ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Автоматика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

ЫБраев Ернұр Ерланұлы

Web-технологияларға кіріспе пәні бойынша электрондық оқулық әзірлеу

ТҮСІНДІРМЕ ЖАЗБА

Дипломдық жобаға

5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету»

Алматы 2022

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ БІЛІМ МИНИСТРЛІГІ

Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Автоматика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
ПИ кафедрасының меңгерушісі
физ.-мат. ғыл.канд, профессор
_____А.Н. Молдагулова
"20" 05 2022 ж.

ТҮСІНДІРМЕ ЖАЗБА
дипломдық жобаға

Тақырыбы: «Web-технологияларға кіріспе пәні бойынша электрондық оқулық әзірлеу»

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

Орындаған Ыбраев Е. Е.

Рецензент
PhD, қауымдастырылған профессор
_____ О.В. Жирнова
"____" _____ 2022 ж.

Ғылыми жетекші
PhD, т.ғ.к., қауымд. профессор
_____ А.Т. Аяпбергенова
"16" мамыр 2022 ж.

Алматы 2022

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ БІЛІМ МИНИСТРЛІГІ

Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Автоматика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

БЕКІТЕМІН
ПИ кафедрасының меңгерушісі
физ.-мат. ғыл.канд., профессор
_____А.Н. Молдагулова
"20" 05 2022 ж.

Дипломдық жобаны орындауға
ТАПСЫРМА

Білім алушыға ЫБраев Ернұр Ерланұлына

Тақырыбы: «Web-технологияларға кіріспе пәні бойынша электрондық оқулық әзірлеу»

Академиялық мәселелер жөніндегі проректор бұйрығының №489-ПВ "24" 12 2021 ж. шешімімен бекітілген.

Орындалған жобаның өткізу мерзімі "24" 05 2022 ж.

Дипломдық жобаның бастапқы мәліметтері: Жобаның төлқұжаты, технология бойынша техникалық құжаттама, техникалық тапсырма, жоба диаграммалары түрінде ақпаратты жинау, деректер қорына сақтау, тестілеу, тексеруге арналған программалық қаматамаларды жасау жүргізілген.

Есеп – түсініктеме жазбаның талқылауға берілген сұрақтардың тізімі:

а) тақырып бойынша талдау және есептің қойылымын жасау;

б) жобаны жобалау және пәндік сала бойынша талдау;

в) пайдаланушы интерфейсін жобалау және дамыту;

г) бағдарламаны құру, кітапханаларды қосу, деректерді қосу және тестілеу;

Графикалық материалдар тізімі (міндетті суреттердің нақты көрсетілуімен):

презентация 20 слайдпен берілген құжат түрінде ұсынылған.

Ұсынылған негізгі әдебиеттер саны: 20

Дипломдық жобаны орындау
КЕСТЕСІ

Бөлімдердің атаулары, зерттелген мәселелердің тізімі	**Ғылыми жетекшіге және кеңесшілерге ұсыну мерзімі**	**Ескерту**
1. Дипломдық жобаның жоспарын құру	14.01.2022	орындалды
2. Тапсырма қойылымы және программалау ортасын таңдау	18.01.2022	орындалды
3. Зерттеу тақырыбы бойынша ғылыми теориялық материалдарды жинау және негізгі бөлім бойынша есеп беру жазбасын дайындау	01.02.2022	орындалды
4. Дипломның екінші және үшінші бөлімдерін, жобалау сызбаларын дайындау	15.02.2022	орындалды
5. Жобаның веб-қосымшасын тестілеуден өткізу	18.03.2022	орындалды
6. Дипломдық жобаға түсіндірме жазба жазуды аяқтау	26.04.2022	орындалды

Дипломдық жұмыс бөлімдерінің кеңесшілерінің аяқталған жұмысқа қойған
қолтаңбалары

Бөлімдер атауы	**Кеңес берушілер (аты-жөні, тегі, ғылыми дәрежесі, атағы)**	**Қолтаңба қойылған мерзімі**	**Қолы**
Нормалық бақылаушы	Жекамбаева М.Н. PhD, қауымдастырылған профессор	20.05.22	
Бағдарламалық бөлім	Марғұлан Қ. тех.ғыл.магистрі, лектор	20.05.22	

Ғылыми жетекші _____ Аяпбергенова А.Т.

Тапсырманы орындауға қабылдап алған студент _____ Ыбраев Е. Е.

Күні «17» 11 2021 ж

АҢДАТПА

Берілген дипломдық жұмыс веб-технологиялардың негізін түсіндіруге, оларды қолдануға үйретуге арналған барлық құрылғылар мен операциялық жүйеге бейімделген веб-қосымша болып табылады. Кез келген қызығушы өзіне керек деген технологиялар бойынша тренинг өтіп, жаттығулар жасап, сертификат алуына мүмкіндік жасалған. Қызығушылардың өткен курстары негізінде сараптама жасалып, маман іздеп жүрген жұмыс берушілер сараптама нәтижесінің негізінде курсты сәтті өткен болашақ мамандарға ұсыныстар жіберуге мүмкіншілік бар.

Дипломдық жобада жүйенің жұмыс жасау логикасы бірнеше тарауға бөлінген болып табылады. Бірінші тарау осы жобаға ұқсас жобалар мен қосымшалар негізінде теориялық салыстыру және сараптама. Екінші тарау осы жобаны құру барысында қолданылған негізгі және қосымшаның жұмысына қолдау көрсетіп тұрған қосалқы технологиялар туралы айылған. Үшінші тарау, қосымшаның сипаттамасы, және жобалау ерекшеліктері.

Жобаны іске-асыру барысында веб-программалау бағытының материалдары мен ақпараттары шетел әдебиеттері мен ғаламторда жарияланған курстар, веб-блогтар және танымал мақалалардан деретер алынды. Бұл алынған деректердің веб-сілтемелері осы түсініктемелік жазбаның соңында тізім ретінде көрсетіледі.

АННОТАЦИЯ

Этот проект представляет собой веб-приложение, адаптированное ко всем устройствам и операционным системам, чтобы объяснить основы веб-технологий, научить их использовать их. Любой желающий имеет возможность пройти обучение, учения и сертификацию по нужным ему технологиям. По результатам анализа работодатели, ищущие специалиста, имеют возможность направить рекомендации будущим специалистам, успешно прошедшим курс.

Логика системы в дипломном проекте разбита на несколько разделов. Первая глава представляет собой теоретическое сравнение и анализ, основанный на аналогичных проектах и приложениях. Во второй главе рассматриваются основные и вспомогательные технологии, использованные при создании данного проекта и обеспечивающие работу приложения. Третья глава представляет собой непосредственное описание того, как работает приложение.

В ходе реализации проекта материалы и информация в области веб-программирования были получены из зарубежной литературы и курсов, опубликованных в Интернете, веб-блогов и популярных статей. Веб-ссылки на эти данные отображаются в виде списка в конце этого пояснительного примечания.

ANNOTATION

This project is a web application adapted to all devices and operating systems to explain the basics of web technologies, teach them how to use them. Anyone has the opportunity to undergo training, exercises and certification on the technologies he needs. Based on the results of the analysis, employers looking for a specialist have the opportunity to send recommendations to future specialists who have successfully completed the course.

The logic of the system in the graduation project is divided into several sections. The first chapter is a theoretical comparison and analysis based on similar projects and applications. The second chapter discusses the main and auxiliary technologies used to create this project and ensure the operation of the application. The third chapter is a direct description of how the application works.

During the implementation of the project, materials and information in the field of web programming were obtained from foreign literature and courses published on the Internet, web blogs and popular articles. The web links to these data are displayed as a list at the end of this explanatory note.

МАЗМҰНЫ

КІРІСПЕ

XXI ғасыр информациялық ғасыр деп бекер айтылмаған. Тіпті осыдан 15 жыл бұрын біздің ғасыр дәл қазіргідей информациялық дамымаған болған деп айтсақ қате емес. Егер салыстыра қарасақ адам өмірі сол кезде технологиялар мен интернет жүйесіне дәл қазіргідей тәуелді болмаған еді. Студенттер мен оқушылар көбінесе кітапханадағы материалдармен информация алған, ал университеттер мен колледждер www.wikipedia.com сайтынан мәліметтер алатын болған.

Қазіргі таңда адамдардың барлық іс-әрекеттері барынша технологиялардың күшімен автоматтандырылуда. Яғни осыдан 10-15 жыл бұрын істеген іс-әректтер осы күні біздің талпынысымызды керек етпейді. Олардың орнын мобильді қосымшалар мен веб-қосымшалар, сервистер басты деп айтсақ та болады. Біз бұрыңғыдай сатушының алдына барып ақша санап немесе сөмкеміздің іздеп тұрмай-ақ смартфонымыздан керекті қосымшаны ашып төлем жасай аламыз. Қандай да бір информация керек болатын болса біз кітапханаға барудың орнына смартфонымызды немесе кез келген интернет жүйесіне бейімделген құрылғы көмегімен-ақ мәлімет іздеп таба аламыз.

Осыған дейін сатылған курстар мен неше түрлі үйірме тапсырмаларын біз қазіргі уақытта интернет желісінен ақысыз оқып, тіпті орындасақ та болады. Тіпті көптеген социалды желілер арқылы талқылап ненің бұрыс, ненің дұрыс екенін және қосымша ақпарат біліп тануға болады.

Тақырыптың өзектілігі - қазіргі таңда информациялық жүйелердің күннен күнге дамуымен, және сол дамуға үлес қосушы мамандардың тапшылығымен түсіндіріледі. Ақпараттық технологиялардың дамуымен сол технологияларды қолданушы, сол технология көмегімен жаңа да ашылу жасаушы жоба мамандарын, сол технологиялар бойынша баптаулар жасайтын мамандар кажет екенін ескеру керек. Ал сондай мамандарды дайындау мақсатында білім алушының осы қосымша арқылы сауатын ашу өзекті тақырыптардың бірі.

Дипломдық жобаның негізгі мақсаты – информациялық жүйелер бағытына қызығушылық танытушылар мен студенттерге арналған электронды кітап әзірлеу болып табылады. Қосымшаны пайдаланушылар тек қана білімдерінің деңгейін көтеріп қана қоймай, жатығу жасап сол тараудан алған ақпарат бойынша өздерін сынай алады. Әр тарауды бітіргеннен кейін сол тарау бойынша сертификат алуға мүмкіншілік беріледі.

Тақырып бойынша бұл жобаның мақсаты “WEB-технологиялар” пәніне кіріспе электронды кітап жасау, алайда заманауи студенттер мен оқырмандар ақпаратты интернет жүйесінен іздеуіне байланысты бұл жоба тек қана ақпарат беруші кітәп емес сонымен қатар пайдаланушыны тексеруші, пайдаланушыны оқытушы ретінде іске асырылды.

WEB-технологиялардың құралдары болып саналатын әр бағыт менің қосымшамда бір тарау болып табылады. Мысалы: HTML ол бізде жеке бір тарау болса, CSS жеке бір тарау болып есептеледі.

Зерттеу мақсаты. Әртүрлі WEB-технологияларды зерттеу және қосымшаның өзін жобалау барысында қолданылатын құралдарды зерттеу.

Зерттеу пәні. Программалау тілдері, клиенттік бөлігін жобалауда қолданылатын технологиялар, серверлік бөлігін жобалауға қолданылатын технологиялар, фреймворктар және деректер қорлары.

Зерттеу нысаны. HTML, CSS, Javascript, JQuery және т.б.

Қосымшаның негізгі аудиториясы: IT-саласына қызығушылар, ақпараттық жүйелерге байланысты мамандықта оқитын студенттер, ерте жастан қызығушылық танытушы мектеп оқушылары.

1 Теориялық бөлім

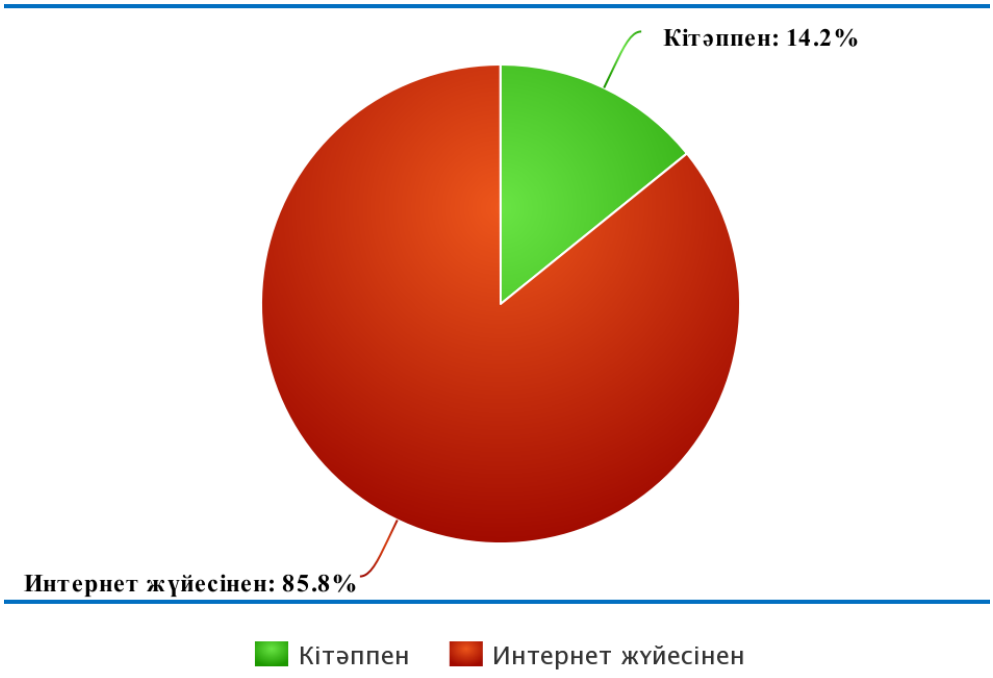
1.1 Жобаның өзектілігін анықтау

Осы жобаны бастамастан бұрын тақырыптың өзектілігі анықталуы керек болды, және қаншалықты өзекті екенін студенттерден алынған сауалнама бойынша сараптама жасалды. Яғни мен осы жобаға байланысты үш сұрақтан тұратын сауалнама жасап, әр студенттің ойын білдім. Әрине, қазіргі таңда сауалнама жасау қиындық тудырмайды, арнайы сервистердің көмегімен онлайн түрде алынған нәтижелер менің ойымды ақтап шықты.

Сауалнама 3 түрлі сұрақтан тұрды:

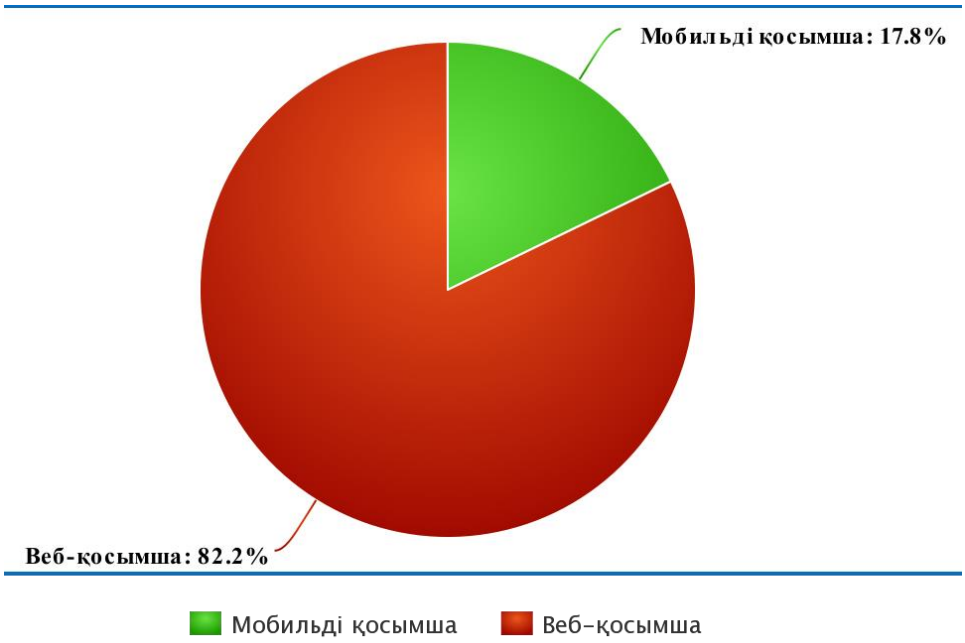
1. Сізге программалауды үйренген кітәппен ыңғайлы ма немесе интернет жүйесінен іздеп оқыған ыңғайлы ма?
2. Егер web-технологияларын үйрететін қосымша бар болса, сол қосымша мобильді болғаны ыңғайлы ма, әлде веб-қосымша ретінде болғаны ыңғайлы ма?
3. Программалауды үйренуде сізге қандай тіл ыңғайлы болатын еді? (қазақ тілінде, орыс ілінде, ағылшын тілінде)

Нәтижелері:



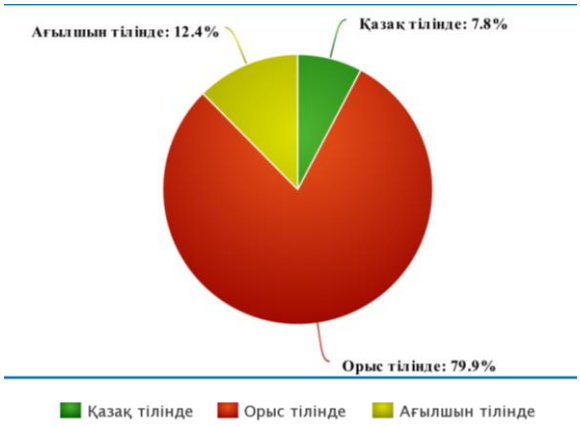
1.1.1-сурет – «1-ші сұрақтың нәтижесі»

- сауалнаманың нәтижесіне қарасақ студенттердің 85 пайызы интернет желісінен оқығанды жөн көреді екен. Келесі сұрақ мобильді қосымшамен оқығанды жөн көре ма екен, немесе веб-қосымша ретінде оқығанды жөн көреді ма екен?



1.1.2-сурет – «2-ші сұрақтың нәтижесі»

2-сұрақтың нәтижесі веб-қосымшаны қолдайды. Сауалнамадан бөлек осы сұрақты бірнеше студенттерге қойған кезде олардан “Неге?” деген қосымша сұрақ қойылған еді. Осы сұраққа олардың жауабы: “Әр тарауда өтіп жатқан тақырыпта дәл сол кезде практика жүзінде қолдансақ, ол көпке есімізде сақталады” – дейді. Соңғы қойылған сұрақ курстардың қандай тілде болғаны ыңғайлы екені туралы еді.



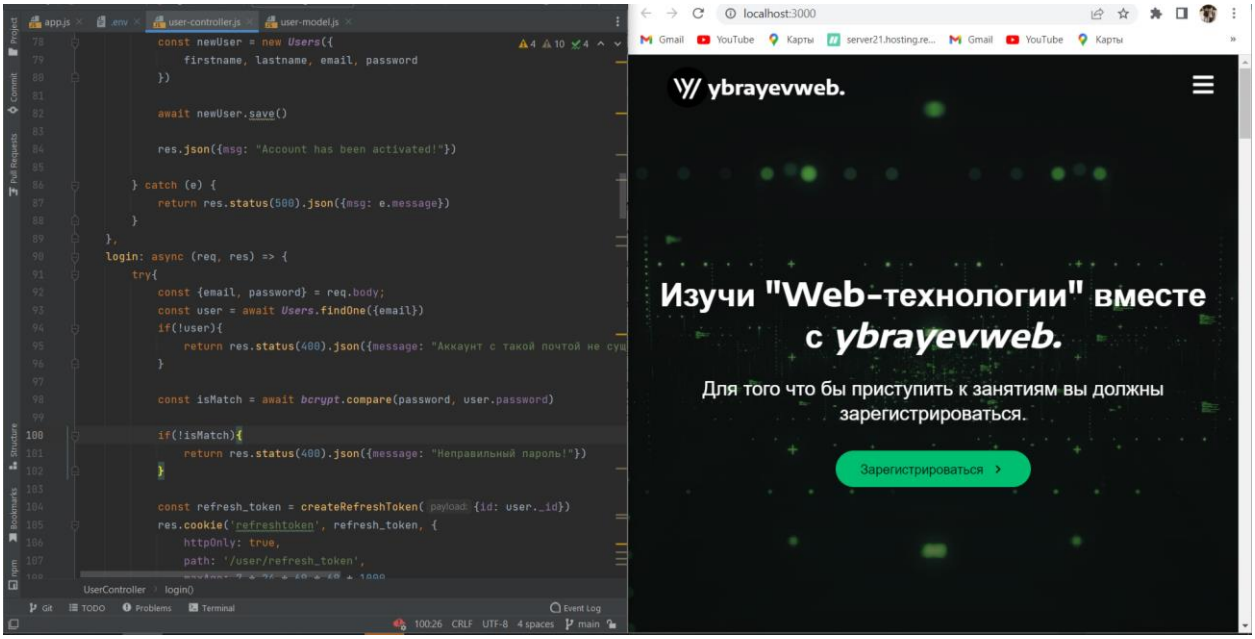
1.1.3-сурет – «3-ші сұрақтың нәтижесі»

- соңғы сұрақтың нәтижесіне қарап біз ешқандай күмәнсіз орыс тілінде қолданушыларға ыңғайлы болатынын айта аламыз.

1.2 Веб-қосымшаның тиімділігі

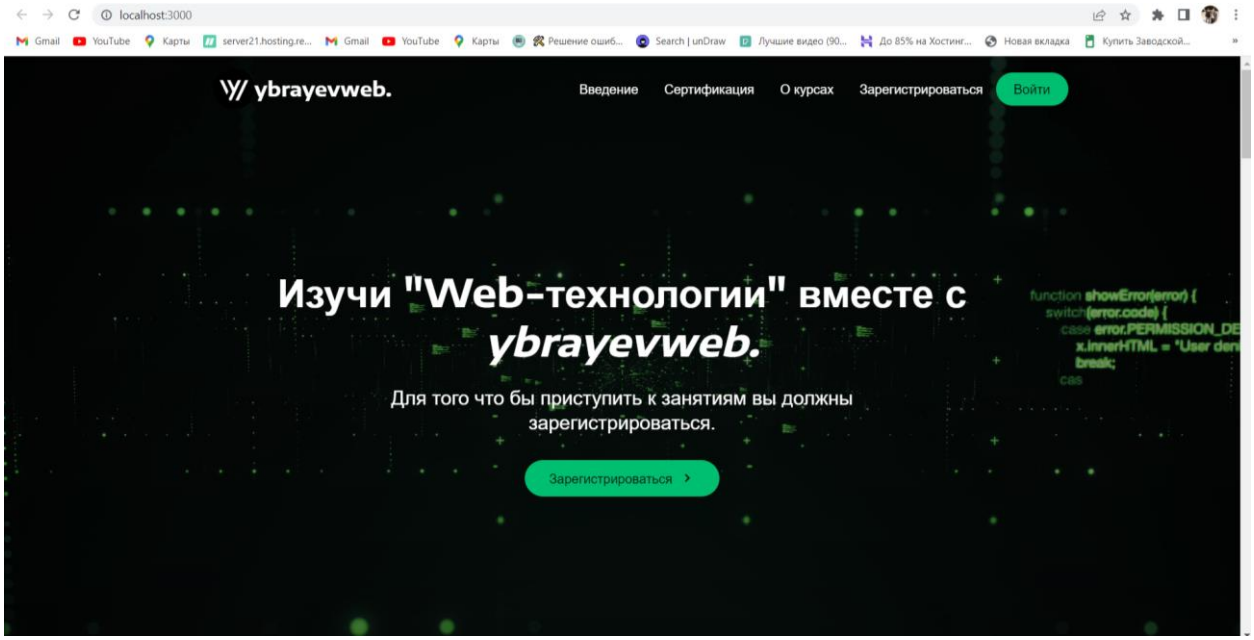
Бұл жобада ең қиын болғаны ол негізгі қосымшаның бейімі, яғни мобильді болғаны дұрыс па әлде веб-қосымша болғаны дұрыс па? Осы мәселені шешуге маған қатты көмектескен ол студенттерден алынған сауалнама еді. Сауалнаманың 2-ші сұрағына сүйене мен еш ойланбастан веб-қосымша болатынын шештім.

Әрине, тақырып бойынша ақпарат алу арқылы біз білімімізді шыңдастырамыз, алайда біз тек теориялық бағытын оқып оны практикада қолдана алмасақ онда курсты толық меңгердім деп айту қиындық танытады. Сол үшін де пайдаланушыларға ыңғайлы болуы үшін олар веб-қосымшаны компьютердің көмегімен ашып, параллельді түрде сол курсқа байланысты ортада жұмыс жасай алса ол өте ыңғайлы болаын еді. Мысалы, мен осы веб-қосымша көмегімен Javascript тілін үйренгелі жатырмын делік. Маған осы веб-қосымшаны веб-браузер көмегімен аша отырып параллельді түрде Javascript редакторында жұмыс жасаған әлдеқайда ыңғайлы болатын еді [1].

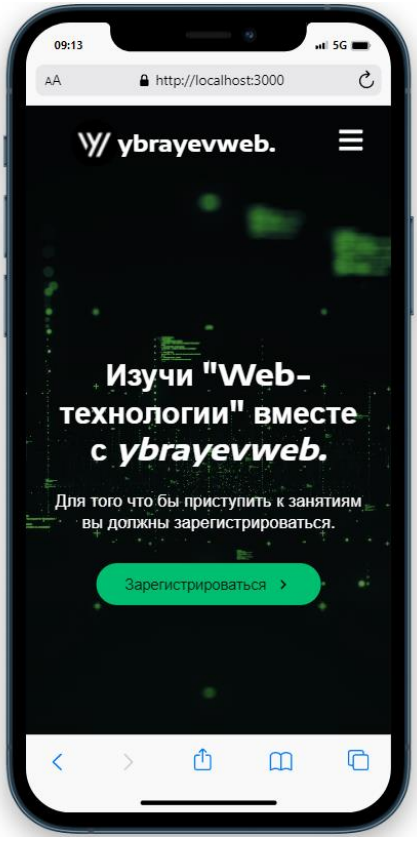


1.2.1-сурет – «Қосымшаны ыңғайлы түрде қолдану мысалы»

Алайда, адами факторларға байланысты, біз теориялық білімімізді кейде ұмытамыз, немесе білмейтін анықтаманы іздеп қалуымыз мүмкін. Тағы бір айта кететін нәрсе, бізде әрқашанда компьютер немесе ноутбук жанымызда болмауы. Сондай жағдайларда біз смартфон немесе планшеттен веб-браузерді ашып ақпарат іздеуіміз мүмкін. Осы жағдайды ескере отыра мен осы жобаны түрлі құрылғыларға бейімделген қылып жасауды ұйғардым. Бұл біздің қолданушыларға әжептәуір жеңілдік беретініне сенімдімін.



1.2.2-сурет – «Қосымшаның компьютерде ашқан кездегі интерфейсі»



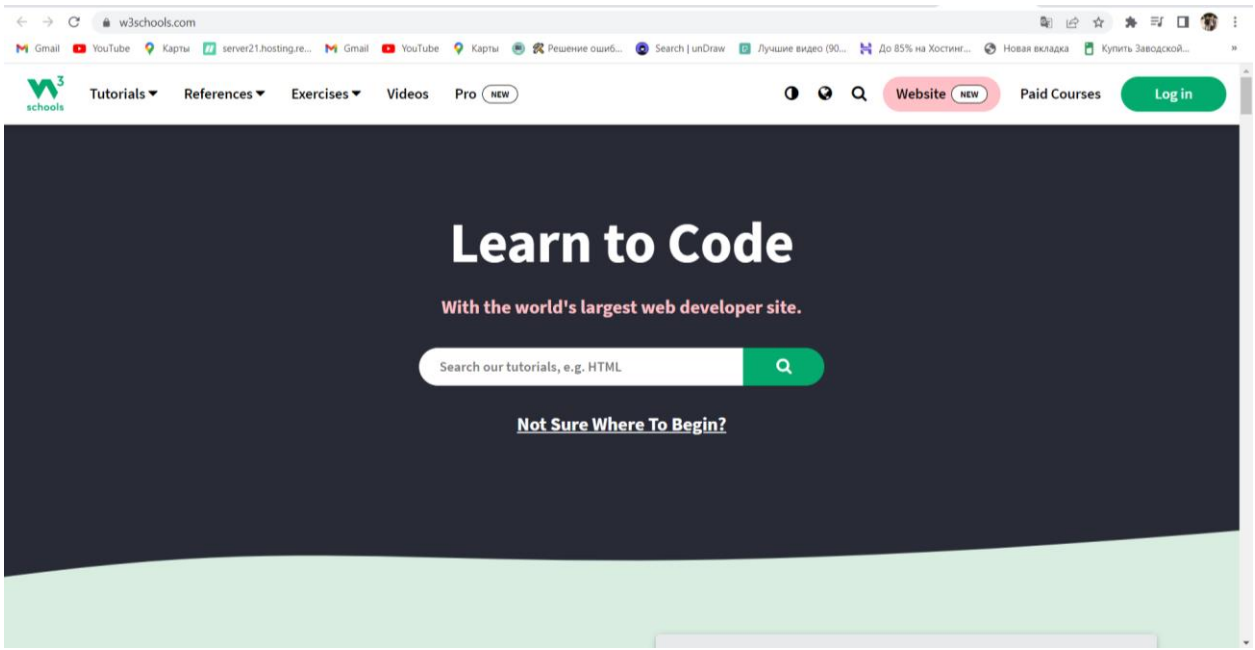
1.2.3-сурет – «Қосымшаның мобильді құрылғыда ашқан кездегі интерфейсі»

1.3 Ұқсас қосымшаларды талдау

Қосымшаларды оның интерфейстерімен не болмаса дизайндарымен салыстыру дұрыс емес деп танимын, өйткені сол екі қосымшаның арасында қайсысында бір артық және пайдалы функционал табатын болсақ, сол қосымшаны басым көреміз. Алайда, қолдану ыңғайлылығы туралы естен шығамау керек екенін ұйғару керек.

Әр жобаның негізгі жетістігі оның бәсекелестік қабілетінде болып табылады. Қосымшаны жобалау барысында оның кемшіліктері мен артықшылықтарын талдау өте маңызды, талдау нәтижесімен біз бәсекеге қабілетті қосымша жасап шығамыз. Осыған сәйкес салыстырмалар жасап мен өзімнің қосымшама ұқсас деген қосымшалардың тізімін құрдым:

1. W3Schools.com(www.w3schools.com) веб қосымшасы – бұл қосымша менің қосымшам секілді web-технологияларын бірнеше бөлімге бөліп қарастырады. Бұл қосымшаның қарапайымдылығы оның әлем бойынша танымал веб-қосымша болмауына ешқандай септігін тигізбейді [2].



1.3.1-сурет – «W3Schools.com веб-қосымшасының басты беті»

Артықшылықтары:

- өте қарапайым интерфейс, барлығы нақты және түсінікті;
- берілген мысалдарды компиляторға салу мүмкіншілігі;
- заманауи технологиялар туралы курстарды қамтиды;
- әр тараудың өзінің тапсырмалары бар;

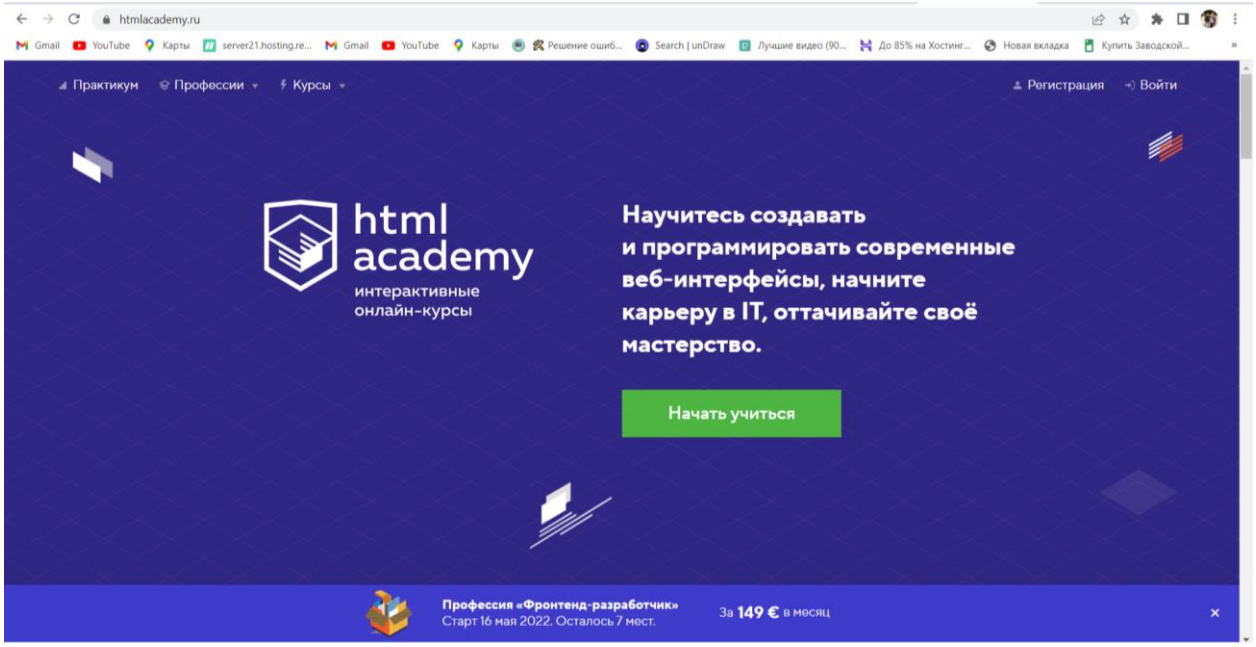
Кемшіліктері:

- ең басты кемшілігі тек ағылшын тілінде болуы, Google Translator API қолданып аударуға болады, алайда ол барлық веб-парақшаны аударған кезде мысал ретінде берілген тапсырма кодтарын өзгертіп жібереді;

- берілген мысалдардың тым қарапайымдылығы, әрине солай сауатымызды ашамыз, дегенмен мысал ретінде күрделі қолданулардың жетіспеушілігі;

- сертификат бағаларының тым жоғары бағада болуы. Сертификат бағасының ен арзаны 95$ болып табылады;

2. HTML Academy(www.htmlacademy.ru) веб қосымшасы – бұл жоба шынайы веб сайттар секілді жобалармен жұмыс жасауды үйретеді. Өте қатты танымал веб-қосымша болмаса да жаңа мамандардың сауатын ашуға арналған жақсы сервис болып табылады.



1.3.2-сурет – «HTML Academy веб-қосымшасының басты беті»

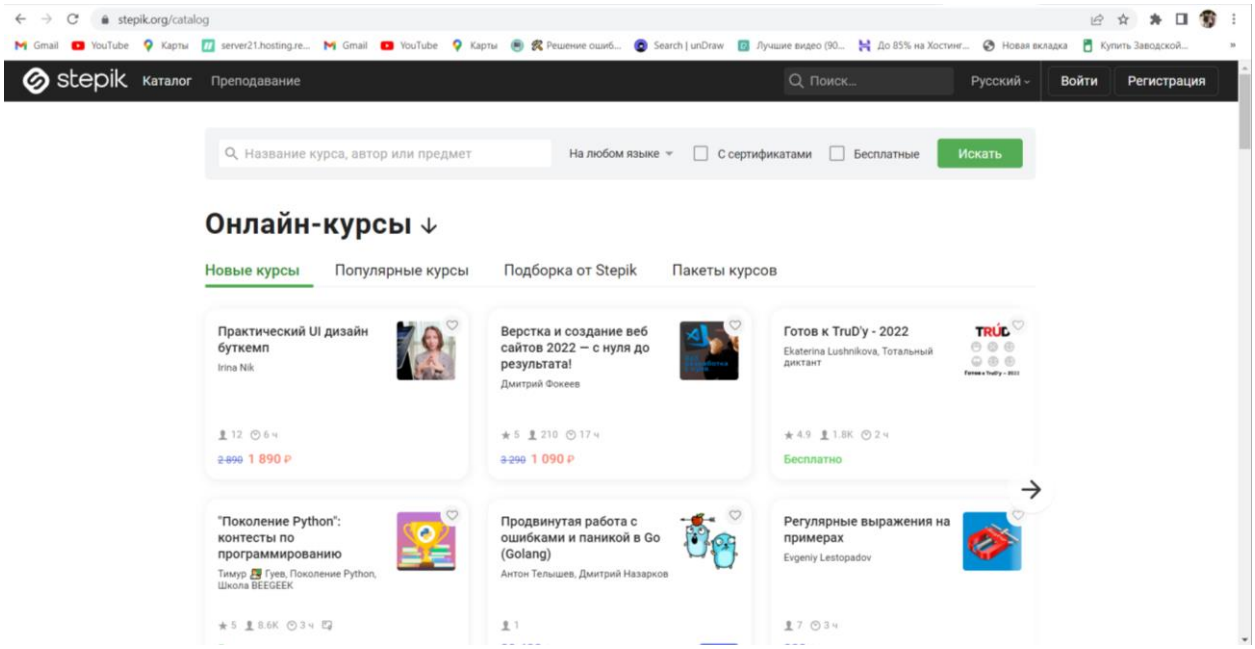
Артықшылықтары:

- өте қызықты және өзекті мысалдар арқылы оқытады;
- қосымша теориялық білім ұсынады;

Кемшіліктері:

- ең үлкен бір кемшілік ол курстардың санында, олар өте аз және әр курс қысқаша материалдармен қамтылған;
- курстардың көбісі ақылы;
- теориялық ақпараттың тапшылығы;

3. Stepik(www.stepik.org) веб платформасы – тек қана web бағытта немесе программалауды үйретуден бөлек басқа да пәндерді қамтитын әмбебап платформа болып саналады.



1.3.3-сурет – «Stepik платформасының басты парақшасы»

Артықшылықтары:

- лекциялардың бейнежазбасы, әр тарауда бірнеше бейнежазба бар, және солардың арқасында тақырыпты түсіну оңайға түседі;
- әр бейнежазбаның жазбаша түрі ұсынылған, яғни материалдарды жазбаша түрде алса да болады;
- тегін сертификат алу мүмкіншілігі бар;

Кемшіліктері:

- кейбір тараудағы тапсырмаларды орындау барысында компилляторлардың қатесі;
- кейбір мысалдардың тым ауыр болуы жаңа пайдаланушылардың курсқа деген қызығушылығын түсіруі;
- басым курстардың ақылы болуы;

1.4 Термин және қысқартылған сөздер, олардың мағыналары

Жобаны іске асыру кезінде қолданылған технологиялар жайында ғылыми терминдер мен қысқартылған сөздер кездесуі мүмкін. Сондай түсініксіз терминдердің анықтамасын 1.4.1-кестеде көрсетілген.

1.4.1-кесте – Қысқартылған және термин сөздердің анықтамалары

UX	User Experience (Қолданушы тәжірибесі)
UI	User Interface (Қолданушы интерфейсі)
IDE	Integrated Development Environment (Интеграцияланған даму ортасы).
UML	Unified Modeling Language (Бірыңғайландырудың модельденген тілі).
HTML	Hyper Text Modified Language (гипемәтінді белгілеу тілі – web-технологияларының бірі болып есептеледі).
CSS	Cascading Style Sheet (каскадты стиль кескіндері – web-технологияларының бірі болып есептеледі).
Javascript	Объектіге бағытталған скрипттік (немесе сценарийлер) бағдарламалау тілі.
JQuery	Бұл JavaScript кітапханасы, оның мақсаты сценарий бағдарламалауды жеңілдету.
API	Application Programming Interface (қолданбалы бағдарламалау интерфейсі)
HTTP	Hyper Text Transfer Protocol (қолданбалы деңгей протоколы)
HTTPS	Hyper Text Transfer Protocol Secure (жетілдірілген қауіпсіздік үшін шифрлауды қолдауға арналған HTTP протоколының кеңейтімі.)
MobX	Қосымшаның сыртқы күйін басқаруға арналған оқшау кітапхана
GraphQL	API және олар іске қосылатын орта үшін сұрау тілі.
Git	Таратылған нұсқаны басқару жүйесі
GET	Сайттан деректерді оқу әдісі
POST	Веб-сервер хабарламаның негізгі бөлігіне енгізілген деректерді сақтау үшін қабылдайтын сұрауды жіберу үшін пайдаланылады.
PUT	Жаңа ресурс жасайды немесе мақсатты ресурс көрінісін сұрау органында берілген деректермен ауыстырады.
PATCH	Қауіпсіз де, идемпотентті де емес және толық немесе ішінара жаңартуларға мүмкіндік беретін әдіс

2 Технологиялық бөлім

2.1 Қосымша жұмысын қамтамасыз ететін жүйелер

Осы дипломдық жұмысты(қосымшаны) жасау барысында веб-қосымшаны қандай технологиялармен жобалау мәселесін шешу өте күрделі болған еді. Алайда қосымша максималды түрде заманауи болуы үшін заманауи технологияларды қолданған жөн деп шешілді.

Қосымшаны прораммалау кезінде серверлік шеңберді жасап шығу үшін және клиенттік шеңберді жасап шығу үшін де көмекші фреймворк технологиялары қолданылған еді. Фреймворк – бұл қандай да бір бағдарламаны программалау процессінде ауыр жобалауды пайдалануды жеңілдетуге арналған технология болып табылады. Фреймворктардын арасында ең танымалдары:

- Laravel фреймфоркі;
- Angular фреймворкі;
- Express фреймворкі;
- Django фреймворкі;

Клиенттік шеңбер (frontend) – пайдаланушы интерфейсін жүзеге асырады, серверге сұраныстарды қалыптасырады, және серверден келген жауаптарды өңдер клиентке жеткізеді. Яғни клиент клиенттік шеңбер көмегімен серверлік шеңберге сұраныс жібереді. Ал серверлік шеңбер сұранысы қабылдап, есептеулер жүргізіп, веб-парақшаны құрастырып HTTP протоколы арқылы қайтадан клиентке жібереді [3].

Серверлік шеңбер (backend) – бұл жүйе болып табылады және қолданбаға, деректер қорына – веб сайтты немесе қолданбаны пайдаланушыға жеткізу үшін сахнаның артында жұмыс істейтін “кодты” білдіреді. Яғни қосымшаның деректер қорымен байланысын да қамтамасыз ететін осы серверлік шеңбер болып табылады [4].

2.2 Осы қосымшаның жұмысын қамтамасыз етіп тұрған стек

Стек дегеніміз – бұл осы қосымшаны жасап шығу үшін қолданылған технологиялардың кейбер жиынтығы болып табылады.

Бүгінгі күні стектердің саны жетерліктей, алайда солардың арасында жетерліктей танымалдарының бірі MEAN стек болып табылады. Оның құрамы:

- Mongo DB (деректер қоры);
- Express.js (серверлік шеңбер фреймворкі);
- Angular.js (клиенттік шеңбер фреймворкі);

- Node.js (серверлік шеңбер);

Алайда, тура осы стек бірақ Angular.js фреймворкының орнына React.js библиотекасын қойып біз жаңа, MERN стегін ала аламыз. Бұл стек те танымал стектердің тізіміне кіреді. Дәл осы стек біздің қосымшаның негізгі жұмысын атқарып тұр. (2.2.1-сурет)

2.2.1-сурет – «Қолданылған стек сипаттамасы»

2.3 MERN стегінің ерекшеліктері

Бұл стекпен жұмыс жасардың алдында мен осы стектің ерекшеліктерін біліп алуды жөн көрдім. Бұл стекті зерттеу барысында ерекшеліктер тізімін жасап шығардым:

1. Серверлік шеңбер (Node.js, Express.js) және клиенттік шеңбер (React.js);

Серверлік шеңбер және клиенттік шеңбер екеуі екі бөлек нәрсе екенін білуіміз керек. Яғни клиентік шеңбер басқа репозиторияда болуы да мүмкін, және олар бір біріне тәуелсіз, өйткені серверлік шеңбер клиенттік шеңберсіз де жұмыс жасай алады [5].

2. API арқылы байланыс;

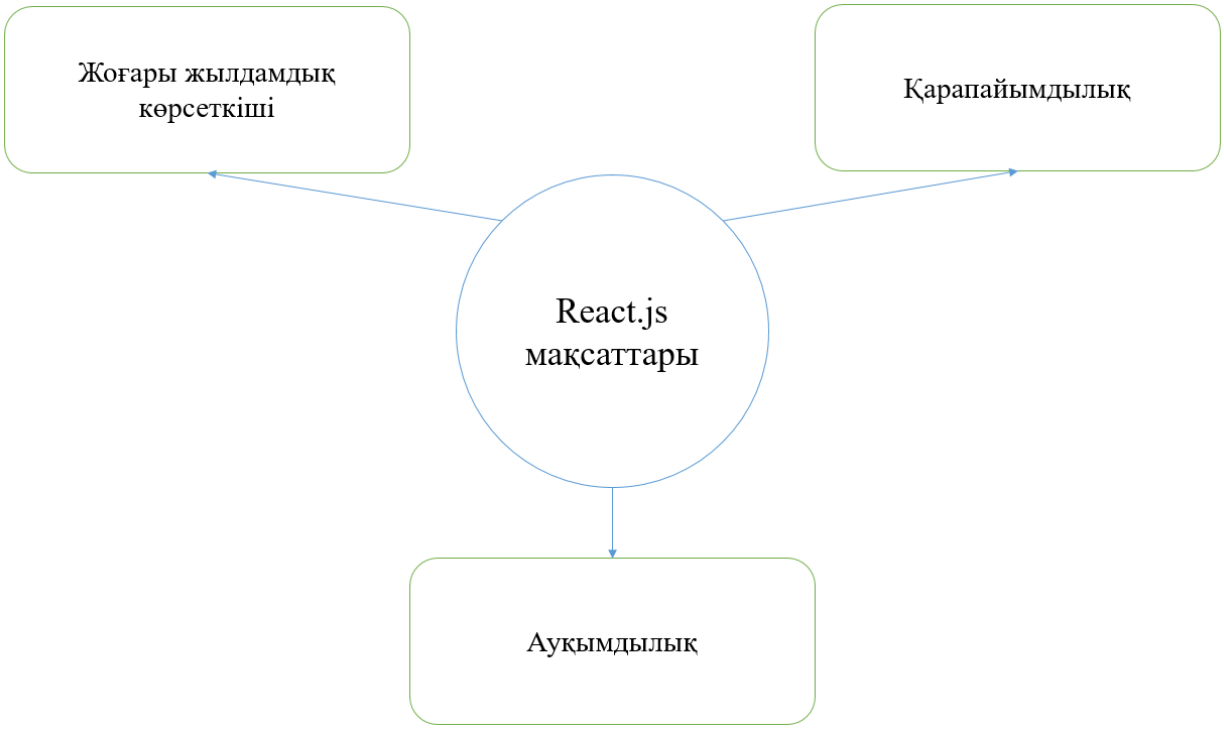
Серверлік және клиенттік жүйелер бір бірімен өзара байланысуы үшін API қолданылуы керек. API серверлік шеңберде жасалынады, өйткені клиенттік шеңберден енген сұраныс осы серверлік шеңберге жіберіледі [6].

2.4 Клиенттік бөлім (React.js)

Үстіде айта кеткендей веб-қосымшаның клиенттік бөлімі JavaScript тілінің React.js библиотекасы арқылы жасалды.

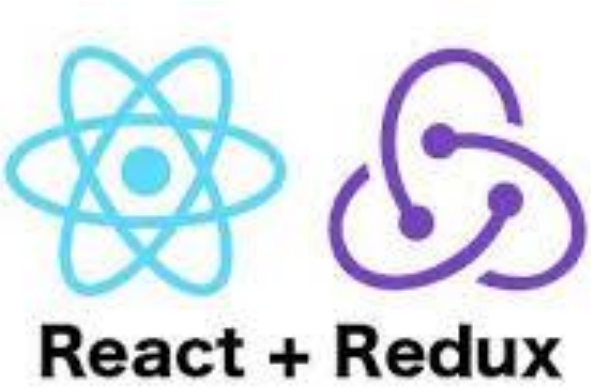
React.js библиотекасы кроссплатформалық қосымшаларды жасауда “frontend” бөлігін программалауға арналған технология. Айта кеткенімдей заманауи технология болып табылады, өйткені ол 2013-ші жылы ғана Facebook, Instagram әзірлеуші-инженерлерінің ықпалымен ойлап табылған. Соңғы шыққан нұсқасы – 18.0.0 (29-наурыз 2022ж.).

React.js көбінесе SPA қосымшалары мен мобильді құрылғыларға арналған қосымшалар үшін қолданылады. Алайда қосымшаның жұмысын жақсартуда белгілі бір мақсатар бар, олар:



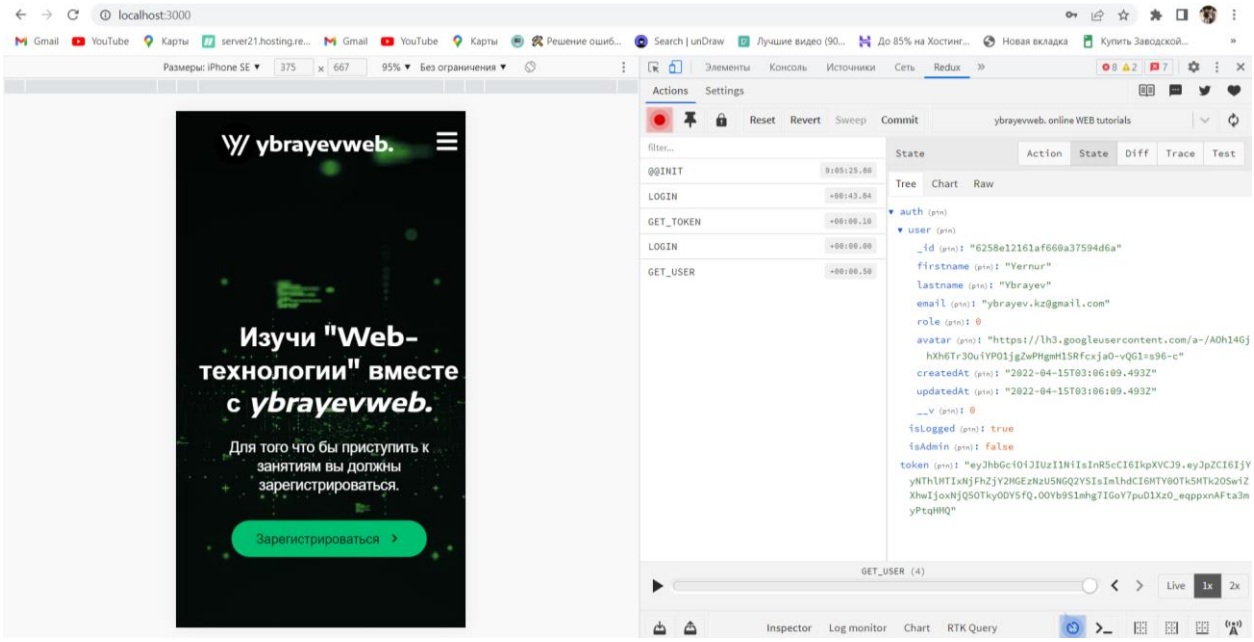
2.4.1-сурет – «React.js библиотекасының мақсаттары»

React.js кейде басқа да библиотекалармен байланысу арқылы жұмыс жасайды, сол библиотекалардың ішінде ең танымалдары MobX, Redux және де GraphQL. Өзімнің жобамда мен осы React.js библиотекасын Redux библиотекасымен байланыстыра отыра жасадым [7].



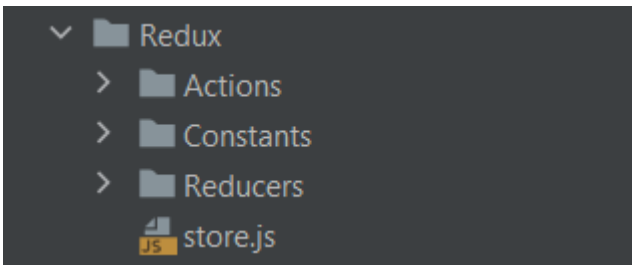
2.4.2-сурет – «React.js және Redux библиотекалары»

Күйлермен жұмысты React.js библиотекасының Context API концепциясымен жасауға да болады, алайда ақпараттың аз болуына орай Redux библиотекасы қолданылды. Осы библиотеканың негізінде қолданушы авторизацияланған ба әлде авторизация жүйесінен өтпеген екенін, токен алған немесе алмағанын, токеннің уақыты өтіп кеткенін біле аламыз. (2.4.3-сурет)



2.4.3-сурет – «Redux ортасының күйді сақтауы»

Redux интеграциясын үш түрлі директория және осыларды басқаратын бір Javascript файл қоса арқылы жасалынды, яғни құрамында 3 сервис болады. (2.4.4-сурет)



2.4.4-сурет – «Redux құрамы»

- Actions директориясы – қолданушының курсты қосқанын немесе басқа да істеген манипуляциларын сақтайды. Яғни әр қолданушының істеген және сақтаған іс-әрекеттері мен деректерін осы “Actions” директориясындағы файлдар сақтатады;
- Constants директориясы – қосымшаның осы Redux библиотекасын инеграциялау барысында қолданған константаларының жариялануы мен тізімі;
- Reducers директориясы – 2.4.3-суреттегі көрсетілген ақпаратты формалайды, яғни сол суретке қарасақ бізге керек ақпаратардың бәрі көрсетілген, олардың қалай көрсетілуі осы “Reducers” директориясының құрамындағы файлдар көрсетеді;
- Осы аталған 3 директорияның басын қосып бір бірімен байланыстыратын “store.js” файлы болып табылады;
- Алайда осы Redux библиотекасынан бөлек клиенттік шеңбердің жұмысын жеңілдеу мақсатында хуктар (React Hooks) қолданылды. Негізі хуктардың 9 түрі бар, алайда бұл қосымшаны программалау барысында соның 6-уы қолданылды. Олар:
  - useState;
  - useReducer;
  - useEffect;
  - useRef;
  - useContext;
  - useCallback;

Осы хуктарды қолдануда мысал ретінде жүйеге кірген/кірмеген күйлерімен жұмыс жасау барысынан бір жағдай көрсетуді шештім.

Қолданушы біздің веб-қосымшаға ең бірінші рет кіргенде оның ешқандай деректері болмайды, және ешқандай деректер қорында ол қолданушының деректері сақталмаған күйде болады. Ал қосымша панелінде тіркелу (регистрация) және жүйеге кіру (войти) батырмалары болады. Ол жүйеге тіркелгенде немесе жүйеге кіргенде Redux қолданушының ақпараттарын қабылдайды ал хуктар күйді ескере отырып панельдегі “тіркелу” батырмасының орнына менің жеке кабинетіме өтетін сілтемені рендеринг жасайды. Ал “жүйеге кіру” батырмасын жүйеден шығу батырмасына алмастырады [8]. (2.4.5-сурет)

```
const [scrollNav, setScrollNav] = useState( initialState: false);

const auth = useSelector( selector: state => state.auth)

const {user, isLogged} = auth
```

2.4.5-сурет – «Хуктардың қолданылуы»

Қалған интеграцияланған пакеттер қосымшаның “frontend” бөлімін түсінікті және қолдануға ыңғайлы болу үшін қолданылды. Барлық қолданылған пакеттер тізімін біз JSON форматта “package.json” файлында көре аламыз.

```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.1",
    "@testing-library/react": "^12.1.2",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^0.26.1",
    "emailjs": "^3.7.0",
    "emailjs-com": "^3.2.0",
    "express-validator": "^6.14.0",
    "materialize-css": "^1.0.0-rc.2",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-facebook-login": "^4.1.1",
    "react-google-login": "^5.2.2",
    "react-icons": "^4.3.1",
    "react-redux": "^7.2.6",
    "react-router-dom": "^6.2.1",
    "react-scripts": "5.0.0",
    "react-scroll": "^1.8.4",
    "react-toast-notifications": "^2.5.1",
    "react-toastify": "^8.1.1",
    "styled-component": "^2.8.0",
    "styled-components": "^5.3.3",
    "web-vitals": "^2.1.4"
```

2.4.6-сурет – «Қолданылған пакеттердің тізімі»

2.5 Серверлік бөлім(Node.js + Express.js)

Серверлік бөлімді жасау жобаны инициализация жасаудан басталады, яғни инициализация жасау жобаға ат беру, жобаның авторын, жобаның Git репозиториясын, қосымшаның басты файлын және қосымша туралы сипаттаманы беруден басталады. Бұл ақпаратты біз клиенттік бөлімдегідей “package.json” файлында көре аламыз. (2.5.1-сурет)

```
{
  "name": "full_auth",
  "version": "1.0.0",
  "description": "server side Ybrayevweb",
  "main": "app.js",
  "scripts": {
    "dev": "nodemon app.js"
  },
  "author": "YernurYbrayev <ybrayev.kz@gmail.com>",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cloudinary": "^1.28.1",
    "cookie-parser": "^1.4.6",
    "cors": "^2.8.5",
    "dotenv": "^16.0.0",
    "express": "^4.17.3",
    "express-fileupload": "^1.3.1",
    "googleapis": "^95.0.0",
    "jsonwebtoken": "^8.5.1",
    "mongoose": "^6.2.4",
    "node-fetch": "^3.2.1",
    "nodemailer": "^6.7.2"
  },
  "devDependencies": {
    "nodemon": "^2.0.15"
  }
}
```

2.5.1-сурет – «Серверлік шеңбердің “package.json” файлы»

Осы процедурадан кейін мен express.js фреймворкын қосып ары қарай деректер қорын қостым. Үстіде айта кеткендей мен осы серверлік бөлімді клиенттің бөлімге қосу мақсатында API құрауым керек еді. Бұл процедура серверлік шеңбердің бас файлында (менің жағдайымда – “app.js”) жасалынады [9]. (2.5.2-сурет)

```
require('dotenv').config()

const express = require('express')
const mongoose = require('mongoose')
const cors = require('cors')
const cookieParser = require('cookie-parser')
const fileUpload = require('express-fileupload')


const app = express()

app.use(express.json())
app.use(cors())
app.use(cookieParser())
app.use(fileUpload( options: {
    useTempFiles: true
}))

app.use('/user', require('./routes/user-routes'))
app.use('/api', require('./routes/upload'))
```

2.5.2-сурет – «app.js файлының бастамасы»

Ары қарай мен қосымшаны жұмыс жасату үшін порт беруім керек. Порт басқа ешқандай қосымшамен жұмыс жасамауы керек, бос порт болуы керек. Егер қандай да бір қосымша сол портпен жұмыс жасап тұрса қазіргі қосымшамыз қосылмайды. Яғни біз серверлік шеңберге бір порт берсек, клиенттік шеңберге бір порт береміз. Ал екі бөлікті біз прокси арқылы байланыстыруымыз керек. Клиенттік бөлімнен келген барлық сұраныстар мен серверлік бөлімнен шыққан барлық жауаптар, операциялар осы прокси арқылы бір біріне жету керек.

Проксиді қосу үшін тағы да “package.json” файлымен жұмыс жасаймыз, алайда біз тек клиенттік шеңбердің файлына ғана өзгерту еңгіземіз [10]. (2.5.3-сурет)

```
"proxy": "http://localhost:5000",
```

2.5.3-сурет – «Клиенттік шеңберден серверге бағытталған прокси»

Клиенттік шеңберден келіп түскен әр сұраныс серверлік бөлім арқылы деректер қорынан деректер сұрайды және тек қана сұраудан бөлек оны өзгертуге, жоюға не болмаса жаңадан дерек қоса алады. Бұндай операцияларды орындату үшін API керек деп үстінде айта кеткенбіз. Алайда API-дің ішінде бағыттар тізімі (маршруты) болуы керек. Менің жағдайымда маршруттар:

user-routes.js | user-controller.js | routes.js

```
const router = require('express').Router()
const userController = require('../controllers/user-controller')
const auth = require('../middlewares/auth')
const authAdmin = require('../middlewares/auth-admin')

router.post( path: '/register', userController.register)
router.post( path: '/activation', userController.activateEmail)
router.post( path: '/login', userController.login)
router.post( path: '/refresh_token', userController.getAccessToken)
router.post( path: '/forgot', userController.forgotPassword)
router.post( path: '/reset', auth, userController.resetPassword)
router.get( path: '/userinfo', auth, userController.getUserInfo)
router.get( path: '/allusersinfo', auth, authAdmin, userController.getUsersAllInfo)
router.get( path: '/logout', userController.logout)
router.patch( path: '/update', auth, userController.updateUser)
router.patch( path: '/update_role/:id', auth, authAdmin, userController.updateUsersRole)
router.delete( path: '/delete/:id', auth, authAdmin, userController.deleteUser)
router.post( path: '/google_login', userController.googleLogin)


module.exports = router
```

2.5.4-сурет – «Осы қосымшаның серверлік шеңберінің маршруттары»

2.5.4-ші суретте біз маршруттарды көріп қана қоймай ол қандай метод (GET, POST, PUT, PATCH, DELETE) екенін көре аламыз. Және де аргументтері ретінде осы маршрутты орындау үшін қандай дерек керек екенін, ал соңынан оның іс-әрекетін беретін контроллер әдісін береміз.

“Клиенттен келген дерек осы маршруттардың тізімінен қайсысымен жұмыс жасау керек екенін қалай біледі?” – деген сұрақ пайда болады. Әрине клиенттік шеңберден келген сұраулар қай маршрутқа бару керек екенін алдын

ала біледі, өйткені біз клиентте болған іс-шараны бір метод арқылы серверге жіберуіміз керек, ал сол методта осы маршрут аргумент ретінде беріледі [11].

```
const loginHandler = async () => {
    try{
        const data = await request( url: '/user/login',  method: 'POST',  body: {...form})
        message(data.message)
        localStorage.setItem('firstLogin', true)
        dispatch(dispatchLogin())
        navigate("/");

    } catch (e) {

    }
```

2.5.5-сурет – «Клиенттен келетін сұранысты ұйымдастыратын метод мысалы»

Ал енді келген сұранысқа белгілі бір операциялар жасау серверлік бөлімнің актерларына (actors), яғни обьектіге байланысты өзінің контроллерінің методы арқылы іске асады. Мысалы, мен осы қосымшаға тіркелгім келеді делік. Мен бұл қосымшада қолданушы рөлін (User) атқарамын. Демек менің контроллерім менің жағдайымда – userController болып табылады. Ал осы контроллердің ішінде бірнеше функциялар бар болып табылады. Мен тіркелу үшін осы контроллердің ішіндегі тіркелуге жауап беретін методты шақырамын (userController.register) 2.5.6-сурет.

```
const UserController = {
    register: async (req, res) => {
        try{
            const {firstname, lastname, email, password, cf_password} = req.body

            if(!firstname || !lastname || !email || !password){
                return res.status(400).json({message: "Пожалуйста заполните все поля."})
            }

            if(firstname.length < 2 || lastname.length < 2){
                return res.status(400).json({message: "Имя и фамилия не должны иметь меньше двух символов."})
            }

            if(!validateEmail(email)){
                return res.status(400).json({message: "Неправильная почта, попробуйте еще раз."})
            }


            const user = await Users.findOne( filter: {email})

            if(user){
                return res.status(400).json({message: "Такой email уже зарегистрировано."})
            }

            if(password.length < 8){
                return res.status(400).json({message: "Пароль должен быть не меньше 8 символов."})
            }

            if(password != cf_password){
                return res.status(400).json({message: "Ошибка при подтверждении пароля, попробуйте заново."})
UserController  ›  login()
```

2.5.6-сурет – «userController контроллеріндегі тіркелу жүйесінің (register) функциясы»

2.6 Жүйеге кіру, тіркелу, құпиясөзді орнына келтіру функционалы

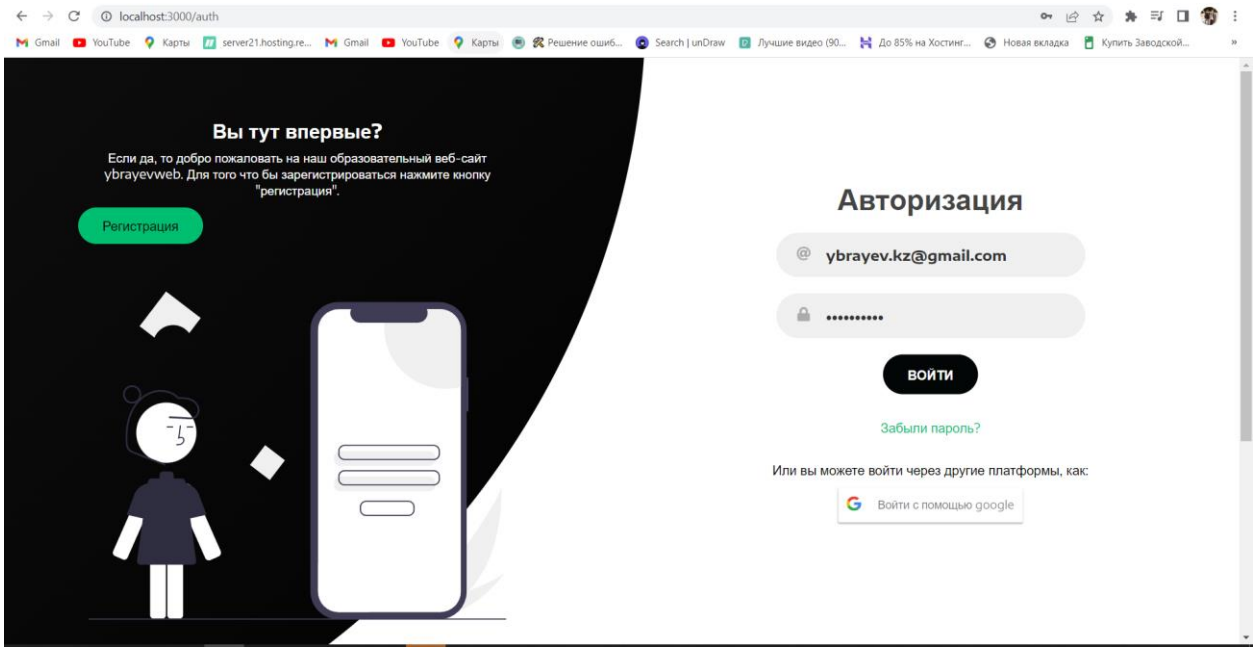
Қосымшаны қолдану үшін авторизациядан өту керек шарт екенін ұмытпауымыз керек. “Авторизация жүйесі не үшін керек?” – деген сұрақ туындауы мүмкін. Егер қосымшаны әркім қолдануына рұқсат етілсе авторизация жүйесі керек те емес деген тұжырымдамалар да болды. Алайда оның біршама пайдасы бар екенін ескере кеткеніміз жөн. Сараптама жасай мен жүйеге кіру мүмкіншіліктерінің осындай пайдалы жақтарының тізімін құрдым.

1. Сақтау. Пайдаланушы курстарды өту барысында белгілі бір жерге келіп тоқтауы мүмкін. Кейін жүйеге қайта кіргенде оның қай жерге келіп тоқтағаны, қандай тапсырмалар орындағаны сақталып тұрса бұл пайдаланушылардың уақытын үнемдеп, қызығушылықтарын арттырады.

2. Автоматтандыру. Пайдаланушы қосымшаны падалану барысында енгізу керек формалар болуы мүмкін. Мысалы, сертификат алу үшін өзінің аты жөнін енгізу, курсқа жазылу барысында толтыру керек формалар сияқты функционалдар автоматтандырылады [12].

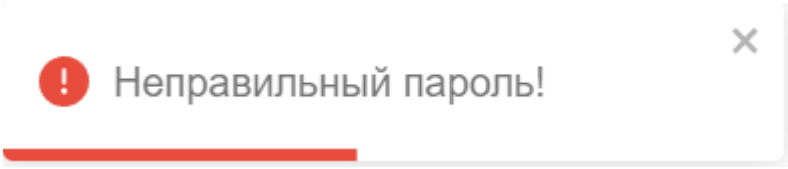
2.6.1 Жүйеге кіру (авторизация)

Жүйеге кіру үшін бізге керегі бар болған тіркелуден өткендегі электронды пошта, және өзіңіз қойған құпиясөз. (2.6.1.1-сурет)



2.6.1.1-сурет – «Жүйеге кіру парақшасы»

“Войти” батырмасын басқаннан кейін сіздің еңгізген ақпараттарыңыз бірнеше қадамнан тұратын валидация-дан (еңгізген ақпараттың дұрыс-бұрысын тексеру операциялары) өтеді. Егер де сіз еңгізген пошта бұл қосымшада тіркелмеген болса немесе сіз еңгізген құпиясөз жалған болса, серверден келген жауап клиент бөліміне “react-toast” библиотекасы арқылы экранға шығарылады [13].



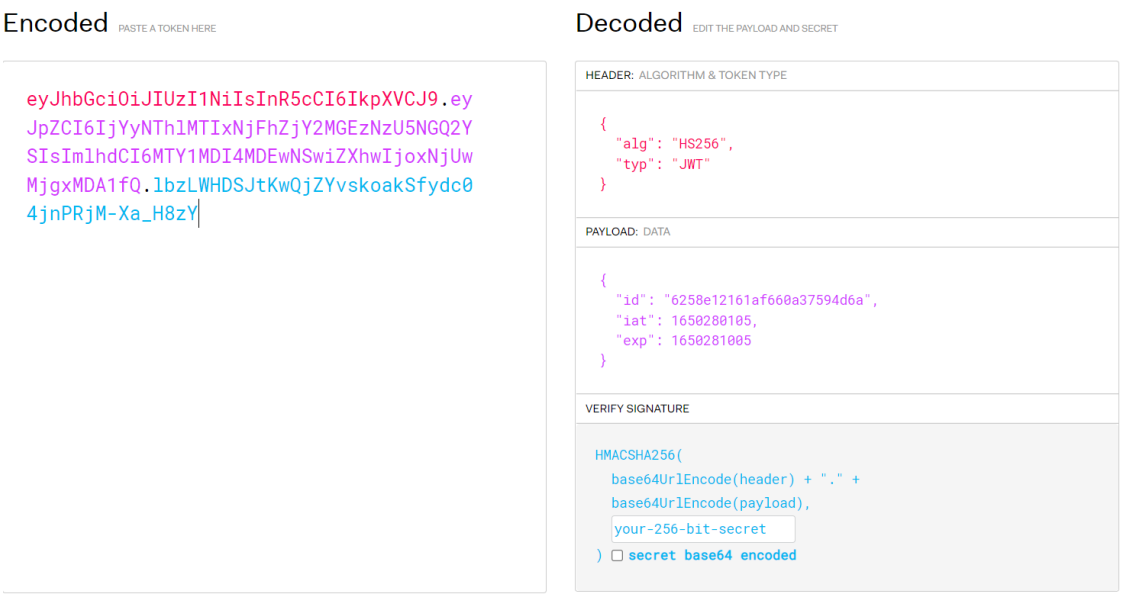
2.6.1.2-сурет – «Қате еңгізілген құпиясөзге серверден келген жауап»

Жүйеге кіргеннен кейін сізге токен (jsonwebtoken библиотекасының қамтамасыз ететін токені) тағайындалады. Токен – сіздің жүйеде жұмыс жасау үшін карточканыз секілді болып табылады. Мысалы біз оқу орнына келгенде біз ID-картамызды көрсеткеннен кейін бірақ кіре аламыз. Яғни қысқаша айтқанда жүйеге кіруде бірілетін шифрланған документ болып табылады. Оның өзінің өзекті болатын уақытысы болады. Егер де қандай да бір жағдайда сіздің токеніңіз жойылса сіз автоматты түрде жүйеден шығасыз [14].


```
▼ auth (pin)
  ▼ user (pin)
      _id (pin): "6258e12161af660a37594d6a"
      firstname (pin): "Yernur"
      lastname (pin): "Ybrayev"
      email (pin): "ybrayev.kz@gmail.com"
      role (pin): 0
      avatar (pin): "https://lh3.googleusercontent.com/a-/AOh14Gj
        hXh6Tr3OuiYPO1jgZwPHgmH1SRfcxjaO-vQG1=s96-c"
      createdAt (pin): "2022-04-15T03:06:09.493Z"
      updatedAt (pin): "2022-04-18T10:37:08.756Z"
      __v (pin): 0
    isLogged (pin): true
    isAdmin (pin): false
  token (pin): "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY
    yNThlMTIxNjFhZjY2MGEzNzU5NGQ2YSIsImlhdCI6MTY1MDI3ODIzNiwiZ
    XhwIjoxNjUwMjc5MTM2fQ.gR5mq5hjyxcCRbm6_L7mmmrPRadKWu0Vr8Im
    chSrqzk"
```

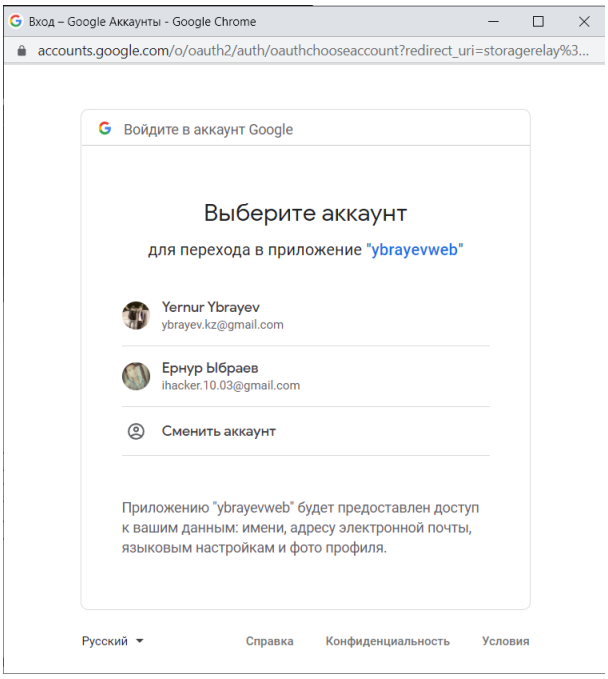

2.6.1.3-сурет – «Жүйеге кіргеннен кейін берілетін токен»

Осы токенді біз дешифрлейтін болсақ ол біз туралы ақпараттан тұратынына көз жеткіземіз, яғни ол біздің id-мызды сақтап тұрғанын және де өзекті секундтарынан тұратынын көре аламыз. (2.6.1.4-сурет)



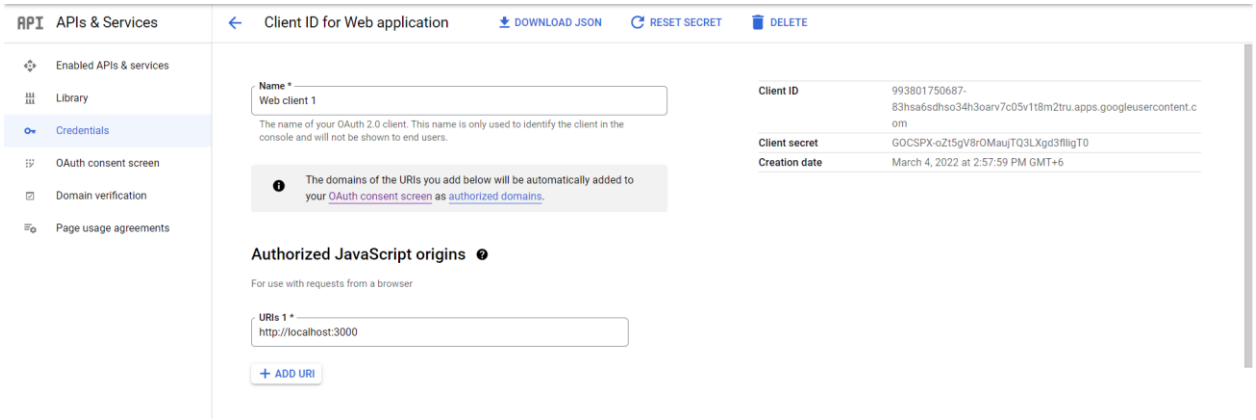
2.6.1.4-сурет – «Дешифрленген токен»

Жүйеге кірудің екінші жолы қарастырылған, бұл – Google OAuth арқылы. Яғни сізге бар болғаны “войти с помощью Google” батырмасын басып өз Google аккаунтыңызға кіру керек болады. (2.6.1.5-сурет)



2.6.1.5-сурет – «Google OAuth арқылы жүйеге кіру»

Google OAuth функционалын қосымшаға енгізу үшін маған Google Cloud Console сервисінің көмегі керек болды. (2.6.1.6-сурет)

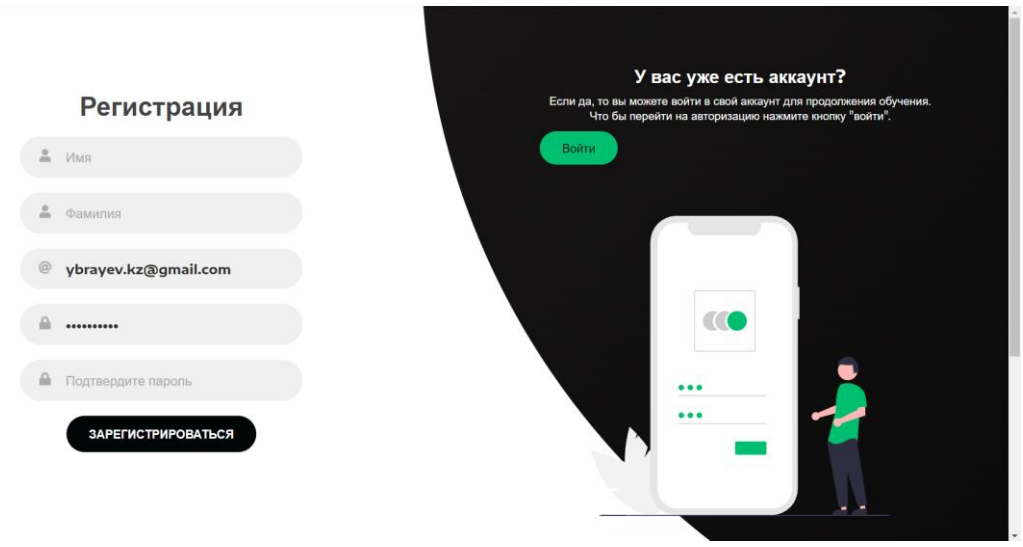


2.6.1.6-сурет – «Google Cloud Console баптаулары»

2.6.2 Жүйеге тіркелу (регистрация)

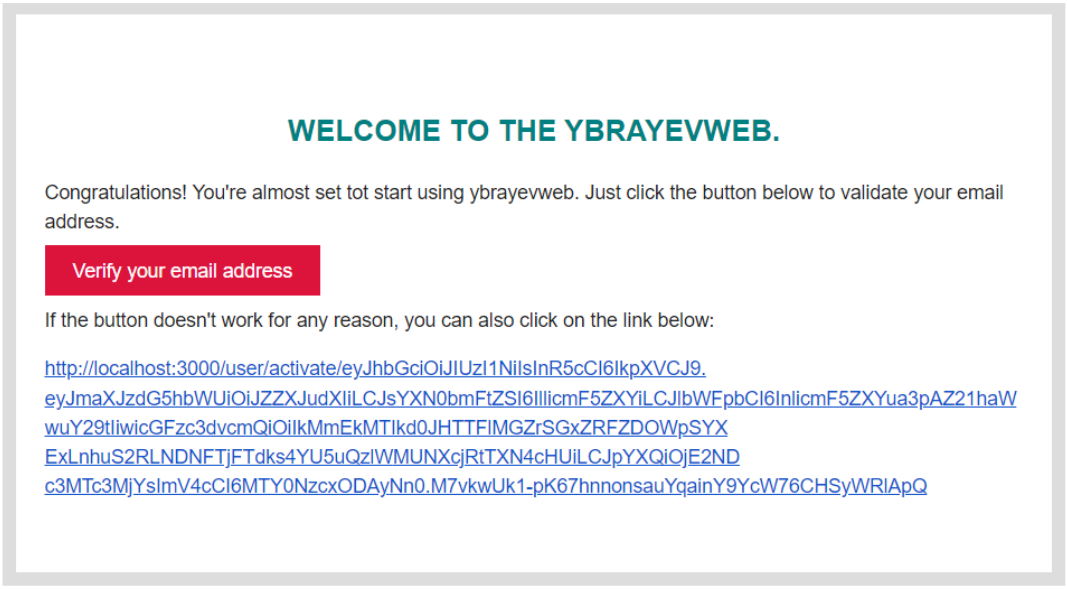
Жүйеге тіркелу барысында сізге толтыру керек (валидация жүйесінің логикасы көрсетілген):

1. Атыңыз – 2 сиволдан кем болмауы керек;
2. Тегіңіз (фамилия) – 2 символдан кем болмауы керек;
3. Электронды поштаңыз – осыған дейін бұл қосымшада тіркелмеген болуы керек, поштаны толтыру кезінде “@” символы болуы керек;
4. Құпиясөз – 8 символдан кем болмауы керек;
5. Құпиясөз (растау) – бірінші толтырған құпиясөзбен бірдей толтырылуы керек;



2.6.2.1-сурет – «Жүйеге тіркелу (регистрация) парақшасы»

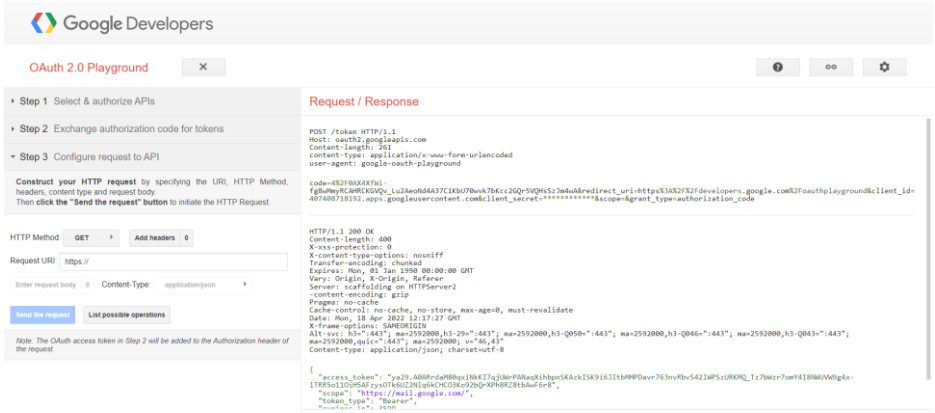
Барлық ақпарат сәтті қабылданған сәтте сіздің еңгізген электронды поштаңызға растауды сұрайтын хабарлама (активация) келіп түседі. Сіз сол хабарламадағы батырманы баса немесе сілтемені баса арқылы растай аласыз [15].



2.6.2.2-сурет – «Растауды сұрауға бағытталған хабарлама»

Осы хабарламадағы батырманы не болмаса сілтемені басқаннан кейін ғана сіздің аккаунтыңыз деректер қорына енгізіледі. Алайда осы процедурадан кейін жүйеге кіру керек болады. Жүйеге кіргеннен кейін сізге токен беріліп ары қарай жұмыс жасай бересіз.

Хабарламалармен жұмыс жасау үшін мен Google Developers және Google Mail сервистерімен жұмыс жасадым. Google Mail сервисін қолдану үшін мен Google Developers сервисі беретін токенді қосымшаға еңгіздім, алайда, Google Developers бұл токенді апта сайын жаңартып тұрады, сол үшін аса қатты мән берген жөн. (2.6.2.3-сурет)

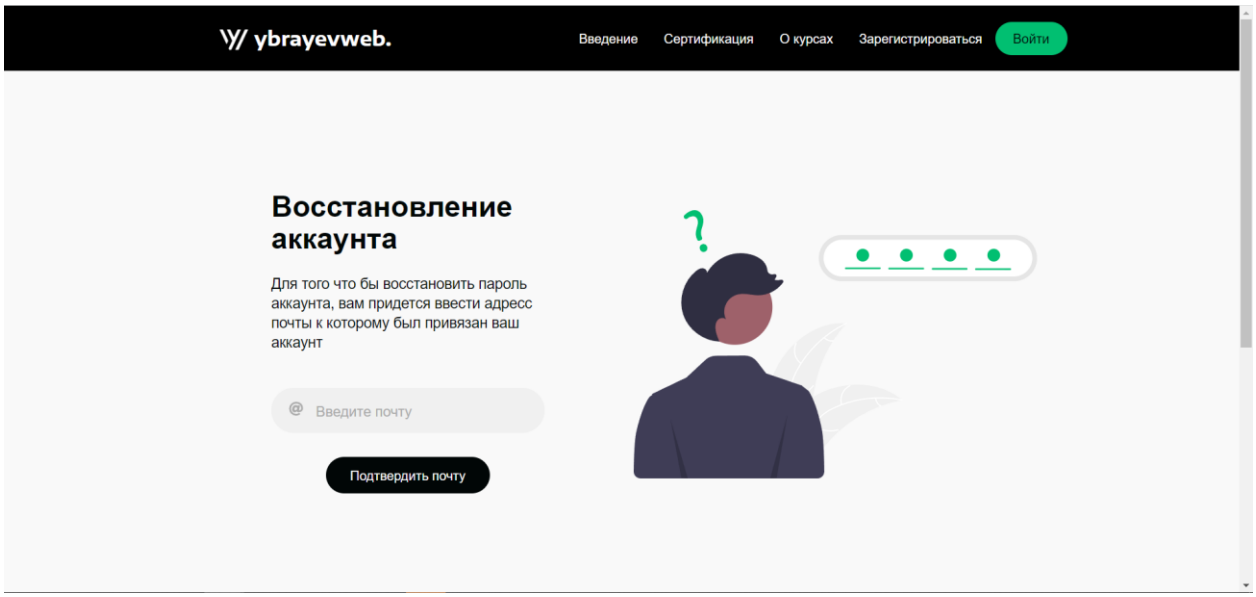


2.6.2.3-сурет – «Google Developers сервисінің баптаулары»

2.6.3 Құпиясөзді қалпына келтіру

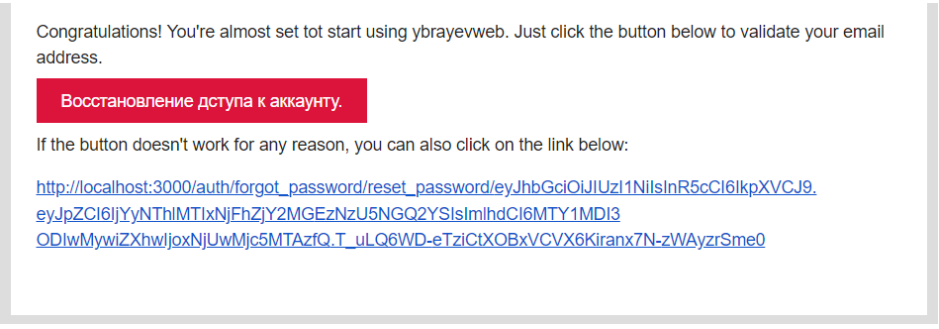
Ұмыту-адами фактор болып табылады, және құпиясөзді ұмыту қазіргі таңда жиі болып тұрады. Яғни қолданушылар өздерінің қойған құпиясөздерін ұмытып қалады. Әрине, біз күніне бірнеше социалды желілірмен жұмыс жасаймыз, әрқайсысына қандай құпиясөз қойғанымызды шатастырып та аламыз. Осындай жағдайдың алдын алу үшін құпиясөзді орнына келтіру функционалы, қосылды.

Жүйеге кіру парақшасында “Забыли пароль?” батырмасын басу арқылы құпиясөзді орнына келтіру функционалының орындалуын бастаймыз. Біз осы сілтемеге өткеннен кейін біз осы қосымшаға қай электронды поштамен тіркелдік, сол поштаны енгізу керекпіз. (2.6.3.1-сурет)



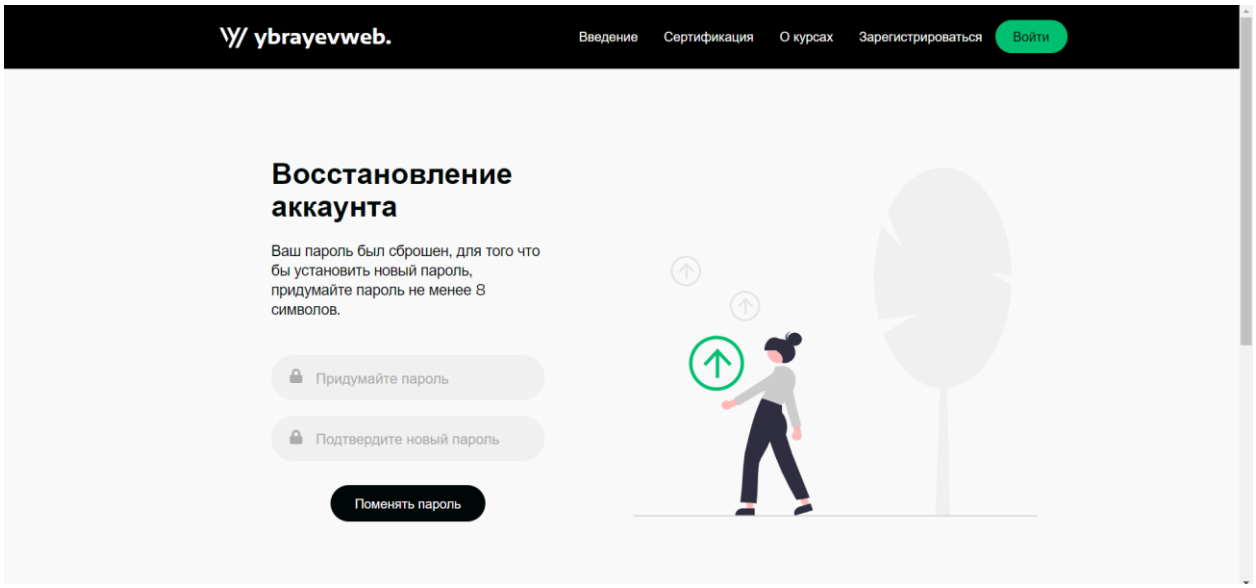
2.6.3.1-сурет – «Құпиясөзді орнына келтіру парақшасы»

Егер электронды поштаны еңгізу барысында бәрі дұрыс еңгізілсе қолданушының электронды поштасына құпиясөзді орнына келтіруге батырма мен сілтеме келеді. (2.6.3.2-сурет)



2.6.3.2-сурет – «Құпиясөзді орнына келтіру хабарламасы»

Сілтеме не батырманы басқаннан кейін сіз өзіңіздің аккаунтыңызға жаңа құпиясөз еңгізесіз. Құпиясөз 8 символдан кем болмауы керек, және құпиясөз дұрыс расталуы керек. (2.6.3.3-сурет)



2.6.3.3-сурет - «Жаңа құпиясөз қою операциясы парақшасы»

Жаңа құпиясөз еңгізгеннен кейін сол құпиясөзбен пайдаланушы өз аккаунтына кіріп ары қарай жұмыс жасай бере алады. Алайда осыған дейін жұмыс жасаған токен жойылып, басқа токен генерацияланады [16].

2.7 Mongo DB – деректер қоры

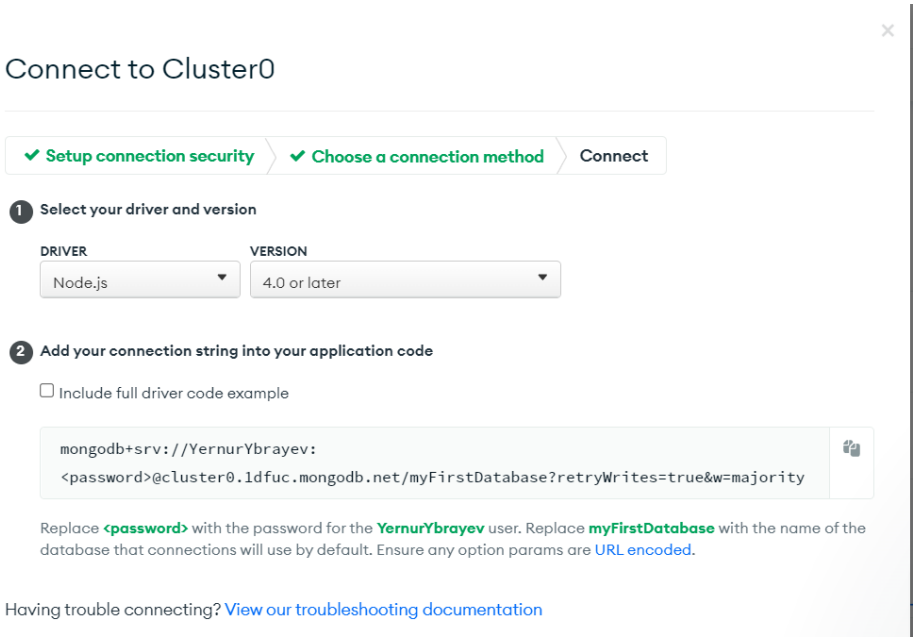
Mongo DB – кесте схемасының сипаттамасын қажет етпейтін құжатқа бағытталған деректер қорын басқару жүйесі болып табылады. Бұл деректер қорын көбінесе MEAN (MongoDB, Express.js, Angular.js, Node.js), MEVN (MongoDB, Express.js, Vue.js, Node.js) және MERN (MongoDB, Express.js, React.js, Node.js) стектерімен жұмыс жасайтын қосымшалар қолданады.

Бұл деректер қоры жүйесін қолданбастан бұрын оның артықшылықтар мен кемшіліктері сарапталған еді. Яғни, осы қосымшаның жұмысы жеңіл және жүйеде көп жадтың қолданылмауы қарастырылған еді. Барлық сараптамалардан кейін Mongo DB артықшылықтарына арнап тізім жасадым, және осы артықшылықтарға сүйене мен осы деректер қорын таңдадым. Mongo DB артықшылықтары:

- Схемалардың аздығы. Егер икемді (гибкий) схема кажет болса, Mongo DB өте қолайлы болып табылады.
- Масштабтандырудың қарапайымдылығы.

- Тегін болуы. Әрине егер үлкейту керек болса, ол тегін емес, бірақ бағалары салыстырмалы өте тиімді болып келеді.
- Бұлтты жүйе. Барлық деректер бұлтты жүйе арқылы жұмыс жасайды.

Mongo DB – қосымшаға өте жеңіл интеграцияланады. Бар болғаны жұмыс жасау кластерін жариялау болып табылады. Кластер жарияланғаннан кейін қосымшаға қосу үшін сілтеме генерециялау керек болады. Сілтеме “Connection → Connect your application” батырмасын баса арқылы алынады. (2.7.1-сурет)



2.7.1-сурет – «Mongo DB интеграциялау сілтемесі»

Бұл сілтемені өзгерте отыра біз кейбір параметрлерін өзгертіп ала аламыз. Үлгі ретінде осы сілтемені қарастырып көрейік:
mongodb+srv://YernurYbrayev:<password>@cluster0.1dfuc.mongodb.net/myFirstDatabase?retryWrites=true&w=majority

- YernurYbrayev – жүйедегі қолданушының аты;
- <password> - жүйеге кіру үшін қолданылатын құпиясөз;
- Cluster0 – жүйеде бірнеше кластер болу мүмкін, солардың ішіндегі қайсысы екенин идентификациялайды;
- myFirstDatabase – кластердің ішінде бірнеше деректер қоры болуы мүмкін, алайда соның қайсысымен жұмыс жасау керек екенін идентификациялайды;

Сілтеме Cloudinary сервисімен жұмыс жасаған секілді конфигурациялық файлға жазылуы керек. Біздің жағдайда “.env” файлы болып табылады.

Сілтемені еңгізгіннен кейін “Mongoose” пакеті арқылы қосымшамен байланыс жасаймыз [17].

Үстіде айта кеткеніміздей деректер Mongo DB жүйесінде JSON форматы секілді сақталады. Осыған орай оны оқу және түсіну бізге ешқандай қиындық көрсетпеуі тиіс. (2.7.2-сурет)

```
_id: ObjectId("620c959eac85cd75c1f228e5")
firstname: "Yernur"
lastname: "Ybrayev"
email: "ybrayev.kz@gmail.com"
password: "$2a$12$RUo2kCxoAfsnmoS20hqpZeT1CT3HQgxnxglNxTQVUsRR.kepMH8/y"
__v: 0
```


2.7.2-сурет – «Mongo DB деректер қорында деректердің сақталу форматы»

2.8 Cloudinary – сурет жүктеу сервисі

Бұл қосымшадағы барлық суреттер және бейнежазбалар осы Cloudinary сервисі арқылы рендер жасап тұр. Яғни Cloudinary бұл суреттер мен бейнежазбалардың сақталатын қоры болып табылады.

Cloudinary сервисінің мүмкіншіліктері:

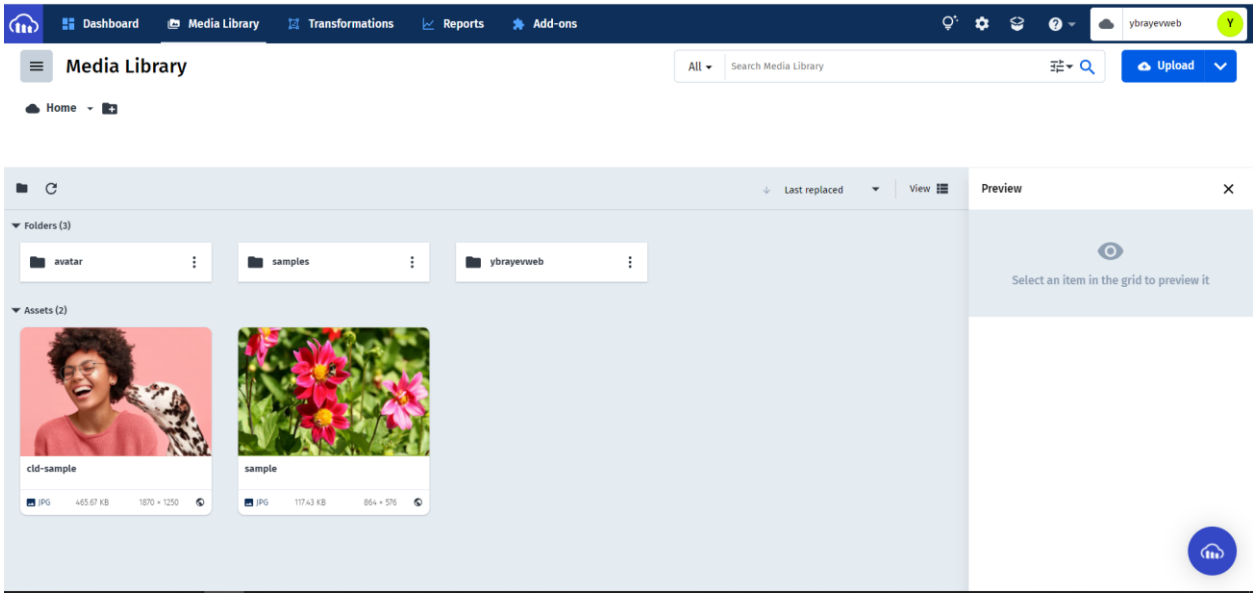
- Суреттер мен бейнежазбаларды сақтау;
- Суреттер мен бейнажазбаларды компрессия жасайды;
- Суреттер мен бейнежазбаларды басқа форматтарға форматтайды;
- Суреттер мен бейнежазбаларға өзгерістер енгізе алады;
- Суреттерге тіпті эффектілер мен белгілер қоса алады;
- Суреттердің өлшемін өзгерте алады;
- Суреттердің пішінін өзгерте алады;
- Және де қай программала тілімен жазып жатқаныңызға байланысты Cloudinary url, https, Java, PHP, Javascript, Node.js және басқа да тілдерге генерация жасайды;

“Суретті бұл жерден қалай алуға болады?” – деген сұрақ туыдауы мүмкін, жәй ғана url, https немесе кодын суретке баса арқылы көшіріп қосымшаның керек жеріне сақтап аламыз.

Тағы бір үлкен артықшылығы суретке не болмаса бейнежазбаға қандай да бір өзгерістер енетін болса парақшаны жаңартпай-ақ осы өзгерістерді көре алса болады.

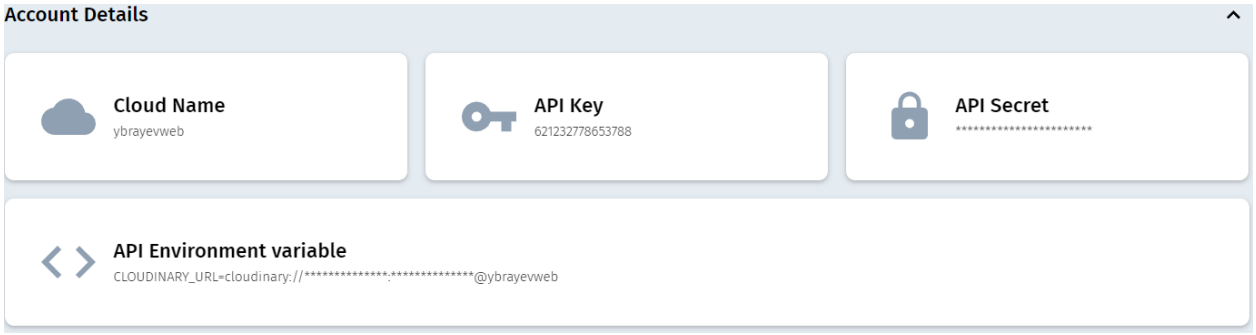
Сервис өте ыңғайлы болғанмен оның шектеуі бар. Өкінішке орай ақысыз қолдана алатын тек 2 ГБ бос орын, ол бізде орташа алғанда 75000 сурет болады. Ары қарай трафикті үлкейту 44$ басталады.

Cloudinary сервисінің интерфейсі өте ыңғайлы және түсінуге өте жеңіл болып келеді. (2.8.1-сурет)



2.8.1-сурет – «Cloudinary сервисінің интерфейсі»

Айта кететін тағы бір басты мәлесе ол интеграция, яғни қалай бұл сервисті біз өз қосымшамызға жалғаймыз? Бізге бар болғаны Cloud Name, API Key, API Secret, API Environment variable (2.8.2-сурет) мәндерін қосымшаның конфигурациялық файлына енгізу керек. Біздің жағдайда конфигурациялық файлдың аты “.env” деп аталады. (2.8.3-сурет)



2.8.2-сурет – «Cloudinary сервисінің интеграция жасалатын баптаулары»

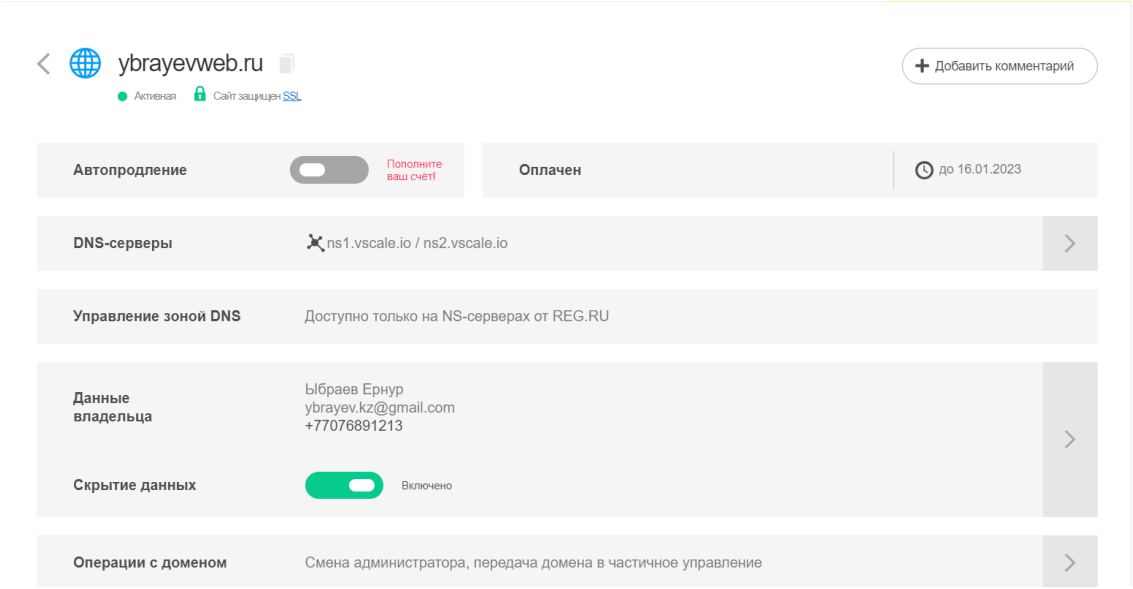
```
CLOUD_NAME = ybrayevweb
CLOUD_API_KEY = 621232778653788
CLOUD_API_SECRET = ku9HhpGp19BsSDD6Z7CU5OfbYNY
```

2.8.3-сурет – «“.env” конфигурациялық файлы»

Барлық сілтемелер мен баптаулар жасалғаннан кейін бұл сервистің баптаулары мен сервиске тіркелу кезіндегі ақпаратты ешкім білмеуі шарт [18].

2.9 Хостинг және домен

Бұл қосымшаны интернет желісіне салу үшін бізге керегі домен және хостинг. Домендік есімді көптеген сервистерден алуға болады, алайда менің жағдайымда “2domains.ru” сайтынан алынған доменмен жұмыс жасаймыз. Бұл сайттан алған себебім, ол осыған дейін бұл сервистен алған домендердің дұрыс жұмыс жасауымен және бағаларының өте тиімді болуымен сипатталады. (2.9.1-сурет)



2.9.1 – сурет – «Доменнің баптаулары»

2.9.1-суретке қарап біз домендік есімнің “ybrayevweb.ru” екенін айта аламыз, және бұл сайт SSL сертификаты бойынша шифрлену қауіпсіздігімен қамтамасыз етілгенін көре аламыз. Яғни біздің қосымшаның деректерді жіберу/алу HTTPS протоколы арқылы жұмыс жасайды.

Хостинг ретінде мен “VDS by Selectel (vds.selectel.ru)” сервисін таңдадым. Бұл сервисті таңдаудың бірнеше себептері бар. Солардың арасындағы ең үлкен артықшылықтары оның тиімділігі және SSD-дискісімен қамтамасыз етілген серверлер.

Хостинг ретінде мен жай ғана сервер емес VPS-серверді таңдадым. Бұндай серверді таңдаудың бірнеше себебі бар, олар:

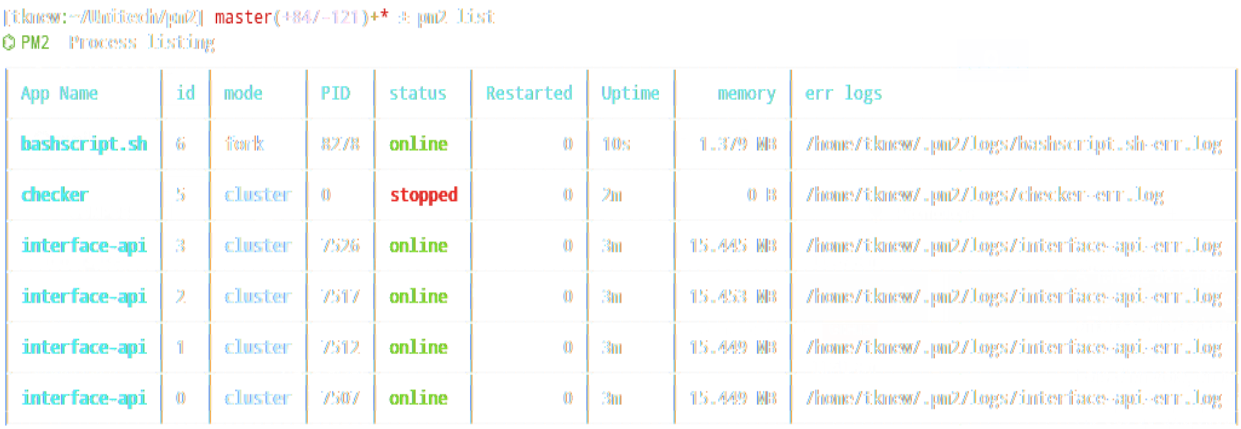
- максималды түрде сервердің баптауларын өзгертуге мүмкіншілік;
- сервердің виртуалдылығы;
- сервердің қуаттылығы;

Бұл сервистің мен ең арзан деген пакетін сатып алдым, оның ішіне 1 гб жедел сақтау жады, 1 ядролы процессор және 30 гб SSD дискісінің жады кіреді.

Қосымша ең бірінші “github.com” репозиториялар жүйесіне 1 репозитория болып жүктеледі, содан кейін, “PuTTY” құралы арқылы мен бұл серверге

қашықтықтан өзгерістер енгіземін. Командалар арқылы мен бұл серверге “Git” орнатып, қосымшаның репозиторияларын осы серверге көшіріп аламын. Осы процедурадан кейін мен қосымшаның пакеттерін осы серверге белгілі бір командалармен орнатамын. Бәрі орнатылғаннан кейін қосымшаның жұмысын белгілі бір командалармен іске асырамын.

Сервер жүйесінен мен шығып кеткен жағдайда қосымша жұмысын тоқтатады. Алайда біз қосымшаның жұмысы тоқтамауы керек болғаннан кейін “pm2” пакетін орнатып, қосымшаның жұмыс жасауын осы пакеттің жұмысына еңгіземін (2.9.2-сурет)



App Name	id	mode	PID	status	Restarted	Uptime	memory	err logs
bashscript.sh	6	fork	8278	online	0	10s	1.379 MB	/home/tknew/.pm2/logs/bashscript.sh-err.log
checker	5	cluster	0	stopped	0	2m	0 B	/home/tknew/.pm2/logs/checker-err.log
interface-api	3	cluster	7526	online	0	3m	15.445 MB	/home/tknew/.pm2/logs/interface-api-err.log
interface-api	2	cluster	7517	online	0	3m	15.453 MB	/home/tknew/.pm2/logs/interface-api-err.log
interface-api	1	cluster	7512	online	0	3m	15.449 MB	/home/tknew/.pm2/logs/interface-api-err.log
interface-api	0	cluster	7507	online	0	3m	15.449 MB	/home/tknew/.pm2/logs/interface-api-err.log

2.9.2-сурет – «pm2 пакетінің жұмыс жасау кестесі»

2.9.2-суретке қарайтын болсақ виртуалды машинасында орнатылған pm2 пакеті бірнеше API және қосымшалардың жұмысын жасап тұр, және осыдан алатын тұжырымдама, бір VPS-сервер бірнеше қосымшаның жұмысын атқара алатыны.

Үстінде айта кеткендей қосымшаның серверлік бөлімі және клиенттік бөлімі бір-біріне тәуелді емес, яғни олар екі түрлі және бөлек қосылады. Алайда, осы бір қиындықты шешу мақсатында “concurrently” пакетін орнаттым. Оның маған беретін пайдасы – сервер және клиенті бір уақытта және бір командамен қоса немесе өшіре алады [19].

2.10 Қауіпсіздік шаралары

Технологиялардың дамуының бізге пайдасы зор болуы мүмкін, алайда, бұның да бір кемшіліктері бар екенін ескерейік. Қазіргі таңда информациялық заман болғаннан кейін информациялық соғыс, информациялық шабуылдар әлем бойынша көп орын алуда. Осыған орай қосымшаның жұмысы барысында ең бірінші қауіпсіздік – деректер қоры деп білемін.

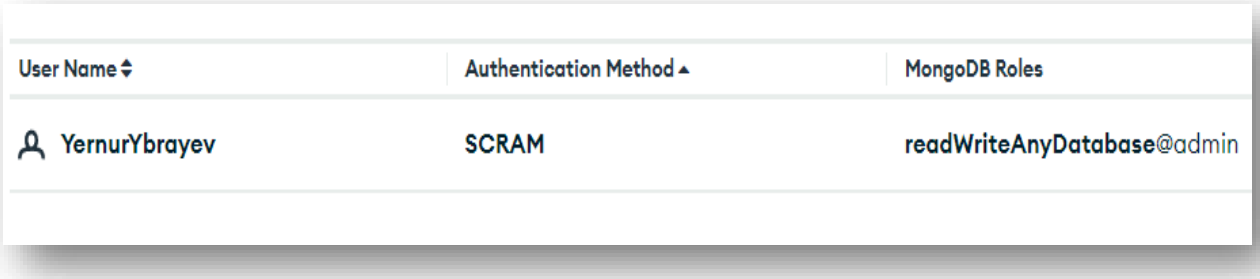
Деректер қоры – басқалар біліп-білмеуге арналған ақпаратты сақтайды, бірақ ол ақпаратты кім және қалай қолданатыны белгісіз. Біздің қолымызда ол

жәй ғана дерек болғанмен, басқа біреудің қолына қару болып баруы әбден мүмкін. Осы проблеманың алдын алу үшін бірнеше сатыдан тұратын қауіпсіздік баптаулары қосылған еді. Олардың тізімі:

1. “bcrypt.js” пакеті (2.10.1-сурет) – қолданушылардың қосымшаға тіркелгенде қойған құпиясөзін хештеу әдісімен шифрлау үшін қолданылды. Яғни, деректер қоры, зиянкестердің қолына түскен жағдайда құпиясөзді олар дешифрлау әдісінен өткізбей, біле алмайды.

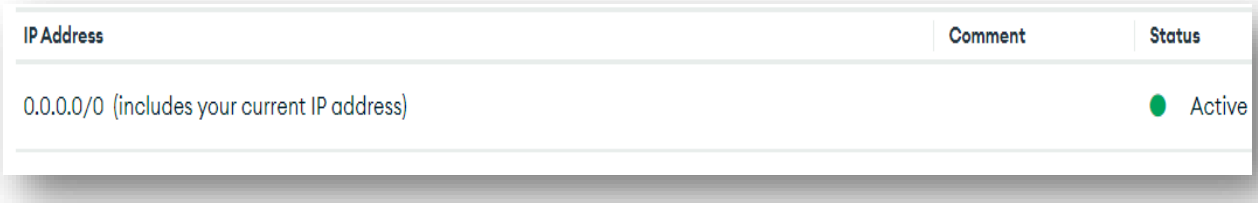
2.10.1-сурет – «“Bcrypt.js” пакеті туралы анықтама сурет»

2. Mongo DB кластерлеріндегі қолданушылар жүйесі – деректер қорына кіре алатын қолданушаларды ғана қамтиды. Кластерді жариялаған кезде баптаулары арқылы кім деректерді көре алатынын, кім өзгерте алатынын өзгерту мүмкіншілігі. (2.10.2-сурет)



2.10.2-сурет – «Қолданушылар жүйесі арқылы қауіпсіздендіру»

3. Mongo DB кластерлеріндегі желі бойынша қауіпсіздендіру жүйесі TCP/IP протоколы бойынша қауіпсіздендіреді. Яғни, деректер қорына кіре алатын тек қана бір немесе одан көп IP-адрестер тізімі болады, алайда, тізімге кірмеген қолданушылар, деректер қорына кіре алмайды. Егер де осы тізімнен бөлек, деректер қорына кіру жағдайы орын алса, қосымшаны әзірлеуші маманның электронды поштасына хабарландыру келеді. (2.10.3-сурет)



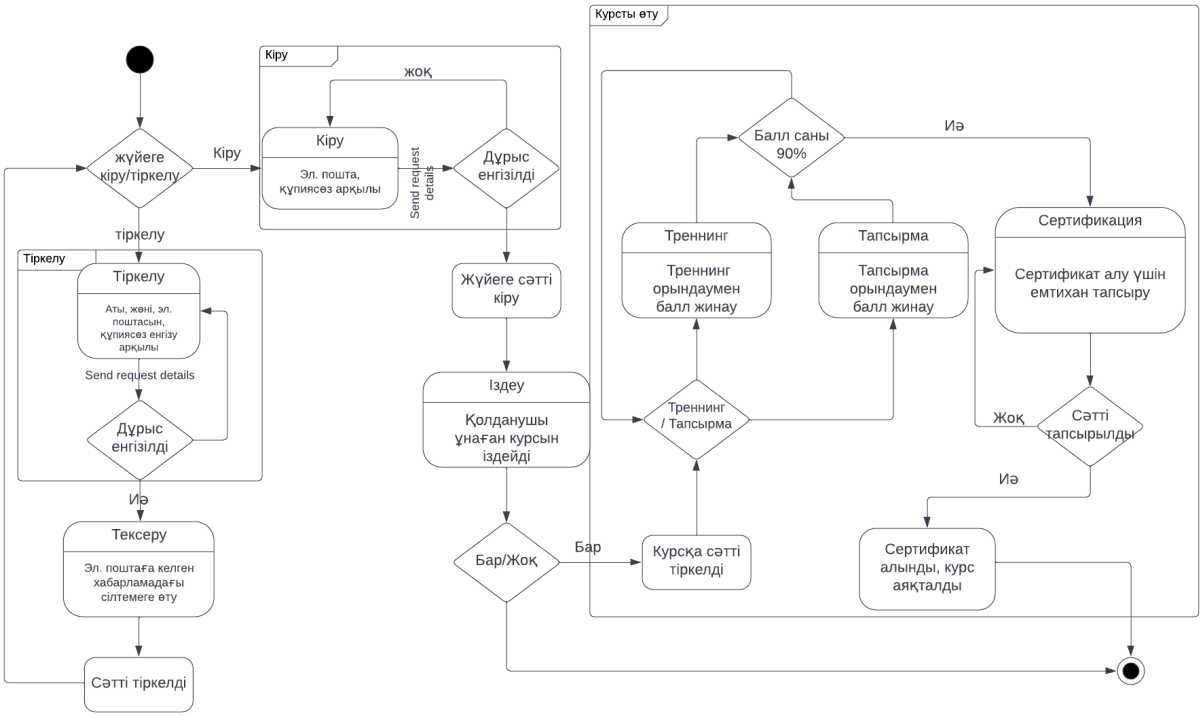
2.10.3-сурет – «Желі бойынша қауіпсіздендіру»

3 Жобалау бөлімі

3.1 Веб-қосымшаның логикалық құрылымы

Барлық технологиялық құрылымдармен танысқаннан кейін қолданушылардың қосымшамен қалай жұмыс жасайтыны және жүйеге кіргеннен кейін әр қолданушының рөлін, іс-әрекеттер тізімін осы бөлімде қарастырамыз. Алайда, бұндай сипаттамаларды диаграмма ретінде қарастырған көзге жағымды, оңай болып табылады.

Бастапқыда UML диаграммасына көңіл аударайық:



3.1.1-сурет – «Жобаның UML диаграммасы»

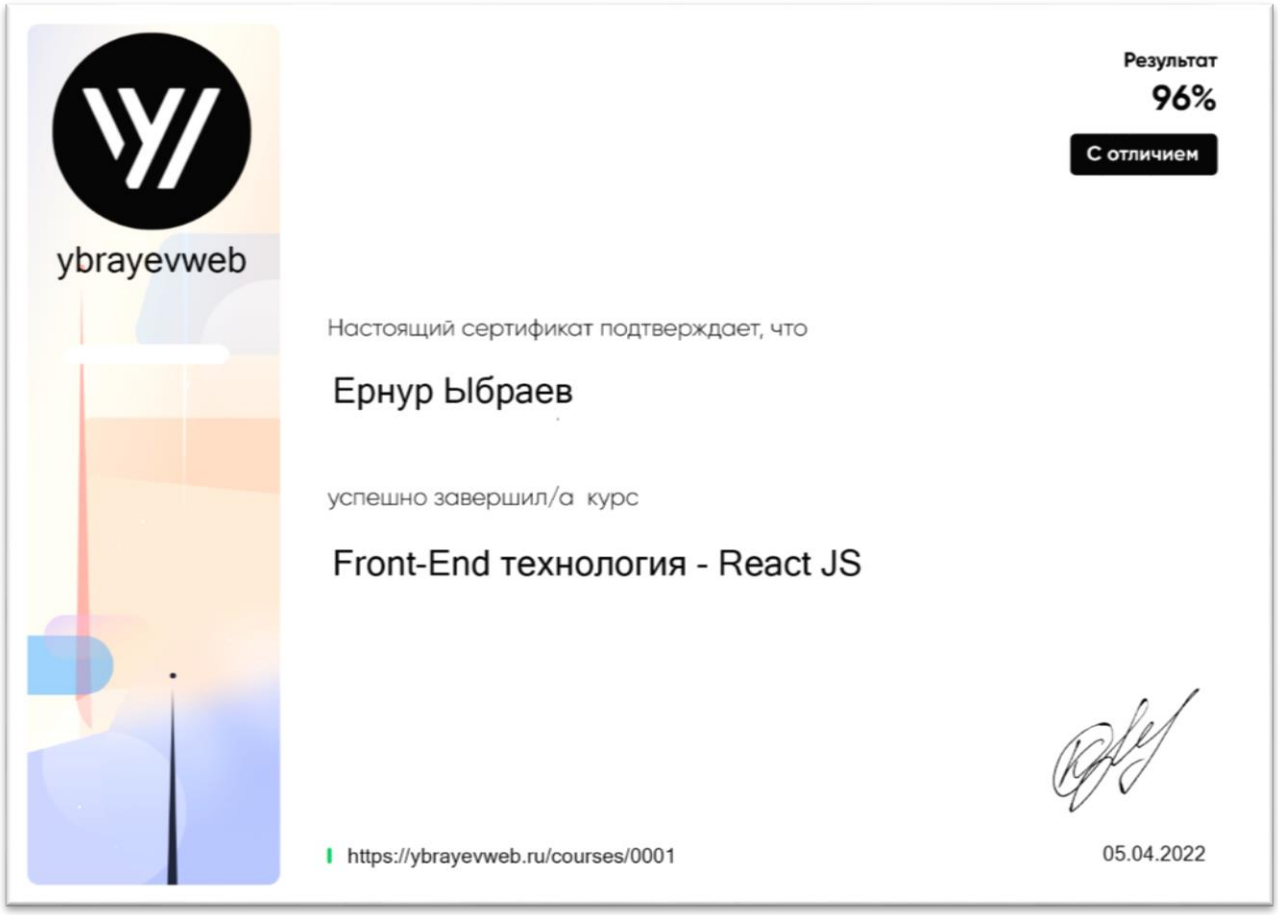
Диаграммаға қарайтын болсақ біздің бастамамыз ешқандай аутентификациядан өтпеген қолданушы күйінен басталады. Яғни диаграмманың бізге айтып тұрғаны – жүйеге кірмеген қолданушы біздің қосымшамызды өз мақсатында қолдана алмайды.

Бұған дейін қолданушы егер осы қосымшаны қолданбаған болса ол басты бетке өтеді. “Регистрация” батырмасы бойынша тіркеуден өткеннен кейін жүйеге “Войти” батырмасы бойынша керек ақпаратты енгізіп жүйеге кіреді.

Жүйедегі қолданушы қосымшамызбен толықтай қолдана алады, яғни, ол кусті іздейді, сол курсқа тіркеле алады, сол курс бойынша треннинг өті алады, сол курсқа тиесілі барлық жаттығулары орындай алады [20].

Курсты толықтай бітірген қолданушы, қосымша тарапынан сертификат ала алады. Алайда, сертификатты алу үшін курс бойынша максималды баллдың кемінде 90% алу керек деп қарастырылды. Қолданушы қалай 90%-ға жетед сол сәтте экзамен тапсыруға мүмкіншілік алады.

Сертификация тест және емтихан форматында іске асады, әр курстың минималды алу керек ұпайы, курстың қиындығына байланысты. (3.1.2-сурет)



3.1.2-сурет – «Сертификат форматы (мысал) »

3.2 Жүйенің логикалық құрылымы

Қосымшаның негізі 2 рөлі бар, олар: “Адинистратор” және “Жәй қолданушы”. Жәй қолданушының барлық мүмкін деген іс-әрекеттері 3.1-бөдімінде айтылды. Алайда администратордың іс-әрекеттері жәй қолданушының іс-әрекеттерінен мынадай артықшылықтармен ерекшеленеді: (3.2.1-сурет)

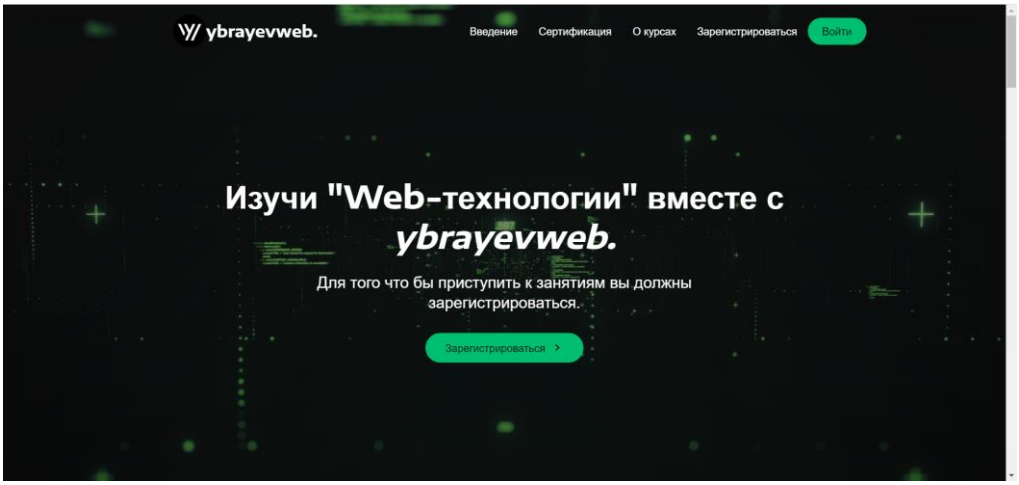
1. Жаңа курс еңгізу;
2. Белгілі бір курсқа өзгертулер еңгізу;
3. Курсты жою;

4. Жаңа тапсырмалар еңгізу;
5. Тапсырмаларды өзгерту;
6. Тапсырмаларды жою;
7. Емтихан сұрақтарын өзгерту;
8. Сертификатты өзгерту;

3.2.1-сурет – «Жобаның UseCase диаграммасы»

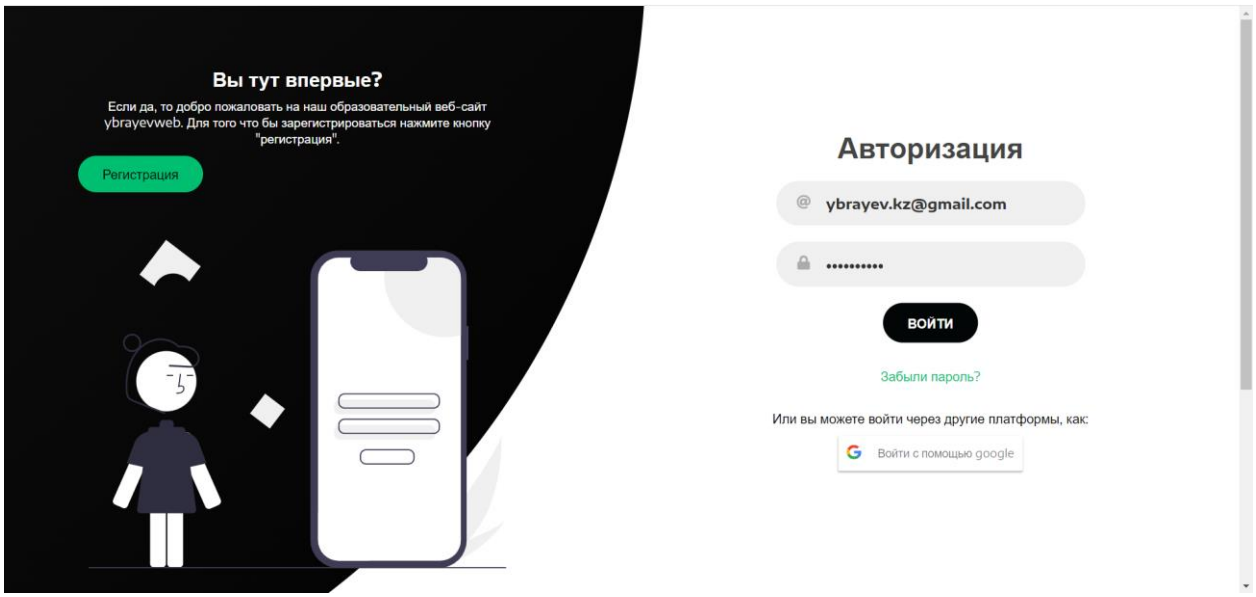
3.3 Қолданушы интерфейсі

“ybrayevweb.” қосымшасына кіргенде қолданушы басты бетті (3.3.1-сурет) көреді. Басты бетте қосымша туралы ақпарат және курстар туралы қысқаша ақпарат ала алады. Басты бетте қосымшаның социалды желілерге сілтемелері, жобаның авторы туралы ақпарат сілтемесі, келісім, шарт туралы сілтемелер және администраторлармен байланыс сілтемелері арқылы толық ақпарат парақшаларына өте алады. Тіркелу және жүйеге кіру үшін батырмалар еңгізілген.

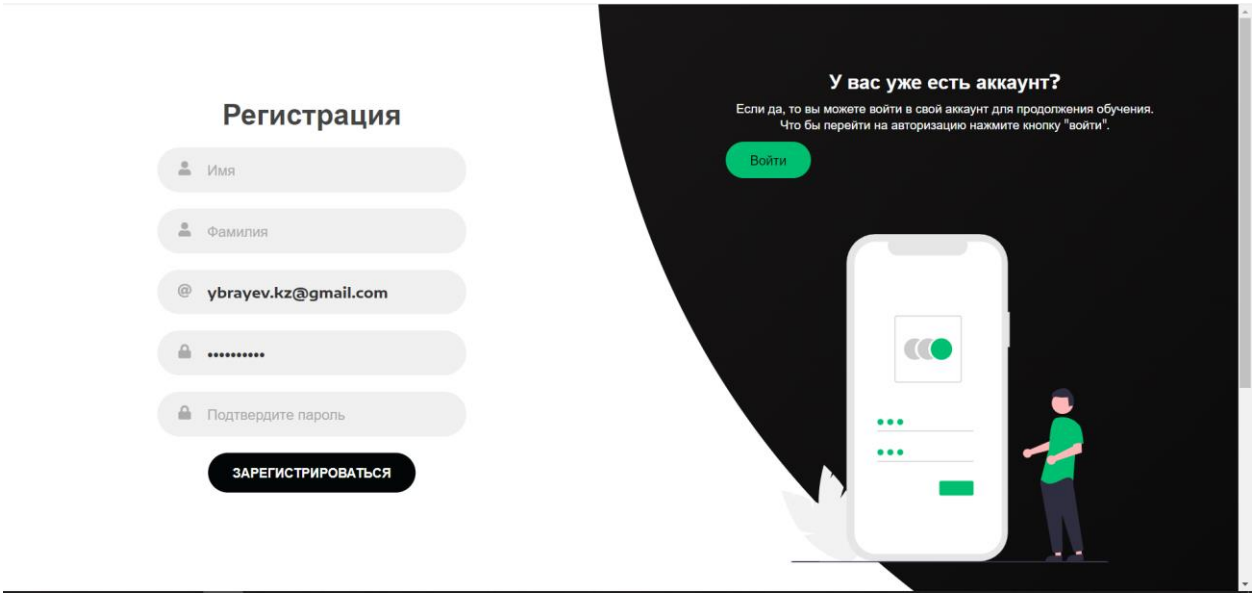


3.3.1-сурет – «Қосымшаның басты беті»

Жүйеге кіру (3.3.2-сурет) және тіркелу (3.3.3-сурет) парақшалары анимацияланған дизайн ретіне іске асты.



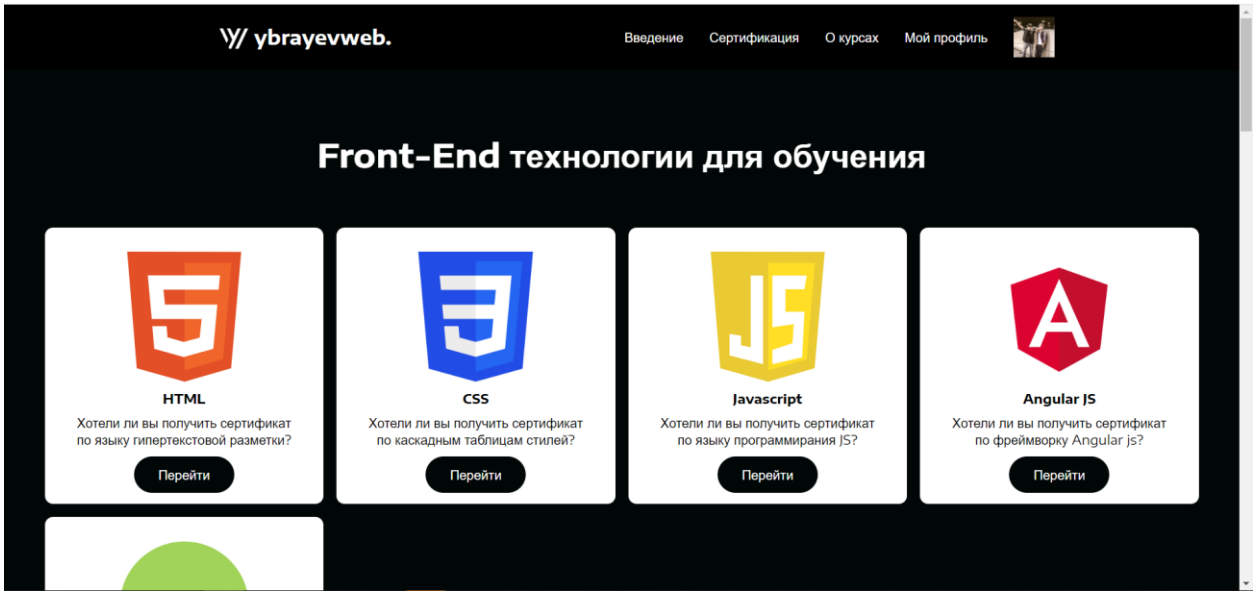
3.3.2-сурет – «Жүйеге кіру парақшасы»



3.3.3-сурет – «Жүйеге тіркелу парақшасы»

Курстар негізі іздемей-ақ тізім болып шығады, алайда, қосымшаға тағы курстар енетін болса қолданушылар іздеп қалмас үшін іздеу функциясы қосылды.

Қосымша интерфейсі барынша қарапайым және жеңіл жасалған. Керек курсты тапқаннан кейін “Перейти” батырмасы арқылы курсты ашамыз (3.3.4-сурет).



3.3.4-сурет – «Курстар тізімі»

ҚОРЫТЫНДЫ

Берілген дипломдық жобада заманауи стандартқа сай, барынша ыңғайлы, әрі қарапайым дизайн ұстана отырып “ybrayevweb.” атты қосымша әзірленді. Әзірленген жүйе қолданушылардың қызығушылығын оятуға, программалау бағыты бойынша сауатын ашуға, және оны дамыту процесін жеңілдету мақсатымен әзірленді.

Бәріміз білетіндей программалау тілдері, құралдары күннен күнге даму үстінде, және олардың бәрін бір кітап қылып, бір бөлім қылып қарастыру өте қиын. Соны ескере тұра барынша бөліп қарастыру дұрыс деп тұжырымдалды.

Әзірлеу барысында қолданылған технологиялар тізімі: бэкенд бөліміне Express.js фреймфоркі және Node.js серверлік бағдарлама, фронтенд бөліміне React.js фреймворкі Javascript тілі негізінде, деректер қоры ретінде реляционды емес Mongo DB қолданылды.

Қосымшаны әлі де дамыту жоспарлануда, яғни тек бір тілде ғана программалаудан бөлек стек қолдана қосымшалар жасау, деректер қорын жобаға интеграциялау, баптауларын оңдау секіліді инструкциялар мен есте қалу мақсатында тапсырамалар қосу жоспарлануда.

Әр курсты бағалау, әр тарауын басқа да қолданушылармен комментарий жазып талқылау қарастырылды, веб-парақшаның заманауи дизайнын үйретететін курстар, видео курстар жоспарлануда. Әр курстың оффлайн форматындағы кітаптары мен материалдарын жүктеу мүмкіншілігі қарастырылуда.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1 Node.js v16.14.2 құжаттамасы // Электрондық нұсқасы https://nodejs.org/dist/latest-v16.x/docs/api/

2 Mongo DB бойынша онлайн әдістеме // Электрондық нұсқасы https://metanit.com/nosql/mongodb/

3 Mongo DB құжаттамасы // Электродық нұсқасы https://www.mongodb.com/docs/manual/tutorial/getting-started/

4 Express.js бойынша құжаттама // Электрондық нұсқасы https://expressjs.com/ru/

5 React бойынша құжаттама // Электрондық нұсқасы https://ru.reactjs.org/docs/getting-started.html

6 Как создать адаптивный сайт с помощью ReactJS // Электрондық нұсқасы https://webformyself.com/adaptivnye-veb-stranicy-s-pomoshhyu-react-response-i-typescript/

7 Email.js пакеті бойынша құжаттама // Электрондық нұсқасы https://www.emailjs.com/docs/

8 React Hooks простыми словами // Электрондық нұсқа https://habr.com/ru/company/simbirsoft/blog/652321/

9 Node.js файлындағы сеанс cookie аутентификациясы (толық мысалдармен) // Электрондық нұсқа https://www.sohamkamani.com/nodejs/session-cookie-authentication/

10 Web-технологии // Электрондық нұсқа https://www.w3schools.com/

11 Бэн Фрейн, книга «Отзывчивый дизайн на HTML5 и CSS3 для любых устройств. 3-е изд.» // Электрондық нұсқа https://habr.com/ru/company/piter/blog/598789/

12 Бэнкс Алекс, Порселло Ева, «React и Redux. Функциональная вебразработка» // Электрондық нұсқа https://www.ozon.ru/product/react-i-redux-funktsionalnaya-vebrazrabotka-react-i-redux-funktsionalnaya-veb-razrabotka-143282355/?sh=TMKUqeROrQ

13 How To Use Styled-Components In React // Электрондық нұсқа https://www.smashingmagazine.com/2020/07/styled-components-react/

14 Что такое хуки и как их использовать: краткий гайд с примерами // Электрондық нұсқа https://highload.today/chto-takoe-huki-i-kak-ih-ispolzovat/

15 Маршрутизация // Электрондық нұсқа https://metanit.com/web/react/4.1.php

16 Learn React toastify with example // Электрондық нұсқа https://www.cloudhadoop.com/reactjs-toastify-example/

17 Установить базу данных MongoDB в Windows // Электрондық нұсқа https://betacode.net/10265/install-mongodb-database-on-windows

18 Кластер MongoDB // Электрондық нұсқа https://elma365.com/ru/help/configure-mongodb.html

19 Build a REST API with Node.js: Routes and Controllers // Электрондық нұсқа https://lo-victoria.com/build-a-rest-api-with-nodejs-routes-and-controllers

20 Creating configuration files in Node.js using node-config // Электрондық нұсқа https://blog.logrocket.com/creating-configuration-files-node-js-using-node-config/

А Қосымшасы
(міндетті)

“ybrayevweb.” веб-қосымшасын құруға арналған
техникалық тапсырма

А.1 Кіріспе

Бүгінді таңда, көптеген технологиялардың қарқынды даму кезеңінде, сол технологиялардың тілін түсінетін, жұмыс жасауын қадағалайтын және жаңа жобалармен айналысатын маман иелері судай керек. Соның ішінде ең керектілерінің бірі веб-әзірлеушілер. Бұны түсінген қоғам тіпті өз мамандықтарын осы веб-саласына алмастыруда.

Веб-технологияларды үйрену айтарлықтай оңай емес, әрине, көптеген қиыншылықтармен кездесеміз, кейбір жаңа технологиялар бойынша ақпарат табу, оны қолдану құжаттарын түсіну айтарлықтай қиындық туғызады. Бұндай жағдайда репетитор не болмаса курс іздейтіндер де көп, бірақ қазіргі таңда салыстырмалы түрде қарасақ, веб-технологияларды үйрену тым қымбат болуы мүмкін. Осы мәселенің алдын алу мақсатында бұл дипломдық жұмысымды, яғни “ybrayevweb.” қосымшасын шешім ретінде ұсынамын.

“ybrayevweb.” веб-қосымшасы негізінен веб-программалау саласына құнығушыларға немесе енді үйрене бастаған қызығушылар мен студенттер үшін арналған. Курстардың барлығы тегін, қолданушы өзіне ұнаған курс бойынша ақпарат алып, тапсырмаларды орындап, сертификат алуға емтихан тапсыра алады.

Сертификат – қызығушының еңбегін бағалау мақсатында жоба тарапынан сыйлық болып табылады.

А.1.2 Қолдану саласы

Қолдану саласы – кез-келген интернетке қосулы қолданушы. Қосымшаны кез-келген браузердің өзекті соңғы 5 версиясында қолдану ұсынылады. Қосымшаға өту үшін www.ybrayevweb.ru домендік атымен браузер арқылы кіру қажет.

А.1.3 Қосымшаның бейімділігі (Адаптация)

“ybrayevweb.” веб-қосымшасы тек қана Windows ОЖ-де ғана емес, барлық браузер ашылатын құрылғылырға, смартфон, планшет секілді техникаға бейімделген.

А.2 Жалпы сипаттамасы

Жалпы веб-қосымша VPS бұлттық серверде орналасқан. Яғни хостинг ретінде виртаулды сервер қолданылған, ал деректер қоры Mongo DB жүйесінін бұлттық қорында орналасқан. Node.js серверлік бағдарламасы деректер қоры және клиенттік бөлімді өзара байланыстырып араларындағы сұраныстарды орындайды.

А.2.1 Пайдаланушы интерфейстер

Веб-қосымша барынша қарапайым интерфейс тәнін ұстана жасалды, яғни қолданушыға барлық қол жетімді ыңғайлы интерфейс әзірленді.

А.2.2 Аппараттық интерфейстер

Құрылғыға қойылатын жалпы талаптар:
- кез-келген браузердің өзекті соңғы 5 версиясы;
- ғаламторға қол жетімділік;
- ақпарат енгізу және басқару құрылғылары;

А.2.3 Программалық интерфейстер

- Web Storm – бастапқы кодты өзгертетін редактор;
- Node.js – серверлік бағдарламасы;
- Postman бағдарламасы – REST сұранымды жүргізу/тексеру бағдарламасы;
- Mongo DB Compass – Mongo DB деректер қорындағы тиесілі кластермен жұмыс, және оны бақылау бағдарламасы;
- GitHub Desktop – нұсқаларды басқару, репозиторияны өзгерту бағдарламасы;

- PuTTY бағдарламасы – виртуалды серверге консольдік қосылуды қамтамасыз ететін бағдарлама;

А.2.4 Коммуникациялық интерфейстер

Қолданушылар орташа не жоғарғы жылдамдықтағы интернет желісімен қамтамасыз етілуі шарт. Сервер және клиенттерді байланыстыратын TCP/IP протоколы болып табылады.

А.2.5 Деректер бойынша шектеулер

Деректер қоры бұлттық жүйеде орналасқаннан кейін бірқатар шектеулер бар. Ашып айтқанда, бұлттық сервистен берілген орын тек 5GB-ты құрайды. Алайда, деректер қорында ешқандай мультимедиялық дерек сақталмауы бізге біршама деректі сақтауға мүмкіншілік береді.

Б Қосымшасы
(міндетті)

Бағдарлама мәтіні

1. Серверлік бөлімнің бас файлы app.js мәтіні

```
require('dotenv').config()
const express = require('express')
const mongoose = require('mongoose')
const cors = require('cors')
const cookieParser = require('cookie-parser')
const fileUpload = require('express-fileupload')
const app = express()
app.use(express.json())
app.use(cors())
app.use(cookieParser())
app.use(fileUpload({
   useTempFiles: true
}))
app.use('/user', require('./routes/user-routes'))
app.use('/api', require('./routes/upload'))
const DB_URL = process.env.MONGODB_URL
mongoose.connect(DB_URL, {
   useNewUrlParser: true,
   useUnifiedTopology: true
}, err => {
   if(err) throw err;
   console.log("Connected to mongodb")
})
const PORT = process.env.PORT || 5000
app.listen(PORT, () => {
   console.log('Server is running on port', PORT)})
```

2. Қолданушының барлық іс-әрекетін, сұраныстарын орындайтын бағыттар файлы user-routes.js мәтіні

```
const router = require('express').Router()
const userController = require('../controllers/user-controller')
const auth = require('../middlewares/auth')
const authAdmin = require('../middlewares/auth-admin')
router.post('/register', userController.register)
router.post('/activation', userController.activateEmail)
router.post('/login', userController.login)
```

```
router.post('/refresh_token', userController.getAccessToken)
router.post('/forgot', userController.forgotPassword)
router.post('/reset', auth, userController.resetPassword)
router.get('/userinfo', auth, userController.getUserInfo)
router.get('/allusersinfo', auth, authAdmin, userController.getUsersAllInfo)
router.get('/logout', userController.logout)
router.patch('/update', auth, userController.updateUser)
router.patch('/update_role/:id', auth, authAdmin,
userController.updateUsersRole)
router.delete('/delete/:id', auth, authAdmin, userController.deleteUser)
router.post('/google_login', userController.googleLogin)
module.exports = router
```

3. Қолданушының сұраныс методтары, қолданушы контроллері user-controller.js мәтіні

```
const Users = require('../models/user-model')
const bcrypt = require('bcryptjs')
const jwt = require('jsonwebtoken')
const {CLIENT_URL} = process.env
const sendMail = require('./send-mail')
const {verify} = require("jsonwebtoken");
const {google} = require('googleapis')
const {OAuth2} = google.auth
const client = new OAuth2(process.env.MAILING_SERVICE_CLIENT_ID)

const UserController = {
   register: async (req, res) => {
      try{
         const {firstname, lastname, email, password, cf_password} = req.body
         if(!firstname || !lastname || !email || !password){
            return res.status(400).json({message: "Пожалуйста заполните все
поля."})
         }
         if(firstname.length < 2 || lastname.length < 2){
            return res.status(400).json({message: "Имя и фамилия не должны
иметь меньше двух символов."})
         }
         if(!validateEmail(email)){
            return res.status(400).json({message: "Неправильная почта,
попробуйте еще раз."})
         }
```

```
                const user = await Users.findOne({email})
                if(user){
                    return res.status(400).json({message: "Такой email уже
зарегистрировано."})
                }
                if(password.length < 8){
                    return res.status(400).json({message: "Пароль должен быть не
меньше 8 символов."})
                }
                if(password != cf_password){
                    return res.status(400).json({message: "Ошибка при подтверждении
пароля, попробуйте заново."})
                }
                const passwordHash = await bcrypt.hash(password, 12)
                const newUser = {
                    firstname, lastname, email, password: passwordHash
                }
                const activation_token = createActivationToken(newUser)
                const url = `${CLIENT_URL}/user/activate/${activation_token}`
                sendMail(email, url, "Verify your email address")
                res.json({message: "Ваши данные приняты, пожалуйста,
подтвердите через почту."})
            } catch (e) {
                return res.status(500).json({msg: e.message})
            }
        },
        activateEmail: async(req, res) => {
            try{
                const {activation_token} = req.body
                const user = jwt.verify(activation_token,
process.env.ACTIVATION_TOKEN_SECRET)
                console.log(user)
                const {firstname, lastname, email, password} = user
                const check = await Users.findOne({email})
                if(check){
                    return res.status(400).json({msg:"This email already exists."})
                }
                const newUser = new Users({
                    firstname, lastname, email, password
                })
                await newUser.save()
                res.json({msg: "Account has been activated!"})
```

```
        } catch (e) {
            return res.status(500).json({msg: e.message})
          }
        },
        login: async (req, res) => {
          try{
            const {email, password} = req.body;
            const user = await Users.findOne({email})
            if(!user){
              return res.status(400).json({message: "Аккаунт с такой почтой не
существует."})
            }
            const isMatch = await bcrypt.compare(password, user.password)
            if(!isMatch){
              return res.status(400).json({message: "Неправильный пароль!"})
            }
            const refresh_token = createRefreshToken({id: user._id})
            res.cookie('refreshtoken', refresh_token, {
              httpOnly: true,
              path: '/user/refresh_token',
              maxAge: 7 * 24 * 60 * 60 * 1000
            })
            res.json({message: "Вход успешно выполнен!"})
          } catch (e) {
            return res.status(500).json({msg: e.message})
          }
        },
        getAccessToken: (req, res) => {
          try{
            const rf_token = req.cookies.refreshtoken
            if(!rf_token){
              return res.status(400).json({msg: "Please login now!"})
            }
            jwt.verify(rf_token, process.env.REFRESH_TOKEN_SECRET, (e,
user) => {
              if(e){
                return res.status(400).json({msg: "Please login now!"})
              }
              const access_token = createAccessToken({id: user.id})
              res.json({access_token})
            })
          } catch (e) {
```

```
            return res.status(500).json({msg: e.message})
            }
        },
        forgotPassword: async (req, res) => {
            try {
                const {email} = req.body
                const user = await Users.findOne({email})
                if(!email){
                    return res.status(400).json({message: "Пожалуйста, заполните все
поля."})
                }
                if(!validateEmail(email)){
                    return res.status(400).json({message: "Неправильная почта,
попробуйте еще раз."})
                }
                if(!user){
                    return res.status(400).json({message: "Такой адресс почты не
зарегистрирован."})
                }
                const access_token = createAccessToken({id: user._id})
                const url =
`${CLIENT_URL}/auth/forgot_password/reset_password/${access_token}`
                sendMail(email, url, "Восстановление дступа к аккаунту.")
                res.json({message: "Мы отправили письмо к вам на почту,
пожалуйста проверьте свою почту."})
            } catch (e) {
                return res.status(500).json({message: e.message})
            }
        },
        resetPassword: async (req, res) => {
            try {
                const {password, cf_password} = req.body
                if(!password || !cf_password){
                    return res.status(400).json({message: "Пожалуйста, заполните все
поля!"})
                }
                if(password.length < 8){
                    return res.status(400).json({message: "Пароль должен составлять
минимум 8 символов."})
                }
                if(password != cf_password){
```

```
        return res.status(400).json({message: "Пароль неправильно подтвержден, пожалуйста, попробуйте еще раз."})
                }
                const passwordHash = await bcrypt.hash(password, 12)
                await Users.findOneAndUpdate({_id: req.user.id}, {
                    password: passwordHash
                })
                return res.json({message: "Пароль успешно заменен. Для того что бы продолжить, переавторизуйтесь!"})
            }
            catch (e) {
                return res.status(500).json({msg: e.message})
            }
        },
        getUserInfo: async (req, res) => {
            try{
                const user = await Users.findById(req.user.id).select('-password')
                res.json(user)
            } catch (e) {
                return res.status(500).json({msg: e.message})
            }
        },
        getUsersAllInfo: async (req, res) => {
            try{
                const users = await Users.find().select('-password')
                res.json(users)
            } catch (e) {
                return res.status(500).json({msg: e.message})
            }
        },
        logout: async (req, res) => {
            try{
                res.clearCookie('refreshtoken', {path: '/user/refresh_token'})
                return res.json({msg: "Logged out."})
            } catch (e) {
                return res.status(500).json({msg: e.message})
            }
        },
        updateUser: async (req, res) => {
            try {
                const {firstname, lastname, avatar} = req.body
                await Users.findOneAndUpdate({_id: req.user.id}, {
```

```
        firstname, lastname, avatar
            })
            res.json({msg: "Update Success"})
        } catch (e) {
            return res.status(500).json({msg: e.message})
        }
    },
    updateUsersRole: async (req, res) => {
        try {
            const {role} = req.body
            await Users.findOneAndUpdate({_id: req.params.id}, {
                role
            })
            res.json({msg: "Update Success!"})
        }
        catch (e) {
            return res.status(500).json({msg: e.message})
        }
    },
    deleteUser: async (req, res) => {
        try{
            await Users.findByIdAndDelete(req.params.id)
            res.json({msg: "Deleted Success!"})
        } catch (e) {
            return res.status(500).json({message: e.message})
        }
    },
    googleLogin: async (req, res) => {
        try{
            const {tokenId} = req.body
            const verify = await client.verifyIdToken({idToken: tokenId, audience:
process.env.MAILING_SERVICE_CLIENT_ID})
            console.log(verify)
            const {email_verified, email, given_name, family_name, picture} =
verify.payload
            const password = email + process.env.GOOGLE_SECRET
            const passwordHash = await bcrypt.hash(password, 12)
            if(!email_verified){
                return res.status(400).json({message: "Email неуспешно
верифицирован. "})
            }
            const user = await Users.findOne({email})
```

```
        if(user){
                    const isMatch = await bcrypt.compare(password, user.password)
                    if(!isMatch){
                       return res.status(400).json({message: "Неправильный пароль,
попробуйте еще раз."})
                    }
                    const refresh_token = createRefreshToken({id: user._id})
                    res.cookie('refreshtoken', refresh_token, {
                       httpOnly: true,
                       path: '/user/refresh_token',
                       maxAge: 7 * 24 * 60 * 60 * 1000
                    })
                    res.json({message: "Вы успешно вошли в аккаунт."})
                 } else{
                    const newUser = new Users({
                       firstname: given_name, lastname: family_name, email, password:
passwordHash, avatar: picture
                    })
                    await newUser.save()
                    const refresh_token = createRefreshToken({id: newUser._id})
                    res.cookie('refreshtoken', refresh_token, {
                       httpOnly: true,
                       path: '/user/refresh_token',
                       maxAge: 7 * 24 * 60 * 60 * 1000
                    })
                    res.json({message: "Вы успешно вошли в аккаунт."})
                 }
              } catch (e) {
                 return res.status(500).json({message: e.message})
              }
           }
        }
        const validateEmail = (email) => {
           const re = /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/
           return re.test(email)
        };
        const createActivationToken = (payload) => {
           return jwt.sign(payload, process.env.ACTIVATION_TOKEN_SECRET,
{expiresIn: '5m'})
        }
        const createAccessToken = (payload) => {
```

```
        return jwt.sign(payload, process.env.ACCESS_TOKEN_SECRET, {expiresIn:
'15m'})
        }
        const createRefreshToken = (payload) => {
            return jwt.sign(payload, process.env.REFRESH_TOKEN_SECRET,
{expiresIn: '7d'})
        }
        module.exports = UserController
```

4. Клиенттік бөлімнің басты беті App.js мәтіні

```
        import React, {useEffect} from "react";
        import './App.css';
        import {BrowserRouter as Router} from "react-router-dom";
        import 'materialize-css'
        import {useRoutes} from "./routes";
        import {ScrollToTop} from "./components/ScrollToTop";
        import {useDispatch, useSelector} from "react-redux";
        import {useHttp} from "./hooks/http.hook";
        import {dispatchLogin, fetchUser, dispatchGetUser} from
'./redux/actions/authAction'
        function App() {
            const routes = useRoutes()
            const dispatch = useDispatch()
            const token = useSelector(state => state.token)
            const auth = useSelector(state => state.auth)
            const {loading, request, error, clearError} = useHttp();
            useEffect(() => {
                const firstLogin = localStorage.getItem('firstLogin')
                if(firstLogin){
                    const getToken = async () => {
                        const res = await request('/user/refresh_token', 'POST', null)
                        console.log(res)
                        dispatch({type: 'GET_TOKEN', payload: res.access_token})
                    }
                    getToken()
                }
            }, [auth.isLogged, dispatch])
            useEffect(() => {
                if(token){
                    const getUser = () => {
                        dispatch(dispatchLogin())
```

```
      return fetchUser(token).then(res => {
                  dispatch(dispatchGetUser(res))
               })
            }
            getUser()
         }
      }, [token, dispatch])
      return (
         <Router>
            <ScrollToTop>
               {routes}
            </ScrollToTop>
         </Router>
      );
}
export default App;
```

ОТЗЫВ

на дипломный проект студента
Satbayev University
по специальности 5В070400 - «Вычислительная техника и программное обеспечение»
Ыбраеву Ернұру Ерланұлы
на тему: Разработка электронного учебника по дисциплине «Введение в Web-технологии»

Цель данного проекта является разработка электронного учебника по дисциплине “Введение в Web-технологии”.

Четко видна актуальность данной работы, обусловленная интенсивным развитием информационных технологии. Все материалы, задании и тесты однозначно помогут всем желающим внедриться в эту специальность.

Высокая практическая ценность исследования дополняется лаконичным изложением материала и удачно подобранными примерами, что должно вызвать наибольший интерес к работе у специалистов из сферы информационных систем и компьютерных наук.

Так же, Ыбраев Е.Е. смог провести опрос среди студентов и любителей данной сферы, для того что бы определить актуальную форму и представление самого веб-приложения. Все что было запланировано с научным преподавателем были выполнены.

В результате было создано функциональное и кретивное веб-приложение под названием “ybrayevweb”. Все диаграммы, рисунки и примеры работы приложения соответствуют по данному ПО.

Пояснительная записка была выполнена на высоком уровне, соответствующим требованиям, предьяляемым к данным видам работ. Считаю, что Ыбраев Ернұр Ерланұлы достоин присвоения квалификации бакалавра в области информационно-коммуникационных технологий по образовательной программе 5В070400 - «Вычислительная техника и программное обеспечение»

Руководитель ДП **Аяпбергенова А.Т.**

"16" мая **2022г.**

РЕЦЕНЗИЯ
НА ДИПЛОМНЫЙ ПРОЕКТ

студента Satbayev University
по специальности 5В070400 - «Вычислительная техника и программное обеспечение»
Ыбраева Ернұра Ерланұлы
на тему: Разработка электронного учебника по дисциплине «Введение в Web-технологии»

Цель данного проекта является разработка тренажерного приложения по дисциплине “Введение в Web-технологии”. Проект довольно содержательна и целиком соответствует выданному заданию. Несомненным достоинством проекта является то, что в нем уделено много внимания рассмотрению современных технологии и способы их изучить применяя на практике.

С полной отдачей и без недостатков сделан обзор материала по всем видам web-технологии которые имеются в приложении и функции доступные пользователям для полноценной работы.

Исследовательская часть выполнена на высоком методологическом уровне. Перед тем как реализовать проект было взято опрос со студентов и благодаря результатам было выявлено конкретные актуальные формы и представлении.

В результате было создано полноценное приложение соответствующее всем требованиям к современным приложениям. Все части дипломного проекта написаны и оформлены в соответсвии со стандартами, аккуратны и грамотны, актуальны.Таблицы и рисунки в приложении и в пояснительной записке выполнены достаточно качественно и корректно.

Данный дипломный проект соответствует всем требованиям, предъявляемым к данным видам работ, а Ыбраев Ернұр Ерланұлы заслуживает присвоения академической степени бакалавра техники и технологий по образовательной программе 5В070400 - «Вычислительная техника и программное обеспечение».

Рецензент, доктор PhD
коммерческий директор МАИН
"16" мая 2022г.

Жирнова О.В.

Университеттің жүйе администраторы мен Академиялық мәселелер департаменті директорының ұқсастық есебіне талдау хаттамасы

Жүйе администраторы мен Академиялық мәселелер департаментінің директоры көрсетілген еңбекке қатысты дайындалған Плагиаттың алдын алу және анықтау жүйесінің толық ұқсастық есебімен танысқанын мәлімдейді:

Автор: Ыбраев Ернұр Ерланұлы

Тақырыбы: Web-технологияларға кіріспе пәні бойынша электрондық оқулық әзірлеу

Жетекшісі: Жибек Алибиева

1-ұқсастық коэффициенті (30): 0.2

2-ұқсастық коэффициенті (5): 0

Дәйексөз (35): 2.1

Әріптерді ауыстыру: 0

Аралықтар: 0

Шағын кеңістіктер: 1

Ақ белгілер: 0

Ұқсастық есебін талдай отырып, Жүйе администраторы мен Академиялық мәселелер департаментінің директоры келесі шешімдерді мәлімдейді :

☐ Ғылыми еңбекте табылған ұқсастықтар плагиат болып есептелмейді. Осыған байланысты жұмыс өз бетінше жазылған болып санала отырып, қорғауға жіберіледі.

☐ Осы жұмыстағы ұқсастықтар плагиат болып есептелмейді, бірақ олардың шамадан тыс көптігі еңбектің құндылығына және автордың ғылыми жұмысты өзі жазғанына қатысты күмән тудырады. Осыған байланысты ұқсастықтарды шектеу мақсатында жұмыс қайта өңдеуге жіберілсін.

☐ Еңбекте анықталған ұқсастықтар жосықсыз және плагиаттың белгілері болып саналады немесе мәтіндері қасақана бұрмаланып плагиат белгілері жасырылған. Осыған байланысты жұмыс қорғауға жіберілмейді.

Негіздеме:

Күні 19.05.2022 *Кафедра меңгерушісі*