МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий

Кафедра "Программная инженерия"

Орынбасаров Бекарыс Талапұлы

Разработка веб-приложения автоматического сбора акустических данных для создания акустического корпуса казахского языка

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

Специальность 5В070400 – Вычислительная техника и программное обеспечение

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий

Кафедра "Программная инженерия"

5В070400 – Вычислительная техника и программное обеспечение

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
доктор Ph.D
_____Тұрдалыұлы М.
"05" 06 2021 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
На тему: " Разработка веб-приложения автоматического сбора акустических данных для создания акустического корпуса казахского языка" по специальности 5В070400 – Вычислительная техника и программное обеспечение

Выполнил Орынбасаров Б.Т.

Научный руководитель
сениор-лектор, маг. тех. наук
_____Бекарыстанқызы А.
"08" 06 2021 г.

Алматы 2021

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

5В070400 – Вычислительная техника и программное обеспечение

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
доктор Ph.D
_____Тұрдалыұлы М.
"05" 06 2021 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся *Орынбасарову Бекарысу Талапұлы*
Тема: *Разработка веб-приложения автоматического сбора акустических данных для создания акустического корпуса казахского языка*
Исходные данные к дипломному проекту: *Описание необходимых функций проекта.*
Перечень подлежащих разработке в дипломном проекте вопросов:
а) реализация функциональных возможностей текстового редактора
б) имплементация системы для сбора акустических данных;
в) проектирование и разработка пользовательского интерфейса;
г) разработка, отладка и тестирование акустических и текстовых данных;
Перечень графического материала (с точным указанием обязательных чертежей): *представлены 14 слайдов презентации.*

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю и консультантам	Примечание
1. Анализ предметной области, разработка технического задания	08.04.21	Выполнено
2. Выбор базового компонента	14.04.21	Выполнено
3. Разработка дизайна интерфейса	16.04.21	Выполнено
4. Разработка системы для сбора акустических данных	20.04.21	Выполнено
5. Разработка функционала текстового редактора	10.05.21	Выполнено
6. Тестирование текстового редактора	15.05.21	Выполнено
7. Написание пояснительной записки к дипломному проекту	19.05.21	Выполнено

Подписи

консультантов и нормоконтролера на законченный дипломный проект
с указанием относящихся к ним разделов проекта

Наименования разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Программное обеспечение	Рамазан А. Б.	31.05.2021	
Нормоконтролер	Марғұлан Қ.	06.06.2021г	

Научный руководитель _____Бекарыстанқызы А.

Задание принял к исполнению обучающийся _____Орынбасаров Б.Т.

Дата "07" 06 2021г.

АННОТАЦИЯ

Данный дипломный проект посвящен разработке автоматического сбора акустических и текстовых данных для создания корпуса казахского языка. Корпус казахского языка – это многофункциональный справочник о языке, в котором можно проследить историю и развитие языка, можно делать научные исследования лексики и грамматики.

Разработанный инструмент позволит упростить работу по сбору и хранению акустических и текстовых данных таким образом что пользователю нет необходимости следить за данными или же отправлять их.

Для реализации проекта использовался “Asp.NET Core MVC” – это модульная платформа для разработки ПО на языке “c#” от компаний “Microsoft”. Проект разрабатывался в многофункциональном редакторе “Visual Studio”.

АНДАТПА

Бұл дипломдық жоба қазақ тілінің корпусын құру үшін акустикалық және мәтіндік деректерді автоматты түрде жинауды дамытуға арналған. Қазақ тілінің корпусы – бұл тіл туралы көпфункционалды анықтамалық, онда сіз тілдің тарихы мен дамуын бақылай аласыз, сөздік пен грамматика бойынша ғылыми зерттеулер жасай аласыз.

Әзірленген құрал акустикалық және мәтіндік деректерді жинау және сақтау жұмыстарын жеңілдетеді, пайдаланушыға деректерді бақылаудың немесе оны жіберудің қажеті болмайды.

Жобаны іске асыру үшін “Asp.NET Core MVC” қолданылды - бұл Microsoft корпорациясының “c#” тіліндегі бағдарламалық жасақтамасын әзірлеуге арналған модульдік платформа. Жоба “Visual Studio” көпфункционалды редакторында жасалған.

ANNOTATION

This thesis project is dedicated to the development of automatic collection of acoustic and text data for the creation of the Kazakh language corpus. The corpus of the Kazakh language is a multifunctional reference book about the language in which you can trace the history and development of the language, you can do scientific research of vocabulary and grammar.

The developed tool will simplify the work of collecting and storing acoustic and text data in such a way that the user does not need to monitor the data or send it.

To implement the project, “Asp.NET Core MVC” was used - this is a modular platform for developing software in the “c #” language from “Microsoft”. The project was developed in the multifunctional editor “Visual Studio”.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Создание корпуса национального языка – одна из самых важных задач государства, в котором затрагиваются вопросы истории развития языка и культуры целой нации. Это трудоемкий процесс, но имеет свои плюсы, это - количество перспектив для исследования и развитие языка, но по стечению времени, из – за многочисленных факторов как глобализация, диалекты, зарубежные термины и др. меняют сам разговорный язык. Язык меняется так же стремительно, как и наш мир, из-за этого нужно постоянно обновлять данные.

Создание различных обучающихся моделей стали бы хорошим решением данной проблемы, но есть одна большая проблема – сбор огромного количества тестовых и обучающих данных. Такая проблема всегда актуальна в таких системах. Чем больше объектов для распознавания, тем гораздо больше данных потребуется для обучения, а самоличное составление этих данных могут занять огромное количество времени.

Актуальность данного проекта заключается в том, что он упрощает сбор и хранение и редактирование данных, которые будут использованы для обучение модели. Кроме того, по мере появлении новых требований проект может быть легко модернизирован для решение схожих проблем.

1 Исследовательский раздел

1.1 Цель разработки

Необходимо разработать автоматический сбор текстовых и акустических данных с возможностью:

– создание возможности для распознавания пользователей (авторизация и регистрация);

– систематическое хранение акустических и текстовых данных в папках (автоматическое создание папки для каждого пользователя, распределение по дате, папка с названием файла, акустическое и текстовые файлы);

– каждый пользователь должен иметь возможность прослушивать акустические данные и редактировать свои текстовые (создание, прослушивание/редактирование и удаление этих данных);

– создание ролей для пользователей (пользователь с ролью администратора должен иметь возможность управлять всеми данными всех пользователей, управлять и редактировать);

С помощью данного проекта, пользователи могут создавать данные в систематизированном порядке, очень быстро и без необходимости транспортировки данных. Это упрощает жизнь не только пользователей, но и программистов, которые будут использовать эти данные.

1.2 Термины и сокращения

Таблица 1 - Термины, сокращения, и их определения

Сокращение или термин	Определение
ПО	Программное обеспечение.
Visual Studio	Интегрированная среда для разработки различных ПО.
C#	(C - sharp) Объектно-ориентированный язык программирование, созданный компанией Microsoft.
MVC	Фреймворк Asp.Net Core

Продолжение таблицы 1

HTML	(сокр. от англ. Hypertext Markup Language) Язык гипертекстовой разметки, является стандартным языком разметки для документов, предназначенных для отображения ввеб-браузере.
CSS	(сокр. от англ. Cascading Style Sheets)Каскадные таблицы стилей — это языктаблиц стилей, используемый для описания представления документа, написанного на языке разметки, например HTML.
JavaScript	Мультипарадигменный язык программирование, поддерживает такие стили как объектно-ориентированный, императивный и функциональный.
AJAX	(сокр. от англ. Asynchronous JavaScript and XML) подход к фоновому обмену данными браузера с сервером.
UI	(сокр. от англ. User interface) Пользовательский интерфейс.
HTTP	(сокр. от англ. Hypertext TransferProtocol) Протокол передачи гипертекста — это прикладной протокол для распределенных, совместных, гипермедиа информационных систем.
SQL	(сокр. от англ. Structured Query Language) Язык программирования для создания, редактирования и управления данными в реляционной базе данных.
Фреймворк	Коллекция библиотек с готовыми решениями.

1.3 Технологический раздел

1.3.1 Фрэймворк “Asp.Net Core MVC”

Кросс-платформенная “ASP.NET Core” предназначена для создания различных веб-приложений, от небольших до крупных веб-порталов. Она является opensource-фреймворком и весь исходной код можно найти на сайте “GitHub”. Она является модульной т.е. можно загружать отдельно компоненты, которые нужны для реализаций проекта через пакетный менеджер “NuGet”. В проекте будут использованы такие компоненты как:

– “GleamTech.FileUltimate”;
– “Microsoft.AspNet.WebApi.Core”;
– “Microsoft.AspNet.Mvc.Core”;
– “Microsoft.EntityFrameWorkCore.SqlServer”;
– “System.Speech”;
– “Unofficial.System.Speech”;

Так как будет использовано фреймворк “MVC” то паттерн проекта будет состоять из:

– Модель (model);
– Представление (view);
– Контроллер (controller);

Таким образом взаимодействие между компонентами можно будет посмотреть на рисунке 1.

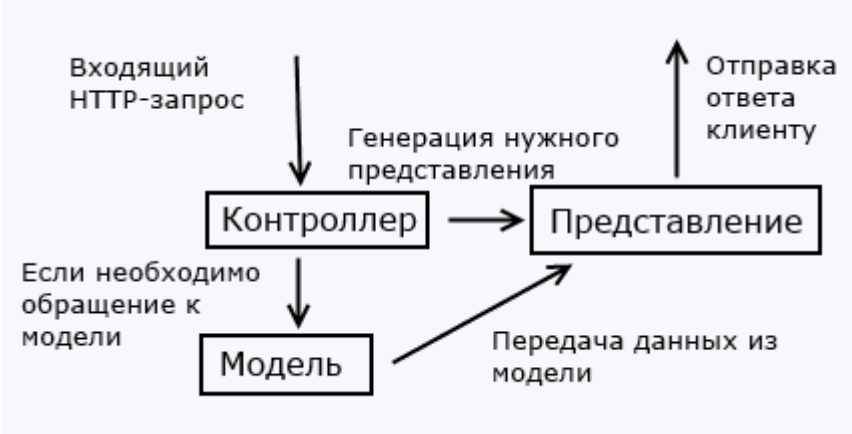


Рисунок 1 – схема “MVC”

Как можно заметить из схемы, от браузера/клиента будет приходить HTTP-запросы (существует различные виды запросов но в данном проекте будут “AJAX” запросы) которые будут обработаны в контроллере. Если нужно будет обработать эти данные как целый объект то будет обращение к модели, а если нет то запрос вернет конечный результат обратно

браузеру/клиенту. Таким образом, каждый компонент будет отвечать только за себя, а это в свою очередь помогает легче разрабатывать, поддерживать и тестировать проект.

1.3.2 Среда разработки

Среда разработки – это выбор вкуса для программистов, если у них есть возможность. Одни могут быть более функциональными чем другие, но в то же время многофункциональные редакторы весят больше и загружают компьютер сильнее. Если пользоваться тем же “Android Studio” на слабом ноутбуке, то возможны различные плачевные последствия такие как:

– Очень медленная работа приложения, компьютер будет сильно шуметь;
– Синий экран из-за нехватки “RAM” памяти и перезарузка;
– Выход компьютера из строя;

Таким образом есть различные критерий, по которым разработчик будет выбирать себе IDE или же простой текстовый редактор на примере “Sublime Text”. “Sublime Text” универсальный текстовый редактор, который может редактировать большое количество файлов, можно одновременно на одном экране открывать несколько файлов что является очень удобным в веб-программированиях, делать различные поиски по файлам и т.д. но в пользу “IDE Visual Studio” можно преподнести такие плюсы как:

– Отладка проекта;
– Компиляция и сборка проекта;
– Интегрированная среда для работы с “SQL”;

Рассмотрев все IDE, выбор был сделан в пользу “Visual Studio”.

2 Проектная часть

2.1 Архитектура проекта

Разрабатываемое приложение будет работать по схеме который нарисован на рисунке 2.

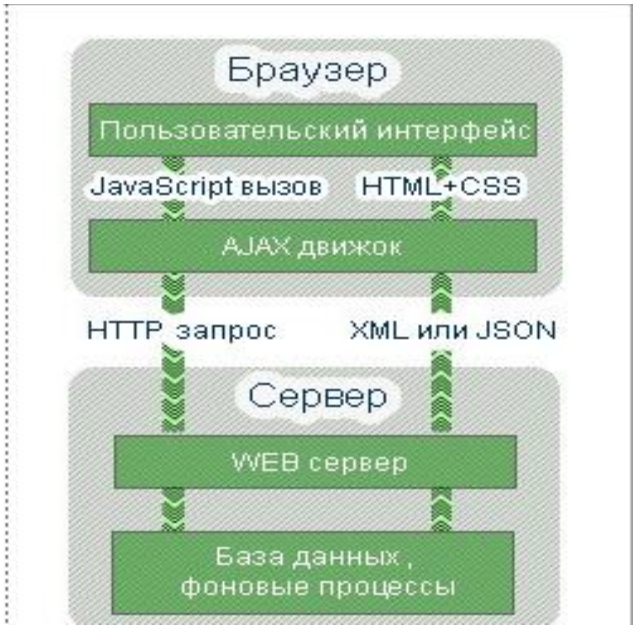


Рисунок 2 – архитектура проекта

Пользовательский интерфейс — это совокупность статичных и динамических элементов. В свою очередь динамические компоненты интерфейса обрабатываются в сервере через “AJAX” запросы и выводятся обратно в “UI”, если только не затрагиваются данные с базы данных. Вся транспортировка данных между слоями, а именно между браузером и сервером будет происходит в формате “JSON” так как парсинг этих файлов в объектах “JavaScript”-а происходит быстрее и к тому же не стоит забывать, что для “AJAX” движков “JSON” формат является самым выгодным форматом нежели чем “XML”. Если бы это был формат “XML”, то пришлось бы делать гораздо больше действий что бы получить эти данные в качестве переменных. После получения “XML” документа пришлось бы использовать “XML DOM” что бы итерировать каждый объект и сохранять их в качестве переменных.

2.2 Описание диаграммы деятельности

Диаграмма деятельности – это визуальное представление

последовательности действий в проекте. Она описывает как проект должен работать, учитывая всевозможные результаты тех или иных действий. Важность этой диаграммы раскрывается при разработке проекта так как в диаграмме указывается исходы всех событии, а если это большой проект, то это единственный друг разработчика наряду с техническим заданием. Еще одним отличительным свойством таких диаграмм является простота в применении так как схема рассматривается как жизненный цикл продукта которое имеет свое начало и конец.

Проект начинается с того, что требует авторизацию, если его нету, то нужно будет сделать регистрацию, после чего пользователь будет перенаправлен в начальную страницу для входа. Авторизация и регистрация – это важная часть проекта так как дальнейшие действия типа записи голоса и редактирование текста будет сохранена в отдельных папках которая будет создана именно для этого пользователя.

После авторизации в проекте будут происходит различные изменения типа создание “cookie” и отдельных папок если это первое посещение после регистрации, но то, что пользователь увидит это домашняя страница в котором будут инструменты для записи голоса, редактирование текста, просмотр всех записей и т.д. но в зависимости от прав пользователя, в этой странице может появится проводник который имеет возможности управления всеми данными всех пользователей.

При создании новой записи, будет проходить проверку на длительность записи, после чего происходит сохранение записи в папке, создается под таким же название текстовый файл в отдельных папках пользователя, и новая запись появится. Можно прослушивать запись, попутно писать в текстовом редакторе. Когда редактирование закончится, идет сохранение текстовых данных. Пользователь может посмотреть все свои записи и нажимая на них редактировать и, если захочется удалить их. При удалении происходит удаление всей папки, в котором хранятся акустические и текстовые данные и после исчезнуть из списка записей пользователей.

Когда этот пользователь является администратором, пользователь может просматривать все записи всех пользователей включая и свои. Появляется возможность управления всеми данными, начиная от простого редактирование до загрузки и выгрузки данных, управление папками и сбором всех записей.

Для того что бы создать такую диаграмму использовалась готовые решения от таких гигантов в “IT” как “Google”. Инструмент называется “draw io” который имеет огромное количество функции при созданиях различных диаграмм. Результат можно увидит на рисунке 3.

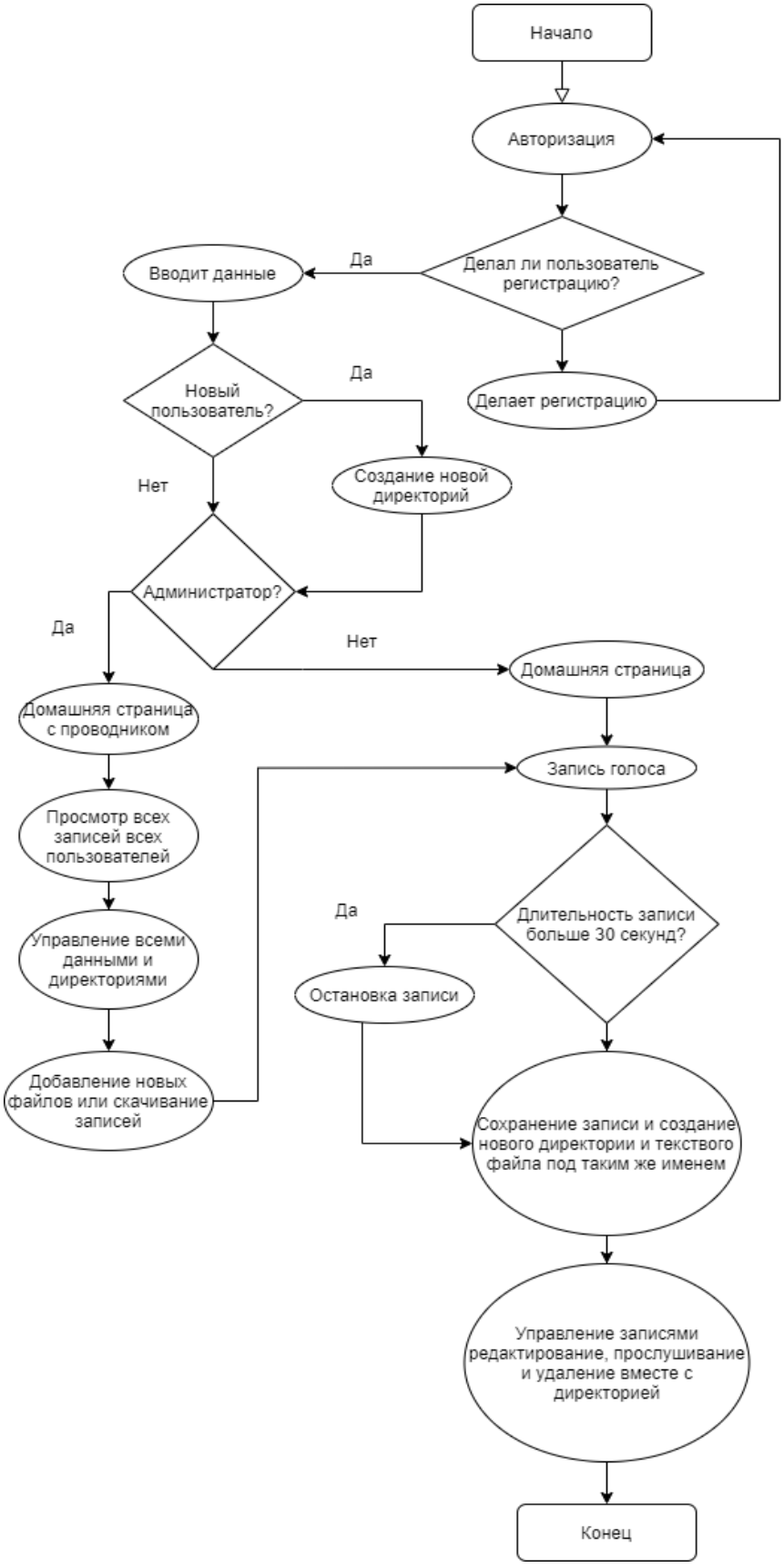


Рисунок 3 – “UML” диаграмма

2.3 Разработка проекта

Часть бэкенда – это неотъемлемая част всех веб-приложений так как вся логика проекта будет находится в нем. Это может быть простая логика которая суммирует двух чисел или сложные процедуры как авторизация и т.д.

Как говорилось раньше, этот проект будет иметь паттерн “MVC” что в свою очередь будет делить бэкенд на 3 разные части. Вся основная логика будет происходит в контроллерах. Модели будут нужны при работе с объектами что бы в дальнейшем, если будет нужно, вывести его через “AJAX” запрос в браузер.

2.3.1 Аутентификация и авторизация

“Asp.NET Core” имеет возможности делать аутентификацию на основе куки. Этот компонент может сериализовать данные пользователя в зашифрованные аутентификационные куки и передавать их пользователю. Когда сервер будет получать от пользователя какие-либо запросы, будет происходит валидация этих куки. Но прежде чем создавать валидацию, нужно для начало создать самих пользователей. Для этого создается новый класс пользователей, а хранить их будем в “MS SQL Server”. Для взаимодействия с базой данных используется “Entity Framework” – отдельный компонент с “Nuget” который помогает работать с данными из базы данных как с объектами в “c#”. После соединения, нужно будет прописать настройки подключения к базе данных который находится в конфигурационном файле под названием “appsettings.json” где дефолтное соединение будет:

“Server=(localdb)\\mssqllocaldb;Database=userdb;Trusted_Connection=True;”

Что бы все это заработало сразу после запуска проекта, нужно определить сценарий работы проекта, нужно для начало добавить сервис аутентификаций внутри которого мы будем использовать куки аутентификацию и указываем путь к файлу где будет происходить аутентификация.

После того как закончили настройку проекта, нужно будет создать контроллер где будем определять как и каким образом будет происходить валидация пользовательских данных. Так как, никаких данных в базе данных не шифруется, при аутентификации пользователя можно посредством “LINQ” запроса поискать пользователя. Если существует такой пользователь, то дальше создаем “Claim” который хранит в себе информацию о текущем пользователе после чего этот набор данных шифруется и добавляется в куки. Если сделать авторизацию то можно будет увидеть в консоли браузера куки которых только что создали на рисунке 4.

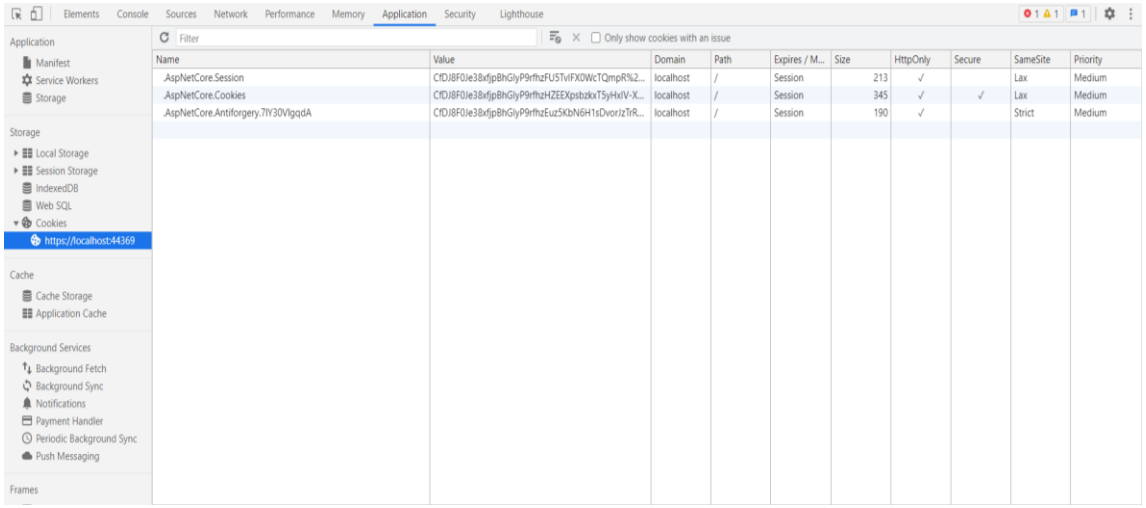


Рисунок 4 – Куки

2.3.2 Запись голоса

В проекте запись голоса осуществляется через “HTML” тэг “audio” которое создается в “JavaScript” файле. Она имеет 2 кнопки “Start” и “Stop”, кнопка проигрывание записи и регулятор громкости как показано на рисунке 5.

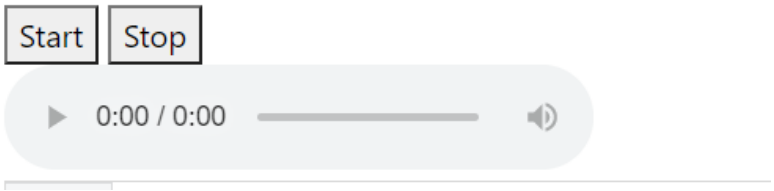


Рисунок 5 – Проигрыватель

– После записи голоса нужно будет его сохранить в удобном формате и месте. Для этого после нажатия кнопки “Stop” или по истечению 30 секунд будет вызван “AJAX” запрос типа “POST” который в качестве параметра будет иметь сам аудиофайл. Этот запрос вызывает метод в контроллере с параметром самого объекта медиафайла. В этом методе нужно будет дать имя этому файлу и выбрать путь по которому будет сохранена запись голоса. Для имени файла используется статичный метод объекта “Guid” который будет генерировать всегда уникальные строковые значения. Дальше путь куда помещается этот файл выбирается таким вот образом:

– имя пользователя;
– текущая дата без года;
– уникальное название файла;

Иными словами создается папка с таким же названием как сам файл.

Причина хранение акустического файла в папке с таким же названием кроется в том что в этой папке создается новый текстовой документ с расширением “.txt“ таким же названием. Это систематизирует хранение всех файлов и создадут удобные условия для тех людей кто потом будет собирать эти данные. Для удобства, после сохранения файла можно будет сразу послушать результат.

2.3.3 Управление текстовыми файлами

Как говорилось раньше, текстовые файлы создаются под такими же именами как голосовые файлы, хранятся в одной папке и создаются после сохранения аудио файлов. Пример на рисунке 6.

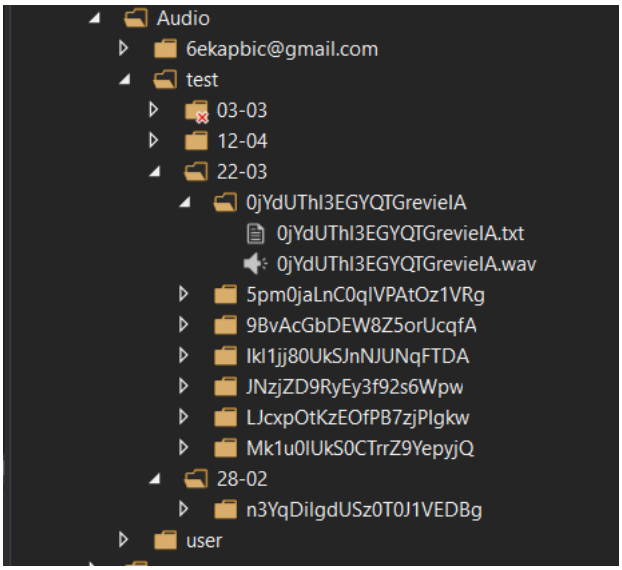


Рисунок 6 – Структура хранения файлов

После создания текстовых документов нужно будет их каким-то образом заполнить данными удобным образом. Для этого существуют разные инструменты. В основном все готовые решения такие как:

– “CKEditor”;
– “Editor”;
– “JSEditor”;
– “Rich text editor”;
– “Quill”;
– “Three.js”;

Попробовав парочку из них, стало очевидно что такие инструменты для редактирования имеют большое количество функционала, некоторые

достигают уровень “Microsoft Words” из-за чего было решено создать собственный редактор для работы именно с “txt” файлами.

Для простой имитации работы с текстовым документом было использовано “HTML” тэг под названием “textarea” который в свою очередь создает поле нужного размера для того что бы пользователь мог что-то написать в нем. Примерно это будет выглядит как показано на рисунке 7. Дальше создаются две кнопки. Один предназначен для того что бы показать все имеющие файлы пользователя а второй для сохранения записей сделанных в текстовом поле.

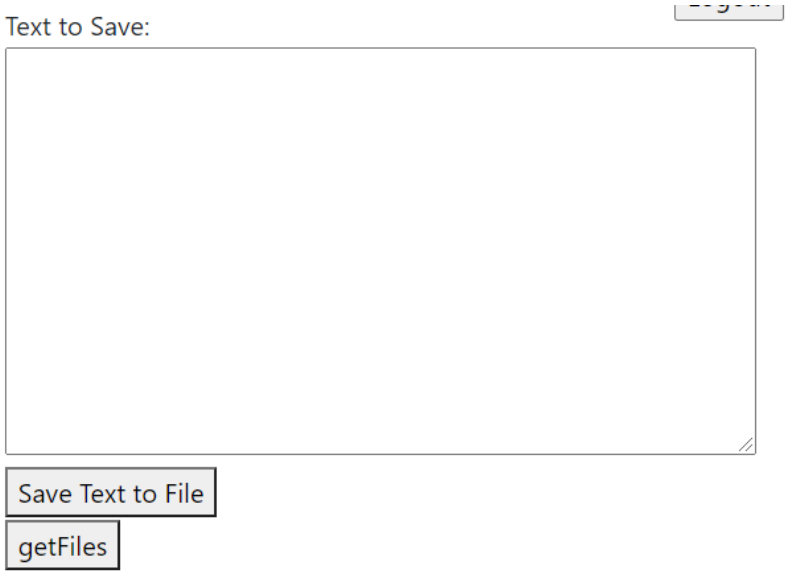


Рисунок 7 – Редактор файла

Пользователь после записи акустических данных, должен будет открыть все свои файлы и затем выбрать среди них тот файл которого только что создали. Затем выбрав этот файл можно будет приступить к написанию текстовых данных а по окончанию нажимается кнопка сохранения.

Две эти кнопки имеют вызов определенной функции “JavaScript” после нажатия, что тот в свою очередь создаст “AJAX” запросы. Для получение всех файлов не нужны никакие параметры, так как единственное что нужно для получение этих файлов это – имя пользователя которое всегда можно получить в самом контроллере через “User.Identity”. Метод контроллера пробежит по всем папкам этого пользователя и получит все txt файлы в виде объекта. Этот объект будет иметь такие свойства как:

– имя файла;
– полный путь;
– папка (имя файла);
– имя пользователя;

Все эти поля класса будут использованы при создании новых элементов в браузере. Они будут использованы в качестве параметров для вызова

следующих функции:

– Редактирование файла;
– Удаление файла;

На рисунке 8, можно заметить что бы удалить или редактировать эти файлы при этом не ошибаясь, нужно будет каким-то образом их друг от друга отличать. Для этого создается универсальная функция которая может принимать в качестве параметра имя файла для того что бы им можно было управлять.

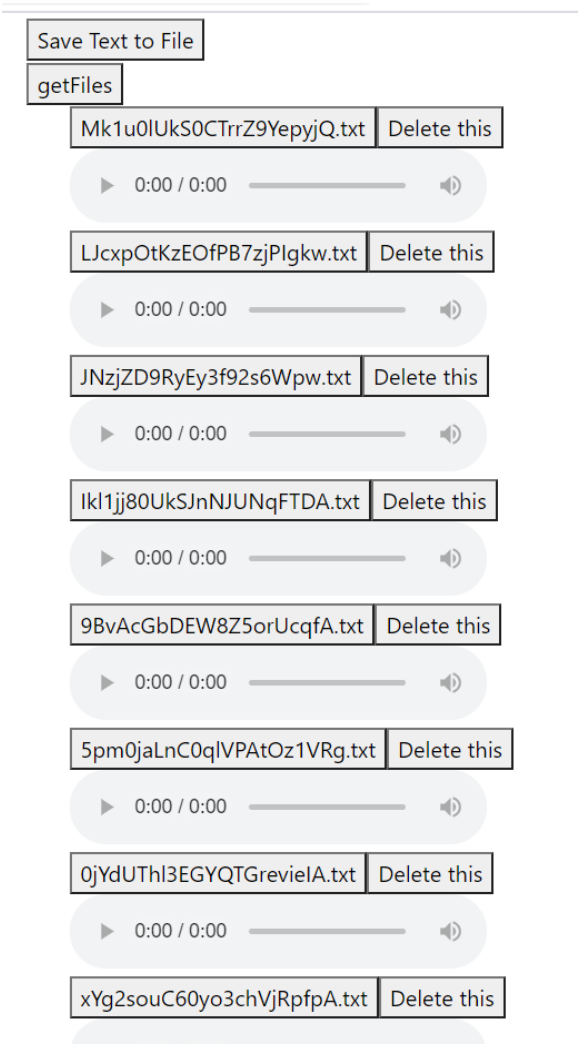


Рисунок 8 – Работа кнопки “getFiles”

После нахождения файла которого нужно будет редактировать, пользователь нажимает на кнопку с названием файла которое находится слева от кнопки “Delete this”. Нажатие на кнопку с названием файла спровоцирует вызов “JavaScript” функции которое получит текст этого файла в “HTML” объект “textarea”. Внутри этого тэга можно будет редактировать, напечатать и сохранить то что нужно. Пример работы этой кнопки можно будет лицезреть в рисунках 9 и 10.

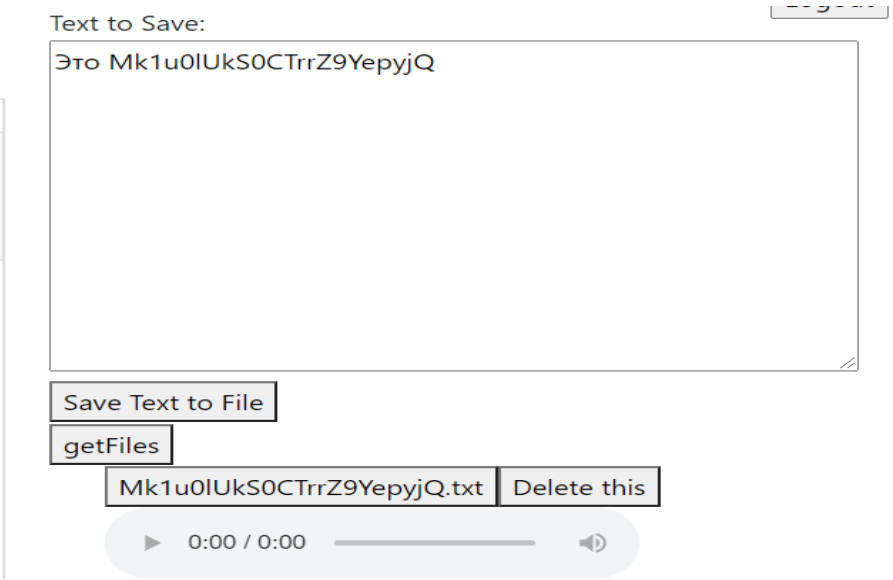


Рисунок 9 – Работа кнопки “Save Text to File”

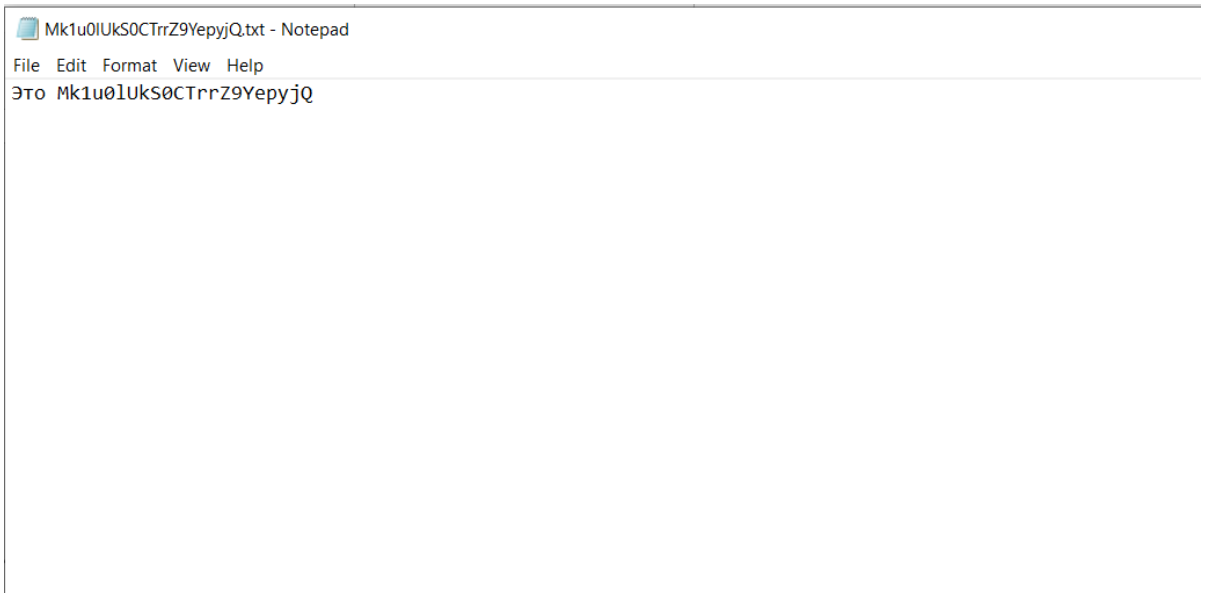


Рисунок 10 – Работа кнопки “Save Text to File”

А если нажать на кнопку “Delete this”, будет вызван “AJAX” запрос с параметром именем файла после чего будет вызван метод в контроллере который будет по данному пользователю искать папку с таким именем, а затем удалит папку и файлы по этому пути. Результат работы можно посмотреть на рисунках 11 и 12.

Рисунок 11 – Работа кнопки “Delete this”

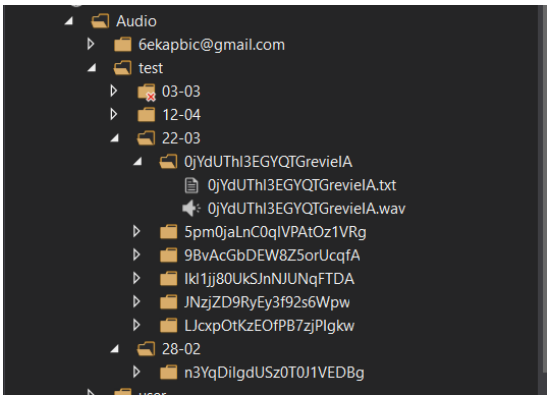


Рисунок 12 – Работа кнопки “Delete this”

Можно заметить что папки и файлов под именем “Mk1u0…” нету.

2.3.4 Роль администратора

Как обычно, роль суперпользователя несет за собой некий дополнительный функционал который никому кроме самого администратора не доступен. Случай в этом проекте тоже не исключение.

Для создания такого функционала нужно будет добавить отдельную колонку в таблице пользователей и соответственно добавить новое поле объекту пользователей в самом проекте. Это простой флаг который заполнен либо нулем либо единицей но несмотря на это в самом проекте от этого флага будет зависеть доступ к дополнительному функционалу. Этот функционал подразумевает собой проводника который очень удобен для ориентировки среди сохраненных файлов.

Было большое количество готовых решений но не все подходили под критерий. Один из важнейших критерий было неограниченное права для управление всеми директориями и файлами внутри них, возможность скачивать или обратно выгружать данные и т.д. из-за чего был выбран решение от компаний “GleamTech”.

Для использование этого решения нужно скачать дополнительный компонент из пакетного менеджера которое называется – “GleamTech.FileUltimate”. После установки, нужно будет добавить его в сервисах и зарегистрировать среди запросов проекта. Далее можно будет использовать его уже в представлениях указав размер, корневую папку и права доступа но в проекте происходит валидация пользователя на роль администратора. Пример можно увидеть на рисунке 13.

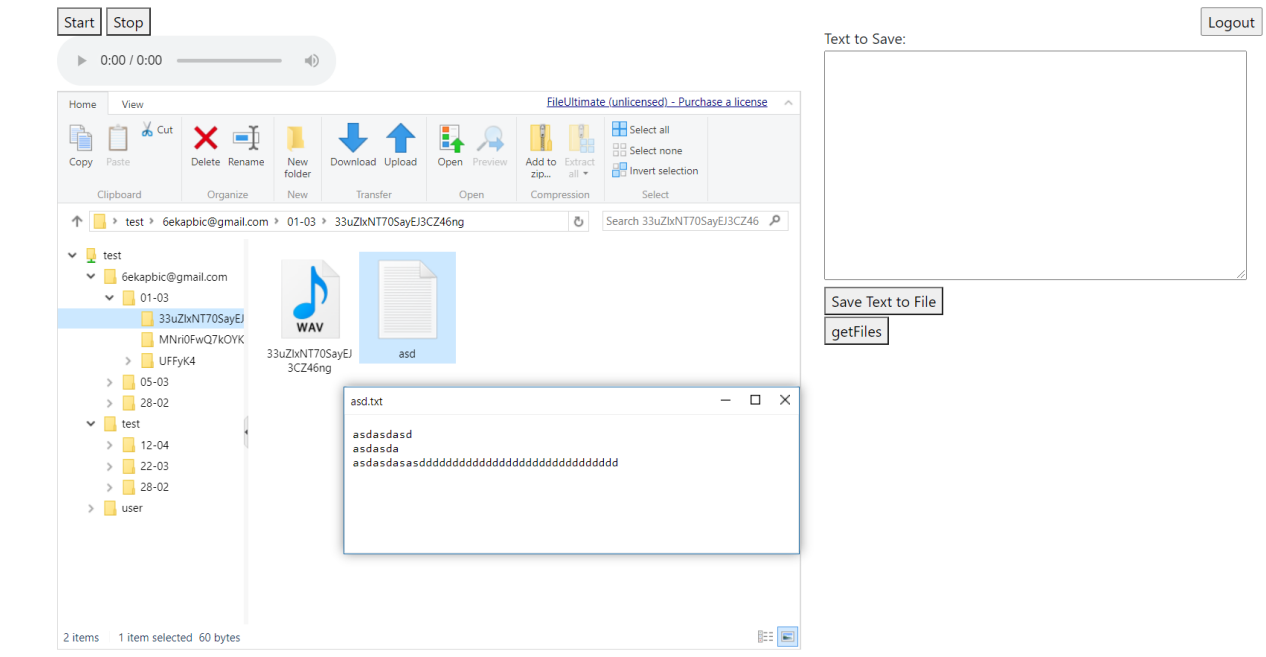


Рисунок 13 – Пример работы проводника

Как можно заметить, можно управлять со всеми папками и файлами, удалять или редактировать, прослушивать или можно даже и скачивать или обратно загружать туда собственные файлы. Есть дополнительные возможности как добавление расширение файлов, смена размер иконок, сортировать все файлы, делать поиски среди всех папок и быстро найти тот самый файл которого нужно найти, можно добавлять в архив или обратно вывести из архива. Делая выводы, можно отметить что это универсальное решение когда нужно реализовать что-то подходящее.

ЗАКЛЮЧЕНИЕ

В рамках данной дипломной работы было построено веб-приложение которое предназначено для автоматизации сбора и хранение акустических и текстовых данных для создания корпуса казахского языка. Цели программы:

– создание возможности для распознавания пользователей (авторизация и регистрация);

– систематическое хранение акустических и текстовых данных в папках (автоматическое создание папки для каждого пользователя, распределение по дате, папка с названием файла, акустическое и текстовые файлы);

– каждый пользователь должен иметь возможность прослушивать акустические данные и редактировать свои текстовые (создание, прослушивание/редактирование и удаление этих данных);

– создание ролей для пользователей (пользователь с ролью администратора должен иметь возможность управлять всеми данными всех пользователей, управлять и редактировать);

Проект был выполнен на платформе “Asp.Net Core” на языке “c#” а в качестве хранилища был использован “MS SQL Server”. Все вышеперечисленные цели были достигнуты.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Introduction to c# concepts // Электронная версия на сайте https:// docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/tutorials/
2. Полное руководство по языку // Электронная версия насайте https:// metanit.com/sharp/tutorial/
3. Деплой сайта // Электронная версия на сайте https:// habr.com/ru/post/413495/
4. BackEnd разработка // Электронная версия на сайте https:// qna.habr.com/q/697334
5. Проблемы BackEnd // Электронная версия на сайте https:// habr.com/ru/post/219579/
6. Linux деплой // Электронная версия на сайте https:// habr.com/ru/post/276013/
7. NginX пакеты // Электронная версия на сайте http://nginx.org/ru/linux_packages.html
8. Установка NginX // Электронная версия на сайте http://nginx.org/ru/docs/install.html
9. Chrome Web Server // Электронная версия на сайте https:// networkinterview.com/web-server-for-chrome/
10. AJAX // Электронная версия на сайте https:// ru.wikipedia.org/wiki/AJAX
11. AJAX with ASP.NET Core // Электронная версия на сайте https://www.thereformedprogrammer.net/asp-net-core-razor-pages-how-to-implement-ajax-requests/
12. Video and Audio APIs // Электронная версия на сайте https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Video_and_audio_APIs
13. GleamTech // Электронная версия на сайте https:// www.gleamtech.com/company
14. Document Viewer and Converter // Электронная версия на сайте https:// www.gleamtech.com/documentultimate
15. SQL Server on Linux // Электронная версия на сайте https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-overview?view=sql-server-ver15

Приложение А

(обязательное)

Техническое задание

Техническое задание на разработку веб-приложения автоматического сбора акустических данных для создания акустического корпуса казахского языка

Настоящее техническое задание распространяется на разработку веб-приложения автоматического сбора акустических данных для создания акустического корпуса казахского языка, отвечающего сбор и хранение акустических и текстовых данных. Предполагается что данное приложение будет использовано сотрудниками корпуса. Данный инструмент позволит оптимизировать процесс работы со сбором, хранением и управлением данных.

Основания для разработки

Проект разрабатывается на основании устного распоряжения научного руководителя для создания корпуса казахского языка.

Назначение

Разрабатываемый проект предназначен для усовершенствование процесса сбора, хранение и управление данных акустических и текстовых данных.

Требования к функциональным характеристикам

Приложение автоматического сбора данных должен обеспечить возможность выполнения следующих функций:

– стандартной работы с фрагментами текста документа (копирование, удаление, перемещение, корректировка текста в режимах замены или вставки символов и т.д.);

– создание возможности для распознавания пользователей (авторизация и регистрация);

– систематическое хранение акустических и текстовых данных в папках (автоматическое создание папки для каждого пользователя, распределение по дате, папка с названием файла, акустическое и текстовые файлы);

– каждый пользователь должен иметь возможность прослушивать акустические данные и редактировать свои текстовые (создание, прослушивание/редактирование и удаление этих данных);

– создание ролей для пользователей (пользователь с ролью администратора должен иметь возможность управлять всеми данными всех пользователей, управлять и редактировать);

Требования к надежности

Предусмотреть длительность акустических данных. При длительности больше 30 секунд останавливать запись. Предусмотреть ограничение добавлений других символов при редактирование текстового файла.

Приложение Б

Текст программы

1. *HomeController.cs*

```
[Microsoft.AspNetCore.Authorization.Authorize]
[System.Web.Http.HttpGet]
public IActionResult Index()
{
    ViewBag.currentUser = currentUser;
    ViewBag.isAdmin = usr.IsAdmin;
    return View();
}
[System.Web.Http.HttpPost]
public IActionResult GetRecording([FromUri] MediaFile parameters = null )
{
    Guid guid = Guid.NewGuid();
    string uniqKey = Convert.ToBase64String(guid.ToByteArray());
    uniqKey = uniqKey.Replace("=", "");
    uniqKey = uniqKey.Replace("+", "");
    fileName = uniqKey;
    string fileNameWitPath = Path.Combine(_host.WebRootPath, "audio",
currentUser, DateTime.Now.ToString("dd-MM"), fileName, fileName + ".wav");
    string fileNameWitPath1 = Path.Combine(_host.WebRootPath, "audio",
currentUser, DateTime.Now.ToString("dd-MM"), fileName, fileName + ".txt");
    string dirPath = Path.Combine(_host.WebRootPath, "audio", currentUser,
DateTime.Now.ToString("dd-MM"), fileName);
    try
    {
        if (!Directory.Exists(dirPath))
        {
            DirectoryInfo di = Directory.CreateDirectory(dirPath);
        }
        using (FileStream fs = new FileStream(fileNameWitPath,
FileMode.Create))
        {
            using (BinaryWriter bw = new BinaryWriter(fs))
            {
                byte[] data = Convert.FromBase64String(parameters.chunks);
                bw.Write(data);
```

```
                bw.Close();
            }
            fs.Close();
        }
        using (FileStream fs = new FileStream(fileNameWitPath1,
FileMode.Create))
        {
            byte[] info = new UTF8Encoding(true).GetBytes(string.Empty);
            fs.Write(info, 0, info.Length);
        }
        return null;
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
}

[System.Web.Http.HttpPost]
public IActionResult SaveToTxtFile([FromUri] TxtModel parameters = null)
{
    try
    {
        using (FileStream fs = new FileStream(Path.Combine(parameters.path,
parameters.fileName), FileMode.Create))
        {
            byte[] info = new UTF8Encoding(true).GetBytes(parameters.data);
            fs.Write(info, 0, info.Length);
        }
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
    return null;
}

public  IActionResult Logout()
{

HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
    return RedirectToAction("Login", "Account");
```

```
    }
     [System.Web.Http.HttpPost]
     public List<TxtModel> GetFiles()
     {
        string[] folders =
System.IO.Directory.GetDirectories(Path.Combine(_host.WebRootPath, "audio",
currentUser), "*", System.IO.SearchOption.AllDirectories);
        List<TxtModel> files = new List<TxtModel>();
        for (int i = 0; i< folders.Length; i++)
        {
           string[] fileNameWitPath = Directory.GetFiles(folders[i]);
           foreach (string filePath in fileNameWitPath)
           {
              string nameOfFile = Path.GetFileName(filePath);
              string fileExtensionMult = ".wav";
              string fileExtensionTxt = ".txt";
              int indexOfDot = nameOfFile.IndexOf(fileExtensionMult);
              if (indexOfDot < 0)
                 indexOfDot = nameOfFile.IndexOf(fileExtensionTxt);
              string folderOfMult = "";
              if (indexOfDot > 0)
              {
                 nameOfFile = nameOfFile.Remove(indexOfDot,
fileExtensionMult.Length);
                 int indexOfFolder = folders[i].IndexOf(nameOfFile) - 6;
                 folderOfMult = folders[i].Remove(0, indexOfFolder);
                 int indexOfSlashes = folderOfMult.IndexOf(@"\");
                 folderOfMult = folderOfMult.Remove(indexOfSlashes,
folderOfMult.Count() - 5);
              }
              files.Add(new TxtModel {
                 fileName = Path.GetFileName(filePath),
                 path = folders[i],
                 folder = folderOfMult,
                 createdUser = currentUser
              });
           }
        }

        for (int i = 0; i < files.Count; i++)
        {
           if (files[i].fileName.Contains(".txt"))
```

```
            {
                using (FileStream fstream =
System.IO.File.OpenRead(Path.Combine(files[i].path, files[i].fileName)))
                {
                    byte[] array = new byte[fstream.Length];
                    fstream.Read(array, 0, array.Length);
                    files[i].data = System.Text.Encoding.Default.GetString(array);
                }
            }
        }
        return files;
    }

    [System.Web.Http.HttpGet]
    public void DeleteFolder(string fileName)
    {
        string path = "";
        string[] folders =
System.IO.Directory.GetDirectories(Path.Combine(_host.WebRootPath, "audio",
currentUser), "*", System.IO.SearchOption.AllDirectories);
        for (int i = 0; i<folders.Length; i++)
        {
            if (folders[i].Contains(fileName))
            {
                path = folders[i];
            }
        }
        DirectoryInfo directoryInfo = new DirectoryInfo(path);
        foreach (FileInfo file in directoryInfo.GetFiles())
        {
            file.Delete();
        }
        Directory.Delete(path);
    }
  }
}
```

2. site.js

```
$(function () {
  let log = console.log.bind(console),
    ul = $('#ul')[0],
```

```
    start = $('#start')[0],
    stop = $('#stop')[0],
    stream,
    recorder,
    counter = 1,
    chunks,
    media;
media = {
    tag: 'audio',
    type: 'audio/ogg',
    ext: '.ogg',
    gUM: { audio: true }
}
navigator.mediaDevices.getUserMedia(media.gUM).then(_stream => {
    stream = _stream;
    recorder = new MediaRecorder(stream);
    recorder.ondataavailable = e => {
        chunks.push(e.data);
        var superBuffer = new Blob(chunks, { type: 'audio/ogg' });
        if (recorder.state == 'inactive') makeLink();
        var reader = new window.FileReader();
        reader.readAsDataURL(superBuffer);

        reader.onloadend = function () {
            base64 = reader.result;
            base64 = base64.split(',')[1];

            $.ajax({
                url: 'Home/GetRecording',
                type: 'POST',
                data: {
                    audioname: "hello",
                    chunks: base64
                }
            });
        }
    };
    log('got media successfully');
}).catch(log);
start.onclick = e => {
    start.disabled = true;
    stop.removeAttribute('disabled');
```

```
        chunks = [];
        recorder.start();
        var timeLeft = 30;

        var timerId = setInterval(countdown, 1000);

        function countdown() {
            if (timeLeft == -1) {
                clearTimeout(timerId);
                stop.disabled = true;
                recorder.stop();
                start.removeAttribute('disabled');
            }
            timeLeft -= 1;
        }
    }
    stop.onclick = e => {
        stop.disabled = true;
        recorder.stop();
        start.removeAttribute('disabled');
    }
    function makeLink() {
        let blob = new Blob(chunks, { type: media.type })
            , url = URL.createObjectURL(blob)
            , div = document.createElement('div')
            , mt = document.createElement(media.tag)
            , hf = document.createElement('a')
            ;
        mt.controls = true;
        mt.src = url;
        hf.href = url;
        hf.download = `${counter++}${media.ext}`;
        hf.innerHTML = `donwload ${hf.download}`;
        div.appendChild(mt);
        ul.appendChild(div);
    }
});
function saveTextAsFile() {
    var txtData = document.getElementById("inputTextToSave").value;
    var txtName = clickedFile;
    var txtPath = clickedFilePath;
    $.ajax({
```

```
url: 'Home/SaveToTxtFile',
        type: 'POST',
        dataType: "json",
        data: {
            fileName: txtName,
            data: txtData,
            path: txtPath
        }
    });
}
var files;
function getFiles() {
    $("ol").empty();
    $.ajax({
        type: "POST",
        url: 'Home/GetFiles',
        data: {  },
        dataType: "json",
        success: function (msg) {
            if (files != msg) {
                files = msg;
                let liFirst = document.createElement('li');
                for (var i = 0; i < msg.length; i++) {
                    if (msg[i].fileName.includes('.txt')) {
                        $("ol").each(function () {
                            $(this).prepend('<div><button id=' + "" + 'getFiles' + "" + " " + '
onclick = ' + "'getData(' + """ + msg[i].fileName + """ + ')' + "" + '>' + msg[i].fileName
+ '</button ><button id=' + "" + 'deleteFiles' + "" + " " + ' onclick = ' +
"'DeleteFolder(' + """ + msg[i].fileName.slice(0, msg[i].fileName.length - 4) + """ + ')'
+ "" + '>' + 'Delete this' + '</button ><audio controls><source src = ' + "" +
'http://127.0.0.1:8887/' + msg[i].createdUser + '/' + msg[i].folder + '/' +
msg[i].fileName.slice(0, msg[i].fileName.length - 4) + '/' +
msg[i].fileName.replace('.txt', '.wav') + "" + 'type = ' + "" + 'audio/wav' + "'>' + '></div
>');
                            i++;
                        });
                    }
                }
            }
        },
        error: function (req, status, error) {
            alert(error);
```

```
        }
    });
}
var clickedFile;
var clickedFilePath;
function getData(nameOfFile) {
    var fileData;
    for (var i = 0; i < files.length; i++) {
        if (files[i].fileName == nameOfFile) {
            fileData = files[i];
        }
    }
    clickedFile = fileData.fileName;
    clickedFilePath = fileData.path;
    document.getElementById("inputTextToSave").value = fileData.data;
}
function DeleteFolder(file) {
    $.ajax({
        type: "GET",
        url: 'Home/DeleteFolder',
        data: { fileName: file },
        error: function (req, status, error) {
            alert(error);
        }
    });
}
```

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий

Орынбасаров Бекарыс Талапұлы

5В070400 – Вычислительная техника и программное обеспечение

ОТЗЫВ НАУЧНОГО РУКОВОДИТЕЛЯ

к дипломному проекту

Тема: «Разработка веб-приложения автоматического сбора акустических данных для создания акустического корпуса казахского языка».

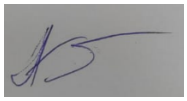
Дипломная работа написана студентом Орынбасаров Бекарыс Талапұлы на актуальную в настоящий момент тему «Разработка веб-приложения автоматического сбора акустических данных для создания акустического корпуса казахского языка».

Данная тема является важной и актуальной по ряду причин. В настоящее время, акустический корпус национального языка сталкивается с проблемами нехватки данных, так как язык постоянно меняется и для этого нужны актуальные данные. Такие данные используются во многих приложениях связанные с набором или определением текста. Поставленные цели и задачи полностью соответствуют теме исследования. Дипломная работа написана на основе современных технологий и статей ученых, авторитетных в данной области.

Дипломная работа состоит из трех глав, введения, заключения и списка использованной литературы. После каждой главы содержатся четкие выводы. Оформление диплома соответствует стандартам. Во введении содержится обоснование актуальности работы, цели, задачи, использованные технологий, а также положения, выносимые на защиту.

В целом студент полно и точно раскрыл тему дипломной работы. Недостатков обнаружено не было. Работа допускается к защите.

Научный руководитель: маг. техн. наук, сениор-лектор



Бекарыстанқызы.А

"08"__06_____2021г.

StrikePlagiarism.com

SATBAYEV UNIVERSITY

Дата отчета 5/24/2021

Дата редактирования ---

Метаданные

Название

2021 Орынбасаров Бекарыс Талапұлы.docx

Автор

Орынбасаров Бекарыс Талапұлы

Научный руководитель

Жибек Алибиева

Подразделение

ИКиИТ

Список возможных попыток манипуляций с текстом

В этом разделе вы найдете информацию, касающуюся манипуляций в тексте, с целью изменить результаты проверки. Для того, кто оценивает работу на бумажном носителе или в электронном формате, манипуляции могут быть невидимы (может быть также целенаправленное вписывание ошибок). Следует оценить, являются ли изменения преднамеренными или нет.

Замена букв		0
Интервалы	A→	17
Микропробелы		22
Белые знаки		0
Парафразы (SmartMarks)	a	6

Объем найденных подобий

Обратите внимание!Высокие значения коэффициентов не означают плагиат. Отчет должен быть проанализирован экспертом.

25
Длина фразы для коэффициента подобия 2

4755
Количество слов

40522
Количество символов

Протокол анализа Отчета подобия Научным руководителем

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

Автор: Орынбасаров Бекарыс

Название: БАК 2021 Орынбасаров Бекарыс Талапұлы - нормконтроль.docx

Координатор: Бекарыстанқызы А.

Коэффициент подобия 1:3.05

Коэффициент

подобия 2:0.53

Замена букв:0

Интерва

лы:17

Микропр

обелы:22

Белые

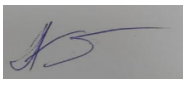
знаки: 0

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:

. . . . . .

. . . . . 06.08.2021 . . . . .  . . . . . .

Дата *Подпись Научного руководителя*