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

Сұнғатов Ғалымжан Тимурұлы

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

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: «Нақты уақыт қосымшаларына арналған платформа»

5В070300 – «Ақпараттық жүйелер» мамандығы

Орындаған Сұңғатов Ғ.Т.

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

Ақпараттық және телекоммуникациялық технологиялар институты

Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

5В070300 – «Ақпараттық жүйелер» мамандығы

БЕКІТЕМІН
КАӨжС кафедра меңгерушісі,
тех.ғыл.канд, ассоц.
профессор
_____ Н.А.Сейлова
«___»_____2020 ж.

Дипломдық жобаны орындауға
ТАПСЫРМА

Білім алушы: Сұңғатов Ғалымжан Тимурұлы
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 жобаның 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 жобаны 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 жоба 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 _____ Сұңғатов Ғ.Т.

Күні «27» қаңтар 2020 ж.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Сәтбаев университеті

Ғылыми жетекшінің пікірі

Дипломдық жұмыс

Сұнғатов Ғалымжан Тимуұлы

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 Сұнғатов Ғалымжан Тимуұлы 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: Сұңғатов Ғ.Т

Название: Платформа для приложении реального времени

Координатор:Ерсұлтан Раскұлбек

Коэффициент подобия 1:0,4

Коэффициент подобия 2:0

Замена букв:0

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: Сұңғатов Ғ.Т

Название: Платформа для приложении реального времени

Координатор: Ерсұлтан Раскұлбек

Коэффициент подобия 1:0,4

Коэффициент подобия 2:0

Замена букв:0

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

Дипломдық жобаның мақсаты – жаңа технологияларды зерттеу, сол технологияларды қолданып, нақты уақыт қосымшаларына арналған платформаның прототипін жасау жцне мәліметтер базасын жобалау.

Қойылған мақсатқа жету үшін келесі міндеттер орындалуы тиіс:

– тақырыпты зерттеу;

– жасалатын жобаға ұқсас жобаларды талдау;

– мәліметтер базасын жобалау;

– жоба құру үшін қолданылатын құралдарды анықтау және оларды қолдану;

– жобаның прототипін құру.

Жобаны құру үшін WebSocket протоколы, Go бағдарламалау тілі, PostgreSQL мәліметтер базасы және Visual Studio Code бағдарламасы қолданылды.

АННОТАЦИЯ

Целью этой дипломного проекта является изучение новых технологий, использование этих технологий для создания прототипа платформы для приложений реального времени и разработка базы данных.

Для достижения поставленной цели необходимо выполнить следующие задачи:

– исследовать тему;

– анализ подобных проектов;

– разработка базы данных;

– определить и использовать инструменты, для создания проекта;

– разработка прототипа проекта.

Для создания проекта использовались протокол WebSocket, язык программирования Go, база данных PostgreSQL и программа Visual Studio Code.

ANNOTATION

The goal of this diploma project is to study new technologies, use these technologies to create a prototype platform for real-time applications, and develop a database.

Objectives of the diploma project:

– explore the topic;

– analysis of such projects;

– database development;

– identify and use tools to create a project;

– development of a prototype project.

To create the project we used the WebSocket protocol, the Go programming language, the PostgreSQL database and the Visual Studio Code program.

МАЗМҰНЫ

КІРІСПЕ

Қазіргі заманда ғылыми техникалық революция, жаңа ғылыми пәндердің және жаңа техниканың көмегімен қиын мәселелер шешіледі және адам өмірін жеңілдетеді.

Қазіргі таңда адамдар кез келген мәселелерді интернет желісі арқылы шеше алады. Және де кез келген жер шарының басқа бөлігінде орналасқан адаммен байланыс жасай алады. Мысалы, чат, аудио және видео конференция, онлайн ойындар және т.б.

Веб-қосымшалар бастапқыда клиент/сервер үлгісінде жасалды, мұнда веб-клиент әрдайым серверден мәліметтерді сұрайтын транзакциялардың бастамашысы болып табылады. Осылайша, сервердің клиенттен алдымен сұрау салмай-ақ, деректерді клиентке дербес жіберу немесе итермелеу тетігі болған жоқ. Бұл кемшілікті жою үшін веб-қосымшаны жасаушылар нақты уақытта қолданылатын әдістерді қолдана басады, мұнда клиент жаңа ақпаратты сұрайтын серверге сұрақ қояды. Сервер жаңа деректер қол жетімді болғанша сұранысты ашық ұстайды. Қол жетімді болғаннан кейін сервер жауап береді және жаңа ақпаратты жібереді. Клиент жаңа ақпаратты алған кезде, ол бірден басқа сұраныс жібереді, және операция қайталанады. Бұл серверді басу мүмкіндігін тиімді түрде шығарады. Осындай байланысты ұйымдастырудың бірнешет түрі бар:

– HTTP Long-Polling – сервермен жаңа ақпарат келгенше байланыста болады.

– WebSockets – сервермен клиент арасындағы байланыс тек ақпарат келген кезде ғана ұйымдастырылады.

Осы дипломдық жобаны жасаудағы алға қойылған мақсат – жоғарыда айтылған технологияларды зерттеу, сол технологияларды қолданып, нақты уақыт қосымшаларына арналған платформаның прототипін жасау, мәліметтер базасын жобалау.

1 Нақты уақыт қосымшаларына арналған платформаларды зерттеу

1.1 Нақты уақыт қосымшасы ұғымына жалпы шолу

Нақты уақыт қосымша(ағылшын RTA – real-time application) – бұл дегеніміз қолданушы жедел немесе ағымдағы ретінде қабылдайтын уақыт шектерде жұмыс істейтін қолданбалы бағдарлама. Қосымшаның нақты уақыт қосымшасы ретінде анықталуы онын тапсырмаларды немесе тапсырмалар жиынтығын орындаудағы ең нашар уақытына байланысты.

Нақыт уақыт қосымшаларының қолдану аясы:

– Бейнеконференция қосымшалары.

– VoIP (Интернет-протокол арқылы байланыс).

– Онлайн ойындар.

– Қоғамдық сақтау шешімдері.

– Кейбір электрондық коммерциялық операциялар.

– Чат.

– Мессенджерлар.

Нақты уақыт қосымша - уақытты есептеу өте маңызды болып табылатын қосымша. Тағы бір «нақты уақыт» термині автоматтандырылған басқару жүйелерімен байланысты, олар адам жауап бере алмағаннан тезірек жауап беруі керек. Адамдардың көпшілігі оқиғаларға 100 миллисекундтан аз уақыт ішінде жауап бере алмайды. Көптеген басқару қосымшалары 100 миллисекундтан гөрі тез жауаптарды қажет етеді. Нақты уақыттағы қосымшалар бизнес жылдамдығын жақсартады, бірақ қашықтық өсіп, желінің толып кетуінің әсерін алған сайын олардың өнімділігі төмендейді. Деректерді жаңарту өте ұзақ уақытты алады. Виртуальды жұмыс үстелдері өте жай. Бейне мен дыбыс қол жетімді болмайды.

Нақты уақыттағы API механизмдеріне байланысты нақты уақыт қосымшаларын төртке бөлуге болады:

– HTTP streaming.

– HTTP long-polling.

– Webhooks.

– WebSockets.

Нақты уақыттағы байланыс - бұл бірінші деңгейлі архитектураны қолдана отырып, нақты уақыт режимінде деректерді тоқтаусыз алмасуға мүмкіндік беретін әдіс. Нақты уақыттағы байланыс жартылай кешенді (жартылай дуплексті) немесе толық кешенді (толық дуплексті) режимде жүзеге асырылады:

– жартылай дуплекс: бір бағыттағы байланыс. Бір уақытта жіберуші немесе қабылдаушы жұмыс істей алады;

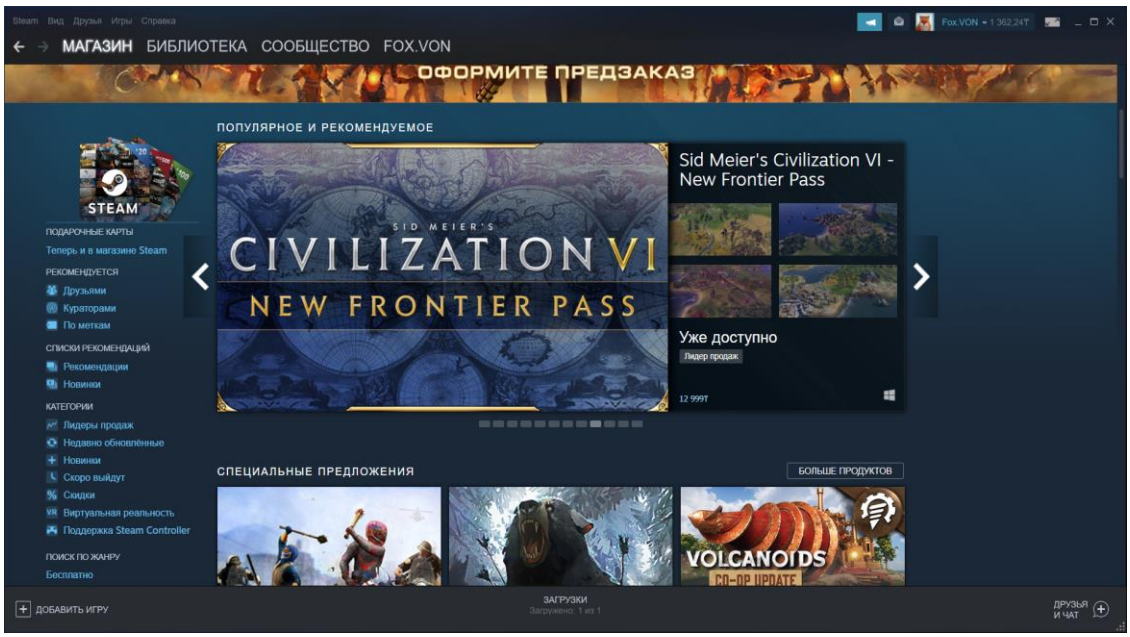
– толық дуплекс: жіберуші мен қабылаушы бір уақытта байланыс арналарының екі қатарынан хабарлама жібере және қабылдай алады.

1.2 Нақты уақыт қосымшаларына арналған платформаларды талдау

Қазіргі таңда қолданыстағы нақты уақыт қосымшаларына арналған платформалардың бірнеше мысалын зерттеп, талдайық. Олардың әрқайсысының ерекшеліктерімен кемшіліктерін қарастырайық.

1.2.1 “Steam” ойын тарату платформасы

“Steam” – Valve компаниясы әзірлеген және қолдайтын компьютерлік ойындар мен бағдарламалардың сандық көшірмесін таратын онлайн платформа. Осы платформа арқылы сіз сонда орналасқан кез келген ойынды жүктеп, ойнай аласыз. Сонымен қатар онда "Қоғамдастық", "Ұсыныстар", "Достар", 20 ГБ сақтау кеңістігі бар "Скриншоттар", чат тәрізді функциялары бар. Ондағы ойындар ақылы және тегін болып келеді. (1.1-сурет).



1.1 Сурет – “Steam” платформасының интерфейсі

Бұл платформа C++ программалау тілін пайдаланады. Ойынды жүктеу үшін клиенттік қосымшаны жүктеуді қажет етеді және кез келген ойынды ойнау үшін де оны жүктеу қажет болады.

1.3 Есептің қойылымы

Нақты уақытта қосымшаларына платформа құру үшін келесі міндеттер қойылып, шешіледі:

– Нақты уақыт қосымшасына қажетті технологияларды зерттеу және оларға талдау жасау қажет;

– Серверлік жағын құру үшін ең тиімді және замануи технологиялар қолдану;

– Қолданушылар арасында байланыс ұйымдастыру;

– Байланысты тиімді ұйымдастыру;

– Байланыс кезіндегі қауіпсіз ақпарат алмасуды ұйымдастыру.

Дипломдық жобаны құру кезінде Go тілі серверге, PostgreSQL мәліметтерді базасын басқару жүйесі қолданылды.

2 Программалау тілін таңдауды негіздеу

2.1 Visual Studio Code бастапқы код редакторы

Visual Studio Code – жеңіл, әрі қуатты кроссплатформалы бастапқы код редакторы. Сонымен қатар ол JavaScript, TypeScript және Node.js қолдайды және басқа тілдерге арналған кеңейтімдерге бай экожүйеге ие (мысалы, C ++, C #, Python, PHP, Go).

Visual Studio Code мынадай функионалдар қарастырылған:

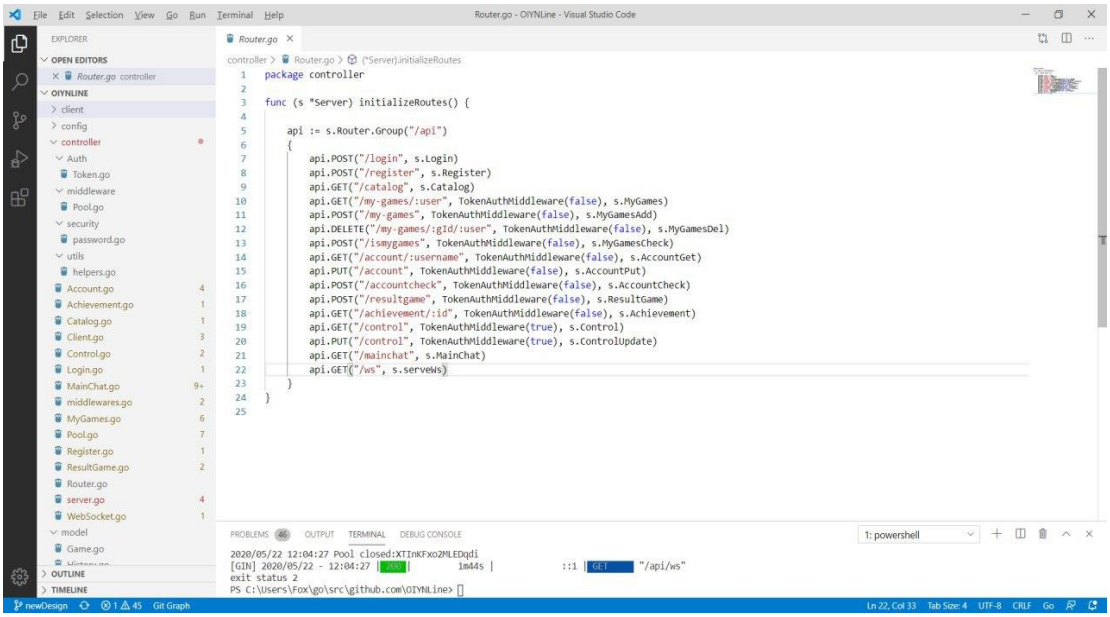
– ақылды кодты яқтау. IntelliSense көмегімен кодтағы айнымалы мәндерді, әдістерді және импортталған модульдерді автоматты түрде аяқтау;

– жылдам, қуатты редакциялау. Код анализаторы, көп курсорды редакциялау, параметрлер кеңестері және басқа қуатты редакциялау мүмкіндіктері;

– кодтық навигация және рефакторинг. Мәзірге қарап анықтамаға өтіп, бастапқы кодты жылдам қарап шығу мүмкіндігі.

Visual Studio Code - бастапқы код редакторы. Ол бірқатар бағдарламалау тілдерін, синтаксисті бөлектеу, IntelliSense, рефакторлау, кодтық навигация, Git қолдауы және басқа мүмкіндіктерді қолдайды. Visual Studio Code-тың көптеген мүмкіндіктеріне графикалық интерфейс арқылы қол жеткізуге болмайды, олар көбінесе командалар палитрасы немесе JSON файлдары арқылы қолданылады(мысалы, қолданушы параметрлері). Пәрмендер палитрасы - бұл пернелер тіркесімі арқылы шақырылатын командалық жолдың ұқсастығы.

Visual Studio сонымен қатар құжатты, жаңа жол таңбаларын және ағымдағы құжаттың бағдарламалау тілін сақтаған кезде код парағын ауыстыруға мүмкіндік береді. 2.1 – суретке сәйкес Visual Studio Code пайдаланушылық ортасы мен интерфейсі көрсетілген.



2.1 Сурет – Visual Studio Code пайдаланушылық ортасы мен интерфейсі

2018 жылдан бастап Visual Studio Code үшін ашық Python кеңейтімі пайда болды. Ол өңдеушілерге кодты өңдеуге, күйін келтіруге және тестілеуге үлкен мүмкіндіктер ұсынады. 2019 жылдың наурыз айынан бастап, өнімнің кіріктірілген интерфейсі арқылы сіз бірнеше бағдарламаны тек қана «бағдарламалау тілдері» санатына жүктей және орната аласыз. Visual Studio Code - 30-дан астам бағдарламалау тілдерімен және файлдық форматтармен, соның ішінде GO, TypeScript, JavaScript-пен жұмыс істейтін код редакторы. Тек код редакторы ғана емес, сонымен бірге қосымша мүмкіндіктері бар пайдалы әзірлеуші құрал [5].

Visual Studio Code пайдалану деректерін (телеметрия) жинайды және оны Microsoft корпорациясына жібереді. Деректерді беру міндетті болып табылмаса да және сіз жеке деректерді беруден бас тартуыңыз мүмкін бар болса да, бірақ жекелендіру сияқты кейбір мүмкіндіктерде мұндай деректерді өшіру мүмкіншілігіне қол жетімді болмайды. Құпиялылық туралы мәлімдемеге сәйкес деректер Microsoft корпорациясының еншілес компанияларға және құқық қорғау органдарына берілуі мүмкін.

Visual Studio Code сипаттамалары:

– тегін ашық мәтіндік редактор;

– онда IntelliSense бар (бірақ ол орнатқаннан кейін бірден жұмыс істемейді, егер Visual Studio орнатылмаған болса, оны MinGW көрсету арқылы конфигурациялау керек және т.б.);

– жүктеудің мөлшері мен жедел жадының қажеттілігінің төмен болуы. IntelliSense шамамен 300 МБ жедел жады қажет етеді;

– кіші компьютерлерде жұмыс істейді. (іске қосу әлі баяу, әсіресе егер PowerShell CMD орнына қолданылса);

– төменгі қолдау (ашық бастапқы код, оны өзіңіз өзгерте аласыз);

– тапсырмаларды құру жобаға байланысты. Егер сіз оны ванильді конфигурацияда жасағыңыз келсе де;

– негізінен веб-әзірлеу үшін қолданылады (бұл барлық тегін мәтіндік редакторларға қатысты);

– жақсы кеңейтілімдердің болмауы;

– жоба/жұмыс кеңістігінің параметрлерін қайта конфигурациялау қиын;

– кросс-платформалық;

– онда орнатылған терминал бар (PowerShell іске қосылуда өте баяу);

– бұл кішігірім жобалар мен тест коды үшін жақсы.

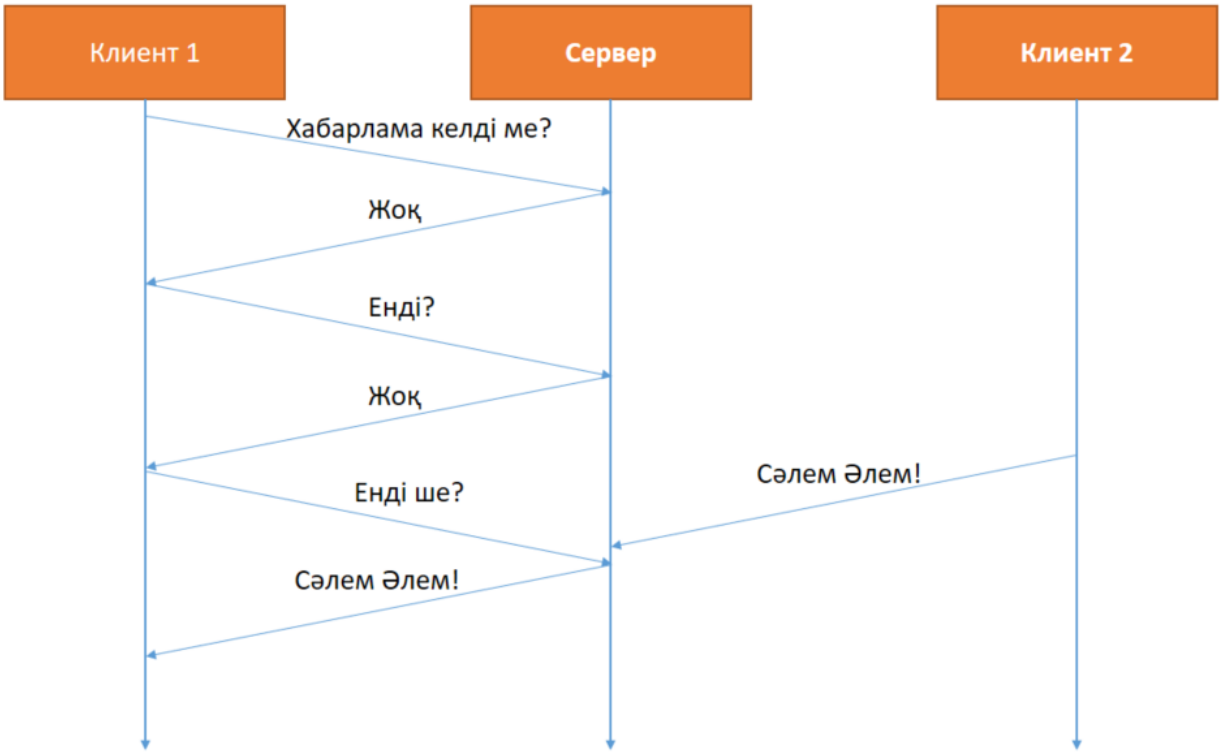
Visual Studio Code Windows, OS X және Linux жүйелерінде жұмыс істейтін компьютерлерде қолдануға болады. Құрал 2015 жылдың көктемінде шығарылды және үнемі жаңартылып отырды. Visual Studio Code өзінің қолданысын кеңейтіп, пайдаланушылардың пікірлері мен ұсыныстарына негізделген қолдау көрсетілетін тілдердің тізімін кеңейтті.

Редактор ашық бастапқы коды бар өнімдерге негізделген, кейде әзірлеушілер үшін маңызды критерий болып табылады, нұсқаны басқару жүйелерімен интеграцияны, кіріктірілген түзеткішті және сыртқы құралдарды қосу мүмкіндігін қолдайды [6].

2.2 WebSocket нақты уақыт кезінде ақпарат алмасу технологиясы

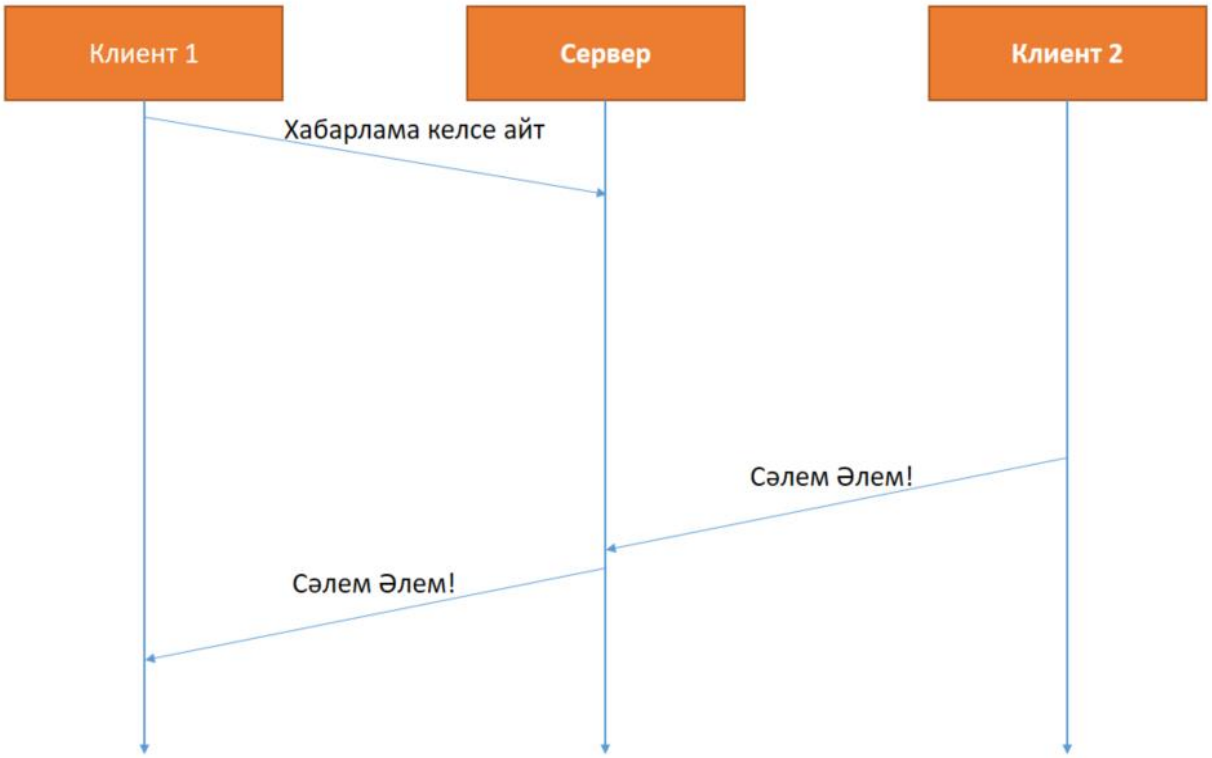
WebSocket - бұл TCP қосылымы бойынша байланыс протоколы, ол клиент пен сервер арасында тұрақты қосылысты қамтамасыз етеді және сол қосылыс арқылы екі жақ кез келген уақытта деректерді жіберуді бастайды.

WebSocket – тің HTTP-дан ең басты айырмашылығы екіжақты мәліметтер ағынымен жумыс істей білуі. Ал, HTTP байланысы арқылы жұмыс істейтін браузер серверден жаңа хабарлама бар ма деп үнемі сұрайды және оларды алады(2.2-сурет).



2.2 Сурет – HTTP протоколы бойынша хабарлама алмасу принципі

WebSocket көмегімен дәстүрлі HTTP сұрауларына байланысты шығындарсыз өзіңіз қалағаныңызша көбірек деректерді жібере аласыз. Деректер WebSocket арқылы хабарламалар түрінде жіберіледі, олардың әрқайсысы сіз жіберетін (жүктеме) мәліметтерді қамтитын бір немесе бірнеше кадрлардан тұрады. Хабарлама клиентке жеткен кезде оны дұрыс қайта құру мүмкіндігіне кепілдік беру үшін әр кадрға жүктеме туралы 4-12 байттан тұратын префикс беріледі. Бұл кадрға негізделген хабарламалар жүйесін пайдалану жүктеме жүктелмейтін деректердің көлемін азайтуға көмектеседі, бұл кідірістің едәуір қысқаруына әкеледі(2.3 – сурет).



2.3 Сурет – WebSocket протоколы бойынша хабарлама алмасу принципі

WebSocket жауап беру үшін қайталанатын сұраулардың қажеті жоқ. Бір сұрауды толтырып, жауап күту жеткілікті. Сіз жай ғана дайын болған кезде сізге хабарлама жіберетін серверді күтіп отырасыз.

WebSocket мына жағдайларда пайдалануға болады:

– нақты уақыттағы қосымшалар;

– чат қосымшалары;

– IoT қосымшалары;

– көп ойыншы ойындары.

HTTP тұрақты домендік сұраулардан айырмашылығы, WebSocket сұраулары бірдей шығу саясатымен шектелмейді. Сондықтан WebSocket серверлері «Cookies» немесе HTTP аутентификациясымен қосылым аутентификацияланған кезде мүмкін болуы мүмкін WebSocket Hijacking шабуылдарын болдырмау үшін (Сайттың сұранысын өшіру сияқты), қосылымды құру кезінде күтілетін бастауларға қарсы «Origin» тақырыбын тексеруі керек. WebSocket арқылы құпия (жеке) деректерді беру кезінде WebSocket қосылымын аутентификациялау үшін токендерді немесе ұқсас қорғаныс механизмдерін қолданған дұрыс.

WebSocket негізгі ерекшеліктері:

– WebSocket протоколы стандартталған, яғни осы протоколды қолдану арқылы веб-серверлер мен клиенттер арасында нақты уақыт режимінде байланыс орнатуға болады.

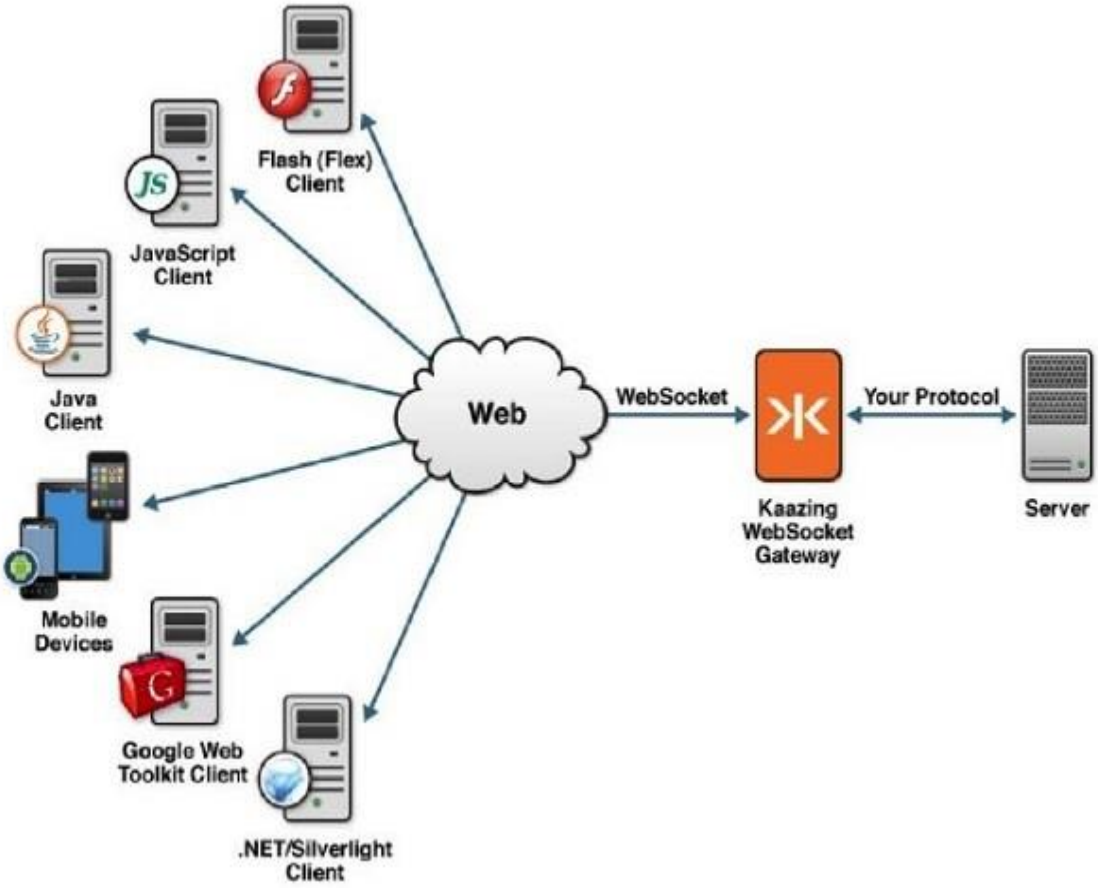
– WebSocket нақты уақыт режимінде клиент пен сервер арасында мәліметтер алмасу үшін кросс-платформалық стандартқа айналады.

– Бұл стандарт қосымшаның жаңа түрін алуға мүмкіндік береді.

– Web Socket-тің ең үлкен артықшылығы - бұл TCP қосылымы бойынша екі жақты байланыс (толық дуплекс).

Web Socket - бұл TCP-ге негізделген тәуелсіз протокол, бірақ ол дәстүрлі түрде тек таза TCP қосылымының жоғарғы жағында жұмыс істейтін кез-келген басқа протоколды қолдауға арналған.

Браузердің жалғыз талабы - WebSocket байланысын түсіндіретін, WebSocket қосылымын орнататын және қолдайтын JavaScript кітапханасын іске қосу. 2.4 суретке сәйкес WebSocket функционалдығы сипатталып, көрсетілген.



2.4 Сурет – WebSocket функционалдылығы

Web Socket - бұл төмен деңгейдегі хаттама. Барлығы, соның ішінде қарапайым сұрау/жауап шаблоны, ресурстарды қалай құру/жаңарту/жою, күй кодтары және т.б., негізінде құрылады. Олардың барлығы HTTP үшін жақсы анықталған.

Web Socket қосылымдары бір серверде тігінен масштабталуы мүмкін, ал HTTP көлденеңінен масштабталуы мүмкін. Web Socket көлденеңінен масштабтауға арналған бірнеше жеке шешімдер бар, бірақ олар стандарттарға негізделмеген. HTTP кэштеу, маршруттау және мультиплексинг сияқты көптеген басқа жақсылықтармен бірге келеді. Мұның бәрі Web Socket-тің жоғарғы жағында анықталуы керек.

2.3 Go – көп ағынды бағдарламалау тілі

Go, немесе Golang – бастапқы коды ашық бағдарламалау тілі. Ол статикалық түрде терілген және құрастырылған машиналық кодтың екілік файлдарын құрады. Бұл бағдарламалау тілі сізге жадыны қауіпсіз пайдалануға, объекттарды басқаруға, қоқыс жинауға және сәйкестікпен қатар статикалық (немесе қатаң) теруді қамтамасыз етуге мүмкіндік беретін құралдарды қамтиды.

Go тілінің пайда болуының негізгі мақсаты басқа бағдарламалау тілдерінің жақсы жақтарын біріктіру болды:

– алдын-ала мәлімдемелер және тақырып файлдары жоқ. Барлығы бір рет жарияланады;

– заманауи өнімділікпен бірге пайдалану жеңілдігі;

– статикалық терумен қатар жоғары деңгейлі тиімділік;

– желіге қосылу және көп ядролы қуатты толық пайдалану үшін жетілдірілген өнімділік;

2.5 суретте Go программалау тілінде код жазылуының ммысалы көрсетілген.

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Println("Hello, Satbayev University")
7  }
8
```

2.5 Сурет - Go тіліндегі қарапайым бағдарлама

Go тілінің артықшылықтары:

– Икемді, қысқа, қарапайым және оңай оқылады.

– Параллелизм. Бұл бірнеше процесті бір уақытта және тиімді орындауға мүмкіндік береді.

– Жылдам нәтиже. Оның құрастыру уақыты өте жылдам.

– Кітапхана. Ол бай стандартты кітапхана ұсынады.

– Қоқыс жинау. Бұл Go тілінің басты ерекшелігі. Жадтың бөлінуіне көп бақылау беріп, қоқыс жинағыштың соңғы нұсқаларында кідірісті азайтты.

– Ол интерфейс пен типті ендіруге жарамды.

– қосымшаны жасау үшін аз уақыт пен ақша жұмсайсыз. Егер сіз өзіңіздің жобаңыз үшін Go бағдарламасын пайдаланатын болсаңыз, сізге үлкен стек қажет емес. Go бағдарламасында жасалған бағдарламалар іс жүзінде машиналық кодтың кодын құрастырады және ешқандай интерпретатордың немесе виртуалды машинаның қажеті жоқ.

– Бағдарламаңыз үшін көбірек өнімділік пен кең аудиторияға ие болыңыз. C немесе C ++ сияқты, Go - бұл құрастырылған тіл және ешқандай түсіндіруді қажет етпейді. Тиісінше, интерпретатордың жоқтығы қуатты босатады және Go бағдарламасына қосымша мүмкіндік береді. Сонымен қатар, Go бөлінген жадыны қалай дұрыс басқаруды біледі.

Go тілінің басқаларынан басты ерекшеліктерінің бірі “goroutine” бар болуы. Гороутиндер параллель операцияларды білдіреді, олар орындалатын функцияға қарамастан орындалады. Гороутиндердің басты ерекшелігі - оларды параллель орындауға болады. Яғни, көп ядролы архитектурада бөлек процессордың өзектерінде бөлек горутиндерді орындауға болады, осылайша гороутиндер параллель орындалады және бағдарлама тезірек аяқталады. Әр горутин, әдетте, функционалды шақыруды білдіреді және оның барлық нұсқаулықтарын дәйекті түрде орындайды. Біз Go бағдарламасын іске қосқан кезде, біз ең аз дегенде бір горутинмен жұмыс істеп жатырмыз, ол main функциямен ұсынылған. Бұл функция оның ішінде анықталған барлық нұсқауларды дәйекті түрде орындайды(2.6-сурет).

2.6 Сурет – Goroutine жұмыс істеу принципі

Гороутиндердің артықшылықтары:

– Гороутиндердің өсіп келе жатқан сегментті стектары бар. Яғни, олар қажет болғанда ғана көп жадты пайдаланады.

– Гороутиндердің ағындарға қарағанда іске қосылу уақыты тезірек болып табылады.

– Горутиндер өздеріне (каналдарға) қауіпсіз байланыс жасау үшін кіріктірілген примитивтермен бірге келеді.

– Гороутиндер мәліметтер құрылымын бөліскен кезде mutex оқшаулауға жол бермеуге мүмкіндік береді.

– Сондай-ақ, гороутиндер мен ОС ағындарымен 1: 1 салыстыру болмайды. Бір горутин бірнеше ағындарда жұмыс істей алады. Гороутиндер ОС ағындарының аз санына көбейтіледі.

– Горутиндер арналарды қолдана отырып байланысады. Арналар дизайны бойынша Goroutine көмегімен ортақ жадқа қол жеткізу кезінде жарыс жағдайларының алдын алады. Арналарды Goroutine байланыстыратын құбыр ретінде қарастыруға болады.

Go тілінде жазылған танымал қосымшалар:

– Docker. Linux контейнерлерін орналастыруға арналған құралдар жиынтығы.

– Kubernetes. Бұл бағдарламаны орналастыруды, масштабтауды және басқаруды автоматтандыруға арналған ашық жүйе.

– SoundCloud. Онлайн аудио тарату платформасы.

– Dropbox. Файлдарды орналастыру қызметі.

– Netflix. Фильмдерді және телебағдарламаларды ағындық көрсеті қызметі.

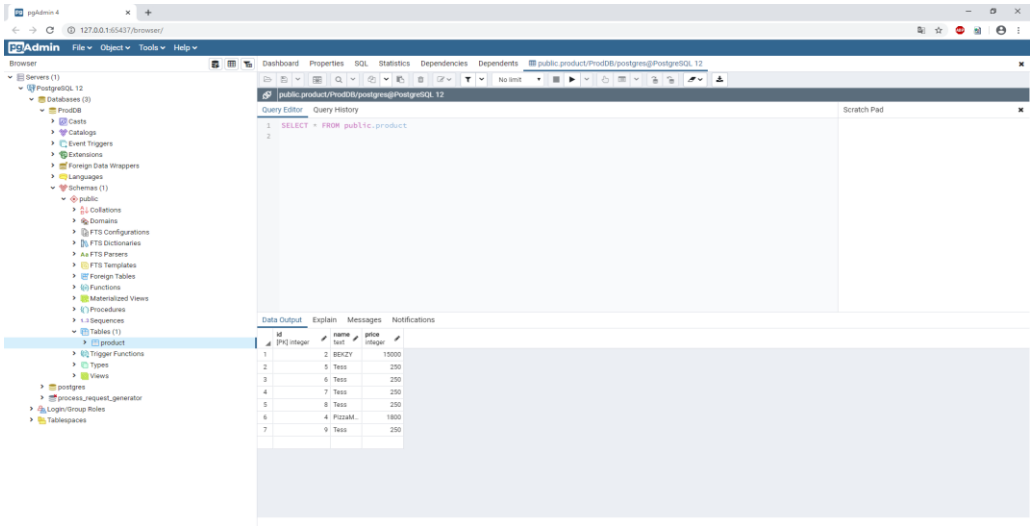
– Twitch. Онлайн ағындағы видео көрсету қызметі.

– Uber және т.б.

2.4 PostgreSQL мәліметтер базасын басқару жүйесі

PostgreSQL - бұл SQL тілін қолданатын және кеңейтетін қуатты, ашық бастапқы объектілік-реляциялық мәліметтер базасын басқару жүйесі(МББЖ), көптеген күрделі деректер жүктемесін қауіпсіз сақтайтын және кеңейтетін көптеген мүмкіндіктермен біріктірілген. PostgreSQL-нің пайда болуы 1986 жылы Берклидегі Калифорния университетінде POSTGRES жобасының аясында басталды және негізгі платформада 30 жылдан астам белсенді дамып келеді.

PostgreSQL өзінің дәлелденген сәулетімен, сенімділігімен, деректердің тұтастығымен, функционалды мүмкіндіктер жиынтығымен, кеңейтіле алуымен және бағдарламалық жасақтаманың ашық көздер қоғамдастығымен әрдайым тиімді және инновациялық шешімдер ұсынуға деген адалдығымен танымал болды. PostgreSQL барлық негізгі операциялық жүйелерде жұмыс істейді, 2001 жылдан бері ACID-мен үйлеседі және әйгілі PostGIS геокеңістік базасын кеңейтуші сияқты қуатты қондырмалар бар(2.7-сурет).



2.7 Сурет – PostgreSQL интерфейсі

PostgreSQL әзірлеушілерге қосымшаларды, деректердің тұтастығын қорғауға және қателіктерге төзімді ортаны құруға көмектесетін, мәліметтер базасының қаншалықты үлкен немесе кіші болмасын басқаруға көмектесетін көптеген мүмкіндіктермен бірге келеді. PostgreSQL еркін және ашық бастапқы кодымен қатар, өте кеңейтілген. Мәселен, сіз өзіңіздің деректеріңіздің түрлерін анықтай аласыз, арнайы функцияларды жасай аласыз, тіпті әртүрлі бағдарламалау тілдерінен кодты дерекқорыңызды қайта толтырмай-ақ жаза аласыз.

PostgreSQL SQL стандартына сәйкес болуға тырысады, егер мұндай сәйкестік дәстүрлі ерекшеліктерге қайшы келмесе немесе нашар сәулет шешімдеріне әкелуі мүмкін болса. SQL стандарты талап ететін көптеген мүмкіндіктерге қолдау көрсетіледі, бірақ кейде синтаксис немесе функция сәл өзгеше болады.

PostgreSQL мәліметтер базасын басқару жүйесі келесі мүмкіндіктері бар:

– деректер түрлері;
– деректердің тұтастығы;
– параллельдік, өнімділік;
– менімділік, апаттарды қалпына келтіру;
– қауіпсіздік;
– кеңейту;
– интернационализация, мәтін іздеу.

PostgreSQL басқаруға болатын мәліметтердің санында да, оны орналастыратын қатардағы пайдаланушыларда да кең масштабталатыны дәлелденді. PostgreSQL өндірістік орталарында көптеген терабайт деректерін басқаратын белсенді кластерлер және петабайттарды басқаратын мамандандырылған жүйелер бар.

2.5 JSON Web Token - берілетін мәліметтерді ұсыну тәсілі

JSON Web Token (JWT) - бұл стандарт (RFC 7519), JSON объекті ретінде тараптар арасында ақпараттың қауіпсіз берілуінің ықшам және өзіндік әдісін анықтайды. Бұл ақпаратты растап және сенуге болады, өйткені ол сандық түрде қол қойылған. JWT-ге құпия (HMAC алгоритмімен) немесе ашық / жеке кілтпен RSA немесе ECDSA көмегімен қол қоюға болады.

Тараптар арасындағы құпияны қамтамасыз ету үшін JWT кодтарын шифрлауға болатынына қарамастан, біз қол қойылған белгілерге тоқталамыз. Қол қойылған таңбалауыштар ондағы шағымдардың тұтастығын тексере алады, ал шифрланған таңбалауыштар бұл талаптарды басқа тараптардан жасырады. Ашық / жеке кілттердің жұптарын қолдана отырып, таңбалауыштарға қол қойылған кезде, қол тек жеке кілтке ие тарап ғана қол қойғанын растайды.

JSON Web Token пайдалану аясы:

– Авторизация: Бұл JWT қолдану үшін ең көп таралған сценарий. Пайдаланушы жүйеге кіргеннен кейін, әрбір келесі сұрауға JWT қосылады, ол

пайдаланушыға осы таңбамен рұқсат етілген маршруттарға, қызметтерге және ресурстарға қол жеткізуге мүмкіндік береді. Бірыңғай кіру - бұл қазіргі уақытта JWT-ны кеңінен қолданатын мүмкіндік, өйткені оның шамадан тыс шығыны және әр түрлі домендерде оңай қолданылуы мүмкін.

– Ақпарат алмасу: JSON веб-токендері - тараптар арасында ақпаратты қауіпсіз түрде берудің жақсы тәсілі. JWT-ге қол қоюға болатындықтан, мысалы, ашық/жеке кілт жұптарын қолдана отырып, жіберушілердің кім екендіктеріне сенімді бола аласыз. Бұған қоса, қолтаңба тақырып пен жүктеменің көмегімен есептелгендіктен, мазмұнның өзгермегенін тексеруге болады.

Өзінің ықшам түрінде JSON Web Token үш нүктеден (.) бөлініп тұрады, олар:

– Header. Әдетте, екі бөліктен тұрады: JWT болып табылатын токен түрі және HMAC SHA256 немесе RSA сияқты қол қою алгоритмі(2.8-сурет).

```
{
  "alg": "RSA",
  "typ": "JWT"
}
```

2.8 Сурет - JWT токеннің header бөлігі

– Payload. Токеннің екінші бөлігі - Payload, онда өтінімдер бар. Өтінімдеп - бұл субъект (әдетте пайдаланушы) туралы мәлімдеме және қосымша мәліметтер(2.9-сурет).

```
{
  "sub": "121221212",
  "name": "Galymzhan Sungatov",
  "admin": true
}
```

2.9 Сурет - JWT токеннің payload бөлігі

– Signature. Signature бөлігін құру үшін кодталған тақырыпты, кодталған жүктемені, құпияны, тақырыпта көрсетілген алгоритмді алып, оған қол қою керек.

Сондықтан JWT әдетте келесідей болады:

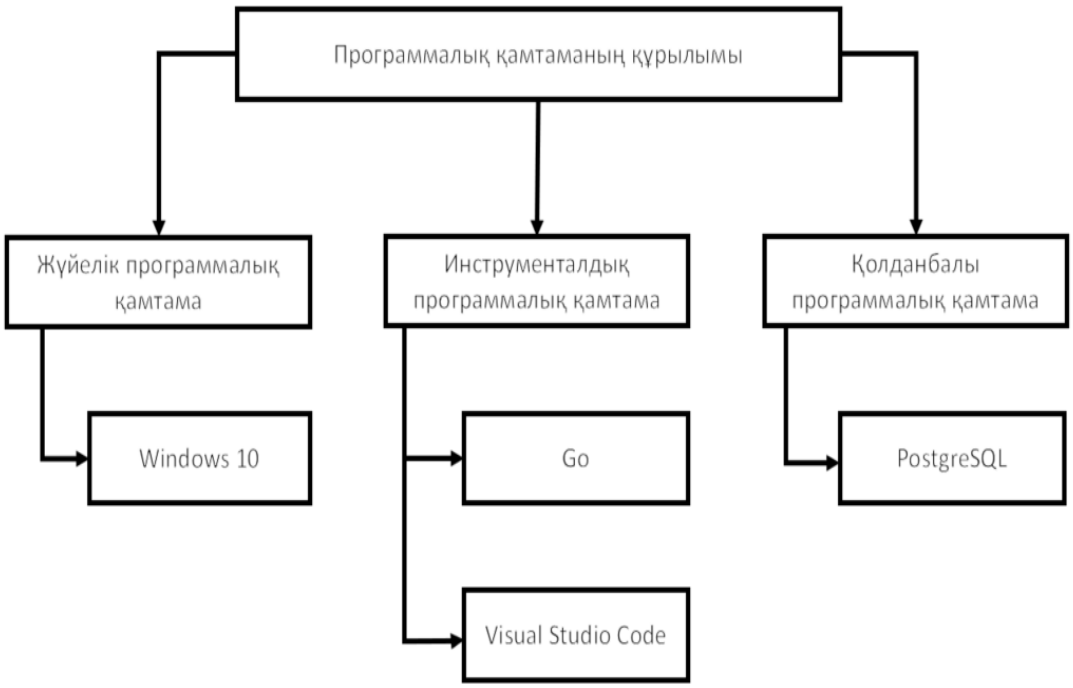
xxxxx.yyyyy.zzzzz

JWT Интернет ауқымында қолданылады. Бұл бірнеше платформаларда, әсіресе ұялы телефондарда, JSON Web токенінің клиенттік өңдеудің қарапайымдылығын көрсетеді.

3 Программалық қамтаманы құру

3.1 Программалық қамтаманың құрылымы

Жобаның серверлік бөлімі Go тілінде және Visual Studio Code ортасында жасалған. Мәліметтер базасы PostgreSQL мәліметтер базасын басқару жүйесін қолдана отырып жасалған(3.1-сурет).



3.1 Сурет – Программалық қамтаманың құрылымы

3.2 Программаның баяндалуы

3.2.1 Жалпы мағлұматтар

Бұл жоба арқылы кез келген адамдар чат және бейнечат арқылы байланыса алады. Сонымен қатар бірнеше адамдар бір-бірімен ойын ойнай алады.Осы байланыстарда қауіпсіз ақпарат алмасу үшін барлығы шифрлау әдісі арқылы жіберіліп отрыады.

3.2.2 Функционалдық тағайындалуы

Жобаның серверлік бөлігі Go (тағы да оны Golang депте атайды) тілінде жазылды және клиенттер арасындағы байланыс WebSocket байланыс протоколы арқылы ұйымдастырылған.

3.2.3 Қолданылған техникалық жабдықтар

Бұл Дипломдық жобаны жазғанда келесі құралдар қолданылды:
– CPU Intel Core i5-9300H, 2.40GHz;
– RAM 8 GB DDR4;
– SSD 500GB;
– GeForce GTX 1650;
– Алынбалы флэш-жинақтағыш 32 ГБ USB 3.1;
– Принтер Canon MF3010;
– Стандартты пернетақта мен тінтуір.

3.2.4 Шақыру және жүктеу

Қосымшаны шақыру үшін, бірінші терминалда Go тілінде жазылған серверді шақырамыз(3.2-сурет):

```
PS C:\Users\Fox\go\src\github.com\OIYNLine> go run main.go
{localhost oyinline password postgres}

(C:/Users/Fox/go/src/github.com/OIYNLine/controller/server.go:29)
```

3.2 Сурет – go run командасы.

Екіншіден, React кітапханасында жазылған клиенттік API-ді “npm start” командасы арқылы шақырамыз(3.3-сурет):

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

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

3.3 Сурет – npm start командасы.

3.2.5 Кіріс мәліметтер

Кіріс мәліметтер ретінде, қолданушының ақпараттары, яғни қолданушыларды тіркеу, оларды өзгерту және қолданушының басқа қолданушыға жіберген хабарламасы жатады.

3.2.6 Шығыс мәліметтер

Қолданушының басқа қолданушыдан алған хабарламасы, дәл қазір жүріп жатқан ойындар туралы ақпараттарды келтіреміз.

3.3 Web-қосымшаның бағдарламалық қамтамасыз етуінің сипаттамасы

Осы жасалған қосымша Visual Studio Code бағдарламасында және Go тілінде жазылған. Оны іске қосу үшін терминалға “go run main.go” командасын енгіземіз(3.4-сурет).

```
PS C:\Users\Fox\go\src\github.com\OIYNLine> go run main.go
{localhost oyinline password postgres}

(C:/Users/Fox/go/src/github.com/OIYNLine/controller/server.go:29)
[2020-05-22 22:30:05]  [55.85ms]  CREATE TABLE "users" ("id" seria
e" varchar(100) NOT NULL UNIQUE,"name" varchar(100) NOT NULL,"l_na
d"))
```

3.4 Сурет – go run main.go командасы

Қолданушылар арасында байланыс ұйымдастыру үшін “Pool” технологиясы қолданды. Келесі суретте NewPool() функциясының коды көрсетілген(3.5-сурет).

```
func NewPool() *Pool {
    name := utils.RandString(16)
    pool := &Pool{
        Name:         name,
        Register:     make(chan *Client),
        Unregister:   make(chan *Client),
        Clients:      make(map[*Client]bool),
        Broadcast:    make(chan Message),
        ReadyClients: make(map[*Client]bool),
        Ready:        make(chan *Client),
    }
    freePools[name] = pool
    log.Printf("Pool %s created", pool.Name)
    return pool
}
```

3.5 Сурет – NewPool() функциясы

Қолданушылар байланыс ұйымдастыра бос “Pool” бар ма тексеріледі, егерде жоқ болса NewPool() функциясы шақырылады(3.6-сурет).

```
var pool *Pool
if len(freePools) > 0 {
    for _, p := range freePools {
        pool = p
        break
    }
} else {
    pool = NewPool()
    go pool.Start()
}
```

3.6 Сурет – “Pool” тексеру

Келесі суретте қолданушының жаңа “Pool” құрып қосылғаны көрсетілген(3.7-сурет).

```
2020/05/22 22:43:50 Pool HLugPkLhTT6gu1HO created
2020/05/22 22:43:50 Client 1 has joined to pool:HLugPkLhTT6gu1HO
```

3.7 Сурет – Жаңа “Pool” құрылуы және қолданушының қосылуы

Барлық қолданушылар арасындағы ақпарат Broadcast арқылы беріледі. Берілген ақпарат JSON форматында жіберіледі(3.8-сурет).

```
case message := <-p.Broadcast:
    for clinet, _ := range p.Clients {

        if err := clinet.Conn.WriteJSON(message); err != nil {
            return
        }
    }
```

3.8 Сурет - Хабарлама жіберілуі

ҚОРЫТЫНДЫ

Интернет технологиялар жыл сайын емес, күн сайын дамып жатыр. Адамдар өмірінде жаңадан әр түрлі техникалар мен технологиялар пайда болып жатыр. Осындай қарқынды даму кезінде нақты уақыт қосымшаларының алатын орны зор.

Кез келген адам жер шарының кез келген нүктесінде отырған адаммен байланыста бола алады. Осы байланыстарға чат, видео және аудио конференция, онлайн ойындар және тағы басқаларын жатқызуға болады. Бұның барлығын бір сөзбен айтқанда нақты уақыт қосымшалары деуге келеді. Нақты уақыт қосымшаларының артықшылықтарымен кемшіліктері бар.

Осы дипломдық жобада нақты уақыт қосымшаларында қолданылатын технологияларды зерттеп, оларды қолдандым. Сол технологияларды қолдану кезінде кездескен өзекті мәселелерді шештім. Жоба прототипін аяқтап, онда кез келген адам байланыс жасап, ойын ойнай алатын мүмкіндік бердім.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1 https://golang.org/doc/

2 https://jwt.io/introduction/

3 https://www.postgresql.org/about/

4 WebSocket: Lightweight Client-Server Communications 1st Edition, Kindle Edition / Andrew Lombardi - O'Reilly Media – 2015 – 144 p

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

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

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

8 Programming in Go: Creating Applications for the 21st Century / Mark Summerfield - 2012

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

А қосымшасы

WebSocket.go

```
package controller
import (
    "fmt"
    "log"
    "net/http"
    "github.com/gin-gonic/gin"
    "github.com/gorilla/websocket"
)
func Upgrade(w http.ResponseWriter, r *http.Request) (*websocket.Conn, error
) {
    var upgrader = websocket.Upgrader{
        ReadBufferSize:  1024,
        WriteBufferSize: 1024,
        CheckOrigin:     func(r *http.Request) bool { return true },
    }
    conn, err := upgrader.Upgrade(w, r, nil)
    if err != nil {
        log.Println(err)
        return nil, err
    }
    return conn, nil
}
func (s *Server) serveWs(c *gin.Context) {
    conn, err := Upgrade(c.Writer, c.Request)
    if err != nil {
        fmt.Fprintf(c.Writer, "%+V\n", err)
    }
    var pool *Pool
    if len(freePools) > 0 {
        for _, p := range freePools {
            pool = p
            break
        }
    } else {
        pool = NewPool()
        go pool.Start()
    }
    client := &Client{
        Conn: conn,
        Pool: pool,
```

```
}
    pool.Register <- client
    client.Read()
}

Pool.go
package controller

import (
    "log"
    "github.com/OIYNLine/controller/utils"
)
var freePools = make(map[string]*Pool)
type Pool struct {
    Name        string
    Register    chan *Client
    Unregister  chan *Client
    Clients     map[*Client]bool
    Broadcast   chan Message
    ReadyClients map[*Client]bool
    Ready       chan *Client
}
func NewPool() *Pool {
    name := utils.RandString(16)
    pool := &Pool{
        Name:        name,
        Register:    make(chan *Client),
        Unregister:  make(chan *Client),
        Clients:     make(map[*Client]bool),
        Broadcast:   make(chan Message),
        ReadyClients: make(map[*Client]bool),
        Ready:       make(chan *Client),
    }
    freePools[name] = pool
    log.Printf("Pool %s created", pool.Name)
    return pool
}
func (p *Pool) Start() {
    defer func() {
        delete(freePools, p.Name)
        log.Print("Pool closed:", p.Name)
    }()
```

А қосымшасының жалғасы

```
        for {
         select {
          case client := <-p.Register:
                p.Clients[client] = true
                log.Printf("Client %d has joined to pool:%s", len(p.Clients), p.Name)
                if len(p.Clients) == 2 {
                   delete(freePools, p.Name)
                   for client, _ := range p.Clients {
                      client.Conn.WriteJSON(Message{Type: 1, Gamer: len(p.Clients), B
ody: bbb, Ready: len(p.ReadyClients)})
                   }
                }
             case client := <-p.Unregister:
                delete(p.Clients, client)
                delete(p.ReadyClients, client)
                if len(p.Clients) == 0 {
                   return
                }
                for clinet, _ := range p.Clients {
                   clinet.Conn.WriteJSON(Message{Type: 1, Gamer: len(p.Clients), Bod
y: bbb, Ready: len(p.ReadyClients)})
                }
             case message := <-p.Broadcast:
                for clinet, _ := range p.Clients {

                   if err := clinet.Conn.WriteJSON(message); err != nil {
                      return
                   }
                }
             case ReadyClients := <-p.Ready:
                if ReadyClients.Ready {
                   p.ReadyClients[ReadyClients] = true
                } else {
                   delete(p.ReadyClients, ReadyClients)
                }
                for clinet, _ := range p.Clients {
                   clinet.Conn.WriteJSON(Message{Type: 1, Gamer: len(p.Clients), Bod
y: bbb, Ready: len(p.ReadyClients)})
                }
            }
         }
      }
```

В қосымшасы

Client.go

```
package controller
import (
    "log"
    "strings"
    "github.com/gorilla/websocket"
)
type Client struct {
    ID    string
    Conn  *websocket.Conn
    Pool  *Pool
    Ready bool
}
type Message struct {
    Type  int      `json:"type"`
    Gamer int      `json:"gamer"`
    Body  []string `json:"body"`
    Ready int      `json:"ready"`
}
var bbb = []string{"", "", "", "", "", "", "", "", ""}
func (c *Client) Read() {
    defer func() {
        c.Pool.Unregister <- c
        c.Conn.Close()
    }()
    for {
        messageType, p, err := c.Conn.ReadMessage()
        if err != nil {
            log.Println(err)
            return
        }
        body := Convert(&p)
        if body[0] == "ready" {
            c.Ready = true
            c.Pool.Ready <- c
            body[0] = body[1]
        } else if body[0] == "notready" {
            c.Ready = false
            c.Pool.Ready <- c
            body[0] = body[1]
        }
```

В қосымшасының жалғасы

```
        message := Message{Type: messageType, Gamer: len(c.Pool.Clients), Body: bo
dy, Ready: len(c.Pool.ReadyClients)}
                c.Pool.Broadcast <- message
            }
        }
        func Convert(b *[]byte) []string {
            sp := strings.Split(string(*b), ",")
            return sp
        }
```