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

Сейтмаханбетов Бейбарыс Сарсенкелдыулы

Сәтбаев Университетінің білім беру жүйесінің эдвайзер, ата-аналар парағының прототипін және техникалық тапсырмасын әзірлеу

ДИПЛОМДЫҚ ЖҰМЫС

5В070300 – «Ақпараттық жүйелер» мамандығы

Алматы 2021

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Кибернетика және ақпараттық технологиялар институты

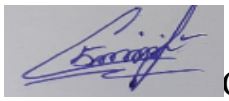
Киберқауіпсіздік, ақпаратты өңдеу және сақтау кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
КАӨжС кафедрасы меңгерушісі
техн. ғыл. канд., ассистент-профессор
_____Н.А. Сейлова
« 27 »___мая___2021 ж.

ДИПЛОМДЫҚ ЖҰМЫС

Тақырыбы: «Сәтбаев Университетінің білім беру жүйесінің эдвайзер, ата-аналар парағының прототипін және техникалық тапсырмасын әзірлеу»

5В070300 – «Ақпараттық жүйелер» мамандығы

Орындаған  Сейтмаханбетов Б.С.

Ғылыми жетекші
т.ғ.м., лектор
_____Зиро А.А.
«26» мая 2021 ж.

Алматы 2021

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
« 27 »___мая___2021 ж.

Дипломдық жұмысты орындауға
ТАПСЫРМА

Білім алушы Сейтмаханбетов Бейбарыс Сарсенкелдыулы
Тақырыбы: «Сәтбаев Университетінің білім беру жүйесінің эдвайзер, ата-аналар парағының прототипін және техникалық тапсырмасын әзірлеу» Университет Ректорының 2020 жылғы «27» қаңтар №762-б бұйрығымен бекітілген.
Орындалған жұмыстың өткізу мерзім «__» _____2021 ж.
Дипломдық жұмыстың бастапқы мәліметтері: Берілген тақырып бойынша әдебиеттерге шолу кезінде жиналған мәліметтер, теориялық материалдар жинау.
Дипломдық жұмыста қарастырылатын мәселелер тізімі:

1. Тақырып аясын талдау және шолу;
2. Техникалық тапсырманы әзірлеу;
3. Бағдарламалық қамтаманы құру;

Графикалық материалдардың тізімі (міндетті түрде қажет сызбалар көрсетілген): жұмыстың ____ слайдтан тұратын презентациясы көрсетіледі.
Ұсынылған негізгі әдебиет 10 кітаптан тұрады.

Дипломдық жұмысты даярлау

КЕСТЕСІ

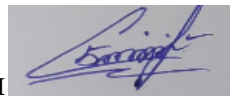
Бөлім атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекші мен кеңесшілерге көрсету мерзімі	Ескерту
Тақырып аясын талдау және шолу	10.01.2021 – 08.02.2021	
Техникалық тапсырманы әзірлеу	05.02.2021 – 10.03.2021	
Бағдарламалық қамтаманы құру	11.03.2021 – 28.04.2021	

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			
Норма бақылаушы	М.А.Кабдуллин, тьютор		

Ғылыми жетекшісі  Зиро А.А.

Тапсырманы орындауға қабылдаған білім алушы 
Сейтмаханбетов Б.С.

Күні «24 » мая 2021 ж.

АҢДАТПА

Бұл дипломдық жұмыста Сәтбаев Университетінің білім беру жүйесінің эдвайзер, ата-аналар парағының прототипін және техникалық тапсырмасын әзірлеу қарастырылады.

Құрылған техникалық тапсырмаға лайықты ата-аналар және эдвайзер парағының прототипінде студенттің оқу процессін, сабаққа деген үлгерімін және де эдвайзері жайлы ақпараттарды көре алу мақсатында жасалады. Студент және ата-ана, эдвайзер және студент арасындағы байланысты нығайту мақсатына бағытталған парақшалар болып табылады. Бұл тек эдвайзерлердің жұмысын жеңілдетіп қана қоймай, уақытты тиімді пайдалануға да көмектеседі.

Дипломдық жұмысты жасау барысында танылмал C# бағдарламалау тіліндегі ASP.NET Core MVC, Razor, Entity Framework, CSS, Bootstrap және PostgreSQL МББЖ қолданылды.

АННОТАЦИЯ

В данной дипломной работе рассматривается разработка прототипа эдвайзерского, родительского страницы и технического задания системы образования Satbayev University.

Страницы родителя, соответствующие созданному техническому заданию, и эдвайзера создаются с целью того, чтобы была возможность мониторинга учебного процесса, успеваемости студента и информацию об эдвайзере. Это страницы, направленны на укрепление связей между студентом и родителем, эдвайзером и студентом. Это не только упростит работу эдвайзеров и поможет более эффективно использовать время.

При создании дипломной работы используется язык программирования C#, ASP.NET Core MVC, Razor, Entity Framework, LINQ выражения, CSS, Bootstrap и PostgreSQL.

ANNOTATION

In this thesis, the development of a prototype of the adviser, parent page and technical task of the Satbayev University education system is considered.

The parent pages corresponding to the created technical task and the adviser are created in order to be able to monitor the educational process, student progress, and information about the adviser. These pages are aimed at strengthening the relationship between the student and the parent, the adviser and the student. This will not only simplify the work of advisors and help them use their time more efficiently.

When creating a thesis, the C# programming language, ASP.NET Core MVC, Razor, Entity Framework, LINQ expressions, CSS, Bootstrap and PostgreSQL are used.

МАЗМҰНЫ

КІРІСПЕ

Бүгінгі күні адамдардың өмірін интернетсіз елестету мүмкін емес. Интернет – алысты жақындатып, ақпаратты лезде алуға жағдай жасайтын жүйе екені бәріне белгілі. Соның ішінде білім мен ғылым саласындағы кеңістікті арттыруда үлкен септігін тигізуде. Заманауи ақпараттық инфрақұрылымдарды құру мақсатында да интернеттің пайдасы мол.

Соның арқасында республикалық, жергілікті атқарушы органдардың, жоғарғы оқу орындарының ресми web-сайттары іске қосылды. Мақсат – студент пен оқытушы арасын байланыстырып, тиімді қызмет көрсетуді қамтамасыз ету. Интернет ресурстары арқылы талай есікті қағып, табан тоздырмай-ақ, үйде отырып, қажетті анықтаманы алып, алдағы өтетін шараларды, оқу орнындағы маңызды жаңалықтарты білуге болады. Яғни, ресми web-сайттар және ресми парақшалар жұмысымызды жеңілдету үстінде.

Менің дипломдық жұмысымның тақырыбы «Сәтбаев Университетінің білім беру жүйесінің эдвайзер, ата-аналар парағының прототипін және техникалық тапсырмасын әзірлеу» болып табылады. Білім беру жүйесі студенттердің әлеуметтік – экономикалық дамуында жетекші роль атқарады, сондай – ақ оны әрі қарай айқындай түседі.

Сәтбаев Университетінің білім беру жүйесінің негізгі мәні – студенттер мен оқытушылар арасындағы байланысты нығайту әрі жұмыстарын жеңілдету. Студенттердің білім деңгейінің нормативтік актілердің талаптарына сәйкестігін бақылау, оқу процесін бақылау, сәйкестікке келтіру – нәтиже ретінде. Студенттерге жаңа оқу материалдарын, оқу жоспарын алдағы іс-шараларды көруге мүмкіндік береді. Білім беру жүйесіне кіріп, өзіне керекті анықтамаларды, сабақ кестесін, семестрлік қорытынды бағаларын, өзінің жеке ақпараттарые көрі мүмкіндігі жасалынған.

Бұл дипломдық жұмысымда Сәтбаев Университетінің білім беру жүйесіне ата-аналар және эдвайзер парақшасынын техникалық тапсырмасын әзірлеу, техникалық тапсырмағы байланысты архитектурасын құрып, мәліметтер базасын құрамын. Программалық ортаны таңдап, парақша интерфейсін құрамын. Орындау үшін C# бағдарламалау тілін, ASP.Net Core MVC (Model-View-Controller) фрэймворкын, PostgreSQL мәліметтер базасын басқару жүйесін, Razor, CSS, Bootstrap қолданамын.

1 ПӘНДІК САЛАНЫ ЗЕРТТЕУ ЖӘНЕ ТАЛДАУ

1.1 Пәндік саланы зерттеу

Білім беру жүйесі - ақпарат көзі болып табылады, ол әртүрлі оқу материалдары мен тапсырмаларына қол жеткізуге, сондай-ақ оқу процесін ұйымдастыруға қатысты ақпаратпен танысуға мүмкіндік береді. Пайдаланушылардың төрттен бір бөлігі біршама алға шығады, олар тек жарияланған ақпаратпен жұмыс жасаумен шектелмейді, бірақ мұндай жұмысқа қосымша - жеке хат алмасу, құжаттармен алмасу мүмкіндіктерін қолдана отырып, жүйенің басқа пайдаланушыларымен белсенді өзара әрекеттеседі. Пайдаланушылардың ең елеусіз, бірақ сонымен бірге ең белсенді бөлігі-бұл порталдың ресурстарын ақпарат алу немесе басқа адамдармен қарым-қатынас жасау үшін емес, жаңа ақпарат алмасу және бірлескен жұмыс алаңдарын құру үшін пайдаланатын қауымдастықтарды ұйымдастырушылар. Көбінесе бұл университет оқытушылары, сондай-ақ оқу және қоғамдық жұмыста белсенді позицияға ие студенттер.

Қазіргі кезде Сәтбаев университетінің білім бері жүйесінің басты парақшасы 2 қойындыдан тұрады. Олар : кесте және студент парағы. Студент парақшасы 20 қойындыдан тұрады. Олар: Силлабус, Менің эдвайзерім, Жеке оқу жоспары, Транскрипт, Кесте, Журнал, Емтихан кестесі, Аттестация, Жалпы кесте, УМКД, Менің мәртебем, Жеке деректер, Құжаттар/нұсқаулар, Қарыздар, Кету қағазы, Байланыс, Анықтамалар, Дипломдау, Ғылыми кітапхана және Ғылыми кітапхана дерекқоры.

1.2 Таңдалған тақырыптың өзектілігі

Қазіргі таңда Сәтбаев Университеті Қазақстан Республикасының аумағындағы техникалық жоғарғы оқу орындарының арасында 1-ші орында болғандықтан, бұл университетке түскісі келетін адам саны көбейю үстінде. Яғни еліміздің барлық елді-мекенінен абитуриенттер келіп, университетімізге түседі. Оқуға түскен студенттерді басқа қалада отырып қадағалау ата-анаға қиындық тудырады, ал келіп қадағалауға кейбірінің қаржылық жағдайы көтермеуі мүмкін. Осындай қиындықтар болмау үшін ата-ана парағы құрылады. Осы Web-парақша арқасында ата-ана еліміздің қай жерінде болсада, тек ғаламтор болса, парақшаға өзінің логин және паролым теріп кіріп өзінің баласының сабақ үлгерімін, сабаққа қатысуын, қандай сабақтар оқып жүргенін сондай-ақ эдвайзері жайлы ақпараттарды көруге мүмкіншілік алады. Егер ата-ана баласын қадағалауы осындай оңай болса, студенттің үлгерімі жақсарып, жоғарғы квалификациялы маман болып шығуына септігін тигізер еді. Ал келесі Web-парақшада эдвайзер өзінің тобында қанша студент бар, қаласа өзіне керекті студент жайлы ақпараттарды көре алады, бұлда эдвайзерлердің жұмысын жеңілдетеді.

Студент және эдвайзер арасындағы қарым-қатынастын жандануына көмегін тигізеді.

1.3 Есептің қойылымы

Жұмыстың мақсаты – студенттердің оқу процессін бақылау және мониторинг жасауға мүмкіндігі бар ата-ана және эдвайзер парақшалар құру болып табылады.

Қойылған мақсатқа жету үшін келесі есептер қарастырылуы керек:

- Пәндік аймаққа шолу жасау;
- техникалық тапсырманы әзірлеу;
- entity framework арқылы мәліметтер базасын құру;
- мәліметтер базасының er-моделін құру;
- парақшаның серверлік жағын жасау
- мәліметтер базасына сұраныстарда linq қолдану;
- валидация жасау;
- модель және контроллер құру;
- миграция құру;
- қолданушы интерфейсін әзірлеу.

Парақшаларда келесі функционалдарды әзірлеу:

Ата-ана студенттің оқу үлгерімін, сабаққа қатысуын мониторинг жасау мүмкіншілігі;

Топтағы студенттер тізімін көру мүмкіндігі;

Топ эдвайзері жайлы ақпаратты көру;

Пән мұғалімдерін қарау;

Оқу журналын қарау мүмкіндігі.

Эдвайзер парағында, топтағы студенттер және олар жайлы ақпараттарды көру функционалы болу керек.

2 ТЕХНИКАЛЫҚ ТАПСЫРМАНЫ ӘЗІРЛЕУ ЖӘНЕ ПАРАҚШАЛАРДЫҢ ПРОТОТИПІН ЖОБАЛАУ.

2.1 Техникалық тапсырманы әзірлеу

Техникалық тапсырма - бұл сайтқа қойылатын талаптар жазылған құжат. Бұл талаптар неғұрлым нақты және егжей-тегжейлі сипатталған болса, процестің барлық қатысушылары оның қандай болу керектігін жақсы түсінеді. Сонымен, бәрі нәтижеге риза болу мүмкіндігі артып келеді. Техникалық тапсырманың пайдасы көп. Әр тарап үшін ол әр түрлі.

Клиент үшін пайдасы: Ол не үшін ақша төлейтінін және сайт қандай болатынын түсініңіз. Бірден құрылымын көруге, түсінуге де болады. Бәрі сәйкес келетінін бағалаңыз. Егер жоқ болса, Даму басталғанға дейін оны еш қиындықсыз өзгертіңіз. Орындаушының құзыреттілігін қараңыз. Егер техникалық тапсырма түсінікті және айқын болса, әзірлеушіге сенім артады. Егер ботқасы жазылған болса, онда ол жүгіріп, артқа қарамауы керек.Орындаушының жауапсыздығынан сақтандыру. Сайт дайын болған кезде оны техникалық тапсырмамен тексеруге болады. Сәйкессіздіктер бар ма? Әзірлеуші оларды түзетуге міндетті. Егер сіз ресми түрде ынтымақтасып, келісім жасасаңыз-сіз тіпті сот арқылы мәжбүрлей аласыз. Орындаушыларды ауыстыруды жеңілдетіңіз. Егер клиент пен әзірлеуші шатасып, қашып кетсе, сайт құру ұзаққа созылуы мүмкін. Егжей — тегжейлі техникалық тапсырма болған кезде оны жаңа командаға беруге болады-ол бірнеше есе жылдам жұмыс істейді.Күрделі өнімді әзірлеу құнын біліңіз. Күрделі веб-қызметті әзірлеудің нақты уақыты мен құнын бірден бағалау мүмкін емес. Алдымен сіз қызметтің қалай жұмыс істейтінін және оның қандай функциялары болатынын түсінуіңіз керек. Ол үшін техникалық тапсырманы дайындау керек.

Орындаушының пайдасы: Тапсырыс берушінің не қалайтынын түсіну. Клиентке ондаған сұрақтар қойылады, мысалдар келтіріледі, шешімдер ұсынылады. Содан кейін олар бәрін бір құжатқа жазып, келіседі. Егер бәрі жақсы болса, сіз оны дұрыс қабылдадыңыз. Клиенттің кенеттен қалауларынан сақтандыру. Кейде тапсырманы жартысына өзгерткісі келетін клиенттер кездеседі. Егер сіз ТТ-мен келісіп, қол қойсаңыз, сіз бұл туралы қорықпайсыз. Бұл жағдайда тіпті сот сіздің тарапыңызда болады. Өз құзыреттілігін көрсету. Жақсы дайындалған техникалық тапсырма клиентке әзірлеушілердің сарапшылығын көрсетеді. Егер компания веб-сайттың дамуына сенетініне күмәнданса, күмән туындауы мүмкін. Даму процесін жеңілдету және жеделдету. Жақсы ТТ-да сайттың құрылымы, әр беттегі қажетті функциялар мен элементтер көрсетілген. Барлық талаптар сіздің көз алдыңызда болған кезде - тек бағдарламалау және жазу ғана қалады.

Техникалық тапсырмалар тізімі:

- Парақша тез жүктелу керек;

- жүктемелерге шыдамды болу керек;
- минималистік ынғайлы интерфейс (қолданышу бір көргеннен немесе қиналмай түсінетіндей);
- суреттер jpeg форматта сақталу керек және парақшада jpeg форматта қолданылуы керек;
- қолданушы әр түрлі браузерді қолданғанда барлығында бірдей болу керек (кроссбраузерлік chrome 1+, opera 10+, ie 9+, safari 4+);
- адаптивтілік (үстел мониторы, ноутбук, смартфон, планшет);
- интерфейс дизайнында қатаң және жұмсақ реңктер болу керек;
- css және javascript файлдарын біріктіріңіз және қысыңыз;
- мүмкіндігінше css- спрайт қолдану;

Парақшалардың функционалдығына қойылатын талаптар.

Парақшаларда пайдаланушыларға келесідей мүмкіндік беруі керек:

- Парақша бойынша навигацияны жүзеге асыру (беттер арасында ауысу);
- әр түрлі құжаттар мен файлдарды жүктеу (қажетті қол жеткізу құқықтары болған жағдайда) ;
- парақша тілін орыс тілінен ағылшын тіліне және керісінше өзгерту;
- парақшаларға тіркелген пайдаланушы ретінде кіруді жүзеге асыру. (тиісті кіру құқықтары болған жағдайда).
- ата-ана топ студенттерінің тізімін, эдвайзер жайлы ақпаратты, баласының оқу үлгерімін, пән мұғалімдерін көру алу функционалы (қажетті құқықтары және тіркелген болған жағдайда);
- эдвайзер топ студенттерін тізімін тек көре қоймай, студент жайлы ақпарат алу мүмкіндігі;
- студенттің сабақтан тым көп қалып жүрген жағдайында хабарлама алу мүмкіндігі.

Сайтты басқару жүйесі мүмкіндік беруі керек:

- Парақша беттерін басқару (олардың мазмұнын қосу, жою, өзгерту);
- мәзір элементтерін басқару;
- парақшаға жаңалықтарды, жаңа ақпараттарды қосу / өзгерту / жою;
- парақшаларға графикалық материалдарды жүктеу (фото-бейне суреттер, түрлі файлдар және т.б.).

Парақша беттерінің толық сипаттамасы.

Ата-ана парағы

Ата-ана парағының басты беттінде студент аты-жөні, мамандық шифры және же мамандық аты жазылу керек. Онымен қоса болып жатқан семестрдегі оқылып жатқан сабақтар тізімі, сол сабақтарды өтетін мұғалім аты-жөні, сабақ қанша кредиттан тұратыны, қанша рет сабақтан тыс қалғаны, және де сол сабақтан емтихан қорытындысы тұру керек. Эдвайзер қойындысында топ эдвайзерінің аты-жөні, қай кафедраның мұғалімі, лауазымы, ұялы телефон номері, электронды поштасы болу керек. Сабақ кестесі қойындысында аты айтып тұрғанда сол семестрдегі сабақ кестесі болу керек, журнал қойындысында студенттің сабаққа үлгерімі және бағалары тұру керек. Мұғалімдер қойындысында пән мұғалімдерінің суреттері, аты-жөні, кафедрасы жазылады. Студенттер қойындысында топтағы студенттер тізімі, аты-жөні болу керек.

Эдвайзер парағы

Басты бетте топтың аты, топта қанша студент бар солардың саны, топтағы студенттер өтіп жатқан сабақтар тізімі, топтың үлгерімі болады. Журнал қойындысында топтың үлгерімін көре алу мүмкіндігі болады. Студенттер қойындысында студенттер тізімі, олар жайлы ақпарат болады. Студентке тінтуірмен басқанда келесі бет ашылып студент жайлы ақпарат шығады. Ата-ана қойындысында студенттің ата анасы жайлы ақпарат шығу керек (ұялы телефон номері, электронды поштасы, аты-жөні және т.б.). Сабақ кестесі қойындысында аты айтып тұрғанда сол семестрдегі сабақ кестесі болады.

2.2 C# бағдарламалау тілін таңдау себебі мен артықшылықтары

C# бағдарламалау тілі - бұл заманауи объектіге бағытталған және типке қауіпсіз бағдарламалау тілі. 1998-2001 жылдары Андерс Хейлсберг пен Скотт Вильтаумоттың басшылығымен Microsoft компаниясының инженерлер тобымен әзірленген Microsoft. NET Framework қосымшаларын әзірлеу тілі ретінде жасалған. C# тілі бағадарламалаушыларға желі экожүйесінде жұмыс істейтін қауіпсіз және сенімді қосымшалардың көптеген түрлерін жасауға мүмкіндік береді. C# белгілі с тілдер тобына жатады және C, C++, Java немесе JavaScript бағдарламалау тілін қолданатын адамдарға таныс болып көрінеді. C# полиморфизмді, мұрагерлікті, операторлардың шамадан тыс жүктелуін, статикалық теруді қолдайды. Нысанға бағытталған тәсіл үлкен, бірақ сонымен бірге икемді, кеңейтілетін және кеңейтілетін қосымшаларды құру мәселелерін шешуге мүмкіндік береді. C# белсенді дамуды жалғастыруда және әр жаңа нұсқада ламбда, динамикалық байланыстыру, асинхронды әдістер және т. б. сияқты қызықты функциялар пайда болады.

C# - та түрлердің біріңғай жүйесі бар. C# барлық түрлері, Int және double секілді примитивтерді қоса алғанда, object бір тамыр түрінен мұра. Осылайша,

барлық типтер операциялардың жалпы жиынтығын қолданады, кез келген түрдегі мәндерді сақтауға, тасымалдауға, қолдануға және өңдеуге болады. Сонымен қатар, C# нысандар үшін жадыны динамикалық бөліп шығаруға, сондай-ақ стэкте қарапайым құрылымдарды сақтауға мүмкіндік беретін теңшелетін сілтемелер түрлері мен мән түрлерін қолдану мүмкіндігі қарастырылған. Артықшылықтарына келетін болсақ:

- C# компиляторы .NET-тің стандартты қондырғысына кіреді, сондықтан ондағы бағдарламаларды Visual Studio сияқты құралдарсыз да жасауға және құрастыруға болады;
- бұл тіл барлық бағдарламалауда объектіге бағытталған тәсілді қолданады. Бұл сізге пәндік аймаққа негізделген дерексіз құрылымдарды сипаттап, содан кейін олардың өзара әрекеттесуін жүзеге асыру керек дегенді білдіреді. Бұл тәсіл өте танымал, өйткені ол сізге барлық ақпаратты есте сақтамауға, бірақ қара жәшік қағидаты бойынша жұмыс істеуге мүмкіндік береді;
- сондай-ақ, тілде синтаксистік артықшылықтардың көптігі бар, бұл бағдарламашының қиын өмірін жеңіл етеді. Ұзын код жолын жазудың орнына, сіз тек дайын дизайнды қолданасыз, ал компилятор сіз үшін барлық қалған лас жұмыстарды жасайды. Бірақ кей жағдайда кейбір осындай конструкциялар өнімділік тұрғысынан ең оңтайлы емес. Бірақ мұның бәрі кодтың оқылуына және дамудың жоғары жылдамдығына байланысты.
- және де артықшылықтарына, .Net Core платформасындағы жақында шыққан кроссплотмалықты айтса болады.

Жоғарыда атап кеткен артықшылықтарды және де тілдің әрі қарай дамып келе жатқанын ескере отыра осы тілді таңдадым.

2.3 ASP.Net Core MVC платформасы

ASP.NET Core платформасы - кез-келген күрделіліктегі веб-қосымшаларды құруға арналған технология. ASP.NET Core бастапқы коды ашық негізінде болып табылады, сонымен қатар әртүрлі операциялық жүйелерде жұмыс істеуге болатын .NET Core платформалық ортасында жұмыс істейді (қарапайым .NET-те жұмыс істей алады). Веб-қосымшалардың өздері Windows үшін әдеттегі IIS серверін қолдана отырып жұмыс істей алады немесе Kestrel сияқты басқа платформалық серверді қолдана алады.

ASP.NET Core құрамына MVC фрэймворкі кіреді. Платформа MVC, Web API және Web Pages функционалдығын біріктіреді, олар алдыңғы нұсқаларда бөлек орындалады, сондықтан көптеген қайталанатын функцияларды қамтиды. Енді олар бір бағдарламалық модельге біріктірілген ASP.NET Core MVC.

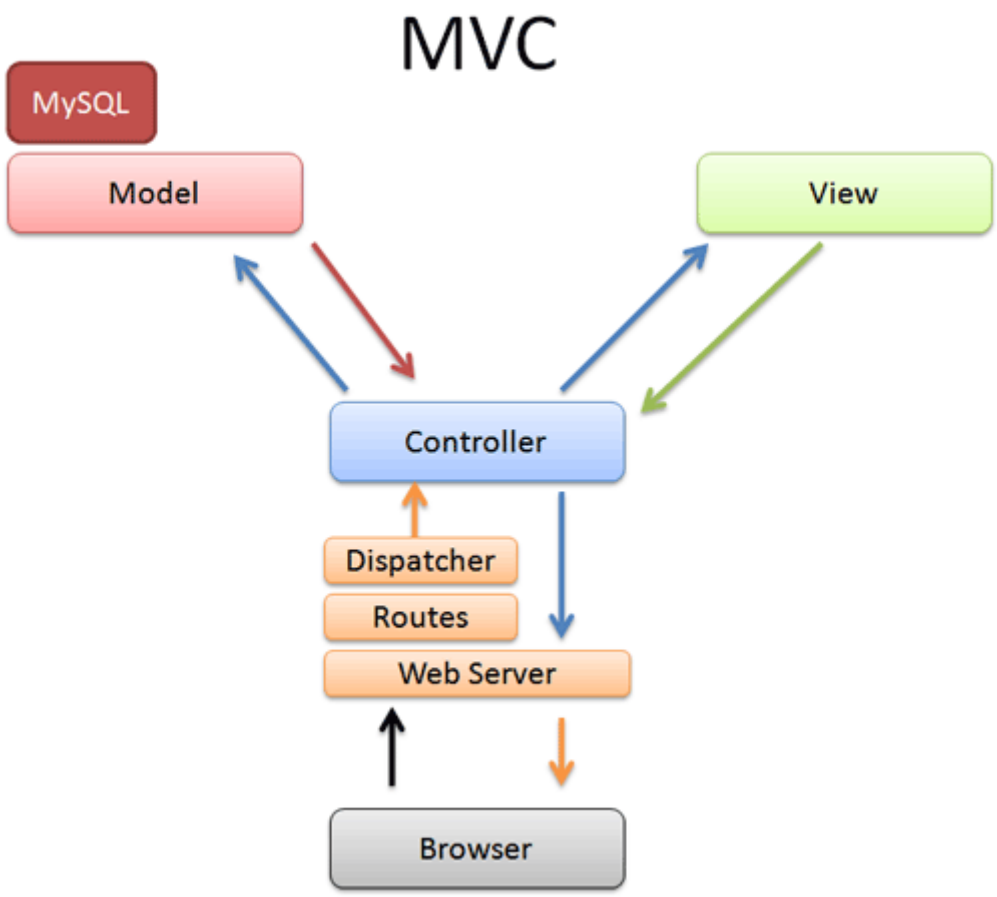
Microsoft ASP.NET MVC-танымал .NET Framework үстіне салынған веб-қосымшаларды әзірлеу платформасы. В ASP.NET MVC платформасы негізінен қолданбалы архитектураны және қолдау көрсетілетін кодты көрсететін дәлелденген дизайн мен тәжірибе үлгілеріне сүйенеді. 2008 жылы алғашқы нұсқасы шығарылды ASP.NET MVC шығарылды. WebForms тәсілінен толық кету, ASP.NET MVC бет негізіндегі архитектурадан бас тартады және оның орнына модель көрінісі контроллерінің архитектурасына сүйенеді. Model-View-Controller шаблоны-бұл қолданбаның әртүрлі бөліктері арасында қатты оқшаулауды қолдайтын сәулет үлгісі. Бұл оқшаулау мәселелерді бөлу немесе жалпы әлсіз байланыс деп аталады. Нашар байланыстырылған стильдегі қосымшаның архитектурасын жасау бірқатар қысқа мерзімді және ұзақ мерзімді артықшылықтарды ұсынады:

- Әзірлеу. Жеке компоненттер басқа компоненттерге тікелей тәуелді емес, яғни оларды оқшаулау оңай. Сондай-ақ, компоненттер оңай ауыстырылады немесе ауыстырылады, бұл бір компонент өзара әрекеттесе алатын басқа компоненттердің құрылысына әсер ететін жағдайларда асқынулардың пайда болуына жол бермейді.
- тексеру мүмкіндігі. Компоненттердің еркін қосылуы компоненттердің өндірістік нұсқаларының орнына сынақ нұсқаларын қолдануға мүмкіндік береді. Мысалы, дерекқорға кіретін компонентті жай ғана статикалық деректерді қайтаратын компонентпен ауыстыру арқылы физикалық мәліметтер базасымен әрекеттесуден аулақ болуға болады. Компоненттердің сынақ көріністерімен орындарды оңай бөлісу мүмкіндігі тестілеу процесін едәуір жеңілдетеді, бұл уақыт өте келе жүйенің сенімділігін едәуір арттырады.
- сүйемелдеу. Компоненттерді оқшаулау дегеніміз, логикадағы өзгерістер әдетте аз мөлшерде компоненттерде оқшауланады-көбінесе біреуінде. Өзгерістің әсері әдетте оның масштабына байланысты болатындығын ескере отырып, аз компоненттердің өзгеруі жақсы.

Бағыттар ASP.NET MVC берілген URL мекен-жайы үшін қандай контроллер әдісін (басқаша контроллер әрекеті деп аталады) анықтауға жауап береді. Келесі сипаттар маршруттармен байланысты: ерекше атау (атауды белгілі бір маршрутқа сілтеме ретінде пайдалануға болады); URL үлгісі (сәйкес URL мекен-жайларын маңызды сегменттерге бөлетін қарапайым шаблон синтаксисі); стандартты мәндер (URL үлгісінде анықталған сегменттер үшін стандартты мәндердің қосымша жиынтығы); шектеулер (сәйкес келетін URL мекенжайын анықтау үшін URL үлгісіне қолдануға арналған шектеулер жиынтығы).

MVC үлгісі тұжырымдамасы қосымшаны үш компонентке бөлуді қамтиды:

- Модель (model): қосымшада пайдаланылатын деректерді, сондай-ақ деректермен тікелей байланысты логиканы, мысалы, деректерді тексеру логикасын сипаттайды. Әдетте, модель объектілері дерекқорда сақталады. MVC-де модельдер екі негізгі типтен тұрады: көріністер деректерді көрсету және беру үшін пайдаланатын көрініс модельдері және деректерді басқару логикасын сипаттайтын Домен модельдері. Модельде деректер болуы мүмкін, осы деректерді басқару логикасын сақтай алады. Сонымен қатар, модельде пайдаланушының өзара әрекеттесу логикасы болмауы керек және сұранысты өңдеу механизмін анықтамауы керек. Сонымен қатар, модель көріністе деректерді көрсету логикасын қамтымауы керек.
- көрініс (View): көрнекі бөлікке немесе пайдаланушы интерфейсіне жауап береді, көбінесе қолданушы қосымшамен өзара әрекеттесетін html беті. Сондай-ақ, көріністе деректерді көрсетуге байланысты логика болуы мүмкін. Сонымен қатар, көріністе пайдаланушының сұранысын өңдеу немесе деректерді басқару логикасы болмауы керек.
- контроллер (Controller): пайдаланушы мен қолданба, көрініс және деректер қоймасы арасындағы байланысты қамтамасыз ететін орталық MVC компонентін білдіреді. Онда пайдаланушының сұранысын өңдеу логикасы бар. Контроллер пайдаланушы енгізген деректерді алады және оларды өңдейді. Өңдеу нәтижелеріне байланысты ол пайдаланушыға белгілі бір нәтиже жібереді, мысалы, модель деректерімен толтырылған көрініс түрінде.



1 – сурет. MVC құрылымы.

2.4 Entity Framework фрэймворкі

Entity Framework деректермен жұмыс істеу үшін .NET шеңберіне негізделген арнайы объектіге бағытталған технологияны ұсынады. Егер ADO.NET дәстүрлі құралдары деректер базасымен өзара әрекеттесу үшін қосылымдар, командалар және басқа объектілер құру үшін мүмкіндік берсе, EF құрылымның мәні абстракцияның жоғары деңгейі болып табылады, бұл дерекқордың өзінен абстракциялауға және сақтау түріне қарамастан деректермен жұмыс істеуге мүмкіндік береді. Егер физикалық деңгейде біз кестелермен, индекстермен, бастапқы және сыртқы кілттермен жұмыс жасасақ, бірақ бет-әлпет аясында бізге ұсынатын тұжырымдамалық деңгейде біз объктермен жұмыс жасаймыз.

Entity Framework бірінші нұсқасы-1.0 2008 жылы шығарылды және өте шектеулі функционалдылықпен болды, ORM үшін негізгі қолдауды (объектілі-реляциялық дисплей - нақты нысандарға деректерді көрсету) және мәліметтер базасымен өзара әрекеттесудің жалғыз тәсілі – Database First. 2010 жылы 4.0 нұсқасы шыққаннан кейін көп нәрсе өзгертілді - сол кезден бастап

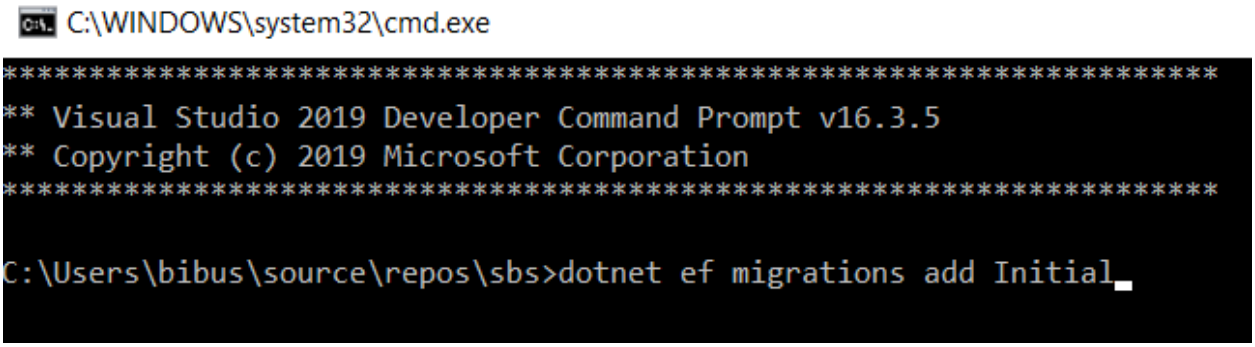
адамдар деректерге қол жеткізу үшін ұсынылған технологияның бөлігі болды, ал жақтаудың өзінде мәліметтер базасымен өзара әрекеттесудің жаңа мүмкіндіктері енгізілді – Model First және Code First.

Entity Framework мәліметтер базасымен өзара әрекеттесудің үш мүмкін әдісін ұсынады:

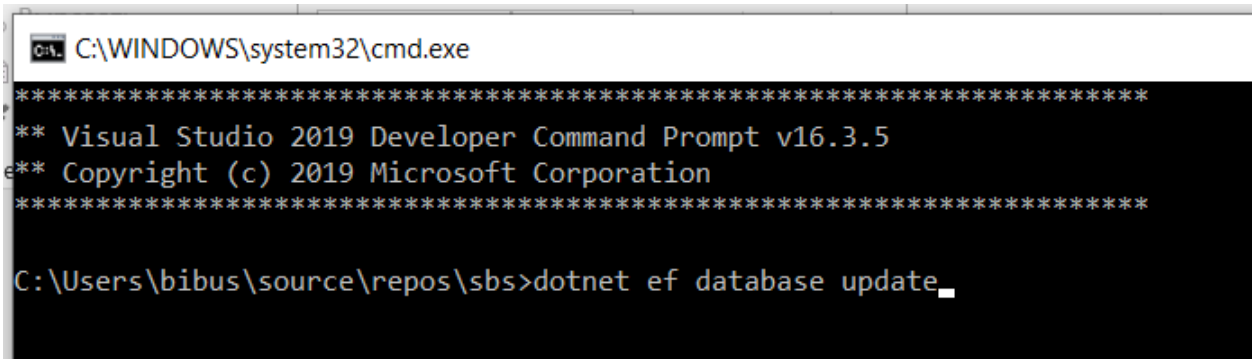
- Database First: Entity Framework нақты мәліметтер базасының моделін көрсететін сыныптар жиынтығын жасайды;
- model first: алдымен әзірлеуші мәліметтер базасының моделін жасайды, содан кейін Entity Framework серверде нақты мәліметтер базасын жасайды;
- code First: әзірлеуші дерекқорда сақталатын деректер моделінің класын жасайды, содан кейін осы модель бойынша Entity Framework дерекқорды және оның кестелерін жасайды.

Жаңа қосымшаны әзірлеу кезінде сіздің деректер моделіңіз жиі өзгереді және әр өзгерген сайын ол дерекқормен синхрондауды бұзады. Сіз осы нұсқаулықтармен жұмыс істей бастадыңыз, егер ол әлі болмаса, мәліметтер базасын құру үшін Entity Framework бағдарламасын орнатасыз. Содан кейін деректер моделі өзгерген сайын — нысан кластарын қосу, жою немесе өзгерту немесе DbContext класын өзгерту арқылы сіз дерекқорды жоя аласыз, содан кейін EF берілген модельге сәйкес келетін жаңасын жасайды және оны сынақ деректерімен толтырады.

Деректер базасын деректер моделімен синхрондаудың бұл әдісі қосымшаны жұмыс ортасына орналастырмас бұрын жақсы жұмыс істейді. Бағдарлама жұмыс ортасында орындалған кезде, ол әдетте сақтағыңыз келетін деректерді сақтайды және жаңа бағанды қосу сияқты әр өзгерісте бәрін жоғалтқыңыз келмейді. EF Core функциясы бұл мәселені EF-ке дерекқор құрудың орнына дерекқор схемасын жаңартуға мүмкіндік беру арқылы шешеді.



2 - сурет. EF migrations жасау.



3 – сурет. Миграциясындан кейінгі МБ жаңарту.

2.5 ASP.NET Core Identity

ASP.NET Core Identity – бұл веб-қосымшаларда пайдаланушыларды аутентификациялау/авторизациялау жүйелерін жазуға арналған платформа. Membership API негізіндегі ескірген тәсілді ауыстыруға арналған, пайдаланушыларды басқаруға арналған Microsoft коорпорациясы ұсынған жаңа API – интерфейс болып табылады. Пайдаланушыларды, парольдерді, профиль деректерін, рөлдерді, мәлімдемелерді, маркерлерді, электрондық поштаны растауды және аутентификация үшін үшінші тарап провайдерлерінің - Google, Facebook, Twitter және т. б. есептік жазбаларын пайдалануға мүмкіндік береді. Жүйе үшін ең іргелі жұмыс ASP.NET Core Identity пайдаланушылардың аутентификациясы болып табылады. Қол жеткізуді шектеудің негізгі құралы іс-әрекет әдістері -Authorize атрибуты, ол MVC инфрақұрылымына тек аутентификацияланған пайдаланушылардан сұраулардың өңделуі керектігін хабарлайды.

```
[Authorize]
Ссылок: 0
public IActionResult Index(string id = null)
{
    User user = id == null ? _userManager.GetUserAsync(User).Result : _userManager.FindByIdAsync(id).Result;
    return View(user);
}
```

4 – сурет. Authorize атрибутын қолдану.

2.6 PostgreSQL мәліметтер базасын басқару жүйесі

PostgreSQL тек реляциялық емес, объектілік-реляциялық МББЖ. Бұл MySQL, MariaDB және Firebird сияқты ашық бастапқы коды бар басқа SQL мәліметтер базасынан кейбір артықшылықтарды береді. Объектілі-реляциялық мәліметтер базасының іргелі сипаттамасы - бұл пайдаланушы объектілерді және олардың сипаттамасын, оның ішінде мәліметтер типтерін, функцияларын, операцияларын, домендерді және индекстерді қолдау. Бұл PostgeSQL-ді керемет икемді және сенімді етеді. Сонымен қатар, ол мәліметтердің күрделі құрылымын жасай, сақтай және шығара алады.

PostgreSQL көптеген мәліметтерді өңдей алады. Өте тиімді әрі қолайлы мәліметтер базасы болып табылады. Қазіргі таңда көптеген фирмалар, банктер осы PostgreSQL МББЖ қолданады. Басқа МББЖ-нен артықшылықтарына келетін болсақ:

- Шексіз өлшемді дерекқорды қолдау;
  - Деректер базасының максималды мөлшеріне шектеулер жоқ;
  - кестеде жазбалар санына шектеулер жоқ;
  - кестеде индекстер санына шектеулер жоқ;
  - кестенің максималды мөлшері - 32 тбайт;
  - жазудың максималды мөлшері-1,6 тбайт;
  - өріс өлшемі-1 гбайт.
- күшті және сенімді транзакциялар мен репликация механизмдері;
- кірістірілген бағдарламалау тілдерінің кеңейтілген жүйесі және C-үйлесімді модульдерді жүктеуді қолдау;
- мұрагерлік. Кестелер басқа кестелерден сипаттамалар мен өрістер жиынтығын иелене алады. Бұл жағдайда, құрылған кестеге қосылған деректер автоматты түрде (егер бұл бөлек көрсетілмесе) ата-ана кестесіндегі сұрауларға қатысады;
- жеңіл кеңейту. PostgreSQL МББЖ үшін бұл пайдаланушы жүйені жаңа мүмкіндіктерді, түрлерді, тілдерді, агрегаттарды, индекстерді және операторларды анықтау арқылы реттей алатындығын білдіреді.;
- сенімділік. Ол ACID (атомарлық, оқшаулау, жүйелілік, деректердің қауіпсіздігі) қағидаттарына сәйкестікпен, көп нұсқалылықпен, Write Ahead Logging (WAL) — барлық қолданыстағы транзакцияларды хаттамалаудың жалпыға бірдей қабылданған механизмімен қамтама;

Entity framework арқылы PostgreSQL МБ-на қосылу үшін Npgsql.EntityFrameworkCore.PostgreSQL кітапханасын орнатып алу керек. Бұл кітапхана ORM - ге мәліметтер базасымен өзара әрекеттесуге және оған қосылуға мүмкіндік береді. Кітапхананы орнатқаннан кейін кейбір өзгерістер енгізу керек :

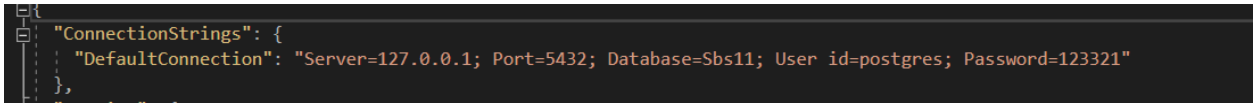
1. Қосылу жолын өзгерту : "DefaultConnection" : "Server=127.0.0.1; Port=5432; Database=Store; User Id=postgres; Password=(дерекқорды орнату және орнату кезінде орнатқан құпия сөзіңізді көрсетіңіз);"

2. Сондай-ақ, Startup файлында біз жаңа провайдерді қолданатынымызды көрсетуіміз керек :

services.AddDbContext<StoreContext>(options => options.UseNpgsql(connection));

3. Жаңа провайдер үшін миграцияны қайта жасау керек (ескі миграцияны өшіріп немесе жойып, жаңасын жасаған абзал)

4. Жобадан SQLite кітапханасын алып тастаңыз.



5 – сурет. Appsettings – те жасалған өзгерістер.

2.7 LINQ арқылы мәліметтер базасына сұраулар жасау

LINQ (Language-Integrated Query) деректер көзіне қарапайым және ыңғайлы сұраныс жасау тілін ұсынады. Деректер көзі IEnumerable интерфейсін (мысалы, стандартты коллекциялар, массивтер), DataSet деректер жиынтығын, XML құжатын жүзеге асыратын объект бола алады. Бірақ мәліметтер түріне қарамастан, LINQ деректерді іріктеу үшін барлығына бірдей тәсілді қолдануға мүмкіндік береді. LINQ-дің бірнеше түрлері бар:

- LINQ to Objects: массивтермен және коллекциялармен жұмыс істеу үшін қолданылады;
- LINQ to Entities: Entity Framework технологиясы арқылы дерекқорға сілтеме жасау кезінде қолданылады;
- SQL-ге SQL: MS SQL Server-де деректерге қол жеткізу технологиясы;
- LINQ to XML: XML файлдарымен жұмыс істеу кезінде қолданылады;
- LINQ to DataSet: DataSet объектісімен жұмыс істеу кезінде қолданылады;
- параллель LINQ (PLINQ): параллель сұрауларды орындау үшін қолданылады.

Тілдің кейбір жаңа ерекшеліктерін қолдана отырып, LINQ SQL-ге ұқсас синтаксисті тікелей C# тілінде жазылған бағдарлама кодында қолдануға мүмкіндік береді:

- Анонимді теру
- Кеңейту әдістері
- Ламбда-есептеу
- Өрнек ағашы (ағылш. Expression tree)
- Стандартты сұрау тілінің операторлары

```
public IActionResult Student()
{
    Parent parent = _db.Parents.FirstOrDefault(p => p.User.Id == _userManager.GetUserId(User));
    ParentAndStudent user = _db.ParentAndStudents.FirstOrDefault(p => p.ParentsId == parent.Id);
    Student Student = _db.Students.FirstOrDefault(s => s.Id == user.StudentId);
    List<Student> Students = _db.Students.Where(s => s.GroupId == Student.GroupId).ToList();

    return View(Students);
}
```

6 – сурет. LINQ сұраныс. МБ-нан студенттерді іздеу сұранысы.

```
public async Task<IActionResult> Index()
{

    bool v = User.IsInRole("parent");
    if (v)
        {
    Parent parent = _db.Parents.FirstOrDefault(p => p.User.Id == _userManager.GetUserId(User));
    ParentAndStudent user = _db.ParentAndStudents.FirstOrDefault(p => p.ParentsId == parent.Id);
    Student Student = _db.Students.FirstOrDefault(s => s.Id == user.StudentId);
        List<Teacher> txt = new List<Teacher>();
        ParentIndexViewModel viewModel = new ParentIndexViewModel
        {
            student = Student,
            subjectAndStudents = _db.SubjectAndStudents.Where(s => s.StudentId == Student.Id).ToList()
        };
        foreach (var item in viewModel.subjectAndStudents)
        {
            txt.Add(_db.Teachers.FirstOrDefault(t => t.SubjectId == item.SubjectId));
        }
        viewModel.Teacher = txt;
        return View(viewModel);
    };

    return View();
```

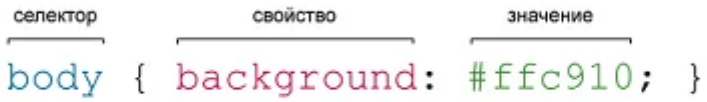
7 – сурет. LINQ сұраныс.

2.8 Парақшалардың прототипін құруда қолданылған технологиялар

CSS - белгілеу тілін (HTML, XHTML, XML) қолдана отырып жасалған құжаттың сыртқы түрін сипаттауды сипаттайтын ресми тіл. Бұл атау "каскадты стиль кестелері"дегенді білдіретін ағылшын Cascading Style Sheets сөзінен шыққан. CSS-тің мақсаты - беттің сыртқы түрін оның мазмұнынан бөлетін нәрсе. Егер құжат тек HTML көмегімен жасалса, онда ол әр элементті ғана емес, сонымен қатар оны көрсету әдісін де анықтайды (түс, қаріп, блоктың орналасуы және т.б.). Егер стильдердің каскадты кестелері қосылған болса, онда HTML тек объектілердің ретін сипаттайды. CSS олардың барлық қасиеттеріне жауап береді. HTML-де әрдайым барлық стильдерді тізімдемей, сыныпты тіркеу жеткілікті. Мұндай технология:

- Салыстырмалы түрде қарапайым және жылдам дамуды қамтамасыз етеді, өйткені бір рет жасалған дизайнды көптеген беттерге қолдануға болады;
- редакциялаудың икемділігі мен ыңғайлылығын арттырады – дизайнды барлық жерде өзгерту үшін CSS-ке түзетулер енгізу жеткілікті;
- элементтердің қайталануын азайту арқылы кодты қарапайым етеді. Бағдарламашылар мен іздеу боттарын оқу оңайырақ;
- жүктеу уақытын тездетеді, өйткені CSS бірінші рет ашылған кезде кэштелуі мүмкін, ал кейінірек тек құрылым мен деректер оқылады;
- мазмұнды көрсету үшін визуалды шешімдердің санын көбейтеді;

– бір құжатқа әртүрлі стильдерді оңай қолдануға мүмкіндік береді (мысалы, мобильді құрылғыларға арналған бейімделген нұсқаны немесе нашар көретіндерге арналған арнайы стильдерді жасау).
– яғни, каскадты кестелер дизайнды жүзеге асыру үшін ғана емес, сонымен қатар сайт құрылысына деген көзқарасты түбегейлі өзгертеді, әзірлеушілердің жұмысын жеңілдетеді және іске асырудың икемділігін қамтамасыз етеді.



8 – сурет. CSS жазылу түрі.

Bootstrap - бұл веб-әзірлеушілер жауап беретін веб-сайттар мен веб-қосымшалардың жылдам әзірлеу үшін қолданатын ашық және ақысыз HTML, CSS және JS фрэймворкі. Bootstrap фрэймворкін бүкіл әлемде тәуелсіз әзірлеушілер ғана емес, кейде бүкіл компаниялар да қолданады. Негізгі қолдану – бұл фронтенд сайттарды әзірлеу және админ интерфейстерін әзірлеу. Ұқсас жүйелер арасында (Foundation, UIkit, Semantic UI, InK және т.б.) Bootstrap шеңбері ең танымал. Bootstrap неге соншалықты танымал? Бұл сайттарды "таза" CSS және JavaScript-ке қарағанда бірнеше есе жылдам орналастыруға мүмкіндік береді. Біздің әлемде уақыт өте құнды ресурс. Оның тағы бір аспектісі – қол жетімділік. Бұл тіпті бастаушы веб-әзірлеушіге (терең білімі мен жеткілікті тәжірибесі жоқ) жеткілікті сапалы макеттер жасауға мүмкіндік береді.

Bootstrap мынадан тұрады:

– Орналасуды құруға арналған құралдар (орауыш контейнерлер, қуатты тор жүйесі, икемді медиа Нысандар, бейімделгіш утилиталар);
– негізгі мазмұнды сәндеуге арналған сыныптар: мәтін, суреттер, код, кестелер және сурет;
– дайын компоненттер: түймелер, пішіндер, көлденең және тік навигациялық панельдер, жүгірткілер, ашылмалы тізімдер, аккордеондар, модальды терезелер, кеңестер және т. б.;
– веб - әзірлеушілерге жиі кездесетін дәстүрлі мәселелерді шешуге арналған утилиталық сыныптар: мәтінді туралау, элементтерді көрсету және жасыру, түс, фон, маргин және падинг шегіністері және т. б.

Bootstrap қолданудың артықшылықтары:

- Жылдамдық. Тіпті жаңадан бастаушы веб-әзірлеушілердің де сапалы бейімделгіш макетін құрудың жоғары жылдамдығы (бұған мамандар жасаған дайын сыныптар мен компоненттерді қолдану арқылы қол жеткізіледі);
- кроссбраузерлік және кросс-платформа (осы шеңбермен қолдау көрсетілетін барлық браузерлерде және операциялық жүйелерде сайттың дұрыс көрсетілуі және жұмыс істеуі);
- әр түрлі құрылғыларда веб-әзірлеушілердің үлкен қауымдастығы сынаған дайын, жақсы ойластырылған компоненттердің көп болуы;
- өз жобаңызға баптау мүмкіндігі, бұған scss айнымалыларын өзгерту және миксиндерді қолдану арқылы қол жеткізіледі (бағандар санын, түстерді, дөңгелектеу радиусын, бағандар арасындағы шегіністерді және т. б. өзгертуге болады).);
- кіру шегі төмен; шеңбермен жұмыс істеу үшін HTML, CSS, JavaScript және jQuery туралы "терең" білімнің қажеті жоқ (тек осы технологиялардың негіздерін білу жеткілікті);
- дизайнның біркелкілігі және оның әртүрлі компоненттер арасындағы сәйкестігі (Bootstrap-да барлық компоненттер бірдей стильде жасалады);
- көптеген қауымдастықтар мен оқу материалдарының болуы; егер қаласаңыз, бұл шеңберді жақсы түсінуге ғана емес, сонымен қатар сізде туындаған кез-келген сұрақтарға жауап табуға көмектеседі.

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/normalize.css@8.0.0/normalize.min.css">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.min.css">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@4.7.0/css/font-awesome.min.css">
```

9 – сурет. Bootstrap-ты өз файлыңызға қосу жолы.

Razor - C# бағдарламалау тілін қолдайды және HTML-ден C#-қа өту үшін @ таңбасын қолданады. Razor C# өрнектерін есептейді және оларды HTML шығуында көрсетеді. Егер @ таңбадан кейін Razor сақталған кілт сөзі болса, ол Razor таңбасына белгілі бір мәнге ауысады. Әйтпесе, ол әдеттегі C# - ге ауысады.

ASP.NET MVC әрқашан "көрініс қозғалтқышы" тұжырымдамасын қолдады, іс жүзінде ол шаблонның әртүрлі синтаксисін таңдауды жүзеге асыратын ауыстырылатын модульдер болып табылады. Бүгін, ASP.NET MVC стандартты көрініс қозғалтқышы ASP.NET Web Forms -.aspx/.ascx/.master ұқсас файлдарды қолданады. Басқа танымал түрлері бар ASP.NET MVC көрініс қозғалтқыштары-Spark және NHaml.

Ықшам, мәнерлі және икемді: Razor файлда жұмыс істеу үшін қажет "пернетақтадағы жүктемені" азайтады, жылдам және икемді бағдарламалау

процесін қамтамасыз етеді. Көптеген шаблон синтаксисінен айырмашылығы, HTML-де серверлік код блоктарын нақты анықтау үшін кодты жазуды үзудің қажеті жоқ. Талдаушы сіз үшін код негізінде жасау үшін жеткілікті ақылды. Сіз теру оңай және көңілді болатын ықшам және экспрессивті синтаксисті аласыз.

Меңгеру оңай: Razor-ны үйрену оңай және қысқа мерзімде ең аз күш-жігермен нәтижелі болуға мүмкіндік береді. Сіз бағдарламалау тілі мен HTML-дің барлық дағдыларын қолданасыз.

Жаңа тіл емес: біз саналы түрде Razor үшін жаңа императивті тілге негізделмейтін жолды таңдадық. Оның орнына, біз әзірлеушілерге C#/VB(немесе басқа тілде) бұрыннан бар дағдыларды қолдануға мүмкіндік бердік және шаблонды белгілеу синтаксисін ұсындық, бұл сізге HTML құрудың керемет процесін береді, тілді таңдауды сізге қалдырады.

Кез-келген мәтіндік редакторда жұмыс істейді: Razor белгілі бір құралды қажет етпейді және кез-келген ескі мәтіндік редакторда нәтижелі болуға мүмкіндік береді (тіпті ноутбук өте жақсы жұмыс істейді) Intellisense қолдауы: Razor белгілі бір құралға немесе Код редакторына сілтеме жасамауды ескере отырып жасалған болса да, Visual Studio-да керемет қолдау болады. Visual Studio 2010 және Visual Web Developer 2010 жақында intellisense үшін жаңартуларды алады. Бірлікті тестілеу: Қозғалтқыштың жаңа іске асырылуы көріністерге бірлік сынақтарын өткізу мүмкіндігін қолдайды (контроллерді немесе веб-серверді қажет етпестен, кез-келген блок-тестілеу жобасына орналастыруға болады - арнайы қолданбалы домендердің қажеті жоқ).

Файлды жүктеуді орындаудын 4 әдісі бар. Олар:

- FileContentResult: файлдан саналған байт массивін жібереді
- VirtualFileResult: виртуалды жолды көрсететін серверде орналасқан файлды қарапайым жіберу
- FileStreamResult: System.IO.Stream ағынынан файлды оқиды және клиентке файлды жібереді
- PhysicalFileResult: файлды қарапайым жіберу, тек толық физикалық жол көрсетілген

```
public class HomeController : Controller
    {
        private readonly IHostingEnvironment appEnvironment;
        public HomeController(IHostingEnvironment appEnvironment)
        {
            this.appEnvironment = appEnvironment;
        }
        public IActionResult DownloadFile()
        {
            string filePath = Path.Combine(appEnvironment.ContentRootPath,
"Files/book.pdf");
            string fileType = "application/pdf";
            string fileName = "book.pdf";
            return PhysicalFile(filePath, fileType, fileName);
        }
    }
```

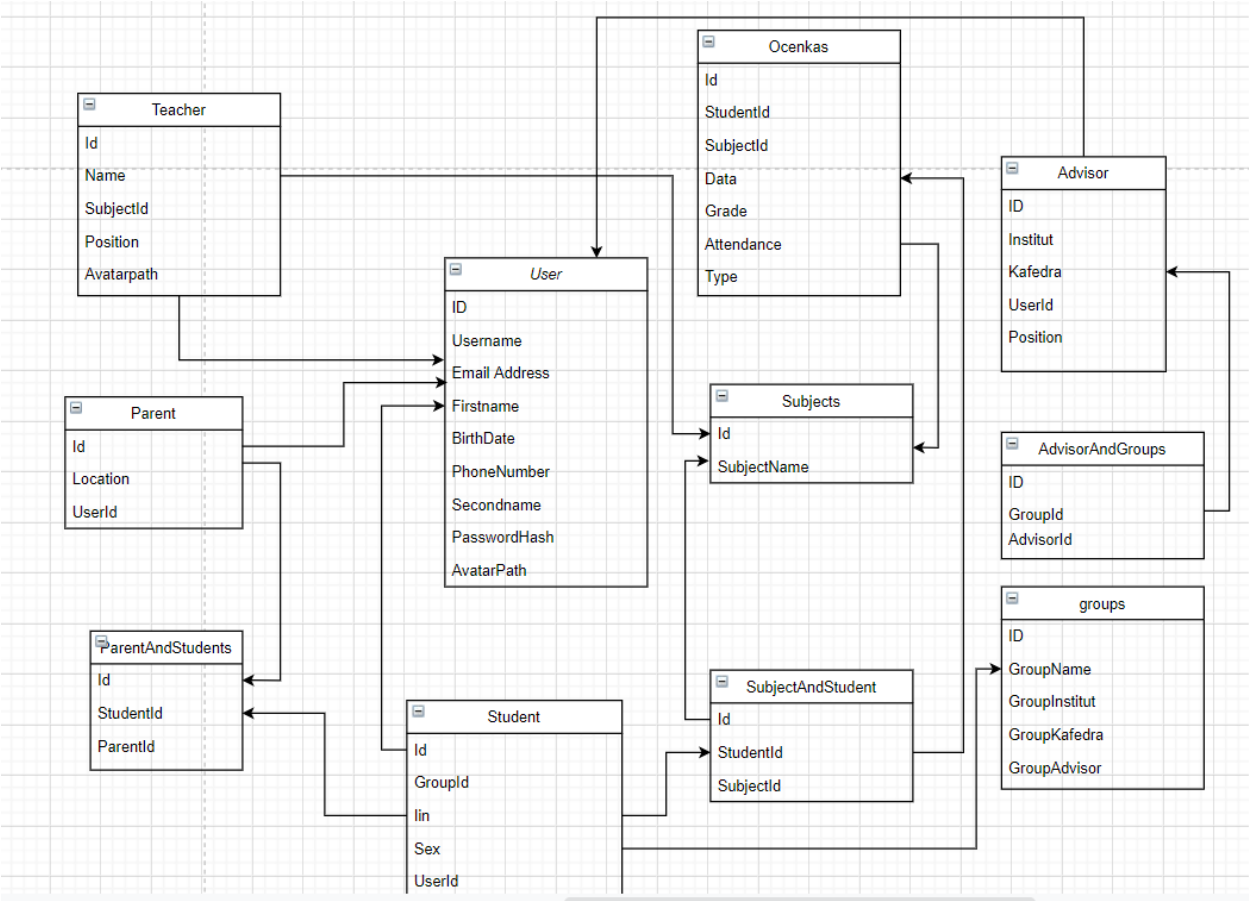
10 – сурет. PhysicalFileResult әдісінің мысалы.

2.9 Мәліметтер базасының ER – моделі

ER-модель (ағылш. Entity-Relationship model, "мәні — байланыс" моделі) - пәндік саланың тұжырымдамалық сұлбаларын сипаттауға мүмкіндік беретін деректер моделі. ER моделі жоғары деңгейлі (тұжырымдамалық) мәліметтер базасын жобалауда қолданылады. Оның көмегімен сіз негізгі нысандарды бөліп, осы нысандар арасында орнатуға болатын байланыстарды белгілей аласыз. Деректер базасын жобалау кезінде таңдалған деректер моделіне (реляциялық, Объектілік, желілік немесе т.б.) негізделген ER-модель негізінде құрылған схеманы нақты дерекқор схемасына түрлендіру жүзеге асырылады. ER-модель-бұл өзін-өзі бейнелеудің ешқандай графикалық құралдарын тағайындамайтын ресми дизайн. ER-модельді визуализациялауға болатын стандартты графикалық нотация ретінде "субъект-байланыс" (ағылш. Entity-Relationship diagram, ERD, ER-диаграмма). "ER-модель" және "ER-диаграмма" ұғымдары көбінесе ерекшеленбейді, дегенмен ER-модельдерді визуализациялау үшін басқа графикалық белгілерді қолдануға болады немесе визуализация мүлдем қолданылмауы мүмкін (мысалы, мәтіндік сипаттама қолданылады).

Мәліметтер базасының ER – моделін жасау үшін UML графикалық сипаттамалау тілін қолдандым. UML бұл - бағдарламалық жасақтама саласындағы объектіні модельдеуге, бизнес-процестерді модельдеуге, жүйелік жобалауға және ұйымдық құрылымдарды көрсетуге арналған графикалық сипаттама тілі. UML — кең

профиль тілі, бұл UML моделі деп аталатын жүйенің абстрактілі моделін құру үшін графикалық белгілерді қолданатын ашық стандарт. UML негізінен бағдарламалық жүйелерді анықтау, визуализация, жобалау және құжаттау үшін жасалған. UML бағдарламалау тілі емес, бірақ UML модельдерінің негізінде кодты құруға болады.

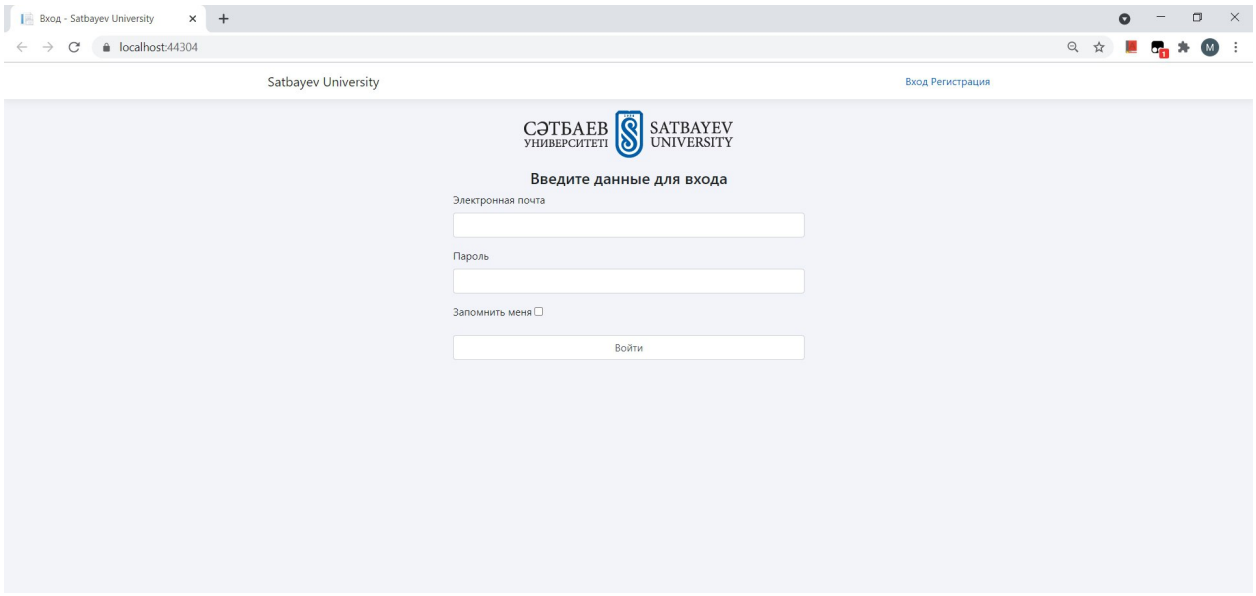


11 – сурет. Мәліметтер базасының ER моделі

3 ПАРАҚШАЛАРДЫҢ ПРОТОТИПІН ЖАСАУ

3. 1 Жүйенің жұмыс істеуін сипаттау

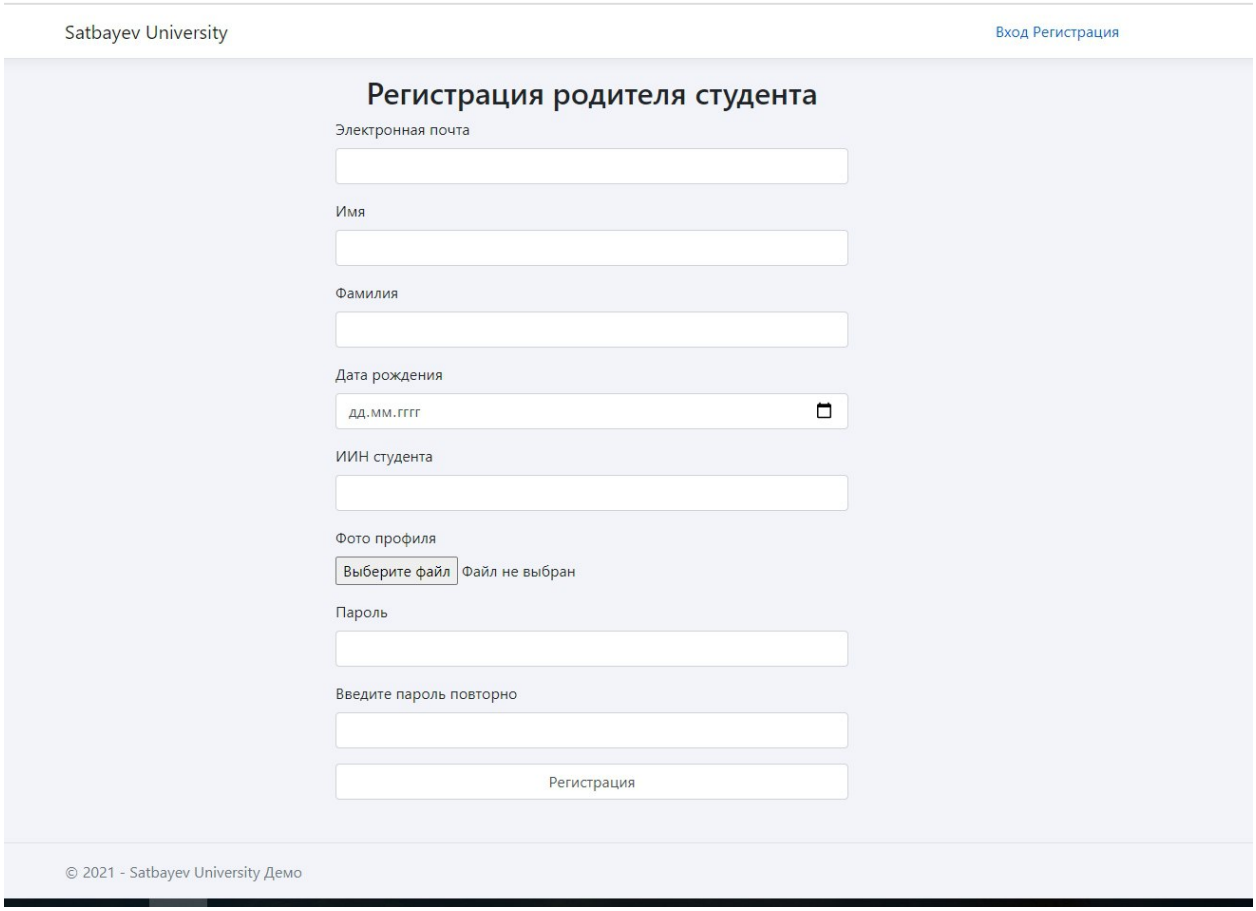
Жасалған дипломдық жобамды Visual Studio 2019 бағдарламалау ортасына кіріп, IIS Server арқылы қосуға болады. Іске асырған кезде бастапқы беті Сәтбаев университетінің білім беру жүйесіндегідей авторизация бөлімі ашылады. Онда тіркелу кезінде жазылған логин және құпия сөзді енгізіп кіру керек.



12 – сурет. Жобаның бастапқы беті.

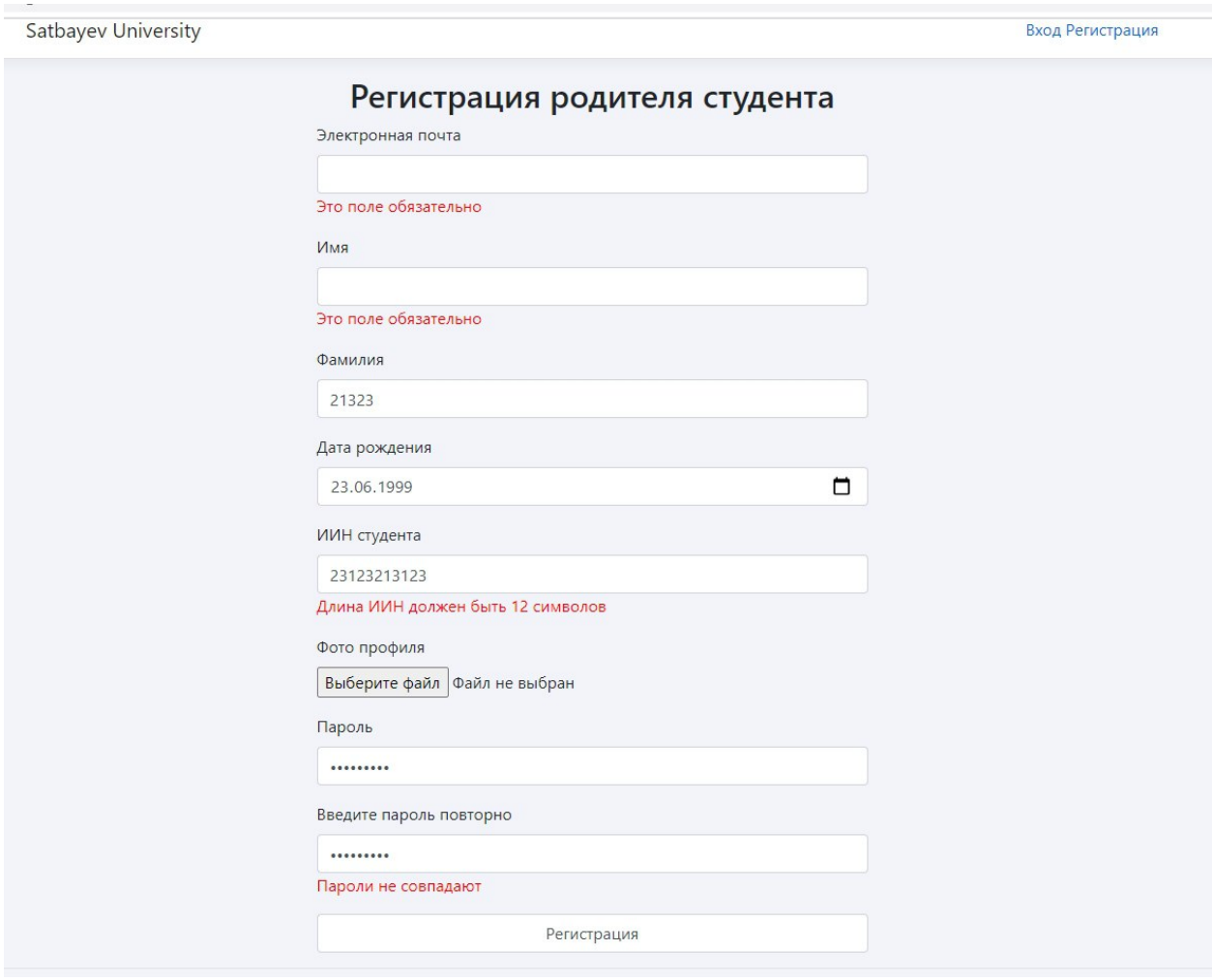
Көріп тұрғандай бастапқы беттің дизайнын ашық түстес қылып жасадым және кірген адамға түсінікті болатындай университеттің логотипін қойдым.

Егер кірген пайдаланушы жүйеге тіркелмеген болса, онда жоғарғы жақта тұрған тіркелу батырмасын басып келесі бетке өту керек. Тіркелу бетін тек ата-аналарға арналған. Себебі тек оларда ғана тіркелмеген, ал эдвайзерлердің барлығы дерлік білім беру жүйесінде тіркелген. Егер эдвайзерлер де міндетті түрде тіркелетін болса, онда мәліметтер базасында ақпараттар қайталанып, мәліметтер базасының тіпті істен шығып жобамның жұмыс жасамай қалуына алып келуі мүмкін.



13 – сурет. Тіркелу беті.

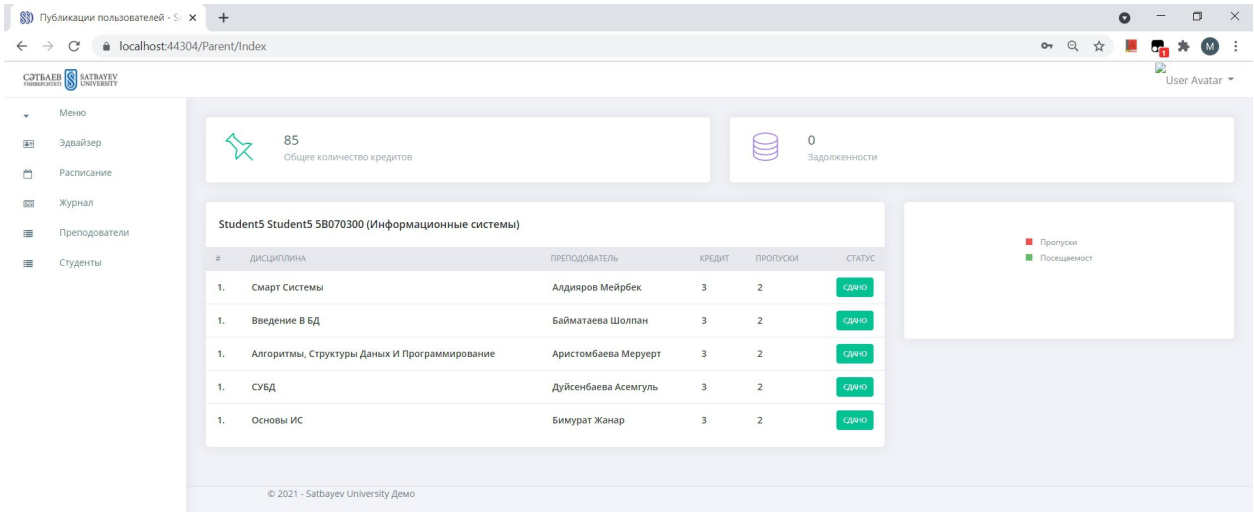
Тіркелу үшін пайдаланушы электрондық поштасын, атын, тегін, туылған жылын және өзінің баласының жеке сәйкестендіру нөмерін, яғни ЖСН (ИИН), жазу керек. Содан соң қалаған жағдайда өзінің суретін енгізсе болады, соңғысы ол құпия сөз орнату керек. Толтырылатын өрістердің барлығын міндетті түрде толтырылу керек, тек суретті ғана өз қалауымен ғана. Өрістерді тексеретін валидация жазылған. Оны келесі суретте көруге болады.



13 – сурет. Тіркелу бетінің валидациясы.

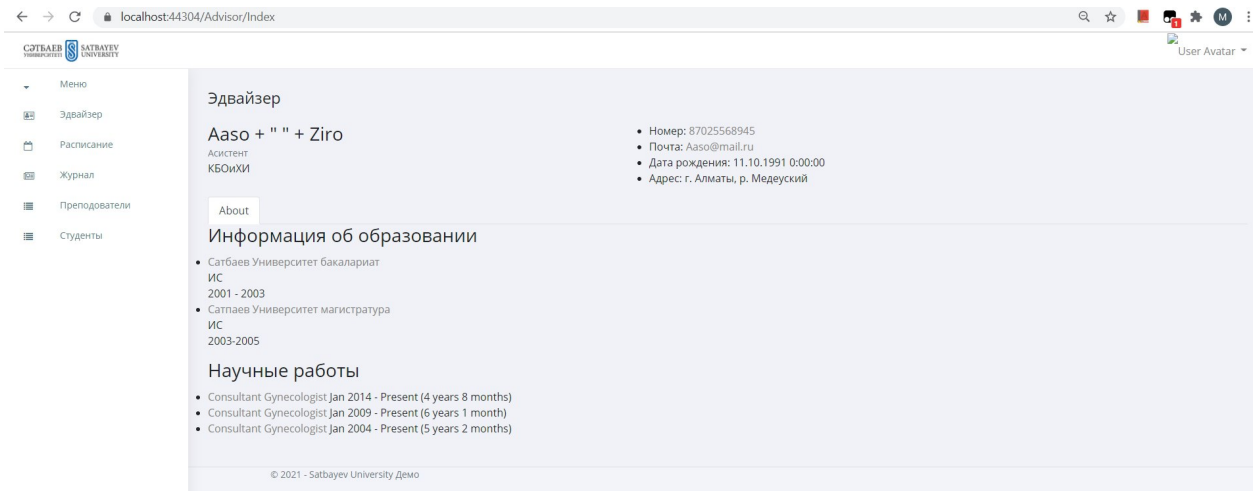
Валидация бойынша екі рет жазылған құпия сөз бірдей болу керек және құпия сөзді жазғанда ұзындығы 8 таңбадан кем болмау керек. 8 таңбанын ішінде міндетті түрде үлкен қаріптегі таңба, кіші қаріптегі таңба және сан болу керек. ЖСН 12 сан болу керек, 12-ден көп және аз болмайды.

Пайдаланушы тіркеліп, жүйеге кірген соң негізгі беті ашылады.



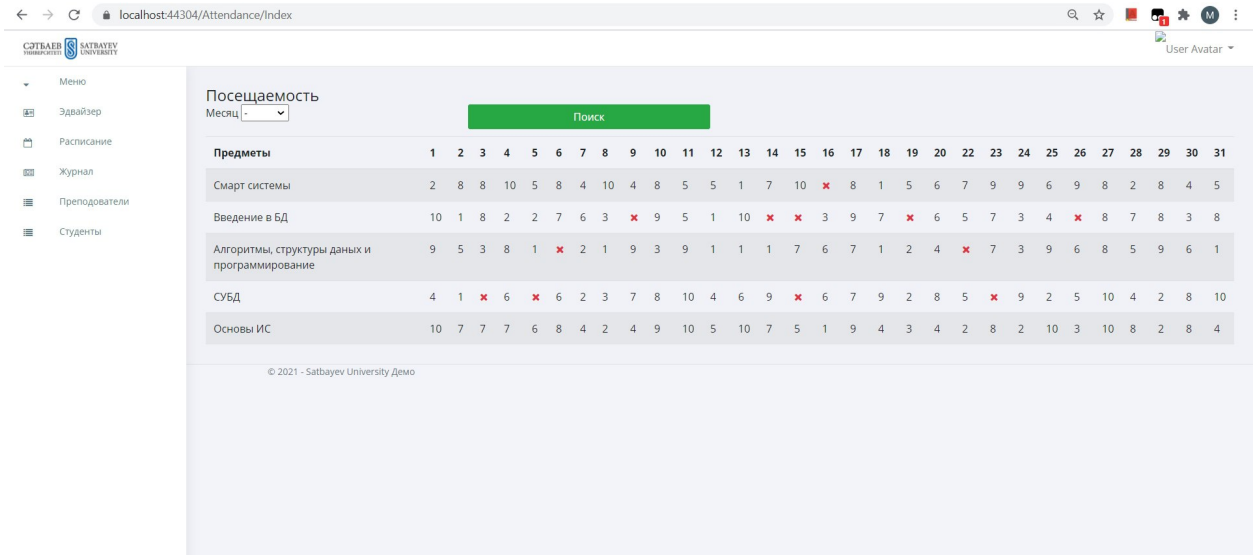
14 – сурет. Негізгі беті.

Негізгі бетте біріншіден студенттің аты-жөні және мамандығы жазылып тұрады. Семестрдегі оқылып жатқан пән аттары және пән мұғалімдерінің есімдері жазылады. Пән жайлы кішігірім ақпаратта болады. Егер жоғарғы курс студенттері болса, осы уақытқа дейін қанша кредит сабақ оқығаны, қарыздары болса қарыз саны жазылады. Жоғарғы оң жақта пайдаланушы профилі болады, батырманы басқанда тізбек шығады. Тізбекте шығу, жеке кабинет, құпия сөзді ауыстыру батырмалары бар.



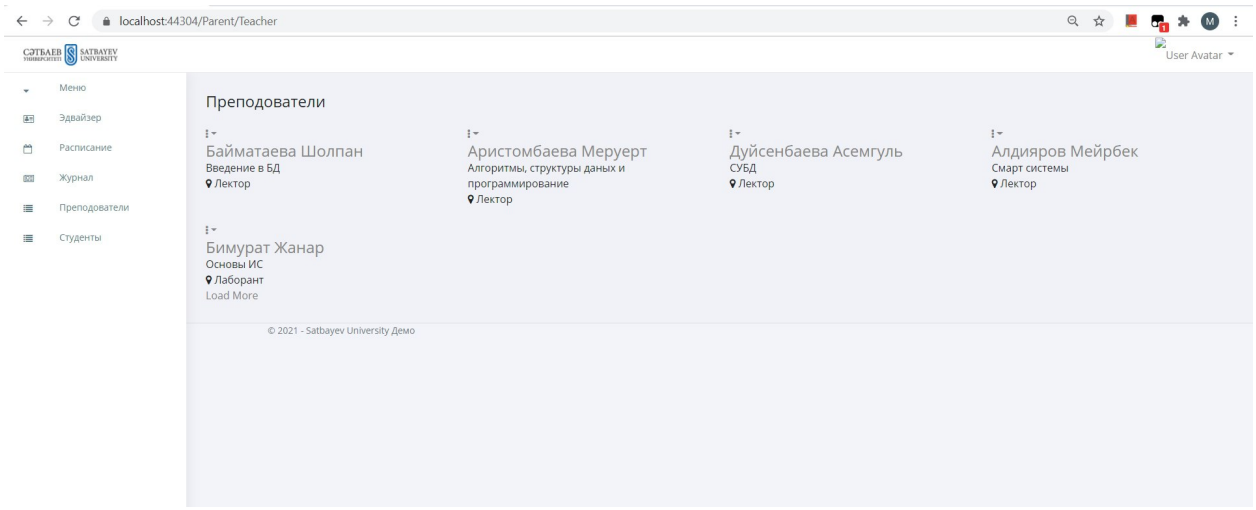
15 – сурет. Ата-ана үшін эдвайзер парақшасы.

Бұл бетте пайдаланушы эдвайзер жайлы мағлұмат ала алады. Оның есімі, қай кафедраның мұғалімі, лауазымы, электронды поштасы және т.б. жазылады. Мұғалімнің қай жоғарғы оқу орнын тәмәмдағаны да жазылып тұрады, егер жазылған ғылыми жобасы болса олда жазылып тұрады.



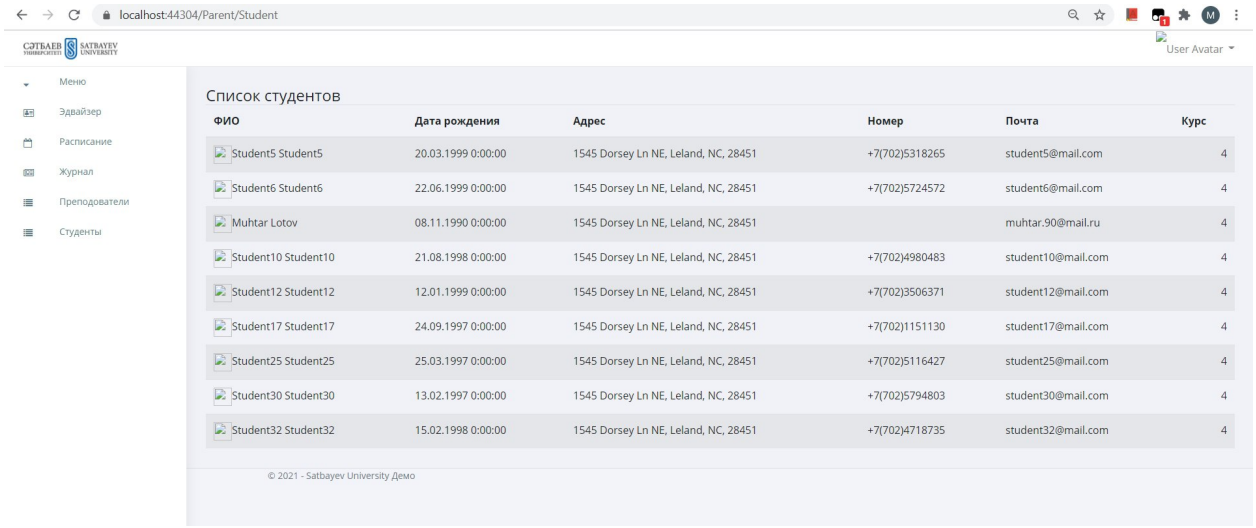
16 – сурет. Журнал беті.

Бұл бетте студенттің оқу үлгерімі көрсетілген. Сабаққа қатысу қатыспауы, сол сабақтан алған бағасы. Менің жобам прототип болған соң бағалау жүйесін 1-10 балл арасында қойдым. Егер студентте сабаққа қатыспаған болса қызыл белгі тұрады. Бетте 1 айдын көлеміндегі ақпарат тұрады, егер одан алдығы айлардыкын көргісі келсе жоғарыда тұрған фильтрді қолдануға болады. Келесі мұғалімдер бетінде мұғалімдер тұрады.



17 – сурет. Мұғалімдер беті.

Пән мұғалімдеріні суреттері және есімі, қай пәннен сабақ өтетіні және сабақ түрі жазылады, яғни лекция, практикалық сабақ немесе зертханалық жұмыс.



18 – сурет. Студенттер беті.

Бұл бетте топта бірге оқитын студенттер тізімі шығады. Менің ойымша кез-келген ата-ана баласының кіммен бірге оқып жүргені қызық болады деп

есептеймін, сондықтан осы бетті қосқан болатынмын. Студенттер жайлы мәліметтер тестовый болып келеді сондықтанда барлығынын аты бірдей болып тұр.

Егер эдвайзер кірген болса студенттер парағында тізімде тұрған кез-келген студенттің жеке парақшасын қарай алады, ал ата-ана тек өзінің баласынықін қарай алады.

ҚОРЫТЫНДЫ

Осы дипломдық жұмыста пәндік саланы зерттеп және талдау жүргізілді. Талдау нәтижесінде керек парақшалардың тізімі жасалынды. Сол тізім бойынша парақшалардың техникалық тапсырмасы әзірленді. Техникалық тапсырма екі парақшаға бөлініп, әр парақшада болу керек функционалдар ескерілді. Мақсатқа сай пайдаланушыларға ыңғайлы интерфейс және де ата-ана мен эдвайзерлердің жұмыс істеуіне ынғайлы прототип парақшалар жасалынып шықты. Димпломдық жұмысымды төменде көрсетілгендей тапсырмалар орындалды:

- Ата-ана және эдвайзер парақшасы;
- Студенттің оқу үлгерімін бақылау;
- Ата-ана үшін эдвайзер жайлы ақпарат, ал эдвайзер үшін ата-ана жайлы ақпарат;
- Талапқа сай мәліметтер базасы;
- Пайдаланушылардың құпия сөзін өзгерту.

Еліміздегі жоғарғы оқу орындарының білім беру жүйесінде осындай функционалы бар жүйе жоқ болған соң, бұл жобаны тек Сәтбаев университеті емес барлық ЖОО-на қолдануға болады деп есептеймін. Жобаны жасау барысында алдынғы технологиялар қолданды. Осы технологиялар арқасында қолдануға оңай және дизайны адаптивті парақшалар әзірленді.

Visual Studio 2019 бағдарламалық кешен ортасында жасалып, мәліметтер базасын басқару жүйесі PostgreSQL таңдалды. Жұмысты және техникалық тапсырманы жобалау кезіндегі тапсырмалар мен функционалдардың барлығына қол жеткізілді. Бұл жүйе ата-ана мен эдвайзерлердің уақытын үнемдеп және жұмысын жеңілдетеді деген үміттемін.

ПАЙДАЛЫНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1 Допира, Р. И. Технология ASP.NET MVC / Р. И. Допира, Н. В. Попова. — Текст : непосредственный // Молодой ученый. — 2018. — № 49 (235). — С. 17-20
2 https://metanit.com/sharp/
3 https://ru.wikipedia.org/
4 https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-5.0
5 https://metanit.com/sharp/aspnet5/3.1.php
6 Ригс, Саймон Администрирование PostgreSQL 9. Книга рецептов / Саймон Ригс , Ханну Кросинг. - М.: ДМК Пресс, 2017. - 364 с.
7 Сергеев А. Н. Социальная сеть и сервисы учебного назначения образовательного портала ВГСПУ // Грани познания. 2015. №6 (40).
8 Уорсли, Дж. PostgreSQL. Для профессионалов (+ CD) / Дж. Уорсли, Дж. Дрейк. - М.: СПб: Питер, 2016. - 496 с.
9 Шмитт К. Рецепты программирования CSS. Cookbook. - СПб.: БХВ-Петербург, 2007. - 656 с.
10 Айтхожаева Е.Ж. Системы баз данных. Учебник. - Алматы, КазНТУ, 2002.

А қосымшасы

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Hosting;
using sbs.Models;
using sbs.Services;
using sbs.ViewModels;

namespace sbs.Controllers
{
   public class ParentController : Controller
   {

      private BlogContext _db;
      private readonly IHostEnvironment _environment;
      private readonly UploadFileService _uploadFileService;
      private UserManager<User> _userManager;
      public ParentController(Models.BlogContext db, IHostEnvironment
environment, UploadFileService uploadFileService, UserManager<User>
userManager)
      {
         _db = db;
         _environment = environment;
         _uploadFileService = uploadFileService;
         _userManager = userManager;
      }
      [Authorize]
      public async Task<IActionResult> Index()
      {

         if (User.IsInRole("parent") || User.IsInRole("admin"))
            {
         Parent parent = _db.Parents.FirstOrDefault(p => p.User.Id ==
_userManager.GetUserId(User));
         ParentAndStudent user = _db.ParentAndStudents.FirstOrDefault(p =>
p.ParentsId == parent.Id);
         Student Student = _db.Students.FirstOrDefault(s => s.Id ==
user.StudentId);
            List<Teacher> txt = new List<Teacher>();
```

```
            ParentIndexViewModel viewModel = new ParentIndexViewModel
            {
                student = Student,
                subjectAndStudents = _db.SubjectAndStudents.Where(s =>
s.StudentId == Student.Id).ToList()
            };
            foreach (var item in viewModel.subjectAndStudents)
            {
                txt.Add(_db.Teachers.FirstOrDefault(t => t.SubjectId ==
item.SubjectId));
            }
            viewModel.Teacher = txt;
            return View(viewModel);
        }
        else if (User.IsInRole("advisor") || User.IsInRole("admin"))
        {
            return RedirectToAction("Index", "Advisor");
        }

        return View();
    }

    public IActionResult Student()
    {
        Parent parent = _db.Parents.FirstOrDefault(p => p.User.Id ==
_userManager.GetUserId(User));
        ParentAndStudent user = _db.ParentAndStudents.FirstOrDefault(p =>
p.ParentsId == parent.Id);
        Student Student = _db.Students.FirstOrDefault(s => s.Id ==
user.StudentId);
        List<Student> Students = _db.Students.Where(s => s.GroupId ==
Student.GroupId).ToList();

        return View(Students);
    }
    public IActionResult Schedule()
    {
        Parent parent = _db.Parents.FirstOrDefault(p => p.User.Id ==
_userManager.GetUserId(User));
        ParentAndStudent user = _db.ParentAndStudents.FirstOrDefault(p =>
p.ParentsId == parent.Id);
        Student Student = _db.Students.FirstOrDefault(s => s.Id ==
user.StudentId);
```

А қосымшасының жалғасы

```
return View(Student);
        }
        public IActionResult Teacher()
        {
            List<Teacher> teachers = _db.Teachers.ToList();
            return View(teachers);
        }
    }
}
```

Б қосымшасы

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Hosting;
using sbs.Models;
using sbs.Services;
using sbs.ViewModels;

namespace sbs.Controllers
{
   public class AccountController : Controller
   {
      private BlogContext _db;
      private readonly UserManager<User> _userManager;
      private readonly RoleManager<IdentityRole> _roleManager;
      private readonly SignInManager<User> _signInManager;
      private readonly IHostEnvironment _environment; //Добавляем сервис
взаимодействия с файлами в рамках хоста
      private readonly UploadFileService _uploadFileService; // Добавляем сервис
для получения файлов из формы
      public AccountController(
         BlogContext db,
         UserManager<User> userManager,
         RoleManager<IdentityRole> roleManager,
         SignInManager<User> signInManager,
         IHostEnvironment environment,
         UploadFileService uploadFileService)
      {
         _db = db;
         _userManager = userManager;
         _roleManager = roleManager;
         _signInManager = signInManager;
         _environment = environment;
         _uploadFileService = uploadFileService;
      }
      public IActionResult Register()
      {
         return View();
```

```
}
        [Authorize]
        public IActionResult Index(string id = null)
        {
            User user = id == null ? _userManager.GetUserAsync(User).Result :
_userManager.FindByIdAsync(id).Result;
            return View(user);
        }
        [Authorize]
        public IActionResult Edit(string id = null)
        {
            User user = null;
            if (User.IsInRole("admin"))
            {
                user = id == null ? _userManager.GetUserAsync(User).Result :
_userManager.FindByIdAsync(id).Result;
            }
            else
            {
                user = _userManager.FindByIdAsync(id).Result;
            }
            UserEditViewModel model = new UserEditViewModel()
            {
                FirstName = user.FirstName,
                SecondName = user.SecondName,
                BirthDate = user.BirthDate,
                Id = user.Id
            };
            return View(model);
        }
        [HttpPost]
        [Authorize]
        public async Task<IActionResult> Edit(UserEditViewModel model)
        {
            if (ModelState.IsValid)
            {
                User user = await _userManager.FindByIdAsync(model.Id);
                if (user != null)
                {
                    user.FirstName = model.FirstName;
                    user.SecondName = model.SecondName;
                    user.BirthDate = model.BirthDate;
                    var result = await _userManager.UpdateAsync(user);
```

```
                if (result.Succeeded)
                    return RedirectToAction("Index");
                foreach (var error in result.Errors)
                    ModelState.AddModelError("", error.Description);
            }
        }
        return View(model);
    }
    [Authorize]
    public async Task<IActionResult> ChangePassword(string id)
    {
        User user = await _userManager.FindByIdAsync(id);
        if (user is null)
            return NotFound();
        ChangePasswordViewModel model = new ChangePasswordViewModel()
        {
            Id = user.Id,
            Email = user.Email
        };
        return View(model);
    }
    [HttpPost]
    [Authorize]
    public async Task<IActionResult>
ChangePassword(ChangePasswordViewModel model)
    {
        if (ModelState.IsValid)
        {
            User user = await _userManager.FindByIdAsync(model.Id);
            if (user != null)
            {
                var passwordValidator =
HttpContext.RequestServices.GetService(typeof(IPasswordValidator<User>)) as
IPasswordValidator<User>;
                var passwordHasher =
HttpContext.RequestServices.GetService(typeof(IPasswordHasher<User>)) as
IPasswordHasher<User>;
                var result = await passwordValidator.ValidateAsync(_userManager,
user, model.NewPassword);
                if (result.Succeeded)
                {
                    user.PasswordHash = passwordHasher.HashPassword(user,
model.NewPassword);
```

Б қосымшасының жалғасы

```
                await _userManager.UpdateAsync(user);
                    return RedirectToAction("Index");
                }
                foreach (var error in result.Errors)
                    ModelState.AddModelError("NewPassword", error.Description);
            }
            ModelState.AddModelError("", "Пользователь не существует");
        }
        return View(model);
    }
    [HttpPost]
    public async Task<IActionResult> Register(RegisterViewModel model)
    {
        if (ModelState.IsValid)
        {
            User user = new User
            {
                Email = model.Email,
                UserName = model.Email,
                BirthDate = model.DateOfBirth,
                FirstName = model.FirstName,
                SecondName = model.SecondName,

             };

            if (model.File != null)
            {
                string path = Path.Combine(_environment.ContentRootPath,
"wwwroot\\images\\");
                string photoPath = $"images/{model.File.FileName}";
                _uploadFileService.Upload(path, model.File.FileName, model.File);
                user.AvatarPath = photoPath;

            };

            var result = await _userManager.CreateAsync(user, model.Password);
            if (result.Succeeded) {
                await _userManager.AddToRoleAsync(user, "parent");
            };

            Parent parent = new Parent
            {
```

Б қосымшасының жалғасы

```
            UserId = user.Id
              };
              _db.Parents.Add(parent);
              Student student = _db.Students.FirstOrDefault(s => s.Iin ==
Convert.ToInt64(model.Iin));
              ParentAndStudent parentAndStudent = new ParentAndStudent();
              parentAndStudent.ParentsId = parent.Id;
              parentAndStudent.StudentId = student.Id;
              _db.ParentAndStudents.Add(parentAndStudent);
              _db.SaveChanges();

              if (result.Succeeded)
              {
                 await _signInManager.SignInAsync(user, false);
                 return RedirectToAction("Index", "Parent");
              }
              foreach (var error in result.Errors)
                 ModelState.AddModelError(String.Empty, error.Description);

           }
           return View(model);
        }
}
```

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Сәтбаев университеті

Ғылыми жетекшінің пікірі

Дипломдық жұмыс

Сейтмаханбетов Бейбарыс Сарсенкелдыулы

5В070300 – Ақпараттық жүйелер

Тақырыбы: Сәтбаев Университетінің білім беру жүйесінің эдвайзер, ата-аналар парағының прототипін және техникалық тапсырмасын әзірлеу

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 Сейтмаханбетов Бейбарыс Сарсенкелдыулы бакалавр академиялық дәрежесін алуға лайықты деп есептеймін.

Ғылыми жетекші

Лектор Зиро А.А.

« _» _____ 2021 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: Сейтмаханбетов Бейбарыс

Название: Сәтбаев Университетінің білім беру жүйесінің эдвайзер, ата-аналар парағының прототипін және техникалық тапсырмасын әзірлеу

Координатор: Зира Аасо

Коэффициент подобия 1: 8,0

Коэффициент подобия 2: 2,13

Замена букв: 21

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:

После анализа отчета по плагиату и работы дипломника выявлено, что заимствования являются добросовестными и не обладают признаками плагиата, так в основном связаны с применением общеизвестных терминов.

26.05.2021….. ……

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: Сейтмаханбетов Бейбарыс

Название: Сәтбаев Университетінің білім беру жүйесінің эдвайзер, ата-аналар парағының прототипін және техникалық тапсырмасын әзірлеу

Координатор: Зиро Аасо

Коэффициент подобия 1: 8,00

Коэффициент подобия 2: 2,13

Замена букв: 21

Интервалы: 0

Микропробелы: 0

Белые знаки: 0

После анализа отчета подобия заведующий кафедрой / начальник структурногоподразделения констатирует 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:

После анализа отчета по плагиату и работы дипломника выявлено, что заимствования являются добросовестными и не обладают признаками плагиата, так в основном связаны с применением общеизвестных терминов.

27.05.2021 ……

Дата *Подпись заведующего кафедрой /начальника*

структурного подразделения