тілдердің қолданысына орай рейтингі

1.5 Frontend бөлімінің құралдарына талдау

Ruby тілің бағдарламаның backend бөлімін құруға қолдансақ, frontend бөлімі JavaScript арқылы істелінген болатын. Енді, JavaScript скрипттік тіліне тоқталып болсақ, JavaScript – бұл бағдарламалау тілі интерактивті HTML-құжаттарды жасау үшін Netscape фирмасымен әзірленген. Бұл-клиент жағында да, сервер жағында да орындайтын кірістірілген қосымшаларды әзірлеудің объектілі-бағытталған тілі. Тілдің синтаксисі Java тілінің синтаксисіне өте ұқсас – сондықтан оны жиі Java ұқсас деп атайды. Клиенттік қосымшалар пайдаланушының машинасында веб-құжаттарды қарау браузерімен орындалады, серверлік қосымшалар серверде орындалады.

JavaScript тілі – бұл web-бағдарламалаудың клиенттік тілі, ол 1995 жылы Брендан Айк құрған. JavaScript әдетте әр түрлі қосымшаларда объектілерді манипуляциялау үшін қолданылады, бірақ ең үлкен танымалдығы сайттарды жасау кезінде қолданылатын негізгі тілдердің бірі ретінде және web-бағдарламалаудың жалғыз клиенттік тілі ретінде сатып алынды.

JavaScript тілінің коды әдетте браузер терезесінде, сайттың ашық бетінде орындалады. Бұл веб-браузерде әдепкі бойынша JavaScript тілінің интерператоры бар, оның арқасында браузер JavaScript тілінде жазылған кодты түсіну және орындау мүмкіндігі бар [10].

JavaScript тілінің интерпретаторы веб-браузердің бір бөлігі болып табылады, веб-браузер сайт бетін ашқан кезде, браузер құжаттардың Объектілік моделін (DOM) жасайды, JavaScript интерпретаторы DOM қатынауға қол жеткізеді және осының арқасында вебмастер әр түрлі скрипттерді (web-қосымшалар) жасай отырып, бет объектілерін (барлық тегтерді: абзацтарды, тақырыптарды, кестелерді, формаларды және т.б.) басқара алады және айла алады.

JavaScript ресми атауы ECMAScript болса да, ол өте тез JavaScript деп аталды. ECMAScript халықаралық ECMA ұйымы (Еуропалық компьютерлер өндірушілер қауымдастығы) әзірлейді және қолдайды.

JavaScript HTML және CSS тілдері сияқты барлық адамзатқа тиесілі, оларға бірде-бір компания немесе тұлға ие емес. Алайда, "JavaScript" сөзі Oracle Corporation компаниясына тиесілі және авторлық құқықтармен проблемалар болмау үшін, осы тілді әзірлеуші ғалымдар ғылыми құжаттарда оны ECMAscript деп атайды [11].

Интерактивті HTML беттерін жасау кезінде JavaScript тілін қолданудың негізгі салалары:

– сценарий арқылы динамикалық құжатты жасау;

– сервер жібергенге дейін пайдаланушы толтыратын HTML нысандарының өрістерінің дұрыстығын жедел тексеру;

– динамикалық HTML-беттерді каскадтық стиль кестелерімен және құжаттың объектілік үлгісімен бірге жасау;

– HTML-бетке енгізілген JavaScript бағдарламасы шешетін "жергілікті" тапсырмаларды шешу кезінде пайдаланушымен өзара әрекеттесу.

JavaScript тілінің сипаттамалары:

а) сценарийлік - бағдарламаларды жасау кезінде, оларды компиляциялау қажет, тек мәтіндік редакторда скрипт-бағдарлама, содан кейін JavaScript интерпретаторы бұл скрипт өңдейді және орындайды;

б) клиенттік - скрипттердің интерпретаторы web-серверде емес, веб-браузерде орындалады, сондықтан JavaScript сценарийлерін жасау және орындау үшін веб-серверді орнатудың қажеті жоқ, тек мәтіндік редактор (мысалы, Блокнот немесе қазір танымал Sublime Text 3) және веб-браузер (мысалы FireFox, Opera, IE немесе Chrome және т.б.) қажет;

в) прототипті бағытталған - JavaScript объектілерді құру мен тұқым қуалауды қолдайды, алайда онда сыныптар, яғни класстар жоқ;

г) динамикалық типизация - Си, Си++ немесе Паскаль сияқты басқа бағдарламалау тілдеріне қарағанда, бағдарламаны орындау кезінде JavaScript айнымалылары өзінің деректер түрін өзгертуге мүмкіндік береді. Бұл JavaScript-те деректер түрі айнымалыны жариялау кезінде емес, оның инициализациясы кезінде беріледі;

д) функциялар - JavaScript тілінде, функциялар бірінші кластағы объектілер болып табылады, бұл мәннің орнына айнымалыны білдіреді, сондай-ақ функцияларды да беруге болады;

е) жадты автоматты түрде тазалау - бағдарламаларды орындау кезінде, JavaScript интерпретаторы жадты пайдаланылмайтын элементтерден өз бетінше тазалайды [12].

JavaScript – бұл сіздің веб-беттеріңізге интерактивтілік пен қайырымдылықты қосатын кросс-платформалық, объектілі-бағытталған скриптік тіл.

JavaScript веб-сайтты әзірлеушіге веб-бетті қалай жүргізу керектігін басқаруға мүмкіндік береді. Бұл JavaScript HTML-ден мүлдем өзгеше, веб-

құжаттың құрылымына жауап беретін тіл, және CSS, веб-беттердің сыртқы түрін қалыптастыратын тіл. JavaScript тілінде жазылған бағдарламалар скриптер деп аталады. Браузерде олар HTML-құжатқа тікелей қосылады және бет жүктелгеннен кейін-бірден орындалады. Скриптті орындау процесі "интерпретация"деп аталады.

JavaScript браузерінде HTML-құжатымен манипуляциялауға, келушімен және сервермен өзара әрекеттесуге қатысты барлық нәрселерді жасай алады:

– теңшелетін HTML формаларының дұрыс толтырылуын тексеру;

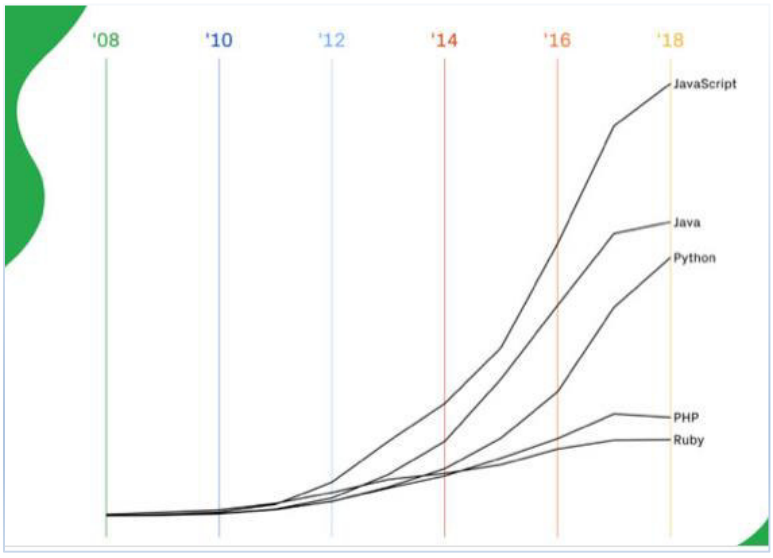
– веб-камерамен, микрофонмен және басқа құрылғылармен өзара әрекет ету;

– HTML мәнерлерін өзгерту, жасыру, элементтер мен т. б. көрсету;

– қалқымалы және диалог терезелерін көрсету;

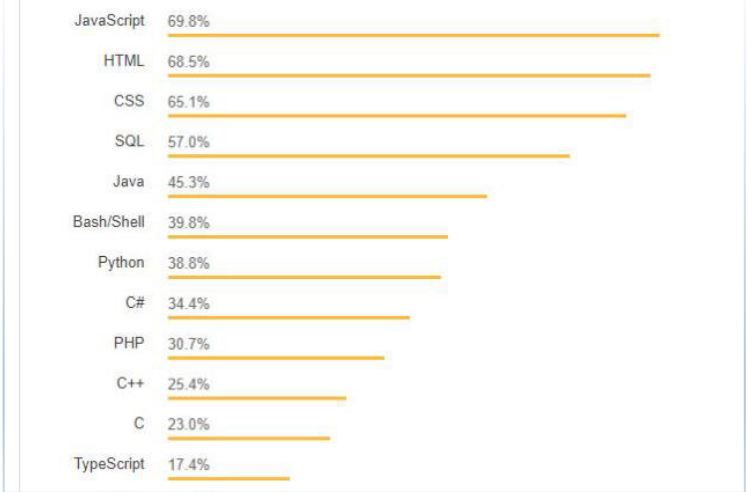
– серверге сұраныстарды жіберу және бетті қайта жүктеусіз деректерді жүктеу.

GitHub программистердің ашық кодты добалармен бөлісетін порталдың жыл сайын есеп өткізеді және статистика бойынша, төменде көрсетілгендей, 2018 жылы JavaScript жетекші орынға ие болды. Бұл сервистің жұмыс істеуі кезінде онда ең көп репозиторий 1.8-суретте көрсетілгендей JavaScriptтілі екенің байқауға болады, яғни 2018 жылы шамамен 1 млн.



1.8-сурет – GitHub порталы бойынша жетекші бағдарламалық тілдердің 2018 жылғы рейтингі

Сонымен қатар, программистердің қауымы жиналып, жас программистерді дағдылап отыратын танымал портал StackOverflow-дың есебі бойынша көп қолданылатын бағдарламалау тілдері кестесінде JavaScript тілі алғашқы орында тұр. 1.9-суретте 69,8%-бен JavaScript бағдарламалық тілі бірінші орынды иелегенің байқаймыз. Одан кейін, HTML, CSS, SQL, SQL, Java секілді тілдер рет бойынша тұрғанын көреміз.



1.9-сурет – StackOverflow порталы бойынша танымалдығы жағынан тілдердің 2018 жылғы рейтингі

1.6 Деректек қоры құралдарына талдау

Енді, бағдарламалық қамтаманың деректер қоры ретінде ДҚБЖ PostgreSQL таңдалынды. Дерек қор, ең алдымен, кестелер жиынтығы. Кесте объектілердің белгілі бір жиынтығының сипаттамалары (атрибуттары) бар екі өлшемді кесте ретінде қарастырылуы мүмкін. Кесте атау мен оған сілтеме жасайтын идентификаторға ие [18].

PostgreSQL -деректер қорын басқарудың еркін объектілік-реляциялық жүйесі (ДББЖ). Unix ұқсас платформаларда, соның ішінде AIX және HP-UX және Irix нұсқаларын, Linux үшін, macOS үшін, Solaris/OpenSolaris және, Tru64, ОЖ QNX үшін, сондай-ақ Windows үшін әртүрлі БСД жүйелерін қоса алғанда көптеген өнімдері бар. PostgreSQL-да кіріктірілген жүйелерде, ірі провайдерлерде, бұлтты есептеулерде және ірі жергілікті қондырғыларда миллиондаған өрістету бар жаңа қосымшаларды әзірлеу үшін жаңа әдістерді қолданады. Басқа жүйелердің артықшылықтары мен айырмашылықтары:

– шамадан тыс өрістетуге сезімталдық;

– қайта бөлу - бұл кейбір патенттелген деректер базаларын жеткізушілер лицензияның сәйкестілігіне байланысты бірінші мәселесі деп санайды;

– патент иеленушілерге қарағанда PostgreSQL деректер базасын құру кезіндегі ең жақсы қолдау;

– кәсіби энтузиастардың жарқын қауымдастығы PostgreSQL;

– штаттық шығыстарда айтарлықтай үнемдеу.

– жетекші патенттелген деректер базаларына қарағанда қызмет көрсету мен теңшеуге қойылатын талаптар төмен, бірақ барлық функцияларды, тұрақтылықты және өнімділікті сақтайды;

– өнімнің оқу бағдарламалары, әдетте, басқа деректер қорының жетекші жабдықтаушыларына қарағанда, практикалық ретінде бағалы қарастырылады;

– аты аңызға айналған сенімділік және тұрақтылық;

– бастапқы код тегін қол жетімді;

– егер пайдаланушы PostgreSQL-ді қандай да бір жолмен теңшеу немесе кеңейту қажет болса, онда ол оны ең аз күшпен және ешқандай шығынсыз жасай алады [17].

Кестенің бағандары нысандардың бір немесе басқа сипаттамаларына өрістерке сәйкес келеді. Әр өріс деректерді сақтаудың атауы мен түрлерімен сипатталады. Өріс атауы әртүрлі бағдарламаларда деректерді өңдеу үшін қолданылатын идентификатор болып табылады.

Дерекқор кестелерін құрастырған кезде, ақпараттың дәйектілігін қамтамасыз ету маңызды. Бұл, әдетте, негізгі өрістерді енгізу арқылы жүзеге асырылад. Бір немесе бірнеше өрістер кілтті болуы мүмкін.

Дерекқорлармен жұмыс істегенде, барлық өзгерістерді кэштеу жиі қолданылады. Бұл деректерге барлық өзгерістер, жаңа жазбаларды енгізу, бар жазбаларды жою, яғни, пайдаланушымен орындалатын барлық деректерді өңдеу ең алдымен дерекқорда жасалмаған, бірақ уақытша виртуалды кестеде жадта сақталады. Тек арнайы командадан кейін, деректер кестесіне енгізілген деректердің дұрыстығын тексергеннен кейін, пайдаланушы бұл деректердің барлығын дерекқорға жаза алады немесе оны қалдырып, өңдеуден бұрын мемлекетке оралуы мүмкін.

Реляциялық және объектілі-реляциялық ДҚБЖ ең көп таралған жүйелердің бірі болып табылады. Олар әр баған ("field" немесе "өріс" деп аталатын) реттелген және белгілі бір бірегей атауы бар кестелерді білдіреді. Жолдар тізбегі (оларды "records" немесе "жазбалар" деп атайды) ақпаратты кестеге енгізу кезектілігімен анықталады. Бұл ретте бағандар мен жолдарды өңдеу кез келген тәртіппен жүргізілуі мүмкін. Деректер кестелері өзара арнайы қатынастармен байланысты, соның арқасында әртүрлі кестелердегі деректермен жұмыс істеуге болады-мысалы, оларды бір сұрау арқылы біріктіруге болады.

Деректер қорының негізгі ерекшеліктері:

– құрылатын деректер қорының құрылымын анықтау, оны инициализациялау және бастапқы жүктеуді жүргізу;

– пайдаланушыларға деректерді манипуляциялау мүмкіндігін ұсыну (қажетті деректерді таңдау, есептеулерді орындау, енгізу/шығару интерфейсін әзірлеу, визуализация),

– қолданбалы бағдарламалардың және (логикалық және физикалық тәуелсіздік) тәуелсіздігін қамтамасыз ету;

– деректер базасының логикалық тұтастығын қорғау;

– физикалық тұтастықты қорғау;

– пайдаланушылардың деректер базасына қол жеткізу өкілеттіктерін басқару;

– бірнеше пайдаланушы жұмысын синхрондау;

– сақтау ортасының ресурстарын басқару;

– жүйелік персоналдың қызметін қолдау.

2 Жобалау бөлімі

2.1 Жобалаудың мақсаты мен міндеттерін анықтау

Дипломдық жұмыстың мақсаты – «Мұғалім студент көзімен» атты сауалнама, яғни түсінікті интерфейсі бар, сауалнаманы құруға оңай бағдарламаны даярлау болып табылады.

Дипломдық жұмысты құру алдында тұрған негізгі міндеттер төмендегідей:

– ЖОО мен студент арасындағы сенімді қарым-қатынасты орнату;

– мұғалімнің кәсіби шеберлігін және сапасы арттыру;

– студенттердің ЖОО-ның оқыту процесіне позитивті ықпал тигізетін ұсыныстарын жинау;

– жоғары кәсіби білімі және белсенді азаматтық ұстанымы бар мамандарды даярлауға бағытталған ЖОО жұмысын жақсарту тәсілі ретінде студенттердің әкімшілік басқаруға қатысуын кеңейту.

Міндеттерді шешудің ең тиімді тәсілі – электрондық сауалнаманы құруға және өткізуге арналған бағдарламаны құру.

Бағдарлама арқылы келесідей функцияларға қол жеткізеді:

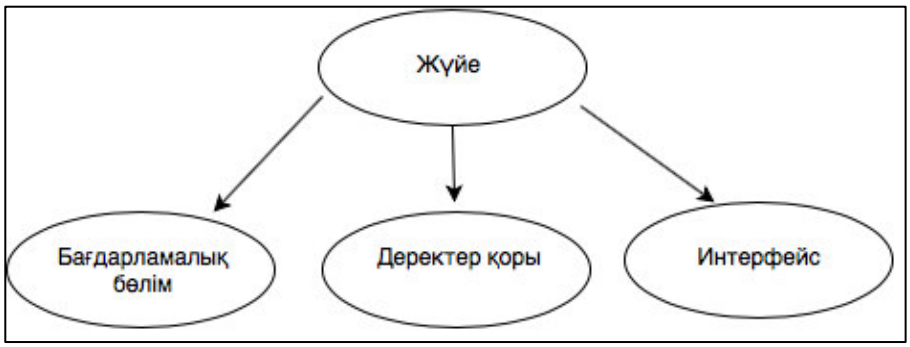
– сауалнаманы бес түрлі сұрақтарды таңдау арқылы оңай құрастыра алады;

– сауалнаманың қорытындылай тікелей сақтай алады;

– сауалнама қорытындыларын сұрыптай алады және ол мәліметті графикалық диаграммалар ретінде көрсетеді;

– бірнеше сауалнамаларды құрастыру мүмкіндігі бар.

Бағдарламалық қамтама ақпараттық, артық немесе қажетсіз ақпаратты қамтымауы тиіс әрі архитектурасы қарапайым және интуитивті ыңғайлы болуы керек және 2.1-суретте көрсетілгендей менеждер, бағдарламалық және админ бөлімінен тұрады.



2.1-сурет – Жүйе архитектурасы

Жүйе архитектурасының бағдарламалық бөлімі өзі frontend және backend бөлімінің өзара байланысы ретінде қарастырылады.

Операциялық бөлімде жүйені әзірлеу қарастырылған және жүйе Ruby тілінде жазылған, ал деректер қоры ретінде PostgreSQL қолданылды. Интерфейсті құру JavaScript тілімен орындалды.

2.2 Бағдарламалық қамтаманы жобалау түрлері

Жобалау функционалдық талаптар мен тиімділік талаптарына, сондай-ақ ол жұмыс істейтін ортаның шектеулеріне сәйкес болуы тиіс шешімді сипаттауға арналған. Бұрын жиналған талаптар ықтимал технологиялық шектеулерді қанағаттандыру үшін нақтыланады және жақсартылады. Жобалау төмендегідей әрекеттерді қамтиды:

– деректер сұлбаларын және кластарды жобалау;
– компонентті жобалау;
– графикалық интерфейсті жобалау;
– жүйе архитектурасын жобалау.

Жүйенің құрылым, мінез-құлық, өзара іс-қимыл, деректер және басқару ағыны сияқты өзіндік ерекшеліктерін жақсы қалыптастыруға көмектеседі. Жауапкершілік саласын бөлуге мүмкіндік береді, бағдарламалық инженерияның негізгі принципі – мәселенің әр түрлі тапсырыстарға бөлу жолымен шешу – күрделілікті жеңуге және бейімдеушілік, қолдаудың қарапайымдылығы, кеңейтушілік және көп мәрте пайдалану сияқты талап етілетін техникалық қасиеттерге қол жеткізуге көмектесе алады.

Әрбір веб-бағдарлама пайдалы деректерді алу үшін веб-серверге http сұрауларын жібереді. Веб-сервердің басқаруындағы бағдарлама деректерді сақтау үшін қандай да бір модельді пайдаланады. Қазіргі әлемде көбінесе SQL деректер базалары қолданылады.

Әр веб-қосымшаны ресми түрде 3 өзара тәуелсіз бөлікке бөлуге болады:

а) веб-браузермен орындалатын модуль. Бұл бағдарлама браузерді қолдайтын кез келген тілде жазылуы мүмкін. Көбінесе JavaScript тілін барынша қолдау және үлкен кітапханалық қолдау ретінде қолданады. Бұл өте маңызды, өйткені жоба бюджетін айтарлықтай үнемдеуге мүмкіндік береді;

б) веб-сервердің басқаруындағы серверлік жағында орындалатын модуль. Бұл қолданба сіз таңдаған web-серверді қолдайтын кез келген тілде жазылуы мүмкін. Соңғы уақытта, жиі, бағдарламалау тілі ретінде Java тілі таңдалады. Бұл тіл де кітапхана қолдауына ие;

в) деректер қоры, бұл салада да кең таңдау бар. Oracle, DB2, PostgreSQL сияқты өнеркәсіптік деректер базасы бар. MySQL сияқты жеңіл деректер базасы бар. Деректер базасы шешілетін міндеттердің мақсаттары мен салаларын негізге ала отырып таңдалады.

Веб-қосымшаның архитектурасын құру кезінде құрылымдық бірліктер арасындағы тәуелділікті барынша азайту қажет. Жалпы жағдайда қосымша үш құрылымдық бірліктен тұрады:

– браузердің басқаруында жұмыс істейтін модуль;
– веб-сервердің басқаруымен жұмыс істейтін модуль;

– деректер қоры.

Веб-технологиялар қарапайым веб-қосымшаларды да, күрделі ақпараттық веб-жүйелерді да әзірлеуге мүмкіндік береді. Мұндай қосымшаларды табысты әзірлеу үшін бағдарламалық жасақтаманы әзірлеудің жалпы процесін (БҚ) түсіну қажет:

– орындалуы тиіс әзірлеудің негізгі іс-әрекеттері;
– іс-әрекеттер бір-бірімен қалай байланысты;
– оларды орындаудың қандай тәртібі;
– осы процеске қандай мамандар қатысуы тиіс.

Әзірлеу процесі жақсы анықталған шекаралар мен мағынасы бар базалық жұмыстар жиынтығына бөлінуі мүмкін:

– мәселені түсіну;
– шешімді жоспарлау;
– жоспарды орындау;
– алынған нәтиженің дәлдігін тексеру;
– ықтимал қателерді немесе дәлсіздіктерді жою мақсатында пысықтау.

Кез келген бағдарламалық қамтаманың әзірлеу процесінде келесі негізгі қызмет түрлерін атап көрсетуге болады:

– бағдарламалық қамтаманың, шешілетін проблеманы түсінуге арналған талаптарын анықтау;
– мәселені шешуді жоспарлау;
– жоспарды жұмыс істеп тұрған бағдарламалық кодқа түрлендіру;
– кодтау қателерін (бағдарламалық код) немесе белгілі бір талаптар мен оларды жүзеге асыру арасындағы сәйкессіздіктерді анықтауға арналған;
– өрістету пайдаланушыларға құрылған бағдарламалық қамтамамен жұмыс істеуге мүмкіндік беру;
– жұмыс жүйесін пайдалануды қадағалауға және оның жұмысқа қабілеттілігін сақтауға арналған;
– даму, яғни әзірленген шешімді уақытпен жақсарту үшін арналған;
– жаңа талаптар нысанында әзірлеу процесі үшін жаңа кіріс деректерін ұсыну.

Инжиниринг талаптар бағдарламалық қамтама жасалатын қажетті мүмкіндіктер мен сипаттамаларды түсінуге арналған. Бұл талдау функционалдық талаптарды (жүйенің қандай функцияларын орындауы тиіс) және функционалды емес талаптарды (ұсынылған шешімнің сапасы) анықтауға бағытталған.

Әзірленіп жатқан жүйенің артында тұрған жалпы идеяны, жаңа жүйе талап етілетін негізгі мүдделі тұлғаларды және жүйе пайдаланылатын шарттарды анықтауды көздейді. Анықталған талаптар осы жүйенің жоғары деңгейлі модельдерін жасау мақсатында өңделеді, ол қарастырылып отырған проблемалық аймақтың қажетсіз егжей-тегжейлерінен абстрагацияланады.

Жобалау процестері бір-бірінен ерекшеленеді және нақты объектіге, жобалаушы ұйымның өлшемдеріне және оның құрылымына, жоба түріне

(типтік шешімдер негізінде жобалау немесе жаңа бұйымның толық түпнұсқалық жобасы) байланысты.

Күрделі объектілерді жобалау процесінің логикалық схемаларын құрудың негізгі принциптері мыналар болып табылады:

– аралық жобалық шешімдер синтезінің қарапайым міндеттеріне жобалау-технологиялық құжаттаманың толық кешенін синтездеудің күрделі міндеттерін бөлу;

– синтездеу және талдау рәсімдерін кезектестіру;

– жобалаудың итерациялылығы;

– түпкілікті жобалық шешімге жақындауына қарай талдаудың мұқият күшейтілуі (модельдердің көптовариантылығы, күрделенуі).

Талдаудың күрделі есебін қарапайым қатарға бөлу жобалауға блоктық-иерархиялық тәсілдің қарастырылған ережелеріне сәйкес орындалады. Бөлу жобалау ұйымының тиісті бөлімшелері арасында жұмысты бөлуге, әзірлеуші ұжымының жобалық рәсімдерді параллель-дәйекті орындауын ұйымдастыруға мүмкіндік береді.

Синтез және талдау процедураларының кезектесуі құрылымдық синтез міндеттерінің көпшілігін орындау үшін ТЗ талаптарын қанағаттандыратын жобалық шешімдерді қатесіз алуды қамтамасыз ететін әдістер жоқтығына байланысты. Бұл синтез есептерін формализациялау қиындықтарымен байланысты, сондықтан негізгі шешімдерді адам эвристикалық тәсілдердің негізінде қабылдайды. Бұл ретте сапалық және сандық талаптардың алуан түрлілігін ескеру және қателерді болдырмау мүмкін емес. Сондықтан синтезде ұсынылған жобалық шешімдердің нәтижелері талдау процедурасының орындалуымен бақыланады.

Жобалау процесінің логикалық сұлбаларын екі иерархиялық деңгейде қарастыруға болады. Жалпы деңгейде жобалау процесінің моделі мыналарды көздейді:

– жобалау мақсаты өзгеріссіз (кем дегенде қандай да бір уақыт ішінде);

– жобаны құру үшін белгілі бір үлгідегі технология білімі талап етіледі;

– жобалау процесі құжатталуы және қандай да бір тәсілмен өндіру үшін пайдаланылуы мүмкін ақпаратты (жобаны) туындатады.

Күрделі техникалық жүйелер деп келесі қасиеттермен сипатталатын техникалық объектілерді түсінеміз: мақсатты, тұтастық және буындылық, иерархиялық, көпаспектілік және даму. Күрделі техникалық жүйелердің қасиеттері мен сипаттамалары:

– мақсатылық. Күрделі техникалық жүйе (КТЖ) кейбір мақсатқа жету үшін құрылады. Ол оның ішкі жүйесін қамтитын құрамда жұмыс істейтіндіктен, осы жүйеде туындайтын қайшылықтарды жою немесе қандай да бір қажеттіліктерді қанағаттандыру мақсаты деп есептеуге болады.

– тұтастылық және буындылық. КТЖ-өзара байланысты элементтерден тұратын біртұтас білім. КТҚ тұтастығы оны жүйенің нақты жағдайларында жұмыс істейтін және мақсатқа қол жеткізу мүддесінде өзара іс-қимыл жасайтын, әртүрлі сапасыз, бірақ үйлесімді бөліктерден тұратын біртұтас

тұтас ретінде түсінумен негізделеді. Жүйе элементтерінің және элементтер арасындағы тұрақты байланыстардың жиынтығы жүйенің құрылымы деп аталады. Жүйенің құрылымын шыңдары оның элементтеріне, ал қабырға – олардың арасындағы байланыстарға сәйкес келетін Графпен сипаттау ыңғайлы.

– әлемділік. КТЖ иерархияның неғұрлым жоғары деңгейіндегі жүйенің элементі ретінде ғана емес, Кіші жүйелер болып табылатын және иерархияның неғұрлым төмен деңгейіне тиесілі элементтердің жиынтығы ретінде де ұсынылуы мүмкін. Кіші жүйелер де бөліктерге бөлінуі мүмкін. Жүйені бөліктерге бөлу және одан кейінгі бөлек зерттеу декомпозициялар деп аталады. Жүйе декомпозициясын одан әрі мүшелігі орынсыз элементтер деңгейіне дейін жалғастыра отырып, КТЖ көпдеңгейлі иерархиялық құрылымын аламыз.

– көпаспекттілік. КТЖ оны жобалау кезінде ескеру қажет қасиеттердің әртүрлі топтарымен (аспектілермен) сипатталады. Қандай да бір аспектіде орындалған КТҚ сипаттамасы оны ұсыну деп аталады. Жүйенің негізгі түсініктері функционалдық, морфологиялық және процесс болып табылады.

– дамытушылық. КТЖ өмір циклінің бойы өз функцияларын, құрылымын, ішкі процестерін өзгертетін дамушы, яғни. КТЖ -дағы өзгерістердің себептері оның сыртқы ортасының өзгеруі және оның КТЖ-ға әсері болып табылады.

Аталған қасиеттер КТЖ сипаттауында туындайтын жобалау процесінің негізгі қарама – қайшылығын қалыптастыруға мүмкіндік береді, - жүйенің тұтастығы мен мақсаттылығын көрсететін тұтас сипаттама алу қажеттілігі мен оның иерархиялығынан, көп аспектілігінен, дамуынан тұратын күрделілігі арасында. Осы қайшылықты шешу үшін КТЖ -ның жоғарыда көрсетілген мағынадағы күрделілігін ескеретін, жасалатын сипаттамаларының кешенді сипатын қамтамасыз ету қажет.

Жобалау нәтижесінде КТЖ-ның жұмыс қабілеттілігін және жұмыс істеу сапасы мен тиімділігінің талап етілетін деңгейін қамтамасыз ететін бірыңғай келісілген жобасы құрылуы тиіс.

КТЖ-ның жұмыс қабілеттілігі КТЖ-ның кіші жүйелерінің өзара және жүйемен үйлесімділігіне, сондай-ақ КТЖ-ның функционалдылығына, яғни оған жүктелген барлық функцияларды орындау қабілетіне негізделеді, олардың тізбесі жобалау алдындағы зерттеулер кезінде қалыптастырылады және техникалық тапсырмада тіркеледі.

Үйлесімділікке КТЖ кіші жүйелері құрылатын үйлесімді типтік құралдарды пайдалану және әзірленетін компоненттерді қанағаттандыруы тиіс жүйелік талаптарды қалыптастыру есебінен қол жеткізіледі.

КТЖ-ның жұмыс істеу сапасы-нақты жұмыс істеу жағдайларында талап етілетін нәтижеге қол жеткізуді негіздейтін қасиеттердің жиынтығы.

КТЖ тиімділігі оның жұмыс істеуінің нәтижелерін және оны құру мен пайдалануға арналған ресурстардың барлық түрлерінің шығындарын салыстырумен айқындалады. КТЖ тиімділігінің жалпыланған критерийі

тиімділіктің көптеген жеке критерийлерінде табады, олардың әрқайсысы жүйенің бір тарапын сипаттайды және бағалануы мүмкін. КТЖ тиімділігіне мынадай міндеттерді шешу арқылы қол жеткізіледі:

– жобалық шешімдерді бағалау критерийлерін таңдаудың дұрыстығы;

– үлгілерді бағалау үшін пайдаланудың дұрыстығы;

– КТЖ компоненттерін құрудың неғұрлым кең ауқымды нұсқаларын генерациялау;

– КТЖ құру бойынша келісілген оңтайлы шешімдерді таңдау.

КТЖ тиімділігінің проблемасы оның жекелеген кіші жүйелерін тәуелсіз оңтайландыру жолымен шешілуі мүмкін емес және тұтас тәсілді талап етеді.

2. 3 Бағдарламалық қамтаманы логикалық жобалау

Use case – бұл пайдаланушы белгілі бір мақсатқа жету үшін жүйені қалай пайдаланатынын сипаттайтын бағдарламалық және жүйелік жобалауға арналған термин. Бұл процесс іске асырылуы тиіс функцияларды анықтайтын бағдарламалық қамтамасыз етуді моделдеу және туындауы мүмкін кез келген қателерді шешу әдісі ретінде әрекет етеді. Процестің үш негізгі элементі бар:

– актерлер – жүйемен өзара әрекеттесетін пайдаланушылардың түрі;

– жүйе – элементтердің болжамды мінез-құлқын анықтайтын функционалдық талаптар;

– мақсат – use case әдетте пайдаланушы өз жетістіктеріне қатысатын әрекеттер мен опцияларды сипаттайтын мақсаттарды орындау үшін бастамашылық етеді.

Әдістеменің сипаттамасы:

– функционалдық талаптарды ұйымдастыру;

– жүйені пайдаланушылардың өзара іс-қимыл мақсаттарын моделдеу;

– соңғы мақсаттарға триггер оқиғаларынан сценарийлерді жазу;

– іс-әрекеттің негізгі барысын және ерекше оқиғалар ағынын сипаттау;

– басқа оқиға функцияларына рұқсат ету.

Диаграммаларды жасау қадамдары:

– жүйе пайдаланушыларын анықтаңыз;

– әрбір санат үшін пайдаланушы профилін жасаңыз. Бұл жүйеге қатысы бар барлық рөлдерді қамтиды;

– жүйені қолдау үшін әрбір рөлге байланысты маңызды мақсаттарды анықтаңыз. Жүйенің баға ұсынысы маңызды рөлді анықтайды;

– үлгіге байланысты әрбір мақсат үшін пайдалану мысалдарын жасап, барлық прецедентте бірдей абстракция деңгейін ұстап тұрыңыз.

Жоғары деңгейдегі пайдалану қадамдары төмен деңгейдегі мақсаттар ретінде қарастырылады.

UML диаграмма – диаграммалар жиынтығын пайдаланып бағдарламалық жасақтаманы визуализациялау тәсілі. Технология негізін қалаушылар-Грэди Буч, Джеймс Рамбо, Ивар Джекобсон және Rational Software Corporation компаниясы. Олардың жұмыстары объектілі-бағытталған

дизайнның негізіне алынды, содан кейін бағдарламалық қамтамасыз етуді әзірлеу жобаларының кең ауқымын қамту үшін спецификациялар кеңейтілді. Қазіргі уақытта UML модельдеу үшін бағдарламалық қамтамасыз етуді әзірлеу үшін стандарт ретінде объектілерді басқару тобымен (OMG) қабылданады [19].

UML ағымдағы стандарттары он үш түрлі диаграммаларды талап етеді: класс, белсенділік, объект, прецедент, дәйектілік, пакет, күй, компонент, байланыс, құрамдас құрылым, өзара іс-қимылға шолу, уақыт және өрістету.

Бұл диаграммалар екі түрлі топқа ұйымдастырылған: құрылымдық және мінез-құлық диаграммалары немесе өзара іс-қимыл.

Құрылымдық, өз кезегінде, диаграмманың келесі түрлеріне бөлінеді:

– класстар — UML қоса алғанда, әрбір объектілі-бағытталған әдістің негізі болып табылады. Олар жүйенің статикалық құрылымын сипаттайды;

– пакеттер — әзірлеушілер кейде оларды жеке әдіс ретінде қарастырады. Пакеттік диаграммалар пакеттер арасындағы байланысты азайту үшін байланысты топтарға жүйе элементтерін ұйымдастырады;

– объект — белгілі бір уақытта жүйенің статикалық құрылымын сипаттайды. Олар дәлдік үшін класс диаграммаларын тексеру үшін пайдаланылуы мүмкін. Композиттік құрылымдық диаграммалар сыныптың ішкі бөлігін көрсетеді. Қатысушылар мен прецеденттерді пайдалана отырып жүйенің функционалдығын модельдейді;

– компоненттер — физикалық бағдарламалық ұйымдастырылуын, оның ішінде бастапқы кодты, орындалатын файлды (екілік кодты) сипаттайды;

– орналастыру — диаграммалары тораптарды, компоненттерді және қосылыстарды қоса алғанда, физикалық ресурстарды жүйеде көрсетеді;

– қызмет — белсенділіктен белсенділікке бақылау ағынын модельдеу арқылы жүйенің динамикалық сипатын бейнелейді. Әрекет жүйенің күйін өзгертуге әкелетін кейбір классты әрекет болып табылады. Әдетте, белсенділік диаграммалары жұмыс процесін немесе бизнес-процестерді және ішкі жұмысты модельдеу үшін қолданылады;

– кезектілік — уақыт ағымында хабар алмасу терминдерінде сыныптар арасындағы өзара әрекеттесуді сипаттайды;

– күй — сыртқы тітіркендіргіштерге жауап ретінде жүйенің динамикалық мінез-құлқын сипаттайды. Күй диаграммалары реактивті объектілерді моделдеу кезінде әсіресе пайдалы;

– байланыс — объектілер арасындағы өзара іс-қимылды бірізділікпен моделдейді. Олар жүйенің статикалық құрылымын да, динамикалық мінез-құлқын да сипаттайды. Көптеген жағдайларда UML 2.0 енгізілген бірлескен жұмыс диаграммасының оңайлатылған нұсқасы болып табылады;

– өзара әрекеттесу шолуы — белсенділік диаграммалары мен кезектілік комбинациясын білдіреді. Олар іс-қимыл реттілігін модельдейді және басқарылатын кірістерге неғұрлым күрделі өзара іс-қимылдарды деконструциялауға мүмкіндік береді;

– уақытша — белгілі бір уақыт кезеңінде болатын процестерге бағытталған мінез-құлық немесе интерактивті UML-диаграмма түрі болып табылады. Тізбек диаграммасының ерекше мысалы болып табылады;

Use case — диаграмма процесін анық түсіну болмаған кезде үлкен мәнге ие емес — ол нақты алгоритм болмаса, қадамдарды орындау тәртібін модельдеуге болмайды. Сарапшылар осы диаграммаларды мәтіндік нұсқаны толықтыру үшін пайдалануды ұсынады. Диаграммада пайдалану нұсқалары, субъектілер мен жүйелер арасындағы өзара байланыс жоғары деңгейде көрсетіледі. Осы себепті жиі саяси партия үшін use case диаграммалары қолданылады [20].

Диаграмма мұндай жағдайларда мінсіз:

– жүйелік-пайдаланушылық өзара іс-қимыл мақсаттарын ұсыну;

– жүйедегі функционалдық талаптарды анықтау және ұйымдастыру;

– контекст пен жүйенің талаптарын анықтау; прецеденттегі оқиғалардың негізгі ағынын модельдеу.

Use case — диаграмма кір жуғыш машиналарының бағдарламалық жасақтамасын моделдеу кезінде оңтайлы визуализация арқасында өте кең қолданылады.

Жүйені модельдеу үшін ең маңызды аспект динамикалық мінез-құлықты басып алу болып табылады, ол жүйені іске қосу және одан әрі жұмыс істеу кезіндегі мінез-құлықты білдіреді.

Диаграмманың мақсаты — жүйенің динамикалық аспектісін түсіру. Дегенмен, бұл анықтама мақсатты сипаттау үшін тым ортақ болып табылады. Басқа төрт диаграммалар (белсенділік, бірізділік, ортақ пайдалану және Statechart) бірдей мақсатқа ие. USE CASE диаграммалары ішкі және сыртқы әсерлерді қоса алғанда, жүйеге қойылатын талаптарды жинау үшін қолданылады, әдетте, бұл дизайн талаптары. Демек, жүйе оның функционалдық мүмкіндіктерін жинау үшін талданғанда, пайдалану мысалдары әзірленеді және қатысушылар сәйкестендіріледі. Бастапқы тапсырма аяқталған кезде, кездейсоқ жағдайлардың диаграммалары сыртқы көріністі көрсету үшін үлгілеледі.

Use case — диаграммаларды құру мақсаты мыналарды атауға болады: талаптар жинау;

– жүйенің сыртқы түрін алу;

– сыртқы және ішкі факторлардың әсері;

– талаптар мен субъектілер арасындағы өзара іс-қимылды визуализациялау.

UML жүйенің динамикалық көрінісін модельдеу үшін бес диаграмма бар. Әрбір модельді пайдаланудың белгілі бір мақсаты бар. Іс жүзінде бұл нақты мақсаттар жұмыс жүйесінің әртүрлі бағыттарында қолданылады. Динамиканы түсіну үшін диаграммалардың әртүрлі түрлерін пайдалану керек. Оның нақты мақсаты — қатысушылардың жүйелік талаптарын жинау. Диаграммалар дизайнның өте жоғары деңгейінде қолданылады, онда деректер толық және практикалық көріністі алғанға дейін бірнеше рет нақтыланады.

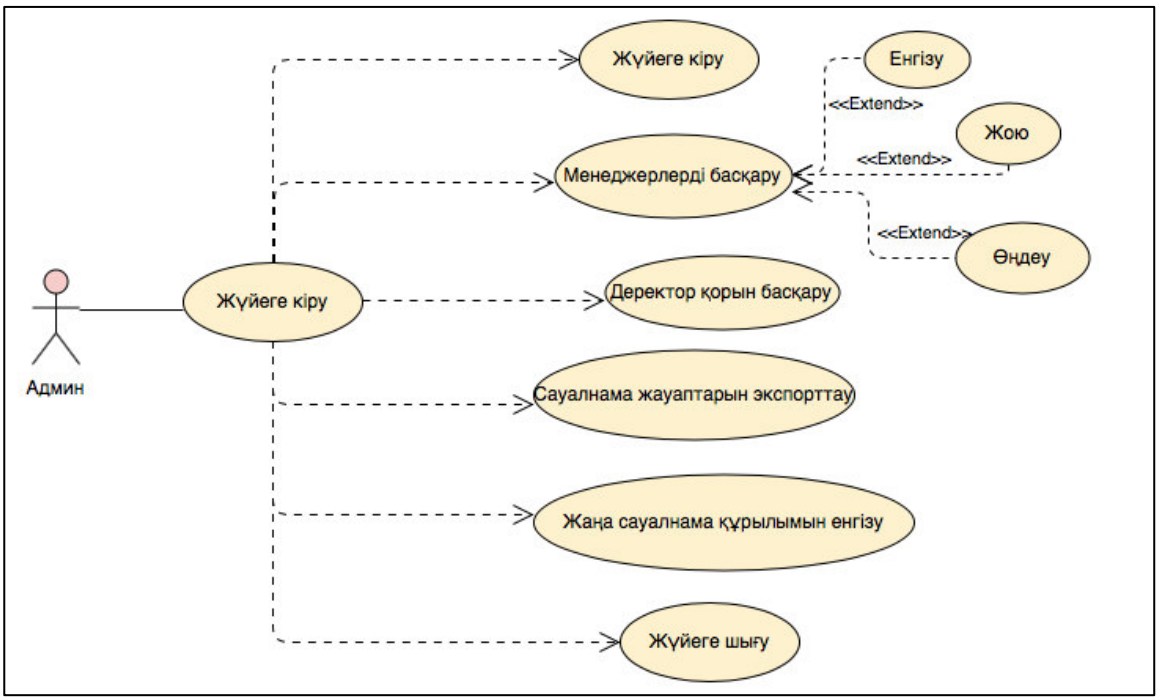
Бұл қосымша элементтер тестілеу кезінде мысалдар жасау үшін қолданылады. Диаграмма инженериясында тест сұрауларын жасау үшін, ал кері жобалау жағдайында қолданыстағы қосымшадан талаптар туралы мәліметтерді дайындау үшін қолданылады. Диаграммалар талаптарды талдау және жоғары деңгейлі дизайн, жүйе контекстін салыстыру және кері инжиниринг үшін пайдаланылуы мүмкін.

Дипломдық жұмысты жобалау кезеңінде бағдарламамен әрекеттесетін рольдерді таңдап, олардың іс-әрекеттерін тағайындау керек. Құрастырған жүйеде үш актер болады: “Админ”, “Менеджер” және “Студент”. 2.2-суретте көрсетілген “Админ” UML диаграммасы – бағдарламада әкімшінің барлық ықтимал әрекеттерін көрсететін пайдалану нұсқаларын көрсетеді (use case).

Әкімшінің қол жеткізе алатын іс-әрекеттері:

– авторизация;
– жүйеге кіру;
– деректерді басқару;
– менеджерлерді басқару.

Әкімші жүйеге кірген кезде, міндетті түрде логин және пароль енгізуі қажет. Кейін, аутентификациядан өткен жағдайда, ол жүйеге кіреді. Жүйеге кірген әкімші тек деректер қоры арқылы менеджерлерді басқара алады, яғни оларды қосуға, жоюға немесе олардың деректерін өңдей алады. Әкімші деректер қорымен жұмыс жасағанның өзінде, сауалнама жауаптарын өзгерте алмайды, тек оларды экспорттау опциясын таңдай алады. Сонымен қатар, деректер қорын басқару кезінде сауална кестелерін қосу, жою стандартты әрекеттерді орындай алады.



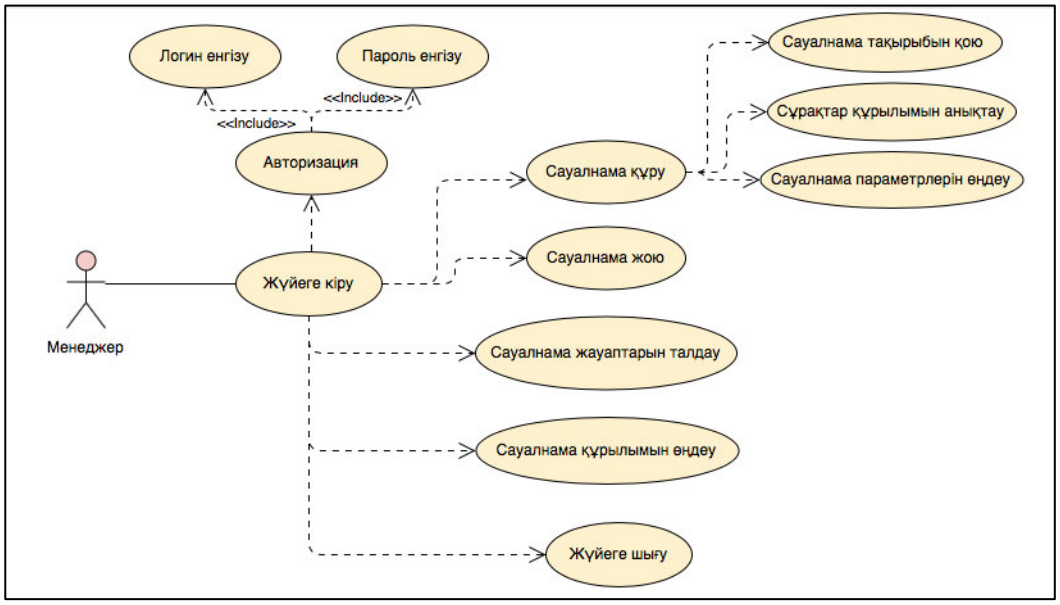
2.2-сурет – Админ use case диаграммасы

Бағдарламалық қамтадағы келесі актер “Менеджер” болып табылады және оның use case диаграммасы 2.3-суретте кескінделген.

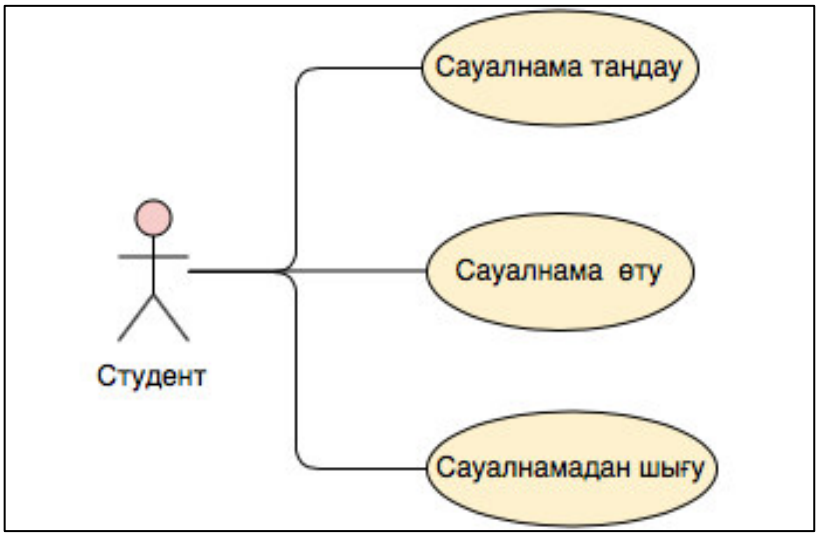
Менеджер қол жеткізе алатын іс-әрекеттері:

– авторизация;
– жүйеге кіру;
– сауалнама құру;
– сауалнама жою;
– сауалнама жауаптарын талдау.

Менеджер авторизациядан өткен соң, жүйеге кіріп, сауалнамаларды басқара алады. Бағдарламалық жүйеде сауалнаманы талдауға арнайы бет құрастырылған, бұл бетте жауаптарды ыңғайлы және оңай сұрыптай алады.



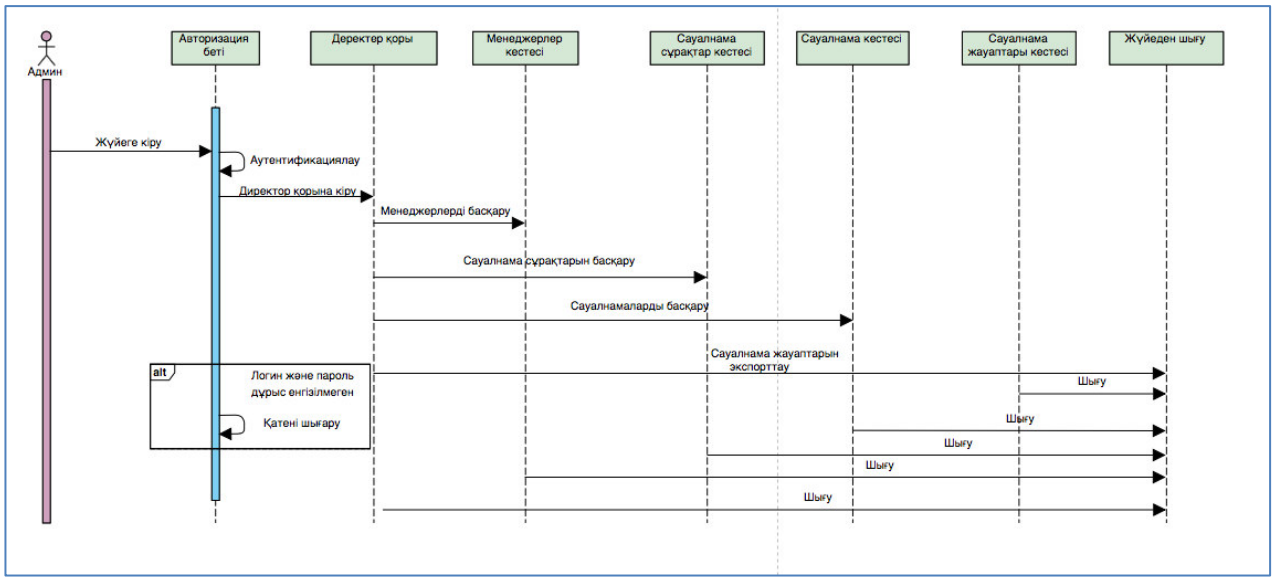
2.3-сурет – Менеджер use case диаграммасы



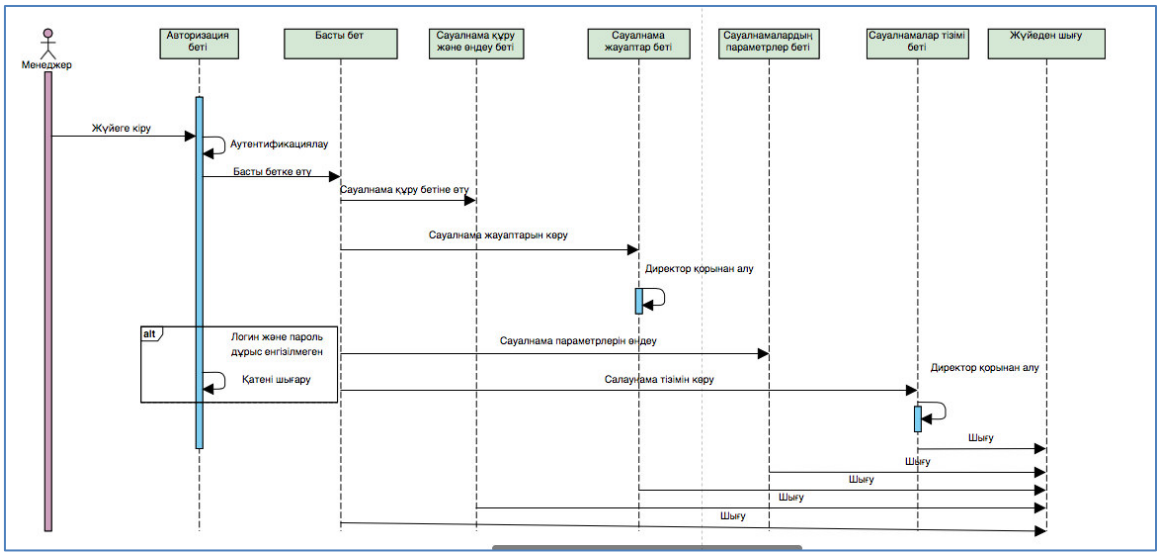
2.4-сурет – Студент use case диаграммасы

Жүйедегі соңғы актер бұл “Студент”, яғни сауалнама өтуші. Сауалнама анонимді түрде болғандықтан авторизация опциясы болмайды, тек сілтеме арқылы сауалнаманы таңдап, оны өтіп, шыға алады. 2.4-суретте іс-әрекеттері көрсетілген.

Менеджер жүйеге тіркелген кезде, әкімші оның қол жеткізу деңгейін анықтап, жүйеге еніге рұқсат береді, ал менеждер өз логин және паролі арқылы жүйеге кіріп, іс-әрекеттерін орындай алады. Әкімші сонымен қатар, менеджер туралы ақпаратты өңдей алады.



2.5-сурет – Администің тізбек диаграммасы



2.6-сурет – Менеджердің тізбек диаграммасы

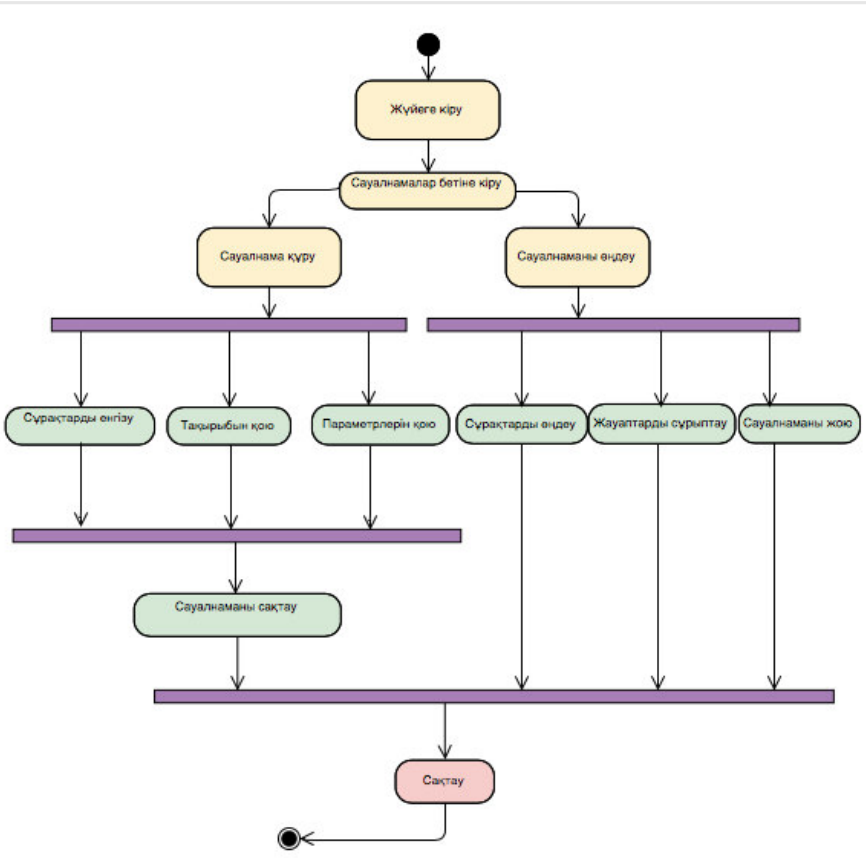
Тізбекті диаграмма және кооперация диграммасы әрекеттесу диаграммаларының жекеше жағдайларында қолданылатын диаграммалар

болып табылады. Администнің жүйе ішіндегі тізбек диаграммасы 2.5-суретте кескінделген. Менеджердің тізбек диаграммасы 2.6-суретте көруге болады. Бұл диаграммаларда актерлердің жүйеге кіргеннен бастап, шыққанға дейінгі іс-қимылдары және олардың финкциялары көрсетілген.

Күйлер диаграммасында әрекет түрі, оқиғасы, өтуі және күйі бар автоматты көрсетіледі. Бұл диграммада жүйенің динамикалық түріне жатады. Олар негізінен интерфейс, класс және кооперация тәртібін моделдеу кезінде маңызды. Олар обьектінің жұмыс істеу тәртібін оқиғаларға байланысты орындауды қамтамасыздандырады.

Күй диаграммалары объект болуы мүмкін түрлі күйлерді модельдеу үшін арналған. Класс диаграммасы кластардың статикалық суретін және олардың байланыстарын көрсетеді, жай-күй диаграммалары жүйенің жүріс-тұрыс динамикасын сипаттау кезінде қолданылады.

Қызмет диаграммасы – бұл күйлер диаграммасының жекеше түрі; олардың арасында жүйе ішіндегі бір басқару ағымынан екінші басқару ағымына өту тәртіптері көрсетіледі. Бұл диграммада жүйенің динамикалық түріне жатады. Менеждердің қызмет диаграммасы 2.7-суретте кескінделген.



2.7-сурет – Менеджер қызмет диаграммасы

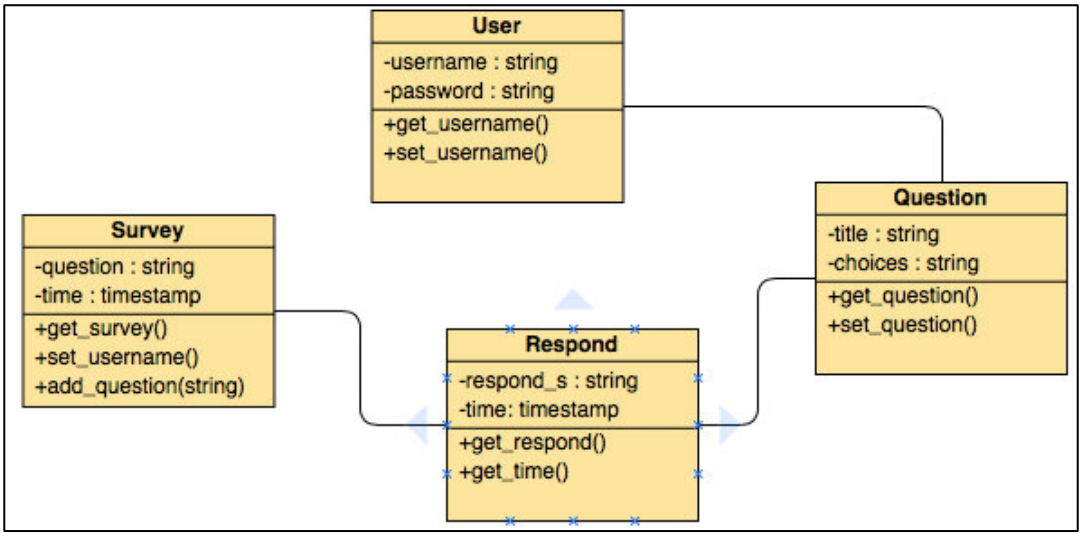
Класстар диаграммасы – жүйенің кластарын, олардың атрибуттарын, әдістері мен олардың арасындағы өзара байланысты көрсететін диаграмма.

Екі түрі бар:

– диаграмманың статикалық түрі - кластардың өзара логикалық байланыстарын қарастырады;

– диаграмманың аналитикалық түрі - жүйеге кіретін сыныптардың жалпы түрі мен өзара байланысын қарастырады.

Бағдарламалық қамтаманың класстар диаграммасы 2.8-суретте кескінделген. Мұнда User, Question, Respond және Survey класстарын көреміз.



2.8-сурет – Бағдарламаның класстар диаграммасы

Орналастыру диаграммасы – бұл жүйенің түйіндерін өңдейтін конфигурациямен және оларға орналастырылған компоненттерін көрсету диаграммасы. Ашып қарау диаграммасы ашып қарау көзқарасы жағынан жүйе архитектурасының статикалық түріне жатады. Бағдарламаның ашып қарау диаграммасы 2.9-суретте көрсетілген.

Орналастыру диаграммасы жүйенің физикалық орналасуын көрсетеді, ол қандай физикалық жабдықта бағдарламалық қамтамасыз етудің сол немесе өзге құрамдас бөлігі іске қосылатынын көрсетеді. Өрістету диаграммасы өте қарапайым, сондықтан біз қысқа боламыз.

Диаграмманың басты элементтері ақпараттық жолдармен байланысты тораптар болып табылады. Түйін (node) – бұл бағдарламалық жасақтама болуы мүмкін. Түйіндер екі түрі бар. ҚҰРЫЛҒЫ (device) - бұл физикалық Жабдық: компьютер немесе жүйемен байланысты құрылғы. Орындау ортасы (execution environment) - Операциялық жүйе немесе контейнер процесі сияқты басқа бағдарламалық жасақтаманы қамтуы мүмкін бағдарламалық жасақтама.

Тораптарда бағдарламалық жасақтаманың физикалық бейнесі болып табылатын артефактілер (artifacts) болуы мүмкін; әдетте бұл файлдар.

Мұндай файлдар орындалатын файлдар болуы мүмкін (файлдар сияқты).файлдары, DLL файлдары, JAR файлдары, жинақтар немесе сценарийлер) немесе деректер файлдары, конфигурациялық файлдар, HTML құжаттар және т.б.

Тораптарда бағдарламалық жасақтаманың физикалық бейнесі болып табылатын артефактілер (artifacts) болуы мүмкін.

Артефактілерді класты тікбұрыштар түрінде бейнелеуге немесе олардың аттарын торап ішінде көрсетуге болады. Егер осы элементтерді кластардың тікбұрышы ретінде көрсетсеңіз, құжат белгісін немесе "artifact"кілт сөзін қосуға болады.

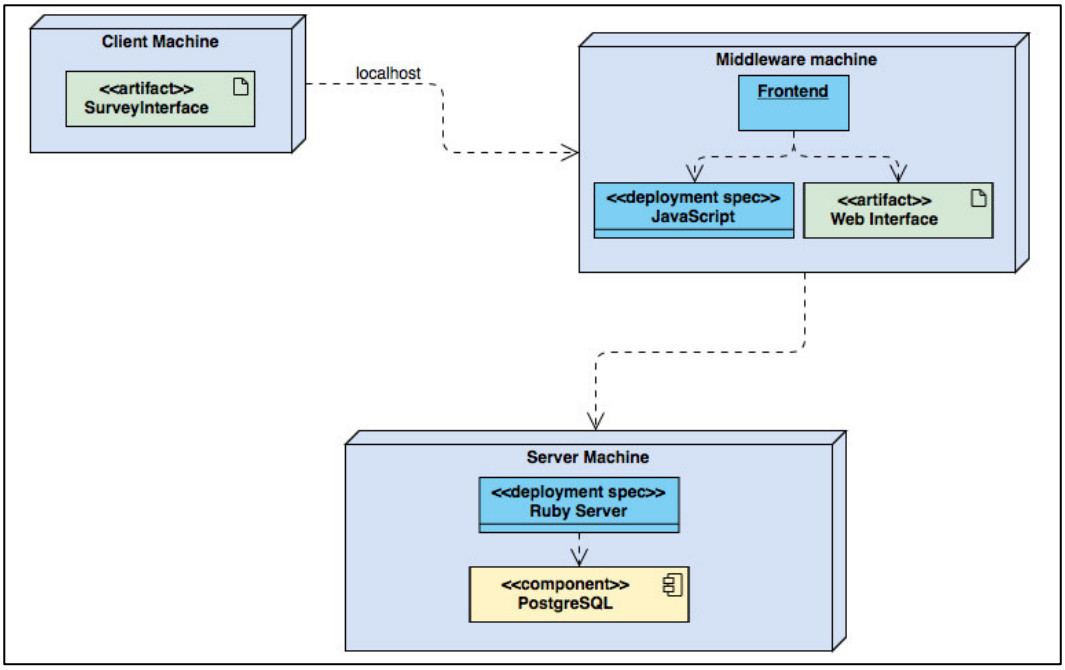
UML тілінде нысандардың өзара әрекеттесуін модельдеу үшін өзара әрекеттесудің тиісті диаграммалары қолданылады. Өзара іс-қимыл аспектілерінің бірі уақыт болып табылады. Объектілер арасындағы хабарламаларды жіберу мен қабылдаудың уақытша ерекшеліктерін ұсыну үшін тізбектілік диаграммасы қолданылады [14].

Тораптар немесе артефактілерді белгі түрінде алып жүруге болады, мысалы, өнім беруші, Операциялық жүйе, орналасқан жері – жалпы, сіздің басыңызға келетін барлық нәрсе туралы қызықты ақпаратты көрсету үшін.

Бір логикалық тапсырманы шешу үшін көптеген физикалық тораптар болады. Бұл фактіні түйіндер тіктөртбұрышын салу немесе белгі ретінде сан қою арқылы көрсетуге болады.

Артефактілер жиі компоненттерді іске асыру болып табылады. Бұл артефактілердің тікбұрыштарының ішіндегі белгі мәндерін қою арқылы көрсетуге болады.

Тораптар арасындағы ақпараттық жолдар жүйеде ақпарат алмасуды білдіреді. Бұл жолдарды пайдаланылатын ақпараттық хаттамалар туралы ақпаратпен сүйемелдеуге болады.



2.9-сурет – Орналастыру диаграммасы

3 Жүзеге асыру және тестілеу бөлімі

3.1 Бағдарламалау тәсілі

Қазіргі уақытта бағдарламаларды, жүйелерді және басқа да артефактілерді құрған кезде оларды басқа ортаға көшірілуі және онда жұмыс істеуі бағдарламашы және сол бағдарламаны қолданып жатқан қолданушы үшін өте маңызды. Әрбір бағдарламаны әзірлеуге кірісе отырып, ол үлкен жүйе екенін ескеру керек. Бағдарлама әзірлеуін жеңілдету үшін тәсілдердің бірі оны бағдарламалық модуль деп аталатын бөліктер бойынша құрастыру болып табылады. Мұндай бағдарламалау әдісі модульдік бағдарламалау деп аталады.

Модуль – бұл белгілі бір мәні бар және басқа бағдарламалық модульдерден берілген функцияны автономды өңдеуді қамтамасыз ететін бағдарламаның дербес бөлігі. Әрбір бағдарламалық модуль бағдарламаның басқа модульдерінен бөлек компиляциядан өткізеді және өзгертіледі. Әрбір бағдарламалық модуль, егер оны құжаттамасында сипатталғандай пайдалану шарттары орындалса, әртүрлі бағдарламалардың құрамына қосылуы мүмкін. Яғни, жақсы ойластырылған модуль бағдарламалауда қайталануды болдырмауға мүмкіндік береді.

Кез келген бағдарламалық қамтаманы құру ыңғайлылығы үшін арнайы құрылым әзірленген:

– әзірлемелер;
– бағдарламалау;
– жөндеу;
– енгізу.

Сондай-ақ, бағдарламалық қамтаманы құрылымдау осындай жағдайда көмек тігізеді:

– жұмысты орындаушылар бойынша бөлу, олардың жүктелуін және қажетті әзірлеу мерзімдерін қамтамасыз ету;

– жобалау жұмыстарының күнтізбелік кестелерін құру және құру процесінде оларды үйлестіруді жүзеге асыру;

– жобалық жұмыстардың құны мен еңбек шығындарын бақылау.

Бағдарламалық құралды құру кезінде бірнеше рет қолданылатын модульдер бөлінеді, оларды типтейді және бірегейлендіреді, соның есебінен жалпы бағдарламалық қамтама әзірлеуге мерзімдері мен еңбек шығындары қысқартылады.

Модульдік бағдарламалау – бұл модульдер деп аталатын шағын тәуелсіз блоктардың жиынтығы ретінде бағдарламаны ұйымдастыру, олардың құрылымы мен мінез-құлықтары белгілі бір ережелерге бағынады. Модульдік бағдарламалауды пайдалану бағдарламаны тестілеу мен қателерді анықтауды жеңілдетуге мүмкіндік береді. Аппараттық – тәуелді қосалқы берулер басқа тапсырыстардан қатаң түрде бөлек болуы мүмкін, бұл құрылатын бағдарламалардың ұтқырлығын жақсартады.

Модуль – бағдарламаның функционалды аяқталған фрагменті. Көптеген тілдерде бастапқы коды бар жеке файл немесе оның үздіксіз бөлігі түрінде ресімделеді. Кейбір тілдер модульдерді пакеттерге біріктіруді қарастырады. Модулдің төрт түрге бөлінеді және олар 3.1-кестеде көрсетілген.

3.1-кесте – Модуль түрлері

Модуль атауы	Сипаттамасы
Бас модуль	Бағдарламалық қамтаманы іске қосуды басқарады
Басқарушы модуль	Басқа модульдерді қоңырауды қамтамасыз етеді, кезекті модульді орындау үшін шақыру ретін анықтайды
Жұмыс модульдері	Өңдеу функцияларын орындайды
Сервистік модульдер	Қызмет көрсету функцияларын жүзеге асырады

Модуль қасиеттері:

– бір кіріс және бір шығыс – кіруде бағдарламалық модуль белгілі бір бастапқы деректер жинағын алады, деректерді өңдеуді орындайды және нәтижені қайтарады;

– функционалдық аяқталу – модуль әрбір жеке функцияны толық құрамда іске асыру үшін операциялар тізбесін орындайды;

– логикалық тәуелсіздік – модуль жұмысының нәтижесі тек бастапқы деректерге байланысты және басқа модульдердің жұмысына байланысты емес;

– басқа бағдарламалық модульдермен әлсіз ақпараттық байланыстар – модульдер арасындағы ақпарат алмасу мүмкіндігінше азайтылуы тиіс;

– көлемі мен күрделілігіне қарай.

Әрбір модуль модульдің спецификациясы мен денесінен тұрады. Спецификация – модульді пайдалану ережелері, ал дене – өңдеу процесін іске асыру тәсілінен тұрады.

Модульдік бағдарламалау-бұл модульдер деп аталатын шағын тәуелсіз блоктардың жиынтығы ретінде бағдарламаны ұйымдастыру, олардың құрылымы мен мінез-құлықтары белгілі бір ережелерге бағынады. Модульдік бағдарламалауды пайдалану бағдарламаны тестілеу мен қателерді анықтауды жеңілдетуге мүмкіндік береді. Аппараттық-тәуелді қосалқы берулер басқа тапсырыстардан қатаң түрде бөлек болуы мүмкін, бұл құрылатын бағдарламалардың ұтқырлығын жақсартады.

Модуль-бағдарламаның функционалды аяқталған фрагменті. Көптеген тілдерде бастапқы коды бар жеке файл немесе оның үздіксіз бөлігі түрінде ресімделеді. Кейбір тілдер модульдерді пакеттерге біріктіруді қарастырады.

Кейбір бағдарламалық құралдар стандартты процедуралар, функциялар, объектілер және деректерді өңдеу әдістері кітапханаларынан дайын модульдерді пайдаланады.

Программа бірнеше модульдерге бөлінеді және мынадай мақсаттар орындалуы тиіс:

– модульдің дұрыс болуы және оның контекстерге тәуелсіз болуы қажет;

– модульдің ішкі жұмыстарын білмей тұрып әртүрлі модульдерден программа құру мүмкіндігінің болуы қажет.

3. 2 Бағдарламаның интерфейсі

Бағдарламалық интерфейс – есептеу жүйесінің компоненттері арасында ақпарат алмасуға арналған біріздендірілген байланыс жүйесі. Бағдарламалық интерфейс қажетті процедуралар, олардың параметрлері мен қатынас тәсілдерін анықтайды.

Бағдарламалық интерфейс – кейбір бағдарламалық компонент басқа бағдарламалық компоненттерге ұсынатын функционалдылық. Мұндай функционалдылықтың екі түрін ажыратуға болады: қолданбалы бағдарламаларды жасау кезінде не пайдаланылатыны және жүйелік компоненттерді жасау кезінде не пайдаланылатыны. Бірінші, әдетте, қосымшаларды бағдарламалау интерфейсі деп аталады, екіншісі операциялық жүйе құрауыштарын бағдарламалау интерфейсі немесе жүйелік бағдарламалау интерфейсі деп аталады. Сонымен қатар, бағдарламалық интерфейстер бағдарламалық код деңгейіне немесе машиналық код деңгейіне қатысты әртүрлі деңгейлі болуы мүмкін.

Қолданбалы бағдарламалау интерфейсі – бұл пайдаланушы қосымшаларды бағдарламалау кезінде пайдаланылатын және пайдаланушы қосымшасы мен операциялық жүйе арасындағы дұрыс өзара іс-қимылды қамтамасыз ететін дайын константалар, құрылымдар мен функциялар жиынтығы.

Қосымшаларды бағдарламалау интерфейсі (API) протоколдар, рәсімдер, функциялар және/немесе бағдарламашылар бағдарламалық қамтамасыз етуді әзірлеу немесе әр түрлі жүйелер арасындағы өзара іс-қимылды жеңілдету үшін қолданатын командалар жиынтығы болып табылады. API үстелдік және мобильді пайдалану үшін қол жетімді және әдетте GUI компоненттерін (графикалық пайдаланушы интерфейсі) бағдарламалау үшін пайдалы, сондай-ақ бағдарламалық қамтамасыз етуге басқа бағдарламадан қызметтерді сұрауға және орналастыруға мүмкіндік береді.

API екі негізгі элементтен тұратын ретінде қарастыруға болады. Техникалық спецификация, ол бағдарламалар арасында ақпаратты қалай алмасуға болатынын анықтайды, өзі өңдеуге сұрау және деректерді жеткізу хаттамаларынан тұрады және осы спецификацияны қандай да бір түрде жариялайтын бағдарламалық интерфейс.

API негізгі концепциясы цифрлық технологиялардың бүкіл тарихы бойында қандай да бір нысанда болды,себебі бірегей бағдарламалар мен цифрлық жүйелер арасындағы өзара іс-қимыл осы технологияның көп бөлігінің негізгі мақсаты болды. Бірақ дүние жүзілік өрмекші пайда болып,

мыңжылдықтар тоғысында келесі бум доткомдар осы технология үшін стимул теңдесі жоқ деңгейге жетті.

Хаттама ұғымы API ұғымына мағынасы бойынша жақын екенін атап өту маңызды. Себебі, екеуіде абстракциялы функционалдылықты көрсетеді, тек бірінші жағдайда ғана деректерді беру туралы, ал екіншісінде — қосымшалардың өзара іс-қимылы туралы.

Бағдарламалық қамтамасыз ету индустриясында стандартты функционалдылық үшін жалпы Стандартты API маңызды рөл атқарады, өйткені олар жалпы API пайдаланатын барлық бағдарламалар бірдей жақсы немесе, кем дегенде, әдеттегі жолмен жұмыс істейтін болады. Бұл бағдарлама жаңа бағдарламалық өнімдерді игеру процесін жеңілдетеді.

Сауалнама жүйесіне енген кезде авторизация беті ашылады. Осы авторизация формасына қолданушы өзінің логин және пороліне енгізу қажет. Егер, қолданушы тіркелмеген болса, 3.1-суретте көрсетілгендей авторизация бетінің жоғарғы оң жақтағы “Register” батырмасы болады. Осы батырманы басып, ол 3.2-суреттегідей тіркелу процесінен өтеді. Осы кезде аты-жөні, электрондық поштасы, құпия сөз, яғни пароль және құпия сөздің растау мақсатында қайтадан енгізіледі.

3.3-суретте тіркелген қолданушының жүйеге кіру мақсатында авторизация өтіп жатқанын көре аламыз. Осы кезде қолданушы тіркелу кезінде енгізген өз поштасын және құпия сөзі арқылы жүйеге кіреді.

Жүйеге енген қолданушыға басты бет ашылады. Басты бетте, 3.4-суретте көрсетілгендей құрылған сауалнамалар тізімі және сауалнама құру “New Survey” батырмасы болады. Осы батырманы бсқан кезде, 3.5-суреттегі бет ашылады. Осы бетте қолданушы “Add Question”, “Edit Question”, “Edit Survey” батырмалары арқылы 3.6-суреттегідей 5 түрлі сұрақты, яғни “Single Line”, “Multiple Line”, “Multiple Choice”, “Checkboxes”, “Dropdown” қосуға, сұрақты өңдеуге және сауалнама туралы ақпаратты өңдеуге болады және сол арқылы сауалнама құрылымы пайда болады.

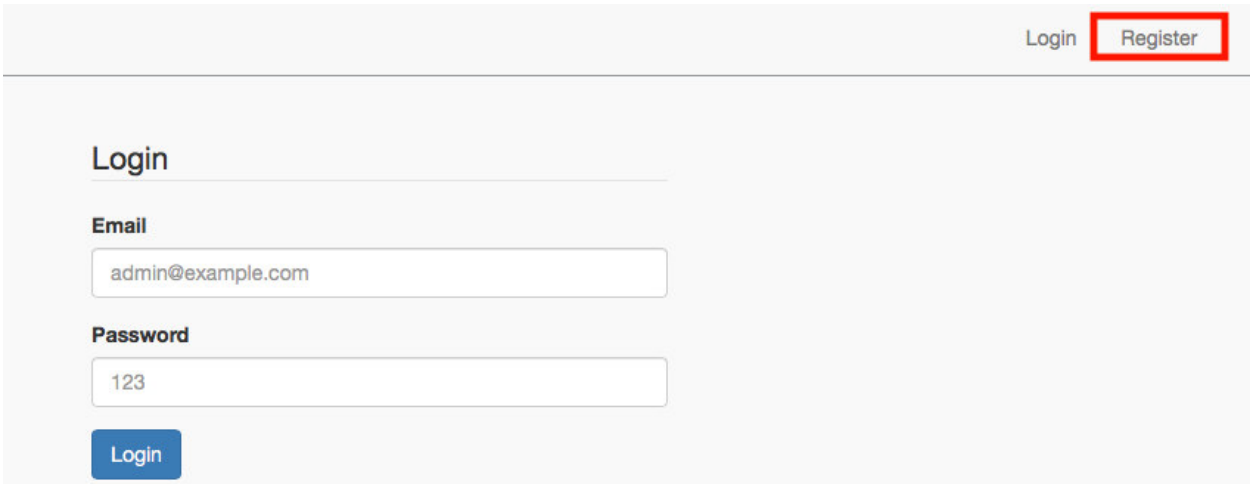
3.7-суретте сауалнаманы қалай сұрақтарды тарту арғылы оңай құрастыруға болатыны көрсетілген. Ал, дайын сауалнама беті 3.8-суретте көрсетілген.

Қолданушы сауалнаманы құрып болған соң, бұл сауалнамаға арнайы сілтеме автоматты түрде шығарылады. Кейін сауалнама өтушілер осы сілтеме арқылы сұрақтарға жауап беріп, сауалнама қорытындыларын жібере алады. Бұл жауаптар

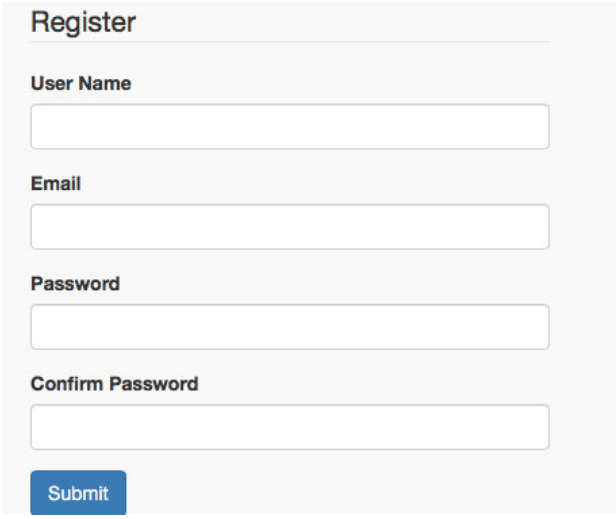
3.9-суретте көрсетілгендей қолданушының құрған сауалнамасына арналған арнайы бет «Data» сақталып және көрсетіліп тұрады. Бұл деректер кесте түрінде сақталынып тұрады және әрбір сауалнама жауаптары жіберілген сайын бұл кесте автоматты түрде сауалнама туралы деректермен инкременттеліп тұрады.

Сонында, егер қолданушы сауалнаманы жойғысы келсе “Delete” батырмасын басу арқылы жоя алады. Бұл батырма 3.12-суретте көрсетілген сауалнама параметрлері тұрған терезеде орналасқан. Ол үшін менеджер

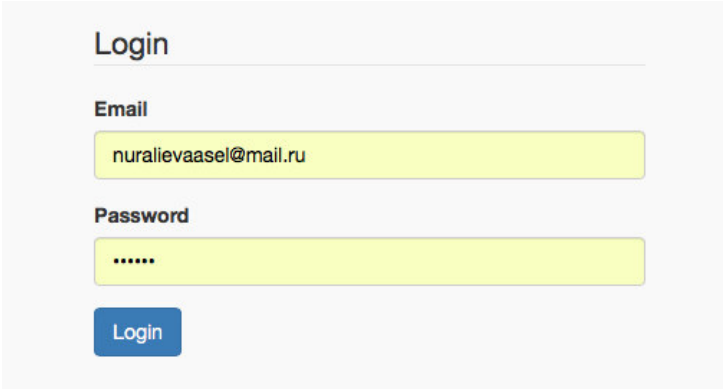
сауалнама параметрлер бетіне кіріп, осы бастырманы басуы керек. Осы кезде сауалнама деректер қорынан және менеджердің жеке бетінен жойылады.



3.1-сурет – Бағдарлама жүйесінің авторизация беті

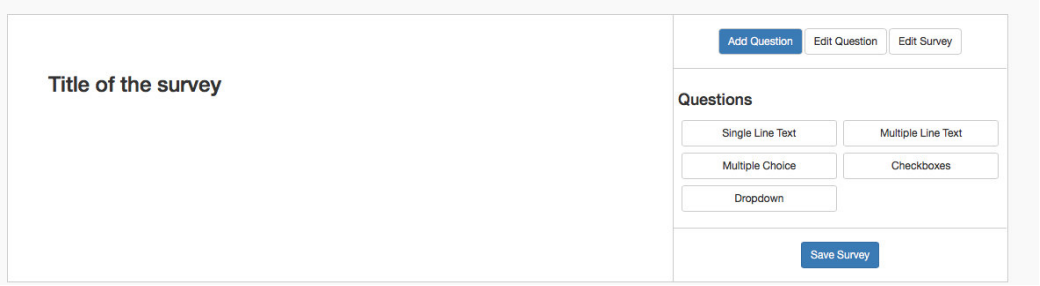


3.2-сурет – Тіркелу терезесі

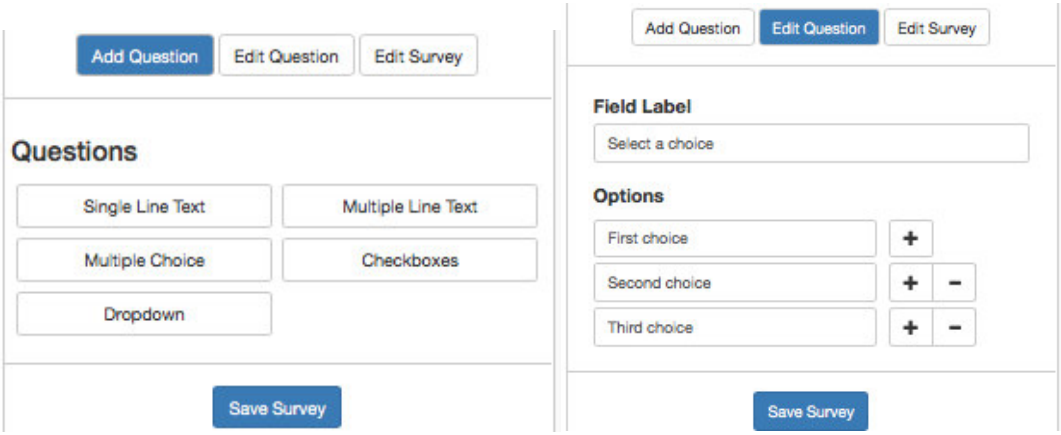


3.3-сурет – Авторизация терезесі

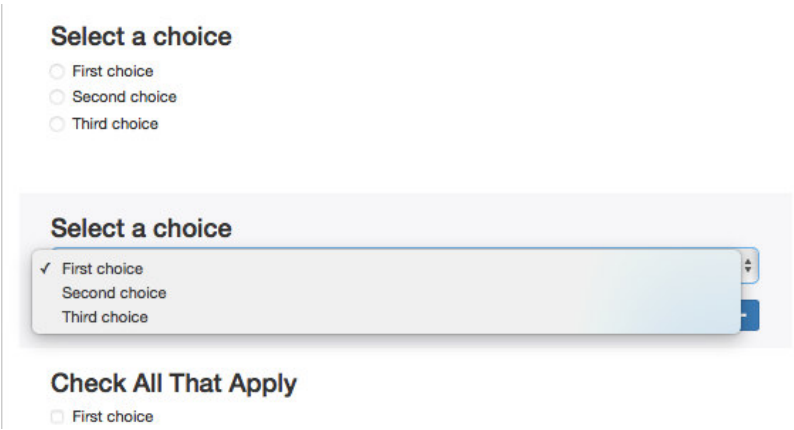
3.4-сурет – Басты бет



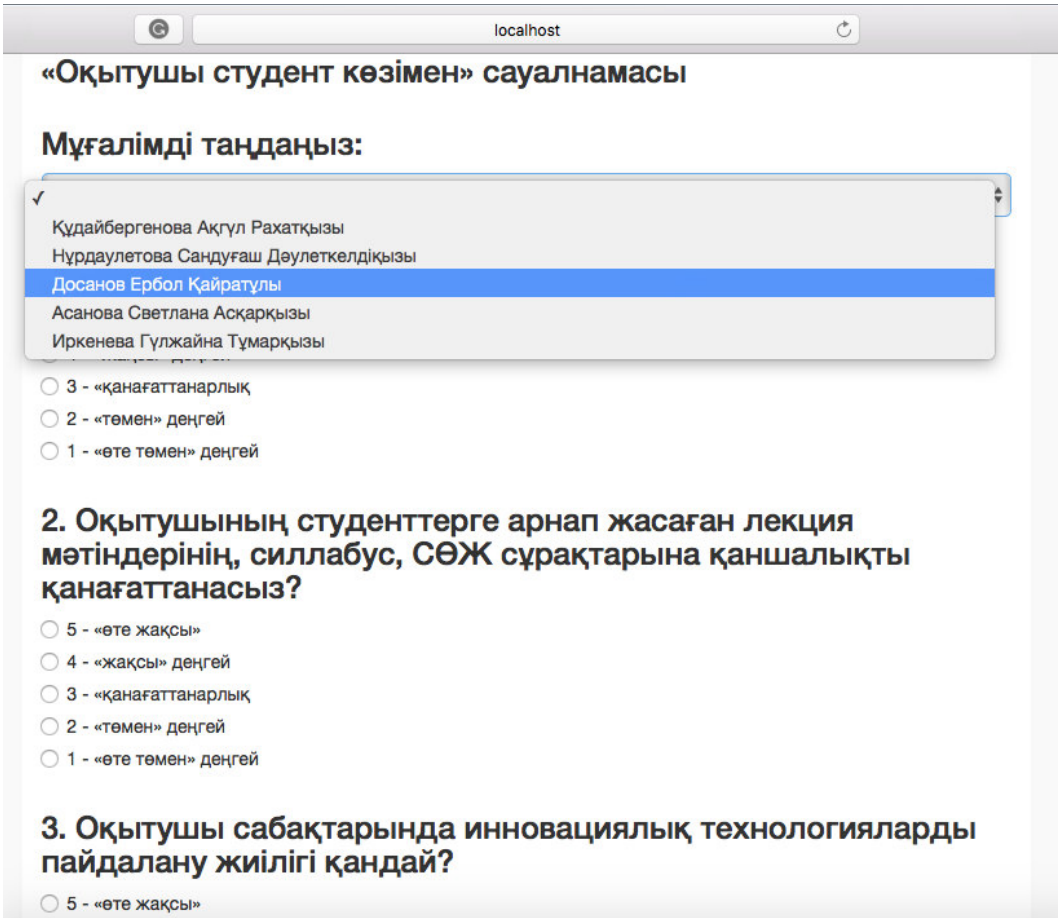
3.5-сурет – “New Survey” батырмасын басқаннан кейінгі терезе



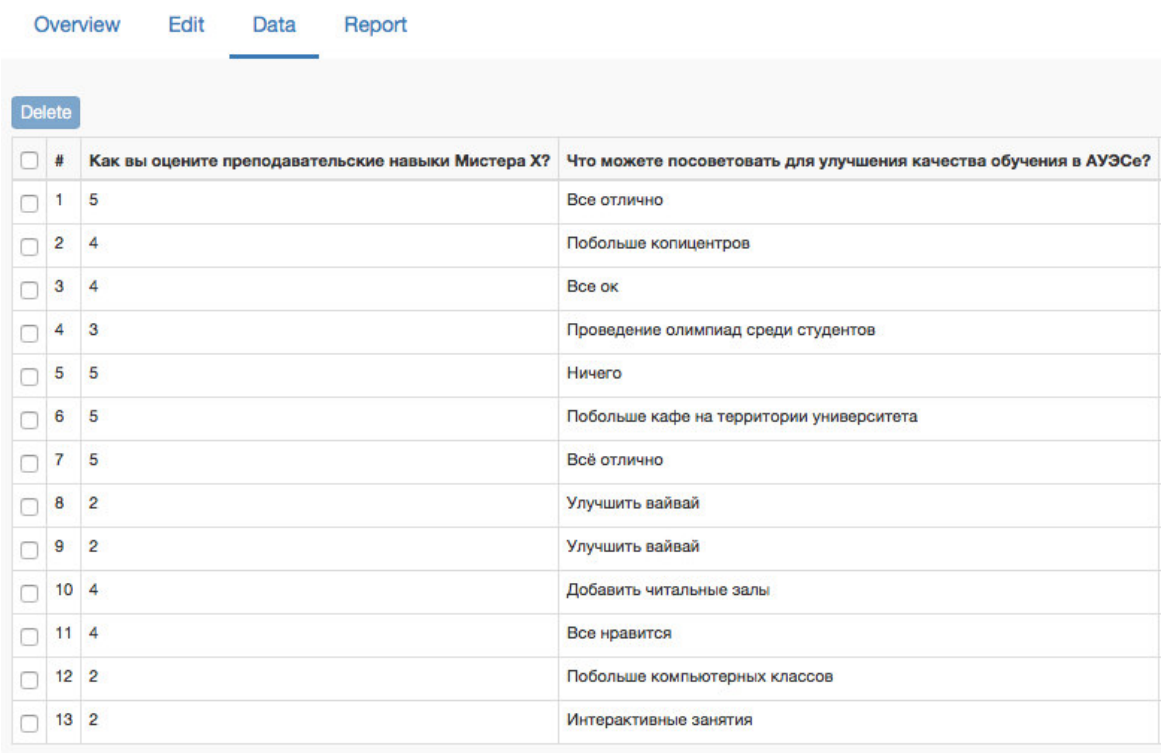
3.6-сурет – “Add Question” және “Edit Question” беті



3.7-суреттер – Сауалнаманы құру беті

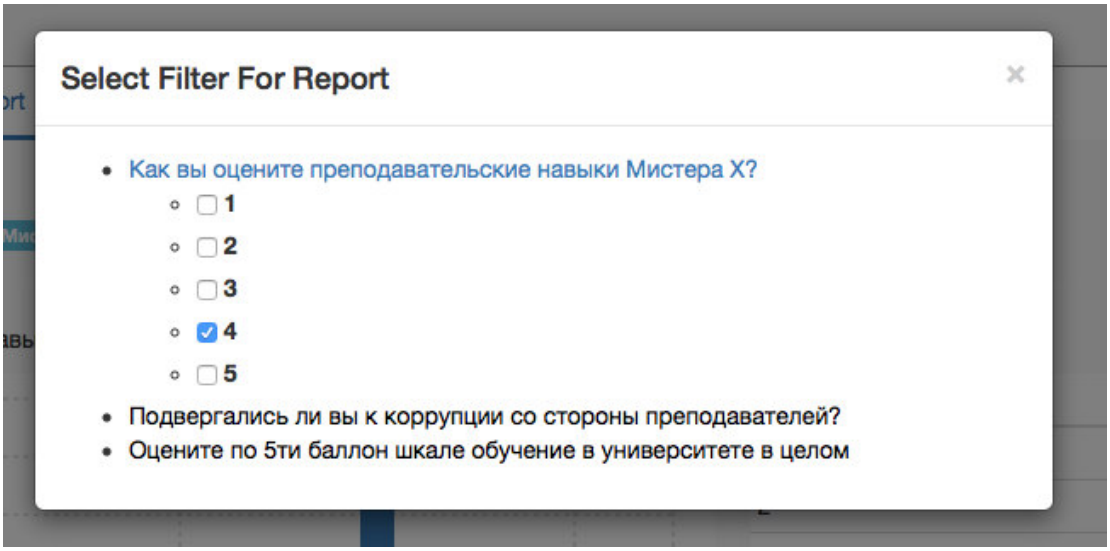


3.8-сурет – Дайын сауалнама беті

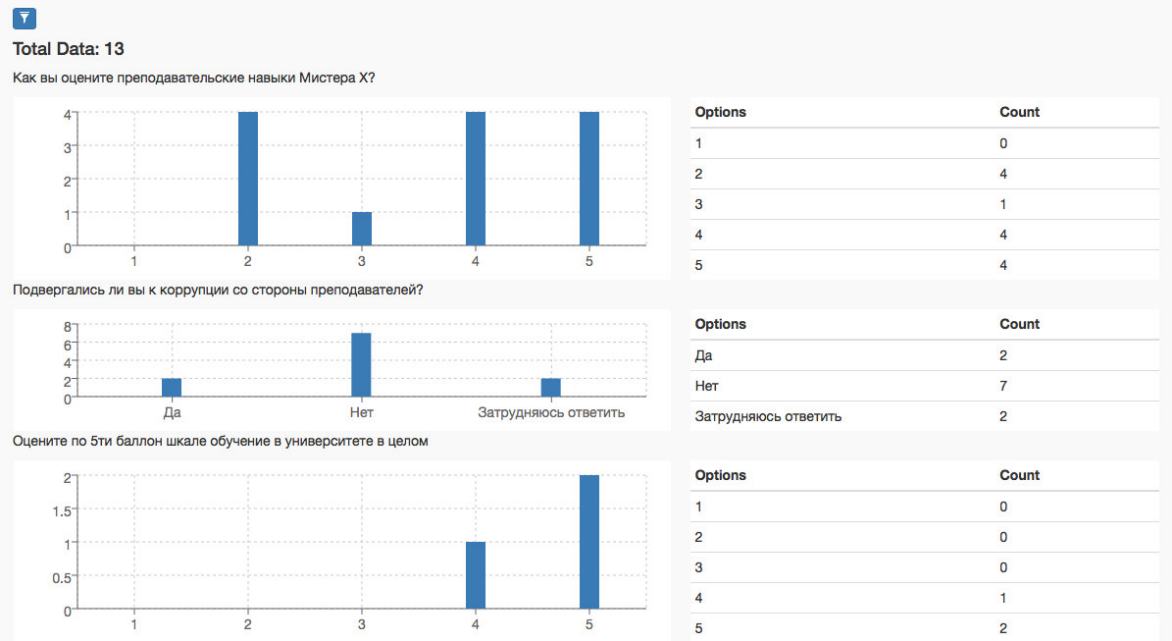


#	Как вы оцените преподавательские навыки Мистера Х?	Что можете посоветовать для улучшения качества обучения в АУЭСе?
1	5	Все отлично
2	4	Побольше копицентров
3	4	Все ок
4	3	Проведение олимпиад среди студентов
5	5	Ничего
6	5	Побольше кафе на территории университета
7	5	Всё отлично
8	2	Улучшить вайвай
9	2	Улучшить вайвай
10	4	Добавить читальные залы
11	4	Все нравится
12	2	Побольше компьютерных классов
13	2	Интерактивные занятия

3.9-сурет – Сауалнама нәтижелері



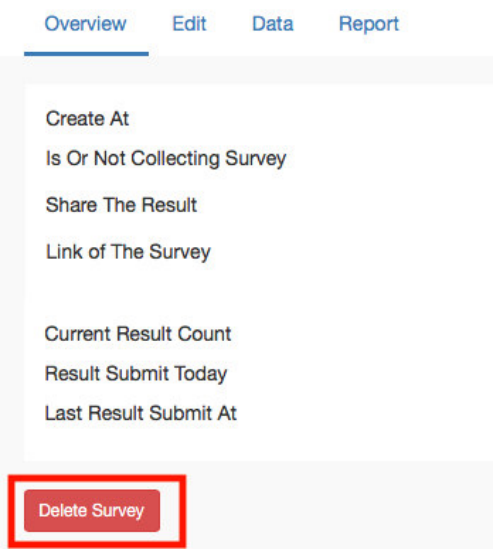
3.10-сурет – Сауалнама нәтижелерін сұрыптау терезесі



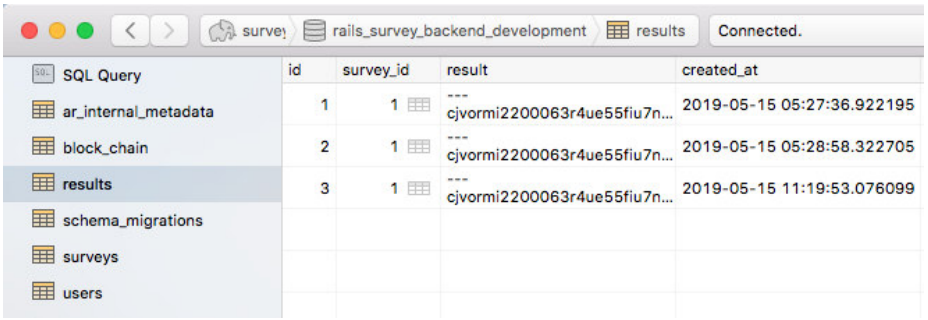
3.11-сурет – Сұрыпталған салаунама нәтижелері

Бұл бағдарламаның басты өзектілігі деректерді сақтау үшін блокчейн технологиясының қолдануында болды. Ол хеширлеу алгоритмін пайдалану арқылы жүзеге асырылды. Дипломдық жобада бұл технология жаңа түскен деректерді алдыңғы дерекпен байланыстырып, блоктар тізімін шығаруда болды, яғни блокчейн технологиясының принципін қолданып, сауалнамадан түсетін деректерді, яғни жауаптарды өңдеуге рұқсат етілмейді. Сонда, деректер қорында жауаптардың хеші сақталулы тұрады, ал толық ақпарат қолданушының сауалнама нітижелер бетінде болады. Сонда, бір-бірімен

байланысқан деректер тізімі пайда болады. Егер, сырттай деректердің арасындағы жауапты жоятын болса хеш толықтай өзгереді.



3.12-сурет – “Delete” батырмасы



3.12-сурет – PostgreSQL деректер қорындағы сауалнама жауаптарының кестесі

3.3 Бағдарлама тестілеу

Бағдарламалық қамтамасыз етуді тестілеу-зерттеу процесі, екі түрлі мақсаттары бар бағдарламалық өнімді сынау:

– әзірлеушілер мен тапсырыс берушілерге бағдарлама талаптарға сай екенін көрсету;

– бағдарламаның мінез-құлқы дұрыс емес, жағымсыз немесе ерекшелікке сәйкес келмейтін жағдайларды анықтау.

Бүгінгі таңда қолданыстағы бағдарламалық қамтамасыз етуді тестілеу әдістері барлық ақауларды бірден және толық анықтауға және талданатын бағдарламаның дұрыс жұмыс істеуін белгілеуге мүмкіндік бермейді, сондықтан тестілеудің барлық қолданыстағы әдістері зерттелетін немесе әзірленетін бағдарламалық қамтамасыз етуді тексерудің формальды процесі шеңберінде әрекет етеді.

Мұндай формальды тексеру немесе верификациялау процесі қолданылатын әдіс тұрғысынан ақаулар жоқ екенін дәлелдеуі мүмкін.

Бағдарламалық қамтамасыз етуді тестілеу және верификациялау мәселелерін шешуге көптеген тәсілдер бар, бірақ күрделі бағдарламалық өнімдерді тиімді тестілеу — бұл қатаң және нақты процедураларды ұстануға немесе осындайларды жасауға келмейтін жоғары дәрежеде шығармашылық процесс.

Бағдарламалық қамтамасыз етуді тестілеу – бұл:

– өнім сапасы туралы ақпарат алу мақсатында БҚ зерттеу процесі;

– жасанды жасалған жағдайларда және белгілі бір түрде таңдалған тестілердің шектеулі жиынтығында оның жұмысын бақылау жолымен жүзеге асырылатын өнімге мәлімделген талаптар мен нақты іске асырылған функционалдылықтың сәйкестігін тексеру процесі;

– жүйені бағалау қандай жүйе болуы тиіс және қандай жүйе арасындағы айырмашылықты табу үшін.

Бағдарламалық қамтамасыз етуді тестілеу-белгілі бір түрде таңдалған тесттердің соңғы жиынтығында жүзеге асырылатын бағдарламаның нақты және күтілетін мінез-құлқының арасындағы сәйкестікті тексеру. Кең мағынада, тестілеу-бұл жұмысты жоспарлау (Test Management), тесттерді жобалау (Test Design), тестілеуді орындау (Test Execution) және алынған нәтижелерді талдау (Test Analysis) бойынша белсенділікті қамтитын сапаны бақылау техникасының бірі.

Бағдарламалық қамтамасыз етудің сапасы (Software Quality) — бұл белгіленген және болжамды қажеттіліктерді қанағаттандыру қабілетіне қатысты бағдарламалық қамтамасыз етудің сипаттамаларының жиынтығы.

Верификация (verification) – бұл осы кезеңнің басында қалыптасқан шарттарды әзірлеудің ағымдағы кезеңінің нәтижелері қанағаттандыратын ма анықтау мақсатында Жүйені немесе оның компоненттерін бағалау процесі.

Валидация (validation) – бұл Пайдаланушының күтулері мен қажеттіліктеріне, жүйеге қойылатын талаптарға сәйкес келетіндігін анықтау.

Сондай-ақ басқа интерпритацияны кездестіруге болады:

Өнімнің нақты талаптарға (ерекшеліктерге) сәйкестігін бағалау процесі және верификация (verification), сонымен қатар өнімнің пайдаланушылардың күтулері мен талаптарына сәйкестігін бағалау валидация (validation) бар.

Тестілеуге арналған қосымшаның кез келген жағдайда дұрыс жұмыс істейтіндігін арттыру. Тестілеуге арналған қосымша барлық сипатталған талаптарға сәйкес болуы ықтималдығын арттыру.

Тестілеу кезеңдері:

– өнімді талдау;
– талаптармен жұмыс;
– тестілеу стратегиясын әзірлеу;
– сапаны бақылау рәсімдерін жоспарлау және жоспарлау;
– тест құжаттарын жасау;
– прототипті тестілеу;

– негізгі тестілеу;
– тұрақтандыру;
– пайдалану.

Тестілеу деңгейлері:

а) Модульдік тестілеу (Unit Testing). Компоненттік (модульдік) тестілеу функционалдығын тексереді және қол жетімді және жеке-жеке тестілеуі мүмкін (бағдарлама модульдері, объектілер, сыныптар, функциялар және т.б.) қосымша бөліктеріндегі ақауларды іздейді.

б) Интеграциялық тестілеу (Integration Testing). Компоненттік тестілеуді өткізгеннен кейін жүйе компоненттерінің арасындағы өзара әрекеттесу тексеріледі.

в) Жүйелік тестілеу (System Testing). Жүйелік тестілеудің негізгі міндеті жалпы жүйедегі функционалдық және функционалдық емес талаптарды тексеру болып табылады. Бұл ретте жүйе ресурстарын дұрыс пайдаланбау, пайдаланушылық деңгей деректерінің көзделмеген комбинациялары, қоршаумен үйлеспеушілік, көзделмеген пайдалану сценарийлері, жоқ немесе қате функционалдылық, пайдалану ыңғайсыздығы және т. б. сияқты ақаулар анықталады.

г) Операциялық тестілеу (Release Testing). Жүйе барлық талаптарды қанағаттандырса да, жүйенің бизнес моделінде анықталғандай, ол пайдаланушының қажеттіліктерін қанағаттандыратын және өзінің пайдалану ортасында өз рөлін атқаратынына көз жеткізу маңызды. Бизнес модельде қателіктер болуы мүмкін екенін ескеру керек. Сондықтан операциялық тестілеуді валидацияның соңғы қадамы ретінде өткізу маңызды. Бұдан басқа, пайдалану ортасында тестілеу бизнес саласында немесе бағдарламалық және электрондық ортада аралас басқа жүйелермен қақтығыс сияқты функционалдық емес мәселелерді анықтауға мүмкіндік береді.; пайдалану ортасында жүйенің жеткіліксіз өнімділігі және т. б. осындай заттарды енгізу сатысында табу-өте қиын және қымбат мәселе. Сондықтан, БҚ-ны әзірлеудің ең ерте кезеңдерінен верификациялауды ғана емес, валидацияны да жүргізу маңызды.

д) Қабылдау тестілеу (Acceptance Testing). Жүйенің төмендгідей талаптарға сәйкестігін тексеретін және тестілеу:

е) жүйе қабылдау критерийлерін қанағаттандыра ма;

ж) тапсырыс беруші немесе басқа уәкілетті тұлға шешім шығарған жағдайда қосымша қабылдайды немесе жоқ.

Ruby тест жазуды өте оңай ұсынады. Себебі, модельдер мен контроллерлерді жасағанда, ол тест кодының қаңқасын жасай бастайды. Ruby сынақтарын іске қосу сіздің кодыңыз кодты қайта жасағаннан кейін де қажетті функцияны ұстанады.

Rails платформасының негізгі ерекшелігі – бұл Ruby тілінің өзі. Динамикалық үлгілеу бар кез келген бағдарламалау тілі сияқты, Ruby жеткілікті икемділік, пайдалану ыңғайлылығы және жақсы өнімділікке ие. Бірақ бәрі өз бағасы бар. Динамикалық типтеу бағдарламалау тілдерінде

қателердің белгілі бір түрлерін, оның ішінде өте кең таралған қателерді және емле қателерін қадағалайтын компилятор жоқ. Динамикалық типиялануы бар объектілі-бағытталған бағдарламалау тілдерін пайдаланушылар өздерінің қосымшаларын тестілеу қажеттігін тез түсінді.

Rails әзірлеу ортасына таңқаларлық сенімді және пайдалануға дайын тестілеу жүйесі орнатылған. Ең аз күш-жігерді жұмсай отырып, деректер қорының ойнатылған баптауларын қоюға, веб-қосымшаларды HTTP тестілік хабарламаларын жіберуге және тестілеудің үш түрін орындауға болады:

– модульдік;
– функционалдық;
– кешенді.

Модульдік тесттер Rails моделінің кодын және кейде helper-әдістерін тексереді. Модульдік тесттер модельдің не үшін жасалғанын және модельдегі қауымдастықтар болжанғандай өзін-өзі ұстағанына көз жеткізуге мүмкіндік береді. Rails модельдері тек бір дерекқор кестесімен жұмыс істейтін объектілер екенін білесіздер. Көп жағдайда деректер қорының әрбір баған-модельдің атрибуты. Helper әдістері – модельдің, көріністің немесе контроллердің кодын жеңілдетуге көмектесетін функциялар. Әрбір модель немесе helper әдіс үшін тест бар екеніне көз жеткізу қажет.

Функционалдық тесттер жеке HTTP-сұраныстардың көмегімен пайдаланушы интерфейсінің жұмысын тексереді. Rails ортасы тест негізін қалыптастыра отырып, HTTP GET және POST жеке командаларын оңай бастауға мүмкіндік береді.

Кешенді тесттер функционалдық тестілерге ұқсас, бірақ олар көптеген каскадты HTTP-сұрауларды бастамауға мүмкіндік береді. Сынақтың принципі мен құрылымы бірдей.

Rails ортасы белгіленген шамалар арқылы қайталану мәселесін шешеді. Әрбір әзірлеуші тест деректері үшін тіркелген шамаларды анықтайды. Rails тестілеу ортасы барлық модельдердің деректерін толығымен өшіреді және әрбір тест деректерінің жиынтығы үшін берілген барлық бекітілген шамаларды жүктейді. Енді тест деректерінің әрбір жинағы нөлдік күйден басталуы мүмкін. Алайда әрбір тестілік деректер жиынтығында бірнеше жеке тест бар, олардың әрқайсысы басқалардан мүлдем тәуелсіз болуы тиіс. Әрбір жеке тест үшін бекітілген шамалардың тұтас жиынтығын жүктеу өте баяу өтеді.

Rails тамаша ымыраға келе отырып, жылдамдық мәселесін ішінара шешеді. Rails әрбір бақылау мысалын орындағаннан кейін деректер базасында жасалған барлық өзгерістерді қайтарады. Кері нөлден бастап барлық бекітілген шамаларды жүктеуден әлдеқайда жылдам жұмыс істейді. Бірақ деректер базасына қол жеткізу шығындары айқын. Деректер базасы негізінде тестілеу кейінге қалдырылған кезде де баяу болып қалады. Егер тесттер баяу жұмыс істесе, әзірлеушілер оларды іске қоспайды. Егер тесттер басталмаса, олар мүлдем пайдасыз. Rails қайталану проблемасын шешсе де, жылдамдық

мәселесі әлі де толық шешілмеген. Тестілеуді орындау жылдамдығы таяудағы жылдары тестілеу стратегияларының дамуына ықпал ететін болады.

Альтернативті тәсілдердің бірі тестілеу үшін жедел жадыда орналастырылған деректер базасын пайдалану болып табылады. Әдетте, SQLite MySQL қарағанда әлдеқайда жылдам жұмыс істейді. Екінші жағынан, тестілеу Сіздің өндірістік жүйесінің платформасынан ерекшеленетін платформада орындалуы мүмкін.

Құрылған бағдарламалық қамтама үш түрлі операциялық жүйелерде жұмыс жасап тұр және кодпен байланысты тестілеулерді өтті.

Бағдарламалық қамтамасыз етуді тестілеудің басты мақсаты қойылған талаптар арасындағы сәйкессіздікті анықтау үшін бағдарламалық өнімдерді зерттеу болып табылады және бағдарламалық қамтамасыз етуді әзірлегеннен кейін бізде не бар. Бүгінгі таңда қолданылатын әр түрлі бағдарламалық жасақтаманы тестілеу әдістері өнімнің барлық ақауларын дұрыс тауып, талдай алмайды, сол сияқты оның жұмыс істеуінің дұрыстылығы. Осыған орай, тестілеу сапасын анықтаудағы басты бағыт оны өткізу рәсімінің бағдарламалық қамтамасыз етуге қатысты формальды ережелер мен стандарттарға сәйкестігі болып табылады.Тестілеу нәтижелерін стандарттаудың әлсіз орны адам факторының әсерінен туындаған қателерді есепке алудың мүмкін еместігі болып табылады, ол өз кезегінде процедураның барлық кезеңдерінде өзін көрсетеді. Осы саладағы басты стандарт – ISO 9126 ("БҚ сапасын бағалауға арналған Стандарт") стандарттау жөніндегі халықаралық ұйымның тестілеу келесі критерийлерді сақтау арқылы жүзеге асырылуы тиіс екенін көрсетеді: Functionality, Reliability, Usability, Efficiency, Maintainability, Portability

Сондай-ақ, IEEE 829-1998 стандарты ("бағдарламалық қамтамасыз етуді тестілеудің бағыттық құжаттамасына арналған Стандарт") жұмыс істейді, ол тестілеуді өткізу үшін қажетті құжаттардың тізбесін анықтайды, атап айтқанда: оны өткізу жоспары мен журналы, тестілеудің жалпы және аралық нәтижелері туралы есептер.

Бағдарламалық өнімдерді тестілеу мен верификациялау мәселелерін шешуге көптеген тәсілдер бар, бірақ күрделі бағдарламалық өнімдерді тиімді тестілеу-бұл қатаң және нақты процедураларды ұстануға немесе осындайларды құруға әкеп соқпайтын жоғары деңгейдегі шығармашылық процесс.

ISO 9126 тұрғысынан, бағдарламалық қамтамасыз етудің сапасын келесі құрауыштарды ескере отырып, зерттелетін БҚ жиынтық сипаттамасы ретінде анықтауға болады:

- сенімділік;
- ілесіп жүру;
- үнемділік;
- тиімділігі;
- ұтқырлық;
- функционалдығы.

4 Экономикалық бөлім

4.1 Экономикалық бөлімде шешілетін мақсаттар мен міндеттер

Бұл дипломдық жобада Blockchain технологиясын қолдану арқылы студенттерге "Мұғалім студент көзімен" сауалнамасына арналған бағдарламалық қамтамасы құрылды.

Жүйені енгізудің тиімділігі білім алушыларға сауалнама жүргізу және құру процесін оңайлатуға негізделеді. Еңбек шығындарын азайту жолымен жалпы қаржылық шығындарды азайту, бұл жалпы өнімділік пен үнемдеуді арттыруға әкеледі.

Бұл бөлімнің негізгі міндеті зерттеу жүргізу шығындарының шамасын, қоғамдық өндірісте осы дипломдық жұмыста қойылған міндеттерді шешу кезінде алынатын негізгі және ілеспе нәтижелерді пайдаланудан экономикалық тиімділікті анықтау үшін өзіндік құнын анықтау болып табылады.

Бұл бөлімде жобаны іске асырудың экономикалық құрауышын қарау жүргізіледі.

4.2 Қолданбалы бағдарламаны әзірлеудің еңбек сыйымдылығын есептеу.

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

ҚБ әзірлеу кезеңдері	Жұмыс түрі	Еңбек сыйымдылығы, адам/сағ.
1 кезең	Жобаға қойылатын талаптарды талдау	18
2 кезең	Техникалық тапсырма	10
3 кезең	Эскиздік жобаны әзірлеу	14
4 кезең	Техникалық жобаны әзірлеу.	54
5 кезең	Бағдарламаны әзірлеу	46
6 кезең	Бағдарламалық құжаттаманы әзірлеу	137
7 кезең	Өнімді тестілеу	122
8 кезең	Бағдарламаны дайындау және тапсыру	13
ҚБ әзірлеудің барлық еңбек сыйымдылығы		414

4.3 ҚБ әзірлеуге жұмсалатын шығындарды есептеу

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

а) материалдық шығындар;

б) еңбек ақы төлеу шығындары;

в) әлеуметтік салық;
г) негізгі қорлардың амортизациясы;
д) басқа шығындар.

4.2-кесте – Материалдар және көмекші бөлшектер шығыны

Материалдық ресурстың атауы	Бірлік өлшемі	Саны	Бірлік бағасы, тг	Сомасы, тг
Ноутбук ASUS X555L	шт.	1	180000	180000
Барлығы				180000

Материалдар және көмекші бөлшектер шығыны (ЗМ) мынадай формула бойынша анықталады:

$$З_м = \sum_{i=1}^{n} Р_i * Ц_i, \quad (4.1)$$

мұндағы, Рі – материалдық ресурстың і-түрінің шығыны;
Ці – материалдық ресурстың і-түрінің бірлігінің бағасы, тг;
і – материалдық ресурстың түрі;
n – материалдық ресурстар түрлерінің саны.

Егер ҚБ әзірлеу үшін электр жабдықтары пайдаланылса, онда 1.3–кестеде келтірілген нысан бойынша электр энергиясына кететін шығындарды есептеу қажет.

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

$$З_э = \sum_{i=1}^{n} М_i * К_i * Т_i * Ц \quad (4.2)$$

мұндағы, Мі – і-ші электр жабдығының паспорттық қуаты, кВт;
Кі – і-ші электр жабдығының қуатын пайдалану коэффициенті (Кі = 0,7÷0,9);
Ті – і–ші жабдықтың барлық әзірлеу кезеңіндегі жұмыс уақыты;
Ц – электр энергиясының бағасы, тг / кВт×сағ;
і – электр жабдығының түрі;
n – электр жабдықтарының саны.

Энергия шығындары бағдарламалық қамтамасыз етуді әзірлеу кезеңінің ұзақтығына, бағдарламалық жасақтамаға жұмсалатын кВт/сағатқа және 1 кВт/сағ тарифіне негізделген. Заңды тұлғалар үшін Алматы қаласындағы тариф 2019 жылы 1 кВт / сағ үшін 18,32 теңгені құрайды, оның ішінде ҚҚС («АлматыЭнергоСбыт» ЖШС ресми сайтында берілген деректер бойынша).

$$З_э = 0.8 \cdot 0.8 \cdot 398 \cdot 18.32 = 4666,47$$

$$З_э = 0.3 \cdot 0.7 \cdot 398 \cdot 18.32 = 1531,18$$

4.3-кесте – Бағдарламалық қамтаманы құру үшін құралдар бағасы

Жабдық атауы	Паспорттық қуат, кВт;	Қуатты пайдалану коэффициенті, кВт	БҚ әзірлеуге арналған жабдықтың жұмыс уақыты, сағ	Электр энергиясы бағасы, тг/кВт×сағ	Сомасы, тг
Ноутбук ASUS X555L	0,8	0,8	398	18,32	4666,47
Жарықтандыру	0,3	0,7	398	18,32	1531,18
Барлығы					6540,24

4.4 Еңбек шығындарын есептеу

2019 жылы инженердің орташа жалақысы 180 000,00 тг, ал бағдарламашы - 180 000,00 теңге (Алматы қаласы үшін).

Қызметкердің айлық жұмыс уақыты мынадай формула бойынша анықталады:

$$Ч_м = N_м \cdot Ч_{рд}, \quad (4.3)$$

мұндағы, Чм – қызметкердің айлық жұмыс уақыты;
Nм - айына жұмыс күндерінің саны;
Чрд - тәуліктік жұмыс сағаттарының саны.

$$Ч_м = 21 \cdot 8 = 168$$

Қызметкердің сағаттық ставкасы мынадай формула бойынша есептелінеді:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i}, \quad (4.4)$$

Инженер-технолог:

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

Программист:

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

мұндағы, ЗПі – і-ші қызметкердің айлық жалақысы, м;

ФРВі - і-ші қызметкердің жұмыс уақытының айлық қоры, сағ.

ҚБ әзірлеудің әрбір қызметкер үшін еңбек сыйымдылығын анықтау үшін 4.1 кестедегі деректер пайдаланылады.

Инженер- әзірлеушінің еңбек сыйымдылығы – 277 адам/сағ. (жобаға қойылатын талаптарды талдау, техникалық тапсырма, эскиздік жобаны әзірлеу, бағдарламалық құжаттаманы әзірлеу, өнімді тестілеу, бағдарламаны дайындау және тапсыру).

$$Т_1 = 18 + 10 + 14 + 54 + 46 + 122 + 13 = 277 \text{ адам/сағ.}$$

ҚБ программистің еңбек сыйымдылығы - 269 адам/сағ. (эскиздік жобаны әзірлеу, техникалық жобаны әзірлеу, бағдарламаны әзірлеу, өнімді тестілеу).

$$Т_1 = 10 + 137 + 122 = 269 \text{ адам/сағ.}$$

Еңбек шығындарының жалпы көлемі (ЗТР) формула бойынша анықталады:

$$З_{ТР} = \sum_{i=1}^{n} ЧС_i \times T_i \text{,} \qquad (4.5)$$

мұндағы, ЧСі - і-ші қызметкердің сағаттық ставкасы, тг;

Ті – ҚБ әзірлеудің еңбек сыйымдылығы, адам/сағ.;

і - қызметкер санаты;

n - ҚБ-ны әзірлеумен айналысатын қызметкерлер саны.

Инженер-әзірлеуші:

$$З_{тр} = 1071{,}43 \cdot 277 = 296786{,}11 \text{ тг}$$

Программист:

$$З_{тр} = 892{,}86 \cdot 269 = 240187{,}41 \text{ тг}$$

Барлығы:

$$З_{тр.о} = 296786{,}11 + 240187{,}41 = 536973{,}52 \text{ тг}$$

$$З_{доп} = З_{тр} \cdot 10\% \text{,} \qquad (4.6)$$

$$З_{доп} = 536973,52 \cdot 0,1 = 53697,35 \text{ тг}$$

Жалақы қоры:

$$Ф_{зп} = З_{тр.о} + З_{доп}, \quad (4.7)$$

$$Ф_{зп} = 536973,52 + 53697,35 = 590670,87 \text{ тг}$$

Әлеуметтік салықты есептеу:

$$Н_c = (Ф_{зп} - ОПВ) \cdot 11\%, \quad (4.8)$$

онда, ОПВ - міндетті зейнетақы салымы, ол ФЗП-ның 10% -ын құрайды.

$$Н_c = \left(53697,35 - (53697,35 \cdot 0.1)\right) \cdot 0.11 = 5416,03 \text{ тг}$$

4.4 кесте – Еңбекке ақы төлеу құны

Квалификация	ҚБ әзірлеудің еңбек сыйымдылығы, адам/сағ.;	Сағаттық ставка, тг/сағ.	Сомасы, тг
Инженер-әзірлеуші:	277	1071,43	296786,11
Бағдарламашы	269	892,86	240187,41
Барлығы			536973,52

Амортизацияның жалпы сомасы мына формула бойынша анықталады:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{Ai} \times Т_{НИРi}}{100 \times Т_{Эфi}}, \quad (4.9)$$

мұндағы, Ф – і-ші ОФ-тың құны, тг;

НАі - і-ші ОФ-тың жыл сайынғы амортизациясы, %;

ТНИРі - ҚБ-ны әзірлеудің барлық кезеңіндегі і-ші ОФ жұмысының уақыты; сағ.;

ТЭфі – бір жылғы і-ші ОФ жұмысының уақытының тиімді қоры, жыл / жыл;

і - ОФ түрі;

n - ОФ саны.

Жылдық амортизация ставкасын есептеу:

$$H_{Ai} = \frac{100}{T_M}, \qquad (4.10)$$

мұндағы, TNi – і-ші ОФ-ті пайдаланудың ықтимал мерзімі;

$$H_{Ai} = \frac{100}{4} = 25$$

ҚБ-ны әзірлеуге арналған бағдарламалық қамтамасыз ету уақытын анықтау үшін 4.1-кестеден алынған мәліметтер қолданылады.

ҚБ-ны әзірлеуге арналған PhpStorm бағдарламалық жасақтамасының жұмыс уақыты 369 сағатты құрайды (техникалық жобаны әзірлеу, бағдарламаларды әзірлеу, өнімді тестілеу).

$$\mathrm{T}_1 = 54 + 46 + 137 + 122 = 369$$

Жабдық:

$$\mathrm{З}_{\mathrm{АМ}} = \frac{180000 \cdot 25 \cdot 414}{100 \cdot 1920} = 9704{,}12 \text{ тг}$$

Бағдарламалық жасақтама:

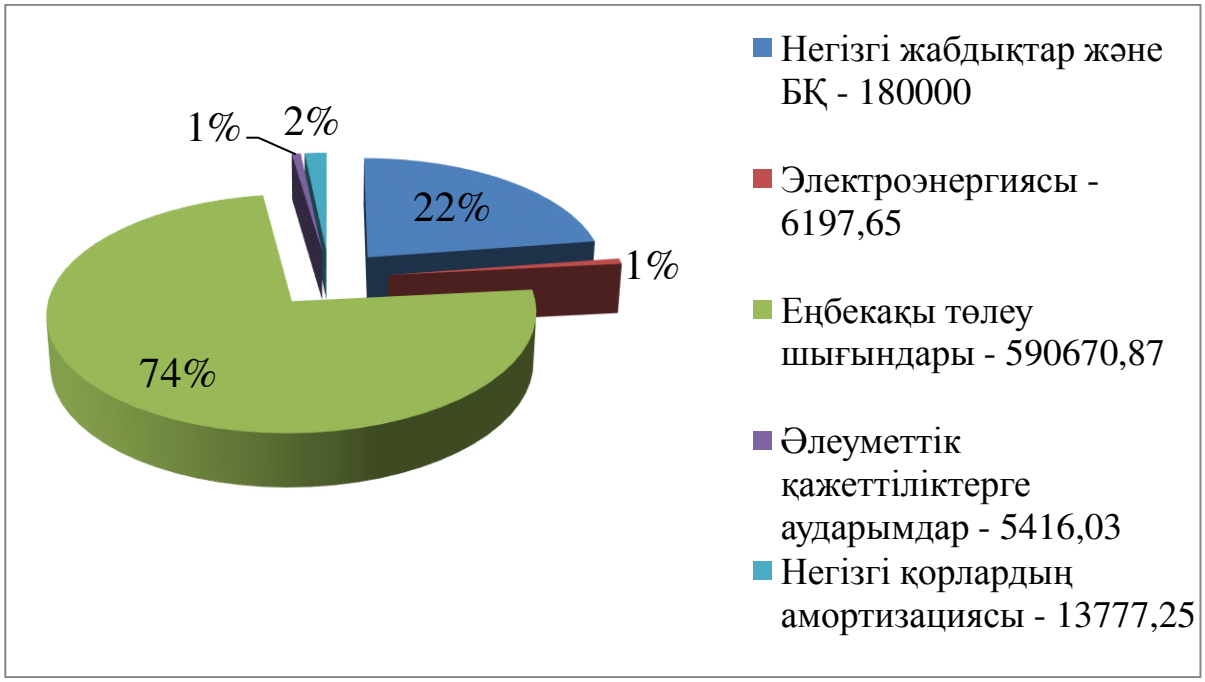
$$\mathrm{З}_{\mathrm{АМ}} = \frac{75709{,}44 \cdot 25 \cdot 351}{100 \cdot 1920} = 3460{,}15$$

4.5-кесте – Негізгі қорлардың амортизациясы

Жабдық пен бағдарламалық жасақтама атауы	Бағасы, тг	Жылдық амортизация, %	Жабдықтың және бағдарламалық қамтамасыз етудің тиімді жұмыс уақыты, сағ./жыл	Кеткен уақыты, сағ.	Сомасы, тг
Ноутбук ASUS X555L	150000	25	1920	414	9704,12
ОС	Тегін	-	1920	414	-
ПО PostgreSQL	Тегін	-	1920	351	-
ПО Ruby	75709,44	25	1920	351	3460,15
Барлығы					13164,28

4.6-кесте – ҚБ әзірлеуге арналған шығындар сметасы

Шығындар	Сомасы, тг
Негізгі жабдықтар және БҚ	180 000
Электроэнергиясы	6197,65
Еңбекақы төлеу шығындары	590670,87
Әлеуметтік қажеттіліктерге аударымдар	5416,03
Негізгі қорлардың амортизациясы	13164,28
Барлығы	795448,83



4.1-сурет – ҚБ әзірлеуге арналған шығындар сметасы

4.5 ҚБ ықтимал (шарттық) бағасын анықтау

Қолданбалы БҚ үшін шарттық баға мына формула бойынша есептеледі:

$$Ц_{Д} = З_{НИР} \times (1 + \frac{Р}{100}), \quad (4.11)$$

мұндағы, ЗНИР - ПҚ әзірлеуге арналған шығындар (4.6 кестесінен), тг;
Р - БҚ рентабельділігінің орташа деңгейі – 25%.

$$Ц_{д} = 795448{,}83 \cdot (1 + 0.2) = 954538{,}59 \text{ тг}$$

Одан кейін, сату бағасы қосымша құн салығын (ҚҚС) ескере отырып анықталады, ҚҚС ставкасы Қазақстан Республикасының Салық кодексінде белгіленеді. 2019 жылы ҚҚС ставкасы 12% деңгейінде белгіленеді.

ҚҚС қоса алғанда, сату бағасы мынадай формула бойынша есептеледі:

$$Цр=Цд + Цд * ҚҚС$$

$$Ц_р = 954538{,}59 + 954538{,}59 \cdot 0.12 = 1069083{,}22 \text{ тг}$$

Пайданы өзіндік құнның 25% мөлшерінде аламыз:

$$П = 954538{,}59*0{,}25=238634{,}64 \text{ тг}$$

4.6 ҚБ экономикалық бөлімі бойынша қорытынды

Бұл бөлімде әзірленген бағдарламалық өнімді енгізудің өзектілігі мен экономикалық тиімділігі тұрғысынан талдау жүргізілді. Ең алдымен, бағдарламалық өнімді жасау бойынша жұмысты орындау мерзімі мен еңбек сыйымдылығы анықталды. Келесі қадам қызметтік ақпарат ауысуды жүзеге асыратын бағдарламалық қамтаманы 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 795448,83 теңгеге тең, сату бағасы 1069083,22 теңгеге тең, ал алынған өнім кірісі 238634,64 теңгеге тең.

5 Өміртіршілік қауіпсіздік бөлімі

5.1 Бөлмедегі еңбек жағдайларының сипаттамасы

Дипломдық жұмыстың тақырыбы «Blockchain технологиясын пайдалана отырып, «Мұғалім студент көзімен» атты сауалнама жүргізетін бағдарлама қамтамасын құру», нәтижесінде пайдаланушыларға сауалнама жүргізуге мүмкіндік беретін бағдарламалық өнім алынуы тиіс. Бағдарлама университет үшін әзірленеді және маған өз жобаммен жұмыс істей алатын жеке бөлмемен қамтыды.

Жұмыс орны – бұл инженер еңбек қызметін жүзеге асыратын және жұмыс уақытының көп бөлігін өткізетін кеңістіктің бір бөлігі, сондықтан менің жұмыс орнымның конструкциясы және оның барлық элементтерінің өзара орналасуы антропометриялық, физикалық және психологиялық талаптарға сәйкес келеді.

Төмендегідей шарттар сақталған:

– жұмыс орнының құрамына кіретін жабдықтарды оңтайлы орналастыру;

– барлық қажетті қозғалыстар мен қозғалыстарды жүзеге асыруға мүмкіндік беретін жеткілікті жұмыс кеңістігі;

– акустикалық шу деңгейі рұқсат етілген мәннен аспауы.

Жұмыс бөлмесінде үш жұмыс ноутбук бар, олардың шуы жоқ. Бөлмедегі желдету жүйесі орнатылған, бұл жұмыс істеу қабілеттілігіне қолайлы фактор болып табылады.

L:B:H=7:5:3 параметрлері бар бөлмедегі еңбек деңгейін арттыруға кедергі келтіретін жалғыз мәселе, менің ойымша, жарықтандырудың жетіспеушілігі болып табылады.

Бұл бөлменің ауданы S = 1 м^2 тең екі терезесі бар және жалпы жарықтандыру шамдарында 3570 лм жарық ағыны бар 4 люминесцентті шамдар қолданылады (шамның түрі – ПВЛМ).

Осы факторларды ескере отырып, мен осы офистік үй-жайдың табиғи және жасанды жарықтандырылуын есептеуді шештім.

Өндірістік үй-жайларда жарықтандырудың үш түрі қолданылады:

– табиғи (оның көзі күн болып табылады);

– жасанды (тек жасанды жарық көздерін пайдалану);

– біріктірілген немесе аралас (табиғи және жасанды жарықтандыру үйлесімі).

Табиғи жарықтандыру табиғи жарық көздерімен – тікелей күн сәулесімен аспан қосындыларының диффузды жарығымен (күн сәулесінен, шашыраңқы атмосферамен) құрылады. Табиғи жарықтандыру адамның көзіне барынша бейімделген биологиялық ең құнды Жарық түрі болып табылады.

Өндірістік үй-жайларда табиғи жарықтандырудың мынадай түрлері қолданылады:

– бүйір-сыртқы қабырғалардағы жарық тесіктері (терезелері) арқылы;

– жоғарғы-жабындардағы жарық шамдары арқылы;
– аралас-жарық шамдары мен терезелер арқылы.

Өндірістік кәсіпорындарда жасанды жарықтандыру жасанды жарық көздері болып табылатын қыздыру шамдарымен және газ разрядты шамдармен жүзеге асырылады.

Табиғи жарықтандыруы жеткіліксіз ғимараттарда табиғи және жасанды жарықтың үйлесімі – біріктірілген жарықтандыру қолданылады.

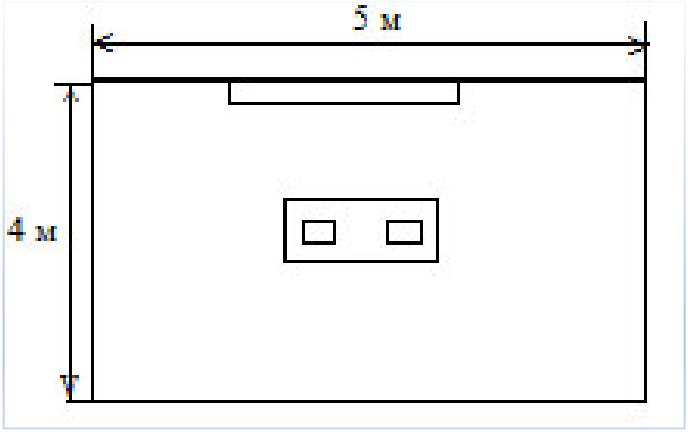
Қосарлы жарықтандыру жүйесіндегі жасанды жарықтандыру тұрақты жұмыс істеуі мүмкін (табиғи жарықтығы жеткіліксіз аймақтарда) немесе ымырт түскеннен кейін қосылуы мүмкін.

5.2 Бөлмедегі табиғи және жасанды жарықтандыруды есептеу

Жарықтандыру қондырғысын пайдалану кезінде жұмыс беттеріндегі жарықтандыру жарық көздерінің жарық ағынын азайту, шамдар мен жарықтандыру шырағданы ластануы, сондай-ақ жарықтандырылатын бөлме 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.

Бастапқы деректер:

Үй-жай түрі: кеңсе бөлімі;
Үй-жай параметрлері:
$L = 7$ м;
$B = 5$ м;
$H = 3$ м;
$h_{ok} = 2$ м;
$S_{ф.о.} = 1$ м^2; n=2;
Көру жұмыстарының разряды: I, а;
Көрсету коэффициенттері:
$\rho_{пот} = 70$;
$\rho_{стен} = 50$;
$\rho_{пол} = 30$;
Терезенің бастапқы биіктігі: $h_{нок} = 1$ м;
Жарық белдеуі: г. Алматы;
$H_{зд} = 12$;
Жақын тұрған ғимаратқа дейінгі қашықтық: $P = 6$.



5.1-сурет – Бөлмедегі терезе мен шамның нақты жағдайы

5.3 Табиғи жарықтандыруды есептеу

Есептеу жұмыс орнындағы жарықтану шарттарының талаптарға сәйкестігін анықтау үшін жүзеге асырылады. Бұл үшін emin табиғи жарықтандыру коэффициентінің ең аз мөлшерін (5.1) формула бойынша анықтаймыз:

$$e_{min} = \frac{100 * S_0 * \tau * r}{S_n * \eta * K_{зд} * K_з} \quad (5.1)$$

Терезелердің жалпы ауданын бүйірлік жарықтандыру үшін (5.2) формула бойынша анықтаймыз:

$$S_0 = \frac{S_n \cdot e_н \cdot \eta_0 \cdot K_{зд} \cdot K_з}{100 \cdot \tau_0 \cdot r_1}, \quad (5.2)$$

мұндағы, S_n– бөлмедегі еден ауданы, м^2:

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

$e_н$ – ТЖК нормаланған мәні: $e_N = e_н \cdot m_N$

$e_н$ - көру жұмысының а-бөлімшесінің I-ші разряды үшін ТЖК мәні $e_{КЕО}$ 2 тең;

m – жарық климатының коэффициенті терезелерді оңтүстікке бағдарлай отырып, жарық климатының ресурстары бойынша жарық ойықтарын бағдарлау үшін анықталады: m=0.8

$$e_н = 2 * 0.8 = 1{,}6$$

$К_з$ - кесте бойынша қор коэффициенті: $К_з = 1.2$

τ_0 - жарық өткізудің жалпы коэффициенті $\tau_0 = \tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4$,

τ_1 - материалдың жарық өткізу коэффициенті: $\tau_1 = 0.8$

τ_2 - жарықтың түптеуіндегі жарықтың жоғалуын ескеретін коэффициент: $\tau_2 = 0.75$

τ_3 - көтергіш конструкциялардағы жарықтың жоғалуын ескеретін коэффициент: $\tau_3 = 0.9$

τ_4- күннен қорғайтын құрылғылардағы жарықтың жоғалуын ескеретін коэффициент: $\tau_4 = 1$

$$\tau_0 = 0.8 * 0.75 * 0.9 * 1 = 0.54$$

η_0 – кесте бойынша терезелердің жарық сипаттамасы;

Анықтау үшін қажетті қатынастар η_0:

$$\frac{L}{B} = \frac{7}{5} = 1.4$$

$$h_1 = h_{н.ок} + h_{ок} = 2 + 1 = 3 \text{ м},$$

мұндағы, h_1 – шартты жұмыс бетінің деңгейінен терезенің үстіне дейінгі биіктік.

$$\frac{B}{h_1} = \frac{5}{3} = 1.67$$

Осылайша, $\eta_0 = 9.5$

r_1 – бөлменің сыртынан және ғимаратқа іргелес төселетін қабаттан шағылысқан жарықтың арқасында бүйірлік жарықтандыру кезінде ТЖК көтерілуін ескеретін коэффициент;

$$\frac{P_{пот} + P_{ст} + P_{пол}}{3} = \frac{70+50+30}{3} = 50\%$$

$$r_1 = 1.1$$

$К_{зд}$ – 3.8-кесте бойынша терезелердің көлеңкеленуін ескеретін коэффициент:

$$\frac{P}{H_{зд}} = \frac{6}{12} = 0.5$$

$$К_{зд} = 1.7$$

Барлық мәндерді есептік формулаға қоямыз:

$$S_0 = \frac{35 * 1.6 * 9.5 * 1.7 * 1.2}{100 * 0.54 * 1.1} \approx 18 \text{ м}^2.$$

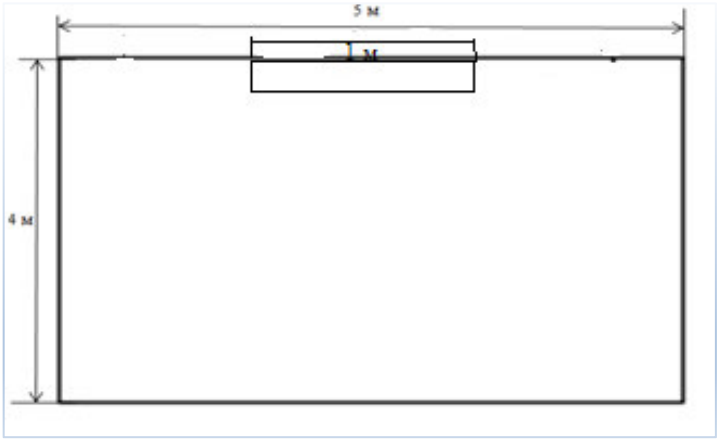
Бір жақты бүйірлік жарықтандыру қарастырылған. Терезе ойықтарының биіктігі 2 м құрайды.

Осылайша, терезенің нақты ауданы есептелген терезенің ауданы аз болды, бұл жұмыс бөлмесіндегі табиғи жарықтандырудың жеткіліксіздігін көрсетеді.

Енді мәндерді (5.1) формулаға қойып, табиғи жарықтандыру коэффициентінің ең аз мөлшерін e_{min} табамыз

$$e_{min} = \frac{100 * 1 * 2 * 0{,}54 * 1{,}1}{35 * 9{,}5 * 1{,}7 * 1{,}2} = 0.175$$

Табиғи жарықтандыру коэффициентінің ең төменгі шамасы КЕО e_{min} -нің нормативтік мәнінен аз болды, сонымен қатар осы жұмыс орны үшін табиғи жарықтандырудың жеткіліксіздігі туралы қорытынды жасауға болады. Бұдан әрі жасанды жарықтандыру есебіне көшеміз.



5.1-сурет – Табиғи жарықтандыру кезіндегі бөлме схемасы

5.4 Жасанды жарықтандыру есебі

Көру жұмысының дәрежесі I (а), сондықтан нормаланған жарықтандыру кесте бойынша – 500 лк.

ПВЛМ типті светильникті таңдаймыз, төрт шамдармен 65 Вт жарық ағынымен 3570 лм есептелген мөлшерден жалпы жарықтандыруы бар.

Жасанды жарықтандырудың қолданыстағы жүйесін бағалау өндірістік бөлменің $E_ф$ нақты жарықтандырылуын осы бөлмеде көру жұмысын орындау үшін қажетті $E_н$ нормативтік жарықтандырумен салыстыру жолымен жүргізіледі. $E_ф$ есебі мынадай формула бойынша жүргізіледі:

$$E_ф = \frac{Ф*N*n*\eta}{S_n*z*К_з} \quad (5.3)$$

мұндағы, Ф – бір шамның жарық ағыны, Ф = 3570 лм ПВЛМ типті;

N – бөлмедегі шамдардың саны;

n – бір шамдағы шамдардың саны;

η - жарық ағынын пайдалану коэффициенті, % (шам мен шамның типтеріне, қабырғалардың, төбенің шағылысу коэффициенттеріне және үй-жай индексіне байланысты анықталады), ПВЛМ типті шам үшін η = 0.42 табамыз;

S_n – бөлме ауданы, м^2;

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

z – ең аз жарықтандыру коэффициенті,

z=1,1 (люминесцентті шамдар үшін z = 1,1, қыздыру шамдары үшін z = 1,15);

$К_з$ – қор коэффициенті, $К_з$ =1,3.

Мәндерді (5.3) формулаға қойып, аламыз

$$E_ф = \frac{3750*2*2*0,42}{35*1,1*1,3} = 125.87 \text{ лк}$$

Өндірістік үй-жайдың $E_ф$ нақты жарықтандыруы Нормативтік жарықтандырудан аз болды, сондықтан бөлмені қайта жаңартуды жүргіземіз, осылайша бөлмедегі шамдардың санын көбейтеміз.

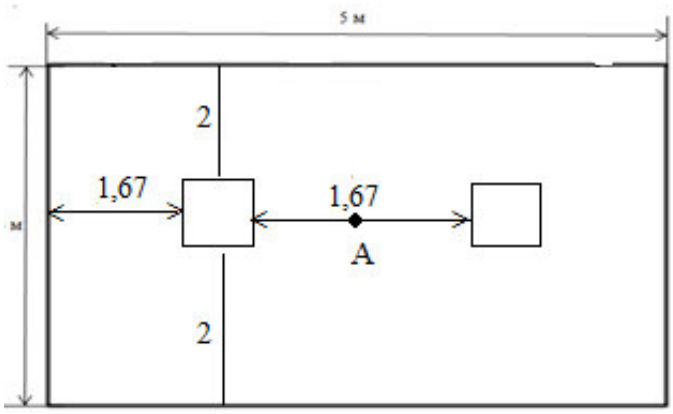
Есептеу коэффициент әдісі бойынша жүргізіледі. Бөлмеге қажетті нормаланған жалпы жарықты жасау үшін шамдардың қажетті саны мынадай формула бойынша анықталады (5.4):

$$N = \frac{E_{min}*S*z*K_з}{Ф*\eta} \quad (5.4),$$

мұндағы, E_{min} = 500 лк I, а разрядты көру жұмысы;

Осылайша, шамдардың саны тең:

$$N = \frac{500 * 35 * 1,1 * 1.3}{4 * 3750 * 0,42} \approx 4 \text{ шт.}$$



5.3-сурет – Бөлмедегі шамдардың орналасуы

5.5 Жасанды жарықтандыруды нүктелік әдіспен есептеу

Шамдарды үй-жайда орналастыру келесі параметрлермен анықталады:
H – үй-жайдың биіктігі;
h_c – шамдардың жабыннан қашықтығы;
$h_{рп}$ – еден үстіндегі жұмыс бетінің биіктігі.
Аспаның есептік биіктігін анықтау:

$$hрасч = H - (hрп + hc)$$

$$hрасч = 3 - (0{,}75 + 1{,}67) = 0{,}58 \text{ м}$$

Шамдар арасындағы қашықтық:

$$LA, B = \lambda \cdot hрасч$$

$$\lambda = \frac{1}{2} = 0{,}5 \text{ м}$$

Сондықтан,
$L_A = 2{,}3$ м;
$L_B = 2{,}3$ м;
$l_A = 1{,}67$ м;
$l_B = 1{,}67$ м.

Ол үшін барлық шамдардың жиынтық шартты жарығын келесідей анықтаймыз:

А нүктесінен d_i-ге дейінгі төбеге қашықтық проекциясын табамыз. Содан кейін төбенің және тік d_i арасындағы бұрышты анықтаймыз. Бұл бұрышта шартты жарықтандыруды табамыз. Шарт орындалатынын тексереміз:

$$E_г \geq E_н$$

$$\text{Е}_\text{г} = \frac{\mu * F_\text{л}}{1000 * \text{К}_\text{З}} * \sum \text{Е}_\text{г} \quad (5.5)$$

Тең жойылған шамдардың әсерін ескеретін коэффициент: μ = 1,15;

$$e_{\Gamma i} = \frac{\text{I}_{\alpha_i} * cos^3(\alpha_i)}{h^2}, \quad (5.6)$$

$$\alpha_i = arctg(\frac{d_i}{h}), \quad (5.7)$$

Егер шамдардың орналасу сұлбасын мұқият қарайтын болсақ, онда А нүктесінен шамға дейінгі қашықтық тең екенін байқауға болады.

Шам №1:

$$d_1 = \sqrt{1{,}15^2} = 1{,}15 \text{ м}$$

$$\alpha_1 = arctg\left(\frac{1{,}15}{0{,}58}\right) = 63{,}035^o$$

$$I_{\alpha_1} = 127{,}2 \text{ кд}$$

$$e_1 = \frac{127{,}2 * cos^3(63{,}035)}{0{,}58^2} = 38{,}023 \text{ лк.}$$

$$\text{Е}_\text{г} = \frac{1{,}15 * 4 * 3750}{1000 * 1{,}3} * 38{,}023 = 504{,}5 \text{ лк.}$$

Көріп отырғанымыздай, шарт орындалады

$$\text{Е}_\text{г} > \text{Е}_{min.}$$

Бұл дегеніміз, өндірістік қойманы қолайлы жарықтандыру үшін бұрын есептелген 4 шам қажет.

5.6 Өміртіршілік қауіпсіздік бөлімі қорытынды

Еңбек шартына зер салып, мен жасанды жарықтандырудың кемшілігі бар деген қорытындыға келдім. Табиғи жарықтандыру коэффициентінің және ТЖК-нің нормативтік мәнінің коэффициентін салыстыра отырып, біз осы жұмыс орны үшін табиғи жарықтандырудың жеткіліктілігі туралы қорытынды жасадық. Бұдан әрі жасанды жарықтандыру есептелінді.

Өндірістік үй-жайдың $\text{Е}_\text{ф}$-нің нақты жарықтандыруы Нормативтік жарықтандырудан аз болғандықтан, біз бөлмені қайта жаңартамыз, осылайша бөлмедегі шамдардың санын көбейтеміз. Бөлмедегі қажетті нормаланған жалпы жарықты жасау үшін шамдардың қажетті саны 4 данаға тең.

Сонымен қатар, жұмыс орнындағы еңбек жағдайын жақсарту үшін жасанды жарықтандырудың жеткілікті мөлшері болу үшін тағы екі шамды қосу қажет.

Қорытынды

Дипломдық жобаны орындау кезінде келесі жұмыстар жасалынды:

– «Мұғалім студент көзімен» атты сауалнама жүргізетін жүйесін құруға арналған талаптары қарастырылды.

– бұл жұмыста жүйені құруға, оның дизайні және дизайн үлгілеріне, жүйелік жобалау әдістеріне, моделіне зерттеу және ғылымитехникалық талдау жүргізілді.

– жалпы теория жүйесіне негізделіп отырып бағдарламаны жобалау нысаны ретінде талдау жүргізілді.

– жүйені құру кезінде қолдануға заманауи техникалық құралдар және қосымшалар пайдаланылады: Ruby, JavaScript тілдері. Жүйенің дизайні жүйелік талдау және нысанға қатысты таңдалған моделдерді зерттеу негізінде құрылды.

Сонымен қатар, бұл бағдарламалық қамтамада PostgreSQL деректер қоры қолданылды. Дипломдық жобаның басты өзектілігі деректерді сақтау үшін блокчейн технологиясының принциптері қолданылуында болды. Осы технологияны қолдану ақрасында сауалнама нәтижелері бір бірімен байланысқан хештердің тізімі ретінде сақталынады. Сондықтан, егер сауалнама жауаптарына сырттай өзгерістер енгізгілсе, бұл қалған блоктардың осы өзгертілген дерекпен байланыспауына алып келеді. Ал, нәтиженің тексттік құрылымы сол сауалнаманы құрған қолданушының нәтижелер бетінде сақталынады.

Әдебиеттер тізімі

1 Алгазинов, Э. К. Анализ и компьютерное моделирование информационных процессов и систем / Э.К. Алгазинов, А.А. Сирота. - М.: Диалог-Мифи, 2016. - 416 с.

2 Архитектуры и топологии многопроцессорных вычислительных систем. Курс лекций / А.В. Богданов и др. - М.: Интернет-университет информационных технологий, 2014. - 176 с.

3 BLOCKBENCH: A Framework for Analyzing Private Blockchains. URL адрес: https://arxiv.org/abs/1703.04057

4 Brian F. Cooper Adam Silberstein Erwin Tam Raghu Ramakrishnan Russell Sears. Benchmarking cloud serving systems. — 2010 — P. 143–154.

5 К. Тапскотт / Технология блокчейн - то, что движет финансовой революцией сегодня; Эксмо - М., 2017. - 750 с.

6 Proof-of-stake. URL адрес https://en.wikipedia.org/wiki/Proof-of-stake

7 Дон Фернандес, Оби Путь Rails. Подробное руководство Дрешер Д. Основы блокчейна; ДМК Пресс - М., 2018. - 735 с.

8 Руководство по созданию приложений в среде Ruby on Rails / Оби Фернандес. - М.: Символ-плюс, 2009. - 768 с.

9 Bruce, A Tate Ruby on Rails – Up and Running / Bruce A Tate. - Москва: Мир, 2006. - 182 с.

10 Jeremy, McPeak JavaScript 24–Hour Trainer / Jeremy McPeak. - Москва: Гостехиздат, 2010. - 360 с.

11 Чаффер, Джонатан Изучаем jQuery 1.3. Эффективная веб-разработка на JavaScript / Джонатан Чаффер, Карл Шведберг. - М.: Символ-плюс, 2010. - 448 с.

12 Флэнаган, Д. JavaScript: подробное руководство / Д. Флэнаган. - М.: Символ, 2008. - 992 с.

13 Вагнер Р., Байк А. Энциклопедия JavaScript Киев: Bhv, 2001. – 400 с.

Дмитриева М.В. JavaScript. Быстрый старт – СПб.: БХВ – Петербург, 2002. – 150 с.

14 А. Леоненков. Самоучитель UML (2-е издание). – СПб.: БХВ – Петербург, 2004. – 215 с.

15 Аманбаев У.А. Экономика предприятия. – А.: Бастау, 2012. – 100 с.

16 Хакимжанов Т.Е. Сборник задач по охране труда и безопасности жизнедеятельности: Пособие для вузов. – Алматы: Эверо, 2007. – 274 с.

17 Ригс, Саймон Администрирование PostgreSQL 9. Книга рецептов / Саймон Ригс , Ханну Кросинг. - М.: ДМК Пресс, 2015. - 364 с.

18 Уорсли, Дж. PostgreSQL. Для профессионалов (+ CD) / Дж. Уорсли, Дж. Дрейк. - М.: СПб: Питер, 2002. - 496 с.

19 Киммел, Пол UML. Основы визуального анализа и проектирования / Пол Киммел. - М.: НТ Пресс, 2008. - 272 с.

20 Пайлон, Д. UML 2 для программистов / Д. Пайлон. - М.: Питер, 2012. - 198 с.