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: Медициналық орталықтарға арналған «Дәріхана» ақпараттық жүйесін құру

Мамандығы: 5В060200 – «Информатика»

Орындаған: Қарсыбай И.Б. Тобы: ИНФк-15-1
Ғылыми жетекші: аға оқытушы Тоғжанова К.О.

Кеңесшілер:
Экономикалық бөлім: аға оқытушы _____ С.К. Тулегенова
«3» 05 2019 ж.

Өміртіршілік қауіпсіздігі: т.ғ.д., аға оқытушы _____ Ш.Ш. Бекбасаров
«10» 03 2019 ж.

Есептеу техникасын қолдану: аға оқытушы _____ Ж.С. Айткулов
«13» 05 2019 ж.

Норма бақылаушы: аға оқытушы _____ К. Мукапил
«15» 05 2019 ж.

Сын-пікір беруші: т.ғ.к., доцент _____ Д.М. Ескендирова
«____» _____ 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 5В060200 – «Информатика»

Дипломдық жобаны орындауға берілген

ТАПСЫРМА

Білім алушы Қарсыбай Инабат Бауыржанқызы

Жобаның тақырыбы: Медициналық орталықтарға арналған «Дәріхана» ақпараттық жүйесін құру

2019 жылғы «01» наурыз № 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 медициналық орталықтарға арналған «Дәріхана» ақпараттық жүйесін құру барысында PHP скрипт тілі, HTML гипермәтіндік тілі, CSS каскадты стильді кестелер тілі, MySQL мәліметтер қоры және frontend пен backend-та жұмыс жасайтын объектілік бағытталған программалау тілі JavaScript тілін қолданамын.

Дипломдық жобада қарастырылған мәселелер тізімі немесе дипломдық жобаның қысқаша мазмұны:

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

Графикалық материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс): 13 кесте, 31 сурет ұсынылған.

Ұсынылатын негізгі әдебиеттер:

1 Кузнецов М., Симдянов И. Объектно-ориентированное программирование на PHP. – СПб.: БХВ-Петербург, 2007. – 267 с.

2 Кузнецов М., Симдянов И. MySQL на примерах. – СПб.: БХВ-Петербург, 2007. – 484 с.

3 Гаффин А. Путеводитель по глобальной компьютерной сети Internet. – М.: Артос, 2006. – 274 с.

4 Хокинс С. Администрирование Web-сервера Apache и руководство по электронной коммерции. – М.: Вильямс, 2001. – 376 с.

Дипломдық жобаның бөлімдеріне қатысты белгіленген кеңес берушілер

Бөлімдер	Кеңесшілер	Мерзімі	Қолы
Экономикалық бөлім	Тулегенова С.К.	3.05.2019 ж	
Өміртіршілік қауіпсіздігі	Бекбасаров Ш.Ш.	19.02.2019 – 10.03.2019	
Программалық қамтама	Айткулов Ж.С.	01.03.2019 – 30.04.2019	
Норма бақылау	Мукапил К.	04.04.2019 – 20.05.2019	

Дипломдық жобаны дайындау
КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Талдау бөлімі	15.01.19 – 31.01.19	Орындалды
Жобалау бөлімі	01.02.19 – 28.02.19	Орындалды
Жүзеге асыру және тестілеу бөлімі	01.03.19 – 20.03.19	Орындалды

Тапсырманың берілген күні «29» қазан 2018 ж.

Кафедра меңгерушісі _____ Т.С. Картбаев

Жобаның ғылыми жетекшісі _____ К.О. Тоғжанова

Тапсырманы орындауға алған
білім алушы _____ И.Б. Қарсыбай

Аңдатпа

Ұсынылып отырған дипломдық жобада «Дәріхана» ақпараттық жүйесінің бағдарламалық қамтамасын құру жүйесі көрсетілген.

Фармацевтикада дәрі-дәрмек сату есебін жүргізудің ыңғайлы құралдары бар программаны құру бұл дипломдық жобаның мақсаты болып табылады.

Түсініктеме жазбада программаға қойылатын талаптар анықталған, оның қызметі мен функциялары, кіріс және шығыс деректері сипатталған, деректер қорының жобалануының әдістемесі сипатталған, деректер қорының моделі құрылған.

Қосымшада программаның техникалық тапсырмасы және программа листингі көрсетілген.

Аннотация

В предложенном дипломом проекте представлена разработка программного обеспечения информационной системы «Аптека».

Целью данного дипломного проекта является создание программы с удобными средствами ведения лекарственных отчетов в фармацевтике.

В пояснительной записке определены требования к программе, ее назначение и функции, описана методика проектирования баз данных, построена модель базы данных.

В приложении представлены техническое задание и листинг программы.

Annotation

The proposed diploma project presents the development of software information system «Pharmacy».

The purpose of this diploma project is to create a program with convenient means of conducting drug reports in the pharmaceutics.

The explanatory note defines the requirements for the program, its purpose and functions, describes the methodology of database design, built a database model.

A requirement specification and program text are presented in an appendix.

Мазмұны

Кіріспе

Күнделікті өмірде жан-жағыңа қарасаң, ақпараттық технологиялар элементтерін әрбір жерден көруге болады. Мысалы, көшеде тұрған банкоматтар, дәріханаларда, дүкендерде сатушылар қолданатын компьютерлер, сол сияқты тағы басқа салалардағы ақпараттық технологиялардың көріністері. Келе-келе мұндай ақпараттық технологиялар адамзат шаруашылығының әр саласына ене бастады. Ал үлкен жетістік ол – ақпараттық жүйелер. Мәліметтер қорлары үлкен-үлкен талай шаршы метр алатын мұражайларды алмастырды. Бірақ бұл шешімдер тек жергілікті, локальды түрде ғана жауап бере алды. Үлкен жүйелерді автоматтандыратын қажеттілік пайда болды. Яғни, кішкене ғана процесті бейнелейтін емес, сол процестерді жиып, бір-бірімен байланыстырып, орталықтан басқарылатын жүйелер жасау керек болды. Мысалға, ондай жүйелердің ең жарқын көріністері: денсаулық саласында, саудада, теміржол қозғалысын автоматтандыруда, банктік жүйелерде және т.б.

Қазіргі заман адамы өз өмірін жаңа ақпараттық технологиялардың қолдануынсыз сипаттай алмайды. Сонымен қатар, қазіргі таңда әлемде таралған ақпараттар легі тым үлкен. Ақпаратқа түрлендіруші, анықтаушы қасиет тән. Информатика индустриясын құру және ақпараттық өнімнің тауарға айналуы қоғамда терең әлеуметтік өзгерістерге алып келеді. Ақпарат материалдық өндірістен әлеуметтік салаға дейінгі қоғамның барлық салаларын қамтиды.

Сонымен қатар іздеу жүйесі пайдаланудың ыңғайлылығымен ерекшелінеді. Соңғы кезде көптеген адамдар ақпараттың бәрін компьютердің мәліметтер қорында сақтауды дұрыс деп шешті. Оның көптеген себептері бар. Дүние жүзін айналып жатқан ақпараттар өте көп. Олар уақыттың өтуімен әрдайым ұлғайып отырады. Сол себепті кез-келген үлкен немесе шағын мекемелерде мәліметтерді басқару күрделі болып табылады. Бұл мәселені шешу үшін кейбір мекемелер сөрелерді қолданады. Бірақ бұл сөрелерден керекті мәліметті табуға көп уақыт керек, әрі ондай сөрелерде маңызды қағаздарды сақтау қауіпті болғандықтан, соңғы кезде көптеген мекемелердің қызметкерлері мәліметтер қордың қолданылуы сөрелерге қарағанда бір шама тиімді деп тапты. Ол сапалы сақтайды, үлкен көлемді мәліметтерді құрайды, белгілі бір жүйеге келтіреді, әрі іздестіру уақытын азайтады. Бүгінгі күннің өзінде мәліметтер қорынсыз қаржы, сауда және сол сияқты көптеген мекемелер жұмысын елестету мүмкін емес. Компьютердегі мәліметтер қоры өте тиімді болып табылады. Себебі, ол бізге өз ақпараттарымызды көп орын алатын сөрелердегі көптеген қағаз беттерінде сақтамай-ақ өзіміздің компьтерімізде ғана сақтауға мүмкіндік береді, әрі мұнда сақтау қауіпсіз болып табылады.

Жалпы деректерді өңдеу жүйесінің әкімшілігі дами отырып, автоматтандырылған ақпараттық жүйелерді басқаруға қарқынды көшуде.

Автоматтандырылған ақпараттық жүйелер – ЭЕМ мен адамның өзара

қарым-қатынасын қамтамасыз ететін, программалық-аппараттық құралдар жиынтығы. Ол келесі функцияларды іске асырады:

– ЭЕМ-ге ақпараттарды енгізу мүмкіншілігі;

– ЭЕМ-нен ақпаратты мониторға, принтерге және т.б. шығару.

Мамандандырылған деректер қорын басқару жүйесі медициналық, бухгалтерлік, банктік және тағы басқа жұмыстардағы деректерді басқару үшін құрылады.

Тапсырыспен жасалатын мәліметтер қорын басқару жүйесі тапсырыс берушінің максималды дәрежеде медицинаның жұмыс спецификациясын есепке ала отырып жасалынады. Қолданушыдан маманды білімді талап етпейді.

Жалпы осы берілген «Дәріхана» ақпараттық жүйесінің құндылығы, оның сақтау және өзгерту мүмкіншіліктерімен анықталады, сонымен қатар, уақыт өте келе жаңа ақпараттарды енгізуге болады және құрылымын өзгертіп, web-қосымша ретінде көрсетуге болады.

1 Талдау бөлімі

1.1 Ақпараттарды автоматтандыру жүйелері

Ғылыми прогресс – білімнің жоғарлауымен және ақпарат көлемінің көп болуымен сипатталады. Мұндай кең көлемдегі ақпараттарды басқару көп қиындық әкелері сөзсіз. XXI ғасыр жаңа технологияның пайда болуы, оның жедел өсуі мұндай мәселелерді шешуге жол ашты. Оны шешудің ең тиімді жолы – автоматтандыру. Қазіргі кезде біздің елімізде, шет елдерде де ақпараттарды автоматтандыру кеңінен қолданылады, оларды пайдаланылу салалары өте көп. Мұндай күрделі ақпараттарды автоматтандыру жүйелерді реттеу функциясын және алдыға қойылған тапсырманы шешу әдісін талдау қызметін атқарады.

Автоматтандыру – техникалық құрал-жабдықтарды, сондай-ақ энергияны, материалды және ақпаратты алу, түрлендіру, жеткізу, пайдалану процестеріне адамның тікелей не ішінара қатысуын босататын экономикалық материалдық тәсілдермен басқару жүйесін пайдалану. Онда [1]:

– технологиялық, энергетикалық, көліктік, тұрмыстық-өндірістік процестер;

– күрделі агрегаттарды, кемелерді, ғарыш кемелерін, өндірістік құрылыстар мен кешендерді жобалау;

– цехты, мекемені, сондай-ақ медицина саласы мен бөлімшелерді ұйымдастыру, жоспарлау және басқару;

– ғылыми зерттеулер, медициналық және техникалық диагностикалау сапасын жақсарту, статистика деректерін өндеу және есепке алу, бағдарламалау, инженерлік есептеулердің барлығы автоматтандырылады.

Жалпы автоматтандыру адамды материалдар, ақпараттарды жеткізу, өңдеу, сақтау, пайдалану процестерінен босату мүмкіндігіне ие.

Автоматтандырудың арқасында адамдар үлкен фармацевтикалық фирмалардағы тауарларды, мекемелердегі құжаттарды, банкідегі есептеулерді, теміржол жүйесін басқаруды, студенттердің мәліметін, сессия қорытындыларын және тағы басқа жүйелерді басқара алады.

Автоматтандырудың мақсаты – еңбек өнімділігі мен өнім сапасын арттыру, жоспарлау, басқару жұмыстарын тиімді етіп орындау, денсаулыққа зиянды жұмыстардан адамды босату.

Автоматтандыру – ғылыми-техникалық прогрестің басты бір бағыты.

Автоматтандырудың екінші бір пайдасы: уақытты үнемдейді және де басқаруды жеңілдетеді. Үлкен кешенді орындарды қолмен басқарудан гөрі оны автоматтандырған әлде қайда жеңіл.

Автоматтандыру дәрежесі – автоматтандыру құралдарын пайдаланып орындалатын операциялар мөлшерінің осы процесс операцияларының толық мөлшеріне қатынасымен анықталатын технологиялық процестің сипаттамасы [1].

1.2 Автоматтандырылған басқару жүйесі

Автоматтандырылған басқару жүйесі немесе АБЖ – ақпараттық және программалық құралдардың кешені, технологиялық процесс рамкаларында, өндірісте, кәсіпорында әртүрлі процесстерді басқаруға арналған. Автоматтандырылған басқару жүйесі әртүрлі өнеркәсіп салаларында энергетикада, көлікте және т.б. қолданылады.

Технологиялық процессті автоматтандырылған басқару жүйесі немесе АБЖ ТП – дәріхана саласында, өнеркәсіпте, энергетикада, көлікте техникалық объектілерді шапшаң басқару және бақылау мақсаттарын шешеді.

Өндірістік автоматтандырылған басқару жүйесі (ӨАБЖ) негізгі өндіріс процесстерімен қоса өндірісті ұйымдастыру мақсаттарын шешеді, кіру және шығу логистикасын жүзеге асырады. Өндіріс қуаттылықтарын есептей отыра, шығарудың қысқа мерзімді жоспарлауын, өнім сапа талдауын, өндіріс процесс үлгілеуін іске асырады [2].

Дәріхана қызметтердің қызметкерлерінің автоматты басқару жүйесінің элементтерін, болашақта олардың дамуын және қолдануын басқару қызметін қарастырамыз.

ЭЕМ-нің негізгі құрылғысы микропроцессор, бағдарламадағы әртүрлі операциялардың орындалуын қамтамасыз етеді. Осы уақытта 32-разрядты микропроцессорлар ең көп таралды, бірақ жақында олардың орнына 64-разрядты микропроцессорлар келеді. Разрядтық екілік кодта жұмысшы сөздің ұзындығын білдіреді. Микропроцессорлар жұмыс істейтін ырғақты жиіліктеріне қарай айырылады. Ырғақты жиілік және разрядтық неғұрлым көбірек болса, соғырлұм процессордың өнімділігі жоғарырақ. Сыртқы есте сақтайтын құрылғының (ВЗУ) әр түрлі үлгілері болады. Ленталық жинақтағыштар магниттік таспада ақпарат сақтау үшін қызмет етеді. Осы шақта бірнеше жүз гигабайт (1 Гб = 1024 Мб) ақпарат сақтай алады. Бұл құралдар әлдеқашан пайда болғанына қарамастан олар осы уақытқа дейін кең көп таралған, тап осы үлкен сыйымдылығына байланысты, резервті көшіріп алуға арналған және ақпаратты ұзақ сақтау үшін қолданылады. Диск тұлғалы жинақтағыштар осы шақта ең кең көп таралған.

Жиналған тәжірибелерді салыстырып қарасақ, автоматтандырылған басқару жүйелері келесі талаптарға жауап беруге тиісті деп айтады [2]:

– маманның ақпараттық және есептеуіш қажеттілігін дер кезінде қанағаттандыруы;

– пайдаланушының сауалдарына жауап ең аз уақытта берілуі;

– пайдаланушының дайындық деңгейіне және оның кәсіпшілік сауалдарына бейімделуі;

– АЖО-да жұмыс істеудің игеру қарапайымдылығы және қатынас жеңілдігі, жұмыс істеу сенімділігі және қызмет ету қарапайымдылығы;

– пайдаланушыға шыдамдылық;

– пайдаланушының тез үйрену мүмкіндігі;

– желі құрамында жұмыс мүмкіндігі.

1.3 Ақпарат жүйелерінің және мәліметтер қорының жалпы сипаттамасы

Қазіргі кезде ақпараттың маңыздылығы адам қоғамындағы шикізат, күш, азық қорларынан кем түспейтін негізгі ұғымдардың біріне айналып кетті. Адамзаттың кез келген іс-шараларының ақпараттық сұранысын әр деңгейде қанағаттандыру қажеттілігі болады деп тұжырымдауға болады. Мысалы, айта кетер болсақ, көшеге шықпас бұрын біз ауа-райы жайлы ақпарат алуға тырысамыз. Біздің барлығымыз ғылыми ақпарат таратушыларды айтпағанда, күніге газеттен, радиодан, телеарналардан, ғаламтордан түрлі ақпарат аламыз. Адам қоғамында ақпараттық жүйенің дамуы ақпарат үрдісінің және байланыс арналарының да дамуына себепкер болды. Ақпарат жинақтау және оның шектеулігінің нәтижесі тек адам қоғамындағы анықтауыш фактор ретінде ғана емес, сонымен қатар адамның өмір сүру қабілеттілігін арттыруда. Ақпарат көлемінің артуы адам өмірімен салыстыра отырып жалпы институттардың ашылуына (кітапхана, архивтеу, баспасөз, есептеу орталықтары және т.б.) және арнайы жүйелер (ғылыми-техникалық ақпараттандыру, анықтамалық қызмет, ақпарат таратушы бүкіл әлемдік компьютер желілерінің) пайда болуына себепкер болды. Есептеу техникасының және ақпараттық технологияның дамуы ақпаратты сақтауда, іздеуде, жекелей жаңа есептеу орталықтарының пайда болуына әкеп соқты. Мысалы, есептеу орталықтары адам өміріндегі табиғи газ, электро шам, су секілді маңызды ұғымдардың біріне айналуда.

Осылайша, ақпарат құндылығына және жобалау жүйесінің сенімділігін арттыруда ақпараттың электрондақ түрде болуының маңызы өте зор.

Ақпараттық жүйе (АЖ) ұғымын талқыламас бұрын алдымен ақпарат сөзінің мағынасын біліп алғанымыз жөн. Ақпарат (лат. informatio − мазмұн, талқылау, жеткізу) − мәтінге байланысты өте кең мағыналы абстрактілі түсініктеме. Бұл сөзді тар мағынада алғанда – белгілі бір формаға байланыссыз тәуелсіз хабарлама беру деген мағынада. (Стратонович Р. Л. Теория информации. М.: Сов. радио, 1975).

Мәлімет (лат. калька data) — кейбір ақпараттық үрдісте алмасуға, өңдеуге жарайтын деректер мен идеялардың белгілі бір түрде көрсетілуі. (Мәліметтерді ассоциациялаудың халықаралық жүйесі).

«Ақпарат» сөзінің айқын семантикалық шектеулігі жоқ, өте кең мағыналы ұғым. Алайда ақпаратты қалай қолдану керек екендігін айтуға әрқашан мүмкіндігіміз болады. Сондықтан да бұл сұрақ жүйелік талдаушылар мен АЖ жасаушыларды, ақпаратты қолданушыларды (негізгі тұтынушылар) бірдей қызықтырады [2].

Қолданушылар мен АЖ-ні жасаушылар көзқарастары бойынша ақпараттың маңызды бір қасиеті бар − ол өңдеуге жарайтын мәліметтердің бірлігі болып саналады. Әдетте ақпарат тұтынушыға мәліметтер түрінде келіп түседі, олар белгілі бір құрылымдағы және типтегі ақпараттан тұратын кестелер, графиктер, фильмдер, ауызша хабарлар болуы мүмкін. Осыған

байланысты ақпаратты жіберу, жинақтау, сақтау секілді белгілі бір формадағы құрылғылар ойлап табылды. Қазіргі уақытта «ақпарат» және «мәлімет» сөзі АЖ ғылымында синоним сөздер сияқты қолданысқа түсіп кетті, алайда олардың арасында айтарлықтай айырмашылықтары бар.

«Форма» (қалып) сөзін ескере отырып, тағы бір айта кететін жайт «адамзаттың» ақпарат құрамына байланысты − әр адамға қалай жету жолдарына да байланысты болады. Мәліметтер белгілі бір құжат арқылы топтастырылып берілуі мүмкін. Ал құжат болса өз кезегінде белгілі бір ішкі құрылымды болуы да болмауы да мүмкін. Мәлімет – дербес компьютердің экранындағы дисплейде бейнеленуі де мүмкін. Сондай-ақ құжат аудио немесе видеоформа түрінде болуы да мүмкін. АЖ құрай отырып, оны кім қолданады (жүйені) және не үшін құрылып жатқанын естен шығармау керек. АЖ ақпарат ұсынуда тек «достастықты» ғана анықтамай, сонымен қатар қолданушылар категориясын көрсетеді. АЖ белгілі бір нақты қолданушылар тобынан және белгілі бір проблемалық бағытталған топтардан құрылуы тиіс.

Воройский Ф.С. анықтайды: ақпарат мәліметтер секілді біршама мағынасы бар (интерпретация) жүйелер жиынтығының ұғымы деп анықтама беріп кетеді. Осыдан кейін ақпаратты кодтау, оны қайта ашу жолдарын интерпретациялау мәселелері қарастырылған [3].

Осы анықтамадан кейін АЖ процесінде келесідей үш негізгі түсініктер қалыптасты, олар: ақпарат-мәліметтер, мәліметтер-мәліметтер, мәліметтер-ақпарат. 1.1-суретте ақпарат ұғымының екі жақты анықтамалары көрсетілген, олар: функциялық және ұсынылмалы. Олардың біріншісі – ақпараттың қолданылу ортасын көрсетсе, ал екіншісі – қолданылу барысында алынған нәтижелерді көрсетеді [3].

АЖ құру барысында ақпаратты қолдайтын және қолданушыға (сол сәттегі) ыңғайлы болу жақтарын қарастыру керек.

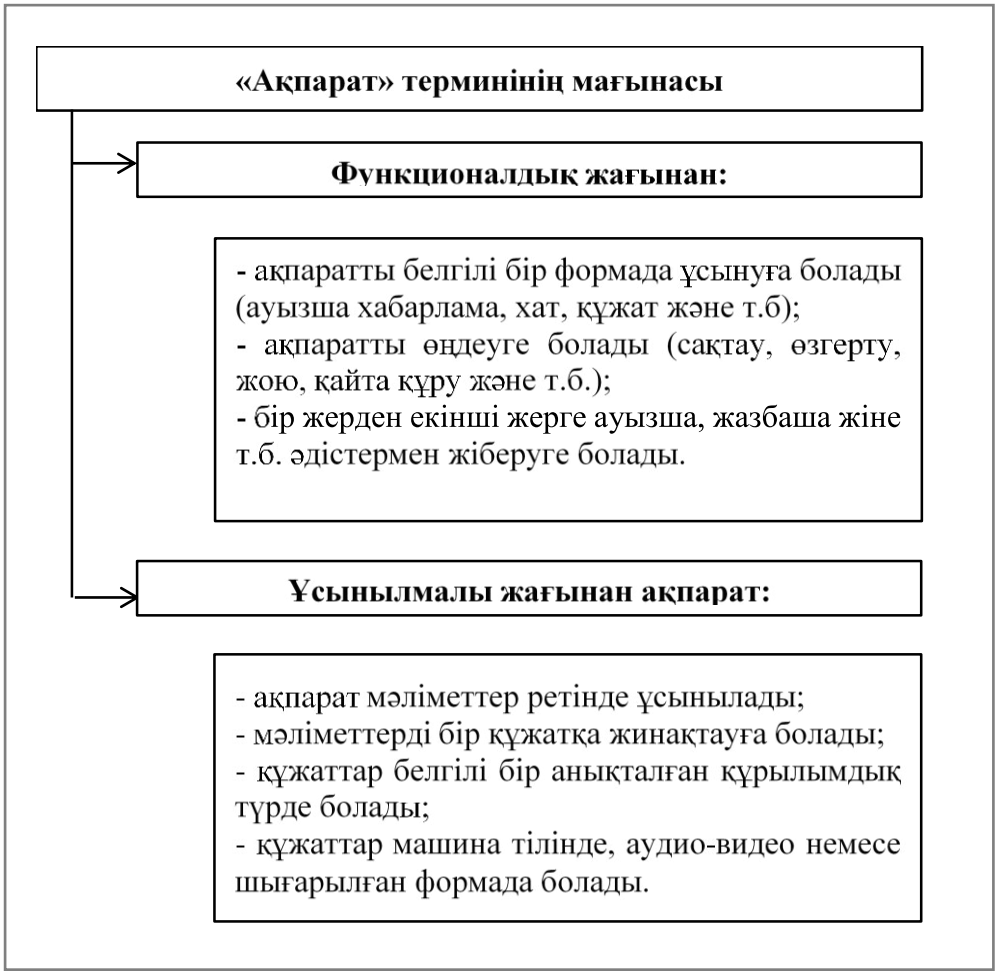
АЖ құру мақсаты – ол қолданушыларға керек ақпаратпен қамтамасыз ете отырып, шарттарын қанағаттандыру және мәліметтерді сақтап қалу. АЖ бүтіндей пайдалану ақпаратқа қол жеткізу ұғымдарымен тығыз байланысты.

Ақпараттану бағытындағы ғылымдардың жедел дамуы ақпаратқа тез қол жеткізудің тағы да бір бағыты анықтады, олар компьютер негізіндегі электрондық-техник, яғни сандық және аналогтық түрдегі құрылғылар. Қазірга кезде АЖ деп аталатын жүйеге, аппараттық және программалық құрамдас, технология және дербес компьютердің күрделі құрылымдары жатады. АЖ құрылымы, аппараттық (hardware), программалық (software), коммуникативтік (netware), аралық қатпар (middleware), лингвистикалық және ұйымдастырылған технологиялық қамтамасыз етулер атты құрамдас бөліктерден тұрады.

Аппараттық қамту ақпараттық жүйенің негізі болып табылады. АЖ аппараттық қамтамасыз етудің есептеуіш-техникадағы атқаратын қызметі өте күрделі және оның маңызы зор, мысалы АЖ ақпарат таратылуына сондай-ақ көптеген техникалық құрылғылармен жұмыс жасайды. (ақпараттың

графикалық бейне түрінде болуы, аудио немесе видео құрылғылар, ақпарат жіберуде және т.б.).

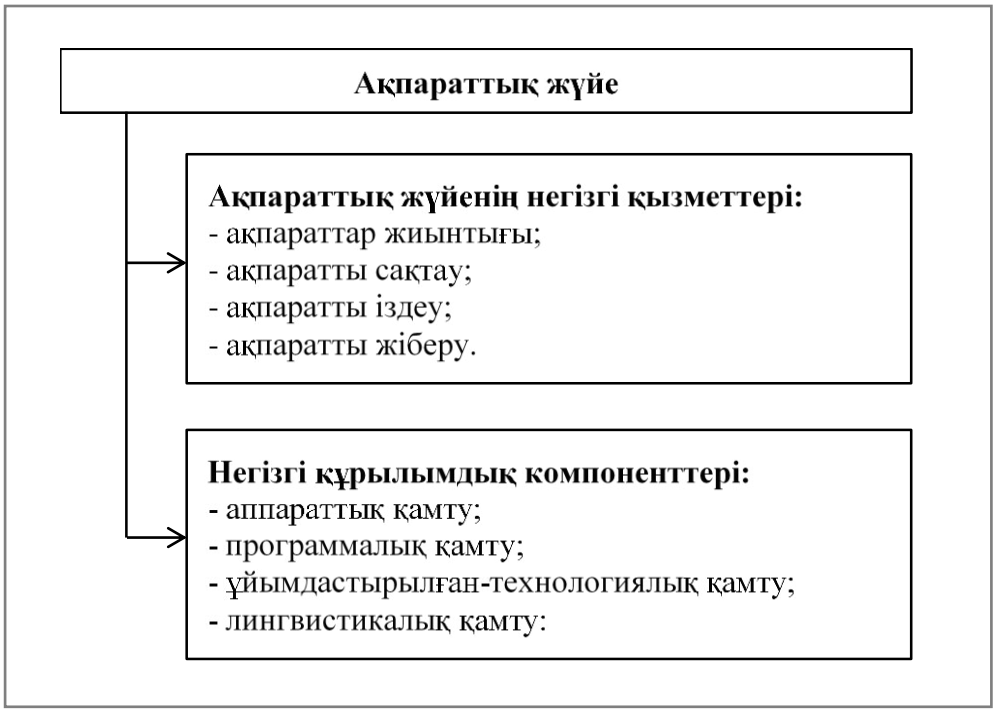
Коммуникативтік (желілік) қамту АЖ программаның және ақпараттың коммуникативтік, яғни желі арқылы жіберілуіне жауап береді. Оның АЖ және Интернеттегі болатын АЖ атқаратын қызметінің маңызы зор болып табылады. АЖ орналастыруда да маңызды рөл атқаратын программалық қамтамасыз етудің бір түрі – аралық қатпар (промежуточный слой). Бұл АЖ программалық қамту түрі қызмет және сервистік қарым-қатынасқа жауап береді. АЖ мәліметтерді енгізуде, оларды машина тіліне аударатын құрылғыларға орналастырылуы, мәліметтер модификациясына жауап беретін ақпараттық қамтамасыздандырудың бір түрі программалық қамту болып табылады. Программалық қамтуды жүйелік (аппараттық-программалық мәселелерді шешуде, қазіргі кезде платформа деп аталады) және қолданушы (компьютерлік ортада қолданушының бизнес-логикасын арттырады) деп екі түрге бөліп қарастыруға болады [3].



1.1-сурет – «Ақпарат» терминінің мазмұны

АЖ лингвистикалық қамту ақпараттың мағыналық мазмұнын анықтай отырып және арнайы мәліметтерді іздеуге қойылатын ақпараттық

қаматамасыз ету түрі (профиль). Мәліметтерді индекстеуде, және олардың классификациясы осы қамтудың классикалық мағынасы болып табылады. АЖ күрделі құрылымды болғандықтан, ақпараттарды бөліп қарастыру үшін термин сөздер (тезаурус) және бірнеше түсініктер енгізілген. Мысалы, қолданушыларға қарай әрбір программаны оқитын арнайы процессорлар шықты (бухгалтерлік ақпараттар және т.б.). АЖ масштабы мен күрделілігінің артуына байланысты түрлі компоненттерді қосып жұмыс жасайтын (аппаратура, программалар және дербес қолданушы) қамтамасыздандыруының бір түрі – технологиялық-ұйымдастырудың да маңызы артып отыр. Осы АЖ дұрыс бағаланбауы жүйені енгізу мен шығаруда және қуаттылығында өз кедергілерін келтіреді. 1.2-суретте АЖ негізгі құрамдас бөліктері мен қызметтері көрсетілген.



1.2-сурет – Ақпараттық жүйенің анықтамасы

Бизнес үрдістерді автоматтандырғанда ұйымның немесе мекеменің бағдары қандай болуы керек? Тапсырмаларды жеңіл және түсінікті шешетін дәстүрлік және ертеден пайдаланылып келе жатқан әдістердің бірі (ақпарат инфрақұрылымының бастапқы этаптарында) автоматтандарылуға арналған. Мысалы, берілгендерді тіркеу, есептеулерді жазу, құжаттарды дайындау. Осындай конъюнкциялық құраммен даярланған АТ-қызметтердің нәтижесі жылдам әрі нақты алынуы мүмкін. Бұрынғы көзқараспен системалық талдауды тәжірибе десек те қателеспейміз. Алайда, бұл автоматтандырылған жүйе шамасы келгенше ғана жұмыс жасайды (АЖ ұйымды бөліктеп қарастыру оның негізгі факторларын анықтайды), екіншіден автоматтандыруға қаржы көп кетпейді. Аты аталған тәсілдер квалификациясы

жоғары емес қызметтерге арналған. Ерте ме кеш пе ол, ақпараттық инфрақұрылымның дамуында тоқтап қалады [3].

АЖ қазіргі кезде қолданылып отырған түрлері де бір кездері тоқтайтын фактор болып табылады. Сондықтан, болашақта оны қалай құру керек немесе қайта басынан түбегейлі өзгертіп құру керек пе деген сұрақтар тұрады. Әрине, басынан құру әрқашанда дұрыс нұсқа болып табылады. Ақпараттану әдістемесіндегі жобалаудың "жоғары-төмен" немесе "төмен-жоғары" әдістерін пайдалану керек. Алайда, ерте ме кеш пе қазіргі заман талабына сай қолданулар ұсынылады.

АЖ құрушылар әрқашан "орталық" (midlle of design) әдісін пайдаланады. Осылайша АЖ құруда немесе қайта жасауда итерациялық әдіс пайдаланылады және осының негізінде дами береді. Сонымен, жоғарыда айтылғандарды ескерсек, АЖ құрушы теоретиктердің талғамымен емес барлығы тек заман талабына сай болуы керек екендігі анықталады [2].

АЖ құрудың тұжырымдамалық негізгі ерекшелігі үрдістердің белгілі бір бағытта болуы − яғни, конвейер болмаған жағдайда да, ақпаратты технологиялық үрдістердің бірге жинақталуы. Осылайша бірнеше технологияның бірігуінің нәтижесінде ақпараттық жүйенің жылдамдығы артады, ақпараттық үрдістердің бір бөлігі болып кетеді және басқару құрылымының иерархиясын азайтады.

АЖ жеке құраммен бөліктеп қарастыру қолданушылардың да сұранысын көбейтетінін ескере кету керек. АЖ − ақпаратты құрудағы пирамиданың бір бөлігі іспетті. Сондықтан да АЖ құруда және пайдалануда бизнес-үрдістерді қайта құруға тура келеді немесе бір негіздері реинжинирингтегі бизнес-ережелері бойынша жұмыс жасау керектігі көрсетіледі. АЖ ұзақ мерзімде қызмет жасауы және пайдасы болуы үшін, оның жобасын анықтап құру керек және дұрыс жобалау керек.

Ақпараттық жүйелерді құруда және автоматтандырудағы негізгі тәсілдерді қарастырайық. АЖ программалық қамтудағы негізгі сұрақтардың бірі, мәліметтер мен программалардың сәйкестігі немесе ақпарат құруды есептеу алгоритмі. Бұл сұрақтың іргелі шешімі − қолданбалы программа мәліметтерге тәуелсіз жұмыс жасайды деген тұжырымдамамен жауап береміз. Оның орталық немесе жайылмалы орналасқан болуы шарт емес. Бұл тұжырымдаманың маңызы программаны мәліметтерден алшақтату емес олардың әрқайсысы бір-бірінен тәуелсіз жұмыс жасауын анықтау [3].

Мәліметтерді құруда және есептеуде қолданбалы программаның көмегінсіз семантикалық мағынада модификациялау арқылы шешу принципі қолданылады. 1962 жылы программа мен мәліметтерді тәуелсіз шешу (есептеу) МҚ құруға және ақпаратты қайта құруға әкеп соқты. Өткен ғасырдың 60 жылдарына дейін программалық қамту жеке файл ретінде жүзеге асқан болатын немесе оны тапсырмалық әдіс деп те атаған. Өткен ғасырдың 80-ші жылдары МҚ бағытталған программа негізіндегі тұжырымдамалар ұсынылды [3]. 1.4-суретте жоғарыда аты аталған тұжырымдамалар

көрсетілген. 1.3-суретте файлдық жүйедегі тұжырымдама, мәліметтер қорындағы тұжырымдама және объектілі-бағытталған тұжырымдама мәліметтерді құрудағы негізгі әдістері көрсетілген.

Инкапсуляция. Объектілер құрылымды және ол операциялар жиынынан тұрады. Объектілерге орындалатын операциялар әдістерді құрайды. Объектіні, оның операциялары арқылы манипуляциялайтын қолданушыға объектінің құрылымы көрінбейді. Объект нақты өмірдің абстракциясы ретінде қарастырылады.

Мұрагерлеу. Бір объектілерді аталық қасиеттерін сақтай отырып, басқа объектілерден туындатуға мүмкіндік беретін механизм.

Полиморфизм. Әртүрлі объектілер бірдей хабарлама алып, өздеріне атаулас әдістердің жүзеге асырылуына сәйкес олардан әрқалай әсер алуы алуы мүмкін.

Мәліметтерді өңдеу әдістері:

Тапсырмалық әдіс

*Маңызы:*программаны өзіндік мәліметтер блоктары және алгоритмдерімен берілген ішкі тапсырмаларға бөлшектеу.
Қиындықтар:
- мәліметтердің көптігі;
- мәліметтер мен программа арасындағы өзара байланыстар;
- жүйенің тұйықтығы;
- мәліметтерді өңдеу схемасының бұзылуы.

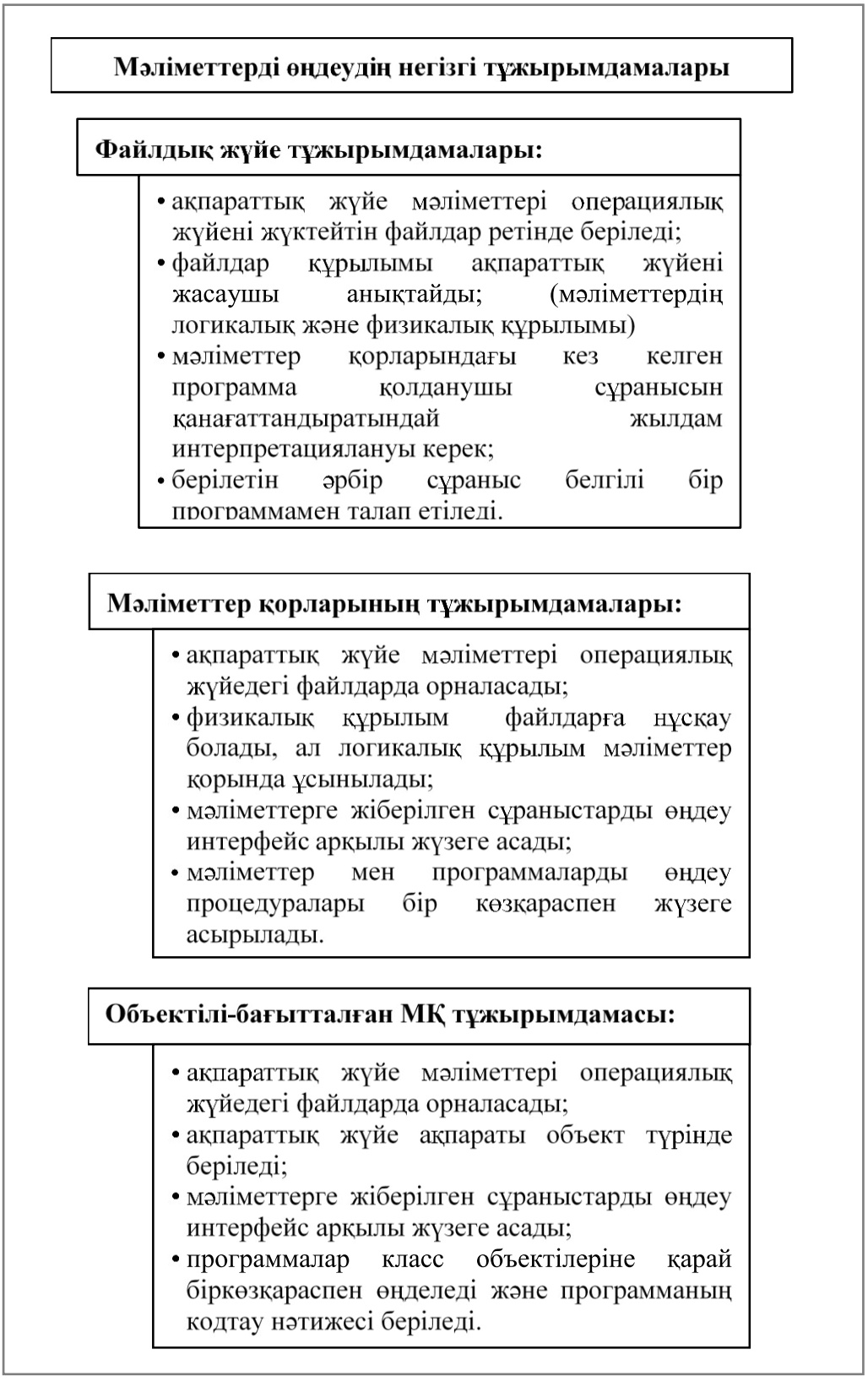
Мәлімет қорларының әдістері

*Маңызы:*мәліметтердің жеке логикалық құрылымын сипаттау, мәліметтеді өңдеу процедураларына бірдей көзқарас.
Қиындықтар:
- мәліметтерді көрсету мәселесі;
- персоналды оқыту мәселесі;
- реинжинирнг мәселесі.

Объектілі-бағытталған МҚ әдістері

*Маңызы:*ақпараттың(мәліметтер+ программалар) хабарламалар арқылы алмасатын (үрдістер қатынасы) объектілермен берілуі.
Қиындықтар:
- объектілерді (кластар) анықтау мәселесі;
- персоналды оқыту мәселесі;
- реинжинирнг мәселесі.

1.3-сурет – Ақпарат құрудың негізгі әдістері [3]



1.4-сурет – Ақпарат құрудың негізгі тұжырымдамалары [3]

Мәліметтерді сақтау программасының негізгі идеясы мәліметтер қорын құруда екі негізге бөліп қарастыру арқылы пайда болды, олар: логикалық бөлу және физикалық деңгейлеріне сай 1964 жылы IBM фирмасының қызметкерлері зерттеуінің нәтижесінде шықты.

Мәліметтер қоры ұйымының атқаратын қызметі өте маңызды. Алайда мәліметтер қоры деген тек бір ғана ұғымды бермейді, яғни ол бір негіздегі тұжырым ретінде қарастырылмайды. МҚ біртектес түсініктегі жанұялық құрам немесе программалық және аппараттық қамтамасыздандырулардың жиынтығы. Мәліметтер қорының бірнеше анықтамасы бар.

МҚ бірнеше анықтамасын қарастырайық [15]:

– мәліметтер қоры программамен байланысты емес, яғни программаға тәуелсіз жұмыс жасайды. Мәліметтерді құруда жалпы белгілі бір әдіс пайдаланылады. Егер мәліметтер қорының құрылымы бойынша қиылыспаған жағдайда, мәліметтер қорының жүйесі туралы айтылып кетеді;

– КОДАСИЛ мәліметтеріне сүйенсек, мәліметтер қоры дыбыстық көшірмелерден тұрады және олардың анықталған нақты схемасы көрсетіледі.

АЖ құрушы мәліметтер қорының концепцияларын пайдалана отырып, мәліметтерді белгілі бір анықтамаларға сәйкес топтастыра отырып құрады. Мұндай операцияларға "Қою" (Insert), "Қосу" (Add), "Өшіру" (Delete) және т.б жатады. Мәліметтердің декларативтік құрамына тілді анықтауды жатқызамыз. Ол МҚ мен қолданушы арасындағы қарым-қатынасты оқи алатын, АЖ концепция тілін түсінетін, сонымен қатар, программа интерфейсін дұрыс құратын болуы шарт.

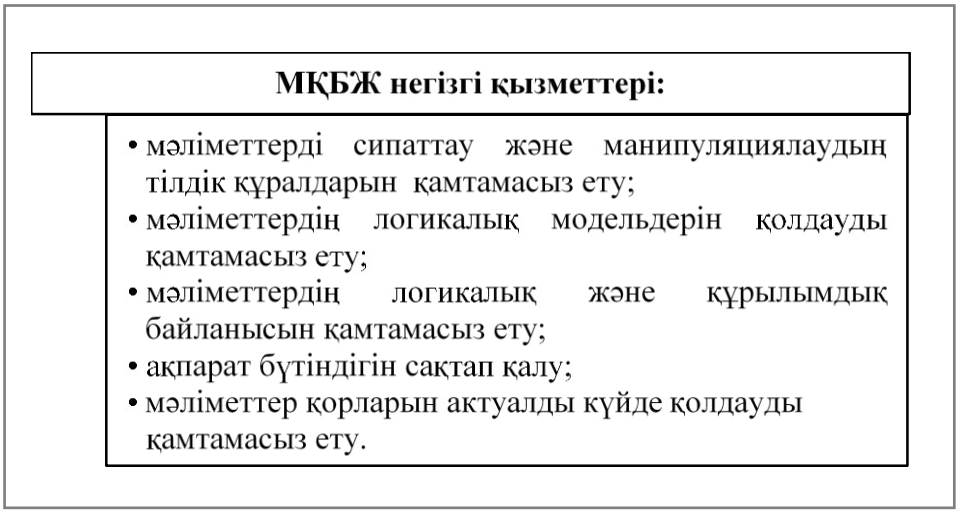
Осылайша мәліметтерді манипуляциялау шартты түрге енеді. Тіл мәліметтерге қол жеткізу үшін және оның логикалық құрылымын сипаттағанда МҚ маңызды рөл атқарады. Мәліметтердің тіл манипуляциясы МҚ мәлімет элменттерінде алгоритм ұсынады.

АЖ құруда мәліметтер қорының концепциясын пайдалана отырып – бұл жүйені кім немесе не қолдайды деген сұрақ тууы мүмкін. Осыдан кейін Мәліметтер қорын басқару жүйесі (МҚБЖ) ұғымы енгізіледі. МҚБЖ түрлі операциялық платформаларда жұмыс жасайтын күрделі құрылымды программа болып табылады.

МҚБЖ-ның негізгі қызметі – қолданбалы программалар мен қолданылған программаларды бір-бірінен тәуелсіз етіп, мәліметтерді манипуляциялайды. Кейінгі уақытта мәліметтер қорында мәліметтерді құру процестерін тұжырымдамалық түрде жүзеге асыратын машиналық құрылымдары шығуда.

Алғашқы МҚБЖ мәліметтер қорының тұжырымдамаларынан кейін 4 жылдан соң яғни 1966 жылы шыққан. 1.5-суретте МҚБЖ негізгі қызметтері көрсетілген.

Мәліметтер қорын басқару жүйесі (Data-base Management System) деп, МҚ-ын құруға қажетті программалардың жиынтығы деп білеміз [15].



1.5-сурет – МҚБЖ негізгі қызметтері [15]

Жоғарыда мәліметтермен жұмыс жасау керектігі, ережелері жайлы айтылып кетті (тәуелсіз программалар принципі, МҚ тұжырымдамалары және т.б.). Мәліметтер арқылы ақпаратты ұсыну объектіге бағытталуын талап етеді. Сондықтан АЖ құрушы үшін объектіні жобалаудың маңызы зор. МҚ моделін таңдай отырып, қандай ақпараттық технологияны қолдану керектігін нұсқайды.

Мәліметтер моделі МҚБЖ таңдау мүмкіндіктерін шектейді, өйткені жеке алынған модель нақты мәліметтер моделін қолдайды.

Мәліметтерді жобалау МҚБЖ қолданушы мен компьютер арасындағы қарым-қатынас тәсілдерін де анықтайды.

Осылайша, мәліметтер моделі түсінігі ақпараттық жүйенің программалық-аппараттық кешен ретінде жүзеге асырылу мехнизмі тәуелді болатын, ақпараттану пәнінің іргелі түсінігі болып табылады.

Мәліметтер моделі дегеніміз не? Жалпы алғанда мәліметтер моделі деп мәліметтердің логикалық түрде ұсынылуын және олармен жүргізілетін операцияларды айта аламыз.

Мәліметтер моделі (Data Model) компьютер функциясына байланысты емес, программалық-аппараттық қамтамасыздандыруға тәуелсіз логикалық құрылым болып табылады [15].

Мәліметтерді өңдеудегі модельдеудің бірнеше маңызды аспектілеріне тоқталып кетейік:

– ақпараттық модельдеу;

– концептуалдық модельдеу (тақырыптық аймақтың семантикасын модельдеу);

– мәліметтерді логикалық модельдеу;

– физикалық модельдеу;

– мәліметтерге берілетін рұқсатты модельдеу;

– аппараттық ортада мәліметтерді физикалық түрде орналастыруды оңтайландыру.

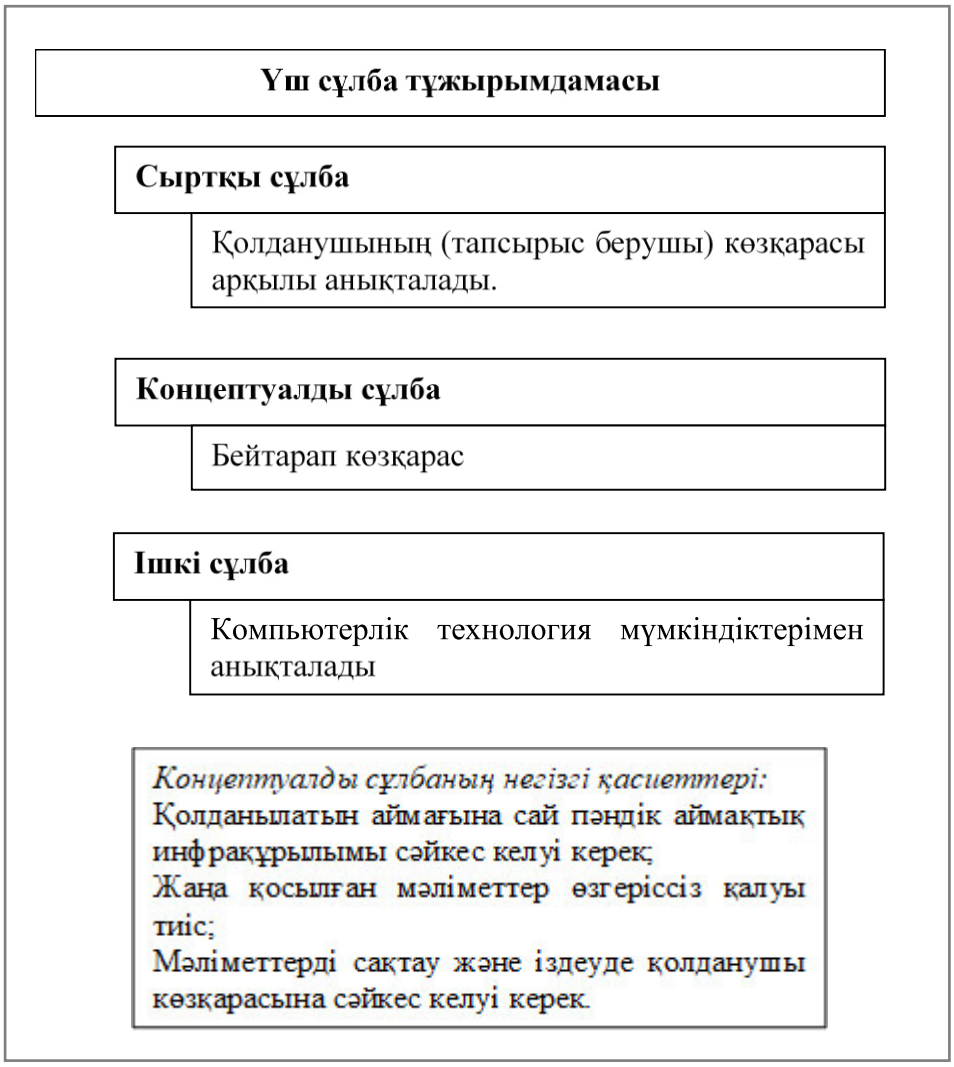
Мәліметтердің ақпараттық жобасын қарастырайық. 1.6-суретте қазіргі кездегі жобалау түрлерінің иллюстрациясы келтірілген.

1.6-сурет – Мәліметтердің ақпараттық моделінің көрсетілуі [15]

Ақпараттық жобалаудың объектісі шынайы өмірдегі пәндік аймағы болып табылады. Кейде олардың бүтіндігін сақтап қалу үшін оларды итема деп те атайды. Объект құрамын атрибуттар деп атайды. Олар осы атрибуттар арқылы бір-бірімен қарым-қатынас жасайды. Аты аталған үш компоненттердің негізінде АЖ МҚ сыртқы сұлбасын береді. Мәліметтер құрылымы және мазмұны қолданушының қолдану ортасына қарай өзгеріп отырады. Компьютер бойынша сыртқы сұлба файлдық құрылымды сақтау үшін және ақпарат іздеуде қолданылады. Бұндай жағдайда мәліметтер құрылымы компьютерлік технологияға байланысты және оны құру нәтижелілігіне байланысты негізделеді. Мәліметтерді ішкі және сыртқы сұлба деп қарастыра отырып, ақпаратты модельдеу мәліметтердің барлық мәселелерін шеше алмайды. Дегенмен, МҚБЖ мәліметтер қорын қолдануда көптеген мүмкіндіктерді ашып береді. МҚБЖ зерттеуші топ ANSI/X3/SPARC мәліметтермен анық жұмыс жасайтын орта үшінші жақты аралық көзқарас деген қорытындыға келді (үш сұлбалы тұжырым ANSI/X3/SPARC). Бұл көзқарас (концептуалды сұлба деп аталады) компьютерде белгілі бір

тақырыптық аймақ бойынша бағыт бағдары жоқ, жалпылай қолданылады ол 1.7-суретте көрсетілген [15].

Концептуалдық сұлбаның негізгі мақсаты мәліметтер арасындағы қарым-қатынастың бұзылмай, интерпретациялы бірігіп мәліметтердің бүтіндігін сақтап қалу. Кез келген ақпараттың моделі МҚБЖ арқылы анықталады [21]. МҚБЖ-де мәліметтер құрылымының анықталуы құрылымдалған мәліметтер қорының түсінігіне алып келеді, яғни мұндай МҚ-дағы мәліметтер өзара байланыстағы элементтер жиынтығы түрінде көрсетілуі керек. Егер жаңа типтер мен байланыстарды орнатудың динамикалық үрдісі мүмкіндіктері пайда болса, онда құрылымдалмаған мәліметтер қорының түсінігіне келеміз. Сонымен қатар, детерминделген сұлбалы МҚ деп аталатын аралық нұсқаларда болуы мүмкін. Мәліметтер қорын осылай бөліп қарастыру ақпараттық жүйені жүзеге асыру үшін МҚБЖ-сін таңдау кезінде өте маңызды сәт болып саналады, өйткені нақты МҚБЖ әдетте нақты мәліметтер моделін қолдайды. Басқаша жағынан алғанда, мәліметтер қорының әрбір түрі үшін оларға сәйкес мәліметтер моделі қолданылады.



1.7-сурет – Үш сұлба тұжырымдамасы [15]

Қазіргі кезде мәліметтердің құрылымдалған қорлары үшін, мәліметтер элементтері арасындағы олар қолдайтын байланыстардың сипатына қарай, мәліметтердің логикалық модельдерінің негізгі үш түрін ажыратады: желілік, иерархиялық және реляциялық. Бұл модельдердегі жіктелу белгілері: байланыстың қаттылық (фиксация) деңгейі, модель құрылымының математикалық түрде көрсетілуі және рұқсат етілетін мәліметтердің типтері 1.1-кестеде көрсетілген.

1.1-кесте − Мәліметтер модельдерінің жалпы сипаттамасы

Мәліметтер моделі	Объектілер арасындағы байланыс сипаттамасы	Формальды көрінісі
Желілік	Жартылай қуатты байланыс	Өзгермелі граф
Иерархиялық	Қуатты байланыс	Ағаш тәрізді құрылым
Реляциялық	Ауыспалы байланыс	Біркелкі файл

1.8-суретте әр модельдің ерекшелігін бейнелейді. Модельдерді сәйкестендіріп қарағанда оның барлығы теориялық тұрғыда эквивалент екенін естен шығармау керек. Бұл фактінің дәлелдемесін Дж. Дейттің МҚ бойынша классикалық монографиясынан табуға болады. Бір модельдің екінші модельге түрленуі «желілік-иерархиялық-реляциялық» модельдер тізбегінде төбелердің еселенуі арқылы көрсетумен алынады [19].

Көп жағдайда МҚБЖ олар қолдайтын мәліметтер модельдерінің типтері бойынша жіктеледі. Сондықтан МҚБЖ-ні желілік, иерархиялық және реляциялық түрлерге ажыратады. Дегенмен, мәліметтерді өңдеу тәжірибесінде МҚБЖ мәліметтер қорының нақты типін қолдау қабілетіне байланысты сипатталады. Жалпы жағдайда МҚ мына түрлерге бөлінеді:

– әртүрлі құжаттардың интегралданған деректер жиынтығын сақтайтын фактографиялық тип;

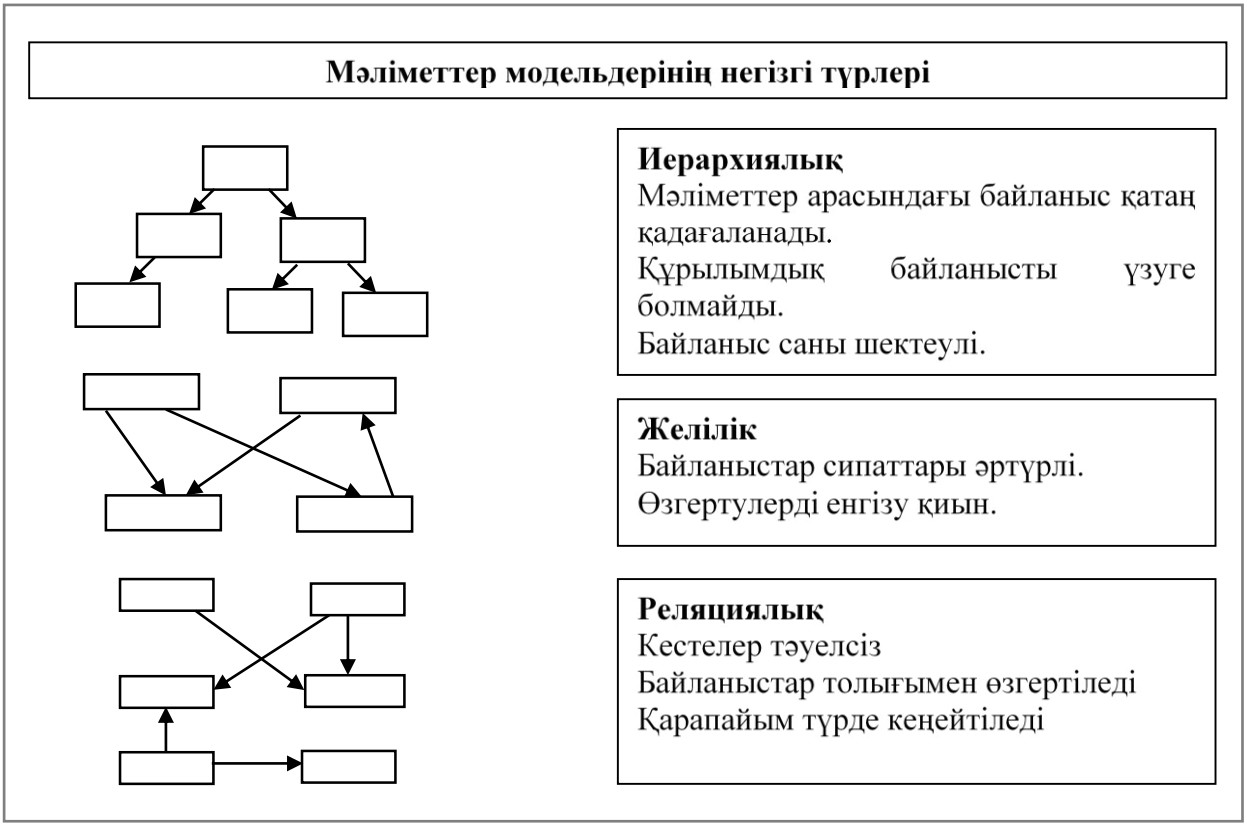
– құжаттарды сақтауға бағытталған құжаттамалық тип;

– екеуіне де сай қызмет көрсететін құжаттамалы-фактографиялық тип.

Сонымен, CDS/ISIS МҚБЖ түйін сөздер арқылы индекстелген құжаттармен жұмыс жасауға бағытталған. ADABAS МҚБЖ фактографикалық МҚ-мен жұмыс жасауда ыңғайлы, ал ORACLE атты МҚБЖ - мәліметтер қорының екі түрлі типіне де сай келеді. МҚ зерттеу жұмыстары әлі де аяқталған жоқ: МҚ енгізудің жаңа типтері де шығуда және жалғаса береді (белсенді, дедуктивті, реляцилық, МҚ графикалық түрі және т.б.).

Көптеген жағдайда АЖ құрушылыр үшін МҚБЖ сипатына қарай бөліктеп қарастыру ыңғайлы: орталықтандырылған және үлестірілген. Үлестіріп өңдеуді қолдану кезінде транзакциялардың өңделу сипатына көңіл аударған қажет, өйткені транзакция жүйенің өнімділігіне әсерін тигізеді. Жалпы жағдайда транзакция деп қолданушының мәліметтер қорынан талап

етіп отырған жұмыс бірлігінің өңдеу сипатынан тәуелсіз түрде берілуін атайды. Көбіне транзакция нәтижесінен кейін мәліметтер қорының басқа жаңарған түрі құрылады немесе МҚ басқа да іс-әрекеттерді орындауға тура келеді. Осыған орай МҚБЖ-дегі ақпараттың бүтіндігін қамтамасыз ететін, ақпаратқа қол жеткізуге шектеу қоятын және т.б. ішкі жүйелік іс-әрекеттер орындалады.



1.8-сурет – Мәліметтер модельдерінің негізгі түрлері [19]

Қайта құруды орнықтыруда транзакцияның көптеген әдіс тәсілдері бар. Болып жатқан транзакциялардың барлығы қай жерде жүйеленеді деген сұрақ туы да мүмкін: компьютер файлын қолданушыларында немесе компьютердің желілік бөлімінде деген жауап ала аламыз. Қолданушының сұранысына қарай уақыты да жүйелене береді. «Қолданушы сұранысына қарай уақыт» параметрі жүйелік жұмыс жасауда анықтамалық ретінде пайдаланылады. Ақпараттық жүйелердің архитектуралық классификациясы басқа да классификация түрлері секілді «аса қуатты» емес екенін ескеру керек.

1.4 Қолданылатын программалық қамтамаларды талдау

1.4.1 Программалық кешенді құру үшін PHP технологиясын пайдалану

PHP тілінің басты артықшылықтарының бірі – оның HTML кодына тура енгізілетіндігінде. Сондықтан программистке HTML-ды шығару үшін көптеген командалардан тұратын бағдарлама жазуының қажеті жоқ.

<html>
<title><? print ”Hello world!”;?></title>
</html>

“Hello world!” хабарламасы web-парақтың тақырыбында шығады. Қызығы, әдете PHP-дің экрандайтын тізбектері <?...?> деп аталатын print командасы аяқталған бағдарлама болып табылады. Инициализациялаудың ұзын кодтарын және кітапхананы қоспай, бағдарлама тек берілген есепті шешетін кодтан ғана тұрады. PHP сценарилерін орындау үшін PHP программалық қамтамасын серверде орнатып, күйге келтіру керек.

PHP Windows NT және Unix-тің көптеген версияларында жұмыс жасайды. Ол Apache-дегі модуль ретінде жіберіле береді. Егер жіберілу Apache модулі түрінде болса, PHP оңай және жылдам жұмыс жасайды. Бұл кезде процессті жасауға байланысты туындайтын қосымша шығындар болмайды. Сондықтан, нәтижесі тез шығады және сервердегі сақтаудағы шығынды азайтатын mod _perl-ді орнатудың қажеттілігі болмайды [4].

Құжат бетіндегі әртүрлі операциялардан басқа PHP көмегімен HTTP-тақырыптарды қалыптастырып, HTTP орнатуға болады. Сонымен қатар, аутентификацияны басқарып пайдаланушыны басқа бетке бағыттауға болады. PHP берілгендер қорына енуге үлкен мүмкіндіктер ашады. Бұл PDF құжаттарды инерациялаудан, XML-дегі грамматикалық талдауды жасауға мүмкіндік береді.

PHP операторлары бетке Web-беттерді қойып береді, сондықтан арнайы ортада жүзеге асырудың қажеттілігі туындамайды. PHP кодының блогы <?php тегінен басталып, ?> тегімен аяқталады. Бұл тегтердің арасындағылардың барлығы PHP код ретінде интерпритацияланады [5].

PHP тілінің синтаксисі Си және Perl синтаксисіне ұқсас келеді. Ауспалыларды оларды пайдаланудан бұрын жарияламау керек. Дегенмен, PHP Apache құрылған жағдайда бәрінен жылдам жұмыс жасайды. PHP Web-сайтында оны Microsoft IIS және Netscape Enterprise Server орналастырудың реті бар. Егер сізде PHP-ді орнатуға программалық қамтамасыз етудің көшірмесі жоқ болса, оны сіз ресми Web-сайттан ала аласыз. Ол жерден сіз PHP-дің барлық ерекшелігі мен қызмет бейнеленген, түсіндірілген жетекшілікті таба аласыз.

1.4.2 CSS (Cascading Style Sheets – каскадты стильдер кестесі)

CSS – бұл веб-дизайнерлер және пайдаланушылардың қажеттерін қанағаттандыру үшін арнайы әзірленген стиль кестелерінің механизмі. Стиль кестелері құжаттар экранда басылған кезде қалай бейнеленуін және қалай айтылуын мазмұндайды. 1994 жылы жарыққа шыққан уақытынан бері интернетте W3C стиль кестелерін пайдалануды белсенді түрде насихаттап

келеді. Style Activity тобы W3C-тың (CSS1, CSS2, XPath, XSLT) бірнеше ұсыныстарын берген болатын. CSS әсіресе браузерлерде кеңінен қолданылады [10].

Интернетте құрылымдалған құжаттарда (яғни, HTML) стиль кестелерін пайдаланып, авторлар мен оқырмандар құжатты өзгертпестен оның бейнеленуіне ықпал жасай алады немесе жаңа HTML элементтерін қоса алады. Стиль кестелерімен эксперимент жасаудың ең қарапайым жолы – бұл CSS-ті қолдайтын браузерді табу. CSS-тің бірегейлігі оны HTML және XML құжаттарды белгілеу үшін пайдалануға болады. Екінші жағынан XSL құжатты өзгертуі мүмкін. Мысалы, XSL веб серверде XML мәліметтерін HTML/CSS құжатына өзгерту үшін пайдаланылуы мүмкін. Бұл жағдайда тілдер бірін-бірі толықтырады және оларды бірге пайдалануға болады. Тілдердің екеуі де XML құжатын форматтау үшін пайдаланылуы мүмкін. CSS және XSL ұқсас пішімдеуді пайдаланады және сөйтіп, зерттеушілер тілдердің екеуінде де бірдей пішімдеу моделіне қолы жетімді болады. W3C пішімдеу модельдерін пайдалану мүмкіндігін қамтамасыз ету жөнінде жұмыс істейді.

CSS файлды «қолмен», яғни әр қандай мәтіндік редактормен құру және редакциялауға болады, бірақ сіз стиль кестелерінің файлдарымен жұмыс істеу үшін ECMAscript, Java немесе әр қандай басқа тілде бағдарлама жасасаңыз болады. Ал іс жүзінде сіз стиль кестелерімен жұмыс істеу үшін жақты софт және кітапханаларды пайдалануыңызға болады. Осындай бағдарламаларды және кітапханаларды әр түрлі компьютер платформаларына импорттауға көмек үшін W3C осындай кітапханалардың барлығында болуы тиіс функциялар жиынтығын белгіленген CSS-DOM деп аталған спецификацияны әзірледі [11].

CSS құжатының объектілік моделі –бұл CSS-ті (және белгілі бір өлшемде стильдердің басқа тілдерін) бағдарламадан редакциялау үшін API (бағдарламалаудың абстрактты интерфейсі). API – бұл программалық кітапхананың спецификациясы. Сіз бұны нұсқаумен салыстыруыңызға болады. Сізде барлық функциялардың сипаттауы және параметрлері болады, бірақ кодтың өзі болмайды.

Әр түрлі платформалар үшін CSS-DOM нің бірнеше кітапханалары қол жетімді. Олардың көпшілігі тегін. Көптеген браузерлер ECMAScript бағдарламаларды пайдалану үшін CSS-DOM-нің ішіне салынған кітапханаларына ие.

SAC (CSS үшін қарапайым API) – бұл CSS-DOM үшін қосымша. CSS-DOM-нің компьютердің жадына жүктелгеннен кейін стиль кестелерімен манипуляция жасауға арналған функциялары бар; SAP-тағы функциялар стиль кестелерін тәртіпке салуға, яғни стиль кестелерін файлдан жадқа көшіруге көмектеседі [10].

1.4.3 JavaScript тілінің негіздері

JavaScript – бұл объектіге және тілдік құралдарға негзделген, және орта

мүмкіндіктері объектімен ұсынылатын, программалау тілі, ал JavaScript сценарилер (программа) бұл өзараәрекеттенуші объектілер жинағы. JavaScript объектісі – бұл реттелмеген қасиеттер жинағы, оның әрбіреуінің ноль немесе одан көп атрибуттары бар, олар бұл қасиет қалай қолданылуы мүмкіндігін анықтайды. Мысалы, егер ReadOnly қасиетінің атрибутына true (шын) мәні меншіктелсе, онда осы қасиеттерді программалау арқылы өзгерту мүмкіндіктері нәтижесіз болады. Қасиет – бұл басқа объектілерден,көп қолданылатын мәндер мен әдістерден құралатын контейнерлер.Көп қолданылатын мәндер – бұл кез келген енгізілген Undefined, Null, Boolean, Number и String типтерінің элементі; объект− бұл енгізілген Object типінің тағы бір элементі; әдіс− қасиетпен арқылы объектпен ассоциированлған функция [11].

JavaScript, Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp сыйақты бірнеше енгізілген объектіден құралады. Бұдан басқа, JavaScript қатаң түрде айтқанда, міндетті түрде функция мен әдіс болып табылмайтын енгізілген операциялар жинағынан, сонымен қатар, программаның орындалу логикасын басқаратын енгізілген операторлардың жинағынан құралады.

JavaScript синтаксисі негізінен Java тілінің синтаксисіне сәйкес келеді, бірақ сценарилер тілін үйренуге жеңілтету үшін онымен салыстырғанда жеңілдетілген. Мысалы, айнымалылар декларациясы оның типінен құралмайды, қасиетте типсіз болады, ал функция декларациясы программа текстінде оны шақырғаннан кейін тұруына болады.

Объектілер жайында жалпы мағлұматтарJavaScript тілі JavaScript және C++ тілдерімен салыстырғанда қатаң түрде объектілер классынан тұрмайды. Оның орнына олар үшін объектілерді құруды жады бөлу жолымен жадыны конструкторларды қолдайды, ол Оның орнына ол объектілерді олар үшін жады бөлу және олардың барлық немесе кейбір қасиеттерін инициализациялау жолымен құратын конструкторларды қолдайды. Барлық конструкторлар объектілер болып табылады, бірақ бар объектілер конструктор болмайды. Әрбір конструктор, протатипті және бөлінетін қасиеттерге негізделген ізін басуды іске асыру үшін қолданылатын prototype қасиетін иеленеді. Объектілер new операциясында конструкторларды шақыру жолымен құрылады; мысалы new String (жаңа жол) String жаңа объектісін құрады. New−сыз конструкторды шақыру нәтижесі конструкторға тәуелді. Сонымен, String("бұл жол") , объект емес көп қолданылатын жолды құрады [11].

JavaScript протатипке негізделген, із басуды қолдайды. Әрбір конструктормен сәйкес протатип байланысқан, және конструктормен құрылған әрбір объект осы прототипке сілтемені құрайды (объектінің порототипі деп аталынатын). Прототип өз кезегінде, өз прототипіне және ары қарай сілтеме құруы мүмкін. Осылайша прототиптер тізбегі құрылады. Объектінің қасиетіне сілтеме− бұл берілген атпен қасиетті құрайтын объектілер прототипінің тізбегінің бірінші прототипіне сілтеме. Басқа сөзбен айтқанда, егер берілген объектінің берілген атпен қасиеті болса, онда осы

қасиетке сілтеме қолданылады, егер жоқ болса, онда осы объектінің қасиеті зерттеледі және т.б..

Объектілер классына негізделген, объектке бағытталған тілдерде, ағымдағы қалып класстар экземплярларымен іске асырылады. әдістер класстармен, ал із басушылық құраммен және мінез құлқымен . JavaScript та ағымдағы қалып және әдістер объектімен іске асырылады, ал құрамы мен мінез құлқы із басады. Барлық объектілер, олардың прототипінен құралатын, қасиеттерден құралады, осы қасиеттерді және оның тағайындалуын бөледі. Класстарға негізделген тілдерден айырмашылығы, қасиеттер объектіге оларға мәндерді меншіктеужолымен динамикалық түрде қосылуы мүмкін. Жеке жағдайда, конструкторлар мәндерді барлығына немесе құрылатын объектінің кейбіркасиеттеріне меншіктеуге міндетті емес.

JavaScript-ті қысқаша түрде сипаттағаннан кейін, оның негізгі түсініктерінің формалды емес анықтамаларын берейік [9].

Тип – берілген деректер мәндерінің жиынтығы.

Примитивті мән – Undefined, Null, Boolean немесе String типтерінің біреуінің элементі. Примитивті мән, бұл – тіл реализациясының (жүзеге асуының) ең төменгі деңгейінде берілетін деректер.

Объект – Object типінің элементі; әрқайсысы примитивті мән, объект немесе функция бола алатын қасиеттердің ретсіз жиынтығы болып табылады. Функция болып табылатын қасиет әдіс деп аталады.

Конструктор – объектілерді жасап, инициализациялайтын функция. Әрбір конструктор, қасиеттерді мұрағаттап, бөлу үшін қолданылатын сәйкес прототипке ие болады.

Прототип – құрылымды, жағдайды және мінез-құлықты мұрағаттаушы жүзеге астыру үшін JavaScript-та қолданылатын объект. Конструктор объектіні жасайтын кезде, объектіге құрылымның прототипіне апаратын нақсыз сілтеме болады, бұл сілтеме осы объектінің қасиеттеріне апаратын сілтемелерге рұқсат беру мүмкіндігін ұсынады. Прототиптің қасиеті, өзінің негізінде жасалған барлық объектілер тарапынан бөлінеді.

Тіл объектісі – сценарийлердің орындалу аясы тарапынан емес, JavaScript тілінің реализациясы тарапынан жүзеге асатын кез келген объект. Тіл объектілерінің бір бөлігі іштей құрылған болып табылады да, басқа бөлігі сценарий орындалу кезеңінде жасалады.

Іштей құрылған объект – орындалу аясына тәуелсіз түрде JavaScript тілінің реализациясы тарапынан жүзеге асатын кез келген объект. Барлық іштей құрылған объектілер тіл объектісі болып табылады.

Аяның объектісі – тілдің объектісі болып табылмайтын және сценарилердің орындалу аясы тарапынан жүзеге асатын кез келген объект.

Белгісіз мән – айнымалыға ешқандай мән берілмегенін білдіретін примитивтік мән.

Undefined типі – белгісіз болып табылатын жалғыз undefined мәнінен тұратын тип.

Нөлдік мән – нөлдік, бос немесе мүлдем жоқ болған сілтемені білдіретін

примитивтік мән.

Null типі – нөлдік болып табылатын жалғыз null мәнінен тұратын тип.

Логикалық мағына – шындықты немесе өтірікті білдіретін Boolean типінің примитивті мәні.

Boolean типі – нақты екі мәннен: true (шындық) және false (өтірік) тұратын тип.

Логикалық объект – іштей құрылған объектінің көшірмесі болып табылатын Object типінің элементі. Басқаша айтқанда, логикалық объект new Boolean (value) өрнегімен жасалады, мұндағы value – логикалық мән. Нәтиже шығаратын объект Boolean типіндегі нақсыз (атаусыз) мәнге ие.

Мәтін жолының мағынасы – string типіндегі элемент. Символдар жолын, яғни 0-ден немесе Unicode-ғы одан да көп символдардан (яғни белгісі жоқ 16 биттік бүтін сандар) тізбектелген массив.

String типі – бұл тип барлық мүмкін болған мәтін жолының мәндерінен тұрады.

Мәтін жолының объектісі – іштей құрылған объектінің көшірмесі болып табылатын Object типінің элементі. Басқаша айтқанда, мәтін жолының объектісі new string (value) өрнегімен жасалады, мұндағы value – мәтін жолының мәні. Нәтиже шығарушы объект string типіндегі нақсыз (есімсіз) қасиетке ие болады.

Сандық мән – Number типінің элементі. Нақты түрде белгілі бір санды береді.

Number типі – барлық мүмкін болған сандық мәндерден тұратын тип.

Шексіздік – Number типінің элементі болып табылатын примитивтік Infinity мәні.

NaN – Number типінің элементі болып табылатын примитивтік NaN (сан емес) мәні.

JavaScript тілінің негізгі функциялары [9]:

а) object (объект) әректтердің бәрі осымен жасалады. Бұл, броузер терезесінде ашылатын құжат немесе броузер терезесінің өзі, не бомаса, құжаттың қандай да бір бөлігі болуы мүмкін. Қарапайым түрде: бұл, өз интернет бетімізде қолданатын кез келген виртуалды «зат» болуы мүмкін. Біз стандартты JavaScript объектілерін қолданып, не болмаса, оларды өзіміз ойлап тауып, жасай аламыз. Онымен қоса, программаның өзі іші қолданыс үшін өз жұмысы барысында объектілерді жасай алады;

ә) property (қасиет) – мұны түсіндіріп жату да қажет емес сияқты. Онсызда, әрбір объектіге тән өз қасиеттерінің болатыны бәріне белгілі. Бір ғана объекті көптеген қасиеттерге ие болуы мүмкін: мысалға, бір үй үлкен және кішкентай, көк және қызыл болуы мүмкін. Әр түрлі объектілер бірдей қасиеттерге ие болуы мүмкін. Мысалға, бір ағаш, жаңағы үй сияқты үлкен және кішкентай, көк және қызыл болуы мүмкін. Объектілердің көптеген қасиеттерін біз әдістер арқылы өзгерте аламыз;

б) method (объект әдісі) – бұл, бізге объектінің белгілі қасиеттерін өзгертуге, яғни бұл объектілерді басқаруға және кейбір жағдайларда олардың

құрамын өзгертуге мүмкіндік беретін әрекет немесе әдіс. JavaScript-те қолданылатын көптеген әдістерді, сіз мұнда таба аласыз;

в) event (оқиға) – сізге бұл сөз таныс, себебі бұл туралы біз оқиғалар өңдеушісінде айтып кеткен боламыз. Оқиға деп барлық болып өткен жағдайды айтамыз: терезенің ашылуы, терезеге құжатты жүктеу, тышқанның батырмаларын басу немесе жай ғана курсорды экран бойынша жүргізу пернетақтадағы батырмаларды басу. Бұл оқиғалар кіші және үлкен программалардың орындалуын бастай алады;

г) оператор, бұл – компьютерге арналған команда, құрылым. Программада кез келген операторға жолыққан кезде, оны компьютер нақты түрде орындайды. Әскердегі сияқты: СОЛ-ҒА! Немесе ОҢ-ҒА!, осыдан кейін тыңдамай көр. Әрбір программалау тілі өзіне тән анық белгіленген операторлар жиынтығына ие болады. Әдістер, қасиеттер және объектілермен салыстырғанда, біз өз операторларымызды жасай алмаймыз: тіл интерпретаторы оны түсінбей, машинаға біздің не айтқымыз келгенін нақты түрде аударып бере алмайды;

ғ) процедура немесе функция, бұл – операторлардың белгілі тізбектілігі, яғни командалар жиыны, олардың тізбекті түрде орындалуы белгілі бір нәтижені тудырады. Мысалға, сіздің тарапыңыздан қойылған және біреу тарапынан орындалатын мына бір функцияның (процедураның): «стаканды ал, кранды аш, су толтыр, сосын маған алып кел» нәтижесі: Сіз краннан алынған сумен толтырылған стакан аласыз;

д) айнымалы, бұл енді, айнымалы! Маған таныс программистердің ешқайсысы маған айнымалының не екенін анық түсіндіре алмады. Яғни олар білсе де, маған түсіндірген кезде, мен еш нәрсе түсінбеймін.Сондықтан, жаңа «айнымалы» түсінігін есте сақтап алыңыз. Программлау тілдерінде айнымалылар белгілі типтегі деректерді сақтау үшін қолданылады, мысалға, объект қасиеттерінің параметрлерін сақтау үшін көмекші болады. Әрбір айнымалы өз атауына ие болып, программаның орындалу барысында өзгере алатын тек бір мәнді ғана сақтайды.

Оқиғалар мен оқиғалардың өңдеушілері JavaScript тілінде программалаудың өте маңызды бөлігі болып табылады. Оқиғалар, қолданушының кез келген әрекеті арқылы басталады. Егер, ол қандай да бір батырманы баса қалса, «Click» оқиғасы орындалаады. Егер де, тышқанның нұсқаушысы гипермәтіннің қандай да бір сілтемесін қиылысып өтетін болса, MouseOver оқиғасы орындалады. Оқиғалардың бірнеше типі болады. Біз, өзіміздің JavaScript программамызды солардың кейбіреуіне реакция беретіндей етіп жасай аламыз. Бұл оқиғаларды өңдейтін арнайы программалардың көмегімен жүзеге асады. Осылайша, батырманы басу арқылы терезе пайда болады. Бұл терезенің пайда болуы Click батырмасын басу оқиғасына берілетін реакция болу керек екенін білдіреді. Бұл жағдайда қолданылатын программа, яғни оқиға өңдеушісі onClick деп аталады. Ол, осы оқиға басталған кезде, компьютерге не істеу керек екенін хабарлап отырады. Төменде келтірілген код onClick оқиғасын өңдеу программасының қарапайым

мысалын береді [12]:

<form>
<input type=”button” value=”Click me” onClick =”alert”(‘Yo’)”>
</form>

Бұл мысалда бірнеше жаңа қасиеттер бар, оларды ретпен қарастырып шығайық. Мұнда батырмасы бар бір форма жасап жатқанымызды көре аласыз (мұның қалай жасалатыны – HTML тілінің жұмысы, сондықтан мен мұны қарастырмаймыз). Бірінші жаңа қасиет - <input> тэгіндегі onClick=”alert (‘Yo’)”. Атап өткеніміздей, бұл атрибут батырма басылған кезде не болатынын анықтайды. Егер Click оқиғасы орындалса, компьютер alert (‘Yo’) шақыруын орындауы керек. Осы, JavaScript (бұл жағдайда <Script> тэгін қолданбайтынымызға мән аударыңыз) тіліндегі кодтың мысалы болып табылады. Alert ( ) функциясы сізге терезе жасауға мүмкіндік береді. Оны шақырған кезде сіз жақшаның ішінде белгілі бір мәтін жолын жазуыңыз керек. Біздің жағдайымызда ‘Yo’ жазылады. Бұл мәтін сол терезеде пайда болатын мәтіннің дәл өзі. Осылайша, оқырман адам батырманы басқан кезде, бізідің скриптіміз ‘Yo’ мәтіні бар терезе тудырады. Кейбір жағдайларда, осы мысалда document.write ( ) командасында біз екілік тырнақшаны (“) қолданған болатынбыз, ал alert ( ) құрылымында тек бірегей тырнақшаны қолдандық. Неге? Көптеген жағдайларда сіз тырнақшаның осы екі түрін де қолдана аласыз. Дегенмен соңғы мысалда onClick =”alert”(‘Yo’)” деп жаздық, яғни біз екілік тырнақшаны да, бірегей тырнақшаны да қолдандық. Егер де біз onClick =”alert”(“Yo”)” деп жазсақ, компьютер біздің скриптімізді түсінбес еді, себебі onClick оқиға өңдеушісінің функциясы құрылымының қай бөлігіне қатысы бар екені түсініксіз болып қалады. Сондықтан осы жағдайда сізге тырнақшаның екі түрін де қолдану қажет. Ол тырнақшаларды қай ретпен қолданатыңыздың маңызы жоқ, бірінші екілік, содан кейін бірегей тырнақшаны немесе керісінше қолдансаңыз болады. Яғни сіз дәл осылай onClick =”alert’(‘‘Yo”)’ жолын жаза аласыз.

Скрипте оқиғаларды өңдеу функцияларының көптген түрлі функцияларын қолдана алады.

Егер сіз Netscape Navigator броузерін қолдансаңыз, онда терезеде JavaScript alert функциясы тарапынан берілген мәтін тұратын болады. Мұндай шектеу қауіпсіздік ережелеріне қарай қойылады. Бұл терезені сіз promt ( ) әдісімен де жасай аласыз. Бірақ та бұл жағдай да терезеде оқырманның енгізген мәтіні шығады. Сондықтан қастандық жасаушының жазған скрипті жүйелік хабарламаның түріне еніп, оқырманнан нақты бір құлып сөзін енгізуін сұрауы мүмкін. Егер де мәтін терезеден шығатын болса, оқырманға бұл терезенің сіздің операциялық жүйеңіз тарапынан емес, web-броузер тарапынан жасалғаны білдіріледі. Бұл шектеу қауіпсіздік ережелеріне қарай қойылғандықтан, сіз пайда болған хабарламаны жай ғана өшіре сала алмайсыз.

JavaScript тіліндегі көптеген программаларымызда, біз функцияларды қолданамыз. Сондықтан енді тілдің осы маңызды элементі туралы айта кетуім

қажет. Көптеген жағдайларда функциялар тек бірнеше командаларды байланыстыру үшін ғана қолданылады. Мысалға, бір мәтінді үш рет қайталап жазатын скриптті жазып көрейік. Алдымен, қарапайым бір жолды қарастырайық [12]:

```
<html>
<script language= “JavaScript”>
<!—hide
document.write (“Менің сайтыма қош келдіңіз! <br>”);
document.write (“Бұл – JavaScript! <br>”);
document.write (“Менің сайтыма қош келдіңіз! <br>”);
document.write (“Бұл – JavaScript! <br>”);
document.write (“Менің сайтыма қош келдіңіз! <br>”);
document.write (“Бұл – JavaScript! <br>”);
//-- >
</script>
</html>
```

Мұндай скрипт келесі мәтінді үш рет жазады:

Менің сайтыма қош келдіңіз!

Бұл – JavaScript!

Егер скрипттің бастапқы қодына қарасаңыз, онда қажетті нәтижені алу үшін, оның белгілі бір бөлігі үш рет қайталанғанын көре аласыз. Енді, айтыңызшы, осы жолды қолданған дұрыс па? Жоқ, дәл осы тапсырманы орындау үшін, біз төменде көрсетілген скриптті қолдана аламыз:

```
<html>
<script language= “JavaScript”>
<!—hide
function myFunction ( ) {
document.write (“Менің сайтыма қош келдіңіз! <br>”);
document.write (“Бұл – JavaScript! <br>”);
}
myFunction ( );
myFunction ( );
myFunction ( );
//-- >
</script>
</html>
```

Бұл скриптте, біз келесі жолдардан тұратын функцияны анықтадық:

```
function myFunction ( ) {
document.write (“Менің сайтыма қош келдіңіз! <br>”);
document.write (“Бұл – JavaScript! <br>”);
}
```

Өрнекті жақшалардың – { } ішінде орналасқан скрипттің барлық командалары командалары myFunction ( ) функциясының меншігі болып табылады. Бұл екі document.write ( ) командасының да бірігіп, көрсетілген

функция шақырылған кезде орындала алатындығын білдіреді. Шынымен де, біздің мысалымызда бұл функция үш рет шақырылады, функцияның анықтамасын бергеннен кейін myFunction ( ) жолын үш рет жазғанымызды көруге болады. Яғни шақыруды үш рет орындадық. Өз кезегінде, бұл функцияның құрамы (өрнекті жақшаның ішінде көрсетілген командалар) үш рет орындалғанын білдіреді.

Бұл функцияны қолданудың ең қарапайым мысалы болып табылғандықтан, сізде бұл функциялардың JavaScript-те неге соншалықты маңызды екендігі сұрақ болып туындалуы мүмкін. Берілген сипаттаманы оқысаңыз, сіз олардың пайдасын түсінесіз. Функцияны шақырған кездегі айнымалыларды бере алу мүмкіндігі біздің скрипттерімізге нағыз ыңғайлылықты қамтамасыз етеді, мұның не екенін біз кейінірек көреміз.

Функциялар, сондай-ақ оқиғаларды өңдеу процедураларымен бірге қолданыла алады. Келесі мысалды қарастырып көрейік [12]:

```
<html>
<head>
<sript language= “JavaScript”>
<!—hide
function calculation ( ) {
var x=12;
var y=5;
var result=x+y;
alert (result);
}
//-- >
</script>
</head>
</html>
```

Мұнда батырманы басқан кезде, calculation ( ) функциясының шақырылуы орындалады. Егер байқаған болсаңыз, бұл функция x, y және result айнымалыларын қолдану арқылы, белгілі есептеулерді жүзеге асырады. Айнымалыны, біз кілттік var сөзі арқылы анықтай аласыз. Айнымалыларды әр түрлі өлшемдерді – сандарды, мәтін жолдарын және т.б. сақтау үшін қолдануға болады. Осылайша, var result=x+y; скриптінің жолы броузерге result айнымалысын қойып, оған арифметикалық x+y (яғни 5+12).

1.5 Объектілер жайында жалпы мағлұматтар

JavaScript тілі JavaScript және C++ тілдерімен салыстырғанда қатаң түрде объектілер классынан тұрмайды. Оның орнына олар үшін объектілерді құруды жады бөлу жолымен жадыны конструкторларды қолдайды, ол Оның орнына ол объектілерді олар үшін жады бөлу және олардың барлық немесе кейбір қасиеттерін инициализациялау жолымен құратын конструкторларды қолдайды. Барлық конструкторлар объектілер болып табылады, бірақ бар

объектілер конструктор болмайды. Әрбір конструктор, протатипті және бөлінетін қасиеттерге негізделген ізін басуды іске асыру үшін қолданылатын prototype қасиетін иеленеді. Объектілер new операциясында конструкторларды шақыру жолымен құрылады; мысалы new String String жаңа объектісін құрады. New-сыз конструкторды шақыру нәтижесі конструкторға тәуелді. Сонымен, String ("бұл жол"), объект емес көп қолданылатын жолды құрады.

JavaScript протатипке негізделген, із басуды қолдайды. Әрбір конструктормен сәйкес протатип байланысқан, және конструктормен құрылған әрбір объект осы прототипке сілтемені құрайды (объектінің порототипі деп аталынатын). Прототип өз кезегінде, өз прототипіне және ары қарай сілтеме құруы мүмкін. Осылайша прототиптер тізбегі құрылады. Объектінің қасиетіне сілтеме – бұл берілген атпен қасиетті құрайтын объектілер прототипінің тізбегінің бірінші прототипіне сілтеме. Басқа сөзбен айтқанда, егер берілген объектінің берілген атпен қасиеті болса, онда осы қасиетке сілтеме қолданылады, егер жоқ болса, онда осы объектінің қасиеті зерттеледі және т.б.

Объектілер классына негізделген, объектке бағытталған тілдерде, ағымдағы қалып класстар экземплярларымен іске асырылады. Әдістер класстармен, ал із басушылық құраммен және мінез құлқымен. JavaScript-та ағымдағы қалып және әдістер объектімен іске асырылады, ал құрамы мен мінез
құлқы із басады. Барлық объектілер, олардың прототипінен құралатын,қасиетт ерден құралады, осы қасиеттерді және оның тағайындалуын бөледі. Класстарға негізделген тілдерден айырмашылығы, қасиеттер объектіге оларға мәндерді меншіктеу жолымен динамикалық түрде қосылуы мүмкін. Жеке жағдайда, конструкторлар мәндерді барлығына немесе құрылатын объектінің кейбір қасиеттеріне меншіктеуге міндетті емес [9].

1.6 PHP мүмкіндіктерін және артықшылықтарын сипаттау

PHP-дің мүмкіндіктері өте үлкен. Ең бастысы PHP-ді қолдану аясы сервер жұмыс істейтін жақтағы скриптерді жазуға көзделген, осылай PHP кез-келген CGI бағдарламасын орындай алады. Мысалы, форманың деректерін өңдеу, динамикалық беттерді генерациялау, cookies жіберу және қабылдау. Бірақ PHP басқа да көптеген міндеттерді орындай алады [6].

PHP-де қолданылатын негізгі үш сала бар:

– сервер жағында орындау үшін скриптерді құру. PHP дәл осылай кеңінен қолданылады. Сізге қажеті тек PHP парсері (CGI бағдарламасы немесе серверлік модуль түрінде), веб-сервер мен браузер. PHP скриптердің браузерде орындалу нәтижесін қарастырып отыру үшін сізге жұмыс істейтін веб-сервер мен орнатылған PHP керек;

– команда беретін жолда орындау үшін скриптер құру. Сіз веб-сервер мен браузерден тәуелсіз жібере алатын PHP скрипт құра аласыз. Бар керегі

PHP парсері. PHP-ді қолданудың мұндай тәсілі жүйелі түрде орындалуы қажет скриптер үшін өте оңды, мысалы cron көмегімен (Unix немесе Linux платформаларында) немесе тапсырмаларды жобалаушы (Task Scheduler) көмегімен Windows платформасында. Бұл скриптер мәтіндерді қарапайым өңдеу үшін де қолданылуы мүмкін;

– клиент жағында орындалатын GUI қосымшаларын құру. Мүмкін PHP мұндай қосымшаларды құру үшін ең жақсы тіл емес шығар, бірақ PHP-ді жақсы білсеңіз және оның кейбір мүмкіндіктерін өзіңіздің клиент қосымшаларыңызда пайдаланғыңыз келсе ондай қосымшаларды құру үшін PHP-GTK қолдануыңызға болады Осылай сіз кросс платформалы қосымшаларды құра аласыз. PHP-GTK PHP-дің кеңеюі болып табылады және PHP дистрибютивімен бірге жеткізілмейді [6].

Linux және Unix-тың көптеген модификацияларын (HP-UX, Solaris, OpenBSD сияқты) Microsoft Windows, Mac OS X, RISC OS және басқаларын қоса көптеген операциялық жүйелер үшін PHP қол жетімді. Сондай-ақ, PHP-ге Apache, Microsoft Internet Information Server, Personal Web Server веб-серверлердің, Netscape пен iPlanet серверлерінің, Oreilly Website Pro, Caudium, Xitami, OmniHTTPd серверлерінің және тағы басқа сол сияқты көптеген заманауи веб-серверлер қолдауы қосылған. Көптеген серверлер үшін PHP модуль негізінде жеткізіледі, басқалар үшін CGI стандартын қолдайтын PHP CGI процессоры ретінде қызмет көрсете алады [13].

Осылай, PHP-ді таңдай отырып сіз операциялық жүйе мен веб-серверді таңдау еркіндігін аласыз. Одан басқа сізге процедуралық немесе нысаналы бағдарланған программалауды немесе олардың үйлесуін қолдануды таңдауыңызға болады.

PHP HTML-ді ғана бермейді. PHP мүмкіндіктеріне бейнелерді қалыптастыру, PDF файлдары және «ат үсті жасалатын» (libswf пен Ming көмегімен) Flash роликтері кіреді. PHP HTML және басқа XML файлдары сияқты кез-келген мәтіндік деректерді бере алады. PHP мұндай файлдарды автоматты түрде генерациялауды және оларды клиентке берудің орнына, сіздің сервердің файлдық жүйесінде сақтай алады, осылай сервер жағында орналасқан динамикалық мазмұны бар кешті ұйымдастырады.

PHP-дің маңызды артықшылықтарының бірі – мәліметтер қорының кең шеңберін қолдай алады. Деректер қорын қолданатын скрипті құру өте оңай. Қазіргі кезде PHP келесі деректер қорын қолдайды [5]:

– Adabas D;
– dBase;
– Empress;
– FilePro (только чтение);
– Hyperwave;
– IBM DB2;
– Informix;
– Ingres;
– InterBase;

– FrontBase;
– mSQL;
– Direct MS-SQL;
– MySQL;
– ODBC;
– Oracle (OCI7 и OCI8);
– Ovrimos;
– PostgreSQL;
– Solid;
– Sybase;
– Velocis;
– Unix dbm.

Сонымен қатар, PHP-ге абстрактілі деңгейде жұмыс істеу үшін DBX қолдауы қосылған, сондықтан, DBX қолданатын кез-келген мәліметтер қорымен жұмыс істей аласыз. Одан басқа PHP ODBC-ны қолдайды (Open Database Connection standard), осылай сіз бүкіл әлем мойындаған стандартты қолдайтын кез-келген деректер қорымен жұмыс істей аласыз. Сонымен бірге, PHP LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (Windows платформасындағы) және тағы басқа протоколдарды пайдаланатын болса, сервистермен қатынасты қолдайды. Оның үстіне желілік сокеттермен тікелей жұмыс істеу мүмкіндігін аласыз, PHP WDDX деректерінің күрделі құрылымдарымен алмасу стандартын қолдайды. Түрлі тілдер арасындағы өзара іс-әрекетке назар аудара отырып, Java нысаналарын қолдауды және оларды PHP нысаналары ретінде пайдалану мүмкіндігін ескеру керек. Алыстатылған нысаналарға жетімді болу үшін, CORBA кеңейтуін қолдана аласыз.

PHP-ге мәтіндік ақпаратты өңдеу құралдары кіреді, Perl немесе POSIX Extended тұрақты таңбалардан бастап, XML құжаттар парсерімен аяқталады. XML парсингі үшін SAX пен DOM стандарттары қолданылады. XML құжаттарын қайта жасау үшін XSLT кеңейтуін қолдана аласыз [6].

Басқа көптеген кеңейтулерді қолдау, мысалы, mnoGoSearch, іздеу машинасының қызметтері IRC Gateway қызметтері қысқа файлдармен жұмыс істеуге арналған қызметтер (gzip,bz2), күнтізбелік есептеу қызметтері, аудару қызметтері және т.б. мәні бойынша емес реттік кезегі бойынша соңғы болып табылады.

PHP тілінің негізгі факторы тиімділігі. PHP программалаушыға жүктелген міндеттерді тез әрі тиімді шешетін құралдар беруі қажет. PHP тиімділігі бес маңызды сипаттамалармен шартталады [7]:

– дәстүрлілік;
– қарапайымдылық;
– тиімділік;
– қауіпсіздік;
– икемділік.

PHP-ді тартымды қылып көрсететін тағы бір сипаттама бар: ол тегін

таратылуы. Оның үстіне негізгі коды ашық (Open).

PHP тілі түрлі салаларда жұмыс істейтін программалаушыларға таныс болып көрінуі мүмкін. Тілдің көптеген құрылымдары Си мен Perl-дан алынған.

PHP коды C немесе Pascal типтегі программаларда кездесетінге өте ұқсас. Бұл PHP-ді оқу кезінде алғашқыда күш салуды азайтады. PHP – Perl мен Си қасиеттерін үйлестіретін және арнайы Интернетте жұмыс істеуге бағытталған, әмбебап және таза синтаксисі бар тіл.

PHP жас тіл болғанымен, веб программалар арасында өте танымалдығы сонша, дәл осы кезде web-қосымша (скрипт) жасау үшін ең танымал тіл болып тұр [6].

1.7 PHPMyAdmin мүмкіндіктерін сипаттау

PHPMyAdmin жалпы MySQL серверінде (ол үшін супер-пайдаланушы құқығы қажет), жеке мәліметтер қорын да басқара алады. Мульти пайдаланушылық қолданыс мүмкін (мануал MySQL “мысалға, жаңа MyAdmin пайдаланушыны қосу”). Соңғы жағдайда пайдаланушылар оларға белгіленген деректерді ғана қолдана алады [8].

Қазіргі кезде PHPMyAdmin рұқсат етеді:

– мәліметтер қорын жасау және жою;
– кестелерді құру, көшіру, жою, атын өзгерту, өзгерту;
– кестелерді алып жүру;
– өрістерді жою, түзету және толықтыру;
– SQL сұраныстарын орындау, оның ішінде пакетті SQL-сұраныста;
– кілттерді басқару;
– мәтіндік файлдарды кестеге жүктеу;
– кестелердің дампын құру мен қарастыру;
– CSV,XML,PDF,ISO/IEC26300-OpenDocument Text and Spreadsheet, Word, Excel және LATEX форматтарында деректерді экспорттау;
– бірнеше серверді әкімшілеу;
– MySQL пайдаланушылары мен тұтастығын тексеру;
– MySQL кестелеріндегі сілтемелік деректердің тұтастығын тексеру;
– (Query-by-example-QBE) үлгісі бойынша сұранысты қолдану, берілген кестелермен автоматты түрде қосылу арқылы кешенді сұраныстар құру;
– PDF формаларында мәліметтер қорының графикалық сызбасын жасау;
– мәліметтер қорында немесе оның бөлімдерінде іздеуді іске асыру;
– белгіленген қызметтерде қолданылатын сақталған деректерді түрлі форматтарға түрлендіру, мысалы BLOB – деректерінің көлеңкесін бейне түрінде немесе жүктелетін сілтеме және т.б.;
– Inno DB кестелерін және сыртқы кілттерді қолдайды;
– MySQL жақсартылған кеңейтуін қолдайды;
– 50-ден астам тілге аударылған.

1.7.1 PHPMyAdmin орнату

Mac пайдаланушыларына ескерту: егер сізде MacOS-тан OSX, Stufflt нұсқасы орнатылса, Mac-архиватор көмегімен архиватордан шығарыңыз. Барлық PHPMyAdmin скриптерін серверге көшірерде Unix ВСЕ стиліндегі BBEdit көмегімен қайта сақтаңыз, себебі PHP жолды көшіру Mac-символы (“\r”) жақсы көрмейді [8].

Тез орнату:

а) дистрибутивті өзіңіздің веб-сервердің түпкі директориясына шешіңіз (директорияның асты да, шешілгеніне көз жеткізіңіз):tar-xzvf phpMyAdmin_x.x.x.tar.gz. Егер сіз веб-кеңістіктің түбіне тура кіре алмасаңыз, директорияны файлдарға локальды машинада шешіңіз де, 3-ші адымнан кейін директорияны веб-серверге көшіріңіз, мысалы, ftp арқылы;

ә) барлық скриптердің нақты иесі бар екеніне көз жеткізіңіз (егер PHP қауіпсіз тәртіпте қосылса, түрлі иелері бар бірнеше скрипттің болуы жұмыс кезінде қиындық тудырады);

б) енді пішіні үйлесімді файл құруыңыз керек (config.inc.php). Пішіні үйлесімді файлды құру – PHPMyAdmin іске қосу үшін қажетті шарт, сонымен қатар кей мүмкіндіктерге тапсырма беру үшін қажет. Бұл тапсырма екі түрлі тәсілмен шығарылуы мүмкін. Дәстүрлі түрде пайдаланушылар config.inc.php файлының көшірмесін қолмен өңдеуге болады. Одан басқа графикалық инсталяцияны артық көретін пайдаланушылар орнату шеберін пайдалана алады.

Пішіні үйлесімді файлды қолмен жасау үшін қарапайым мәтіндік редактор жетіп жатыр, соның көмегімен config.inc.php файлын жасау керек. Тапсырманы жеңілдету үшін config.sample.inc.php файлының ішіндегісін, конфигурациялық айнымалылардың минималды саны бар – жұмыс конфигурациясының үлгісі, мәтіндік редакторға көшіре саласыз.

Құрылған файлды PHPMyAdmin негізгі басқы директорияға салу керек. (онда index.php) бар. PHPMyAdmin алдымен libraries/config.default.php жүктейді, содан соң, config.inc.php-ден табылған айнымалылармен онда жазылған мәндерді ауыстырады. Егер бірден мәні (libraries/config.default.php белгіленген директориялардың) сізді қанағаттандырса, оны config.inc.php-ге қосудың қажеті жоқ. PHPMyAdmin қосу үшін сізге бірнеше директива қажет болады, ең қарапайым конференция мынадай болады [15]:

```
<?php
$cfg['blowfish_secret'] = 'ba17c1ec07d65003'; // use here a value of your choice
$i=0;
$i++;
$cfg['Servers'][$i]['auth_type']='cookie';
?>
```

Немесе егер сіз қайта-қайта логин (құпия сөзді енгізгіңіз келмесе) бұл

конференция қауіпсіздік оймен ұсынылмайды, конференциялық файл мынадай болуы мүмкін:

```
<?php
$i=0;
$i++;
$cfg['Servers'][$i]['user']='root';
$cfg['Servers'][$i]['password'] = 'cbb74bc'; // use here your password?>
```

Config.inc.php файлын қолмен редакциялаудың орнына орнату скриптін қолдануға болады. Алдымен сізге негізгі директорияда қолмен PHPMyAdmin «config» папкасын құру керек. Бұл – қауіпсіздік шарасы. ОС Linux/Unix-де жұмыс істегенде келесі командаларды пайдалана аласыз:

cd phpMyAdmin;

mkdir config # сақтау үшін директория құрады;

chmod o+rw config # оған жазуға барлық қолданушыларға мүмкіндік береді.

Басқа платформаларда жай ғана директория құрыңыз да сіздің веб-серверіңіз оқу мен жазба жасауға құқығы бар екеніне көз жеткізіңіз.

Содан соң, браузерде script/setup.php ашыңыз. Өзгерістер «Configuration» блогындағы Save түймешесін баспайынша дискте сақталмайтынын есте сақтаңыз. Егер бәрі жақсы болса, скрипт жаңа config.inc.php директорияда сақтайды, ал егер веб-сервердің тиісті құқығы жоқ болса қате тұр «Cannot load or save configuration» хабарламасын көресіз. Ондай жағдайда config/ директориясы құрылғаны және тиісті құқығы бар екеніне көз жеткізіңіз немесе конференциялық файлды локальды дискке сақтау және кейін серверге (мысалы, FTP арқылы) жүктеу үшін «Download» сілтемесін пайдаланыңыз.

Файл сақталған соң, оны config/ директориясынан негізгі PHPMyAdmin директориясына ауыстыру қажет және қауіпсіздік мақсатында құқықтан бас тарту керек [14]:

mv config/config.inc.php. # файлды ағымдағы директорияға ауыстырады; chmod o-rw config.inc.php # қалған барлық қолданушылар үшін оқу және жазу құқықтарын алып тастайды. Енді файл пайдалануға дайын. Дәлдеуі орнату скриптімен қарастырылмаған, егер кейбір кеңейтілген опцияларды орнату керек болса, сіз конференцияның файлда мәтіндік редакторды қарай және редакциялай аласыз. Егер сіз «config» аутентификациясын (auth_type) қолданатын болсаңыз, PHPMyAdmin орнатылған директорияны жүйеге рұқсатсыз кіруден қорғау қажет, себебі бұл тәртіп кез-келген пайдаланушыға phpMyAdmin-ге логин/құпия сөзді алдын-ала енгізбей-ақ кіруіне мүмкіндік береді. Аутентификацияның балама тәсілдері ұсынылады, мысалы, HTTP-AUTH (.htaccess файл көмегімен), немесе аутентификацияның екі тәсілдерінің бірін пайдалану арқылы: cookie немесе http.

Браузердегі PHPMyAdmin негізгі директориясын ашыңыз. PHPMyAdmin сәлемдесу терезесі мен сіздің деректер қорыңыз немесе логинді енгізу терезесі пайда болуы керек, егер HTTP-режимін немесе cookie-

аутентификацияны қолдансаңыз.

/libraries субдиректорияға кіруге веб-сервердің құралдарымен тыйым салуыңыз керек. Apache веб-серверін директорияны қорғау үшін қолданған жағдайда .htaccess файлын қолдануға болады. Басқа веб-серверді қолданғанда сіз ./libraries директориясына кіруге өзбетіңізше тыйым салуыңыз керек. Мұндай конференция осалдықты жолды ашу мүмкіндігі мен сценарийді сайт-аралық орындау(path exposure) тауып алу жағдайындағы табылған шара (Cross-site Scripting, XSS).

Локальды серверлерді пайдаланудың себебі көп – бізге PHP-ді үйрену керек болды өзіңіздің хостингіңізде Web-қосымшалары тестілеу қымбатқа түседі және бұндай мүмкіндік мүлдем жоқ. Бұл жағдайда сізде локальді машинада Apache+PHP байланыстары қажет болады [13].

Ең алдымен Apache және PHP-дің архивін шығарып алу керек. Apache http://www.apache.org/dyn/closer.egi. ресми сайтындағы келтірілген беттен алуға болады. Іздестіруде мынаны есте сақтаған жөн. Apache өзінің UNIX тегі доменінің аты бойынша httpd деп те аталуы мүмкін. Әдетте сайт бетінде әртүрлі файлдар көп, мысалы, httpd-2.0.49-win 32-src.zip, httpd-2.0.49. tar. Gz, apache 2.0.50-win 32-x86-no ssl-exe.

PHP 5-ті http://www.php.het/downloads.php сайтынан табуға болады. Сайтта PHP екі формада мүмкін болады. Алғашқы кодта (Complete Source Code) және компилицияланған нұсқада (Windows Binsries). Бізді ек нұсқада таралатын компиляцияланған нұсқа қызықтырады: орнату түрінде (php-5.0.0-installer exe) және zip архив түрінде (php-5.0.0-Win 32. zip). Орнатушы орнатуда ыңғайлы, бірақ PHP версиясы шектеулі. Оның үстіне автоматты инсталляторды пайдалану бізді Apache серверінің конфигурациялаушы файлын құру қажеттілігінен арылтпайды. Сондықтан да, zip-архивті жүктеу ұсынылады. Сіз PHP сайтында бола тұрып, PHP құжаттаманы алуыңызға да болады. Windows-ты пайдаланушылар үшін chm-формат әлдеқайда ыңғайлы [6].

1.8 MySQL мәліметтер қорымен байланысу функциялары

SQL – реляциялық мәліметтер қорымен өзара іс-әрекетте қолданылатын стандартты тіл ретінде сипатталады. Бірақ, SQL – C++ немесе PHP сияқты программалаудың тілі болып табылмайды [17].

Бұл деректер қорымен түрлі операциялар орындау үшін қолданылатын, интерфейсті құрал және пайдаланушылардың билігіне берілетін, стандартты командалар жинағы. SQL мүмкіндіктері қордан деректерді таңдап алумен шектелмейді. SQL-де деректер қорымен өзара іс-әрекет ететін түрлі мүмкіндіктер қолдау табады, оған қоса:

– деректер құрылымын анықтау – деректерді сақтау үшін қолданылатын құрылымдарды анықтау;

– деректерді таңдау – қордан деректерді жүктеу мен оларды ыңғайлы форматта көрсету;

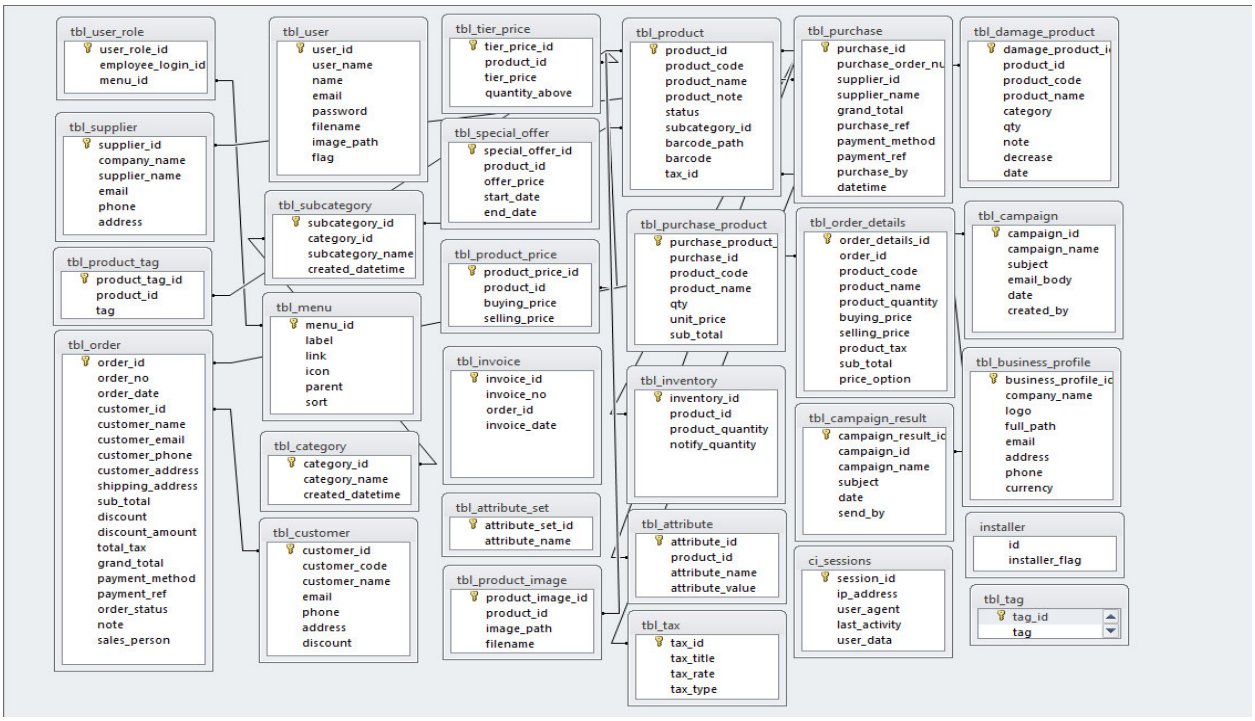
– деректерді өңдеу – қою, жаңарту және ақпаратты жою;

– кіруді бақылау – жеке пайдаланушылар деңгейінде деректерді таңдауға рұқсат ету немесе тыйым салу, қою, жаңарту және жою;

– деректердің тұтастығын бақылау – параллельді жаңарту немесе жүйелік істен шығару сияқты мәселелер туғанда деректер құрылымын сақтап қалу. Деректердің реляциялық қорының мысалы 1.9-суретте бейнеленген [20].

SQL-ге сипаттама бергенде, бұл тіл деректердің реляциялық қорымен жұмыс істеуге арналған деп айтылған. Реляциялық ДҚБЖ деректер өзара байланыстағы кестелер жинағы түрінде ұйымдастырылады. Кестелер арасындағы байланыстар басқа кестелер деректеріне сүйену түрінде іске асырылады. Кестені екіөлшемді алқап (массив) түрінде елестетуге болады, ондағы әр элементтің орналасуы жол мен бағананың мәндерімен сипатталады.

1.9-суретте көрсетілгендей, әр кесте жол (жазба) және бағанадан (өріс) тұрады. Әр өріске бірегей (берілген кесте аясында) ат беріледі. Customer мен orders кестелері арасындағы байланысқа назар аударыңыз, жебемен белгіленген. Тапсырыс туралы ақпаратқа клиенттің қысқаша идентификаторы қосылады, ол клиенттің аты мен реквизиттерін артық сақтаудан құтылу мүмкіндігін береді. Бейнеленген деректер қорында тағы бір байланыс бар orders пен products кестелері арасында. Бұл байланыс клиент тапсырыс берген (custjd өрісі белгілейтін) тауар идентификаторы сақталатын prod_id өрісінде орнатылады. Бұл байланыстардың болуы, клиент пен тауардың толық деректеріне қарапайым идентификатор арқылы оңай сүйенуге болады. Дұрыс ұйымдастырылған деректер қоры күшті ұйымдастыру құралына айналады және минималды артықшылықпен деректерді тиімді сақтайды [16].



1.9-сурет – Деректердің реляциялық қорының мысалы

1.9 Есептің қойылымы

Дипломдық жобаның мақсаты – медициналық орталықтарға арналған «Дәріхана» ақпараттық жүйесін құру болып табылады. Қазіргі кезде жұмыс орнын автоматтандыру интернет желісінде ең тез дамып келе жатқан қызмет түрінің бірі. Құрылған ақпараттық жүйе арқылы администратор дәрі-дәрмек деректер қорымен жұмыс жасай алады.

Бұл жобаның негізгі мақсаты: дәрі-дәрмек сататын медициналық орталықтарға барынша ыңғайлылық тудыру.

Дипломдық жобаның нәтижесінде келесі негізгі тапсырмалар орындалуы қажет:

– АЖО құрылымын құру;

– программалық қамтаманы таңдау;

– тиімді және түсінікті қолданушылық интерфейсті құру;

– ақпараттық жүйеге әкімшілік ету жүйесін ұйымдастыру.

Медициналық орталықтарға арналған «Дәріхана» ақпараттық жүйесін құру барысында PHP скрипт тілі, HTML гипермәтіндік тілі, CSS каскадты стильді кестелер тілі, MySQL деректер қоры және frontend пен backend-та жұмыс жасайтын сценарийлерді жазуға арналған объектілік бағытталған программалау тілі JavaScript қолданылады.

2 Жобалау бөлімі

2.1 Пәндік саланы модельдеу

Жүйені құру үшін оны ең алдымен дұрыс жобалау қажет. Қазіргі кезде барлық программалау орталары объектілі бағытталған программалау концепцияларын қолдайды, сондықтан да құрылатын жүйені жобалау үшін UML (Unified Modeling Language) тілін қолдану өте тиімді болады. Осы UML тілі объектілі бағытталған концепцияларының негізінде құрылатын жүйелерді жобалау үшін арналған және оларды жобалаудың стандартты құралы болады, себебі олардың негізгі міндеті – жобалау процесін бір жүйеге келтіріп, жобалаудың стандартын жасау. Жобалаудың басында ең алдымен заттық аймақты анықтау қажет.

Заттық аймақты модельдеу кәсіпорын масштабында программалық жүйені жобалау кезінің ең бір маңызды кезеңдері болып табылады. Бүгінгі күні программалық өнім – нарығында пәндік облысты модельдеу мақсатында CASE-құралдарының кең спектрі ұсынылған. Біздің елдегі ең белгілі CASE-құрылымдарын Rational Rose PPwin Silvenrun Process Analyst жатады. Пәндік облысты осы құралдарда модельдеудің айырмашылығынан гөрі ұқсастықтары көп. Бірақ біздің көзқарасымызда бір унифицирланған нотацияны қолдану және комплекстік қадам тек пәндік облысты модельдеуде емес, программалық жүйені келесі этаптарында CASE Rational Rose орын алады [21].

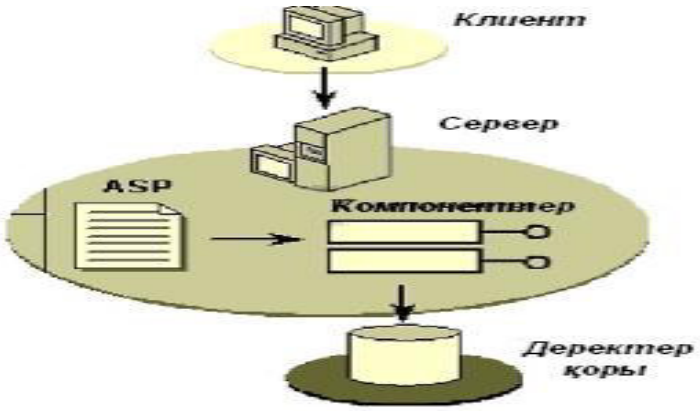
Мысалдарды негізінен UML-ді қолданып, пәндік облысты модельдеу қадамдарын демонстрлайды және олардың объектік және құрылымдық жобалау әдістерінің үлкен жетістігін жинаған CASE және Rational Rose. Сонымен пәндік облысты модельдеу кезіндегі негізгі міндеттер:

– кәсіпорынның бизнес-процесстері;
– бизнес-процестерге әсерлесетін тұлғалар және олардың функциялары;
– бизнес-маңыздылық;
– автоматтауға жататын бизнес-функция орындау сценарийлары;
– бизнес-маңыздылығының жағдайы;
– бизнес-ереже.

Модельде тек мына бөлімдер көрінеді, автоматталатын әрекеттесуші тұлғалар мен олардың функциялары. Модельді құрудың бизнес-процестер шарасымен этап бойынша құруға болады:

– жүйенің қолданушы құруға болады;
– жүйенің деректер қорын жобалау.

Төменде 2.1-суретте пәндік облыстың моделі қарастырылған [22].



2.1-сурет – Пәндік облыстың моделі

Пәндік облысты модельдеу модельдің негізгі статикалық бөлімі болып табылады.

Зат облысын модельдеу процесі нақты дүниедегі абстракция шығуы мүмкін. Нақты өмірдегі негізгі абстракция ақпаратты құру бағытында қайта қолдануға мүмкінділігі болып табылады. Зат облыс моделін модельдеу кезінде «іштен сыртқа» жоба методикасы қолданылады.

Бұл жүйеде кілт объектілерінен бастауын білдіреді. Содан соң сырты қандай объектілермен олар қарым-қатынаста болатының жылжып үйренеді. Осындай әдіспен өткенде орын алғаның көрсетуін немесе жүйенің диналикалық бөлімінде сырттан ішке қозғалысы жүреді, ал статикалық модельді шығарар кезде іштен сыртқа жүреді.

Зат облысының модельдеуі негізгі статикалық модель бөлімі болып табылады. Ол нақты дүниеде, концептуалдық объектілерде, жүйелерде кездеседі. Объектілерді жобалау кезінде бағытталған ақпаратты қамтамасыз етуіне құрылымдық ақпарат қамтамасыздығы талап етіледі, өйткені ортады құрылған тапсырмадан сол объекттер болу керек. Ақпаратқа талап тез өзгереді, нақты өмірге қарағанда [22].

Объект моделінің негізгі және статикалық модельдеу зат обласына сол абстракцияның модельдерін шақыру болып табылады. Зат обласының модель термин сөздігін білдіреді. Оны шығару үшін және өткенді сипаттау объектілері мен класстары үшін қолданылады.

Зат облысынан көрсетілген объектілер кірерде олардың арасында қандай байланыс бар екенін білу қажет. Ең қажетті байланыстар болып ол реакция (бөлім мен бағыт арасындағы қатынас) және жинақтап қорыту жатады.

Командалық және жеке жобалау және оған керекті аспаптар. Текстуальды және визуальды жобалау – технологиялары, қолданатын аспаптары. Визуалды модельдеу дегеніміз не? Визуальды модельдеу (visual modeling) – нақты өмірдің объектілері мен ұғымдарын көрсететін және өзі көрініс табатын абстракциялардың көмегімен проблемаларды қабылдау әдісі.

Модельдеу проблемаларын талдау, барлық қызығушылығы бар жақтар

арасында ақпараттардың алмасуы (қолданушылар, пәндік аумақтағы мамандар, аналитиктер, дизайнерлер және т.б.), программалық қолданбалар мен мәліметтер қорын жобалау, документацияны дайындау үшін пайдалы құрал болып табылады. Модельдеу талаптарын жақсы қабылдауға, жүйенің дизайн сапасын жақсартуға және оны басқару деңгейін жоғарылатуға өз септігін тигізеді. UML – объекті бағдарланған жүйелермен құрастырылатын мәндердің документациясын белгілеу мен көріністендіру үшін қолданылатын тіл [21].

2.2 Rational Rose аспабымен жұмыс

Rational Rose өнімдерінің сериясы құрастырушыны нақты уақыт жүйелерінде және «клиент-сервер» орталарында қолдануға жарайтын және қазіргі кездегі бизнес талаптарын қанағаттандыратын тиімді де сенімді шешімдерді қабылдауға көмектесетін визуалдық модельдеудің толық құралдар жиынымен қамтамасыз етеді. Rational Rose құралдары біркелкі стандарттарға негізделген және модельдеуді, оларға жақын сфералардағы бизнес-процестерді оптимизациялауға талаптанатын, компьютерлік ғылымдармен онша таныс емес тұлғалармен қатар, программалық қолданбалардың логикасын модельдеу құралдарын қажет ететін мамандар үшін оңай етеді. Rational Rose тексермелік (пробная) пакетінің көшірмесін Rational Software Corporation сайтында http://www. rational.com адресі бойынша табуға болады.

Программаны құрудың бірде-бір әдістемесі белгілі бір құралсыз болмайды. Қазір нарықта құралдардың ауқымды түрлері келтірілген қарапайым сызба программалардан бастап, объектілері модельдеудің ең күрделі жүйелеріне дейін. Кітап парақтарында сіздер Rational Rose 2003 құралдық қабықшамен танысасыздар. Модельдеу барысында біз жасайтын кез-келген әрекет, Rational Rose функцияларын қолдану бойынша сәйкес инструкциялармен толықтырылып отырады [23].

Объектілі-бағдарланған талдау және жобалаудың көптеген әдістерінде модельдеу тілдері және модельдеу процесін сипаттау қоса жүреді. Модельдеу тілі дегеніміз жобаны сипаттау әдісі ретінде қолданылатын нотация (негізінен графикалық). Нотация модельдерде қолданылатын графикалық объектілердің жиынтығы және ол модельдеу тілінің синтаксисі болып табылады. Мысалы, кластар диаграммасы класс, ассоциация және жиынтық сияқты элементтер мен түсініктердің берілу түрін анықтайды. Процесс дегеніміз жобаны құрастыру барысында орындалатын қадамдарды сипаттау.

Қазіргі кезде UML-тілін OMG (Object Management Group-объектілі бағдарланған тәсілдер мен технологиялар облысындағы стандарттау ұйымы) стандарттаудан өткізуде. Қазіргі кезде UML модельдеудің стандартты тілі ретінде қабылданып, ПҚ индустриясында кеңінен қолдау тапты. UML сипаттамасы орыс тілінде М.Фаулер және К.Скот кітабында келтірілген.

UML-ды құрастырушалыр оны программалық жүйені анықтау, ұсыну,

жобалау және құжаттау тілі ретінде жасақтады. 1997 жылы OMG қабылдаған UML 1.1-версия стандарты моделдеу үшін келесі диаграммалар құрамасын ұсынады: қолдану варианттары диаграммасы (use case diagrams) ұйымның іскерлік процесін модельдеу үшін (жүйеге талаптар); кластар диаграммасы (class diagrams) жүйе кластарының статикалық құрылымы мен олардың арасындағы байланысты модельдеу үшін; жүйе тәртібі диаграммасы (behavior diagrams); өзара әсер диагарммасы (interaction diagrams); объектілер арасындағы хабар алмасу процесін моделдеу [23].

Rational Rose – UML әдістемесін тарату аспабы. Ұқсас есептерді шешу үшін қолданылатын басқа да визуальды аспаптар. Өңдеу функцияналдығын сипаттау – қолдану варианттар диаграммасы (талаптар және шектеулер), функция орындалу реттері – әрекеттердің тізбектер диаграммасы (талаптар және шек теулер), өңдеу элементерінің бір-бірімен байланыс және бар болу сипаттамасы – кооперативті диаграммалар (талаптар және шектеулер).

Интерфейсті жобалау. Обьектілік жобаның өңделуі, ОМ 6 қосымшасына сәйкес, негізінен «прецеденттер диаграммасы» (ПД) немесе, екінші аталуы – «Қолдану варианттар диаграммасы» (ҚВД) басталады. Диаграмма прецедент барлық қалған жоба диаграммалары үшін арналып жасалады. Ал оның екінші аталуы, негізінен, оның атқаратын рөлін толық көрсетеді деп айтуға болады – пайдаланушы жүйесінің қолдану варианттар диаграммасы (актерлар, әрекет жасаушы обьект). Сонымен бұл диаграмма негізінен, егер бұл ұстанымды Коуда (Coad) атымен беріп орындайтын болсақ, онда жобалаушы сұрақтарына жауап ретінде: кім, қашан және не істегендігін білуге болады. Келесі диаграмма «Тізбек диаграммасы» (жеке қолдану вариантын орындау) орындауды жеке түсіндіреді. Бұл диаграммада, егер Коуда әдістемесін қолданатын болсақ, онда актер вариантты орындау кезінде кіммен (немен) байланысқандығы көрсетіледі. Бұл диаграммада орындау уақыты анықталады, ол жүйеге нақты уақыт (RTS) жүйесінің қалай қызмет көрсететіндігін көрсетеді және жүйе ON-LINE типінде реакцияны шектеумен ерекшеленетіндігін де көрсетеді. Бұл диаграммада вариантты қолданушылардың арасындағы өтулер қолданушы әрекетімен анықталады. Сонымен қолданылатын обьектіге кандидат (немесе обьект атрибуты) болып табылады, ал ол орындайтын әрекет – обьектінің әдісінің кандидаты болып табылады. Сондықтан, диаграмма обьектілерді анықтаудың негізі болып табылатындығын айтып кетуге болады. Бұл диаграммада қолданушылар ретінде интерфейстің қолданылып отырған элементтерін және басқа ақпараттық обьектілерді алуға болады.

Объект нақты немесе абстрактылы мән ретінде болады. Объект дегеніміз программалық қосымшада нақты бір шекаралар, мағыналар және мәндермен берілген түсінік немесе абстракция.

Жүйенің әрбір объектісінің үш сипаттамасы болады – жағдайы, тәртібі және біркелкілік белгісі.

Объект жағдайы атрибуттар – касиеттернінің жиынтығы және басқа абстракциямен байланыс арқылы анықталады.

Мінез-құлық сипаттамасы объектінің функционалдық өмірін қамтуы, басқа объектер сұранысына әсерін зерттеп және операциялар жиыны түрінде іске асырылады.

Біркелкілік белгісі объектінің әмбебаптығын анықтайды – басқа объектілермен бірдей болған жағдайда да клас объектілер тобын ортақ атрибуттар , қасиеттер, мінез-құлық (функционалдар), семантика және басқа объектілер мен байланыс арқылы анықтайды. Класты басқаша объектні құруға арналған шаблон деп те атауға болады. Әрбір объект тек бір кластың данасы болып табылады. Класты құрған кезде оны құжаттандыру керек. Сипаттама класс құрылымын емес, оның мәнін беру керек.

Rational Rose өнімдерінің сериясы құрастырушыны нақты уақыт жүйелерінде және «клиент-сервер» орталарында қолдануға жарайтын және қазіргі кездегі бизнес талаптарын қанағаттандыратын тиімді де сенімді шешімдерді қабылдауға көмектесетін визуалдық модельдеудің толық құралдар жиынымен қамтамасыз етеді [23].

Rational Rose құралдары біркелкі стандарттарға негізделген және модельдеуді оларға жақын салалардағы бизнес процестерді оптимизациялау талаптанатын, компьютерлік ғылымдармен онша таныс емес тұлғалармен қатар, программалық қолданбалардың логикасын модельдеу құралдарын қажет ететін мамандар үшін оңай етеді.

2.3 Прецеденттер диаграммасы

Қолдану варианты деп сыртқы объект (орындаушы) тудырған оқиғаға жүйе жауап ретінде орындаған әрекеттердің тізбегін айтады. Мысалы, қарапайым текстік процессорды қолданудың типтік екі варианты – белгілі бір текстi «курсивті ету» және «индекс жасау». Осындай қарапайым мысалдың өзінде қолдану вариантының кейбір қасиеттерін көрсетуге болады: ол қолданушыға таныс белгілі бір функцияны қамтуы мүмкін, сондай-ақ кіші немесе керегінше үлкен де бола алады және қолданушы үшін белгілі бір дискретті есепті шеше алады. Қолданушының қандай да бір функцияларды іске асырғысы келетінін талқылау процесінде қолдану варианты анықталады.

1994 жылы Якобсон қолдану вариантын программалық қамтамасын (ПҚ) жасау процесінің негізгі элементі етуді және оларды қолдану варианты диаграммасының көмегімен көрнекі етуді ұсынады. Қолдану вариантындағы адам фигуралары орындаушылар, сопақ шеңберлер қолдану варианты, ал сызықтар мен бағыттауыш сызықтар – орындаушы мен қолдану варианты арасындағы байланыстарды көрсетеді [23].

Орындаушы (actor) – қолданушының жүйеге қатысты ойынындағы рөлі. Орындаушылар рөлдерді береді, яғни ол нақты адам немесе жұмыс атын көрсетпейді. Қолдану варианты диаграммаларында орындаушылар адам фигурасы түрінде келтірілсе де, олар осы жүйеден белгілі бір ақпарат алғысы келетін сыртқы жүйе болуы мүмкін (мысалы, есептеу жүйесі). Орындаушыға белгілі бір орындау варианты керек болғанда ғана, оны диаграммада көрсету

қажет.

Орындаушы (actor) – қолданушының жүйеге қатысты ойынындағы рөлі. Орындаушылар рөлдерді береді, яғни ол нақты адам немесе жұмыс атын көрсетпейді. Қолдану варианты диаграммаларында орындаушылар адам фигурасы түрінде келтірілсе де, олар осы жүйеден белгілі бір ақпарат алғысы келетін сыртқы жүйе болуы мүмкін (мысалы, есептеу жүйесі). Орындаушыға белгілі бір орындау варианты керек болғанда ғана, оны диаграммада көрсету қажет.

Кез-келген қолдану варианты жүйенің функциялығына қойылған сыртқы талаптармен байланыста болады. Айталық, есептеу жүйесі файлды талап етсе, онда ол талап қанағаттандырылуы қажет. Қолдану вариантарын әр уақытта қолданушының нақты есептерін анықтап және оларды шешудің альтернативті әдістерін қарастыра отырып, жүйенің орындаушыларымен бірге анализдеу қажет [23].

Қолдану вариантын иденфикациялаудың көзі болып сыртқы оқиғалар қызмет атқарады. Алдымен жүйе реакция беретін сыртқы әлемде болып жатқан оқиғаларды тізбектеуден бастаған дұрыс. Кейбір нақты оқиғалар қолданушыны араластырмай жүйенің реакциясын тудыруы немесе қолданушының өз реакциясын тудыруы мүмкін. Реакция беруді қажет ететін оқиғаларды идентификациялау қолдану вариантарын айқындауға көмектеседі.

Орындаушы және қолдану вариантары арасындағы байланысқа қосымша, «қолдану» (uses) және «кеңейту» (extends) деген байланыстың екі типі бар. Қолдану вариантының бірі екіншісіне ұқсас, бірақ одан көбірек жүктеме алған жағдайда «кеңейту» типті байланыс қолданылады.

Жүйе тәртібінің бір фрагменті бірнеше қолдану нұсқасында қайталанса, онда оның сипаттамасын әр нұсқаға көшірмеу үшін «қолдану» байланысы қолданылады. «Кеңейту» және «қолдану» байланыстарының арасындағы ұқсастық пен айырмашылықтар. Бұл екеуі де бірнеше қолдану нұсқаларының ортақ тәртіптер фрагменттерін жеке нұсқаға бөлектеп, оны басқада нұсқаларға «қолданады» немесе оларды «кеңейтеді». Екінші жағынан, бұл жағдайлар оны әртүрлі мақсатта жасайды [23].

Қолдану нұсқалары программалық қамтамаға талаптарды құру сатысындағы қажетті құрал болады. Әрбір қолдану нұсқасы – бұл жүйеге қойылатын потенциал талаптар және оларды айқындамай жүйені іске асыруды жоспарлау мүмкін емес.

Жүйені құру үшін зат облысын білу қажет. Сонымен қатар жүйенің қалай жұмыс істейтіні туралы хабарлар болу керек. Диаграмма жүйесі орындайтын операция тізімін құру үшін арналған. Ол берілген функциядағы жүйе объектісі анықталған тізімдер жүйесін орындайды. Осындай түрде орындалатын жүйенің функция жүйесі құрылады және жүйемен қарым-қатынастағы объектілер әрекетінің сценарийлары жазылады. Прецедент процесін модельдеу үшін қолданушы іс-әрекетін толық көрсетуіне және толық реакциясына бағытталған. Себебі, іс-әрекеті олардың шарттарына негізделген.

Жүйенің жұмысы онымен қалай айналысады және не қажет ететіне байланысты.

Программалық жүйенің (БЖ) келесі сатысы прецеденттер диаграммасы болып табылады [22].

Прецеденттер диаграммасында прецедент элементтері, актерлері мен олардың қатынастар көрсетеді. Прецеденттер диаграммалары көмегімен жүйе үшін прецеденттің статикалық көрінісі құрылады. Бұл ПЖ-нің талаптарын алдын-ала модельдеу және ұйымдастыру.

Прецеденттер диаграммасы қолданушы көзқарасы бойынша жүйе тәртібін анықтайды. Прецедент диаграммасы жүйе динамикасын алғашқы модельдеуі үшін басты құрал ретінде қарастырылады, өндірілетін жүйелерге талаптарды анықтау қолданылады, кейінгі өндірулерді жүргізуге мүмкіндік беретін формаға бұл талаптарды тіркейді.

Прецедент элементтері мен актерлері прецедент диаграммаларының негізі болып табылады. Олардың белгілемелері 2.2-суретте көрсетілген [23].

Актер Прецедент элементі

2.2-сурет – Прецедент элеменеттері

Актер (субъект) – бұл нақты элемент (прецедент элементі) пен тікелей өзара әрекеттесетін жүйе сыртындағы объект рөлі.

Прецедент элементі – бұл жүйемен орындалатын және бөлек актер үшін көрінетін нәтижені өндіретін әрекеттер кезектілігінің сипаты.

Прецеденттер (прецедент элементі) типті, тәуелділік байланысын көрсете алады:

– include – қосылатын байланыстар;
– extend – кеңейтілген байланыстар.

Қосылатын байланыс – сол функцияның орындалғанынан басқа прецеденттің анықталғандығын және осы прецедентке басқа да бір прецеденттердің біріккенін көрсететін байланыс.

Кеңейтілген байланыс – шартты емес жағдайда туындайтын міндетті емес функцияның жиынтығынан туындайтын байланыс.

Программалық қамтаманы құру белгілі бір өмірлік циклге (ӨЦ) сүйенеді.

Программалық қамтаманың (ПҚ) өмірлік циклы ПҚ-ға сұраныс туған уақытынан бастап, оның өндірістен алып тасталғанға дейіңгі уақыт аралығы яғни осы аралықтағы ПҚ-ны құру, қадағалау процестерінінің уақыт аралығы. Бір сөзбен айтсақ адам өмірі сияқты. ПҚ-ның ӨЦ-лы халықаралық стандарт ISO12207-пен реттеледі. Бұл стандарт құрылатын ақпараттық жүйенің

процестеріне жалпылай нұсқаулар береді. ISO12207 стандарты бойынша ПҚ-ның барлық процестері негізгі 3 топқа бөлінеді [22]:

а) негізгі процесстер – бұған кіретін жұмыс процестері программаны алу, құру, эксплатациялау және жұмысын ары қарай қадағалау;

ә) көмекші процесстер – бұл негізгі процестің орындалуын қамтамасыз етеді. Оған кіретіндер: құжаттау, тестілеу, верификациялау, бағалау т.б. (верификациялау – ПҚ құру кезінде белгілі кезеңдердің талаптарына ПҚ-ң нәтижелері сәйкес келетіндігін тексеру);

б) ұйымдастыру процесстері – жобаны басқару, жобаның инфра құрылымын анықтау, жобаның өмірлік циклын дәлдеу (нақтылау), жақсарту, оқыту.

Жекеленген ӨЦ-діні жеті сатыға бөлуге болады [22]:

– талаптарды орналастыру;
– талаптар спецификациясы;
– архитектураны жобалау;
– жекеленген жобалау;
– тарату;
– интеграциялау;
– қатыстыру.

Біздің мақсатымыз бірінші сатыны толық қарастыру – яғни талаптарды орналастыру сатысы. Бұл сатыда программалық жүйеге, қолданушылардың ұсынған талаптары қарастырылады:

– жүйе ақпаратты іздеуге және өңдеуге кеткен уақыт мөлшерін азайту қажет;

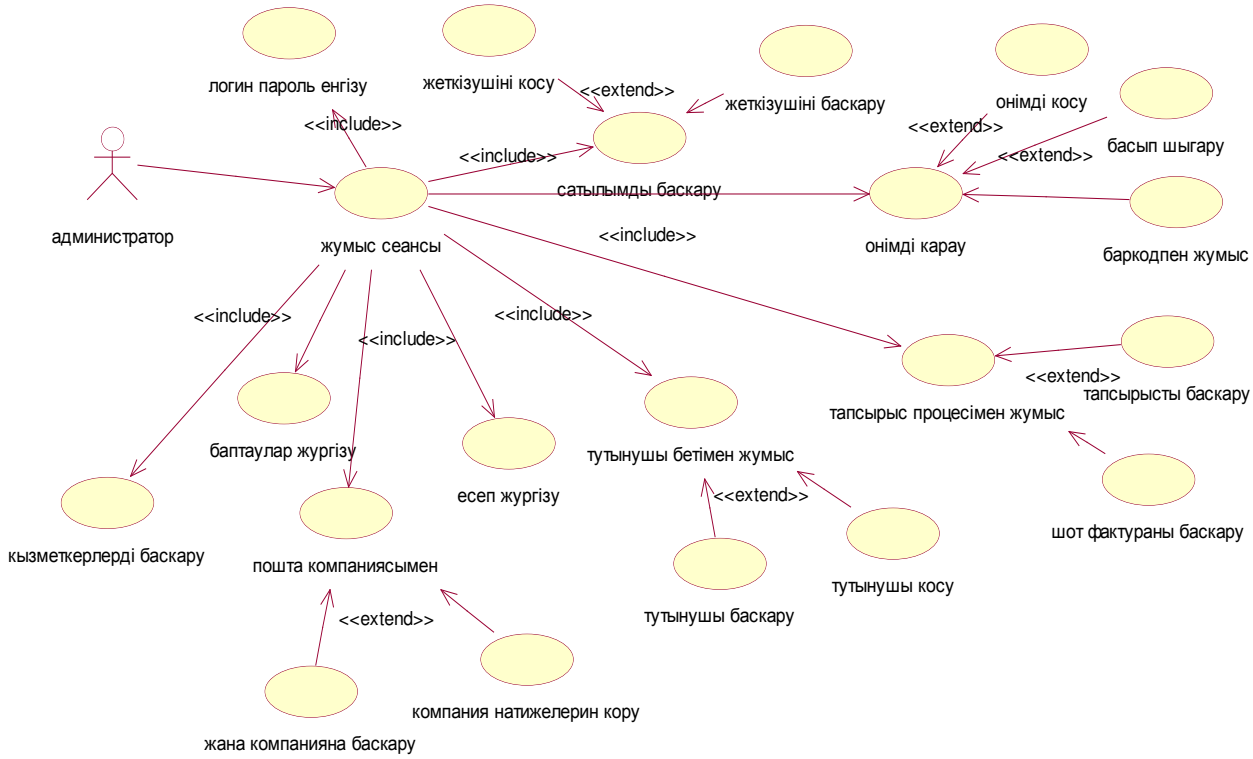
– қолданушы компьютерді минимальді түрде білгендіктен программалық жүйені оңай қолдана алу мүмкіндігі;

– деректер қорын мәліметтермен толықтыру, өзгерту, өшіру және таңдаған мәліметтің есебін алу мүмкіндіктерінің болуы;

– бірнеше қолданушы бір мезгілде жұмыс істей алатын клиент-серверлік, желі арқылы байланыста болу мүмкіндігі. Прецеденттер диаграммасы 2.3-суретте көрсетілген.

Қолдану варианттары (Use Case) жүйе мен активті субьект арасындағы диалогты модельдеуге мүмкіндік береді және функцияны соңында бейнелейді. Жүйені қолдану варианттар жиыны оны қолданудың көптеген тәсілдер назарына еңбегі сіңген. Қолдану варианттары – транзакция жүйесімен орында латын тізбектілік, бұл арқылы анықталған активті субьект қызыққан нәтижені алуға болады.

Қолдану вариантының диаграммасы (Use Case Diagram) – бұл активті субьектілердің көптеген графикалық көрсетімі оны қолданудың, сол немесе басқа варианттардың амалдарымен өзара әрекеттеседі. Жүйені жоспарлағанда жүй енің кілттік функциясын және көптеген қолданушыларды ұсынатын негізгі диаграмма (Main Use Case Diagram) конструкцияланады [23].



2.3-сурет – Rational Rose ортасында құрылған прецеденттер диаграммасы

2.4 Тізбектер диаграмасы

Тізбек диаграммаларда объект төбесінде пунктирлі вертикаль сызығы бар төртбұрыш түрінде бейнеленеді.

Тізбек диаграммалары өте қарапайым және көрнекі (олардың негізгі артықшылығы) және жүйе тәртібі процесін түсінуге көмектеседі. Тізбек диаграммасында қайтару бар болуы мүмкін, ол белгілі бір хабардан қайтарудың болғанын көрсетеді. Диаграммада қайтарудың бағыттауышы жұп сызықты болуы, оны қарапайым хабардан ерекшелейді [23].

Тізбек диаграммасына жаңа элементтер қатары енгізілуі мүмкін Біріншісі, әдістің не орындалу барысында, не белгілі бір процедураның орындалу нәтижесін күткен уақытында пайда болатын белсендеу.

Тізбектер диаграммасы төрт негізгі элементтерден тұрады:

а) прецеденттегі ізбасар мәтінінің іс-әрекеті. Ол сол жақтан жоғарыдан төменге жазылады. Сол терезеде іс-әрекет сипатталуы болып, жұмыс уақытындағы орындалатын ақпараттар қызмет етеді.

ә) объекттер «объект-класс» форматында аты немесе объект данасының нөмірі және класс объектісінің аты жазылады.

б) хабарландыру. Бағытпен көрсетілген бір объектіден келесіге бағытталған іс-әрекет туралы ақпарат жолдамасынан тұрады. Белгілі бір уақытта орындалуы және осы іс-әрекеттегі жүйенің жауапты реакциясы болуы мүмкін.

в) әдістері (операциялар). Тікбұрыш түрінде көрсетілген. Олар үздік

сызықта орналасқан. Яғни, әдістерге кіретін сол объектілер келеді. Тікбұрыш ұзындығы ізбасарда басқару фокусын көрсетуде қолдануға болады. Тікбұрыш бітетін әдіс түгелдей нүктесіне дейін басқарумен иеленеді. Бұл үшбұрыштар объекті түзуі деп аталады.

Екі объектінің арасындағы әрбір хабар олардың өмірлік сызықтары арасындағы бағыттауыш түрінде көрсетіледі. Хабарлар жоғарыдан төмен қарай шығарылады. Әрбір хабар минимум дегенде хабар атымен белгіленеді. Қалауыңызға қарай аргументтер, белгілі бір басқарушы ақпараттарды қосуға болады және сонымен қатар өзін делегациялауды (self-delegation) көрсетуге болады, яғни объект өзіне өзі хабар жібереді, әрі бұл кездегі хабар бағыттауышы бір ғана өмірлік сызығын көрсетеді. Барлық мүмкін болатын басқарушы ақпараттардың ішіндегі келесі екі түрдің мәні зор.

Тізбек диаграммалары өте қарапайым, көрнекі (олардың негізгі артықшылығы) және жүйе тәртібі процесін түсінуге көмектеседі [23].

Тізбек диаграммасында қайтару бар болуы мүмкін, ол белгілі бір хабардан қайтарудың болғанын көрсетеді. Диаграммада қайтарудың бағыттауышы жұп сызықты болуы, оны қарапайым хабардан ерекшелейді.

Тізбек диаграммасына жаңа элементтер қатары енгізілуі мүмкін Біріншісі, әдістің не орындалу барысында, не белгілі бір процедураның орындалу нәтижесін күткен уақытында пайда болатын белсендеу.

Тізбектелген диаграммасы уақыт бойынша объектілердің өзара бірлесу операцияларының орындалу ретін безендіреді және сценарийлермен қарастырылған функциялардың орындалу процесіндегі алмасатын объектілердің хабарламалар тізбектерінің реттелген сценарийге кіретін объектілер мен класстар бейнеленеді. Тізбектелген диаграммалар жалпыда Logical View пакетіндегі есептелінген қолдану варианттарының жүзеге асырумен бірге ассоциаланады.

Өзара әрекеттесу диаграммасының екі түрі бар – тізбек диаграммасы мен кооперация диаграммасы. Тізбек диаграммасы – бұл хабарламаларды уақыт бойынша тәртіпке келтіруді айқындайтын өзара әрекеттесу диаграммасы, яғни бір уақыт мезетіндегі іс-әрекетті жобалау. Кооперация диаграммасы – бұл хабарламаларды жіберетін және қабылдайтын объекттерді құрылымдық ұйымдасуын айқындайтын өзара әрекеттесу диаграммасы. Өзара әрекеттесу диаграммаларының элементтері – объекттер, байланыстар, хабарламалар болып табылады.

Өзара әрекеттесу диаграммасы объектілер мен олардың арасындағы қатынастар, сондай-ақ объектілер арасындағы бір-біріне жіберілетін хабарламалар жиынын қосатын өзара әрекеттесулер. Өзара әрекеттесу диаграммалары жүйенің динамикалық көрінісін беруді қамтамасыз етеді.

Тізбек диаграммалары жүйеде тәртіп сценарийін бере отырып, бұл диаграмма хабарларды беру тәртібінің анағұрлым айқын көрінісін қамтамасыз етеді. Бірақ ол кооперация диаграммасында көрінетін бөлшектерді көрсетуге мүмкінді береді [22].

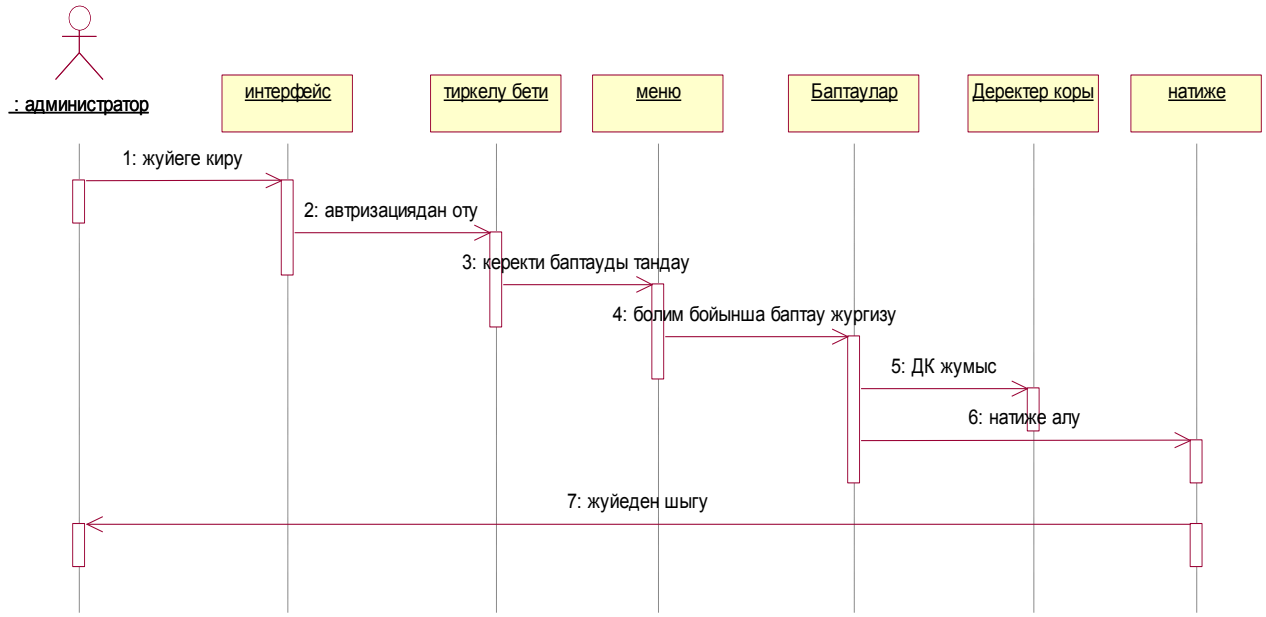
Графикалық түрде тізбек диаграммасы Х осі бойымен объекттерді, У осі

бойымен уақыт бойынша келтірілген хабарламаларды көрсетеді.

Тізбек диаграммасында тіктөртбұрыш түрінде берілетіндер – әдістер (операциялар).

Өзара әрекеттесу (Interaction) – бұл тәртіптің жұмысы, әрекеттесетін объектілер арасындағы белгілі бір мақсатқа жету үшін хабарлар алмасу. Графикалық түрде бағдарша (→) түрінде болады. Ал не істеу керектігі жөнінде бағдаршаның үстіне жазылады.

UML талаптарымен сәйкес объект тізбектелген диаграммасында тік төртбұрыш түрінде көрсетіледі. Ол асты сызылған объектінің атауын құрайды. Жоғарыда көрсетілгендей объектіні 3 түрлі әдіспен атауға болады: тек оның атауын көрсету, объект пен класстың атауын беру, не класстың атауымен шектелу қажет (жасырын объект үшін). 2.4-суретте тізбектер диаграммасы корсетілген [21].



2.4-сурет – Rational Rose ортасында құрылған тізбек диаграммасы

Тізбектелген диаграммада активті субъектілермен өзара бірлесу фактілер жүйесінің көрсетілуін рұқсат ететін шекаралар класы қосылады (қолданушылармен және басқа жүйелермен). Осындай қабылдау анализдің ерте сатыларында бекітілуін талап етеді және интерфейстер талаптарының құжатталуына рұқсат етеді. Активтік субъектідегі алынған шекаралар класының нақты құралған хабарламалардың реттелген операциялардың әдістері жөніндегі ақпаратпен бірге өңдеуінің қосымша ортасын қолдану ерекшеліктерімен шартталған. Ал ондай ортаның таңдалуы жобалаудың соңғы кезеңінде жүзеге асырылады. Сондықтан жүйенің даму барысына сәйкес және осындай детальдар сияқты сұрақтардың көп санына сәйкес жауаптың алуы өзгертіледі және дәлірек болады.

2.5 Кооперация диаграммасы

Кооперация диаграммасы – бұл хабарламаларды жіберетін және қабылдайтын объекттерді құрылымдық ұйымдасуын айқындайтын өзара әрекеттесу диаграммасы.

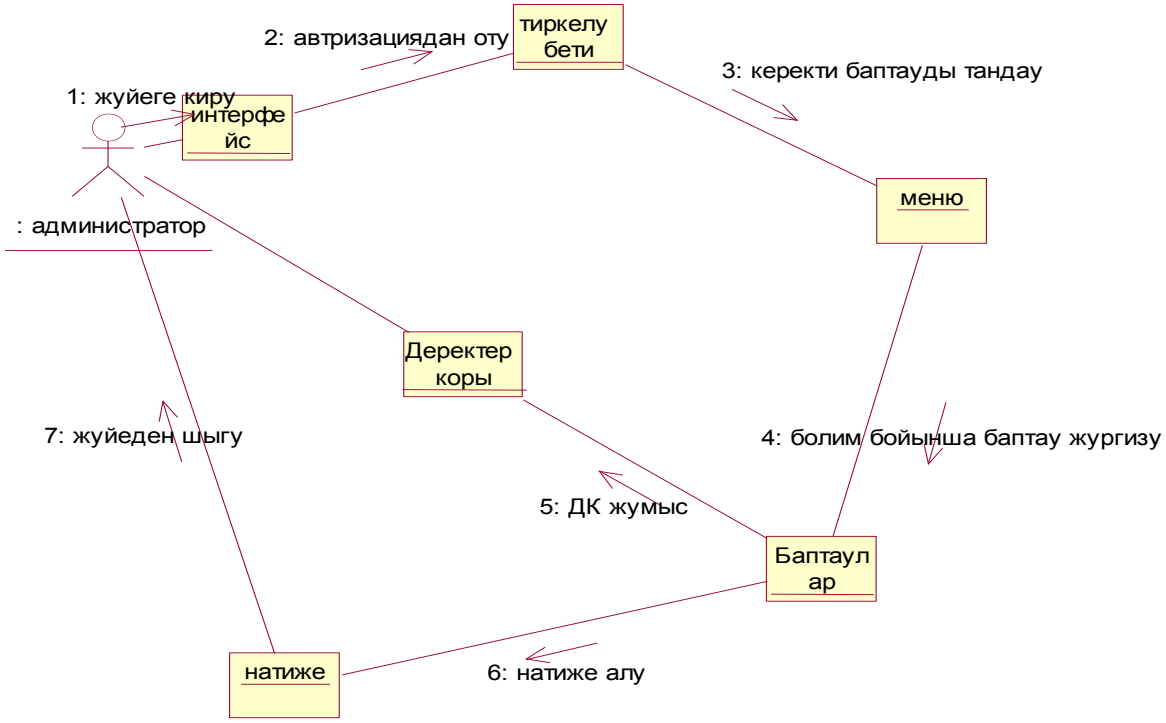
Кооперация диаграммалары жүйе жұмысы барысында объектілердің өзара әрекеттесуін бейнелейді. Мұндай диаграммалар жүйе тәртібінің сценарийлерін моделдейді. Объект аты сызылады және әрдайым беріледі, ал қасиеттері таңдаулы түрде көрсетіледі. Объект белгісі 2.5-суретте көрсетілген [21].

2.5-сурет – Объект белгісі

Кооперациялар (әріптестіктер) ПҚ-ны жоғарғы, архитектуралық деңгейде өндірудегі кешенді шешімдерді көрсететін құрал болып табылады. Бір жағынан, кооперациялар программалық өнімнің мақсаты спецификациясының жинақтылығын қамтамасыз етсе, екінші жағынан – басқару және деректер ағындарының, сондай-ақ деректер құрылымының жүзеге асырылуынан құралады. Кооперацияның статикалық құрамасы бірлесіп жұмыс істейтін кластар мен басқа элементтердің (интерфеистер, компоненттер, түйіндердің) құрылымын береді. Көп жағдайда ол үшін бір немесе бірнеше кластар диаграммаларын қолданады. Кооперацияның динамикалық құрамасы бірлесіп жұмыс істейтін элементтердің тәртібін анықтайды. Әдетте анықтау үшін бір немесе бірнеше тізбектілік диаграммаларын қолданады.

Тізбек диаграммаларымен кооперация диаграммалары изоморфты, яғни бұл бір диаграмманы басқа диаграммаға тасымалдауға болатынын білдіреді. Кооперация диаграммасы 2.6-суретте көрсетілген [21].

Осылайша, егер кооперация «қаптамасының» астына қарасақ, әртүрлі диаграммалар жиынтығын көреміз. Мысалы, ақпараттық жүйеде талаптар Use Case элементтерінің әрбіреуі бөлек кооперацияда жүзеге асырылатын Use Case элементтерінің жиынымен беріледі.



2.6-сурет – Rational Rose ортасында құрылған кооперация диаграммасы

2.6 Класстар диаграммасын құру

Класстар диаграммалары класстар және интерфейстер, кооперация және олардың қатынастарын көрсетеді. Объектілі-бағытталған жүйелерді модельдеу кезінде кластар диаграммалары өте жиі қолданылады. Класстар диаграммалары жүйенің статикалық жобалық көрінісін қамтамасыз етеді. Пәндік облыстан объектіні анықтау барысында олардың арасында қандай байланыстар бар екендігін анықтау қажет. Агрегация қатынасы (бүтін мен бөлік арасындағы қатынас) және жалпылау (ішкікласс пен суперкласс арасындағы қатынас) өте маңызды байланыстар болып табылады. Статикалық модельдің негізіне пәндік облыстың моделін бейнелейтін диаграммасын қоямыз.

Кластар диаграммасы объектілі-бағдарланған тәсілдің орталық буыны болып табылады. Кластар диаграммасы жүйе объектілерінің типтерін және олардың арасындағы болатын түрлі статикалық байланыстарды анықтайды [22].

Статикалық байланыстың негізгі екі түрі бар:

– ассоциациялар (мысалы, клиент тапсырыс бере алады);
– ішкі типтер (жеке клиент клиенттердің белгілі бір түрі болады).

Кластар диаграммасында кластар атрибуттары, кластар операциясы және объектілер арасында қойылатын шектеулер бейнеленеді.

Кластар диаграммасын сипаттаудан бұрын, осы диаграммаларды құрастырушының қолдану сипатына байланысты сәтке назар салған жөн. Бұл сәт құжаттандырылмайды, бірақ ол диаграммаларды интерпретациясы

тәсіліне әсер етеді, сондықтан моделдер көмегімен сипаттауға оның қатысы маңызды.

Клас (Class) ортақ қасиетттері (атрибуттары), тәртібі (функциялары), семантикасы және басқа объектермен байланысы бар объектер тобын анықтайды. Кластың объектіні құруға арналған шаблон ретінде қарауға болады. Әрбір объект қандайда бір ғана кластың нұсқасы.

Дұрыс құрылған класс тек бір ғана абстракцияны бере алады. Мысалы, студент туралы мәлімет сақталған, сонымен қатар барлық оқу барлығында студент өткен курстар тізімі функциясы көрсетілген класты сәтті құрылған деп айта алмаймыз, өйткені ол екі әртүрлі операциялар тобын қамтиды. Аспект түсінігі кластар диаграммасын тұрғызумен қатар, оны оқуда да үлкен маңызы бар. Бірақ аспектілер арасындағы айырмашылық айқын емес, сондықтан көптеген құрастырушылар диаграмма тұрғызуда олардың ығысу мүмкіндігіне рұқсат етеді [22].

Диаграмма тұрғызуда жалғыз аспектіні таңдап алу қажет. Диаграмманы оқу кезінде оның қандай аспектіге сәйкес тұрғызылғанын білу қажет. Яғни бұл білім бізге диаграмманы дұрыс интерпретациялауда қажет болады.

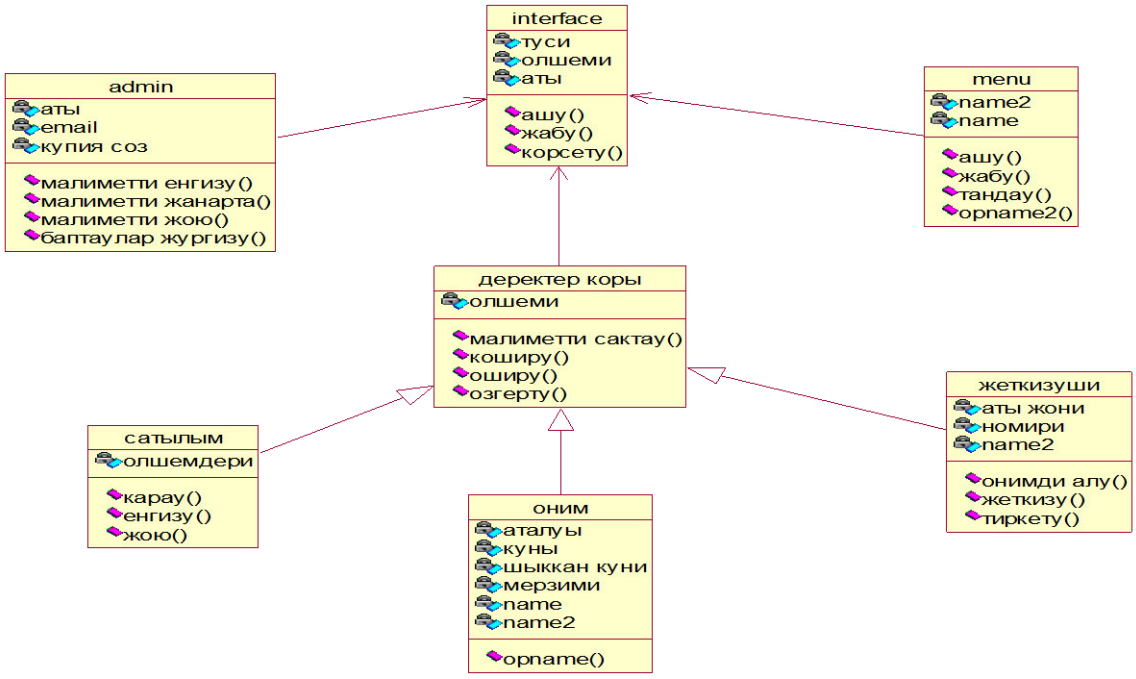
Кластың атауы үшін пән облысына сәйкестендіріліп қабылданған терминдерді қолданған дұрыс. Класс аты ретінде жобаланатын түсінікті толығымен сипаттай алатын зат есіміңіз жекеше түрі қолданылады. Кейде қысқартылған атауларда қолданылып, класты құжаттандырғанда міндетті түрде мағынасын ашып көрсету керек. Егер аббревиатура бірдей мағыналы емес интерпретация жіберсе, онда сәйкесінше айтылудың толық түрін қолданамыз. UML-да класс аймақтарға бөлінген тіктөртбұрыш түрінде кескінделеді. Жоғары аймақта класс аты, ортасында оның құрылымы (атрибуттар тізімі), оның астындағысында тәртібі сипаттамаларын анықтайтын функциялар беріледі.

Кластар диаграммасы UML-дың ресми бөлігіне жатпағанымен модельдерді тұрғызу мен анализдеуде өте пайдалы болады. UML конструкциясын көрудің кез-келген үш нүктесінде қолдануға болады.

Тұжырымдық деңгейде «Клиент аты» атрибутының бар болуы клиенттің аты болатынын көрсетеді. Спецификация деңгейінде бұл атрибут Клиент объектісі өз атын хабарлай алады және оны анықтаудың белгілі бір механизміне ие болады. Іске асыру деңгейінде клиентте оның атына сәйкес өріс болады (айнымалы немесе мәліметтер элементі делінеді).

Диаграмманың тетіктеліну дәрежесіне қарай атрибут белгілеуінде үнсіз келісім арқылы атрибут аты, типі және мәні көрсетіледі.

Кластар диаграммасы 2.7-суретте көрсетілген [21].



2.7-сурет – Rational Rose ортасында құрылған класс диаграммасы

UML синтаксисінде бұл келесі түрде болады:

<көріну белгісі><аты>:<тип>=<үнсіз келісім мәні>

мұндағы, көріну белгісінің мәні келесі тарауларда сипатталатын операциялармен сәйкес болады.

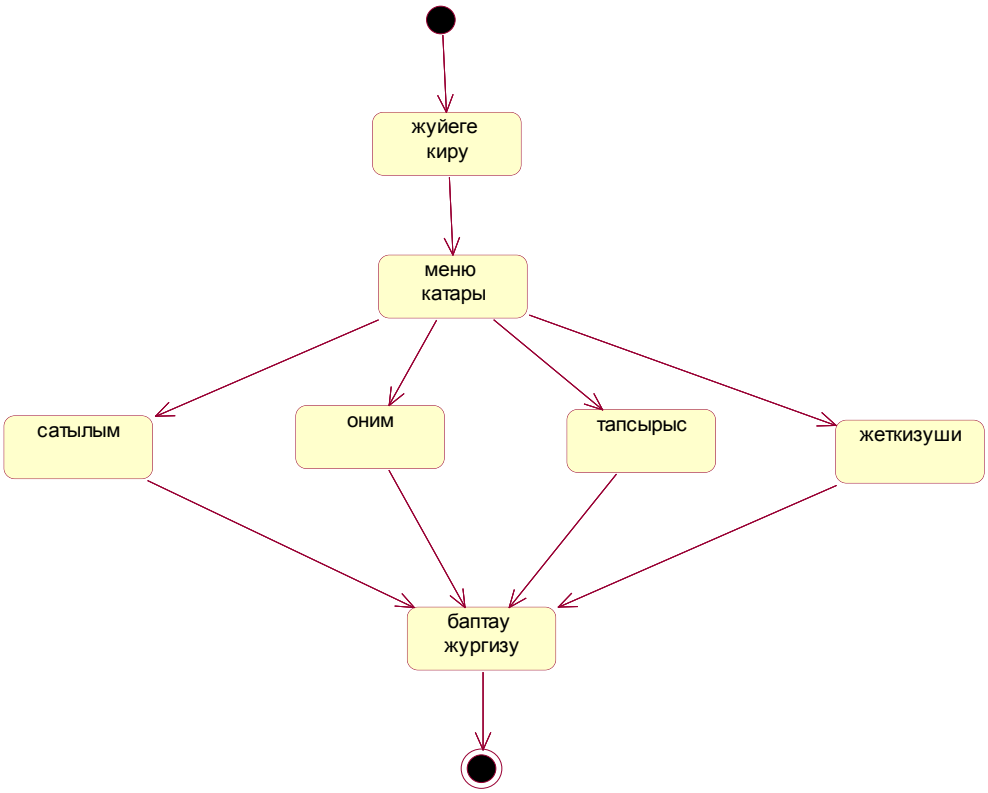
Атрибуттар әруақытта жалғыз мәнге ие болады. Әдетте диаграммада атрибутың міндетті немесе міндетті емес екені көрсетілмейді.

Операция кластар іске асыратын процестерді береді. Операция мен кластарға қолданылатын әдістер арасында сәйкестік болады.

Спецификация деңгейінде операциялар типтерге қолданылатын әдістермен сәйкес келеді. Әдетте диаграммада атрибуттарды манипуляциялау операциялары көрсетілмейді, себебі олар онсызда бар. Кейде берілген атрибут тек оқу үшін (read only) немесе оның мәні тұрақты (immutable) екенін көрсетуге қажет болады. Іске асыру моделінде сондай-ақ операциялардың құпиялық және қорғаныс деңгейі бейнеленуі мүмкін. Ал сайттың қызмет диаграммасы 2.8-суретте келтірілген. Іскерлік диаграммасы – жүйе ішінде әрекеттен әрекетке дейінгі ағынды көрсететін күйлер схемалары диаграммаларының арнайы түрі. Қызмет диаграммалары жүйенің динамикалық көрінісін қамтамасыз етеді. Олар жүйенің функционалдығын модельдеу кезінде өте маңызды және объектілер арасындағы басқару ағынын айқындайды.

Қызмет диаграммасы есептеу процесі мен жұмыс ағындары көрсетілетін соңғы автоматтың ерекше формасы беріледі. Мұнда объекттің жай күйлері емес, орындалатын есептер күйлері – әрекеттер күйлері айқындалады. Бұл кезде есептеу процесі сыртқы оқиғалармен үзілмейді деп тұжырымдалады. Қызмет диаграммалары алгоритмдердің блок-схемаларына өте ұқсас болып

келеді [21].



2.8-сурет – Rational Rose ортасында құрылған күй диаграммасы

3 Жүзеге асыру және тестілеу бөлімі

3.1 Программа кешенін құру

Дипломдық жобаға берілген тапсырма бойынша мекеменің ақпараттық бөліміне программалық кешенін жасауды ұйымдастыру үшін қойылымын дұрыс құрып алу қажет. Берілген тапсырманың бастапқы моделі дұрыс қалыптаспаған жағдайда жобаның жоспары нәтижесіз қалуы мүмкін. Сондықтан есептің қойылымына тоқталып кетейік.

Жоғарыда айтылғандай ақпараттық технологиялар мен компьютерлік программаларды адамзат қызметінің әртүрлі салаларында қолдануға болады. Мысалы, мекеменің ақпараттық бөлімінің мәселесі тиімді шешіледі. Осының негізінде ақпараттық бөлімінің брондау жүйесі қарастырылып, брондау қызметшісінің жұмысын жеңілдетуге ұсынылады. Осы түрдегі қосымшалар әртүрлі технологияларда орындалуы мүмкін. Бұл – PHP, MySQL, XHTML және т.б. Бұл технологияларды таңдау қойылған нақты мақсаттарға байланысты шешілді.

Берілген дипломдық жобада келесі мәселелерді шешу қажет:

– компанияның ақпараттық бөлімінің құрылымын жасау;
– таңдалған программалық құралды дәлелдеу;
– тиімді, ыңғайлы қолданушылық интерфейс құру.

Сонымен қатар қосымшаның серверлік және клиенттік бөлімінің модульдер кешенін құру құрамында:

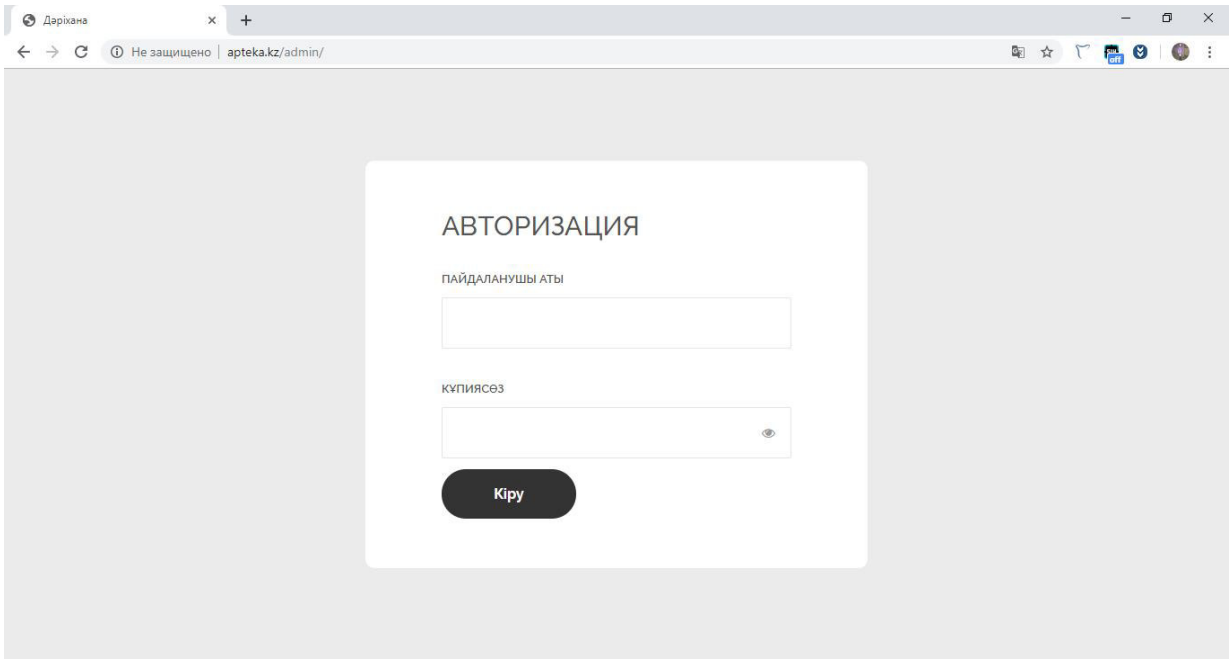
– интерфейстік модуль;
– білімді бақылау модулі;
– транзакцияны өңдеу модулі;
– авторизация және санкциялық қатынау модулі;
– сақталынатын Web-ресурстар құрылымы;
– қолданушы модулінің құрылымы.

Жалпы жасалынатын программалық кешен әртүрлі программалық-аппараттық платформаларда қолдануға дайын болу керек. Қосымша қолданушы үшін қажет көмектен және қызмет көрсетуден тұру керек. Қолданушы интерфейсі және дизайні қазіргі заманғы талаптарға сәйкес болу керек.

Ақпараттық технология негізінде бұл кешен толықтай басқаруға және электронды фармацевтика жобасының бақылауын жоғарылатады. Жүйе жедел тасырмаларды орындау негізінен жұмыс жылдамдығын жоғарылатуға мүмкіндік береді. Программалық қамтамасыздандудың стандартты функциясы (базалық модуль, сатулар, әкімшілік, бухгалтерия, статистика және есеп, іс-шаралар) мекеменің барлық қызметін автоматтандыруға мүмкіндік береді.

3.2 Backend бөлмі

Программалық кешенді іске қосу үшін браузерді ашып адрестік жолға http://apteka.kz/admin/ деп теру қажет. Сол кезде браузердің терезесінде төменде берілген 3.1-суретте ашылады, яғни программалық кешенінің администраторлық бөлімінің тіркелу беті.



3.1-сурет – Администратордың тіркелу беті

«Дәріхана» программалық кешені басты бет, сатылымды басқару, өнім, тапсырыс процесі, есеп, пошта компаниясы, баптаулар, қызметкерлерді басқару бөлімдерінен тұрады. Администратор тапсырыс процесі бөлімінде шот фактураны баптау беті арқылы тапсырыс жөніндегі толық мәліметті шығара алады.

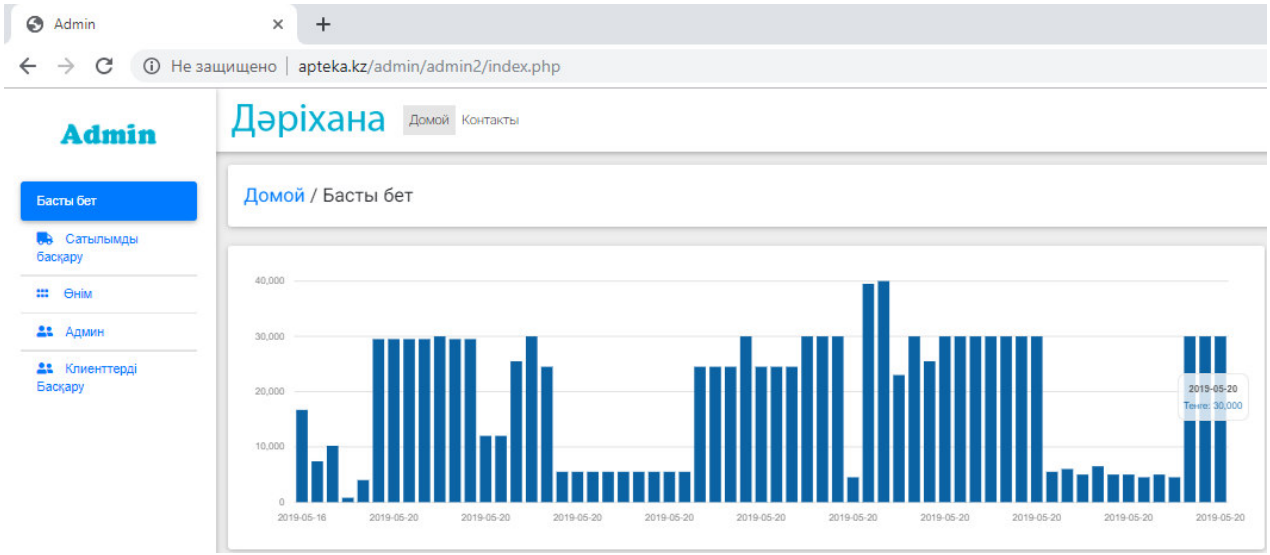
Басты бетте сатылым есебін көруге болады. Сатылым есебі бетінде қандай да бір аралықта қаншалықты жалпы табыс, жалпы салық және жалпы құн өзгеріске ұшырағанын 3.2-суреттен көре аламыз.

Сонымен қатар, администратор сатылымды басқару бөлімінен тауарды жеткізушіні қосып және оны басқара алады. Жеткізуші жөніндегі толық мәліметті енгізеді. Жеткізушіні қосу және басқару төмендегі 3.3 және 3.4-суреттерде келтірілген.

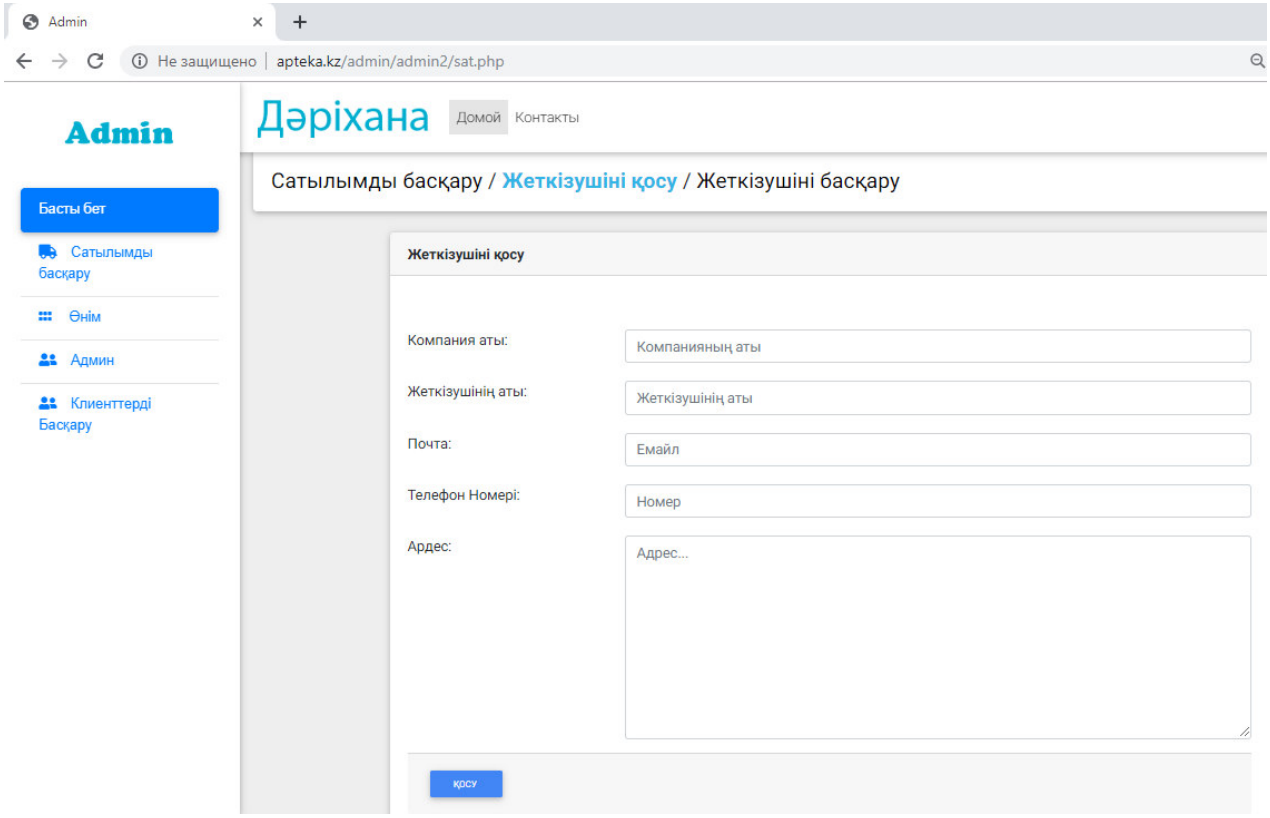
Автоматтандырылған жұмыс орнындағы администратор өнімді қосып, басқара алады. Ол өнімнің сатылымда бар не жоқтығын, оның статусын белсенді, сол сияқты өнімді таңдау және басқа да іс-әрекеттер орындай алады. Бұл іс-әрекеттердің барлығы 3.5 және 3.6 суреттерде көрсетілген.

«Дәріхана» жұмыс орны кешеніне администратордың клиенттерді басқару мүмкіндігі 3.8-суреттегідей.

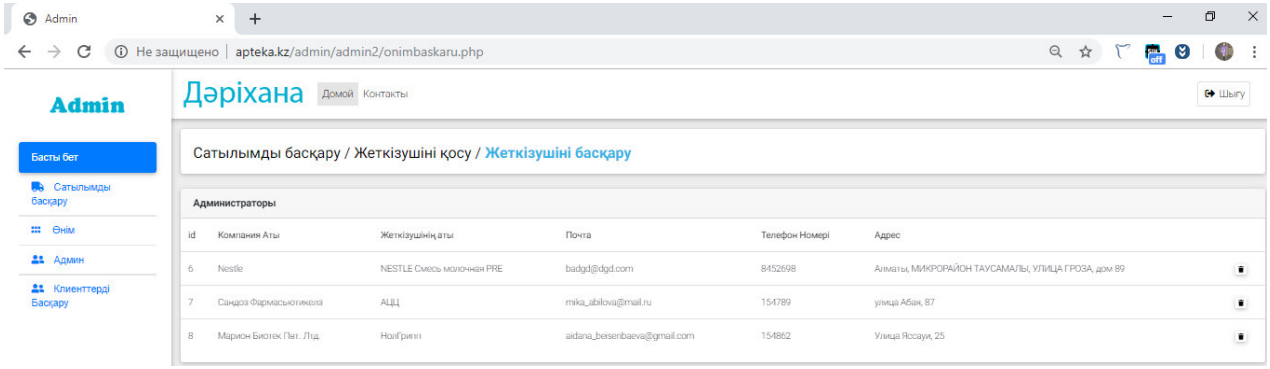
Сонымен қатар, программалық кешеннің бөлімдерінің ішінде администратордың қолданушыны басқару және құра алатын құзыреттілігі бар.



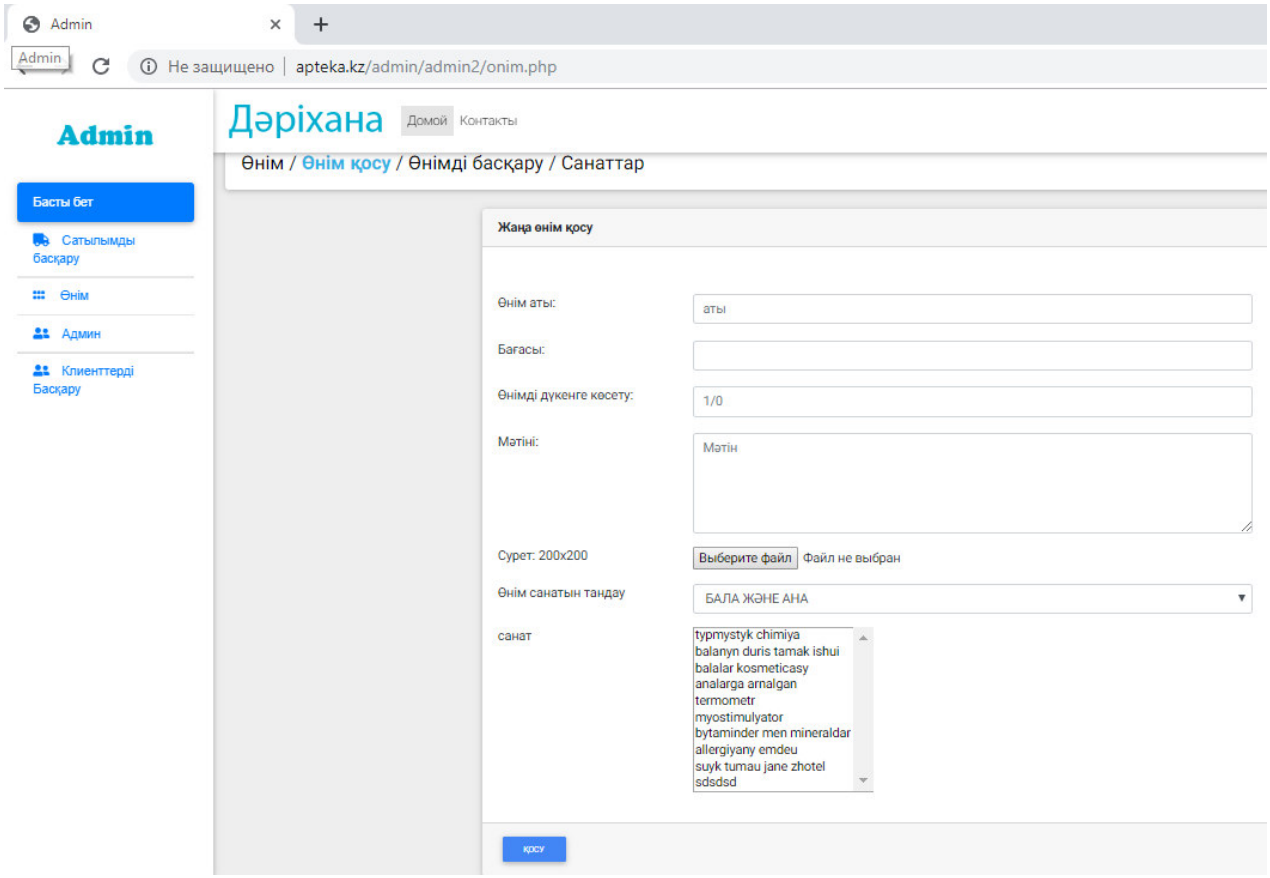
3.2-сурет – Администратордың басты беті



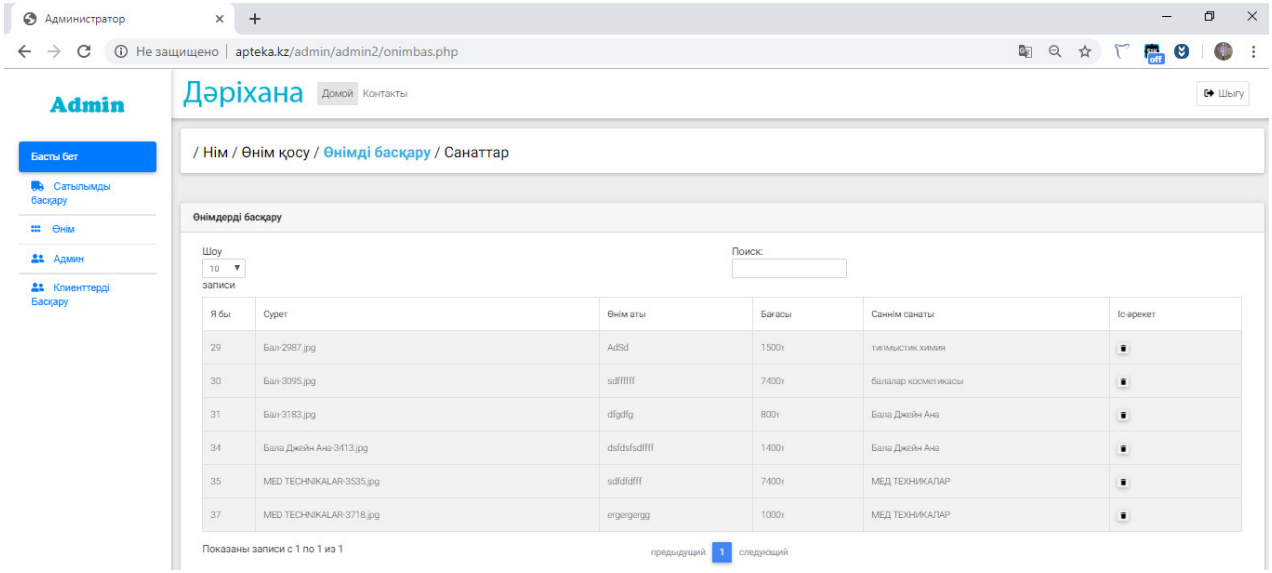
3.3-сурет – Жеткізушіні қосу беті



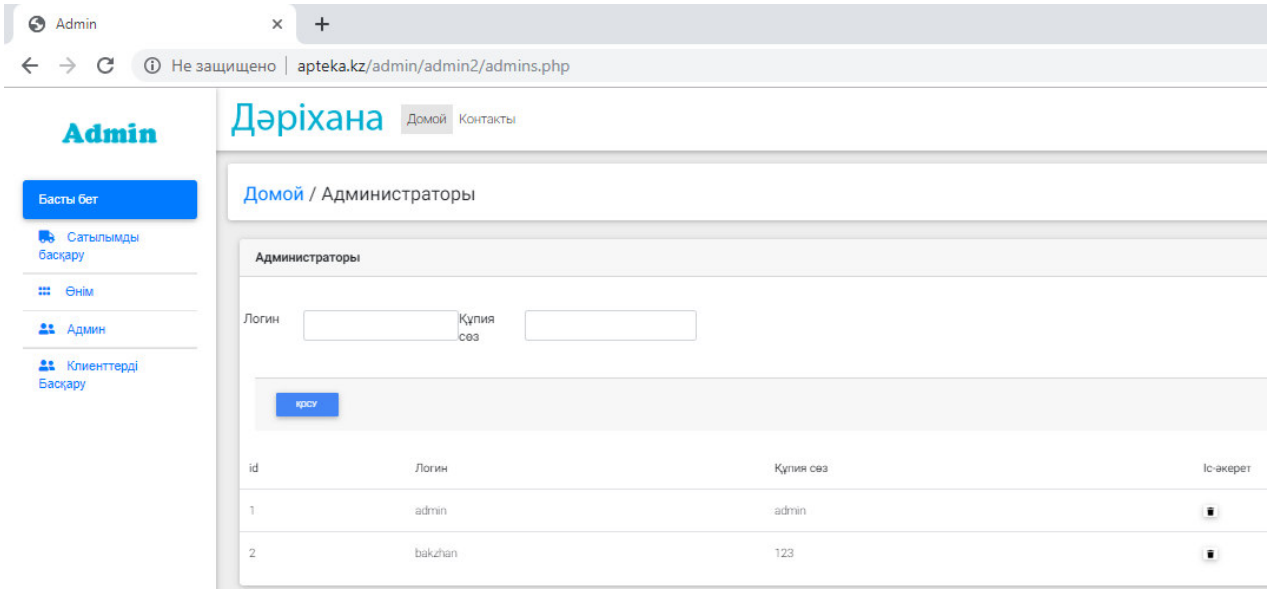
3.4-сурет – Жеткізушіні басқару беті



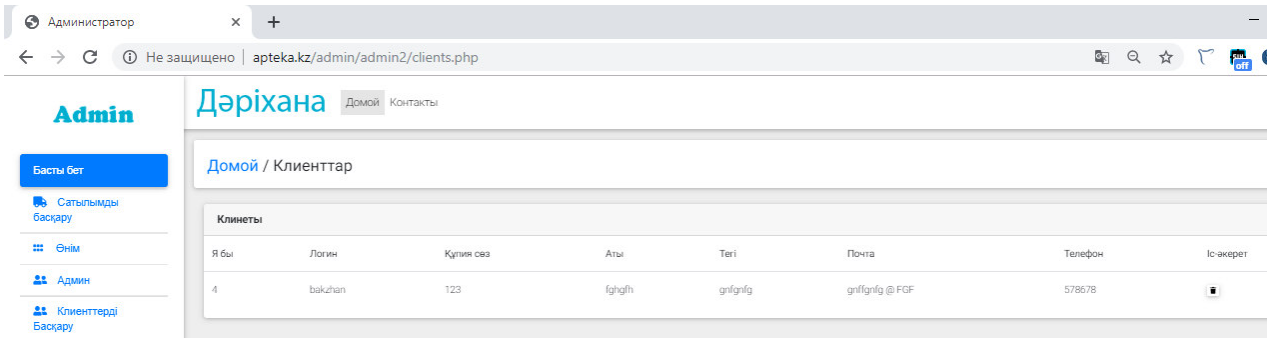
3.5-сурет – Жаңа өнім қосу беті



3.6-сурет – Өнімді басқару беті



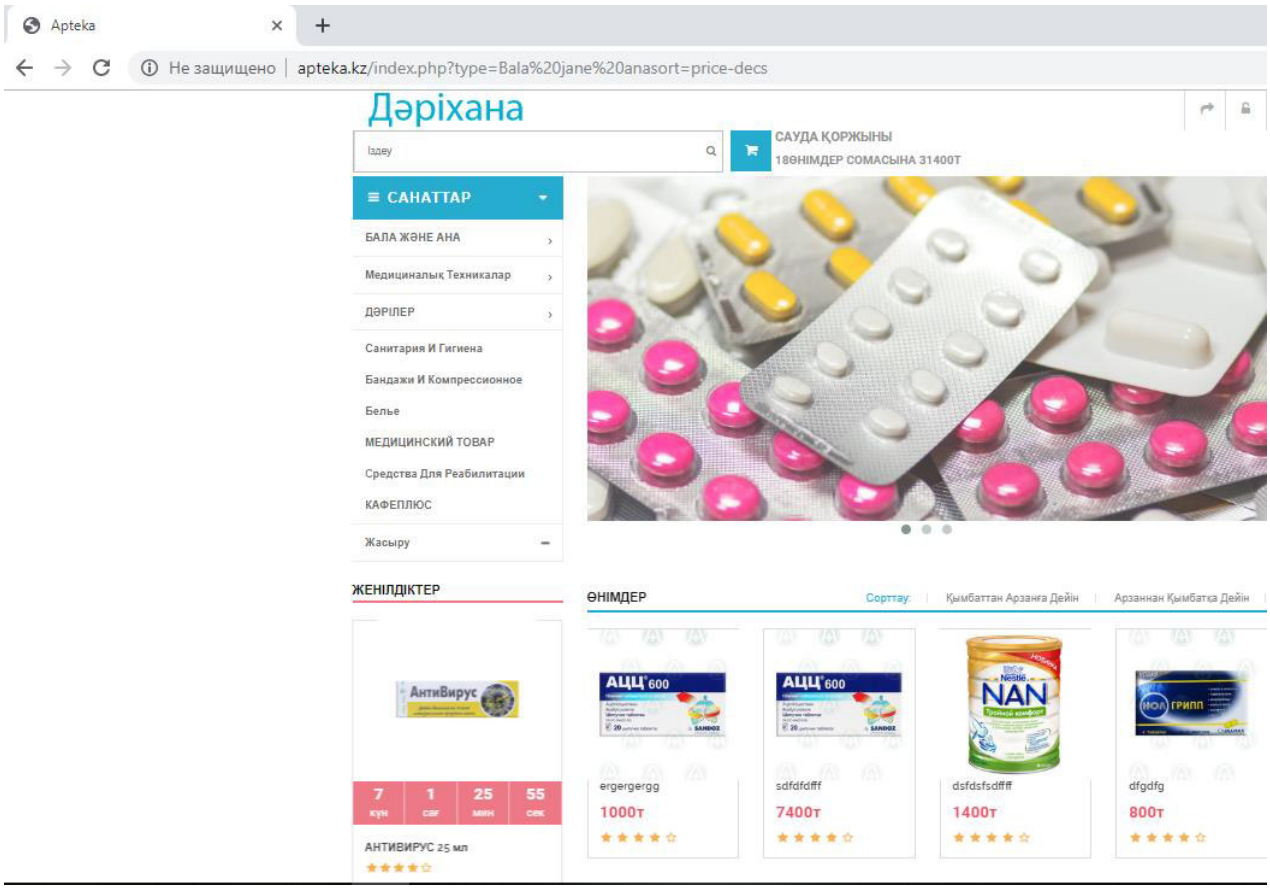
3.7-сурет – Администраторы беті



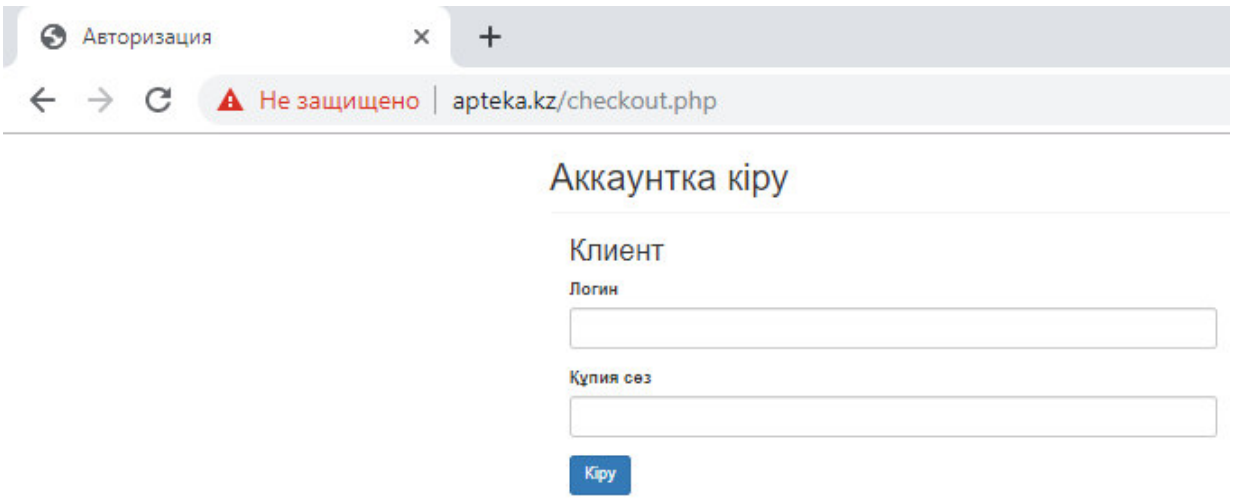
3.8-сурет – Клиенттерді басқару беті

3.3 Frontend бөлімі

Программалық кешенді іске қосу үшін браузерді ашып адрестік жолға http://apteka.kz деп теру қажет. Сол кезде браузердің терезесінде төменде берілген 3.9-суретте ашылады, яғни программалық кешенінің клиенттік бөлімінің басты беті.



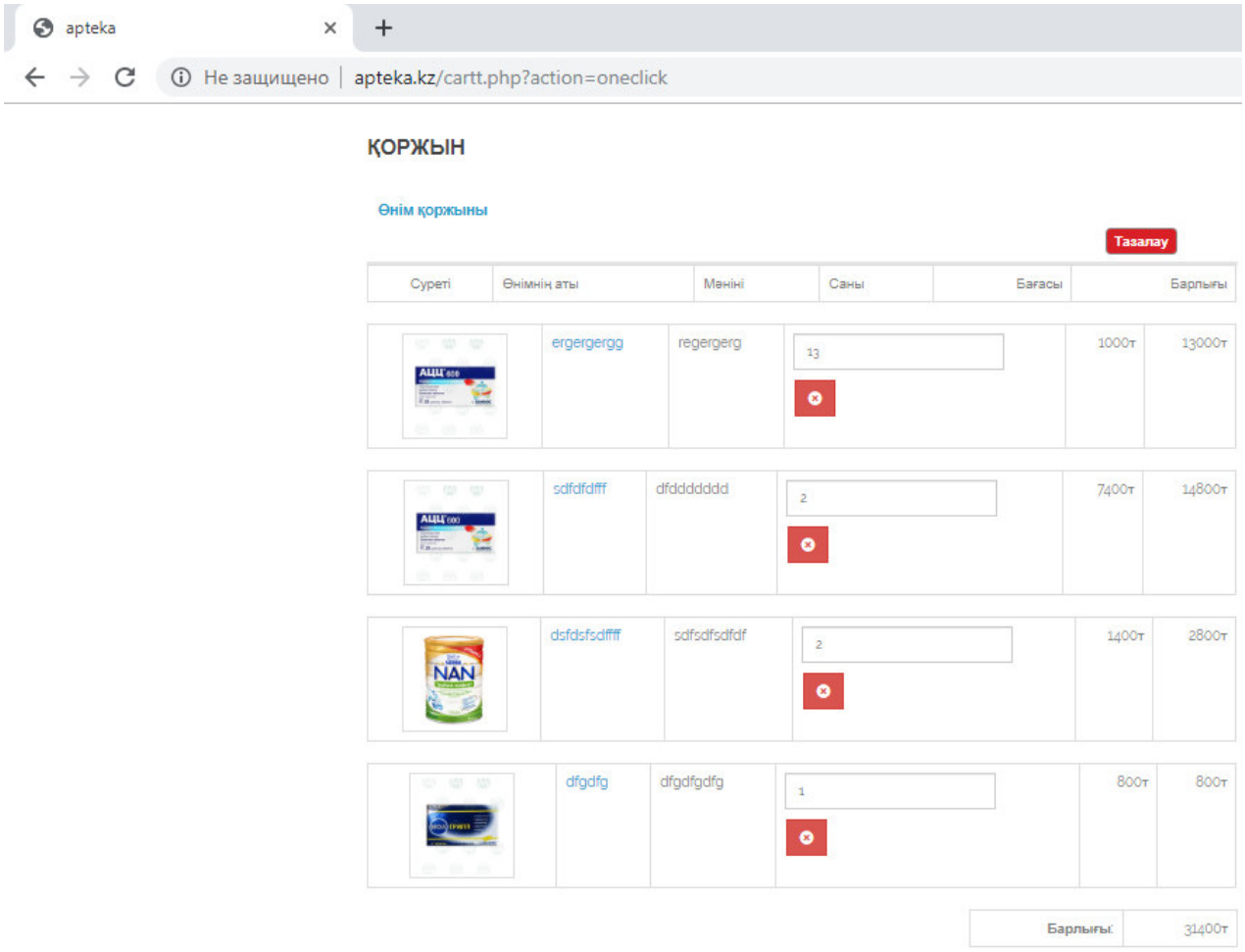
3.9-сурет – Клиенттің басты беті



3.10-сурет – Клиенттің тіркелу беті

«Дәріхана» программалық кешені санаттар, қоржын, іздеу, жеңілдіктер бөлімдерінен тұрады. Клиенттер санаттар арқылы кез келген тауарды көре алады.

Қоржында сақталған дәрі-дәрмектердің тізімін 3.11-суреттен көре алады.



3.11-сурет – Қоржын беті

4 Экономикалық бөлім

4.1 Жұмыстарды ұйымдастыру және жоспарлау

Бұл бөлімнің мақсаты «Дәріхана» ақпараттық жүйені құруға арналған шығындарды есептеу болып табылады. Есептеу нәтижесінде қолданбалы бағдарламаның өзіндік құны табылады.

Өзіндік құнды табу үшін ескеру қажет [26]:

а) техникалық құралдар мен программалық қамтаманы толық қалпына келтіруге арналғын амортизациялық аударымдар;

ә) ақпараттық жүйе құрудың еңбек сыйымдылығы;

б) программистке еңбекақы төлеу;

в) техникалық құралдармен жұмсалатын электр энергиясының шығындары;

г) ақпараттық жүйесінің ықтимал (шарттық) бағасын анықтау;

ғ) бірыңғай әлеуметтік салық.

4.2 Ақпараттық жүйес құрудың еңбек сыйымдылығын есептеу

4.1-кесте – Жұмыстарды түрлері мен кезеңдері бойынша бөлу және олардың еңбек сыйымдылығын бағалау

Әзірлеу кезеңі	Жұмыс түрі	Жұмыстың еңбек сыйымдылығы, адам × с.
1	Есептің сипаттамасы	24
2	Алгоритмді әзірлеу	56
3	Алгоритмнің блок-сызбасын әзірлеу	48
4	Жобаның клиенттік бөлігін іске асыру	128
5	Жобаның әкімшіліктік бөлігін әзірлеу	202
6	Бағдарламаны жөндеу және тестілеу	100
7	Құжаттарды, пайдаланушыға нұсқаулықты және түсіндірме жазбаны рәсімдеу	55
Барлығы		613

4.3 Ақпараттық жүйе құруға арналған шығындарды есептеу

Ақпараттық жүйе құруға арналған шығындарды анықтау мыналарды қамтиды [26]:

а) материалдық шығындар;

ә) электр энергиясының шығындары;

б) еңбекке ақы төлеу шығындары;

в) негізгі қорлардың амортизациясы;
г) әлеуметтік салық.

4.3.1 Материалдық шығындарды есептеу

Материалдық шығындарға программалық құралдарды әзірлеу барысында қажетті үстел, орындық, тонер мен басқа да кеңсе тауарларына жұмсалған шығындар есептелген [27]. Олардың атауы, саны және бағасы 4.2-кестеде келтірілген. Бағалары «Abdi» фирмасының прайс парағынан алынған.

4.2-кесте – Материалдық ресурстарға шығындар

Материалдық ресурстың атауы	Бірлік өлшем	Саны	Бірлігі үшін баға, тг	Сомасы, тг
Қағаз	Бума	5	1350	6750
Қалам	Дана	5	65	325
Тонер	Дана	1	4174	4175
Үстел	Дана	1	10000	10000
Орындық	Дана	1	3000	3000
Барлығы				24250

Материалдық ресурстарға шығындардың жалпы сомасы ($Ш_М$) (4.1) формула бойынша анықталады:

$$Ш_м = \sum_{i=1}^{n} Ш_i \times Б_i \quad (4.1)$$

мұндағы, $Ш_i$ - і-ші материалдық ресурстың шығысы, заттай бірліктер;
$Б_i$ - і-ші материалдық ресурстың бірлігінің бағасы, тг;
і - материалдық ресурстың түрі;
n - материалдық ресурстардың саны.

4.3.2 Техникалық құралдармен жұмсалатын электр энергиясының шығындарын есептеу

Егер ақпараттық жүйені құру үшін электр жабдықтары пайдаланылса, онда 4.3-кестеде келтірілген нысан бойынша электр энергиясына кететін шығындарды есептеу қажет.

Электр энергиясына жұмсалатын шығындардың жалпы сомасы ($Ш_Э$) (4.2) формула бойынша есептеледі:

$$Ш_э = \sum_{i=1}^{n} Қ_i \times К_i \times Т_i \times Б \quad (4.2)$$

мұндағы, $Қ_i$ – і-ші электр жабдығының паспорттық қуаты, кВт;

$К_i$ – і-ші электр жабдығының қуатын пайдалану коэффициенті ($К_i$=0.9 қабылданады);

$Т_i$ – і-ші жабдықтың барлық әзірлеу кезеңіндегі жұмыс уақыты;

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

і - электр жабдығының түрі;

n - электр жабдықтарының саны.

Алматы қаласы бойынша заңды тұлғалар үшін 2019 жылы ҚҚС есебімен (ЖШС «АлматыЭнергоСбыт» ресми сайтында ұсынылған деректерге сәйкес) 1 кВт/сағ үшін 17,09 теңгені құрайды.

$$Ш_э = 0{,}9 \times 0{,}9 \times 613 \times 17{,}09 \approx 8485{,}7 \text{ тг}$$
$$Ш_э = 0{,}9 \times 0{,}9 \times 155 \times 17{,}09 \approx 2145{,}65 \text{ тг}$$
$$Ш_э = 0{,}3 \times 0{,}7 \times 613 \times 17{,}09 \approx 2200 \text{ тг}$$

4.3-кесте – Электр энергиясына кететін шығындар

Жабдық-тың атауы	Паспорт-тық қуаты, кВт	Қуатты пайдалану коэффициен-ті	Бағдарлама-лық өнімді әзірлеуге арналған жабдықтың жұмыс уақыты, сағ	Электр қуаты-ның бағасы, тг/кВт· сағ	Сомасы, тг
Ноутбук ASUS FX705DY	0,9	0,9	613	17,09	8485,7
Принтер	0,9	0,9	155	17,09	2145,65
Жарық-тандыру	0,3	0,7	613	17,09	2200
Барлығы					12831,35

4.3.3 Программистке еңбекақы төлеу шығындарын есептеу

Программистің орташа жалақысы 2019 жылы 200 000 теңгені құрайды (Алматы қаласы үшін).

Қызметкердің бір айдағы жұмыс сағаттары (4.3) формула бойынша анықталады:

$$С_а = N_а \times С_{жк} \quad (4.3)$$

мұндағы, $С_а$ - бір айдағы қызметкердің жұмыс сағаты;

$N_а$ - бір айдағы жұмыс күндерінің саны;

$С_{жк}$ - күніне жұмыс сағаттарының саны.

$$С_а = 21 \times 8 = 168 \text{ сағ.}$$

Қызметкердің сағаттық бағасы (4.4) формула бойынша есептеледі:

$$СБ_i = \frac{АЖ_i}{ЖУАҚ_i} \quad (4.4)$$

мұндағы, АЖ$_i$ - і қызметкердің айлық жалақысы, тг;

ЖУАҚі - і қызметкердің жұмыс уақытының айлық қоры, сағ.

Программист:

$$СБ_i = \frac{200000}{168} = 1190{,}48 \text{ тг.}$$

Ақпараттық жүйенің еңбек сыйымдылығын анықтау үшін 4.1-кестедегі деректер пайдаланылады.

Программистің ақпараттық жүйесінің еңбек сыйымдылығы 613 адам×сағ тең.

$$T_2 = 24 + 56 + 48 + 128 + 202 + 55 = 613 \text{ адам} \times \text{сағ.}$$

Еңбекақы төлеуге жұмсалатын шығындардың жалпы сомасы (Ш$_{еа}$) (4.5) формула бойынша анықталады:

$$Ш_{еа} = \sum_{i=1}^{n} СБ_i \times ЕС_i \quad (4.5)$$

мұндағы, СБ$_i$ - і қызметкердің сағаттық ставкасы, тг;

ЕСі - ақпараттық жүйесін құрудың еңбек сыйымдылығы, адам×сағ;

і - қызметкердің санаты;

n - ақпараттық жүйені құрумен айналысатын қызметкерлердің саны.

Программист:

$$Ш_{еа} = 1190{,}48 \times 613 = 729764{,}24$$

4.4-кесте – Еңбекке ақы төлеу шығындары

Біліктілігі	Программалық өнімді әзірлеудің еңбек сыйымдылығы, адам×сағ	Сағаттық баға, тг/сағ	Сомасы, тг
Программист	613	1190,48	729764,24
Барлығы			729764,24

Қосымша еңбекақы:

$$ЕА_{қос} = Ш_{еа} \times 10\% \quad (4.6)$$

$$ЕА_{қос} = 729764{,}24 \times 0{,}1 = 72976{,}42 \text{ тг}$$

Еңбекақы қоры:

$$Қ_{еа} = Ш_{еа} + ЕА_{қос} \quad (4.7)$$

$$Қ_{еа} = 72976{,}42 + 729764{,}24 = 802740{,}66$$

Әлеуметтік салықты есептеу:

$$С_{ә} = (Қ_{еа} - МЗЖ) \times 11\% \quad (4.8)$$

мұндағы, МЗЖ - міндетті зейнетақы жарналары - $Қ_{еа}$-дан 10%.

$$С_{ә} = (802740{,}66 - (802740{,}66 \times 0{,}1)) \times 0{,}11 = 793910{,}51 \text{ тг}$$

4.3.4 Техникалық құралдар мен программалық қамтаманы толық қалпына келтіруге арналғын амортизациялық негізгі қорын есептеу

Осы жобада келесі техникалық құралдар қолданылады:
– ноутбук ASUS FX705DY (90NR0192-M00900);
– принтер лазерный Samsung SL-M2020.

Зерттеулердің соңғы нәтижесі болып табылатын бағдарламалар пакетін құру үшін мынадай программалық қамтама қолданылды:
– Apache server.

Амортизациялық аударымдардың жалпы сомасы (4.9) формула бойынша анықталады:

$$Ш_{амор} = \sum_{i=1}^{n} \frac{К_i \times Н_{Аi} \times Т_{бкi}}{100 \times Т_{тқi}} \quad (4.9)$$

мұндағы, $К_i$ – і-ші негізгі қор құны, тг;

$Н_{Аi}$ – і-ші негізгі қор амортизациясының жылдық нормасы, %;

$Т_{бкі}$ - ақпараттық жүйені құрудың барлық кезеңіндегі і-ші негізгі қордың жұмыс уақыты, сағ.;

$Т_{тқі}$ - і-ші негізгі қордың бір жылдағы жұмыс уақытының тиімді қоры, сағ / жыл;

і - негізгі қор түрі;

n - негізгі қорлар саны.

Негізгі қор амортизациясының жылдық нормасын есептеу:

$$\text{Н}_{\text{А}i} = \frac{100}{T_{\text{ы}i}} \quad (4.10)$$

мұндағы, $\text{Т}_{\text{ы}i}$ – негізгі қорды пайдаланудың ықтимал мерзімі, жыл.

$$\text{Н}_{\text{А}i} = \frac{100}{4} = 25.$$

Қосымша әзірлеу үшін программалық қамтама жұмыс уақытын анықтау үшін 4.1-кестедегі деректер пайдаланылады.

Apache server программалық қамтамасының жұмыс уақыты 430 сағатты құрайды (жобаның клиенттік бөлігін іске асыру, жобаның әкімшілдік бөлігін іске асыру, бағдарламаны жөндеу және тестілеу).

$$\text{Т}_{\text{і}} = 128 + 202 + 100 = 430 \text{ сағ.}$$

Құрал-жабдықтар:

– ноутбук ASUS FX705DY (90NR0192-M00900):

$$\text{Ш}_{\text{амор}} = \frac{399099 \times 25 \times 613}{100 \times 1920} = 31855{,}17 \text{ тг}$$

– принтер лазерный Samsung SL-M2020:

$$\text{Ш}_{\text{амор}} = \frac{44900 \times 25 \times 613}{100 \times 1920} = 3583{,}81 \text{ тг}$$

Программалық қамтама:

$$\text{Ш}_{\text{амор}} = \frac{70555 \times 25 \times 430}{100 \times 1920} = 3950{,}34 \text{ тг}$$

4.5-кестеде көрсетілген техникалық құралдардың бағасы “Белый Ветер” фирмасының прайс парағынан алынған.

4.5-кесте – Негізгі қорлардың амортизациясы

Жабдық-тың атауы және ПҚ	Жабдық-тың және ПҚ-ның құны, тг	Жылдық аморти-зация нормасы, %	Жабдықтың және ПҚ-ның жұмыс уақытының тиімді қоры, сағ/жыл	Ақпараттық жүйені құруға арналған жабдық және ПҚ-ның жұмыс уақыты, сағ	Сомасы, тг
Ноутбук ASUS FX705DY (90NR0192-M00900)	354199	25	1920	613	31855,17
ОС Windows 10	Тегін	-	1920	613	-
Apache server	70555	25	1920	430	3950,34
Принтер лазерный Samsung SL-M2020	44900	25	1920	613	3583,81
Барлығы					39389,32

4.6-кесте – ПҚ әзірлеуге арналған шығындар жоспары

	Шығындар баптары	Сомасы, тг
1	Материалдық шығындар, оның ішінде: - материалдар - электр энергиясы	 24250 12831,35
2	Еңбекақы төлеу шығындары	729764,24
3	Әлеуметтік қажеттіліктерге аударымдар	793910,51
4	Негізгі қорлардың амортизациясы	39389,32
5	Басқа шығындар	10000
Барлығы		1610145,42

4.4 Ақпараттық жүйенің ықтимал (шарттық) бағасын анықтау

Қолданбалы ақпараттық жүйе үшін шарттық баға ($Б_ш$) (4.11) формула бойынша есептеледі:

$$Б_{ш} = Ш_{Бө} \times (1 + \frac{P}{100}) \quad (4.11)$$

мұндағы, $Ш_{бө}$ - ақпараттық жүйені құруға арналған шығындар (4.6-кестеден), тг;

Р - ақпараттық жүйе табыстылығының орташа деңгейі – 25%.

$$Б_{ш} = 1610145{,}42 \times (1 + 0{,}25) = 2012681{,}77 \text{ тг}$$

Бұдан әрі өткізу бағасы қосылған құн салығын (ҚҚС) есепке ала отырып анықталады, ҚҚС бағасы ҚР заңнамалық Салық кодексімен белгіленеді. 2019 жылға ҚҚС бағасы 12% мөлшерінде белгіленген.

ҚҚС есебімен өткізу бағасы (4.12) формула бойынша есептеледі:

$$Ц_{ө}=Б_{ш} + Б_{ш} * ҚҚС \quad (4.12)$$

$$Ц_{ө} = 2012681{,}77 + 2012681{,}77 \times 0{,}12 = 2254203{,}58 \text{ тг.}$$

4.5 «Дәріхана» ақпараттық жүйені құру бойынша экономикалық бөлімге қорытынды

Барлық шығындарды ескере отырып, «Дәріхана» ақпараттық жүйесін іске асыру бағасы 2254203,58 теңгені құрайды. Шығынның негізгі бөлігін еңбекақы төлеу шығындары (32,37%) және әлеуметтік қажеттіліктерге аударымдар (35,22%) құрайды.

Жоғарыда айтылғандарды негізге ала отырып, осы ақпараттық жүйені қолдану ақпаратты жинау процесін толық автоматтандыруға мүмкіндік береді және оны ұсынуды жеңілдетеді.

Сонымен, «Дәріхана» ақпараттық жүйесін құру және енгізу экономикалық жағынан тиімді.

Қорытынды экономикалық көрсеткіштер 4.6-кестеде көрсетілген.

5 Өміртіршілік қауіпсіздігі

5.1 Дәрі-дәрмектерді пайдалану кезіндегі дәріхана қызметкерлерінің еңбек жағдайларын талдау

Бұл жобаның мақсаты дәрі-дәрмек сатумен айналысатын медициналық орталықтар үшін "Дәріхана" ақпараттық жүйесін құру болып табылады. Құрылған жоба арқылы әкімшілік дәрі-дәрмектердің мәліметтер қорымен жұмыс істей алады.

Персонал екі қызметкерден тұрады: бас техникалық маман және дәріханашы.

Қызметкерлерінің жұмысы компьютер, дәрі-дәрмек және зиянды заттармен байланысты болғандықтан, олардың еңбек өнімділігін төмендетеді.

Мұндай факторларға жатқызуға болады:

– жарықтың дұрыс түспеуі;
– микроклиматтың бұзылуы;
– шу.

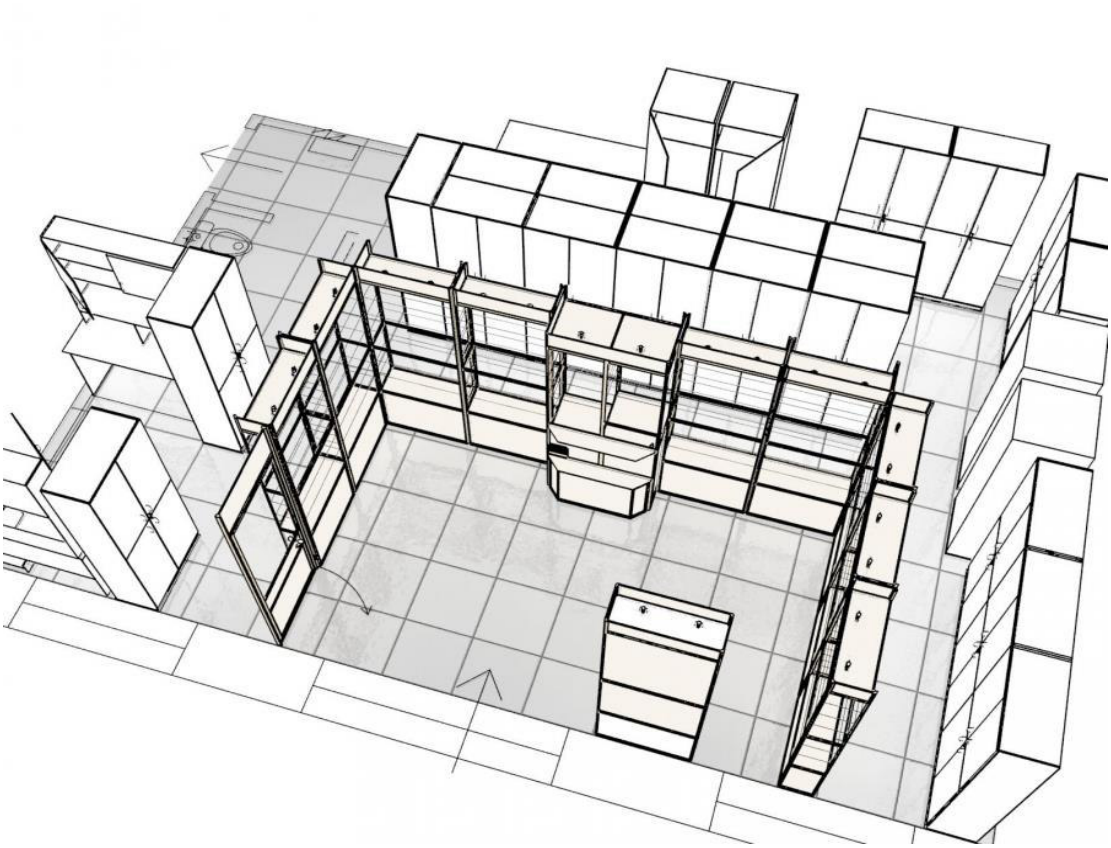
Дәріханаларда стендтердің барлық сөрелерін және оларға орналастырылған дәрілік препараттарды толыққанды қарауға мүмкіндік беретін жарық жеткілікті. Дәріханадағы шу нормаға сәйкес келеді, өйткені оқшаулау принципі қолданылады, атап айтқанда барлық агрегаттар мен құрылғылар шу қорғайтын экрандармен жабдықталған. Дәріхананың өлшемі: ұзындығы (А) = 5 метр, ені (В) = 4 метр, биіктігі (Н) = 2 метр.

Дәрі-дәрмек қоюға арналған дәріхананың жоспары және персоналдың жұмыс орны 5.1-суретте бейнеленген.

5.2. Табиғи желдету есебі

Дәріхананың геометриялық өлшемдері: ұзындығы А=5; ені В=4; биіктігі Н=2. Қабырғалары көгілдір-ақ, төбесі ақ, полимер плиткамен қапталған бетон еден.

Дәріханада батысқа қарайтын екі қабатты шынысы бар екі терезе орналасқан, оның өлшемі 1.5х1 м. Жазда сыртқы ауаның ең жоғары температурасы +30 °С, қыста ең төменгі температура -20 °С. Ауаның салыстырмалы ылғалдылығы φ=50%. Жазда сыртқы ауаның ең жоғарғы температурасы +30 ° С сәйкесінше бөлменің салыстырмалы ылғалдылығы 60%.



5.1-сурет – Дәріхана жоспары

Ғимаратта этанол бөлінген кезде қажетті ауа алмасуды есептеу (5.1) формула бойынша анықталады:

$$L = \frac{G}{K(C_{\text{ПДК}} - C_0)} = \frac{2 \cdot 10^6}{1(1000-0)} = 2000 \text{ м}^3/\text{сағ}, \quad (5.1)$$

мұндағы, L - ғимараттан берілетін немесе шығарылатын ауаның қажетті саны, $м^3$/сағ;

G - ғимараттан зиянды заттың бөліну қарқындылығы (5.1-кесте), мг/сағ;

К - ғимараттағы желдету ауасының біркелкі таралуының өлшемсіз коэффициенті (оны біркелкі бөлу кезінде К = 1 деп қабылданады);

$C_{ПДК}$ - ғимараттың жұмыс аймағындағы зиянды заттардың шекті рұқсат етілген концентрациясы (ШРК), мг/$м^3$ (5.2-кесте);

C_o - бұл заттың ғимаратты желдету үшін түсетін ауадағы концентрациясы, мг/$м^3$; әдетте $C_o = 0$.

5.1-кесте – Зиянды заттардың бөліну қарқындылығы [25]

Заттар	Зиянды заттардың бөліну қарқындылығы, мг/сағ
Күкірт сутегі H_2S	3×10^2

Сұр ангидрид SO_3	1×10^3
Бутанол C_4H_9OH	3×10^4

5.1-кестенің жалғасы

Заттар	Зиянды заттардың бөліну қарқындылығы, мг/сағ
Күкіртті ангидрид SO_2	2×10^4
Күкірт көміртегі CS_2	2×10^3
Метанол CH_3OH	2×10^4
Марганец Mn	8×10^2
Анилин $C_6H_5NH_2$	2×10^2
Этанол C_2H_5OH	2×10^6
Бензол C_6H_6	2×10^4
Ацетон $(CH_3)_2CO$	4×10^5
Металл сынап Hg	1×10^2
Бенз(а)пирен $C_{20}H_{12}$	$1{,}8\times10^5$

5.2-кесте – Жұмыс аймағындағы зиянды заттардың шекті рұқсат етілген концентрациясы [24]

Заттар	ШРК, мг/м^3	Қауіптілік класы	Заттар	ШРК, мг/м^3	Қауіптілік класы
1	2	3	4	5	6
Альдегид изомасляный	5,0	3	Полипропилен нестабилизированный	10,0	3
Альдегид масляный	5,0	3	Поливинилхлорид	6,0	3
Амилацетат	100	4	Пиридин	5,0	2
Аммиак	20,0	4	Пропиламин	200	4
Күкіртті ангидрид	10,0	3	Төмен қысымды полиэтилен	10,0	3
Ангидрид фталевый	1,0	2	Күкірт сутегі	10,0	2
Ацетальдегид	5,0	3	Күкірт көміртегі	1,0	2
Ацетон	200,0	4	Скипидар	300	4
Бензин	300,0	4	Бутил спирті	10,0	3
Бромбензол	3,0	2	Амиль спирті	10,0	3
Бензол	5,0	2	Пропил спирті	10,0	3
Винилацетат	10,0	3	Этил спирті	1000	1
Гексахлорбензол	0,9	2	Стирол	5,0	3
Диметилэтанолам	5,0	3	Толуол	50,0	3

ин					
Диметиламин	1,0	2	Трифторстирол	5,0	3
Дихлорбензол	20,0	4	Трихлорбензол	10,0	3
Дихлорэтан	10,0	2	Трихлорэтилен	10,0	3
Диэтиламин	30,0	4	Триэтиламин	10,0	3
Диэтилбензол	10,0	3	Уайт-спирит	300	4
Диэтилді эфир	300,0	4	Углеводороды пред	300	4
Изопрен	40,0	4	Фенол	0,3	2
Изопропилбензол	50,0	4	Фенопласты	6,0	3
Керосин	300,0	4	Формальдегид	0,5	2

5.2-кестенің жалғасы

1	2	3	4	5	6
Күкірт қышқылы	1,0	2	Фосфорлы сутегі	0,1	1
Тұз қышқылы	5,0	2	Фторлы сутегі	0,5	2
Сірке қышқылы	5,0	3	Фторопласт –4	10,0	3
Ксилол	50,0	3	Хлор	1,0	2
Минералды майлар	5,0	3	Хлорлы сутегі	5,0	2
Метилацетат	100,0	4	Хлорбензол	50,0	3
Бромды метил	1,0	1	Хлорпрен	0,05	1
Хлорлы метил	5,0	2	Циклогексан	80,0	4
Метилэтилкетон	200,0	4	Циклогексиламин	1,0	2
Нафталин	20,0	4	Циклопентадиен	5,0	3
Мұнай	100,0	4	Этилацетат	200	4
Көміртек тотығы	20,0	4	Этиленмеркаптан	1,0	2
Этилен тотығы	1,0	2	Этилтолуол	50,0	4
Пентафторбензол	5,0	3	Бромдық этил	5,0	3
Пентафторфенол	5,0	3	Хлорлы этил	50,0	4

Ауа алмасу L санын алдын ала есептеу бойынша бөлмедегі ауа алмасу еселігін $К_{об}$ (1/сағ) орнатамыз, яғни бір сағат ішінде қанша рет ондағы ауаны таза ауамен ауыстыру қажет, бұл үшін қарастырылып отырған зиянды заттың құрамы ШРК-дан аспауы керек:

$$K_{\text{об}} = \frac{L}{V} = \frac{2000}{40} = 50\ \frac{1}{\text{сағ}}, \qquad (5.2)$$

мұндағы, V - желдетілетін ғимараттың көлемі, $м^3$.

Содан кейін (5.3) формула бойынша желдету арналары қимасының жиынтық ауданын есептейміз:

$$\sum F = \frac{L}{15948*\psi\sqrt{\frac{h(\rho_н-\rho_в)}{\rho_н}}} = \frac{2000}{15948*0{,}5\sqrt{\frac{4(1.165-1.193)}{1.165}}} = 0.809\ м^2, \qquad (5.3)$$

мұндағы, Ψ - каналдардағы ауа қозғалысының кедергісін ескеретін коэффициент (әдетте Ψ =0,5);

h - сору каналдарының биіктігі, м;

$р_н$ - сыртқы ауаның тығыздығы, кг/м^3;

$р_в$ - ішкі ауаның тығыздығы, кг/м^3.

Ауа тығыздығы, кг/м^3:

$$\rho = \frac{353}{273+t}, \qquad (5.4)$$

мұндағы, t - тығыздықты анықтайтын ауа температурасы, °С.

Сыртқы ауаның тығыздығы:

$$\rho_н = \frac{353}{273+t} = \frac{353}{273+30} = 1.165\ кг/м^3.$$

Ішкі ауа тығыздығы:

$$\rho_в = \frac{353}{273+t} = \frac{353}{273+23} = 1.193\ кг/м^3.$$

Бір сору шахтасының қимасының ауданы дефлекторлардың нормаланған қатары ескеріле отырып, конструктивті түрде қабылданады. (5.5) формула бойынша арналар санын есептейміз:

$$\eta_{выт} = \frac{\sum F}{f}, \qquad (5.5)$$

мұндағы, f - шахта түбі қимасының ауданы, м^2.

Алдымен шахтаның түбі қимасының ауданын есептеу керек:

$$f = \frac{L}{3600*\upsilon} = \frac{2000}{3600*2.7} = 0.206\ м^2 \qquad (5.6)$$

$$\eta_{выт} = \frac{0.809}{0.206} = 3.927 \approx 4.$$

Бір дефлектор арқылы шығарылатын ауаның көлемі:

$$L_Д = \frac{L}{\eta_{выт}} = \frac{2000}{4} = 500\ м^3/сағ \qquad (5.7)$$

Дефлектордың түтігінің диаметрі:

$$D_{\text{П}} = 0.0188\sqrt{\frac{L_{\text{Д}}}{K_{\text{эф}}*\upsilon_{\text{е}}}} = 0.0188\sqrt{\frac{500}{0.4*2.7}} = 0.405 \text{ м} \qquad (5.8)$$

мұндағы, $К_{Эф}$ - тиімділік коэффициенті: ЦАГИ дефлекторлары үшін $К_{Эф}$ = 0,4, жұлдыз тәріздес үшін $К_{Эф}$ = 0,42;

$\upsilon_{\text{е}}$ - желдің орташа жылдамдығы, м/с (5.3-кесте).

5.3-кесте – Қазақстанның кейбір қалалары үшін сыртқы ауаның жылдамдығы мен орташа температурасының мәні [24]

Қалалар	$\upsilon_{\text{е}}$, м/с	$t_{суық}$, °С	$t_{жылы}$, °С
Ақтау	3,7	-24	30
Ақтөбе	1,8	-22	29,2
Алматы	2,7	-11	29,7
Астана	2,9	-23	27
Атырау	2,8	-25	32,1
Жамбыл	1,6	-7	31,9
Балқаш	4,0	-20	28,7
Қарағанды	2,7	-21	26,8
Көкшетау	3,9	-25	25,8
Қостанай	2,5	-24	26,7
Қызылорда	2,5	-18	34,1
Павлодар	2,6	-24	27,7
Петропавл	3,6	-24	24,9
Семей	2,6	-23	28,9
Талдықорган	1,9	-11	30,8
Орал	2,4	-19	29,5
Өскемен	2,4	-26	28,2
Шымкент	2,6	-6	33

Қорытынды. Нормаланған мәнді қамтамасыз ету үшін мойын диаметрі 1000 мм астам дефлекторларды жасау қажет және жалпы алмасатын сору-сыртқа тарату желдеткішін қою қажет.

5.3 Аэрацияны есептеу

Аэрация есебі жазғы уақыт үшін жел қысымын ескермей, осы процесті жүзеге асыру үшін неғұрлым қолайсыз жүргізіледі. Есептеу мәні сору және сыртқа тарату ойықтарының ауданын анықтаудан тұрады [24].

Ең алдымен қажетті ауа алмасуын анықтаймыз, $м^3$/сағ. Ғимараттағы ауаға зиянды заттар бөлінетін заттар (5.1) формуласы бойынша анықталады.

Жылудың артықтығы бойынша есептеу кезінде қабырғалардағы ойықтар арқылы түсетін және аэрациялық фонарлар арқылы шығарылатын ауаның мөлшері L, $м^3$/сағ, (5.9) формула бойынша есептеледі:

$$L = \frac{\chi \cdot k_a \cdot Q}{6.28 \cdot (t_в - t_н)} = \frac{1.04 \cdot (-2.33) \cdot 174}{6.28 \cdot (23-30)} = 9.591 \text{ м}^3/\text{сағ}, \quad (5.9)$$

мұндағы, χ - еденнен ағатын ойықтар орталықтарының орналасу биіктігін ескеретін коэффициенті (5.4-кесте);

Q - ғимараттан бөлінетін жылу қуаты, Вт;

$t_в$ - жұмыс аймағындағы ауа температурасы, °С;

$t_н$ - жылдың ең ыстық айының 13 сағаттағы орташа температураға тең қабылданатын сыртқы ауаның есептік температурасы, °С (5.3-кесте);

k_a - бөлмедегі температуралық режимді ескеретін коэффициент, (5.10) формула бойынша анықталады:

$$k_a = \frac{t_в - t_н}{t_у - t_н} = \frac{23-30}{33-30} = -2.33 \quad (5.10)$$

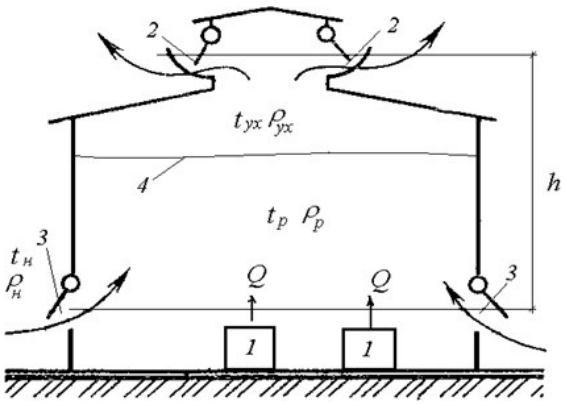
мұндағы, $t_у$ - өрнектен анықталатын жұмыс аймағындағы шығарылатын ауаның температурасы. Ол (5.11) формула арқылы анықталады:

$$t_у = t_в + a(h_в - 2) = 23 + 5(4-2) = 33 \text{ °C} \quad (5.11)$$

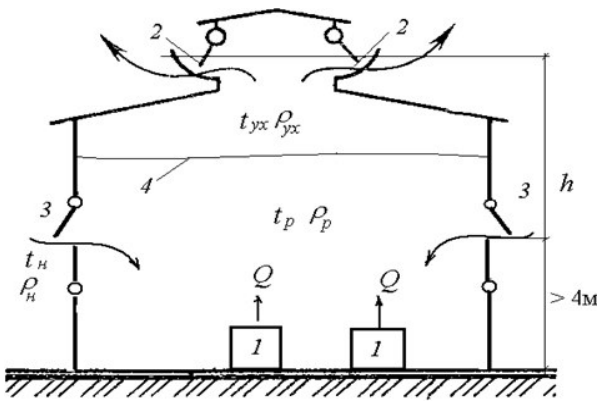
5.4-кесте – Х коэффициентінің қабылдайтын мәндері [24]

Еденнен ойықтың осіне дейінгі қашықтық, м	2	3	4	5
χ	1,04	1,1	1,2	1,35

А)



Б)



5.2-сурет – Дәріхана ғимаратын аэрациялау схемасы

А, Б – жылдың жылы және суық кезеңінде;

1 – жылу бөлу көздері; 2 – сору ойықтары; 3 – сыртқа тарату ойықтары; 4 – температуралық жабу.

Ауа алмасудың алынған мәні бойынша сору және сыртқа тарату ойықтарының ауданын (5.12) формула бойынша есептейміз:

$$\sum F = \frac{L}{15948 \cdot \psi_p \sqrt{\frac{h_a \cdot (\rho_н - \rho_в)}{\rho_н}}} = \frac{9.591}{15948 \cdot 0.5 \sqrt{\frac{4 \cdot (1.165 - 1.193)}{1.164}}} = 0.004 \text{ м}^2 \qquad (5.12)$$

мұндағы, $\Psi_р$ - жармалар мен олардың ашылу бұрышындағы сору және сыртқа тарату ойықтарында орнатылатын конструкцияларға байланысты шығын коэффициенті: $\Psi_р$ = 0,15... 0,67;

$h_{выт}$ - биіктік бойынша сору және сыртқа тарату ойықтарының орталықтары арасындағы қашықтық, м.

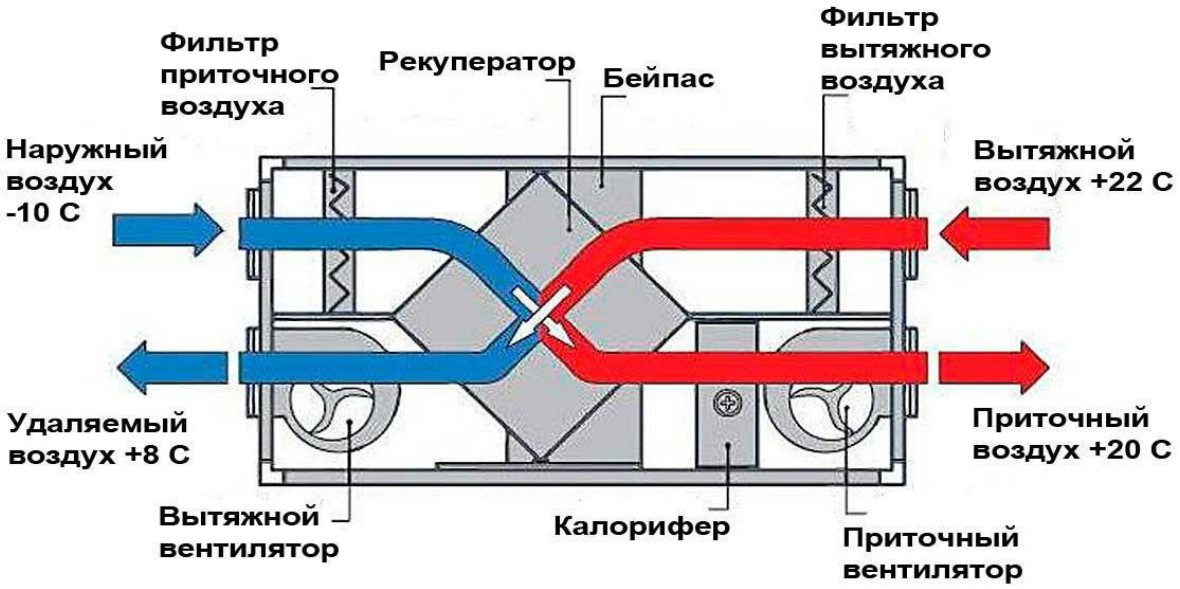
Әр түрлі конструкциялардың жармалары үшін $\Psi_р$ шығын коэффициентінің мәні және олардың ашылу бұрыштарында а = 15...90° 5.5-кестеде келтірілген.

5.5-кесте – $\Psi_р$ шығыс коэффициентінің мәні [24]

Жарманың түрі	Орташа салмақты сыртқа тарату	Жоғарғы салмақты сыртқа тарату	Жоғарғы салмақты сору
$\Psi_р$	0,15-0,64	0,25-0,62	0,3-0,67

5.4 Механикалық вентиляцияны есептеу

Ғимараттың табиғи желдеткіші және механикалық желдеткіші бар, ол ғимаратқа таза ауаны беруді және ластанған ауаны тез жоюды қамтамасыз етеді. Ол желдету нормативтерге сәйкес келмейді.



5.3-сурет – Рекуперациясы бар желдету схемасы [24]

$L_{СГ}$ мәнін есептеу анық немесе толық жылулықтың артығымен, бөлінетін зиянды заттардың массасымен, ылғалдың (су буының) артығымен, ауа алмасудың нормаланатын еселігімен және сыртқа тарататын ауаның нормаланатын үлестік шығынымен жүргізіледі. Бұл ретте $L_{СГ}$ мәндері жылы және суық жыл кезеңдері үшін сыртқа тарататын және шығарылатын ауаның тығыздығы кезінде жеке анықталады ρ = 1,2 кг/м^3 (20 °С температурада).

Ғимараттағы айқын $Q^{я}_{изб}$ жылу болған кезде қажетті шығынды (5.13) формула бойынша анықтаймыз:

$$L_{я} = \frac{3.6 \cdot Q^{я}_{изб}}{1.2 \cdot (t_{у} - t_{п})} = \frac{3.6 \cdot 20000}{1.2 \cdot (30-23)} = 8571 \text{ м}^3/\text{сағ}, \quad (5.13)$$

мұндағы, $t_у$ және $t_п$ - ғимараттан шығарылатын және келіп түсетін ауаның температуралары.

Бөлмедегі бөлінетін зиянды заттар (бу, газ, шаң, мг/сағ) болған кезде қажетті шығынды (5.14) формула бойынша анықтаймыз:

$$L_{вр} = \frac{G_{вр}}{C_{ПДК} - C_{П} \cdot C_{Д}} = \frac{2 \cdot 10^6}{1000 - 0.3 \cdot 1000} = 2857 \text{ м}^3/\text{сағ}, \quad (5.14)$$

мұндағы, $С_д$ - ғимараттан шығарылатын нақты зиянды заттың шоғырлануы, мг/м^3;

$С_{ПДК}$ - ғимараттың жұмыс аймағындағы зиянды заттың шекті рұқсат етілген концентрациясы, мг/м^3. Жұмыс аймағындағы $С_п \leq 0,3$ ШРК.

Өрт қауіпсіздігі нормаларын қамтамасыз ету үшін ауа шығынын жарылысқа қабілетті осы ғимаратта бөлінетін зиянды заттардың массасы (5.15) формула бойынша жүргіземіз:

$$L_\sigma = \frac{G_{\text{вр}}}{0.1 \cdot C_{\text{нк}} - C_{\text{п}}} = \frac{2 \cdot 10^6}{0.1 \cdot 60 - 0.3 \cdot 1000} = -6803 \text{ м}^3/\text{сағ}, \quad (5.15)$$

мұндағы, $C_{\text{нк}} = 60$ г/м^3 - шаң-ауа қоспалары бойынша жалын таралуының төменгі концентрациялық шегі.

Сыртқы ауаның ең аз шығыны бойынша (5.16) формула бойынша анықтаймыз:

$$L_{min} = n \cdot m \cdot z = 25 \cdot 1.3 \cdot 2 = 65 \text{ м}^3/\text{сағ}, \quad (5.16)$$

мұндағы, m = 25 м^3/сағ - бір қызметкерге арналған ауа нормасы;

z =1,3 - қор коэффициенті;

n = 2 - қызметкерлер саны.

Соңғы: $L_M = 8571$ м^3/сағ.

Желдету желісінің әрбір учаскесі үшін мәндердің ұзындығы L, м, және ауа шығыны L, м^3/сағ, берілген мәндер кезінде аэродинамикалық есептеулер жүргіземіз. Бұл үшін анықтаймыз:

– магистральдық және басқа да ауа өткізгіштері бойынша сору ауасының саны;

– учаскелер бойынша жергілікті кедергілер коэффициенттерінің жиынтық мәнін (5.17) формула бойынша анықтаймыз:

$$\sum_{i=1}^{n} \xi_i = \xi_{\text{пов}} + \xi_{\text{вт}} = \xi_{\text{сп}}, \quad (5.17)$$

мұндағы, $\xi_{\text{пов}}$ - бұрылыстың жергілікті кедергісінің коэффициенті;

$\sum \xi_{\text{вт}} = \xi_{\text{вт}} \cdot n$ - тартпалы үштіктердің жергілікті кедергісінің жиынтық коэффициенті;

$\xi_{\text{сп}}$ - ағынының астында өткір бұрышпен жергілікті кедергісі жанасқан коэффициент, $\xi_{\text{сп}} = 0{,}4$.

Ауа өткізгіштердің салынған схемасына сәйкес жергілікті кедергінің коэффициентін анықтаймыз. Ауа тартқыштың сорғыш бөлігі төрт сортты біріктіреді және желдеткіштен кейін ауа екі бағыт бойынша айдалады.

Кірісіндегі жергілікті кедергі коэффициенті конустық коллектордың таңдалған конструкциясына байланысты болады. Соңғы анықтама мәліметтері бойынша α = 30 °С бұрышында коэффициент 0,8 тең. Екі бірдей дөңгелек бұрылыс α = 90°С бұрышымен жобаланған коэффициент 2 тең. Олар үшін жергілікті кедергі коэффициенті $\xi_0 = 0{,}15$.

Осылайша, жергілікті кедергілердің жиынтық коэффициенті:

$$\sum \xi = 0.8 + 2 \cdot 0.15 = 1.1.$$

Ауа шығынының теңдеуінен ауа өткізгіштердің диаметрін анықтаймыз:

$$d = 1.13 \cdot \sqrt{L/V} = 1.13 \cdot \sqrt{8571/40} = 16.541 \text{ м} \qquad (5.18)$$

Есептелген диаметрлер ең жақын стандартты диаметрлерге дейін дөңгелектенеді. Диаметрлердің алынған мәндері бойынша жылдамдық қайта есептеледі.

Ауа өткізгіш учаскелеріндегі ауа қозғалысының динамикалық қысымы мен келтірілген кедергі коэффициентін анықтаймыз. Қысым шығыны:

$$H_{\text{уч}} = \frac{\rho V^2}{2} \cdot \left(\frac{\lambda l}{d} + \sum \xi\right) = \frac{1.193 \cdot (14)^2}{2} \cdot \left(\frac{0.02 \cdot 1}{16.541} + 1.1\right) \approx 130 \text{ Па} \qquad (5.19)$$

мұндағы, ρ - ауа тығыздығы, кг/м^3 (5.4-формула);

v - әртүрлі шаңды тасымалдауға қажетті құбырдағы ауа қозғалысының жылдамдығы, м/с (5.6-кесте);

λ - ауа тартқыш учаскесіндегі ауа қозғалысының кедергі коэффициенті: металл құбырлар үшін λ = 0,02, полиэтилендік құбырлар үшін λ = 0,01;

l - учаскенің ұзындығы, м;

d - ауа өткізгіштің диаметрі, м;

$\sum \xi$ - жергілікті кедергілердің жиынтық коэффициенті.

5.6-кесте – Ауа қозғалысы жылдамдығының ұсынылатын мәндері

Ауамен тасымалданатын шаң түрі	Ауа қозғалысының жылдамдығы, м/с
Жеңіл құрғақ (ағаш, темекі, ұн және т. б.)	8…10
Тоқыма, астық, бояулардың шаңы	10…12
Минералды	12…14
Ауыр минералды	14…16

Қорытынды. Алынған есептеулер негізінде мынадай қорытынды шығаруға болады: осы ғимаратта берілген шарт жағдайында желдеткіш Ц4-70 №3 желдеткішін ($Q_в$ – 8400 м3/сағ., Мв – 1400 Па, $\eta_в = 0.84, \eta_П = 1$) орнату қажет. Осыдан электрқозғалтқыштың белгіленген қуатын есептейміз:

$$N_у = \frac{1.1 \cdot Q_в \cdot N_в}{\eta_в \cdot \eta_П \cdot 3600 \cdot 102} = \frac{1.1 \cdot 8400 \cdot 1460}{0.84 \cdot 1 \cdot 3600 \cdot 102} = 44 \text{ кВт}, \qquad (5.20)$$

мұндағы, $Q_в$ - қабылданған желдеткіш өнімділігі,

$N_в$ - қабылданған желдеткіш арыны,

$\eta_в$ = η - желдеткіштің ПӘК-і,

$\eta_П$ - беру ПӘК-і.

Есептеулер бойынша N = 44 кВт және ω = 1000 айн/мин болатын МА-35 электроқозғалтқышы таңдалды [25].

Қорытынды

Қазіргі заманда коммуникацялық ресурстардың қарқынды түрде дамуы ақылды, ойлы адамзат баласының алдында автоматтандырылған программалық кешендердің жаңа өрісін ашты, сонымен қатар жаңа шешілуге тиісті мәселелер де ала келді. Соның бірі осы дипломдық жобада қолға алынған мәселе – медициналық орталықтарға арналған “Дәріхана” ақпараттық жүйесінің программалық кешенін құру.

Ақпараттық технологиялардың өркендеп дамуы, компьютердің қоғамда кеңінен қолдану мүмкіндігі, елімізде Internet-тің арзандауы және т.б. – осының барлығы әр түрлі алаларда қолданылатын жүйелердің автоматтандырылуына әсерін тигізді. Жобада жоспарланған жұмыстың соңғы нәтижесі – ақпараттық жүйенің программалық кешені құрылуы қажет. Бұл кешен барлығының қатынау мүмкіндігі бар, яғни ғаламдық компьютерлік Internet торабына қосылған, дәрі дәрмек туралы толық ақпаратпен қамтылған және де дәріхана қызметкерлерінің жұмысын жеңілдетуге, көлемі үлкен ақпараттарды сақтауға, керек мәліметті жылдам табуға көмектесетін болады.

Медициналық орталықтарға арналған «Дәріхана» ақпараттық жүйесі программалық кешеннің көптеген артықшылықтары бар:

– ақпараттық бөлімде жинақталатын мәліметтер жүйелі түрде сақталады;

– дәріхана қызметкерлерінің кешенге енгізген барлық мәліметтері тікелей дерекқорға тіркеледі;

– жүйенің онлайн режимде қолданылуы, яғни қызметкер кез келген интернет желісі қосылған жерде өзіне керекті мәліметтерді ала алады, өзгерте алады немесе қажет емес дәрі дәрмектерді өшіре алады;

– программалық кешенге қатынауға рұқсаты бар кез келген қызметкер өзінің жасаған жұмыстары туралы ақпараттарды орталыққа хабарлама арқылы жібере алады. Яғни, бұл дипломдық жобада ұсынылған нәтижелер дәріхананың ақпараттық жүйесін басқару саласы бойынша телекоммуникациялық технологиялар көмегімен жасалған қазіргі заман ағымына сай программалық кешен. Көптеген дәріханалардың жұмыс сапасын жақсартуға, өнімділігін арттыруға зор ықпалын тигізеді.

Мен дипломдық жобада көптеген технологияларды қолданып, сол технологиялар туралы көп мағлұматтар алдым. Осы дипломдық жобаны жасай отырып, мен MySQL мәліметтер қоры, PHP скрипт, объектілік бағытталған программалау тілі JavaScript және CSS каскадты стильді кестелер тілдерінің көптеген мүмкіндіктерімен танысып, олармен жұмыс жасадым.

Әдебиеттер тізімі

1 Гаффин А. Путеводитель по глобальной компьютерной сети Internet. – М.: Артос, 2006. – 274 с.

2 Гилстер П. Навигатор INTERNET: Путеводитель для человека с компьютером и модемом. – М.: Джон Уайли энд Санз, 2005. – 735 с.

3 Соломенчук В. Интернет: краткий курс: Пособие для ускоренного обучения. – СПб.: Питер, 2000. – 288 с.

4 Кузнецов М., Симдянов И. Объектно-ориентированное программирование на PHP. – СПб.: БХВ-Петербург, 2007. – 267 с.

5 Леки-Томпсон Э., Коув А., Новицки С., Айде-Гудман Х. PHP 5 для профессионалов. – М.: Диалектика, 2006. – 618 с.

6 Кузнецов М., Симдянов И. Самоучитель PHP 5. – 2-е изд., перераб. и доп.. – СПб.: БХВ-Петербург, 2006. – 715 с.

7 Котеров Д., Костарев А. PHP. В подлиннике. – СПб.: БХВ-Петербург, 2005. – 294 с.

8 Электронды нұсқасы http://www.php.ru/ сайтында

9 Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. - М.: Триумф, 2014**.** - 464 с.

10 Дебольт HTML и CSS. Совместное использование / Дебольт, Вирджиния. - М.: НТ Пресс, 2013**. -** 512 с.

11 Лазаро, Исси Коэн Полный справочник по HTML, CSS и JavaScript / Лазаро Исси Коэн, Джозеф Исси Коэн. - М.: ЭКОМ Паблишерз, 2014**. -** 938 с.

12 Прохоренок, Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А. Прохоренок, В.А. Дронов. – Москва: СПб. [и др.]: Питер, 2015. – 768 с.

13 Кузнецов М., Симдянов И., Голышев С. PHP 5. Практика создания Web-сайтов. – СПб.: БХВ-Петербург, 2005. – 350 с.

14 Томсон Л., Веллинг Л. Разработка Web-приложений на PHP и MySQL. – М.: DiaSoft, 2003. – 655 с.

15 Кузнецов М., Симдянов И., Голышев С. PHP 5 на примерах. – СПб.: БХВ-Петербург, 2005. – 372 с.

16 Шелдон Р., Мойе Д. MySQL: базовый курс. – М.: Диалектика, 2007. – 518 с.

17 Кузнецов М., Симдянов И. MySQL на примерах. – СПб.: БХВ-Петербург, 2007. – 484 с.

18 Дюбуа П. MySQL. – 3-е изд. – М.: Вильямс, 2006. – 287 с.

19 Кузнецов М., Симдянов И. MySQL 5. В подлиннике. – СПб.: БХВ-Петербург, 2005. – 560 с.

20 Кузнецов М., Симдянов И. Самоучитель MySQL 5. – СПб.: БХВ-Петербург, 2006. – 472 с.

21 Мацяшек, Лешек, А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML: Перс англ. – М.:

Издательский дом «Вильямс», 2002.

22 Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов: Пер. с англ. – М.: ДМК Пресс, 2002.

23 Буч Г. Рамбо Д. Джекобсон А. Язык UML. Руководство пользователя. – М.: ДМК, 2000.

24 Мананбаева С.Е., Санатова Т.С., Бегимбетова А.С., Бекмуратова Н.С. Безопасность жизнедеятельности. Часть I «Производственная санитария». – Алматы: АУЭС, 2016.- 95с.

25 Борщенская Т.И., Бацукова Н. Л., Павлов А. В. Гигиеническая оценка вентиляции. – Минск: БГМУ, 2018. – 28 с.

26 Еркешева З.Д, Боканова Г.Ш. Методические указания по выполнению расчетно-графических работ для студентов специальности 5В070400 – Вычислительная техника и программное обеспечение. – Алматы: АУЭС, 2017 – 36 с.

27 Бекишева А.И. Методические указания к выполнению экономической части дипломной работыбакалавров специальности 5В070300 – Информационные ситемы. – Алматы: АУЭС, 2013. – 24с.

А Қосымшасы
(міндетті)

Техникалық тапсырма

А.1 Кіріспе

XXI ғасыр интернет технологиялардың даму кезеңі. Интернет технологиялар жылдан жылға қарқынды даму үстінде. Бұрын біз жеке пайдаланушыға арналған тек бір компьютерде ғана орындалатын локальді-қосымшаларды құрумен ғана шектелсек, енді уақыт талабы бойынша бірнеше ондаған, мыңдаған, тіпті миллиондаған пайдаланушыларға арналған клиент-серверлі немесе Web-қосымшалар құруға көшуімізге тура келеді, яғни бұрынғыдай локальді-қосымшаларды құрғаннан Web-қосымшаларды құру анағұрлым қызықтырақ.

Қазіргі кезде ақпараттық жүйе құруға қолданылатын бірнеше технологиялар бар: HTML, PHP, PERL, ASP.NET, Silverlight және т.б. Бұл технологиялардың әрқайсысы белгілі бір мақсатқа байланысты құрылған.

Интернеттегі ақпарат веб-қосымшалар түрінде ұсынылады. Веб-қосымша (қосымша, интернет қор көзі, портал) – ортақ тақырыппен, навигациямен, ортақ URL-мекенжайымен біріктірілген, өзара еренсілтемелер көмегімен байланысып, бір серверде орналасқан веб-беттері жиынтығы. PHP – веб-беттерді құру және дерекқормен жұмыс істеуге арналған скрипті программалау тілі.

Менің жұмысым PHP, MySQL технологиялары көмегімен медициналық орталықтарға арналған «Дәріхана» ақпараттық жүйесінің құру болып табылады.

А.1.1 Өңдеудің мақсаты мен қызметі

Дипломдық жобаның мақсаты – «Дәріхана» дәрі-дәрмек сатумен айналысатын медициналық орталықтың ақпараттық жүйесін құру болып табылады. Қазіргі кезде жұмыс орнын автоматтандыру интернет желісінде ең тез дамып келе жатқан қызмет түрінің бірі. Құрылған жоба арқылы администратор дәрі дәрмек деректер қорымен жұмыс жасай алады.

Бұл жобаның негізгі мақсаты дәрі дәрмек сатумен айналысатын компаниялардың жұмысына барынша ыңғайлылық тудыру.

Дипломдық жобаның нәтижесінде келесі негізгі тапсырмалар орындалуы қажет:

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

А қосымшасының жалғасы

АЖО құру барысында PHP скрипт тілі, HTML гипермәтіндік тілі, CSS каскадты стильді кестелер тілі, MySQL деректер қоры және клиенттің жағында да, сервердің жағында да жұмыс жасайтын сценарийлерді жазуға арналған ОБП (объектілік бағытталған программалау) тілі – JavaScript қолданылады.

А.1.2 Қолдану саласы

Қолдану саласы – дәрі дәрмек сатумен айналысатын Дәріхана саласы.

А.1.3 Анықтамалар, терминдер және қысқартулар

Анықтамалар, терминдер және қысқартулар А.1-кестеде көрсетілген.

А.1-кесте – Анықтамалар, терминдер және қысқартулар

Терминдер немесе қысқартулар	Анықтамалар
Интернет	Интернет сөзі ағылшынша Interconnected Networks (байланысқан желілер) екі сөздің туындысы. Интернет ұғымы ақпараттық және есептеу техникалары ресурстарының әлемдік телекоммуникациялық желісі дегенді білдіреді.
ДҚ	Деректер қоры
ДҚБЖ	Деректер қорының басқару жүйесі
MySQL	MySQL – тығыз көп тасқынды деректер базасының сервері. MySQL қолдануда үлкен жылдамдықпен, орнықтылықпен және жеңілдікпен бейнеленеді.
PHP	PHP (ағылшынша: PHP: Hypertext Preprocessor – «PHP: еренмәтінді алдын-ала үдірістегіш») – Веб-серверде HTML беттерін құру және дерекқормен жұмыс істеуге арналған скрипті программалау тілі.
HTML	HTML сөзі – ағылшын тіліндегі “HyperText Markup Language” деген тіркестің қысқартылған нұсқасы. html тілі – ғаламторда документтерді (файл, Интернет парақшасы…) белгілеудің стандартты тілі.

А.2 Жалпы сипаттамасы

Дерекқор жұмысы клиент-сервер технологиясына негізделген. Администратор бөлімінде – пайдаланушыдан түскен ақпараттар іріктеледі, егерде пайдаланушы тарапынан қойылып отырған мәселе немесе ұсыныстар

дәріхана саласынан тыс немесе басқа мақсаттарда болған жағдайда алынып тасталынады. Пайдаланушылар тарапынан түскен сұрақтарды администратор мазмұнына қарай топтайды.

А.2.1 Пайдаланушылық интерфейстер

Деректер қоры және АЖО әрлендірілуі орындаушымен толық игеріледі.

Бұл жобада үш пайдаланушылық интерфейс қарастырылған: тіркелмеген пайдаланушыларға арналған интерфейс; тіркелген пайдаланушыларға арналған интерфейс; администраторға арналған интерфейс. Тіркелмеген пайдаланушыларға арналған интерфейсте тек іздеген сұрақтың басқа қолданушылар немесе администратор жазған жауаптарын ғана қарауға, сайт мазмұны бойынша қарапайым іздеу мүмкіндіктері қарастырылған. Тіркелген пайдаланушыға арналған интерфейсте тіркелмеген пайдаланушылар интерфейсінің мүмкіндіктеріне қосымша басқа қолданушылар қойған сұрақтарға өзінің пароль және логині арқылы кіріп, жауап беру қарастырылған.

Администраторға арналған интерфейсте дерекқорда бар мүмкіндіктердің барлығы рұқсат етілген, администратор мүмкіндіктері жоба аумағында шексіз.

А.2.2 Аппаратттық интерфейстер

Компьютерлерге қойылатын жалпы талаптар:

– процессор – жоғары;
– оперативті жады – 256 МВ жоғары;
– диск тұлғалы кеңістік – ~ 52 МВ + қолданушылар файлдарын сақтауға қажет орын;
– Internet желісіне қатынау;
– жұмысты қолдайтын видеокарта;
– пернетақта, манипулятор тышқан.

А.2.3 Программалық интерфейстер

Серверлік программалы компоненттер:

– Windows 2000 Server операциялық жүйесі;
– ДҚБЖ MySQL Server 5.0;
– PHP соңғы нұсқасы сайтты басқару жүйесі;

Клиенттік программалы компоненттер:

– Windows 2000/XP/Vista/Seven операциялық жүйесі;
– Internet Explorer, Mozilla Firefox, Opera.

А.2.4 Коммуникацилық интерфейстер

А қосымшасының жалғасы

Пайдаланушылардың компьютерлерінде интернетке қатынауға бар модем құралдары немесе жылдамдығы 100 Мб/с кем емес желілі карта болуы тиіс. Сервер мен клиенттерді байланыстыратын TCP/IP протоколы қолданылады.

А.2.5 Жады бойынша шектеулер

Серверлік және клиенттік машиналар үшін жалпы жады бойынша қойылатын талаптар, оларда орналасқан жұмысқа қажетті программалық компоненттердің қойылатын талаптармен анықталады.

А.2.6 Адаптация бойынша талаптар

Программалық кешен Windows 2000/2003 Server/7/Vista/XP платформаларында жұмыс істеуі тиіс, бұл немесе басқа платформаларды қолдануда қажетті программалық компоненттер ыңғайлы орнатылуы керек.

А.3 Өңдеуге талаптар

А.3.1 Функциональды талаптар

Медициналық орталықтарға арналған «Дәріхана» ақпараттық жүйесіне әрбір қолданушысы өзінің логині және паролмен қатынас құру керек. Сол кезде қолданушы өзіне қажетті сұрақтарға жауап алады және өзге де сұрақтарға жауап бере алады.

А.3.1.1 Шығыс деректері

Серверге кіретін мәліметтер ретінде, қолданушылардың сұранысы, объектінің түрі және қосымша мәліметтерді жіберу.

Сонымен қатар, қолданушылармен байланыс орнатын, контакты ақпараттар, яғни қолданушының аты-жөні, телефон номері, мекен-жайы сияқты толық мәліметтер болуы қажет.

Администратор жағынан мынандай мәліметтер шығуы қажет: пайдаланушыдан түскен ақпараттар іріктеледі, егерде пайдаланушы тарапынан қойылып отырған мәселе немесе ұсыныстар информатика ғылымы саласынан тыс немесе басқа мақсаттарда болған жағдайда алынып тасталынады.

А.3.1.2 Нәтижелер

MySQL жүйесі жойылған компьютерлерді қолданылушылар қосылуға мүмкіндігі бар өзінің серверін ұсынады. ДҚБЖ MySQL – PHP-де қолданылатын

көптеген дерекқорлардың арасындағы бірі болып табылады.

SQL(Structured Query Language) – құрылымдық сұраныс тілі. Бұл тіл – әр түрлі деректер қорына қатынау үшін арналған стандартты құрал ретінде болып келеді.

А.3.1.3 Функциональды талаптарды ұйымдастыру

Жүйенің жұмысын, техникалық тапсырманың талаптарымен салыстыру. Барлық сынау жұмыстары, анықталған құрал – жабдықтарда өткізілуі қажет. Ол келесідегідей құралдармен, яғни жергілікті желі арқылы байланысқан екі компьютер арқылы жүзеге асады. Оның біріншісі сервер болады да, оған Windows (32) операциялық жүйесі, Apache сервері және PHP 5, MySQL Server 5.0 программалық қамтамалары орнатылады. Ал клиенттік машинаға Windows (32) операциялық жүйесі, интернет қосылу мүмкіндігін беретін бағдарламалар орнатылады.

Класс бойыншы талаптарды ұйымдастыру – бұл технология алдымен жіктелуін анықтау керек болатын талаптар ұйымының обьектті-бағдарланған стилін қолданады, яғни класстарды таңдау эквиваленті.

Заттық облысының негізгі (базалық) түсініктерін жіктегеннен кейін, қалған талаптарды алынған категориялар немесе класстар бойынша

орналастырады. Ол үшін істің екі ыңғайы бар:

– класстарды талаптарды ұйымдастыру тәсілдері сияқты қарастыруға болады, бірақ жобалауда оларды міндетті қолданушы деп есептемеу;

– обьектілі-бағдарланған жобалауда және іске асыру талаптары үшін өңделген класстарды қолдануға болады.

Екінші пункті бағдарламалау тілінің функциясымен және әрбір функционалды толық талаптар арасында сәйкестік құрып, әдістермен және толық талаптар арасындағы біртекті жалғасуды қолдайды.

Іске асыру және жобаның, талаптар арасындағы қатал талаптарды қолдау, келешекте жобалауда қолданылатын класс бойынша талаптарды ұйымдастыру үлкен артықшылығы болып табылады. Бұдан басқа қазіргі әлем түсініктеріне сәйкес келетін кластар үшін, оларды қайта қолдану үлкен ықтималдылығы бар.

А.3.1.4 Функциональды диаграммалар

Rational Rose өнімдерінің сериясы құрастырушыны нақты уақыт жүйелерінде және «клиент/сервер» орталарында қолдануға жарайтын және

қазіргі кездегі бизнес талаптарын қанағаттандыратын тиімді де сенімді шешімдерді қабылдауға көмектесетін визуалдық модельдеудің толық құралдар жиынымен қамтамасыз етеді. Rational Rose құралдары біркелкі стандарттарға негізделген және модельдеуді, оларға жақын сфералардағы бизнес-процестерді оптимизациялауға талаптанатын, компьютерлік ғылымдармен онша таныс емес

тұлғалармен қатар, программалық қолданбалардың логикасын модельдеу құралдарын қажет ететін мамандар үшін оңай етеді.

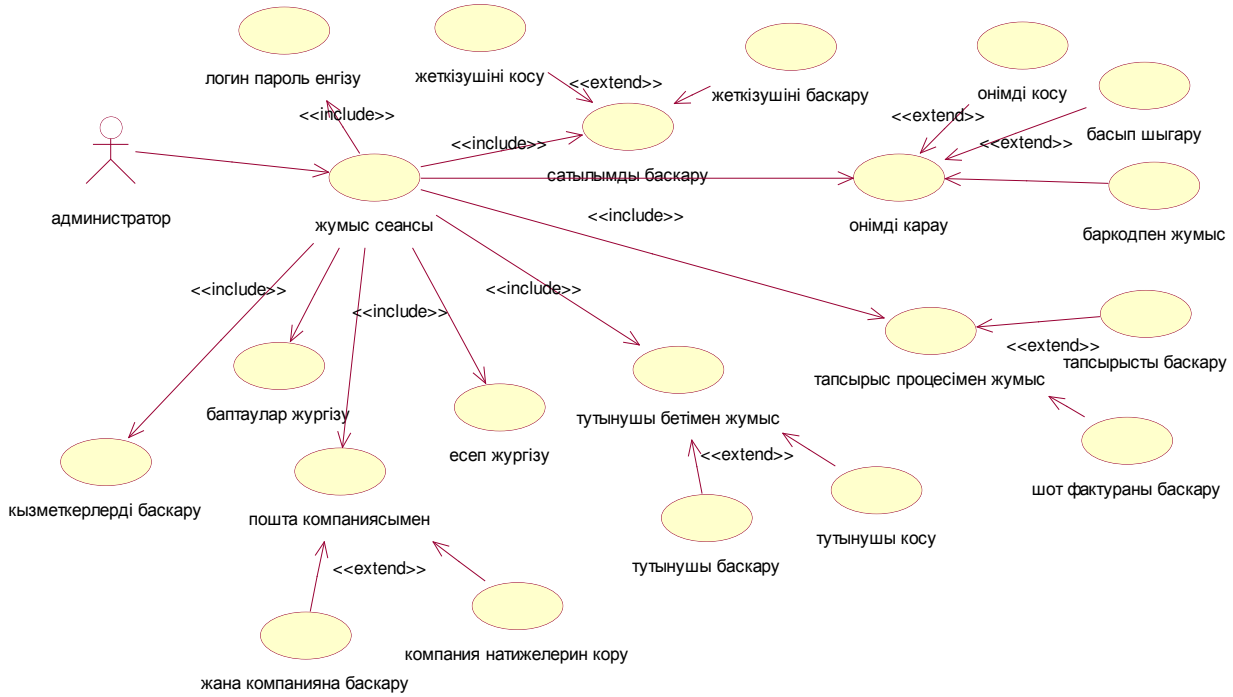
Rational Rose – UML әдістемесін тарату аспабы. Ұқсас есептерді шешу үшін қолданылатын басқа да визуальды аспаптар.

IBM Rational Rose 2003-ші бағдарламаның жұмыс интерфейсі негізгісі болып табылған әр түрлі элементтерден тұрады:

– бас мәзір;
– стандартты аспаптар құралы;
– арнайы аспаптар құралдары;
– жобаның браузерінің терезесі;
– диаграмманың суретінің жұмыс облысы немесе диаграмманың терезесі;
– құжаттаманың терезесі;
– журналдың терезесі.

А.3.1.5 Қолдану вариантты бойынша

Диаграммалардың бұл түрі қай операциялардың тізімі қалай жасауға мүмкіндік беретін жүйені орындайды. Прецендентер диаграммалардың жиынның негізінде жүйеге талаптарының тізімі жасалады. Әрбір мұндай диаграммады Use case – бұл қатысушы кейіпкерлер шығатын мінез-құлықтың сценариының сипаттамасы. Прецеденттер диаграммасы А.1-суретте көрсетілген.



А.1-сурет – Rational Rose ортасында құрылған прецедент диаграммасы

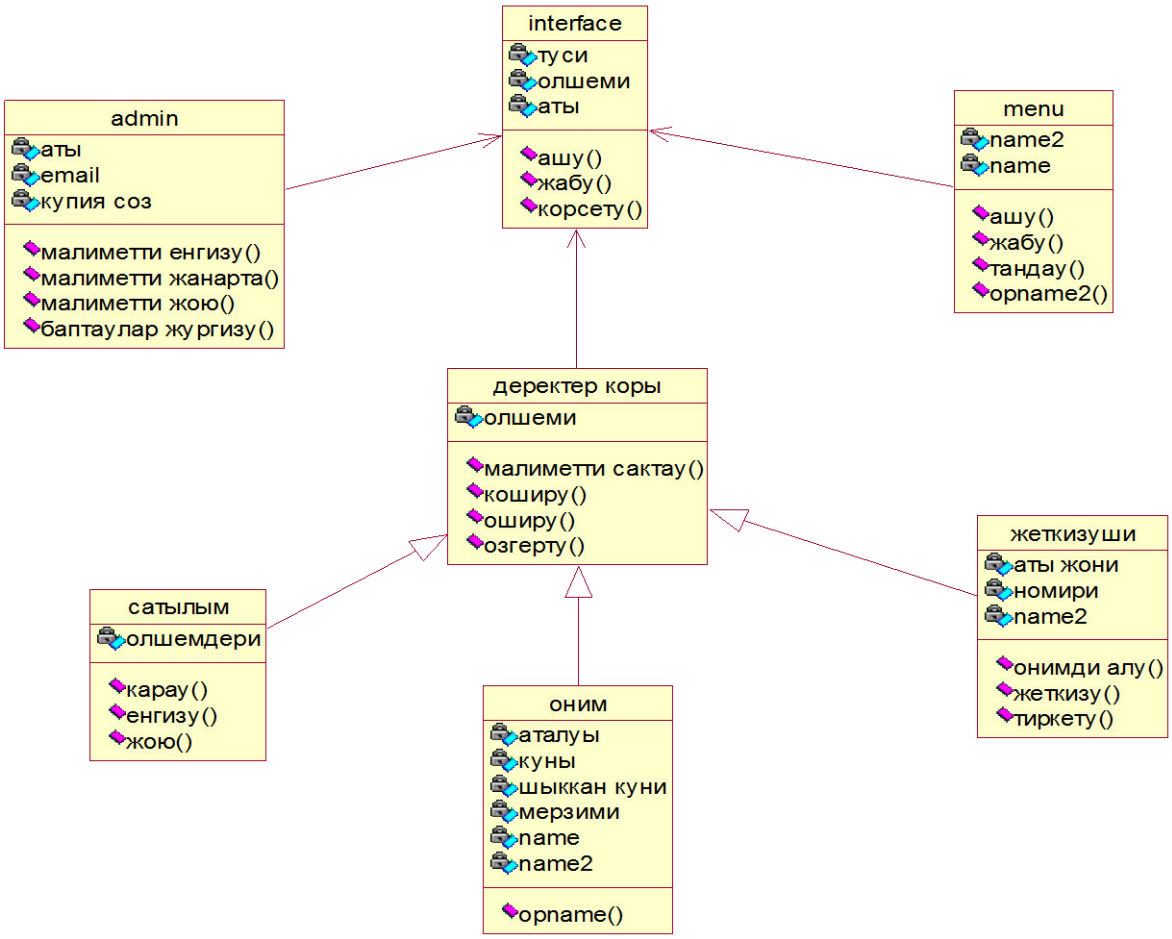
А қосымшасының жалғасы

Қолдану вариантының диаграммасы (Use Case Diagram) – бұл активті субьектілердің көптеген графикалық көрсетімі оны қолданудың, сол немесе басқа вариантардың амалдарымен өзара әрекеттеседі.

А.3.1.6 Кластар бойынша

Класс диаграммасы модельдің негізгі логикалық көрінісі болып табылады және объектті-бағытталған программалық жүйенің ішкі құрылғысы туралы немесе пргораммалық жүйенің архитектурасы туралы заманауй терминалгияны пайдаланып толық ақпараттардан тұрады. Класстар диаграммасы А.2 – суретте көрсетілген.

Класс диаграммасы – қосымша кодын енгізу үшін негізгі диаграмма классы диаграмма көмегімен жүйенің ішкі жүйесі шығарылады, яғни мұралауыды сипаттауын және қосымша класстар бір-бірімен байланысты. Осында жүйенің логикалық көрінісі сипатталады.



А.2-сурет – Rational Rose ортасында құрылған класс диаграммасы

Класс диаграммасы – қосымша кодын енгізу үшін негізгі диаграмма

А қосымшасының жалғасы

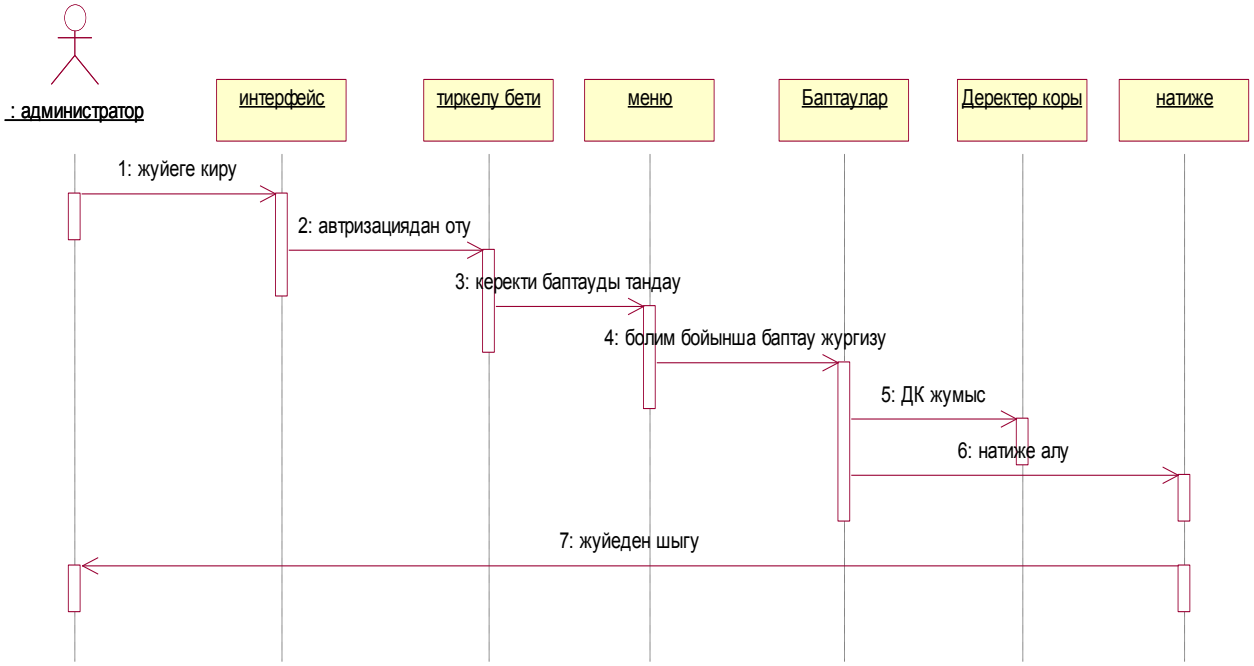
классы диаграмма көмегімен жүйенің ішкі жүйесі шығарылады, яғни мұралауыды сипаттауын және қосымша класстар бір-бірімен байланысты. Осында жүйенің логикалық көрінісі сипатталады.

А.3.1.7 Деректер ағымы диаграммасы бойынша

Тізбектелген диаграммасы уақыт бойынша объектілердің өзара бірлесу операцияларының орындалу ретін безендіреді және сценарийлермен қарастырылған функциялардың орындалу процесіндегі алмасатын объектілердің хабарламалар тізбектерінің реттелген сценарийге кіретін объектілер мен класстар бейнеленеді.

Тізбектелген диаграммалар жалпыда Logical View пакетіндегі есептелінген қолдану варианттарының жүзеге асырумен бірге ассоциацияланады. UML талаптарымен сәйкес объект тізбектелген диаграммасында тіктөртбұрыш түрінде көрсетіледі. Ол асты сызылған

объектінің атауын құрайды. Тізбектелген диаграмма А.3-суретте көрсетілген.



А.3-сурет – Rational Rose ортасында құрылған тізбек диаграммасы

Тізбектелген диаграммасы төрт негізгі элементтерден тұрады:

– прецеденттегі ізбасар мәтінінің іс-әрекеті. Ол сол жақтан жоғарыдан төменге жазылады. Сол терезеде іс-әрекет сипатталуы болып, жұмыс уақытындағы орындалатын ақпараттар қызмет етеді;

– объекттер "объект-класс" форматында аты немесе объект данасының

номері және класс объектісінің аты жазылады;

– хабарландыру. Бағытпен көрсетілген бір объектіден келесіге бағытталған іс-әрекет туралы ақпарат жолдамасынан тұрады. Белгілі бір уақытта орындалуы және осы іс-әрекеттегі жүйенің жауапты реакциясы болуы;

– әдістері (операциялар). Тікбұрыш түрінде көрсетілген. Олар үздік сызықта орналасқан. Яғни, әдістерге кіретін сол объектілер келеді. Тік бұрыш ұзындығы ізбасарда басқару фокусын көрсетуде қолдануға болады: Тікбұрыш бітетін әдіс түгелдей нүктесіне дейін басқарумен иеленеді. Бұл үшбұрыштар өмір объекті түзуі деп аталады.

А.3.2 Функциональды емес талаптар

Берілген сұрақ бойынша жауап алу үшін уақыт керек.

А.3.2.1 Өнімділігі

Жүйе сервердің максималды жүктелуі кезінде және кез-келген операциялық жүйелерде тиімді жұмыс істеуі тиіс. Көрсету және іздеу жүйелері үлкен көлемдегі мәліметтермен жұмысқа ықшамдалған болуы керек.

Web-парақтарды жүктеудің орташа уақыты қосу жылдамдығы 28.8 Кбит/с кезінде 30с аспауы керек. Бөлек парақтардың жүктелу уақытының артуы 35с-қа дейін рұқсат етіледі, бірақ сайт парақтарының жалпы санының 30 пайызынан көп болмауы керек. Басты парақтың жүктелу уақыты 40с-тан аспауы керек.

А.3.2.2 Қатынау және сенімділік

Жүйе пайдаланушылардың сенімді тоқтаусыз жұмысын қамтамасыз етіп, олардың кез-келген ісіне тоқтаусыз қызмет түрін көрсетуі керек.

Жүйе бірнеше клиенттік түйіндер қосылған кезде тиімді жұмыс істеуі тиіс және клиент – компьютерлер мен сервер-компьютердің синхронизациясын қамтамасыз етуі керек.

А.3.2.3 Қатені өңдеу

Жүйедегі барлық қателар өңделуі тиіс және қателер туралы хабарлар пайдаланушылар қабылдай алатын түрде көрсетілуі керек. Хабарлау жүйесінде

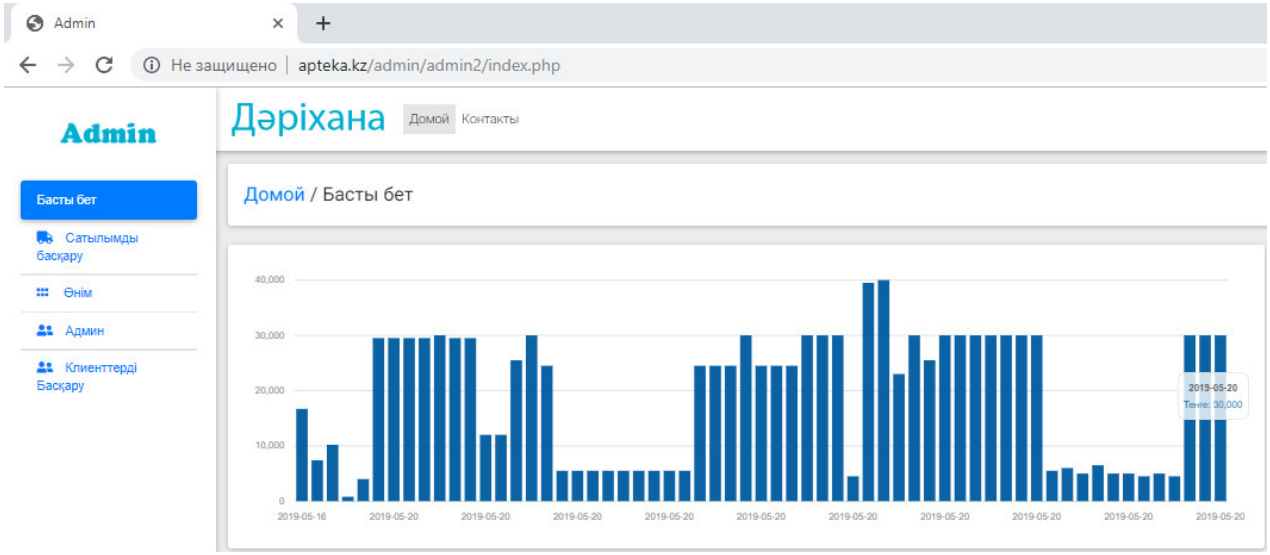
пайдаланушылардың мүмкін емес әрекеттерін, ақпарат енгізудің қате үлгілерін іске асыру керек.

А.3.2.4 Интерфейсті талаптар

А қосымшасының жалғасы

Дерекқордың құрылымынының талаптары – дерекқор навигациясы бас мәзір арқылы орындалады. Дерекқор қазақ тілдерінде құрылған. Дерекқорды өзгертуге тек администратор құқылы.

Келесі кезең басталады – АЖО сүйемелдеу, яғни кешендегі ақпараттарды өзгерту және тұрақты жаңартып отыру. Кешенді ыңғайлы және шапшаң жаңарту үшін, кешенді басқару құралдары игеріледі. Оның арқасында кешенді сүйемелдеуді – АЖО администрациялаудың негізгі түсініктерін білетін пайдаланушы асыра алады. АЖО ақпаратты жаңарту Web-интерфейс арқылы жүзеге асады. АЖО дерекқорды өзгерту администратор интерфейсі арқылы орындалады. Төмендегі суретте АЖО бас беті көрсетілген.



А.4-сурет – Администратор терезесі

А.3.2.5 Шектеулер

Әкімшілік бөлім сайт безендірілуіне және оның құрылымына өзгеріс енгізуіне мүмкіндік бермейді, тек қана оның ақпараттық бөліміне рұқсат етеді.

АЖО жұмысқа қабілеттілігі провайдер серверінің жұмысына тығыз байланысты, яғни сервер жұмысының өзгерісі дерекқор жұмысында көрсетілім табады.

А.3.3 Кері талаптар

Барлық енгізілетін мәліметтер алдын ала тексерістен өту керек.

Ә Қосымшасы
(міндетті)

Программа листингі

database.php

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
DATABASE CONNECTIVITY SETTINGS
EXPLANATION OF VARIABLES
$active_group = 'default';
$query_builder = TRUE;
$db['default'] = array(
        'dsn'    => '',
        'hostname' => 'localhost',
        'username' => 'root',
        'password' => '',
        'database' => 'torg-portal4.loc',
        'dbdriver' => 'mysqli',
        'dbprefix' => '',
        'pconnect' => FALSE,
        'db_debug' => TRUE,
        'cache_on' => FALSE,
        'cachedir' => '',
        'char_set' => 'utf8',
        'dbcollat' => 'utf8_general_ci',
        'swap_pre' => '',
        'encrypt' => FALSE,
        'compress' => FALSE,
        'stricton' => FALSE,
        'failover' => array(),
        'save_queries' => TRUE
);
```

Index.php

```
<?php
if (defined('ENVIRONMENT'))
{
   switch (ENVIRONMENT)
   {
      case 'development':
         error_reporting(E_ALL);
         break;
      case 'testing':  case 'production':
error_reporting(0);
         break;
      default:
         exit('The application environment is not set correctly.');
   }
}
/*
```

```
$system_path = 'system';
if (defined('STDIN'))
{
        chdir(dirname(__FILE__));
}
if (realpath($system_path) !== FALSE)
{               $system_path = realpath($system_path).'/';
}
// ensure there's a trailing slash
$system_path = rtrim($system_path, '/').'/';

// Is the system path correct?
if ( ! is_dir($system_path))
```

skins.css

```
/*
 * Skin: Blue
 * ----------
 */
.skin-blue .main-header .navbar {
  background-color: #09a600;
}
.skin-blue .main-header .navbar .nav > li > a {
  color: #ffffff;
}
.skin-blue .main-header .navbar .nav > li > a:hover,
.skin-blue .main-header .navbar .nav > li > a:active,
.skin-blue .main-header .navbar .nav > li > a:focus,
.skin-blue .main-header .navbar .nav .open > a,
.skin-blue .main-header .navbar .nav .open > a:hover,
.skin-blue .main-header .navbar .nav .open > a:focus {
  background: rgba(0, 0, 0, 0.1);
  color: #f6f6f6;
}
.skin-blue .main-header .navbar .navbar-custom-menu > .nav {
  margin-right: 10px;
}
.skin-blue .main-header .navbar .sidebar-toggle {
  color: #ffffff;
}
.skin-blue .main-header .navbar .sidebar-toggle:hover {
  color: #f6f6f6;
  background: rgba(0, 0, 0, 0.1);
}
.skin-blue .main-header .navbar .sidebar-toggle {
  color: #fff;
}
.skin-blue .main-header .navbar .sidebar-toggle:hover {
  background-color: #367fa9;
}
```

Ә қосымшаның жалғасы

```
@media (max-width: 767px) {
  .skin-blue .main-header .navbar .dropdown-menu li.divider {
    background-color: rgba(255, 255, 255, 0.1);
  }
  .skin-blue .main-header .navbar .dropdown-menu li a {
    color: #fff;
  }
  .skin-blue .main-header .navbar .dropdown-menu li a:hover {
    background: #367fa9;
  }
}
.skin-blue .main-header .logo {
  background-color: #ff0000;
  color: #ffffff;
  border-bottom: 0px solid transparent;
}
.skin-blue .main-header .logo > a {
  color: #ffffff;
}
.skin-blue .main-header .logo:hover {
  background: #34495E;
}
.skin-blue .main-header li.user-header {
  background-color: #3c8dbc;
}
.skin-blue .content-header {
  background: transparent;
}.skin-blue .user-panel > .info,
.skin-blue .user-panel > .info > a {
  color: #fff;
}
.skin-blue .sidebar-menu > li.header {
  color: #4b646f;
  background: #1a2226;
}
.skin-blue .sidebar-menu > li > a {
  border-left: 3px solid transparent;
  margin-right: 1px;
}
.skin-blue .sidebar-menu > li > a:hover,
.skin-blue .sidebar-menu > li.active > a {
  color: #f4f4f4;
  background: #018006;
  border-left-color: #000000;
}
.skin-blue .sidebar-menu > li > .treeview-menu {
  margin: 0 1px;
  background: #018006;
}
```

Ә қосымшаның жалғасы

```
.skin-blue .wrapper,
.skin-blue .main-sidebar,
.skin-blue .left-side {
  background: #09a600;
}
.skin-blue .sidebar a {
  color: #fff;
}
.skin-blue .sidebar a:hover {
  text-decoration: none;
}
.skin-blue .treeview-menu > li > a {
  color: #fff;
  padding-left: 40px;
}
.skin-blue .treeview-menu > li.active > a,
.skin-blue .treeview-menu > li > a:hover {
  color: #fff;
}
.skin-blue .sidebar-form {
  border-radius: 3px; border: 1px solid #374850;
  margin: 10px 10px;
}
.skin-blue .sidebar-form input[type="text"],
.skin-blue .sidebar-form .btn {
  box-shadow: none;
  background-color: #374850;
  border: 1px solid transparent;
  height: 35px;
  -webkit-transition: all 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
  -o-transition: all 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
  transition: all 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
}
.skin-blue .sidebar-form input[type="text"] {
  color: #666;
  border-top-left-radius: 2px !important;
  border-top-right-radius: 0 !important;
  border-bottom-right-radius: 0 !important;
  border-bottom-left-radius: 2px !important;
}
.skin-blue .sidebar-form input[type="text"]:focus,
.skin-blue .sidebar-form input[type="text"]:focus + .input-group-btn .btn {
  background-color: #fff;
  color: #666;
}
.skin-blue .sidebar-form input[type="text"]:focus + .input-group-btn .btn {
  border-left-color: #fff;
}
.skin-blue .sidebar-form .btn {
```

```
  color: #999;
  border-top-left-radius: 0 !important;
  border-top-right-radius: 2px !important;
  border-bottom-right-radius: 2px !important;
  border-bottom-left-radius: 0 !important;
}
.skin-blue.layout-top-nav .main-header > .logo {
  background-color: #3c8dbc;
  color: #ffffff;
  border-bottom: 0px solid transparent;
}
.skin-blue.layout-top-nav .main-header > .logo > a {
  color: #ffffff;
}.skin-blue.layout-top-nav .main-header > .logo:hover {
  background: #3b8ab8;
}
Admin.css
@import
url(//fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600,700,300italic,400italic,600i
talic);
==
html,
body {
  min-height: 100%;
}
.layout-boxed html,
.layout-boxed body {
  height: 100%;
}
body {
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  font-family: 'Source Sans Pro', 'Helvetica Neue', Helvetica, Arial, sans-serif;
  font-weight: 400;
  overflow-x: hidden;
  overflow-y: auto;
}
/* Layout */
.wrapper {
  min-height: 100%;
  position: static;
  overflow: hidden;
}
.wrapper:before,
.wrapper:after {
  content: " ";
  display: table;
}
.wrapper:after {
```

```
  clear: both;
}
.layout-boxed .wrapper {
  max-width: 1250px;
  margin: 0 auto;
  min-height: 100%;
  box-shadow: 0 0 8px rgba(0, 0, 0, 0.5); position: relative;
}
.layout-boxed {
  background: url('../img/boxed-bg.jpg') repeat fixed;
}
/*
 * Content Wrapper - contins main content
 * ```.right-side has been deprecated as of v2.0.0 in favor of .content-wrapper ```
 */
.content-wrapper,
.right-side,
.main-footer {
  -webkit-transition: -webkit-transform 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
  -moz-transition: -moz-transform 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
  -o-transition: -o-transform 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
  transition: transform 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
  -webkit-transition: margin-left 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
  -o-transition: margin-left 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
  transition: margin-left 0.3s cubic-bezier(0.32, 1.25, 0.375, 1.15);
  margin-left: 230px;
  z-index: 820;
}
.layout-top-nav .content-wrapper,
.layout-top-nav .right-side,
.layout-top-nav .main-footer {
  margin-left: 0;
}
@media (max-width: 767px) {
  .content-wrapper,
  .right-side,
  .main-footer {
    margin-left: 0;
  }
}
@media (min-width: 768px) {
  .sidebar-collapse .content-wrapper,
  .sidebar-collapse .right-side,
  .sidebar-collapse .main-footer {
    margin-left: 0;
  }
}
@media (max-width: 767px) {
  .sidebar-open .content-wrapper,.sidebar-open .right-side,
```

```
  .sidebar-open .main-footer {
    -webkit-transform: translate(230px, 0);
    -ms-transform: translate(230px, 0);
    -o-transform: translate(230px, 0);
    transform: translate(230px, 0);
  }
}
.content-wrapper,
.right-side {
  min-height: 100%;
  background-color: #ecf0f5;
  z-index: 800;
}
.main-footer {
  background: #fff;
  padding: 15px;
  color: #444;
  border-top: 1px solid #eee;
}
/* Fixed layout */
.fixed .main-header,
.fixed .main-sidebar,
.fixed .left-side {
  position: fixed;
}
.fixed .main-header {
  top: 0;
  right: 0;
  left: 0;
}
.fixed .content-wrapper,
.fixed .right-side {
  padding-top: 50px;
}
@media (max-width: 767px) {
  .fixed .content-wrapper,
  .fixed .right-side {
    padding-top: 100px;
  }
}
.fixed.layout-boxed .wrapper {
  max-width: 100%;}
/* Content */
.content {
  min-height: 250px;
  padding: 15px;
  margin-right: auto;
  margin-left: auto;
  padding-left: 15px;
```

```
  padding-right: 15px;
}
/* H1 - H6 font */
h1,
h2,
h3,
h4,
h5,
h6,
.h1,
.h2,
.h3,
.h4,
.h5,
.h6 {
  font-family: 'Source Sans Pro', sans-serif;
}
/* General Links */
a {
  color: #000;
}
a:hover,
a:active,
a:focus {
  outline: none;
  text-decoration: none;
  color: #72afd2;
}
/* Page Header */
.page-header {
  margin: 10px 0 20px 0;
  font-size: 22px;
}
.page-header > small {
  color: #666; display: block;
  margin-top: 5px;
}
/*
 * Component: Main Header
 * ----------------------
 */
.main-header {
  position: relative;
  max-height: 100px;
  z-index: 1030;
}
.main-header > .navbar {
  margin-bottom: 0;
  margin-left: 230px;
```

```
  border: none;
  min-height: 50px;
  border-radius: 0;
}
.layout-top-nav .main-header > .navbar {
  margin-left: 0!important;
}
.main-header #navbar-search-input {
  background: rgba(255, 255, 255, 0.2);
  border-color: transparent;
}
.main-header #navbar-search-input:focus,
.main-header #navbar-search-input:active {
  border-color: rgba(0, 0, 0, 0.1) !important;
  background: rgba(255, 255, 255, 0.9);
}
.main-header #navbar-search-input::-moz-placeholder {
  color: #ccc;
  opacity: 1;
}
.main-header #navbar-search-input:-ms-input-placeholder {
  color: #ccc;
}
.main-header #navbar-search-input::-webkit-input-placeholder {
  color: #ccc;
}
.main-header .navbar-custom-menu,
.main-header .navbar-right { margin-right: 5px;
```