ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты
«Программалық инженерия» кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
тех. ғыл. кандидаты,
ассоц. профессор
_____ Юнусов Р.
“_____” _____2020ж.

ДИПЛОМДЫҚ ЖҰМЫС

Тақырыбы: “Electronic Timesheet” электронды табель жүйесі

Мамандығы 5В070400 – Есептеу техникасы және бағдарламалық қамтамасыз ету

Орындаған Берікқалиева А.Б

Ғылыми жетекші
техн. ғыл. магистрі, сениор-лектор
_____ Куникеев А.
“____” _____2020ж.

Алматы 2020

Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Электронды табель - шығу табелін автоматтандыру

Название: Берікқалиева Айтолқын Бекбергенқызы

Координатор:Жибек Алибиева

Коэффициент подобия 1:1,6

Коэффициент подобия 2:0

Замена букв:6

Интервалы:0

Микропробелы:0

Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;

☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;

☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

……

…… ……

Дата *Подпись Научного руководителя*

ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

5В070400 – Есептеу техникасы және бағдарламалық қамтамасыз ету мамандығы

Берікқалиева Айтолқын Бекбергенқызы

Дипломдық жоба

Тақырыбы: «Electronic Timesheet» веб-қосымшасы

ҒЫЛЫМИ ЖЕТЕКШІНІҢ ПІКІРІ

Берілген дипломдық жобада университетіміздің барлық жұмысшыларының еңбекақысын есептеу үшін қажет болатын олардың жұмыс уақытысы мен жауапты қызмет иелерінің растау ретінде қол қоюын қамтитын құжаттың, яғни табель және акттың толтыру барысын толық электронды жүйеге ауыстыру қарастырылған. Жүйе тек Сәтбаев университетіне арналып жасалынған. Бірқатар қызметкерлердің жұмыс барысын жеңілдетуге және басты ерекшелігі – электронды қол қою жүйесін енгізуге негізделгендіктен пайдасы тиері сөзсіз.

Дипломдық жобаны Берікқалиева Айтолқын Бекбергенқызы және Ахетова Қарлығаш Ғалымқызы командалық жүйеде жоспарлы түрде жасап шықты. Зерттеу барысында жүйенің қалай іске асуын толық анықтап, оқу орнымыздың осы жүйеге қатысты барлық дерлік қызметкерлерімен кездесіп, жобаға қатысты сұрақтарына жауап алып, қосымшаны қалай жүзеге асыру қажет екенін, қандай ақпараттарды, функционалдарды қамту керек екенін жобалады. Салыстыра келе жобаны әзірлеуге Angular, Django және PostgreSQL бағдарламалар аясын таңдады. Әр бағдарламаның ерекшеліктерін қарастыра отырып, олардың қажетті мүмкіндіктерін пайдаланып, электронды табель жүйесін бірлесе жасап шығарды.

Электронды табель жүйесі акт, жұмыс табелі және демалыс табельдерін қамтиды. Тікелей қолданушылары – құжаттарды құрастырушы және дұрыстығына қарай растайтын басқарма мүшелері. Осыған орай, жүйеде табельші, растаушы және авторизация бөліктері жүзеге асырылды.

Команда мүшесі Берікқалиева Айтолқын Бекбергенқызы растаушы, авторизация интерфейсін және серверлік бөлігін, ал Ахетова Қарлығаш Ғалымқызы табельші интерфейсі мен оның серверлік бөлігін әзірледі.

Студенттер тарапынан ұсынылған жоба ыңғайлы, түсінікті және қарапайым интерфейске ие. Құрамындағы ақпараттары қолданыстағы құжаттың толтырылу стандарттарына толығымен сай.

Жұмыс барысында бар зейініндерін сала отырып, әрбір бөліктерін мұқият зерттеп, қойылған талаптарға сай және жүйелі түрде жасап, жақсы нәтиже көрсете білді.

Осы нәтижелер негізінде, Берікқалиева Айтолқын Бекбергенқызы Есептеу техникасы және бағдарламалық қамтамасыз ету – 5В070400 бакалавры академиялық дәрежесін беруге лайық деп есептеймін.

Ғылыми жетекші: «Программалық инженерия» кафедрасының техн. ғыл. магистрі, сениор-лектор _____ Куникеев А.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Кибернетика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

Мамандығы 5В070400– Есептеу техникасы және бағдарламалық қамтамасыз ету

БЕКІТЕМІН
Кафедра меңгерушісі
тех. ғыл. кандидаты
ассоц. профессор
_____ Р. Юнусов
“___” _____2020ж

Дипломдық жұмыс орындауға
ТАПСЫРМА

Білім алушы Берікқалиева Айтолқын Бекбергенқызы
Тақырыбы : «Electronic Timesheet» электронды табель жүйесі
Университет Ректорының *20___жылғы "___" ____ №__-б* бұйрығымен бекітілген
Аяқталған жұмысты тапсыру мерзімі *20 __жылғы "___"_____*
Дипломдық жұмыстың бастапқы берілістері: Ұсынылатын дипломдық жобада электронды табель және акт құжаттарын толтыру веб-қосымшасын жасау.
Дипломдық жұмыста қарастырылатын мәселелер тізімі:
а)зерттеу бөлімі;
б)жобалау бөлімі;
в)технологиялық бөлім;
г) жоба құрылымы;
д) А Қосымшасы. Техникалық тапсырма;
е) Б Қосымшасы. Бағдарлама мәтіні.
Сызба материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс)
Жобаның презентациялық 15 слайды ұсынылған
Ұсынылатын негізгі әдебиет *12 әдебиеттер тізімінен тұрады.*

Дипломдық жұмысты (жобаны) дайындау

КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	**Ғылыми жетекші мен кеңесшілерге көрсету мерзімдері**	**Ескерту**
1. Дипломдық жоба тақырыбының қолданылу аясы мен өзектілігін зерттеу, жүйеге қатысты тұлғалармен кездесіп, сұрақтарға жауап алу.	20.01.2020	
2. Кіріс ақпараттарын жинау, UML тілінде жобалау: тізбек және прецеденттер диаграммаларын сызу. Деректер қорының жобасын құру.	01.02.2020	
3. Жоба интерфейсін құру, әзірлеу технологияларын таңдау.	05.02.2020	
4. Жобаның веб-қосымшасын әзірлеу.	10.03.2020	
5. Дипломдық жобаны диплом алды есеп алу комиссия мүшелеріне таныстыру, ескертулерін түзетіп дипломдық жобаны аяқтау. Қорытынды жасау.	27.04.2020	
6. Сипаттау мәтіні мен сызба материалды аяқтау.	20.05.2020	

Дипломдық жұмыс (жоба) бөлімдерінің кеңесшілері мен норма бақылаушының аяқталған жұмысқа (жобаға) қойған қолтаңбалары

Бөлімдер атауы	**Кеңесшілер, аты, әкесінің аты, тегі (ғылыми дәрежесі, атағы)**	**Қол қойылған күні**	**Қолы**
Бағдарламалық бөлім	Куникеев А. Тех. ғыл. магистрі, сениор-лектор		
Нормалық бақылаушы	Марғұлан Қ. Тех. ғыл. магистрі, лектор		

Ғылыми жетекші _____ Куникеев А.Д.

Тапсырманы орындауға алған білім алушы _____ Берікқалиева А.Б

Күні "_____"_____2020 ж.

АҢДАТПА

Жобаны сипаттауға арналған мәтін кіріспе, негізгі бөлімді құрайтын үш тарау (зерттеу, технологиялық, жобалау), қорытынды, пайдаланылған әдебиеттер тізімі және қосымшаны қамтиды. Жалпы 46 беттен, 8 сурет, 2 қосымша және 1 кестеден тұрады. Кестеде термин сөздер мен қысқартылған сөздердің анықтамасы түсіндірілді. Жұмысты жазуға 12 әдебиеттер мен мақалалар қолданылды.

«Electronic Timesheet» жобасы Сәтбаев университетінің табель және акт толтыру процесін электронды жүйеге ауыстыру үшін жасалды. Аталған құжат қызметкерлердің жұмыс уақытысы және жауапты тұлғалардың растау ретінде қол қоюынан тұрады.

Алдағы уақытта оқу орнымыздың белгілі бір құрылымдарының өзгеріп, жаңаруы құралдың ары қарай да қолданыста болуына кедергі келтірмейді, веб-қосымша өзгерістер енгізуге болатындай жобаланды.

Қосымшаның қолданысқа ие болуы жүйелі, ыңғайлы интерфейс пен тартымды дизайн негізінде жасалуына тікелей байланысты. Бұл тұста жобаның құрылымын толық зерттеу барысында танымал бағдарламаларды салыстыра отырып, жүйеге қолайлы деп таңдалған Angular, Django және PostgreSQL озық технологияларының мәтінде айтылатын көптеген мүмкіндіктері пайдаланылды.

Жоба құжатқа электронды қол қою заманауи мүмкіндігімен ерекшеленеді. Бірқатар функционалдары автоматтандырылғандықтан қолданушы қызметкерлердің жұмыс үрдісі жеңілдеп, уақыттарын біршама үнемдеуге көмектесері сөзсіз.

АННОТАЦИЯ

Текст для описания проекта включает введение, три главы составляющие основную часть (исследовательские, технологические, проектные), заключение, список использованной литературы и приложение. Всего 46 страниц, 8 рисунков, 2 приложения и 1 таблица. В таблице объясняется определение терминов и аббревиатур. Для написания работы использовались 12 литературы и статьи.

Проект "Electronic Timesheet" разработан для перевода в электронную систему процесса заполнения табеля и акта университета Сатпаев. Данный документ состоит из учета рабочего времени сотрудников и подписи ответственных лиц в качестве подтверждающих.

Дальнейшие изменения и обновления определенных структур вуза не влияет на функционирование веб-приложения и спроектировано таким образом, что можно было внести дополнительные изменения.

Использование приложений напрямую зависит от создания системного, удобного интерфейса и привлекательного дизайна. Для этого в результате полного изучения структуры проекта использовано множество возможностей, излагаемых в тексте передовых технологий, такие как Angular, Django и PostgreSQL, которые были выбраны как приемлемые для системы, сравнивая популярные программы.

Проект отличается современной возможностью электронной подписи документа. В связи с тем, что был автоматизирован ряд дополнительных функционалов, проект облегчает процесс работы сотрудников и значительно сэкономить время.

ANNOTATION

The text for the project description includes an introduction, three chapters that make up the main part (research, technology, project), conclusion, list of references and appendix. A total of 46 pages, 8 figures, 2 appendices and 1 table. The table explains the definition of terms and abbreviations. 12 literature and articles were used for writing the work.

The project "Electronic Timesheet" is designed to translate the process of filling out the report card and act of Satbayev University into an electronic system. This document consists of a record of employees ' working hours and the signatures of responsible persons as confirmatory documents.

Further changes and updates to certain university structures do not affect the functioning of the web application and are designed in such a way that additional changes can be made.

The use of applications directly depends on creating a system, user-friendly interface and attractive design. For this purpose, as a result of a full study of the project structure, many features outlined in the text of advanced technologies, such as Angular, Django, and PostgreSQL, were selected as acceptable for the system, comparing popular programs.

The project has a modern electronic document signature capability. Due to the fact that a number of additional functions have been automated, the project facilitates the work of employees and significantly saves time.

МАЗМҰНЫ

КІРІСПЕ

Қазіргі таңда заман мен техниканың, әртүрлі ақпарат көздері мен ақпарат тарататын құрылғылардың үздіксіз дамып келе жатуы көптеген мүмкіндіктерге жол ашуда. Мұндай мүмкіндіктерде электрондық есептеуіш машиналардың алатын орны ерекше. Олар – күнделікті өмірде, жұмыс орындарында үздіксіз қолданыста жүрген компьютер, ноутбуктар. Осы құрылғыларды пайдалана отырып, түрлі программалау тілдері арқылы күнделікті жұмыс барысымызға қажетті қосымшаларды жасап шығаруымызға мүмкіндіктер жетерлік. Қосымшаларға күннен-күнге сұраныстар саны да артуда. Осы орайды пайдаланып «Electronic Timesheet» қосымшасы таңдалынды.

Жасау барысында соңғы жылдарда жоғары сұранысқа ие және болашағы сенімді бағдарламалар қолданылды. Кез келген қосымша жасау кезінде клиенттік, серверлік бөлімдер және өзіндік дерек қоры болатыны белгілі. Жобада көш бастаушы Angular, Django және PostgreSQL платформалары негізге ие болды. Клиенттік фреймворк ретінде таңдалынған Angular – Google компаниясының командасымен жүзеге асырылған, веб-қосымшаларды әзірлеуге арналған ашық және еркін платформа. Ұсынылып отырған қосымшада TypeScript тілінде жазылған нұсқасы қолданылды. Django – күрделі сайттар мен веб-қосымшаларды дайындау үшін Python бағдарламалау тілі негізінде жасалған, үлкен мүмкіндіктерге ие бағдарламалық каркас. Ал деректер қоры реляционды сипатта болғандықтан әлдеқайда дамыған, ашық және еркін объектілік-реляциялық жүйе PostgreSQL-мен жасалынды.

Қосымшаның қолданылу аймағы – оқу орнымыздың акт және табель толтыруға атсалысып жүрген қызметкерден барлық жұмысшылардың еңбекақысын есептеп шығарып жүрген қызмет иелеріне дейін қажет болатын жұмыс уақытысын электронды түрде қысқа уақыт ішінде көрсете алу мүмкіндігін қамтамасыз ету үшін бағытталған.

Дипломдық жұмыстың мақсаты: Университетіміздегі барлық қызметкерлердің еңбекақысын есептеу барысында олардың жеке жұмыс уақытысы туралы мәліметтерден тұратын, қағазбастылықты жоюға үлкен септігін тигізетін электронды табель жасап, ұсыну. Осы міндеттерді атқарып жүрген қызметкерлерге уақытын тиімді жұмсап, жұмыс барысын жеңілдетуіне бірден бір септігін тигізеді. Оқу орнымыздың өте үлкен және қызметкерлерге толы болуына байланысты бұл электронды табельдің пайдасы тиеді деген үміттеміз.

Техниканың дамыған заманына қарамастан, құжаттарға белгілі бір қызмет иелеріне қол қойдыру қызмет орындарында әлі де тоқтау таппады. Сол үшін де электронды қол қою жобаның негізгі жаңашылдығы болып табылады. Уақыт, қаражат және қолайлылық жағынан тиімді болатыны сөзсіз.

1 Зерттеу бөлімі

1.1 Жобаның өзектілігі

Қазіргі уақыт – мүмкіндіктер уақыты. Технологияның, әр түрлі электронды машиналардың, құрылғылардың күннен-күнге тоқтаусыз дамып келе жатуы соның дәлелі. Дегенмен, атап көрсетілген құрылғыларға қосымшаларын енгізбесек, пайдасы айтарлықтай көп бола қоймайды. Сол үшін де оларды кеңінен қолдану үшін түрлі қажетті қосымшалар жасап шығарылып отырса, адамдардың өміріне пайдасын тигізетіні сөзсіз. Білім саласы болғандықтан, осы тараптағы мүмкіндіктердің артуы бір орында тұрып қалмай, ары қарай дамып, жоғары көрсеткіштер көрсетуге жол көрсетеді. Осы себептерді негізге ала отырып, электронды табель қосымшасының да орны бөлек болады деген сеніммен, қолданысқа ие болатындай түрлі әдістерді қолдана отырып, жасап шығаруды ұйғардық.

Атап көрсетілген бағдарлама тек Қ.И.Сәтбаев университетіне арналып жасалынды. Оқу орнымыздағы бірқатар қызмет етушілердің жұмысын жеңілдетуге тікелей негізделген қосымша болып саналады. Осы уақытқа дейін табель құрастырушы қызметкер оларды қағазға шығарып, жауапты тұлғаларға қол қойдырып жүрген болса, табельге жауапты тұлға қосымша арқылы тек өз аккаунтына кіріп, жұмысшыларды таңдап, жауапты қызмет иелеріне жіберсе жеткілікті. Жобаның маңыздылығы осында.

Бұл дипломдық жобада қосымша тек веб-парақша ретінде жүреді. Маңыздылығы жоғары құжаттармен жұмыс істейтіндіктен жүйе мобильді қосымшада жасалынуы тиімсіз болады.

1.2 Веб-қосымшалардың маңызы

Веб-қосымша – клиенттің браузер арқылы веб-сервермен өзара әрекеттесетін серверлік қосымша [1]. Олардың артықшылықтары:

1) қолданушы жұмысын толыққанды орындауы үшін өзінің компьютеріне ауыр бағдарламаны орнатып әуре болмайды, тек операциялық жүйемен бірге орнатылған браузер және ғаламтор болса жеткілікті;

2) компьютерге қосымшаны орнатып қана қоймай, администраторлық қызметке де жауапкершілік алу қажет болады, сонымен қатар бір бөлігі дұрыс болмай қалған жағдайда, тез арада жөндеуді қажет етеді. Бұл жағдайлар техникада тәжірибесі азырақ қолданушыларға ыңғайсыздық тудырып жатады. Біздің жағдайда серверде жатқандықтан, аталған мәселелерді алаңдамауға болады;

3) сонымен қатар жаңартумен мәселе туындамайтынын атап өтуге болады, жаңа нұсқасы дайындалған жағдайда автоматты түрде жаңарту жүреді;

4) веб-қосымшалар өз қолданушыларына еркін болуына үлкен мүмкіндік береді: ғаламтор қол жетімді жерлерде өз жұмысын атқаруына болады;

5) операциялық жүйе немесе браузер таңдамайды.

1.3 Қолданыстағы табель жүйесіне шолу

Табель толтырушы қызметкер иелігінде екі маңызды құжат толтырылады: акт және табель. Олар – әрбір қызметкердің жұмыс уақытысы көрсетілетін тізім. Акт – қосымша табыс үшін, ал табель – айлық табыстары үшін толтырылады. Қазіргі күнге дейін оқу орнымызда жұмысшылардың табыстарын есептеп шығару үшін осы тізімді жасауға жауапты тұлғалар болады. Олар әр кафедраның бір қызметкеріне және жұмысшылар тобының өз каменданттарына жүктеледі. Құжаттарды толтырудың орындалу тәртібі:

1) жауапты тұлға белгілі бір дайын кестеге қажетті жұмысшының есімін, жұмыс жасау ставкасын, әр күні қанша сағат қызмет жасағанын енгізіп, жалпы бір айлық немесе белгіленген мерзімдегі сағат санын есептеп жазады;

2) дайын болған акт немесе табельді А4 парағына шығарып, оны растауы қажет болатын тұлғаларға, яғни, кафедра басшысына тексертіп, олардың растағанын көрсету үшін қол қойдыруы қажет;

3) тексеру барысында қателер табылған жағдайда, табель құрастырушы көрсетілген қателерді түзетіп, жауапты тұлғаларға өз ретімен тағы да қол қойдырып шығуы қажет болады;

4) жоғарыдағы тәртіп барысын орындап болған соң ғана табель бухгалтер қызметкерлерінің қолына өтеді, сол арқылы университетіміздің әрбір жұмысшыларының еңбекақысы тағайындалады.

Кез келген жұмыс орнында жұмыскерлердің еңбекақысын нақты және дұрыс есептеп шығару өте маңызды. Сондықтан да осы міндетті атқару жоғары жауапкершілікті талап етеді. Жоғарыда келтірілген тәртіпке мән берер болсақ, мәліметтерді енгізу барысында қате жіберіліп қойылуы әбден мүмкін. Сонымен қатар, оқу орнымыздың үлкен және қызметкерлерге толы болуы табельге жауапты тұлғаларға уақыт жағынан, тіптен физикалық тұрғыдан үлкен ауырлықтар тудыратыны белгілі. Осындай мәселелердің алдын алу негізінде жаңа жоба ойластыру үлкен маңызға ие.

Сипатталған табель мен «Electronic Timesheet» жобасын салыстырып көрер болсақ, айырмашылықтары жетерлік. Негізгі артықшылықтары:

– қағазбастылықты біршама азайту;
– электронды қол қойдыру мүмкіндігі;
– шығынды көп болдырмау;
– уақытты үнемдеу;
– бірқатар қызметкерлерге жеңілдік;
– мәліметтер сенімді жерде сақталады.

Техниканың үздіксіз дамып келе жатқанына қарамастан, көптеген жұмыс орындарында қағазбастылық әлі де жойыла қоймады. Бұл жағдай уақыт жағынан болсын, қаражат жағынан болсын теріс жақтары бар екені белгілі. Толтырылған табель маңызды құжат болып саналатындықтан, оны өз уақытысымен сақтап отырудың өзі үлкен жұмыс. Ал осыны шеше отырып, жаңа мүмкіндіктерге жол аша аламыз. Сондықтан да қолмен толтырылып жүрген табельді толығымен электронды түрге көшіру оқу орнымыз үшін жаңаша дүние болмақ.

1.4 UML тілінде жобаны модельдеу

Кез келген жобаны бастамас бұрын зерттеу барысында жобаның ішінде орындалатын іс-әрекеттерді, функциясын ойластырып, модельдеу қажет. Бұл жұмыс барысын толық зерттеуге, оны барынша жете түсінуге және қателерді болдырмауға көмектеседі. Жүзеге асыру үшін Unified Modeling Language (UML) ұғымы қолданылады.

UML – талдау және жобалау үшін қолдануға болатын объектілі-бағытталған белгілер жүйесі [2]. Бағдарламалық жүйелерді спецификациялау, визуализациялау, құрастыру және құжаттау үшін пайдалануға болады. Бұл тіл өте күрделі және аса ауқымды жүйелерді модельдеуде бұрыннан пайдаланылатын озық технологиялардың үйлесімін көрсетеді.

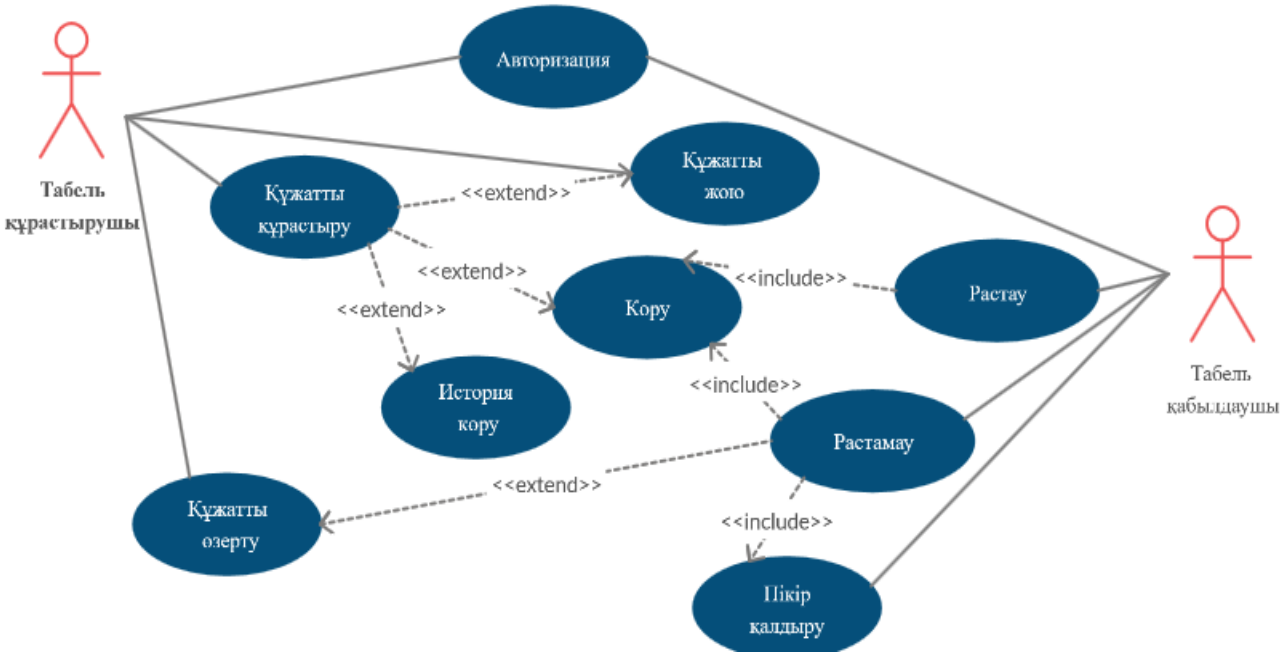
Көптеген бағдарламашылар күрделі жұмыс барысында бұл кезеңді елемей, жобалау – бұл уақытты жоғалту және ол тек кедергі келтіреді деп ойлайды. Бірақ мұндай жағдайда алдын ала жоспарлау және модельдеу бағдарламалауды едәуір жеңілдетеді. Сонымен қатар, бастапқы кодқа қарағанда кластар диаграммаларына өзгерістер енгізу жеңілірек. Егер зерттеу барысында жұмысты алдымен қағазға сипаттау, бұл да бір жетістік.

Модельдеу үшін UML тілінің арнайы диаграммаларын қолдануға болады. Бағдарламаның құрылымын анық көрсетуге, оның қалай жұмыс жасайтынын көрсетуге және жасалған өзгерістерді диаграмма түрінде сызып алу маңызды. Оның көптеген түрлері бар. Әр диаграмма әр түрлі қызмет атқаратындықтан, бағдарламаға қажеттісін таңдап алған жөн. Олар негізгі 3 топтан тұрады: құрылымдық, мінез-құлық және өзара әрекеттесу диаграммалары [2]. Әрқайсысы өз сипаттары және міндеттеріне байланысты өзара тағы бірнеше түрлерге жіктеледі.

Дипломдық жұмыста құрылатын жүйенің моделі мінез-құлық бөлігінің ішінде пайдалану нұсқаларының диаграммасы, яғни прецеденттер диаграммасы арқылы жасалынды.

Пайдалану нұсқаларының диаграммасы (Use Case Diagram) – пайдалану нұсқаларының топтары мен процеске қатысатын әрекет етуші адамдардың арасындағы өзара қарым-қатынасты және тәуелділікті сипаттайтын құжаттық модель [3]. Пайдаланушы бағдарламалық және жүйелік жобалау кезінде белгілі бір мақсатқа жету үшін, жүйені қалай пайдалану қажет екенін түсіндіру және

түрлі қателіктерді болдырмау үшін қолдана алады. Диаграмма жасалынған жүйенің ішкі құрылымын сипаттауға арналмаған. Жүйедегі әр қолданушының иелігіндегі іс-әрекеттерді көрсетуге және оларға барынша түсінікті болуына негізделген [3]. Атап айтқанда, процеске қатысатын әрекет етуші адамдар мен прецедент элементтері арасындағы өзара қарым-қатынасты және тәуелділікті сипаттайды. Демек, актер, прецедент элементтері және олардың қатынастар жиынынан тұрады. Қандай да бір қолданушыны немесе субьектіні бағдарламада белгілеу үшін актерді пайдаланамыз. Жобамызда қолданушылар екіге жіктеледі: табель құрастырушы және табель қабылдаушы. Аталған жобаны модельдеу түрі төмендегі 2.1-суретте көрсетілген.



1.1-сурет – Прецеденттер диаграммасы

2 Технологиялық бөлім

2.1 Фреймворк ұғымына түсінік

Дипломдық жоба барысында соңғы жылдары күрделі бағдарламар жасап шығаруда қолданысқа ие болып жүрген Angular және Django фреймфорктары таңдалынды. Аталған ұғымдар келесі бөлімшелерде жеке анықталатын болады.

Фреймворк – қосымша құруды жеңілдетуге және техникалық күрделі немесе ауыр жобаларды қолдануға арналған бағдарламалық өнім [4]. Оның екі негізгі функциясы бар : клиенттік бөлімдегі жұмыс (фронтенд) және серверлік бөлімдегі жұмыс (бэкенд).

Клиенттік фреймворк – қолданушы оқу, экранға шығару және іске қоса алу мүмкіндіктері анықталатын, қолданушыға көрінетін қосымшаның сыртқы бөлігі. Қазіргі таңда клиенттік бөлікті жүзеге асыратын көптеген фреймворктар қолданушыларға ұсынылады, ең танымалдары:

– Angular;
– React;
– Vue.js;
– Foundation;
– Ember;
– Backbone.js.

Серверлік фреймворк – пайдаланушы браузерінде немесе компьютерінде мүлде көрінбейтін, қосымшаның логикасы іске асырылатын, ішкі серверлік бөлігіне жауап беретін фреймворк. Бағдарламашылар арасындағы көп қолданыстағылары:

– Node.js (Express);
– Django;
– Rails;
– Laravel;
– Spring.

Қарастырылатын дииплomдық жобадағы жүйенің құрылымының ажыратылуына бір функционалымен мысал келтірер болсақ: табельге жауапты тұлға жаңа акт не табель құрады, толтыру барысында керекті мәліметтер енгізеді және ол қол қоюшы тұлғаларға бірден жіберледі. Мұндағы табель не акттың мәліметтер базасында сақталу және керекті тұлғаларға жіберілу процесінің қалай жүзеге асатыны қолданушыға белгісіз, процесс көрінбейді. Бұл функциялардың жүзеге асуын бэкенд бөлігі қамтамасыз етеді. Ал фронтенд бөліміне қолданушының жүйеге логин мен құпиясөз енгізуі, табель толтыру барысында керекті формаларға мәліметтер енгізуі, құрылған табельдің статусын көруі секілді қолданушы әрекеттері жатады.

Бизнес-логикасы бар және жоғары жылдамдықты, сенімділік пен қауіпсіздікті талап ететін жобаларда SaaS, CMS немесе CMF секілді платформалардың түрлеріне қарағанда фреймворктарды пайдалану тиімді [4].

Фреймворк қолданбай бағдарлама әзірлеуші өзі қалаған кез келген қосымшаны толығымен жаза алады. Бірақ жазылған қосымша әртүрлі платформада іске асуы үшін қосымша үлкен кодтауды қажет етеді. Себебі фреймворк көптеген дайын компоненттерді қамтиды, басты мақсаты – әзірлеушіге бағдарламаны оңай және аз уақыт аралығында құруды қамтамасыз ету.

2.2 Клиенттік бөлім – Angular фреймворкі

Angular – веб-қосымшаларды әзірлеу үшін жасалынған ашық платформа. Google және басқа да қауымдастықтың бағдарламашыларының негізінде әзірленген. Атауы барлық дерлік тегтерде қолданылатын “<>” таңбасының негізінде қойылған.

Фреймворктың екі танымал нұсқасы бар: AngularJS және Angular [5]. Айырмашықтары: бірі JavaScript, екіншісі TypeScript тілінде жазылуында және өзгеру жаңашылдығында. Angular соңғы шыққан нұсқа болғандықтан, қолданушыларға тиімді мүмкіндіктері жетерлік.

2.2.1 Angular ерекшелігі

Angular фреймворкінің ерекшеліктері:

1) аз уақыттың ішінде танымал фреймворктардың қатарына қосылып, қазіргі таңда 1,5 миллионнан астам бағдарламашылардың қолданысында;

2) NBC, Intel, ABC News секілді танымал 9000-нан астам сайттарда қолданылады;

3) angular core team атты қауымдастығы қалыптасқан. Қауымдастықта Google жүйесінің 20 қызметкері және Angular жобасының барлық мықты әзірлеушілері қолданушылардың тарапынан туындаған кез келген мәселесін шешуде көмектеседі;

4) туториалдар мен бағыттаушылар жетерлік, фреймворкті оқып меңгеруге негізделген компаниялар саны жылдан жылға артып келеді;

5) фреймворк көптеген шабуылдарға төтеп бергендіктен, қолданыс аясын кішірейткен емес.

Құрылымы арқасында бұл фреймворкті қолданатын бағдарламашылар көбеюде. Осы аталған көрсеткіштерден әлі де ұзақ жылдар қолданыстан шықпайтыны байқалады. Жоғарыда айтылған себептер негізінде дипломдық жобада фреймворктің Angular нұсқасы таңдалынды.

Фреймворк орнатылуының өзіндік жолдары бар. Ресми сайттан керекті операциялық жүйеге сәйкес *node.js* файлы жүктеліп, орнатылуы қажет [5]. Орнатылуы кезінде Angular-дың менеджер пакеті болып саналатын *npm* автоматты түрде бағдарлама әзірленетін ортаға (компьютер, ноутбук) жүктеледі. Ортаның командалар жолында:

1) npm install -g@angular/cli – Angular CLI инфраструктурасы ортада глобалды орнатылады;

2) npm install – жобаға керекті модульдар орнатылады*;*

3) ng new name – жаңа жоба ашу, name сөзінің орнына жоба аты жазылады;

4) ng serve – жоба localhost:4200 адресінде тестілеу режимінде қосылады.

2.2.2 Фреймворк құрылымы

Angular құрылымы бірнеше бөліктерден тұрады: компонент, шаблон, директива, сервис және модуль [5].

Фреймворк модульдік архитектурамен сипатталады және ол NgModule модульдік жүйесі деп аталады. Әрбір Angular бағдарламасында AppModule деп аталатын түбірлік модуль бар, ол app.module.ts файлында орналасады. Қолданушы өзіне қажетті қосымша функционалды модульдер құра алады. Оны түбірлік модуль орналасқан файлда импорттау қажет.

Маңызды қасиеттері:

– declarations – модульге қатысты компоненттер жарияланатын бөлік;

– exports – басқа модульде қолданылатын осы модульдің компоненттер мен директивалар тізімі;

– imports – осы модульде қолданылатын басқа модульдер тізімі (кітпханалар);

– providers – модульдегі сервистер тізімі;

– bootstrap – браузерде жүктелетін компонент атауы. Әдетте, бұл түбір компоненті болады және бұл қасиет тек түбірлік модульде жазылады.

Компонент – қолданушы интерфейсінің негізгі құраушы бөлігі [5]. Оның қолданушыға көрінетін бөлігін шаблон, ал әрбір қызметінің орындалуын сервис қамтамасыз етеді. Сонымен қатар, компоненттердің өз әдісі, қасиеттері, кіріс және шығыс оқиғалары болады. Анықтайтын негізгі объекттері:

– selector –компонент атауы;

– template URL – HTML парағын компонентпен байланыстыру;

– style URLS – CSS файлымен байланыстыру;

Data Binding – компонент пен шаблон арасындағы мәлімет алмасуды қамтамасыз етеді [5]. Angular оның бірнеше түрін ұсынады:

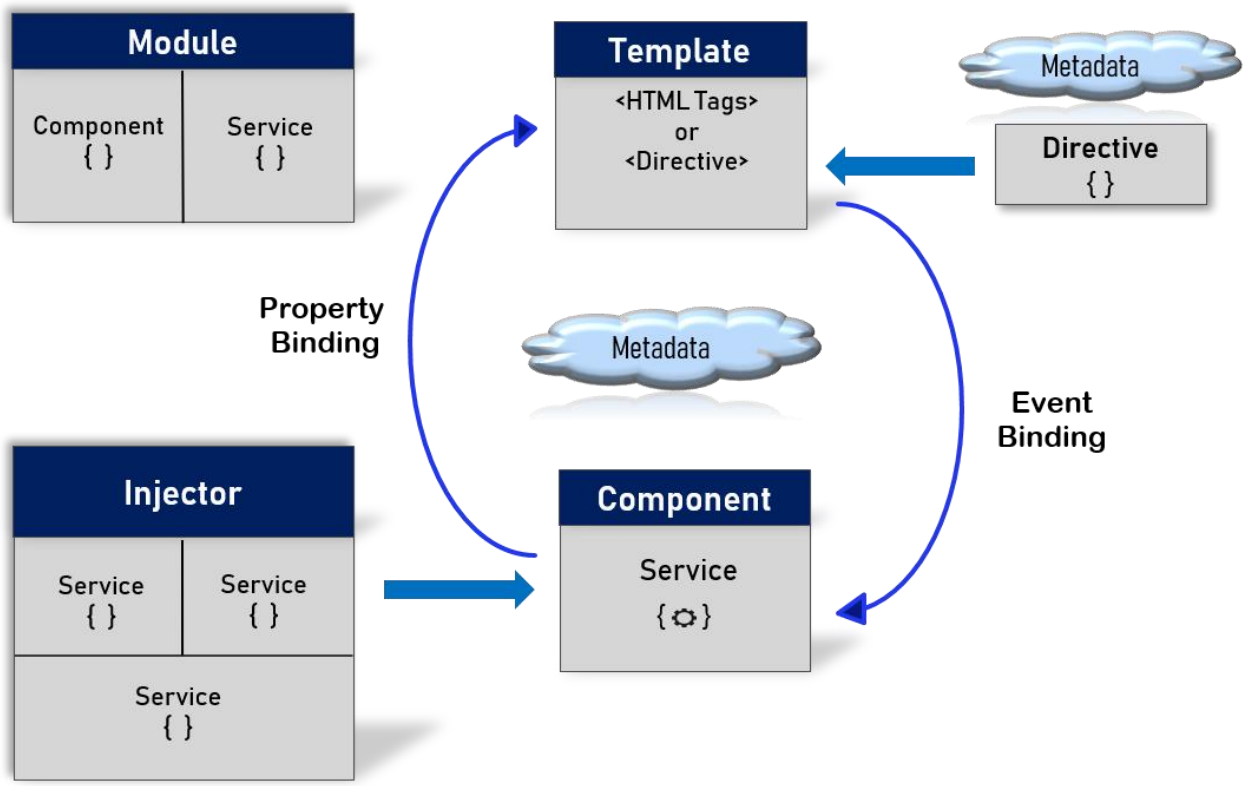
1) interpolation – компонент қасиетін экранға шығару. Белгіленуі: {{title}};

2) property binding – DOM сипатын және кіріс оқиғаларын өзгертуге қолданылады;

3) event binding – элемент оқиғасын анықтайды. Мысалы: батырманың басылуы және dropdown тізімінен таңдау;

4) two-way bata binding – шаблон мен компонент бөлігін тез байланытыруды қатамасыз етеді. Бір жақтан өзгеріс енгізілсе автоматты түрде екінші жақтан да өзгертіледі.

Жоғарыда аталған фреймворк архитектурасының құраушы бөліктерінің арасындағы байланыс төмендегі суретте көрсетілген.



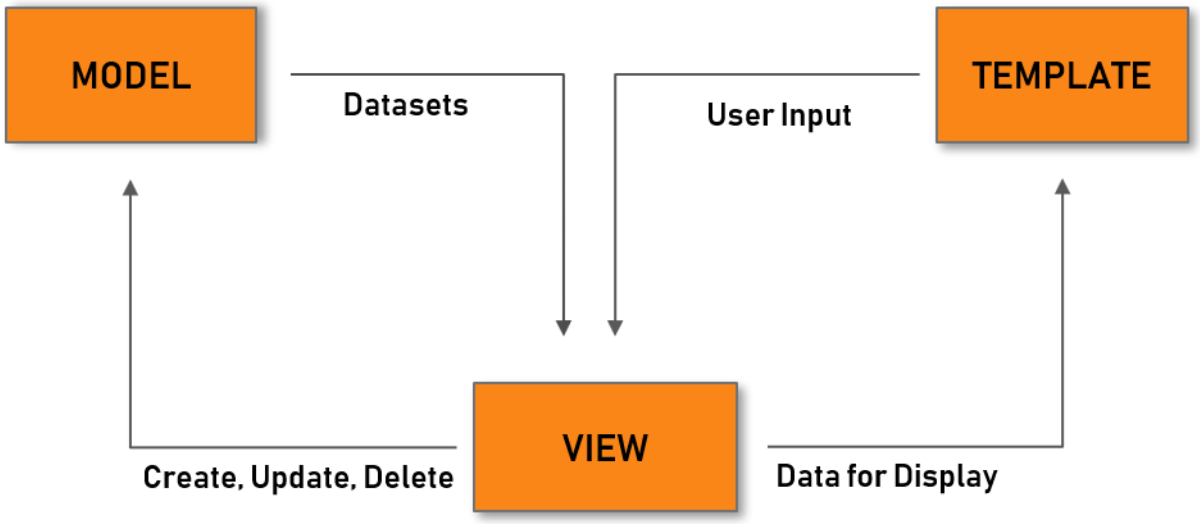
2.1-сурет – Angular құрылымы

2.3 Серверлік бөлім – Django фреймворкі

Django – веб-қолданбалар әзірлеуге арналған MVC жобалау үлгісін пайдаланатын Python тіліндегі тегін фреймворк. Оны Lawrence-Journal World газетінің Эдриан Головатый және Саймон Виллисон атты әзірлеушілері басылымның жаңалықтарын жариялау үшін ғаламтор сайты қажет болғандықтан жүзеге асырды [6].

Django архитектурасы "модель-көрініс-контроллер" (MVC) жүйесімен сипатталады [6]. MVC классикалық моделінің контроллері Django-да көрініс деп

аталатын деңгейге сәйкес келеді (View), ал таныстыру логикасы Шаблондар деңгейімен жүзеге асырылады (Template). Осыған байланысты Django деңгейлік архитектурасын жиі "модель-үлгі-көрініс" (MTV) деп атайды. Архитектурасы бір қарағанда қиын болып көрінгенімен, қолданушыларға көп мүмкіндіктер жасалынған. Төменде көрсетілген суреттен байқауға болады.



2.3-сурет – Django архитектурасы

Сонымен қатар, фреймворк құрылымында DRY (don’t repeat yourself) принципі жақсы әзірленген [6]. Олар қосымша құру барысында қолданушыларға жеңілдік тудыру мақсатымен жасалған. Яғни, бір кодты қайтадан бірнеше рет жазып отырудың қажеті жоқ. Жылдам даму жобаларында өте тиімді, Python тілінде түсінігі бар бастаушы адамдарға қолайлы. Django прагматикалық және таза дизайн үлгісі бойынша салынған және күрделі веб-қосымшаларды жасау үшін қажетті барлық негізгі компоненттерді қамтиды.

2.3.1 Django немесе Node.js

Әрбір фреймворк өзіндік артықшылықтарға ие. Жоба түріне, құрылымына байланысты әзірлеуші жұмыс барысына қолайлысын таңдағаны жөн. Дипломдық жобаның серверлік бөлімінің бағдарламасын таңдауды қорыта келгенде екі нұсқа болды: Django немесе Node.js.

– таңдау кезінде басымдылық деректер қорына берілді, Django фреймворкінде деректер қорымен жұмыс оңтайландырылған. Ал Node.js-те мәліметтердің едәуір санымен жұмыс жасағанда жоғары өнімділікті қажет етеді;

– node.js – браузердің клиенттік ортасынан алынған JavaScript орындау ортасы, ал Django-Python ортасы. Осы құралдармен жұмыс жасау үшін, олардың

негізгі бағдарламалау тілімен жұмыс істеу ыңғайлы болуы керек. Ал соңғы уақыттарда Python тілінің қолданылу аясы кеңейіп келе жатыр;

– node.js үшін көптеген онлайн оқулықтар бар, алайда көптеген мысалдар толық қарастырылмаған, бұл бағдарламалауды жаңадан үйреніп жүрген қолданушыларға айтарлықтай қиын мәселе тудырады;

– node.js -ты қолданатын танымал атаулар: Uber, Twitter, eBay, Netflix, DuckDuckGo, PayPal, LinkedIn, Trello, PayPal, Mozilla и GoDaddy. Ал жобада таңдалған фреймворктің танымал қолданушылары: Pinterest, Instagram, Eventbrite, Sentry, Zapier, Dropbox, Spotify и YouTube [7].

Осындай салыстыру дәлелдемесі негізінде және фреймворктің қолайлы архитектурасы, өзіндік ерекшеліктері назарға алынып, жобаның серверлік бөліміне Django таңдап алынды.

2.3.2 Django фреймворкінің өзіндік ерекшеліктері

1) *Әкімшілік тақта.* Жобаны жасап бастаған кезде әкімшілік тақта автоматты түрде құрылады. Бұл әзірлеушіні қолмен әкімші жасау қажеттілігінен босатады. Әкімшілік қосымша барлық жасалған әрекеттерді протоколдай отырып, қажетті кез келген нысандарды жасауға, өзгертуге және жоюға мүмкіндік береді және пайдаланушылар мен топтарды басқару үшін интерфейсті ұсынады;

2) *Даму жылдамдығы.* Django 2005 жылы ұсынылды. 14 жыл ішінде әрдайым өзертіліп, дамып және жаңарып отырды. Әдетте кез келген фреймворкта жаңа мүмкіндіктер пайда болып, ал ескі нұсқалары жетілдіріліп, өзгертіліп отырады. Бастысы, Django-ны меңгеру барысында сұрақтар туындап, жауап іздеген жағдайда бұл аса қатты қиындық тудырмайды. Себебі мыңдаған мамандар осы кезге дейін туындаған мәселелерді шешіп, ғаламтор желісінде өз тәжірибесімен бөліскен;

3) *ORM.* Django-да деректер қорымен қосымшаның өзара әрекеттесуін қамтамасыз ететін объектілі-реляциялық бейнелеу (ORM) жүзеге асырылған. Ол деректерді PostgreeSQL немесе MySQL, SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere және Oracle секілді деректер қорынан бағдарлама кодында қолданылатын объектілерге автоматты түрде жібереді;

4) *Қауіпсіздік***.** Фреймворк қауіпсіздіктің жоғары деңгейін қамтамасыз етеді. Қорғаныс әдістері әдетті жағдайға енгізілген.

Жобаны бастау үшін алдымен ортада керекті бағдарламалар орнатылуы қажет. Сондықтан серверлік бөлігін бастау үшін Python және оны басқару барысында орнату, жаңарту және жою секілді амалдар орындауды қамтамасыз ететін pip құралы керек болады.

Ресми сайттан бағдарлама әзірленетін ортадағы операциялық жүйеге сәйкес Python бағдарламалау тілі жүктелінеді. Процесс барысында pip пакеті Python-ның соңғы жаңа нұсқаларынан бастап автоматты түрде бірге орнатылады.

Бір себептермен орнатылмай қалған жағдай болса пайдаланушы еш мәселесіз қолмен жүктей алады. Орнатылған құрал PATH-да тіркелінеді. Консольда келесі командалар жүзеге асырылады:

– *pip list* – орнатылған пакеттер тізімін көрсету;

– *pip3 install virtualenwrapper-win* – Django виртуалды ортасын орнату;

– *mkvirtualenv* – виртуалды ортаны құру;

– *pip3 install django* – фреймворкті орнату;

– *django-admin startproject name* – жаңа жоба ашу, *name* сөзінің орнына жоба аты жазылады;

– *python manage.py runserver* – localhost:8000 адресінде жобаны іске қосу.

2.4 Python бағдарламалау тілі

Python – кең қолданылатын, жоғары деңгейдегі әмбебап бағдарламалау тілі. Оны 1989 жылдың соңында Гвидо Ван Россум жасап, ұсынды [8]. Бағдарламашылар тарапынан көп сұранысқа ие болған бұл жаңа интерпретациялық бағдарламалау тілі өте тез танылды. Аталған көрсеткішке жаһандық жобаларды жүзеге асыру үшін Python тілін пайдаланатын Google, Microsoft, Facebook, Yandex, YouTube, Wargaming, Instagram, Mozilla секілді танымал компаниялар дәлел бола алады. Және Windows, Mac OS X, Linux, Unix секілді барлық дерлік операциялық жүйелерде жұмыс жасайтын болғандықтан кроссплатформалы бағдарламалау тілі болып саналады.

Әрдайым қолданыста болып, қарқынды дамуының арқасында жылдан жылға қолданылу аймағы кеңейе түсті. Оның көмегімен интернет-сайттар, ойындар мен десктопты қосымшалардан роботтар мен жүйелік сервистерге дейінгі диапазонда кез келген қосымшаларды жасауға болады. Әмбебап бағдарламалау тілі деп атайтыны да сондықтан.

Қазіргі таңда ірі корпорациялар өз жұмыстарын Python бағдарламалау тілінде әзірлеуге көшуде. Себебі технология әлі де ұзақ уақыт қолданыста болады және жұмыс барысында мамандарды іздестіруде еш мәселе туындамайды.

2.4.1 Басқа тілдермен салыстырғандағы артықшылықтары

Python басқа бағдарламалау тілдерімен оңай бәсекелесе алатындай артықшылықтары жетерлік. Біріншіден, түсінікті және қарапайым тіл. Динамикалық типизация – Python тілінің тағы бір айта кетуге тұрарлық басты артықшылықтарының бірі [8]. Бұл қолданушылар үшін кодты жазуды жеңілдетіп, жұмыс барысында көптеген қателерді болдырмау мүмкіндігін

қамтамасыз етеді. Пайдаланушыларға қолайлы болу мақсатында төмендегідей тағы бірқатар ерекшеліктері жіктелген:

– қолданушыға оқып меңгергенге жеңіл;

– жылдам прототиптеу және динамикалық семантика құралдары;

– жаңа қолданушыларға көмек көрсету негізінде өзіндік қоғамы қалыптасқан;

– жоба жасау кезінде көптеген пайдалы кітапханаларын оңай қолдануға болады;

– модульдік механизмдер жақсы ойластырылған және оңай пайдаланылуы жеңіл;

– басқа кейбір тілдердегі секілді қиындық тудырып жататын операторлық жақшалар жоқ.

Алайда, ірі ауқымды жобалар жасау кезінде бағдарламаларды орындау жылдамдығы – бағдарламалау тілінің осал тұсы. Бұл тұста Python Java, C, C++ сияқты тілдерге жол береді [8].

Сонымен, Python – бүгінгі таңда сұранысқа ие және болашаққа үлкен әлеуеті бар бағдарламалау тілі.

2.5 Деректер қоры

Деректер қоры (ДҚ) – кестелік құрылыммен дайындалған деректердің аса үлкен жиынтығы. Құру барысында Дерек қорын басқару жүйесі (ДҚБЖ) қолданылады. ДҚБЖ – ол деректер қорындағы жазбаларды құру, өңдеу және шығару үшін қажетті бағдарламалық қамтамасыз ету құралы [9,10].

ДҚБЖ жұмыс жасағанда бірнеше тізбектелген кезеңдерді бөліп көрсетеді:

– деректер қорын жобалау;

– құрылымын құру;

– мәліметтермен толтыру, тексеру және редактрлеу;

– қажет жазбаларды іздеу;

– ақпаратты іріктеу;

– есеп беруді дайындау.

Шын мәнінде, деректер қоры – абстрактілі ұғым, кесте – тек ақпаратты сақтау тәсілі, кестелер жиынтығы бір-бірімен логикалық байланыста болады және жиынтықты деректер қоры дейді [9,10]. Ал ДҚБЖ көмегімен сақталған мәліметтер басқарылады. Ол 2 топқа жіктеледі:

1) реляциялық – деректер жолдар мен бағандардан тұратын кесте түрінде қатаң жүйеде жазылады;

2) реляциялық емес – еркін жүйеде жазу, яғни әдеттегі жүйеге қарағанда жолдар мен бағандардың кестелік схемасы қолданылмайды. Жүйеде сақталатын деректер түрінің нақты талаптарына оңтайландырылған сақтау моделі қолданылады.

Жобаның деректер қорын жүзеге асыру үшін реляциялық модель таңдалынды. Оның негізгі ерекшелігі өз ішіндегі объектілер екі өлшемді кестелердің жиынтығы түрінде сақталатыны болып табылады. Әрбір кестеде жеке, алдын ала анықталған атаулы өрістер жиынтығы бар. Реляциялық база кестелерінің бағандарында сан, жол немесе күн сияқты тіркелген типтегі скалярлық деректер болуы мүмкін.

Танымал реляциялық деректер қорын басқаруға негізделген танымал ДҚБЖ- лер жетерлік. Олар: MySQL, MSSQL, Oracle, PostgreSQL және т.б.

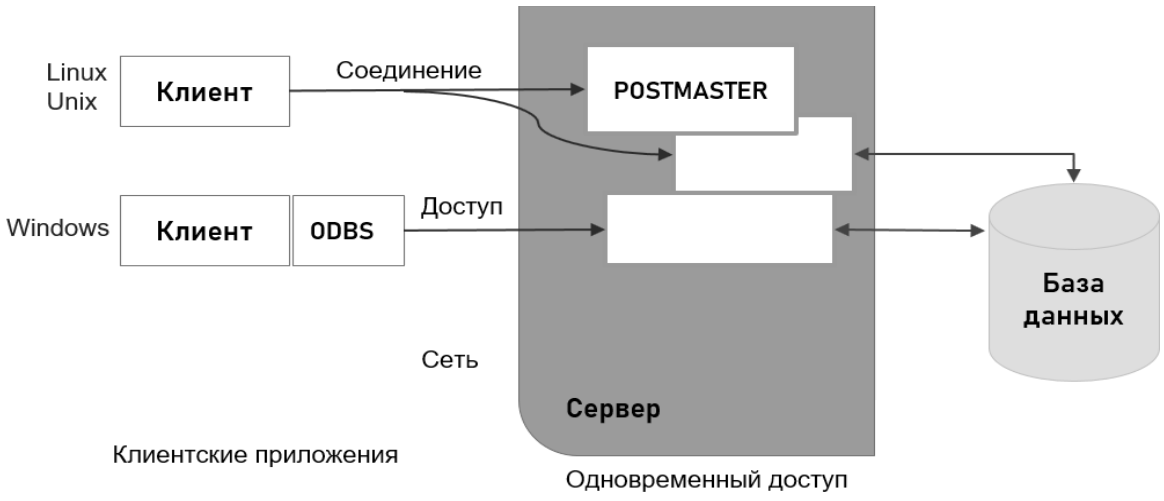
2.6 PostgreSQL сервері

PostgreSQL – бұл тұрақты SQL сұраныстарының стандартты тілін қолдайтын және деректер қоры үшін реляциялық модельді пайдаланатын деректер қорын басқару жүйесі [10,11].

Сервер көптеген түрлі мүмкіндіктерді ұсынады, сенімді және өнімділігі бойынша жақсы сипаттамалары көп. Ол барлық дерлік UNIX-платформаларда, тіптен FreeBSD және Linux жүйелерінде де жұмыс істейді [10]. Басқа серверлерде бар барлық мүмкіндіктерге, сондай-ақ кейбір қосымшаларға ие.

PostgreSQL-дың әрбір нұсқасы өте мұқият тексеріледі, бета нұсқасы кемінде айлық тестілеуден өтеді. Көптеген пайдаланушылар қоғамдастығының және кодқа ашық кіру арқылы қателер өте тез түзетіледі.

PostgreSQL өзіндік архитектурасымен ерекшеленеді. Оны төмендегі 2.1-суреттен байқауға болады.



2.4-сурет – PostgreSQL архитектурасының көрінісі

Суретте көрсетілгендей, бірнеше клиент серверге желі арқылы қосылады. PostgreSQL TCP/IP протоколымен жұмыс жасайды, бұл жергілікті желі немесе

ғаламтор болуы мүмкін [10]. Ең алдымен, әрбір клиент деректер қорының негізгі серверлік процесімен (көріністе-Postmaster) байланысады.

Postmaster клиенттің сұранысына қызмет көрсету үшін құрылатын жаңа серверлік процесс. Бұл серверге бір мезгілде көп қолданушылар қол жеткізгенде де, PostgreSQL мәліметтердің бүтіндігін ұстап тұруға мүмкіндік береді.

Клиенттік қосымшалар деректер қорымен PostgreSQL арнайы хаттамасы бойынша байланысады. Клиент жағында қажетті қосымшамен жұмыс істеу қолданушыға ыңғайлы, түсінікті болуы үшін стандартты интерфейсті ұсынатын бағдарламалық жасақтаманы орнатуға болады. Мысалы, ODBC немесе JDBC стандарты бойынша. ODBC драйверінің қол жетімділігі PostgreSQL-ді қор ретінде пайдалануға мүмкіндік береді.

2.6.1 PostgreSQL немесе MySQL

Деректер қоры құрылымдық сақтауға және әр түрлі деректерге жылдам қол жеткізуге арналған. Әрбір деректер қорының мәліметтерден басқа, деректерді өңдеу барысында орындалатын жұмыстың белгілі бір моделі болуы тиіс.

Бұл бөлімшеде неге PostgreSQL ДҚБЖ таңдалынғандығы, MySQL-мен салыстырма және өзіндік артықшылықтары сипатталатын болады. Олар жұмыстың дұрыстығы, жұмыстың тұрақтылығы, өнімділікке тікелей байланысты.

MySQL – танымал және ірі серверлік ДҚ [12]. Қолданылуы оңай, қолдаушылар қоғамы үлкен және ол туралы желіде ақпараттар көп, барлық дерлік мәселелерге шешім табылған. MySQL SQL стандарттарына толығымен сәйкес келмейді, алайда кең функционал тізімін ұсынады. Өзіндік артықшылықтары:

Қарапайымдылығы. MySQL оңай орнатылады. ДҚ-мен жұмысты бастауды жеңілдететін, визуалды құралдарды қоса алғанда, көптеген қосымша құралдар бар.

1) *Көптеген функциялар.* SQL функционалының көп бөлігін қолдайды;
2) *Қауіпсіздік.* Қауіпсіздік функциялары да жетерлік;
3) *Масштабталу.* MySQL шын мәнінде үлкен деректер көлемімен жұмыс істей алады және масштабталатын қосымшалар үшін жақсы;
4) *Жылдамдық.* Кейбір стандарттарты қолдамауы ДҚБЖ-ның өнімділігін арттырады.

Дегенмен, кемшіліктері де кездеседі:

1) *Белгілі шектеулер*. MySQL кейбір қызметтерді атқара алмайды және белгілі бір функционалдылық шектеулері бар;
2) *Сенімділік мәселелері.* Кейбір операциялары басқа РДҚБЖ-мен салыстырғанда сенімділігі төмен;
3) *Әзірлеудегі қиындық.* Open-source өнімі болса да, онымен жұмыс тежелген, бұл өз кезегінде, ауқымды жобаларда қиындық тудырады.

PostgreSQL – ANSI/ISO SQL стандарттарына толық сәйкес келетін және масштабталуға икемді ең озық РДҚБЖ-лерінің бірі [12]. Объектілі-бағытталған функционалға ие, соның ішінде ACID (Atomicity, Consistency, Isolation, Durability) тұжырымдамасының толық қолдауына ие.

MVC (Multiversion Concurrency Control) жүйесіне негізделіп бірнеше тапсырмаларды бір мезгілде өңдеу жақсы дамытылған, бұл сондай-ақ ACID-пен үйлесімділікті қамтамасыз етеді [12].

PostgreSQL MySQL сияқты танымал болмағанымен, жұмысын жеңілдетуге мүмкіндік беретін көптеген қосымша құралдар мен кітапханалар ұсынады.

Басты артықшылықтары:

1) Толық SQL-үйлесімділік;

2) *Қауымдастық.* PostgreSQL 24/7 тәжірибелі бағдарламашылармен қолдау көрсетіледі;

3) *Танымал ұйымдардың қолдауы.* Өзіндік озық функцияларына қарамастан, өзге ұйымдар қолданысқа көптеген құралдар ұсынады;

4) *Кеңейтілуі.* Сақтау процедуралар есебінен бағдарламалық кеңейтілу оңай жүзеге асырылады;

5) *Объектілік-бағдар.* Тек реляциялық емес, сонымен қатар Объектілік-бағытталған ДҚБЖ.

Өз кезегінде төмендегідей кемшіліктері бар:

1) *Өнімділік*. қарапайым оқу операцияларында қолайсыз;

2) *Танымалдығы.* Күрделілігіне байланысты құрал өте танымал емес;

3) *Хостинг.* Жоғарыда аталған факторларға байланысты қолайлы провайдерді табу қиын.

3 Жобалау бөлімі

3.1 «Electronic Timesheet» жобасы туралы

«Electronic Timesheet» жобасы маңызды құжаттармен тікелей байланысты болғандықтан веб-қосымша негізінде жасалынды.

Қосымша оқу орнымыздың бірқатар қызметкерлерінің жұмыстарын жеңілдетуге бағытталған. Бағдарламада орындалатын негізгі жұмыс тәртібі төмендегідей:

1) бағдарламашы белгілі бір жауапты тұлғаларды лауазымдары бойынша деректер қорына енгізіп, жеке аккаунт ашып береді. Лауазым түрлері негізгі 2 топқа жіктеледі: табель құрастырушы және табель қабылдаушы;

2) табель құрастырушы акт, жұмыс табелі және демалыс табелі деп аталатын бөлімдерге жауапты. Акт – жұмысшы қосымша қызмет атқарғанда, жұмыс табелі – тұрақты, ал демалыс табелі – демалыс немесе мейрам күндері жұмыс жасаған қызметкерлер үшін толтырылады. Толтыру барысында электронды қол қоюы қажет болатын табель қабылдаушыларды таңдап, оларға жіберуі қажет;

3) әрбір қабылдаушы тұлға өз аккаунтына кіре отырып, келген жұмысты жете тексеріп, растау жасау арқылы электронды қол қояды, қате тапқан жағдайда, өзінің пікірін жазып жібере алады;

4) осылайша құжат бухгалтер қызметкерлеріне жетіп, сол арқылы барлық жұмысшылардың еңбекақысын өз қосымшалары арқылы есептей алады.

Жүйе негізгі екі модульдан тұрады:

– auth – авторизация парақшасының құрастырылуы;

– home – жобаның қалған барлық бөлімдерінің құрастырылуы жүзеге асырылған.

Жоба екі адамдық командамен жасалынды. Бұл мәтінде авторизация парақшасы және жоғарыда таныстырылған табель қабылдаушының парақшасы мен оның мүмкіндіктері баяндалатын болады.

3.2 Авторизация парақшасы

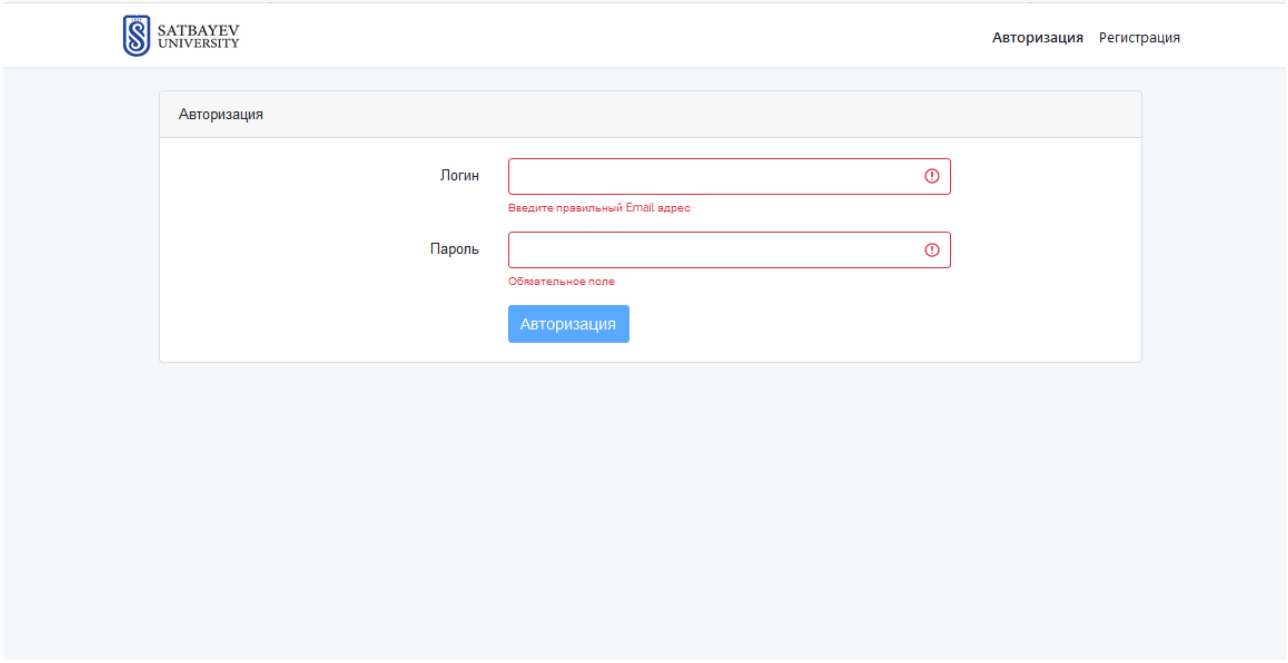
Жүйені іске қосқанда ашылатын авторизация парақшасы жобаның барлық қолданушыларына арналып жасалынды.

Парақша логин мен құпиясөз жазатын қорапшадан және «Авторизация» батырмасынан тұрады. Аталған батырманы басып ары қарай өтуі үшін арнайы қатаң ережелер құрастырылды:

– қолданушы арнайы ДИС меңгерушілерінің құрастырып берген аккаунты арқылы ғана кіре алады;

– терілген логин қате болған жағдайда «Введите правильный E-mail адрес», ал құпиясөзде «Неправильный пароль» ескертпесі көрсетіледі. Деректер қорында тіркелмеген аккаунт болса «Вы еще не авторизованы» арқылы ескертіледі;

– жоғарыда көрсетілген заңдылықтарды дұрыс орындап, қолданушы жүйенің келесі парақшаларына өте алады.



3.1-сурет – Авторизация бетінің көрінісі

3.3 Табель қабылдаушының парақшасы

Оқу орнымыздың қызметкерлерінің жұмысын жеңілдету негізінде жасалынған бағдарламада табель қабылдаушының өзіндік рөлі бар және өз ішінде жұмыс лауазымдары бойынша бөлінеді. Олар:

– ДИС меңгерушілері(жүйеге кіретін қолданушыларға аккаунт құрастырады);

– департамент басшылары;

– әр кафедра меңгерушілері;

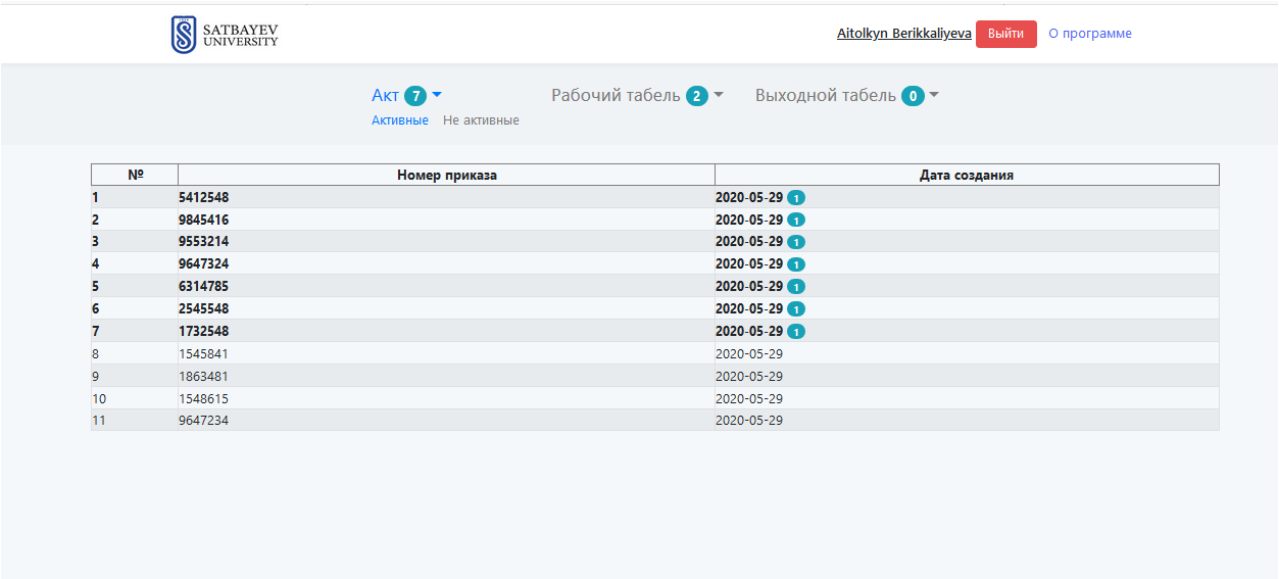
– кадр бөлімінің меңгерушілері;

– топтық жұмысшылардың (электрик,сантехник және т.б) комендanттары және басшылары;

– бухгалтер қызметкерлері.

Аталған қызметкерлердің негізгі міндеті: электронды түрде қабылдап алған табель немесе актті мұқият тексеріп растау жасау. Ал егер тексеру

барысында жұмыстан қате байқаған жағдайда пікірін жазып, кері қайтара алады. Табель қабылдаушының парақшасының көрінісі 3.1-суретте көрсетілген.



3.1-сурет – Негізгі бет көрінісі

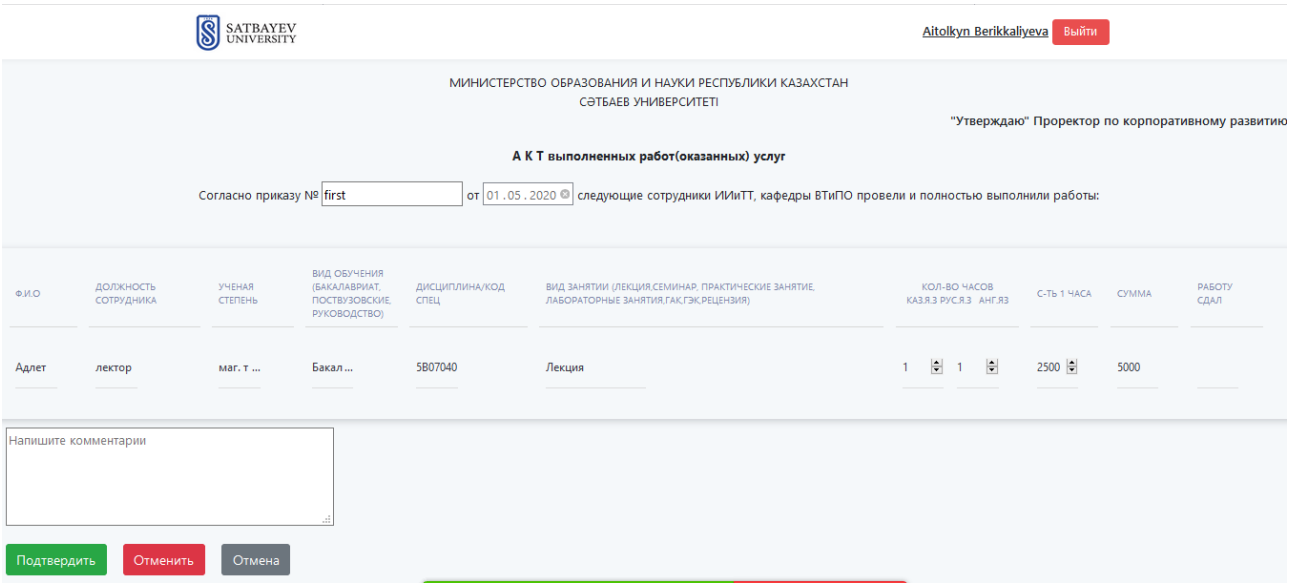
Парақша басты мәзір мен кестеден тұратын негізгі бетті құрайды. Басты мәзірде акт және табельдің екі түрі таңдалынады Әрбір мәзірдің ішіндегі кестеде барлық жаңадан келген және бұрынғы акт пен табельдердің тізімі болады. Тізім құжаттың реттік саны, бұйрық номері және құрылған күнін қамтиды.

Жаңа құжат келгенде арнайы жаңа хабарлама келгендігін көрсететін сан арқылы белгі пайда болады.

Табель қабылдаушы тарапынан расталған құжат «Не активные» тізіміне өтеді. «Активные» тізімі өз ішінде екі түрге жіктеледі: еш амал орындалмаған болса құжат майлы(жирный) сипатта қала береді, расталмаған жағдайда тек майлы сипатынан алынып, осы тізімде қалады. Осы тұста табель қабылдаушыға расталмаған құжат хабарлама ретінде барады, демек өзгертіп қайта жіберуі қажет.

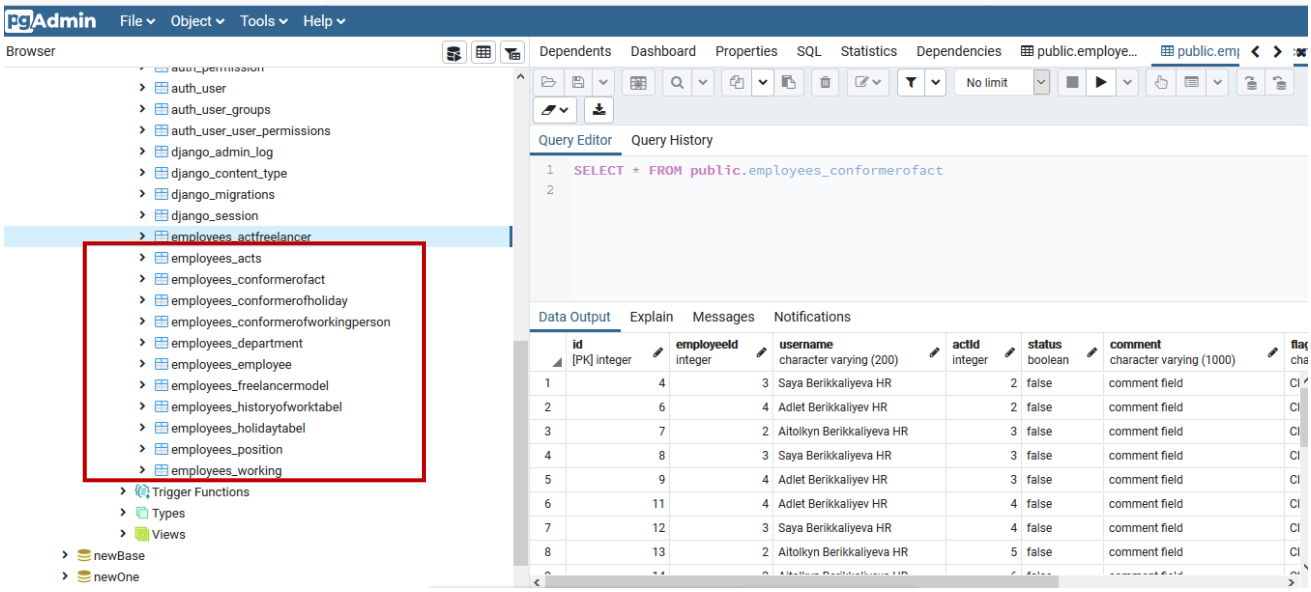
Табель қабылдаушы құжатты растауы үшін «Подтвердить» батырмасын, ал кері қайтаруы үшін «Отменить» батырмасын басуы қажет. Жоғарыда электронды қол қоюдың маңызы жете түсіндірілген бойынша, іс-әрекет растау бетінде жүзеге асырылады. Төменде 3.2-суретте растау беті көрсетілген.

Парақшалардың ыңғайлы және қарапайым жасалынды. Кез келген бағдарлама тез қолданысқа ие болуы үшін осындай сипаттамаларға ие болуы маңызды.



3.2-сурет – Растау бетінің көрінісі

Жобаның табель қабылдаушы және авторизация бөлімдеріне арналған деректер қорының арнайы кестелері 3.3-суретте қызыл белгі арқылы көрсетілген.



3.3-сурет – Деректер қорындағы көрініс

ҚОРЫТЫНДЫ

Дипломдық жобаны қорытындылай келсек, Қ.И.Сәтбаев атындағы университет қызметкерлерінің жұмысын біршама жақсарту мақсатымен «Electronic Timesheet» веб-қосымшасы жасалынды. Құрылымы күрделі Angular және Django фреймворктарымен жасалынғандықтан, жұмыс барысында көптеген ерекше тәсілдер қолданылды. Бұл тәсілдер қосымшаның қарапайым, қолданысқа ыңғайлы және түсінікті болуына көмегін тигізді.

Бағдарламаның негізгі ерекшілігі – заманауи мүмкіндігі. Атап айтқанда, электронды қол қою. Қосымша арқасында осы уақытқа дейін қағаз жүзінде іске асырылып келе жатқан табель көп функциясы автоматты түрде жасалынған жаңашылдыққа ие болды. Маңызды құжаттармен тікелей байланысты болғандықтан үлкен жауапкершілікті талап ететіні сөзсіз. Осы жауапкершілікті сезіне отырып, әрбір бөліміне жеке мән беріліп жасалынды.

Веб-қосымша университетіміздің барлық қызметкерлерінің еңбекақысын есептеу барысында қажет болатын жұмыс уақытысын есептеп көрсетіп, осы істерге жауапты тұлғаларға жіберіп, электронды қол қойдыру үшін арналған.

Angular және Django фреймворктарын мұқият зерделегеннен кейін, бұл атаулардың күннен-күнге функционалдық жағынан жоғары шыңдарды бағындыратынын сенімділікпен айтуға болады. Көптеген маңызды және күрделі қосымшалар осы фреймворктар көмегімен жасалуда. Жұмыс барысы үлкен еңбекті және мұқияттылықты талап ететіні де сондықтан.

Жоба қолданылу аясы шексіз, тегін және тиімді веб-қосымша ретінде жасалынды.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1 Web-приложения – преимущества и недостатки // Сайттың электронды нұсқасы https://mydiv.net/arts/view-web-prilozhenija_preimuxhestva_i_nedostatki

2 UML // Сайттың электронды нұсқасы https://ru.wikipedia.org/wiki/UML#UML_1.x

3 UML — диаграмма вариантов использования (use case diagram) // Сайттың электронды нұсқасы https://habr.com/ru/post/47940/

4 Фреймворки в веб-разработке // Сайттың электронды нұсқасы https://web-creator.ru/articles/about_frameworks

5 Angular Course // Сайттың электронды нұсқасы https://www.ngdevelop.Tech /angular/architecture/

6 Дементий.Д Почему Django — лучший фреймворк для разработки сайтов // Сайттың электронды нұсқасы https://turbo/pochemu-django-luchshiy-freymvork-dlya-razrabotki-saytov

7 Node.js против Django: Что лучше JavaScript или Python? // Сайттың электронды нұсқасы https://infozone.pro/nodejs-vs-django-javascript-better-than-python/

8 Язык программирования Python: особенности и преимущества // Сайттың электронды нұсқасы https://ipap.ru/poleznye-stati/4-useful/yazyk-programmirovaniya-python-osobennosti-i-preimushchestva

9 Базы данных // Сайттың электронды нұсқасы https://zametkinapolyah.ru/zametki-o-mysql/bazy-dannyx-vidy-i-tipy-baz-dannyx-struktura-relyacionnyx-baz-dannyx-proektirovanie-baz-dannyx-setevye-i-ierarxicheskie-bazy-dannyx.html

10 Моргунов Е.П. PostgreSQL. Основы языка SQL – СПб.: БХВ-Петербург, 2018. — 336 б.

11 Что такое PostgreSQL? Плюсы и минусы бесплатной базы данных // Сайттың электронды нұсқасы https://oracle-patches.com/common/postgresql

12MySQL и PostgreSQL: сравниваем популярные реляционные СУБД // Сайттың электронды нұсқасы https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/

А Қосымшасы
(міндетті)

Техникалық тапсырма

А.1 «Electronic Timesheet» веб-қосымшасын әзірлеуге арналған техникалық тапсырма

Бұл техникалық тапсырма барлық қызметкерлердің жалақысын есептеу барысында қажет болатын олардың жұмыс уақытысы мен осы істерге жауапты тұлғалардың электронды қол қоюы үшін жасалынған автоматтандырылған жүйенің әзірленуіне арналған. Зерттеу болжамы бойынша әр кафедра меңгерушісі және қызметкері, жұмысшы топтардың комендanttары және бухгалтер қызметкерлері қолдана алады. Қолмен толтырылып жүрген құжаттардың көп бөлігі электронды табель жүйесінде автоматты түрде орындалатындықтан қателіктердің азайып, қағазбастылықтың жойылуына және оқу орнымыздың осы саладағы қызметкерлерінің жұмыс барысының жеңілдеуіне мүмкіндік береді.

А.2 Функционалдық сипаттама

Жүйені қолданушы екі топтың функционалдық сипаттамалары өзара жіктеледі.

Табель құрастырушының мүмкіндіктері:

– авторизация;

– жаңа құжат құру;

– құжатты растаушыларға жіберу;

– расталмаған жағдайда, түзетіп және қайта жіберу;

– барлық жауапты тұлғамен расталған болса, «не активные» тізіміне өтеді;

– егер бір жауапты тұлға растамаған жағдайда, майлы (жирный) сипатта өзіне келіп түседі;

– құжатты жоя алады;

– әрбір құжаттың «История» арнайы терезесін көре алады.

Табель қабылдаушының мүмкіндіктері:

– авторизация;

– жаңа құжатты майлы сипатта қабылдай алады;

– құжатты растап немесе қателік болған жағдайда өз пікірін жазып кері қайтара алады;

Расталған құжат арнайы «не активные» тізіміне өтеді. Расталмаған құжат майлы сипаты өзгеріп, «активные» тізімінде қала береді.

А.3 Сенімділік бойынша талаптар

Жобада қарастырылған сенімділікке қатысты талаптар:

– тек оқу орнымыздың қызметкерлері қолдана алады;

– қолданушы деректер қорына тіркелмеген аккаунтпен жүйеге кіре алмайды;

– қолданушылар екі топқа жіктеледі, авторизация жасалған соң өздеріне қатысты парақшалары ғана ашылады;

– табель құрастырушы жаңа құжатты толтыру барысында құжаттың атын, қажет мәліметтерін және растаушы тұлғаларды белгілеуге міндетті;

– табель қабылдаушы акт немесе табельді растамай кері қайтаратын жағдайда міндетті түрде себебін жазуы қажет.

А.4 Қосымша құжаттамасына анықтама

Әрбір құжат белгілі бір арнайы тәртіппен толтырылуы қажет. Жүйені электронды нұсқаға көшіру барысында өзіндік стандарттары сақталынып жасалынды.

А.5 Адаптация бойынша талаптар

Қосымша операциялық жүйе мен браузер таңдамайды. Барлық Windows, Linux, Mac OS секілді платформаларда және Mozilla Firefox, Google Chrome, Яндекс пен Opera секілді браузерлерде жұмыс істей алады.

А.6 Программалық интерфейстер

Жобаның әзірленуіне қатысты қажет бағдарламалық компоненттер:

– node.js және npm менеджерлік пакеттері;

– angular пакеті;

– python бағдарламалау тілі және pip құралы;

– django пакеті;

– postgreSQL және PgAdmin бағдарламасы.

А.7 Коммуникациялық интерфейстер

Сервер мен клиенттерді байланыстыратын TCP/IP протоколы қолданылады. Ол PostgreSQL ДҚБЖ-сі негізінде іске асырылады. Процесс кезінде пайдаланушының ғаламтор желісіне қосылу мүмкіндігі болуы қажет.

А.8 Аппараттық интерфейстер

Қолданушы компьютеріне қойылатын жалпы талаптар:
– оперативті жады – 512 MB;
– диск тұлғалы кеңістік – ~ 52 MB + қолданушылар файлдарын сақтауға қажет орын;
– қатты диск 40 Gb; 1024×768 рұқсатта жұмыс қолдайтын видеокарта;
– процессор – P4 және жоғары, Celeron;
– пернетақта, манипулятор тышқан.

А.9 Анықтамалар, терминдер және қысқартулар

Терминдер, қысқартулар және олардың анықтамалары А.1-кестеде көрсетілген.

А.1-кесте – Анықтамалар, терминдер және қысқартулар

Терминдер және қысқартулар	Анықтамалар
Жоба	Белгілі бір уақыт ішінде нақты қойылған мақсаттарға қол жеткізуге негізделген бір реттік, қайталанбайтын қызмет немесе іс-әрекеттер жиынтығы.
TypeScript	Веб-қосымшаларды әзірлеу, JavaScript мүмкіндіктерін кеңейте түскен бағдарламалау тілі.
Менеджерлік пакет	Бағдарламаның әр түрлі компоненттерін орнату,баптау және жаңарту процесін басқаруға мүмкіндік беретін бағдарламалық жасақтама жиынтығы.

А Қосымшасының жалғасы

Командалар жолы(консоль)	Адам мен компьютер арасындағы мәтіндік интерфейстің түрлері, онда компьютерге нұсқаулар негізінен пернетақтадан мәтіндік жолдарды енгізу арқылы беріледі.
Localhost	Компьютерлік желілерде стандартты, жеке IP-мекен-жайлары үшін ресми резервтелген домендік атау.
Сервер	Адамның қатысуынсыз сервистік бағдарламалық қамтамасыз етуді орындауға арналған арнайы жабдық (қызметтік компьютер немесе жұмыс станциясы). Қызметтік жабдықтар немесе бағдарламалық қамтамасыз ету мағынасын береді.
TCP/IP	Сандық түрде ұсынылған деректерді жіберудің желілік моделі. Ақпарат көзінен қолданушыға деректерді беру тәсілін сипаттайды.
SaaS	Software as a service. Әзірлеуші веб-қосымшаны жасап, оны тапсырыс берушіге интернет арқылы бағдарламалық қамтамасыз етуге қол жеткізуді ұсынуына мүмкіндігі бар және дербес басқаратын бағдарламалық қамтамасыз етуді сату және пайдаланудың бизнес-моделі.
CMS	Content Management System. Пайдаланушыларға әзірлеушінің қатысуынсыз, сайт құрамындағы дайын ақпаратты орналастыруға немесе өзгертуге мүмкіндік беретін бағдарламалық қамтамасыз ету.
CMF	Content Management Framework. Мазмұнды басқару жүйелерін жобалау үшін арналған каркас.

Б Қосымшасы
(міндетті)

Бағдарламаның мәтіні

1. Models of database

```
class Employee(models.Model):
    firstName = models.CharField(max_length = 200, blank = False)
    lastName = models.CharField(max_length = 200, blank = False)
    middleName = models.CharField(max_length = 200, blank = False)
    speciality = models.CharField(max_length = 200, blank = False)#tester, teacher, director, dean
    rate = models.FloatField(blank = False)
    postionId = models.IntegerField(blank = False)
    password = models.CharField(max_length = 200, blank = False)
    email = models.CharField(max_length = 200, blank = False)
    def __str__(self):
        return self.password
class Position(models.Model):
    name = models.CharField(max_length = 200, blank = False)
    departmentID = models.IntegerField(blank = False)
class Department(models.Model):
    name = models.CharField(max_length = 200, blank = False)
class ConformerOfWorkingPerson(models.Model):
    employeeId = models.IntegerField(blank = False)
    username = models.CharField(max_length = 200, blank = False)
    workingTabelId = models.IntegerField(blank = False)
    status = models.BooleanField(blank = False)
    comment = models.CharField(max_length = 1000, blank = True)
    flag = models.CharField(max_length = 200, blank = False)
class ConformerOfHoliday(models.Model):
    employeeId = models.IntegerField(blank = False)
    username = models.CharField(max_length = 200, blank = False)
    workingTabelId = models.IntegerField(blank = False)
    status = models.BooleanField(blank = False)
    comment = models.CharField(max_length = 1000, blank = True)
    flag = models.CharField(max_length = 200, blank = False)
class ConformerOfAct(models.Model):
    employeeId = models.IntegerField(blank = False)
    username = models.CharField(max_length = 200, blank = False)
    actId = models.IntegerField(blank = False)
    status = models.BooleanField(blank = False)
    comment = models.CharField(max_length = 1000, blank = True)
```

2. Url addresses

```
path('login/', views.login),
path('actList/<int:id>/<str:type>', views.ListOfActs),
```

```
    path('worktabelList/<int:id>/<str:type>', views.ListOfWorkTabel),
    path('holidaytabelList/<int:id>/<str:type>', views.ListOfHolidayTabel),
    path('editActConformer/<int:id>/<int:actId>',views.conformerActEdit),
    path('editWorkingTabelConformers/<int:id>/<int:workingTabelId>', views.conformerWorkingTa
belEdit),
    path('editHolidayTabel/<int:id>/<int:holidayTabelId>', views.conformerHolidayTabelEdit),
    path('worktabelCheck/<int:id>', views.WorkTabelStatus),
    path('holidaytabelCheck/<int:id>', views.HolidayTabelStatus),
    path('actCheck/<int:id>', views.ActStatus)
```

3. Auth.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { LoginComponent } from './page/login/login.component';
import { AuthRoutingModule } from './auth.routing';
import { RegisterComponent } from './page/register/register.component';
import { AuthComponent } from './page/auth/auth.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
@NgModule({
  declarations: [LoginComponent, RegisterComponent, AuthComponent],
  imports: [
    CommonModule,
    AuthRoutingModule,
    FormsModule,
    ReactiveFormsModule,
  ]})
export class AuthModule { }
```

4. Auth.routing.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { LoginComponent } from './page/login/login.component';
import { RegisterComponent } from './page/register/register.component';
import { AuthComponent } from './page/auth/auth.component';
const routes: Routes = [
  {
    path: '',
    redirectTo: 'auth',
    pathMatch: 'full'
  },
  {
    path: '',
    children: [
      {
        path: 'auth',
        component: AuthComponent
      },
```

```
    {
      path: 'login',
      component: LoginComponent
    },
    {
      path: 'register',
      component: RegisterComponent
    }
   ]
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class AuthRoutingModule { }
```

5. Profile-list.component.ts

```
<nav class="navbar navbar-expand-lg navbar-light navbar-laravel">
  <div class="container">
     <a class="navbar-brand" href="#">Satbayev University</a>
     <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
     </button>
     <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav ml-auto">
           <li class="nav-item">
              <span class="user-text">{{user}}</span>
           </li>
           <li class="nav-item exit-btn">
              <a [class.blue-link] = "isRegister" (click)="exit()">Выйти</a>
           </li>
        </ul>
     </div>
  </div>
</nav>
<nav class="navbar navbar-light bg-light justify-content-between">
<div class="container">
   <ul class="menu">
    <li [class.active] ="isAct"  (click)="onMenuChange('act')" class="nav-item dropdown">
      <a class=" dropdown-toggle" id="navbarDropdownMenuLink">Акт</a>
       <ng-container  *ngIf="isAct">
        <div style="color: #7c7c7c; display:block; font-size: 15px;">
         <a [class.active] = "isActOneActive" (click)="onAct('active')">Активные</a>
<a [class.active] = "isActOneHistory" (click)="onAct('history')">Не активные</a>
```

```
      </div>
    </ng-container>
  </li>
  <li [class.active] = "isWork" (click)="onMenuChange('work')" class="nav-item dropdown">
    <a class=" dropdown-toggle" id="navbarDropdownMenuLink">Рабочий табель</a>
    <ng-container  *ngIf="isWork">
      <div style="color: #7c7c7c; display:block; font-size: 15px;">
       <a [class.active] = "isTabelOneActive" (click)="onWorkTabel('active')">Активные</a>
       <a [class.active] = "isTabelOneHistory" (click)="onWorkTabel('history')">Не активные</a>
      </div>
    </ng-container>
  </li>
  <li [class.active] = "isWeekend" (click)="onMenuChange('weekend')" class="nav-
item dropdown">
  <a class=" dropdown-toggle" id="navbarDropdownMenuLink">Выходной табель</a>
   <ng-container  *ngIf="isWeekend">
      <div style="color: #7c7c7c; display:block; font-size: 15px;">
       <a [class.active] = "isTabelTwoActive" (click)="onWeekendTabel('active')">Активные</a>
       <a [class.active] = "isTabelTwoHistory" (click)="onWeekendTabel('history')">Не активные
</a>
      </div>
    </ng-container>
  </li>
 </ul>
</div>
</nav>
<div *ngIf="isWork">
<div style="margin:10px">
 <table class="table table-striped table-bordered" *ngIf="sourceOne">
  <thead>
   <th>№</th>
   <th>Название</th>
   <th>За месяц</th>
   <th>Дата создания</th>
  </thead>
  <tbody>
   <tr *ngFor="let t of sourceOne; let i = index;" class="clickable-
row" (click)="onRowSelected(t.id)">
    <td [className]="t.status === 'CREATED'? 'active': 'onProcess'">{{i+1}}</td>
    <td [className]="t.status  === 'CREATED'? 'active': 'onProcess'">{{t.name}}</td>
    <td [className]="t.status  === 'CREATED'? 'active': 'onProcess'">{{t.month}}</td>
    <td [className]="t.status  === 'CREATED'? 'active': 'onProcess'">{{t.createDate}}</td>
   </tr>
  </tbody>
 </table>
</div>
</div>
<div *ngIf="isWeekend">
<div style="margin:10px">
```

```
  <table class="table table-striped table-bordered" *ngIf="source">
    <thead>
      <th>№</th>
      <th>Название</th>
      <th>За месяц</th>
      <th>Дата создания</th>
    </thead>
    <tbody>
      <tr *ngFor="let t of source; let i = index" class="clickable-row" (click)="onRowSelected(t.id)">
        <td  [className]="t.status === 'CREATED'? 'active': 'onProcess'">{{i+1}}</td>
        <td [className]="t.status === 'CREATED'? 'active': 'onProcess'">{{t.name}}</td>
        <td [className]="t.status === 'CREATED'? 'active': 'onProcess'">{{t.month}}</td>
        <td [className]="t.status === 'CREATED'? 'active': 'onProcess'">{{t.createDate}}</td>
      </tr>
    </tbody>
  </table>
</div>
</div>
<div *ngIf="isAct">
<app-act-
list [data]="data" [conformerList] = "conformerList" [conformerAct] ="conformerAct" [isActOneA
ctive]="isActOneActive" [isActOneHistory]="isActOneHistory"></app-act-list>
</div>
```

6. Profile-list.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { ProfileService } from 'src/app/data/services/profile.service';
import { HomeService } from 'src/app/data/services/home.service';
import { getMonthes } from 'src/app/data/utils/data-generator';
@Component({
  selector: 'app-profile-list',
  templateUrl: './profile-list.component.html',
  styleUrls: ['./profile-list.component.scss']
})
export class ProfileListComponent implements OnInit {
  id: any;
  isRegister: boolean;
  user: any;
  postionId: any;
  isAct = true;
  isTabelOneHistory: boolean;
  isTabelOneActive: boolean;
  isTabelTwoHistory: boolean;
  isTabelTwoActive: boolean;
  isWork: boolean;
  array = [];
  isWeekend: boolean;
```

```
isConformer = true;
isActOneActive = false;
isActOneHistory = false;
monthes = getMonthes();
sourceOne = [];
source = [];
conformerList = [];
workingTabelList = [];
holidayTabelList = [];
conformerAct = true;
data = [];
changedRes = [];
constructor(
  private router: Router,
  private service: ProfileService,
  private homeService: HomeService
) { }
ngOnInit() {
  this.initValues();
  this.onMenuChange("act");
}
initValues() {
  this.isAct = true;
  this.isActOneActive = true;
  this.id = localStorage.getItem('id');
  this.user = localStorage.getItem('name');
  this.postionId = localStorage.getItem('postionId');
}
getMonthValue(value) {
  const v = getMonthes();
  const a = v.find(x => x.value === value).name;
  return a;
}
onMenuChange(type) {
  if (type === 'act') {
    this.isAct = true;
    this.isWork = false;
    this.isWeekend = false;
    this.conformerAct = true;
  } else if (type === 'work') {
    this.isAct = false;
    this.isWork = true;
    this.isWeekend = false;
  } else if (type === 'weekend') {
    this.isAct = false;
    this.isWork = false;
    this.isWeekend = true;
  }
}
```

```
collectingWork(type) {
  this.service.getWorkingTabel(this.id, type).subscribe((res: any) => {
    this.sourceOne = [];
    this.workingTabelList.push(...res);
    for (let el of res) {
      this.homeService.getTabelByID(el.workingTabelId).subscribe((result: any) => {
        this.changedRes = result.map(x=>({
          ...x,
          month: this.getMonthValue(x.month),
          status: el.flag
        })
        );
        this.sourceOne.push(this.changedRes[0]);
      });
    }
  });
}
collectingHoliday(type) {
  this.service.getHolidayTabel(this.id, type).subscribe((res: any) => {
    this.source = [];
    this.holidayTabelList.push(...res);
    for (let el of res) {
      this.homeService.getHolidayTabelByID(el.workingTabelId).subscribe((result: any) => {
        this.changedRes = result.map(x=>({
          ...x,
          month: this.getMonthValue(x.month),
          status: el.flag,
        })
        );
        this.source.push(this.changedRes[0]);
      });
    }
  });
}

collectingDeclinedWork(type) {
  this.service.getWorkingTabel(this.id, type).subscribe((res: any) => {
    this.workingTabelList.push(...res);
    for (let el of res) {
      this.homeService.getTabelByID(el.workingTabelId).subscribe((result: any) => {
        this.changedRes = result.map(x=>({
          ...x,
          month: this.getMonthValue(x.month),
          status: el.flag
        })
        );
        this.sourceOne.push(this.changedRes[0]);
      }); }
    this.sourceOne.reverse();
```

```
  });
}
collectingDeclinedHoliday(type) {
  this.service.getHolidayTabel(this.id, type).subscribe((res: any) => {
    this.holidayTabelList.push(...res);
    for (let el of res) {
      this.homeService.getHolidayTabelByID(el.workingTabelId).subscribe((result: any) => {
        this.changedRes = result.map(x=>({
          ...x,
          month: this.getMonthValue(x.month),
          status: el.flag
        })
        );
        this.source.push(this.changedRes[0]);
      });
    }
    this.source.reverse();
  });
}
onAct(type) {
  this.conformerAct = true;
  if (type === 'active') {
    this.isActOneActive = true;
    this.isActOneHistory = false;
    this.isAct = false;
  } else {
    this.isActOneActive = false;
    this.isActOneHistory = true;
    this.isAct = false;
  }
}
onWorkTabel(type) {
if (type === 'active') {
   this.isTabelOneActive = true;
   this.isTabelOneHistory = false;
   this.collectingWork('CREATED');
   this.collectingDeclinedWork('DECLINED');
} else {
   this.collectingWork('DONE');
   this.isTabelOneActive = false;
   this.isTabelOneHistory = true;
}
}
onWeekendTabel(type) {
 if ( type ===  'active') {
   this.isTabelTwoActive = true;
   this.isTabelTwoHistory = false;
   this.collectingHoliday('CREATED');
   this.collectingDeclinedHoliday('DECLINED');
```

```
  } else {
   this.collectingHoliday('DONE');
   this.isTabelTwoActive = false;
   this.isTabelTwoHistory = true;
  }
 }
 exit() {
  this.router.navigate(['auth']);
  localStorage.removeItem('name');
  localStorage.removeItem('id');
  localStorage.removeItem('postionId');
 }
 onRowSelected(id) {
  if (this.isWeekend) {
    if (this.conformerAct) {
      const flag = this.holidayTabelList.find(x => x.workingTabelId === id).flag;
      this.router.navigate(['holiday'], {queryParams: { conformerWork: true, id, flag}});
    } else {
      this.router.navigate(['holiday'], {queryParams: { conformerWork: true, id}});
    }
  } else if (this.isWork) {
    if (this.conformerAct) {
      const flag = this.workingTabelList.find(x => x.workingTabelId === id).flag;
      this.router.navigate(['working'], {queryParams: { conformerWork: true, id, flag}});
    } else {
      this.router.navigate(['working'], {queryParams: { conformerWork: true, id}});
    }
  }
 }
}
```

Форматы	Зонасы	Поз-сы.	Белгілеулер	Аталулары	Саны	Қорытын ды

					Дипломдық жоба	Бет
Өлш	Бет	Құжат №	қолы	күні		46