ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Кибернетика және ақпараттық технологиялар институты

Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

Сағынбек Бекзат Әділұлы

Веб-қосымшаларды әзірлеу «Нақты уақыт қосымшаларына арналған платформа»

ДИПЛОМДЫҚ ЖОБА

5В070300 – «Ақпараттық жүйелер» мамандығы

Алматы 2020

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы қазақ ұлттық техникалық зерттеу университеті

Кибернетика және ақпараттық технологиялар институты

Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
КАӨжС кафедрасы меңгерушісі
техн. ғыл. канд., ассистент-профессор
_____ Н.А. Сейлова
«__» _____2020 ж.

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: «Нақты уақыт қосымшаларына арналған платформа»
5В070300 – «Ақпараттық жүйелер» мамандығы

Орындаған Сағынбек Б.Ә.

Ғылыми жетекші
Тьютор
_____ Рысқұлбек Е.А.
«____» мамыр 2020 ж.

Алматы 2020

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Кибернетика және ақпараттық технологиялар институты

Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

5В070300 – Ақпараттық жүйелер

БЕКІТЕМІН
КАӨжС кафедрасы меңгерушісі
техн. ғыл. канд., ассистент-профессор
_____ Н.А. Сейлова
«__» _____2020 ж.

.

Дипломдық жобаға орындауға
ТАПСЫРМА

Білім алушы: Сағынбек Бекзат Әділұлы
Тақырыбы: «Нақты уақыт қосымшаларына арналған платформа»
Университет Ректорының 2020 жылғы «27» қаңтар №762 -б бұйрығымен бекітілген
Аяқталған жұмысты тапсыру мерзімі 2020 жылғы « 25 » мамыр
Дипломдық жұмыстың бастапқы берілістері: диплом алдындағы практикалық жұмыс қорытындысы, тақырып бойынша әдебиеттерге шолу нәтижелері,теориялық мәліметтердің жиыны
Дипломдық жұмыста қарастырылатын мәселелер тізімі:
а)қойылған мәселенің қазіргі жағдайын пайымдау
ә)ақпараттық қамтаманы құру
б)программалық қамтаманы құру
Сызбалық материалдар тізімі: Power Point бағдарламасындағы слайдтар
Сызба материалдар: 15 слайдпен көрсетілген
Ұсынылатын негізгі әдебиет: 9 атау

Дипломдық жұмысты даярлау

КЕСТЕСІ

Бөлім атаулары, қарастырылатын мәселелер тізімі	Ғылыми жетекшіге, кеңесшілерге өткізу мерзімі	Ескерту
Пәндік саланы талдау	10.01.2020 – 08.02.2020	
Деректер базасының моделін құру	05.02.2020 – 10.03.2020	
Программалық қамтаманы құру	11.03.2020 – 28.04.2020	

Дипломдық жұмыс бөлімдерінің кеңесшілері мен норма бақылаушының аяқталған жұмысқа қойған

қолтаңбалары

Бөлімдердің атауы	Кеңесшілер (аты-жөні, тегі, ғылыми дәрежесі, атағы)	Қол қойылған мерзімі	Қолы
БҚ жасау	Қабдуллин М.А., ассистент		
Норма бақылаушы	Бауыржан М.Б., тьютор		

Ғылыми жетекшісі _____ Рысқұлбек Е.А.

Тапсырманы орындауға қабылдаған білім алушы _____ Сағынбек Б.Ә.

Күні «27» қаңтар 2020 ж.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Сәтбаев университеті

Ғылыми жетекшінің пікірі

Дипломдық жұмыс

Сағынбек Бекзат Әділұлы

5В070300 – Ақпараттық жүйелер

Тақырыбы: «Нақты уақыт қосымшаларына арналған платформа»

Бұл дипломдық жұмыс өзінің логикалық құрылымымен ерекшеленген. Түсіндірме жобаның құрамы кіріспеден, 3 бөлімнен, қорытындыдан, әдебиеттер тізімінен және қосымшадан тұрады.

Менің пікірімше, диплом жобалаушы алдына қойылған тапсырманы толығымен орындады және кейінгі технологияларын меңгергендігін көрсетті.

Жалпы дипломдық жоба профессионалдық деңгейде орындалған. Түсіндірме жазба сауатты бейнеленген, жоба бойынша барлық қажетті ақпараттар бар.

Кемшілік ретінде кейбір шағын стилистикалық қателерді атап кетуге болады.

Жоғарыда айтылғандарға байланысты, дипломдық жұмыс 5В070300 – «Ақпараттық жүйелер» мамандығының бітіру жұмыстарына қойылатын талаптарына сәйкес және дипломдық жұмыс қорғауға жіберіле алады, ал оның авторы Сағынбек Бекзат Әділұлы бакалавр академиялық дәрежесін алуға лайықты деп есептеймін.

Ғылыми жетекші

Тьютор Рысқұлбек Е.А.

«___» _____ 2020 ж.

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: Сағынбек Бекзат

Название: Платформа для приложении реального времени

Координатор:Ерсұлтан Раскұлбек

Коэффициент подобия 1:1,1

Коэффициент подобия 2:0

Замена букв:1

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*

Протокол анализа Отчета подобия

заведующего кафедрой / начальника структурного подразделения

Заведующий кафедрой / начальник структурного подразделения заявляет, что ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Сағынбек Бекзат

Название: Платформа для приложении реального времени

Координатор: Ерсұлтан Раскұлбек

Коэффициент подобия 1:1,1

Коэффициент подобия 2:0

Замена букв:1

Интервалы:0

Микропробелы:0

Белые знаки:0

После анализа отчета подобия заведующий кафедрой / начальник структурного подразделения констатирует следующее:

☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, работа признается самостоятельной и допускается 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, работа не допускается к защите.

Обоснование:

……
……
……
……
……

…… ……

Дата *Подпись заведующего кафедрой /*

начальника структурного подразделения

Окончательное решение в отношении допуска к защите, включая обоснование:

……
……
……
……
……

…… ……

Дата *Подпись заведующего кафедрой /*

начальника структурного подразделения

АҢДАТПА

Бұл жоба нақты уақытта жұмыс жасауға мүмкіндік беретін заманауи технологияларды қолданады. Нақты уақыт қосымшалар қазірде өте өзекті бағыттардың бірі болып саналады. Бұдай жүйелер онлайн ойын, чат қосымшалары, видеоконференция, стриминг, мониторинг жүйелері, қашықтықтан басқару жүйелері және тағы басқа сфераларда кең таралған.

Веб қосымша екі бөлектен тұрады, клиенттік және серверлік. Жобаның серверлік бөлігі Go (тағы да оны Golang депте атайды) тілінде жазылды, клиенттік интерфейс бөлімі javascript кітапханасы React-та қолданылып жазылды. Visual studio code код редакторы пайдаланылды.

АННОТАЦИЯ

Данный проект использует современные технологии, позволяющие работать в реальном времени. В настоящее время приложения реального времени являются одним из самых актуальных направлений. Такие системы широко распространены в сфере онлайн-игры, приложения к чату, видеоконференции, стриминга, системы мониторинга, системы дистанционного управления и др.

Веб-приложение состоит из двух частей, клиентская и серверная часть. Серверная часть проекта написаны на языке Go (называют еще как Golang), часть клиентского интерфейса реализован с помощью javascript библиотеки React. Для реализации проекта был использован редактор кода Visual studio code.

ANNOTATION

This project uses modern technologies that allow you to work in real time. Currently, real-time applications are one of the most relevant areas. Such systems are widely used in the field of online games, chat applications, video conferencing, streaming, monitoring systems, remote control systems, etc.

The web application consists of two parts, the client and server parts. The server part of the project is written in Go (also known as Golang), and the client interface is implemented using the React JavaScript library. The Visual studio code editor was used to implement the project.

МАЗМҰНЫ

КІРІСПЕ

Қазірде интернет жүйесі қарқынды дамып келуіне байланысты, нақты уақыт қосымшаларына сұраныс пен талаптар артып келеді. Күннен күнге адамдарға, бизнестерге онлайн қызмет өзектілігі артуда. Бұл қосымшаларды пайдалану бизнес-модельдердің кең спектрі үшін қолайлы жолы болып табылады. Бұл жүйе онлайн ойын, чат қосымшалары, видеоконференция, стриминг, мониторинг жүйелері, қашықтықтан басқару жүйелері және тағы басқа сфераларда кең таралған. Қашықтықтан оқу, сөйлесу, жұмыс жасау қазірде танымал және өзекті болып келеді.

Нақты уақыт қосымшаларының жұмысы негізінде теориялық тұрғыдан алсақ бұл "деректерді дереу жеткізу"дегенді білдіреді. Алайда, іс жүзінде ақпарат тиісті кезеңде, уақытта жеткізілуі тиіс және бұл кезең әрқашан контекстке байланысты болады. Мұнда тағы бір маңызды сәт-ол клиент тарапынан әрдайым тұрақты сұрау салусыз деректерді жеткізу қажет.

Техникалық жағынан бізде нақты уақыттағы қосылыстарды жасаудың бірнеше әдістері бар:

- HTTP long polling - клиент жаңа ақпаратты сервер сұрайды, сервер жаңа деректер қол жетімді болғанға дейін сұрауды ашық ұстайды.
- HTTP ағынын жіберу - клиент пен сервер арасындағы тұрақты байланыстарға негізделген.
- WebSocket - бұл бір TCP байланысы арқылы сервер мен клиент арасындағы екі бағытағы байланысқа мүмкіндік береді.

Веб қосымшада React кітапханасын қолданатын API құрылады, серверлік бөлік жаңа әрі жылдам, Go тілінде жазылады және ол Postgresql мәліметтер базасын басқару жүйесімен байланысады. Серверге сұраныстар json форматында алмасады және Rest желідегі үлестірілген өзара қосымшалардың өзара әрекетесуін қамтамасыз ететін архитектуралық стилі қолданылады.

Бұл дипломдық жобаның өзектілігігі анық. Жобаны жасауда қолданыстағы жүйелерді талдау, платформаның прототипін жасау, жаңа әрі өзекті заманауи технологияларды қолдану, олардың артықшылықтарына, кемшіліктеріне талдау, мәліметтер базасын талдау бойынша жұмыстар жүргізіледі.

1 Нақты уақыт қосымшаларына арналған платформаға шолу

1.1 Нақты уақыт қосымша ұғымына шолу

Нақты уақыт қосымшалары(real-time application(RTA)) бұл тікелей немесе ағымды қабылдайтын уақытша шектерде жұмыс істейтін қолданбалы бағдарлама немесе қосымша. Нақты уақыт бағдарламалары "соңғы мерзім" деп аталатын белгілі бір уақытша шектеулер ішінде жауап беруге кепілдік беруі тиіс. Нақты уақытта жауап миллисекунд, кейде микросекунда келеді. Синхронды бағдарламалау тілдерін , нақты уақыт операциялық жүйелерін және нақты уақыт желілерін барлығын немесе бірнешеуі қолданылады, бұлар нақты уақыт бағдарламалық қосымшасын құру үшін қажетті негізді қамтамасыз етеді. Көптеген маңызды қосымшаларға осы нақты уақыт режимінде жұмыс жасайтын системаларды пайдалануы керек, мысалы, автопилот режимінде жұмыс жасауға қабілеті бар автокөліктер немесе бір маңызды нысанды қадағалайтын камералар, бұл екі жағдайда да жылдам және дәл ақпарат беру қабілеті болу керек.

Нақты уақыт қосымшаларын көптеген сферада пайдаланады, олар:

- бейнеконференцияларға арналған қосымшалар;
- видео хабарламаларды қолданатын қосымшалар;
- онлайн ойындар;
- чат қосымшалары;
- аудио хабарласу қосымшалары;
- кейбір электрондық коммерция операциялары;
- IP-телефония;
- бұлтты технологияларда бірлескен ресурстарды пайдалану жатады.

Бірақ, нақты уақыт режимінде деректерді алмасу электрондық поштаны, хабарландыру тақталарын, блогтарды немесе басқа да интернет-коммуникация нысандарын қамтымайды, онда пайдаланушылар контентті жіберіп, оны жариялайтын уақытты ескермей оқиды.

Нақты уақыт қосымшалары ұзақ уақыт бойы бір орыннан екінші орынға берілетін ақпараттың үздіксіз ағынын қамтиды. Бұл іс-қимылды “ағын” деп атайды, байланысты ақпараттар тізбегін береді, мысалы, аудио немесе видео деректер. Жүйені нақты жүйе деп атайды, егер толық операциялар дұрыстығы оның логикалық дұрыстығына ғана емес, сонымен қатар ол орындалатын уақытқа байланысты болса.

Нақты уақыттағы сауда жүйелері, нақты уақыттағы алаяқтықты зерттеу, нақты уақыттағы жарнамаларды жеткізу, нақты уақыттағы мониторинг немесе нақты уақыттағы әлеуметтік желілерді талдау, бұл деректер көлемі үлкен, сондықтан нақты уақыттағы қосымшаларға қойылатын талаптар жоғары, ал деректер көздері үздіксіз. Келген жаңа деректер дереу өңделуі тиіс, әйтпесе келесі деректер жинақталып, өңдеу ешқашан аяқталмайды. Бір секундтан аз немесе тіпті миллисекундтан аз жауап беру уақыты қажет,бұл ағындық есептеулердің жоғары масштабталуын талап етеді.

Нақты уақыт байланысы(real-time communication(RTC)) - бұл біррангты архитектураны пайдалана отырып, таратуда кідіріссіз нақты уақыт режимінде деректерді алмасуға мүмкіндік беретін әдістеме. Нақты уақытта байланыс жартылай кешенді(Полудуплекс) немесе толық кешенді (Толық дуплекс) режимде жүзеге асырылуы мүмкін:

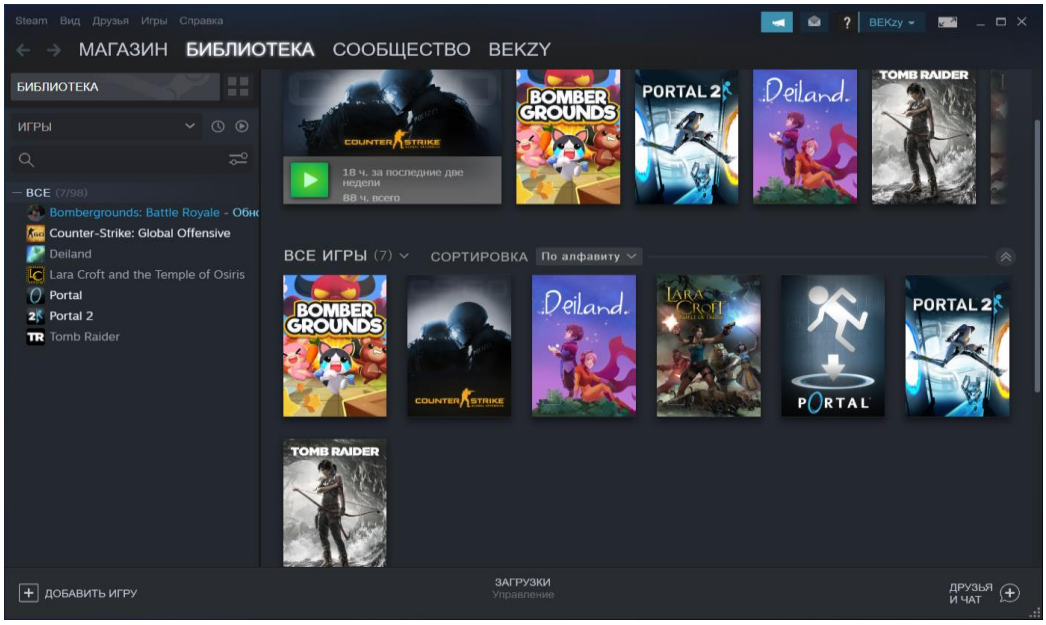
- Полудуплекс-бір рет бір бағытта бір арна бойынша байланыс. Жіберуші немесе алушы жіберуі мүмкін, бірақ бір уақытта алмауы мүмкін.
- Толық дуплекс-жіберуші мен алушы екі қатар байланыс арналары бойынша бір мезгілде хабарламаларды жібере және ала алады.

1.2 Веб қосымшаларда нақты уақыт байланыс технологияларын қолданылуына шолу

Нақты уақыт жүйелерін әзірлеу бизнес-модельдердің кең спектрі үшін қолайлы тәсіл болып табылады. Веб қосымшаларда технология жағынанда қалыс қалмауда. Қазіргі заманауи браузерлер WebRTC, WebSocket сияқты технологияларды қолдана алады.

1.2.1 Steam платформасы

Steam – Valve компаниясымен 2003 жылы әзірлеген, ойындар мен бағдарламаларды сандық таратудың онлайн-қызметі. Авторлық құқықтарды қорғаудың техникалық құралы ретінде саналады және ойындар мен стримингтік платформа болып табылады, сондай-ақ тіркелген ойыншылар үшін әлеуметтік желі рөлін атқарады.



1.1 Сурет – Steam платформасының негізгі беті

Big Picture режиміндегі пайдаланушы интерфейсін SteamOS пайдаланады. Бұл операциялық жүйе Steam Machines шағын компьютерінде жеткізіледі, бұл тегін ОЖ дербес қондырғылар үшін бөлек шығарылатын болды. Бұл операциялық жүйеде стационарлық компьютерден теледидарға трансляциялау мүмкіндігі бар, Steam Cloud (бірыңғай аккаунтты пайдалану кезінде кез келген құрылғыдан сатып алынған барлық ойындарға қолжетімділігі). Операциялық жүйе ойындардан басқа, аудио және бейне файлдарын ойнатуға қабілетті. Музыка Steam клиенті арқылы ресми түрде таратылады және аудиоплеер тікелей клиентке орнатылған.

Steam-ді қолдану үшін оның клиенттік платформасын орнату қажет. Ойынды ойнау үшін немесе бағдарламаны қолдану үшін, бірінші оны жүктеу қажет және оны әр дайым жаңартуларын жасап отыру қажет. Кейбір ойындар мен бағдарламалар клиентін компьютеріне ауыр немесе қолдамайтын болып келеді, сол үшін клиенттер шығындалып жаңа құрылғы алуға мәжбүр немесе оны мүлдем қолданбайды. Steam платформасында видеоконференция, стриминг, видеохабарласу жасау мүмкіндігі қарастырылмаған.

1.3 Есептің қойылымы

Нақты уақытта жұмыс жасай алатын веб қосымшаны құру үшін кеселі талаптар орындалу қажет:

- Нақты уақыт қосымшасына қажетті технологияларды зерттеу және оларға талдау жасау қажет.
- Веб қосымшаны құру кезінде қазіргі заманауи технологияларды пайдалану.
- Мәліметтер базасымен байланыстыру.
- Мәліметтер базасының концептуалды модельін құру.
- Мәліметтер базасын жобалау.
- Single page application (SPA) қолдану.

Дипломдық жобаны құру кезінде клиент интерфейсін жасауға javascript кітапханасы React қолданылды, Go тілі серверге қолданылды, Postgresql мәліметтерді базасын басқару жүйесі қолданылды.

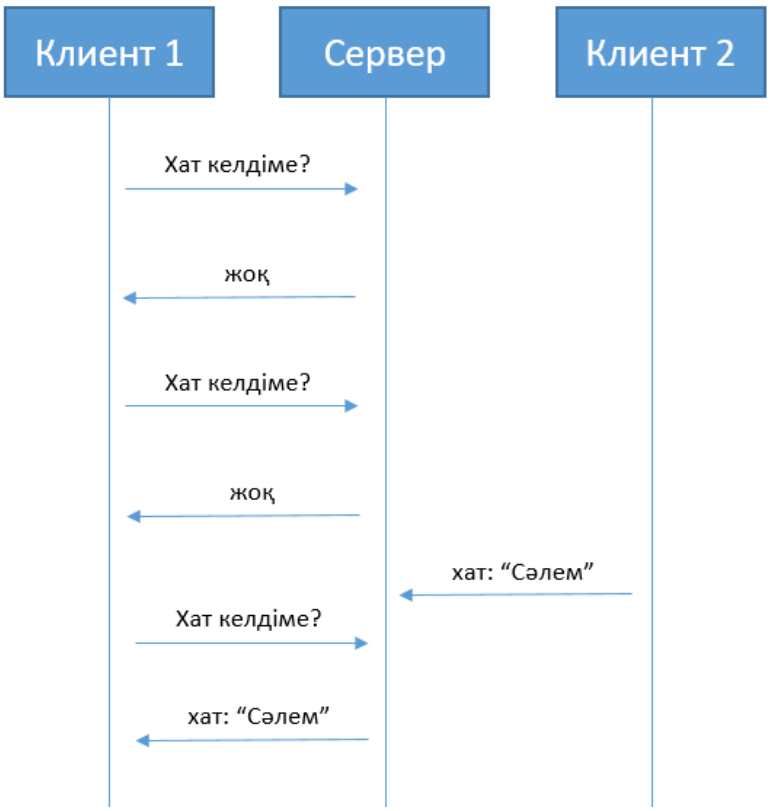
2 Нақты уақыт веб қосымшасын құру үшін технологияларға шолу

2.1 WebSocket нақты уақыт режимінде хабар алмасудың озық технологиялардың бірі

WebSocket дегеніміз - тұрақты байланыс арқылы браузер мен сервер арасында деректер алмасу мүмкіндігін нақты уақыт аралығында қамтамасыз етеді. Алмасатын деректер “пакеттер” ретінде ешқандай сұрауларсыз жіберіледі.

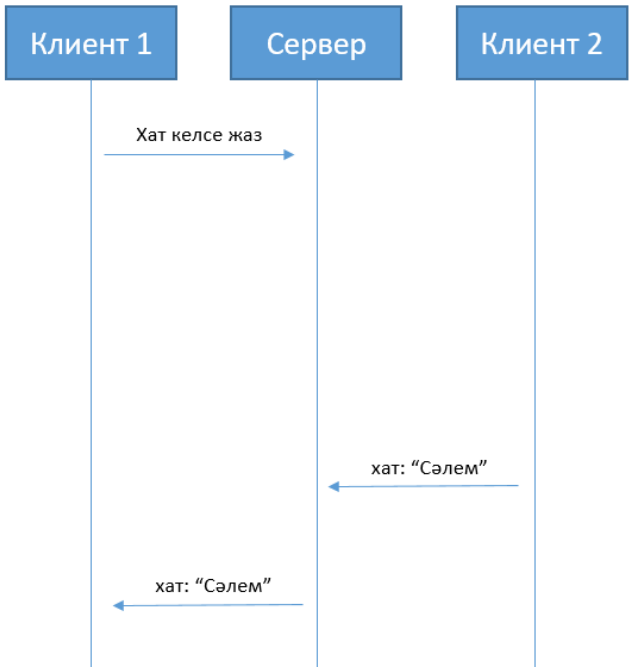
Көп жағдайда WebSocket – ті пайдаланады:

- нақты уақыт қосымшаларында;
- чат қосымшалар;
- ойындар;
- Internet of things(IoT) қосымшаларда.



2.1 Сурет – HTTP арқылы хабарлама алмасу схемасы

Браузер үнемі серверден жаңа хабарламаны сұрайды, егер болмаса, хабарламаның жоқтығы туралы жауап қайтарады, болса хабарламаны алады.



2.2 Сурет – WebSocket арқылы хабарлама алмасу схемасы

Веб-сокеттерге жауап қайтару үшін үнемі серверден сұраулар қажет емес. Бір рет қана сұрау беріп, жауаты тек басқа клиент сұрау жібергенде ғана аласыз.

Javascript-те веб сокет байланысын құру үшін WebSocket обьектін құру қажет.

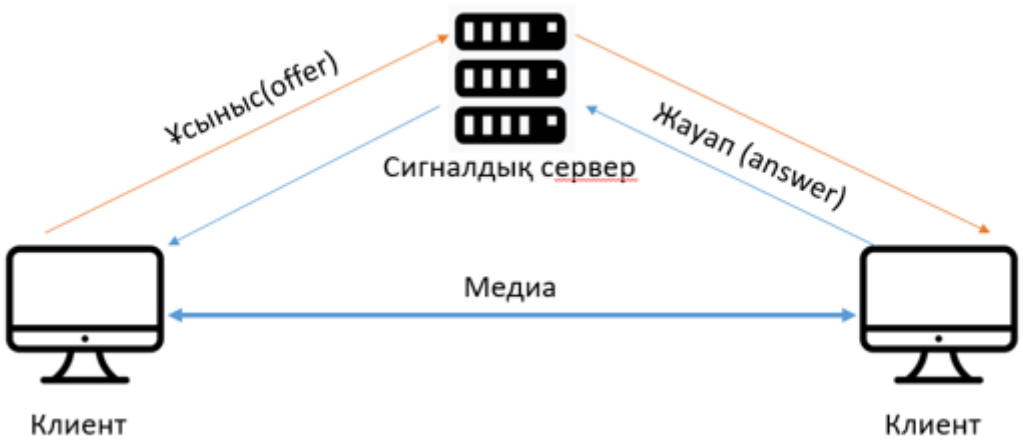
```
1
2   const socket = new WebSocket("ws://localhost:8080/api/mainchat");
3
4   socket.onopen = function() {
5       alert("Байланыс орнатылды.");
6   };
7
8   socket.onclose = function(event) {
9     if (event.wasClean) {
10      alert('Байланыс жабылды');
11    } else {
12      alert('Байланыс үзілді');
13    }
14    alert('Код: ' + event.code + ' себебі: ' + event.reason);
15  };
16
17  socket.onmessage = function(event) {
18    alert("Ақпарат алу " + event.data);
19  };
20
21  socket.onerror = function(error) {
22    alert("Қате " + error.message);
23  };
24
```

2.3 Сурет – WebSocket байланысын құру мысалы.

Егер байланыс сәтті орындалса onopen оқиғасы орындалады. Бұл алғашқы байланыс орнатылғанда орындалады. Егерде қате болса onerror оқиғасы орындалады. Байланыс тоқтағанда немесе үзілгенде onclose орындалады. Send функциясын пайдаланып, серверге хат жіберуге болады, сервер хатты барлық байланысқан қолданушыларға жібереді, ол қолданушыларда onmessage оқиғасы орындалуы арқылы алады.

2.2 WebRTC ағын деректерді жіберу технологиясы

WebRTC - браузерлер немесе осы технологияны қолдайтын бағдарламатар арасындағы ағын деректерді жіберуді басқаратын және Peer-to-Peer байланысын орнатуға мүмкіндік беретін ашық жоба. Осы негізінде жатқан технологиялар ашық веб-стандарт ретінде іске асырылған және барлық заманауи негізгі браузерлерде қарапайым JavaScript API түрінде қол жетімді. Технологиянын артықшылықтарына браузерде конференцияны өткізу оңай және басқа қосымшалар орнату қажет етпейді, қауіпсіздік денгейі жоғары, медиапотоктар шифрланады және DTLS қолданады, байланыс қосылыстары қорғалған, пайдаланатын кодектер жақсы байланыс сапасын қамтамасыз етеді, HTML5 және javascript интерфейстің кез-келген элементерін жүзеге асыруға мүмкіндік береді.



2.4 Сурет – WebRTC арқылы peer-to-peer байланысу схемасы.

1-ші клиент(инициатор) 2-ші клиентке “ұсыныс” сигналды сервер арқылы жібереді, бұл 1-ші клиент туралы байланысуға керек барлық ақпарат сақтайды. 2-ші клиент “ұсыныс” қабылдап, өңдегеннен кейін, сигналды сервер арқылы “жауап” жібереді. Мұндағы “ұсыныс” және “жауап” SDP(Session Description Protocol) арқылы сақталады және жіберіледі, сипаттамада жіберілетін медианын

түрі, оның форматы, пайдаланылатын тарату хаттамасы, IP мекенжайы және соңғы нүкте порты туралы және басқада керекті ақпараттар сақталады.

Осы технологияны қолдайтын браузерлерде getUserMedia(2.5-сурет) функциясы бар. Бұл функция 3 аргумент қабылдайды, бірінші құрылғы түрі, ол аудио және видео болу мүмкін немесе жеке-жеке, екінші қолданушыда құрылғы түрі табылса және рұқсат алса, онда осы процесті басқарушы функцияға беріледі, үшінші байланыс орнатылмаса немесе құрал түрі табылмаған жағдайда орындалатын, қате кодын және түрін шығаратын функция орындалады.

```
1
2   navigator.getUserMedia({audio: true, video: true}, success, error);
3
4   function success(localMediaStream) {
5       /* видео және аудио потоктарды өңдеу функциясы*/
6   }
7
8   function error(error) {
9       /* қате шығару */
10      console.log(error);
11  }
12
```

2.5 Сурет – getUserMedia функциясын қолдану мысалы.

2.3 React танымал javascript кітапханасы

Reactjs – пайдаланушы интерфейсін жасауда қолданылатын, дамып келе келе жатқан, танымал javascript кітапханасы. Бұл кітапхананы facebook компаниясы өз қосымшаларына әзірлеген, кейін танымалдығы артқан. Қазірде үлкен компания бұл кітапхананы дамытуда және қолдауда. Көп мамандарға өзінің жеңілділігімен, ынғайлылығымен, оңай масштапталуымен белгілі.

Кітапхана шағын және бір-бірінен оқшауланған компоненттерден тұратын веб-интерфейс жасауға мүмкіндік береді. Компоненттер дегеніміз – болашақ веб-қосымшаның кішігірім бөліктерік, кейін react бұл кішігірім бөліктерді біріктіреді. Жазылған компонентерді, қайтадан пайдалануға болады. Сол үшін, компоненттерді қайталап пайдаланатындай бөлу қажет.

Компоненттерді жазуды жылдамдату әрі жеңілдету мақсатында көбісі арнайы jsx синтаксисін пайдаланады. Бұл синтаксисті пайдалану міндетті емес, тек таза javascript тілінде жазуғада болады, бірақ көп мамандар осы синтаксисті пайдаланады және өз уақытыңызды үнемдейсіз, сол себебті пайдалануға ұсынылады.

```jsx
const element = (
  <h1 className="header">
    Привет, мир!
  </h1>
);

```

2.6 Сурет – Jsx синтаксисінде жазылған кодтың мысалы.

JSX – ті пайдаланып жазылған код, кейін javascript-ке аударылады. Javascript-ке аудару үшін babel компиляторы қажет.

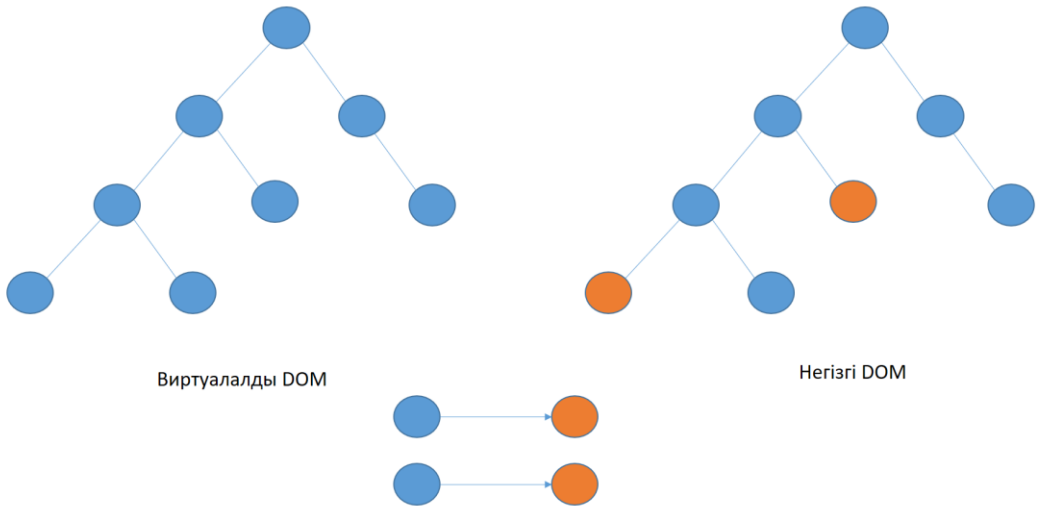
```jsx
const element = React.createElement(
  'h1',
  {className: 'header'},
  'Привет, мир!'
);

```

2.7 Сурет – Jsx синтаксисін пайдаланбай жазылған код.

Тағы бір ерекшелігі ол бір жақты деректер ағыны. Компонентер бір-біріне деректер жібере алады. Бір жақты дейді, өйткені, компонент тек өзінен жоғарғы компонеттен ақпарат ала алады және ол ақпаратты өзгерте алмайды. Ол үшін, компонент өзінен жоғарғы тұрған компоненттің берген ақпаратын өзгертетін кері шақыру функциясын қолданады.

React интерфейсті тиімді жағарту үшін, VirtualDom пайдаланады. Ол интерфейстің Document Object Model(DOM) (2.8-сурет) структурасын кэш-жадына сақтап қояды. Нақты DOM өзгеріс болғанда, қазіргі және бұрынғы DOM структурасын салыстырып, тек айырмашылығын ғана озгертеді. Сонын арқасында интерфейс тиімді әрі жылдам өзгереді.



2.8 Сурет – VirtualDom жұмысының үлгісі.

2.4 Go әмбебап бағдарламалау тілі

Go (Golang депте аталды) – компиляциялатын, көп ағынды, құрылымдылық типизацияланған бағдарламалау тілі. 2007 жылы Google компаниясымен жасалып басталған, 2009 жылы ресми ұсынылған. Синтаксисі C және Pascal бағдарламалау тілдеріне ұқсас. Барлығына оңайлылығымен, тездігімен және горутиндерімен танымал. Жадыны үнемді және тиімді басқарады, қоқысты тазалаушы(garbage collection) бар. бағдарламаларды баяу жинау, кодты оқу және оны қолдау қиындығы, бағдарламалардың қиын және нашар құжатталуы, жаңартулардың қиындығы, асинхронды бағдарламалардың ауырлығы және басқада проблемаларды шешуге арналып жасалған тіл.

Go тілінің негізгі артықшылықтары:

- қатаң типизацияланған;
- жылдам компиляция;
- пакеттерді қашықтан басқару (go get) және пакеттер бойынша онлайн құжаттама;
- параллелизмнің жеңілдетілген процестері(go routines), арналар(channels) және select операторы;
- қоқысты жинаушы;
- бастапқы коды ашық таратылады;
- сиандартты кітапханалары көп;
- қарапайым және тұрақты грамматика.

```go
package main

import "fmt"

func main() {
    name := "Марат"
    fmt.Println("Сәлем,", name)
}

```

2.9 Сурет – Go тіліндегі қарапайым бағдарлама.

Go тіліндегі әрбір бағдарлама пакеттерден (packages) тұрады. Басты main пакеті болып табылады, одан бағдарламаны орындау басталады. Мысалы, төбедегі қарапайым бағдарламада, “fmt” пакеті импортталған, ішінде Println функциясы анықталған, бұл функция кез-келген деректерді шығаруға жауапты.

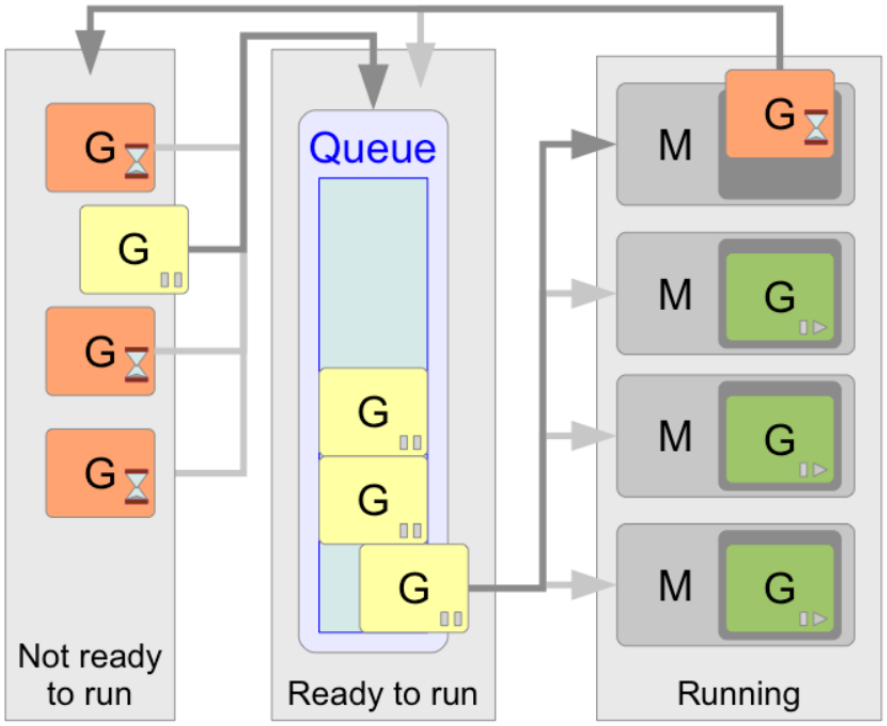
Горутиндер (goroutines) олар өздері іске қосылған функцияға қарамастан, олардан тәуелсіз, параллель операцияларды ұсынады. Горутиндердің негізгі басты ерекшелігі - олар параллель орындалуы мүмкін. Яғни, көп ядролы архитектурасы бар процессорлардың жекелеген ядроларында жеке горутиналарды орындау мүмкіндігі бар, сол арқылы горутиндер паралелльді орындалады және бағдарлама тезірек аяқталады.

Орташа алғанда, бір горутиннің салмағы шамамен 4,5 kb. Яғни, мысалы, 8Gb RAM бар, сіз 1600 мың жұмыс істейтін горутин қолдана аласыз. Горутиндер келесі жағдайларда қолдануға пайдалы:

- Егер асинхрондық қажет болса. Мысалы, біз желімен, дискімен, мьютекс ресурстарымен, қорғалған деректер қорымен және т. б. жұмыс істегенде.

- Егер функцияны орындау уақыты өте үлкен болса.

Жоспарлаушының мақсаты (scheduler) орындауға дайын горутиндарды (G) бос машиналар (M) бойынша бөлу(2.10-сурет).



2.10 Сурет – Жоспарлаушының горутиндермен жұмыс жасау схемасы

Орындауға дайын горутиндер кезек тәртібімен орындалады, яғни FIFO (First In, First Out). Горутинаның орындалуы тек қана ол орындала алмайтын кезде ғана үзіледі: яғни жүйелік шақырылу болса немесе синхрондалған объектілерін пайдалану салдарынан.

Docker, kubernetes, syncthing, goLearn, g3n, Dropbox, Netflix, Twich секілді үлкен жобалар осы тілде жасалған.

2.5 PostgreSQL мәліметтер базасын басқару жүйесі

PostgreSQL бұл обьектілік-реляциялық, әмбебап тегін мәліметтр базасын басқару жүйесі(МББЖ). UNIX ұқсас платформаларда жұмыс жасау үшін арналған, дегенімен, басқада Mac OS X, Solaris және Windows сияқты түрлі платформаларда жұмыс істей алады. PostgreSQL Бекркли қаласындағы Калифорния университетінде еркін және толық ашық бағдарламалық қамтама ретінде жасалып шыққан.

Ол SQL стандартарының көп бөлігін қолдайды және көптеген қазіргі заманауи функцияларды ұсынады: күрделі сұраныстар, триггерлер, транзакциялардың бүтіндігі, сыртқы кілттер, жаңартылған көріністер, мультиверсияның параллелизмін басқару. 2.7 кестеде PostgreSQL мәліметтер өлшемі көрсетілген.

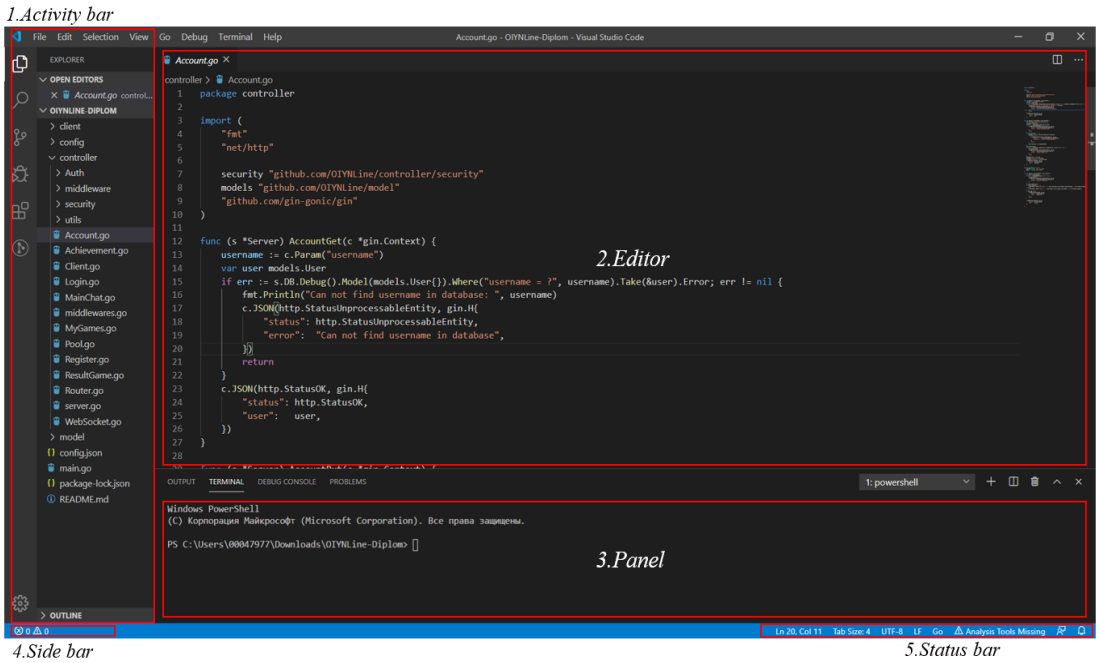
2.1 Кесте - PostgreSQL көптеген деректерді өңдей алады

Ең үлкен деректер қоры	Шексіз
Кестелердің максималды мөлшері	32 Тбайт
Жолдың максималды мөлшері	1,6 ТБ
Өрісінің ең үлкен өлшемі	1 Гбайт
Кестелердегі жолдардың максимал саны	Шексіз
Кестеге сәйкес колонкалардың максималды саны	250-1600 баған
Кестедегі индекстердің ең көп саны	Шектелмеген

Сонымен қатар, PostgreSQL қолданушымен көптеген тәсілдермен кеңейтілуі мүмкін, мысалы жаңа деректер түрлерін, функцияларды, операторларды, агрегаттық функцияларды, индекстік әдістерді, процедуралық тілдерді қосу арқылы

2.6 Visual Studio Code код редакторы

Visual Studio Code(VS code) - 30-дан астам бағдарламалау тілі мен файл пішімдерін қолдайтын, Windows, Linux және mac OS үшін әзірленген, жеңіл әрі қуатты, тегін код редакторы. Бұл Microsoft компаниясымен 2015 жылы әзірленіп бастап, толық нақты версиясы 2016 жылы шыққан. Синтаксисті жарықтандыру, IntelliSense, git-пен жұмыс жасау құралдары, кеңейту қосымшаларын тез орнатылуы, рефакторинг құралдары, код бойынша навигация секілді басты функциялары бар(2.11 - сурет).



2.11 Сурет – Visual Studio Code редакторының интерфейсі

Редактордың интерфейсін негізгі 5 бөлікке болуге болады:

1. Іздеу, git-пен жұмыс, баптау, қосымшаларды орнату, секілді басты функциональдар орналасқан.

2. Файлы ашу, оны өзгерту, жаңарту, бірнеше файлды қатарынан ашу осы бетте орындалады.

3. Әр түрлі панельдер орналасқан. Мысалы терминал, шығарулар, ескертулер, қателер, проблемалар панельдері бар. Қолданушыға қарай бұл панельдерді алуға немесе қосуға болады.

4. Проектің немесе бағдарламаның активті көрінісі.

5. Проект немесе бағдарлама туралы ақпараттар орналасқан.

Visual studio code пайдалану туралы деректер жинайды және сол деректерді Microsoft корпорациясына жібереді. Деректерді ұсыну міндетті шара емес, оларды баптаулардан өшіріп тастауға болады, бірақ жекелендіру секілді кейбір функциялары өшіру мүмкін емес. Бұл деректер Microsoft-тың филиалдарына, еншілес компанияларға немесе құқық қорғау органдарына берілуі мүмкін.

2018 жылдан бастап Visual Studio code үшін Python бағдарламалау тілінің кеңейтімі пайда болды. Бұл әзірлеушілерге бағдарламалық кодты жөндеу, өңдеу және тестілге кең мүмкіндіктер береді. 2019 жылдың наурыз айындағы жағдай бойынша, өнімде бар пайдаланушы интерфейсі арқылы сіз мыңдаған кеңейтімдерді "бағдарламалау тілдері" санатынынан жүктеп, орната аласыз. Visual Studio Code – бұл C#, TypeScript, JavaScript танымал бағдарламалау тілдерін қоса алғанда, 30 астам бағдарламалау тілдерімен және файлмен жұмыс істейтін ыңғайлы код редакторы.

Редактордың артықшылықтары:

- ол тегін және коды барлығына ашық таратылады, сол себепті оны әркім өзінің ыңғайына келтіре алады.

- Жүктегенде көлемі көп емес, жедел жадыны көп алмайды, сондықтан ескі компьютерлерде де қосуға болады.

- Кросс-платформалы, бірнеше операциялық жүйеде қолдануға болады.

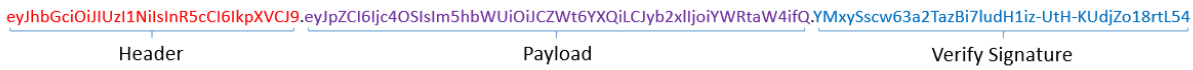
- Тестілеуге және кез-келген көлемдегі проектерге жақсы.

- Көптеген бағдарламалыу тілдеріне арналған кеңейтілімдері бар және оларды жылдам орнату.

2.7 JSON Web Token ақпаратты қауіпсіз жіберудің ықшам тәсілі

JSON Web Token (JWT) - бұл RFC 7519 ашық стандартында анықталған, JSON объектісі түрінде тараптар арасында ақпаратты қауіпсіз жіберудің ықшам және автономды тәсілін анықтайды. Көп жағдайда, клиент-серверлік қосымшаларда аутентификация деректерді жіберу үшін пайдаланылады. Екі қатысушы арасында ақпарат алмасудың қауіпсіз тәсілдерінің бірі ретінде саналады. Деректердің қауіпсіздігін жасау үшін, токеннің жалпы ақпараты бар тақырып (header), пайдаланушының id, оның рөлі және тағы басқа пайдалы

деректер(payload), және қолтаңба (signature) қажет(2.12-сурет). Таңбаларды сервер жасайды, құпия кілтпен қол қояды және осы белгіні одан әрі өзінің жеке басын растау үшін пайдаланады.



2.12 Сурет – JWT токені және оның құрылысы

Тақырыбы(header) әдетте екі бөліктен тұрады(2.13-сурет): токеннің типі және қолданылатын қолтаңбаның алгоритмі(HMAC SHA256 немесе RSA).

```
1
2  {
3    "typ": "JWT",
4    "alg": "HS256"
5  }
6
```

2.13 Сурет – JWT токеннің header бөлігі.

Пайдалы жүктеме(payload) токеннің екінші бөлігі, ішінде өтінімдер(claims) қамтиды. Өтінімдердің стандартты түрлері бар, олардың бірнеше түрі:

- iss (issuer) - токен жіберілетін қосымшаны анықтайды;
- exp (expiration time) - токеннің қолдану уақыты;
- sub (subject) - токеннің тақырыбын анықтайды;
- iat (issued at) – токенді құру уақыты;
- aud (audience) – токенді алатын аудитория;
- jti (JWT id) – токеннің идентификаторы.

Бұл стандартты өтінімдерді қолдану міндетті емес, олар токенді құрғанда, пайдаланданда пайдалы болу мүмкін. Өтінімдердің санына шектеу жоқ, бірақ өтінім көп болған сайын, JWT салмағыда артады, бұл кейін сервермен байланыс уақытын ұзарту мүмкін(2.14-сурет).

```
1  {
2    "sub": "user789",
3    "id": "789",
4    "name": "Bekzat",
5    "role": "admin",
6  }
```

2.14 Сурет – JWT токеннің пайдалы жүктеме бөлігі.

Қолтаңба (signature) тақырыппен пайдалы жүктеме бөліктері арқылы құралады. Соның арқасында егер токендегі ақпарат өзгерген жағдайда, ол қолтаңбаны кілтсіз өзгерте алмайды. Бірінші, тақырып және пайдалы жүктеме JSON форматынан base64 – ке аударылады. Екінші, тақырыпты және пайдалы жүктемені нүкте арқылы қосып, оны тақырыпта берілген алгоритм бойынша, жабық кілтімен хештайды. Бұдан алынған нәтиже қолтаңба болып табылады.

3 Программалық қамтаманы құру

3.1 Программалық қамтаманың құрылымы

Жоба фронт бөлігі React кітапханасымен пайдаланылып бөлек API ретінде жазылды, серверлік жақ Go тілінде жазылды, сервер міліметтер базасымен байланысты және ол Postgresql мәліметтер базасын басқару жүйесі қолданылды(3.1-сурет).

3.1 Сурет – Программалық қамтаманың құрылымы

Серверлік бөлік пен клиенттік бөлік json форматында ақпаратты алмасу арқылы қарым-қатынас жасайды.

3.2 Программаның баяндалуы

3.2.1 Жалпы мағлұматтар

Бұл жоба арқылы адамдар бір-бірімен чат, бейне байланыс арқылы қарым-қатынас жасай алады. Қолданушы, бірінші тіркеліп, басқа қолданушымен түрлі ойындар ойнай алады. Қазіргі сандық заманда, ақпарат үлкен рөл атқаруда, сол себепті қолданушының жеке ақпараты берікті әдіспен шифрланады. Ол ақпаратты сервер дешифрлап, соны ары қарай қолданады.

3.2.2 Функционалдық тағайындалуы

Жобаның серверлік бөлігі Go (тағы да оны Golang депте атайды) тілінде жазылды, клиенттік интерфейс бөлімі javascript кітапханасы React-та қолданылып жазылды. Қолданушы тіркеле алады, аутентификация jwt токені арқылы жүреді. Қолданушы кіргенде және оның мәліметтері базадағы сол

қолданушының сақталған мәліметімен сәйкес келсе токен құрылып, оны қолданушыға тіркейді.

Үлестірілген қосымшаларды байланыстыратын Rest архитектуралық стилде байланыс жасалған және ақпараттар json форматында жіберіледі. Алмасатын мәліметтер jwt арқылы шифрланып, қауіпсіздікті қамтамасыз етеді.

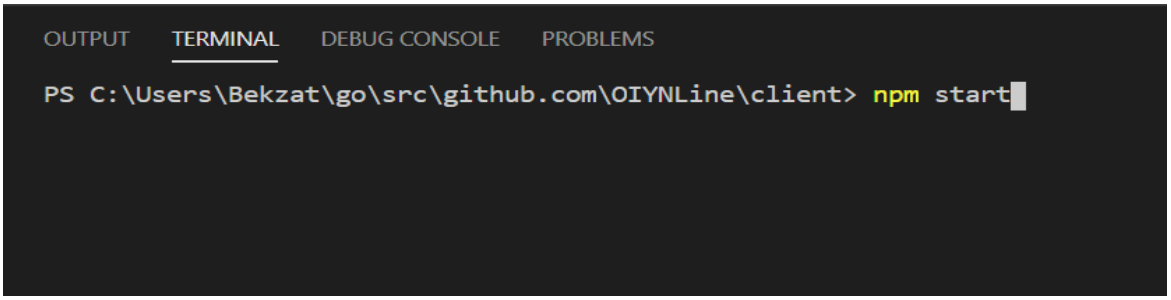
3.2.3 Қолданылған техникалық жабдықтар

Бұл Дипломдық жобаны жазғандуға келесі құралдар қолданылды:

- SPU Intel core i5 9700k;
- RAM 8 GB DDR4;
- SSD 500GB;
- Geforce GTX 1050TI;
- Алынбалы флэш-жинақтағыш 32 ГБ USB 3.1;
- Принтер Samsung dj 940c;
- Стандартты пернетақта мен тінтуір.

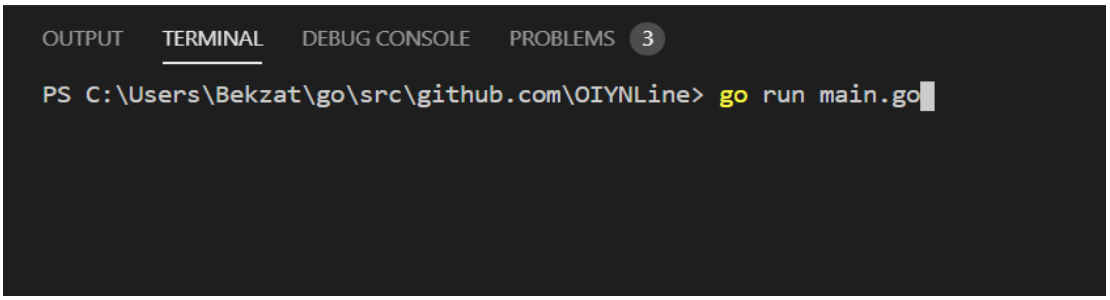
3.2.4 Шақыру және жүктеу

Қосымшаны шақыру үшін, бірінші консольда React кітапханасында жазылған клиенттік API-ді “npm start” командасы арқылы шақырамыз, ол үшін nodejs орнатылуы қажет(3.2-сурет).



3.2 Сурет – npm start командасы

Екіншіден, Go тілінде жазылған серверді шақырамыз(3.3-сурет).



3.3 Сурет – go run командасы

3.2.5 Кіріс мәліметтер

Кіріс мәліметтер ретінде, қолданушының ақпараттары, яғни қолданушыларды тіркеу, оларды өзгерту және кіріс мәліметіне ойынды тіркеу, оның ережесін, атын, тағы сипаттамасын жатқызамыз.

3.2.6 Шығыс мәліметтер

Қолданушы туралы ақпараттар жатады, оның осы кезге дейінгі жеткен талдаудан өткізілген нәтижелері жатады. Оның өзіне сақтап қойған ойындары жатады.

3.3 Single page application(SPA) құру

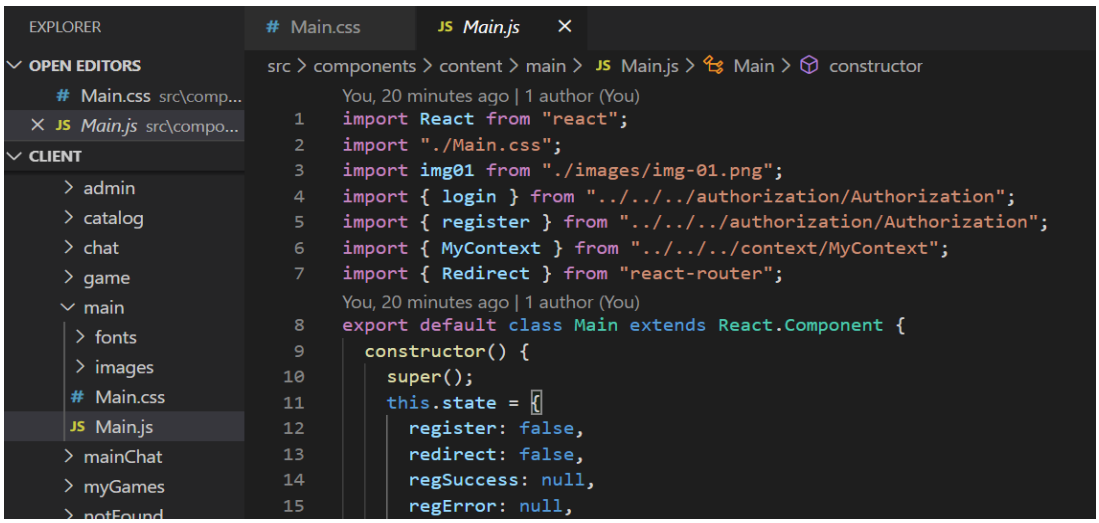
Single page application(SPA) - бұл web-қосымшалардың бір түрі, онда қажетті код бір HTML бетті пайдаланып, керек элементтерді динамикалық ауыстырады, элементтерді қайта жүктеу уақытын үнемдеуге мүмкіндік береді.

React кітапханасында SPA құрі үшін create-react-app командасын консольға теру арқылы біз жобаны құрамыз(3.4-сурет). Бұл құрал JavaScript жаңа мүмкіндіктерін пайдалану үшін ортаны баптайды, продакшен қосымшасын оңтайландырады және әзірлеу кезінде жайлылықты қамтамасыз етеді. Сізде Nodejs-тің нұсқасы 8.10 төмен болмасу қажет және npm 5.6 нұсқасынан төмен болмауы керек.

```
npx create-react-app my-app
cd my-app
npm start
```

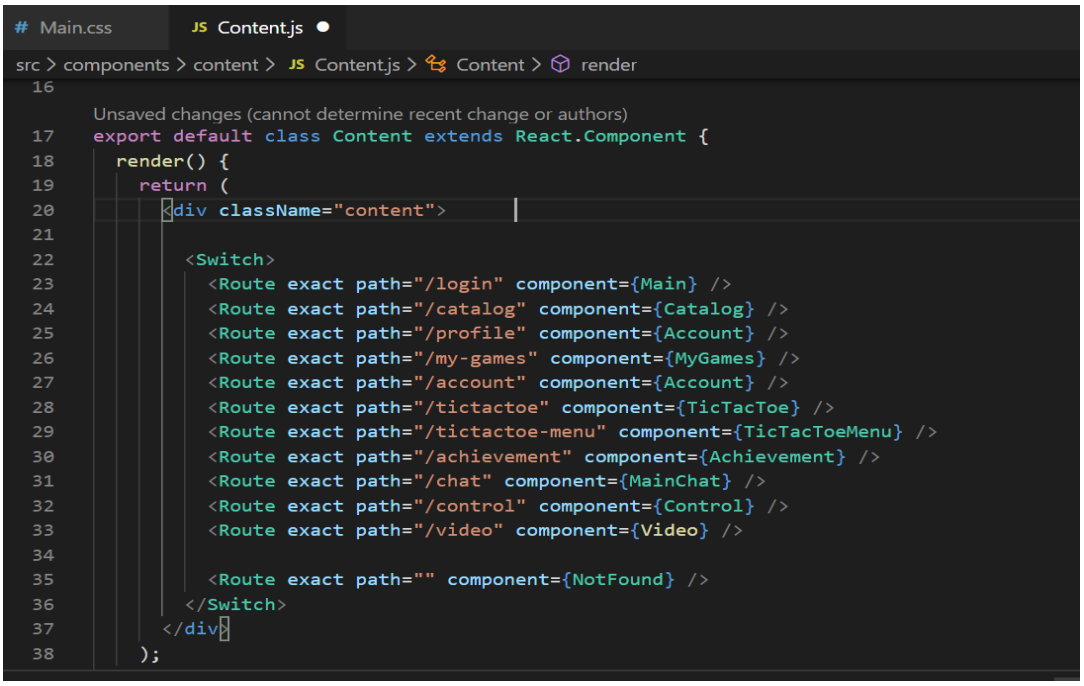
3.4 Сурет – Create-react-app командасын

Құрылған жобаға src директориясының ішінен components деген папка ашамыз, кейін бұл папкада құрылатын барлық компоненттер орналасады. Components директориясына main папкасын құрып, оның ішіне Main.js файлын құрамыз. Сол файлдың ішіне керек элементтерді жазамыз(3.5-сурет).



3.5 Сурет – Javascript файл Main.js

Әр түрлі компоненттің көрсетілуін қамтамасыз ету үшін "Switch" және "BrowserRouter" модульдерін пайдаланамыз және ол үшін браузерде қолданушы енгізетін жол бойынша осы жолда көрсетілуі тиіс компонентті көрсету керек(3.6-сурет)



3.6 Сурет – Javascript файлы Content.js

Браузерде шығатын элементтерді жазу үшін, біз jsx синтаксисін пайдаланамыз. Бұл синтаксис тің құрылысы HTML-ге ұқсас келеді және бұндай тәсіл адамдарға ыңғайлы болып келеді. Бұл React-тың ерекшеліктерінің бірі. Келесі файлға шығатын элементті жазамыз(3.7-сурет).

```
return (
  <div class="limiter">
    {this.Red()}
    <div class="container-login100">
      <div class="wrap-login100">
        <div class="login100-pic js-tilt" data-tilt>
          <img src={img01} alt="IMG" />
        </div>

        <form…

        <form…
      </div>
    </div>
  </div>
);
```

3.7 Сурет – Main компонентінің қайтаратын элементі

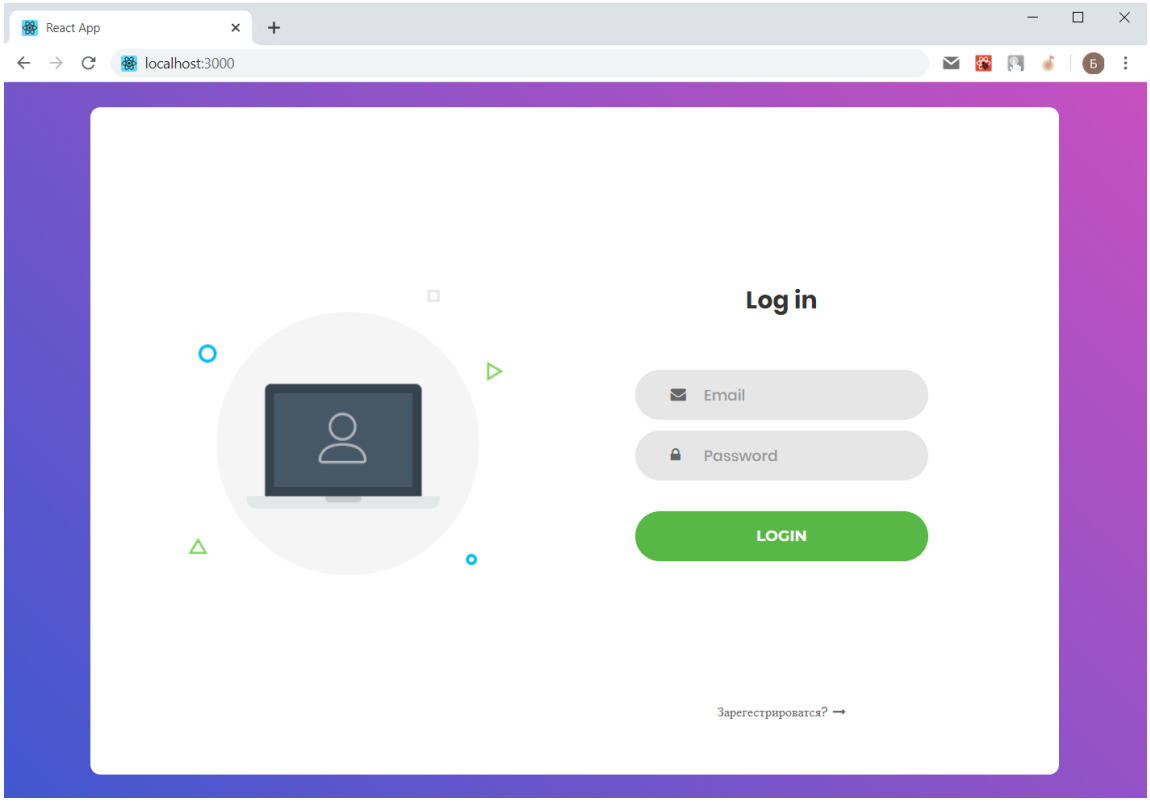
Жобаны қосу үшін бізге консольға немесе powershell-ге “npm start” командасы арқылы шақырамыз(3.8-сурет).

```
PS C:\Users\Bekzat\go\src\github.com\OIYNLine\client> npm start

> client@0.1.0 start C:\Users\Bekzat\go\src\github.com\OIYNLine\client
> react-scripts start && go run ../main.go
```

3.8 Сурет – Npm start командасы арқылы жобаны жүргізу

Жоба http: //localhost:3000 қосылады. Осы “npm start” командасынан кейін, бізге баптаутайларда үндеместен тұрған браузер ашылады(3.9-сурет).



3.9 Сурет – Веб-қосымшаның кіру беті

Браузерде Route – те турған басты компонент ашылады. Әр жолға өзіндік шығаратын компонент жазылған. Егерде қолданушы дұрыс емес компонент жазып жатса, онда үндеместен барлық табылмаған жолдарға, “NotFound” компоненті ашылады, ол компонент қолданушымен терілген адрессте ешқандай бет табылмағандығын айтады.

ҚОРЫТЫНДЫ

Интернет желісі күннен-күнге үздіксіз даму үрдісінде, жыл сайын адамдардан немесе бизнестерден жаңа талаптар, жаңа ұсыныстар пайда болумен қатар, сонымен бірге жаңа технологиялар қарқын алуда. Қазіргі өзекті болып тұрған ұсыныстар қатарына, нақты уақыт қосымшалар жатқызуға болады.

Адамдар әрдайым басқа адамдармен қарым-қатынас жасайды, ол жұмыс бойынша, мұғаліммен оқушы арасында, болмаса ойын-сауық ретінде ойын ойнау арқылы қарым-қатынас болу мүмкін. Интернет желісі бұл адамдардың қажеттілігін елемей қоймайды. Бұны біз чат қосымшалары, аудиохабарламалар, видео байланыстар видеоконференциялар, вебинарлар, онлайн ойын ойнау арқылы және тағы басқа байланыс әдістері жасалған. Осы қарым-қатынастарды интернет желісінде жасау мүмкіндігі болу үшін, бізге жаңа байланыс тәсілдері керек, өйткені адаммен адам бір-бірімен тек тікелей ғана байланысады. Бұл байланыстарды нақты уақыт аралығында жұмыс жасауға мүмкіндік беретін технологияларды пайдаланады. Бұл технологиялардың өзіндік артықшылықтары мен кемшіліктері бар.

Бұл жобада осы нақты уақытта жұмыс жасауға мүмкіндік беретін қазіргі заманауи өзекті технологияларды пайдаландым. Пайдаланған технологиялар заманауи браузерлерді қолдайды және ол ұялы телефонменде байланыса алады. Қолданушылар ойын ойнау арқылыда байланыс құра алады. Жоба өзінің өзектілігін, қазіргі интернет желісіндегі қолдаушылардың сұранысымен дәлелдеді.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1 https://ru.reactjs.org/docs/getting-started.html

2 https://golang.org/doc/

3 https://medium.com/@madhanganesh/golang-react-application-2aaf3bca92b1

4 Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Webсайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. - М.: Триумф, 2018. - 464 с

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

6 Alex B., Eve P. Learning React: Functional Web Development / OReilly – 2017. – P. 164-300.

7 Prathamesh S. ReactJS by Example - Building Modern Web Applications with React / OReilly – 2016. – 125 p.

8 Caleb D. Introducing Go - Build Reliable, Scalable Programs / O'Reilly Media – 2016-62 p.

9 Alan A., Brian W. The Go Programming Language / Addison-Wesley – 2015 -380p.

А қосымшасы

App.js

```
import React from "react";
import "./App.css";
import Menu from "./components/menu/Menu";
import Content from "./components/content/Content";
import { MyContext } from "./context/MyContext";
import { BrowserRouter as Router, Link } from "react-router-dom";
import { logout } from "./authorization/Authorization";
import { Redirect } from "react-router";
import Main from "./components/content/main/Main";

export default class App extends React.Component {
  constructor() {
    super();
    this.state = {
      Auth: {
        isAuthenticated: false,
        user: null,
      },
      setAuth: this.setAuth,
    };
    if (localStorage.getItem("token")) {
      this.state = {
        Auth: {
          isAuthenticated: true,
          user: localStorage.getItem("user_data"),
        },
        setAuth: this.setAuth,
      };
    }
  }

  setAuth = (dis) => {
    switch (dis) {
      case "LOGIN":
        console.log("LOGIN swith");
        this.setState({
          Auth: {
            isAuthenticated: true,
            user: localStorage.getItem("user_data"),
          },
        });
        return;
```

А қосымшасының жалғасы

```
      case "LOGOUT":
        this.setState({
          Auth: {
            isAuthenticated: false,
            user: null,
          },
        });
        return;
      default:
        this.setState({
          Auth: {
            isAuthenticated: false,
            user: null,
          },
        });
        return;
    }
  };

  lout = () => {
    logout(this.state.setAuth);
    this.setState({ redirect: true });
  };
  redirectFunc = () => {
    if (this.state.redirect) return <Redirect to="/" />;
  };

  render() {
    return (
      <MyContext.Provider value={this.state}>
        <Router>
          {!this.state.Auth.isAuthenticated ? (
            <Main />
          ) : (
            <div className="">
              {this.redirectFunc()}
              <div className="wrapper ">
                <Menu />

                <div className="main-panel">
                  <nav className="navbar navbar-expand-lg navbar-
transparent navbar-absolute fixed-top ">
```

А қосымшасының жалғасы

```
<div className="container-fluid">
 <div className="navbar-wrapper">
  <a className="navbar-brand" href="javascript:;">
   User Profile
  </a>
 </div>
 <button
  className="navbar-toggler"
  type="button"
  data-toggle="collapse"
  aria-controls="navigation-index"
  aria-expanded="false"
  aria-label="Toggle navigation"
 >
  <span className="sr-only">Toggle navigation</span>
  <span className="navbar-toggler-icon icon-bar"></span>
  <span className="navbar-toggler-icon icon-bar"></span>
  <span className="navbar-toggler-icon icon-bar"></span>
 </button>
 <div className="collapse navbar-collapse justify-content-end">
  <ul className="navbar-nav">
   <li className="nav-item dropdown">
    <a
     className="nav-link"
     href="javascript:;"
     id="navbarDropdownProfile"
     data-toggle="dropdown"
     aria-haspopup="true"
     aria-expanded="false"
    >
     <i className="material-icons">person</i>
     <p className="d-lg-none d-md-block">Account</p>
    </a>
    <div
     className="dropdown-menu dropdown-menu-right"
     aria-labelledby="navbarDropdownProfile"
    >
     <Link className="dropdown-item" to="/profile">
      Profile
     </Link>

     <div className="dropdown-divider"></div>
```

А қосымшасының жалғасы

```
                    <a
                      className="dropdown-item"
                      href="#"
                      onClick={this.lout}
                    >
                      Log out
                    </a>
                  </div>
                </li>
              </ul>
            </div>
          </div>
        </nav>

        <Content />
      </div>
    </div>
  </div>
 )}
 </Router>
 </MyContext.Provider>
 );
 }
}
```

Main.js

```
import React from "react";
import "./Main.css";
import img01 from "./images/img-01.png";
import { login } from "../../../authorization/Authorization";
import { register } from "../../../authorization/Authorization";
import { MyContext } from "../../../context/MyContext";
import { Redirect } from "react-router";
export default class Main extends React.Component {
 constructor() {
  super();
  this.state = {
   register: false,
   redirect: false,
   regSuccess: null,
   regError: null,
   email: "",
```

```
    password: "",
    password2: "",
    username: "",
    name: "",
    lastname: "",
    passwordError: "",
    //logError: "",
  };
}
register = () => {
  if (this.state.register) {
    this.setState({ register: false });
  } else {
    this.setState({ register: true });
  }
};

static contextType = MyContext;

registerSubmit = (e) => {
  e.preventDefault();
  if (this.state.password !== this.state.password2) {
    this.setState({
      passwordError: "Пароль не совпадают",
      regSuccess: false,
      regError: false,
    });
    return;
  } else {
    this.setState({ passwordError: "" });
  }
  let reg = {
    email: this.state.email,
    password: this.state.password,
    username: this.state.username,
    name: this.state.name,
    lastname: this.state.lastname,
  };
  register(reg, this.regResult);
};
regResult = (res) => {
  switch (res) {
```

А қосымшасының жалғасы

```
    case "SUCCESS":
      console.log("switch suc");
      this.setState({
        regSuccess: "регистрация прошла успешно",
        regError: null,
        email: "",
        password: "",
        password2: "",
        username: "",
        name: "",
        lastname: "",
        passwordError: "",
      });
      return;
    case "ERROR":
      console.log("switch err");
      this.setState({
        regError: "email или username существует",
        regSuccess: null,
        passwordError: "",
      });
      return;
    default:
      this.setState({
        regError: "Error",
        regSuccess: null,
        passwordError: "",
      });
      return;
  }
};

logResult = (res) => {
  if (res === "ERROR") {
    this.setState({ logError: "неверный логин или пароль" });
  } else {
    this.setState({ logError: "" });
    this.setState({ redirect: true });
  }
};
loginSubmit = (e) => {
  e.preventDefault();
```

А қосымшасының жалғасы

```
  let email = e.target.elements["email"].value;
  let pass = e.target.elements["password"].value;
  let reg = {
    email: email,
    password: pass,
  };
  login(reg, this.context.setAuth, this.logResult);
};

Red = () => {
  if (this.state.redirect) {
    return <Redirect to="/account" />;
  }
};

changeInput = (e) => {
  this.setState({ [e.target.name]: e.target.value });
};
render() {
  const reg = !this.state.register
    ? { display: "none" }
    : { display: "block" };
  const log = !this.state.register
    ? { display: "block" }
    : { display: "none" };
  return (
    <div class="limiter">
      {this.Red()}
      <div class="container-login100">
        <div class="wrap-login100">
          <div class="login100-pic js-tilt" data-tilt>
            <img src={img01} alt="IMG" />
          </div>

          <form
            class="login100-form validate-form"
            style={log}
            onSubmit={this.loginSubmit}
            action="/login"
          >
            <span class="login100-form-title">Log in</span>
```

А қосымшасының жалғасы

```
<div
  id="logform"
  class="wrap-input100 validate-input"
  data-validate="Valid email is required: ex@abc.xyz"
>
  <input
    class="input100"
    type="text"
    name="email"
    placeholder="Email"
  />
  <span class="focus-input100"></span>
  <span class="symbol-input100">
    <i class="fa fa-envelope" aria-hidden="true"></i>
  </span>
</div>

<div
  id="logpass"
  class="wrap-input100 validate-input "
  data-validate="Password is wrong"
>
  <input
    class="input100"
    type="password"
    name="password"
    placeholder="Password"
  />
  <span class="focus-input100"></span>
  <span class="symbol-input100">
    <i class="fa fa-lock" aria-hidden="true"></i>
  </span>
</div>

<div class="container-login100-form-btn">
  <button class="login100-form-btn">Login</button>
</div>
{this.state.logError ? (
  <p style={{ color: "red" }}>{this.state.logError}</p>
) : (
  ""
)}
```

А қосымшасының жалғасы

```
  <div class="text-center p-t-136">
    <a class="txt2" href="#" onClick={this.register}>
      Зарегестрироватся?
      <i
        class="fa fa-long-arrow-right m-l-5"
        aria-hidden="true"
      ></i>
    </a>
  </div>
</form>

<form
  class="login100-form validate-form"
  style={reg}
  action="/register"
  onSubmit={this.registerSubmit}
>
  <span class="login100-form-title">Sign in</span>

  <div
    class="wrap-input100 validate-input"
    data-validate="Valid email is required: ex@abc.xyz"
  >
    <input
      class="input100"
      type="text"
      name="email"
      placeholder="Email"
      onChange={this.changeInput}
    />
    <span class="focus-input100"></span>
    <span class="symbol-input100">
      <i class="fa fa-envelope" aria-hidden="true"></i>
    </span>
  </div>

  <div
    class="wrap-input100 validate-input"
    data-validate="Valid email is required: ex@abc.xyz"
  >
    <input
      class="input100"
```

```
    type="text"
    name="username"
    placeholder="Username"
    onChange={this.changeInput}
   />
   <span class="focus-input100"></span>
   <span class="symbol-input100">
    <i class="fa fa-address-book" aria-hidden="true"></i>
   </span>
  </div>

  <div
   class="wrap-input100 validate-input"
   data-validate="Valid email is required: ex@abc.xyz"
  >
   <input
    class="input100"
    type="text"
    name="name"
    placeholder="name"
    onChange={this.changeInput}
   />
   <span class="focus-input100"></span>
   <span class="symbol-input100">
    <i class="fa fa-user-circle" aria-hidden="true"></i>
   </span>
  </div>
  <div
   class="wrap-input100 validate-input"
   data-validate="Valid email is required: ex@abc.xyz"
  >
   <input
    class="input100"
    type="text"
    name="lastname"
    placeholder="Last name"
    onChange={this.changeInput}
   />
   <span class="focus-input100"></span>
   <span class="symbol-input100">
    <i class="fa fa-user-circle" aria-hidden="true"></i>
   </span>
```

А қосымшасының жалғасы

```
</div>
<div
  class="wrap-input100 validate-input"
  data-validate="Password is required"
>
  <input
    class="input100"
    type="password"
    name="password"
    placeholder="Password"
    onChange={this.changeInput}
  />
  <span class="focus-input100"></span>
  <span class="symbol-input100">
    <i class="fa fa-lock" aria-hidden="true"></i>
  </span>
</div>
<div
  class="wrap-input100 validate-input"
  data-validate="Password is required"
>
  <input
    class="input100"
    type="password"
    name="password2"
    placeholder="Password"
    onChange={this.changeInput}
  />
  <span class="focus-input100"></span>
  <span class="symbol-input100">
    <i class="fa fa-lock" aria-hidden="true"></i>
  </span>
</div>
{this.state.regSuccess ? (
  <p className="success">{this.state.regSuccess}</p>
) : (
  ""
)}
{this.state.regError ? (
  <p className="error">{this.state.regError}</p>
) : (
  ""
```

А қосымшасының жалғасы

```
            )}
            {this.state.passwordError ? (
              <p className="error">{this.state.passwordError}</p>
            ) : (
              ""
            )}
            <div class="container-login100-form-btn">
              <button class="login100-form-btn">Sign in</button>
            </div>
            <div class="text-center p-t-136">
              <a class="txt2" href="#" onClick={this.register}>
                Есть аккаунт?
                <i
                  class="fa fa-long-arrow-right m-l-5"
                  aria-hidden="true"
                ></i>
              </a>
            </div>
          </form>
        </div>
      </div>
    </div>
  );
 }
}
```