

Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Модели данных и системы управления базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Сервис создания и прохождения опросов

БГУИР КП 1–40 04 01 ПЗ

Студент: гр. 753503

Сидоренко К. А.

Руководитель:

Удовин И. А.

Минск 2020

Оглавление

Введение.....	3
1. Основные требования	4
1.1. Цель разработки	4
1.2. Основные функции	4
1.3. Требования к организации системы.....	4
1.4. Требования к внешнему интерфейсу	4
1.5. Выбор языка и средств программирования и среды разработки..	5
2. Используемые технологии	6
2.1. Язык программирования C#	6
2.2. ASP.NET Core.....	7
2.3. SQL	9
2.4. MySQL.....	9
2.5. MySQL Workbench.....	10
2.6. Figma.....	10
2.7. Git.....	11
2.8. HTML	12
2.9. CSS.....	12
2.10. JavaScript.....	13
2.11. Firebase	13
2.12. Visual Studio.....	14
3. Моделирование предметной области.....	16
3.1. Описание backend части	16
3.2. Описание front-end части	17
3.3. Описание используемой базы данных	18
4. Реализация приложения	21
Заключение	26
Список используемых источников.....	27
ПРИЛОЖЕНИЕ А. Исходный код приложения	28

Введение

Интернет стал неотъемлемой частью жизнедеятельности людей. С помощью различных интернет-сервисов каждый человек может пообщаться с друзьями, посмотреть интересные видео, узнать новости, заказать еду, купить практически любые вещи и многое другое. Все это стало возможно благодаря развитию стремительному развитию и распространению информатики и технологий программирования.

Каждый интернет-сервис так или иначе работает с данными. Возникает потребность в правильной организации и хранении этих данных. В разное существовали и были популярными множества подходов к менеджменту хранимой информации, но венцом этого развития стали *базы данных*. База данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных. Для управления созданием и использованием баз данных существуют специальные программные комплексы, которые называются *системой управления базами данных*. СУБД прежде всего обеспечивает безопасность, надёжность хранения и целостность данных. Благодаря СУБД манипулирование данными при разработке программного обеспечения значительно упрощается.

В качестве демонстрации возможностей и результата изучения современных баз данных и СУБД было принято разработать небольшой интернет-сервис, предназначенный для создания и прохождения опросов.

1. Основные требования

1.1. Цель разработки

Во многих сферах деятельности часто возникает потребность в сборе анонимных данных. Например, это может быть получение обратной связи о продукте либо о сервисе, сбор отзывов на приложение, сбор контактных данных для участия в мероприятии и т.д. Так было решено разработать интернет-сервис, который бы организовывал описанный процесс сбора данных.

1.2. Основные функции

Интернет сервис должен выполнять следующие функции:

- Регистрация и авторизация пользователей
- Создание форм для опроса
- Прохождение опросов
- Получение информации от прохождения опроса

1.3. Требования к организации системы

В качестве архитектурного решения была выбрана организация приложения в виде веб-сервиса (Web API). Работа с данными будет производится на сервере backend части приложения. Коммуникация с frontend частью приложения организована посредством протокола HTTP, формат передаваемых данных – JSON.

В структуре Web API выделяются контроллеры и модели данных. Контроллеры отвечают за интерпретацию действий пользователя. Модели описывают данные, используемые в приложении.

Также к проекту системы должна быть подключена СУБД для организации и управления данными. Должны быть написаны соответствующие классы для работы с базой данных.

1.4. Требования к внешнему интерфейсу

Макет интерфейса пользователя будет разработан с помощью Figma. Сам интерфейс приложения будет разработан с использованием Javascript, HTML, CSS. Интерфейс сервиса должен быть интуитивно понятным, не вычурным, сохранять простоту и лаконичность, быть репрезентативным.

1.5. Выбор языка и средств программирования и среды разработки

В качестве основного языка программирования выбран C# версии 8.0, а в качестве фреймворка разработки – ASP.NET Core версии 3.1.

C# – объектно-ориентированный язык программирования, который создан, активно развивается и продвигается компанией Microsoft.

Платформа ASP.NET Core представляет технологию от компании Microsoft, предназначенную для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов.

2. Используемые технологии

2.1. Язык программирования C#

Язык программирования C# Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников – языков C++, Delphi, Модула, Smalltalk и, в особенности, Java – C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на VB.NET, J# и др.

Существует несколько реализаций C#:

- Реализация C# в виде компилятора csc.exe включена в состав .NET Framework (включая .NET Micro Framework, .NET Compact Framework и его реализации под Silverlight и Windows Phone 7).
- В составе проекта Rotor (Shared Source Common Language Infrastructure) компании Microsoft.

- Проект Mono включает в себя реализацию C# с открытым исходным кодом.
- Проект DotGNU также включает компилятор C# с открытым кодом.
- DotNetAnywhere – ориентированная на встраиваемые системы реализация CLR, поддерживает практически всю спецификацию C# 2.0.

2.2. ASP.NET Core

С одной стороны, ASP.NET Core является продолжением развития платформы ASP.NET. Но, с другой стороны, это не просто очередной релиз. Выход ASP.NET Core фактически означает революцию всей платформы, ее качественное изменение.

Разработка над платформой началась еще в 2014 году. Тогда платформа условно называлась ASP.NET vNext. В июне 2016 года вышел первый релиз платформы. А в декабре 2019 года вышла версия ASP.NET Core 3.1, которая, собственно, и будет охвачена в текущем руководстве.

ASP.NET Core теперь полностью является opensource-фреймворком. Все исходные файлы фреймворка доступны на GitHub.

ASP.NET Core может работать поверх кроссплатформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS, Linux. И таким образом, с помощью ASP.NET Core мы можем создавать кроссплатформенные приложения. И хотя Windows в качестве среды для разработки и развертывания приложения до сих пор превалирует, но теперь уже мы не ограничены только этой операционной системой. То есть мы можем запускать веб-приложения не только на ОС Windows, но и на Linux и Mac OS. А для развертывания веб-приложения можно использовать традиционный IIS, либо кроссплатформенный веб-сервер Kestrel.

Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget. Кроме того, в отличие от предыдущих версий платформы нет необходимости использовать библиотеку System.Web.dll.

ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages. В предыдущих версии платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности. Сейчас же они объединены в одну программную модель ASP.NET Core MVC. А Web Forms полностью ушли в прошлое.

Кроме объединения вышеупомянутых технологий в одну модель в MVC был добавлен ряд дополнительных функций.

Одной из таких функций являются тэг-хелперы (tag helper), которые позволяют более органично соединять синтаксис HTML с кодом C#.

ASP.NET Core характеризуется расширяемостью. Фреймворк построен из набора относительно независимых компонентов. И мы можем либо использовать встроенную реализацию этих компонентов, либо расширить их с помощью механизма наследования, либо вовсе создать и применять свои компоненты со своим функционалом.

Также было упрощено управление зависимостями и конфигурирование проекта. Фреймворк теперь имеет свой легковесный контейнер для внедрения зависимостей, и больше нет необходимости применять сторонние контейнеры, такие как Autofac, Ninject. Хотя при желании их также можно продолжать использовать.

В качестве инструментария разработки мы можем использовать последние выпуски Visual Studio, начиная с версии Visual Studio 2015. Кроме того, мы можем создавать приложения в среде Visual Studio Code, которая является кроссплатформенной и может работать как на Windows, так и на Mac OS X и Linux.

Для обработки запросов теперь используется новый конвейер HTTP, который основан на компонентах Katana и спецификации OWIN. А его модульность позволяет легко добавить свои собственные компоненты.

Если суммировать, то можно выделить следующие ключевые отличия ASP.NET Core от предыдущих версий ASP.NET:

- Новый легковесный и модульный конвейер HTTP-запросов
- Возможность разворачивать приложение как на IIS, так и в рамках своего собственного процесса
- Использование платформы .NET Core и ее функциональности
- Распространение пакетов платформы через NuGet
- Интегрированная поддержка для создания и использования пакетов NuGet
- Единый стек веб-разработки, сочетающий Web UI и Web API
- Конфигурация для упрощенного использования в облаке
- Встроенная поддержка для внедрения зависимостей
- Расширяемость
- Кроссплатформенность: возможность разработки и разворачивания приложений ASP.NET на Windows, Mac и Linux
- Развитие как open source, открытость к изменениям

Эти и другие особенности и возможности стали основой для новой модели программирования.

2.3. SQL

SQL – это декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL считается языком программирования, в общем случае (без ряда современных расширений) не является тьюринг-полным, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений.

Со временем SQL усложнился – обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) – и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

2.4. MySQL

MySQL свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle. Изначальным разработчиком данной СУБД была шведская компания MySQL AB. В 1995 году она выпустила первый релиз MySQL. В 2008 году компания MySQL AB была куплена компанией Sun Microsystems, а в 2010 году уже компания Oracle поглотила Sun и тем самым приобрела права на торговую марку MySQL. Поэтому MySQL на сегодняшний день развивается под эгидой Oracle.

Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

2.5. MySQL Workbench

MySQL Workbench – инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL.

Возможности программы:

- Позволяет наглядно представить модель базы данных в графическом виде.
- Наглядный и функциональный механизм установки связей между таблицами, в том числе «многие ко многим» с созданием таблицы связей.
- Reverse Engineering — восстановление структуры таблиц из уже существующей на сервере БД (связи восстанавливаются в InnoDB, при использовании MyISAM – связи необходимо устанавливать вручную).
- Удобный редактор SQL запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы.
- Возможность редактирования данных в таблице в визуальном режиме.

При разработке данного приложения была использована редакция MySQL Workbench под названием «Community Edition».

2.6. Figma

Figma – онлайн-сервис для разработки интерфейсов и прототипирования с возможностью организации совместной работы в режиме реального времени. Сервис имеет широкие возможности для интеграции с корпоративным мессенджером Slack и инструментом для высокоуровневого

прототипирования Framer. Позиционируется создателями как основной конкурент программным продуктам компании Adobe.

Сервис доступен по подписке, предусмотрен бесплатный тарифный план для одного пользователя. Ключевой особенностью Figma является её облачность, у сервиса нет оффлайн-версии. За счет этого также достигается принцип кроссплатформенности, который не могут гарантировать ближайшие конкуренты — Sketch и Adobe XD.

Figma подходит как для создания простых прототипов и дизайн-систем, так и сложных проектов (мобильные приложения, порталы). В 2018 году платформа стала одним из самых быстро развивающихся инструментов для разработчиков и дизайнеров.

Устойчивая модель развития Figma, упростившая сотрудничество во всем процессе создания цифровых продуктов для дизайнеров, разработчиков, менеджеров и маркетологов, позволила в начале 2018 года привлечь дополнительные 25 миллионов долларов инвестиций от партнеров.

В 2019 году создатели редактора привлекли 40 миллионов долларов инвестиций от венчурного фонда Sequoia Capital при оценке самой компании в 400 миллионов долларов. Решение об инвестициях было принято на основе факта, что около половины портфельных компаний фонда используют редактор в работе. К началу 2019 года Figma вышла на 1 миллион зарегистрированных пользователей, став серьезным конкурентом для традиционных графических редакторов и средств прототипирования.

2.7. Git

Git – распределенная система контроля версий для отслеживания изменений в исходном коде во время разработки программного обеспечения. Он предназначен для координации работы между программистами, но его можно использовать для отслеживания изменений в любом наборе файлов. Его цели включают скорость, целостность данных, и поддержку распределенных, нелинейных рабочих процессов.

Git был создан Линусом Торвальдсом в 2005 году для разработки ядра Linux, а другие разработчики ядра внесли свой вклад в его первоначальную разработку. Текущий сопровождающий с 2005 года – Хунио Хамано. Как и в большинстве других распределенных систем контроля версий, и в отличие от большинства клиент-серверных систем, каждый каталог Git на каждом компьютере является полноценным хранилищем с полной историей и возможностями полного отслеживания версий, независимо от доступа к сети или центрального сервера. Git – это бесплатное программное обеспечение с

открытым исходным кодом, распространяемое на условиях Стандартной общественной лицензии GNU версии 2.

2.8. HTML

HTML (HyperText Markup Language – «язык гипертекстовой разметки») – самый базовый строительный блок Веба. Он определяет содержание и структуру веб-контента. Другие технологии, помимо HTML, обычно используются для описания внешнего вида/представления (CSS) или функциональности/поведения (JavaScript) веб-страницы.

Под гипертекстом ("hypertext") понимаются ссылки, которые соединяют веб-страницы друг с другом либо в пределах одного веб-сайта, либо между веб-сайтами. Ссылки являются фундаментальным аспектом Веба. Загружая контент в Интернет и связывая его со страницами, созданными другими людьми, вы становитесь активным участником Всемирной паутины.

HTML использует разметку ("markup") для отображения текста, изображений и другого контента в веб-браузере. HTML-разметка включает в себя специальные "элементы", такие как `<head>`, `<title>`, `<body>`, `<header>`, `<footer>`, `<article>`, `<section>`, `<p>`, `<div>`, ``, ``, `<aside>`, `<audio>`, `<canvas>`, `<datalist>`, `<details>`, `<embed>`, `<nav>`, `<output>`, `<progress>`, `<video>` и многие другие.

HTML-элемент выделяется из прочего текста в документе с помощью "тегов", которые состоят из имени элемента окруженного "<" и ">". Имя элемента внутри тега не чувствительно к регистру. То есть, оно может быть написано в верхнем или нижнем регистре, или смешано. Например, тег `<title>` может быть записан как `<Title>`, `<TITLE>`, или любым другим способом.

2.9. CSS

Каскадные таблицы стилей (CSS) – это язык таблиц стилей, используемый для описания представления документа, написанного на HTML или XML (включая диалекты XML, такие как SVG, MathML или XHTML). CSS описывает, как элементы должны отображаться на экране, на бумаге, в речи или на других носителях.

CSS является одним из основных языков открытого Интернета и стандартизирован для всех веб-браузеров в соответствии со спецификацией W3C. Ранее разработка различных частей спецификации CSS велась синхронно, что позволяло создавать версии последней рекомендации.

Возможно, вы слышали о CSS1, CSS2.1, CSS3. Тем не менее, CSS4 никогда не стал официальной версией.

В CSS3 область применения спецификации значительно возросла, и прогресс в различных модулях CSS стал настолько сильно отличаться, что стало более эффективно разрабатывать и выпускать рекомендации отдельно для каждого модуля. Вместо того чтобы создавать версии спецификации CSS, W3C теперь периодически делает снимок последнего стабильного состояния спецификации CSS.

2.10. JavaScript

JavaScript (часто просто JS) – это легковесный, интерпретируемый или JIT-компилируемый, объектно-ориентированный язык с функциями первого класса. Наиболее широкое применение находит как язык сценариев веб-страниц, но также используется и в других программных продуктах, например, Node.js или Apache CouchDB. JavaScript это прототипно-ориентированный, мультипарадигменный язык с динамической типизацией, который поддерживает объектно-ориентированный, императивный и декларативный (например, функциональное программирование) стили программирования. Подробнее о JavaScript.

Стандартом языка JavaScript является ECMAScript. По состоянию на 2012, все современные браузеры полностью поддерживают ECMAScript 5.1. Старые версии браузеров поддерживают по крайней мере - ECMAScript 3. 17 июня 2015 года состоялся выпуск шестой версии ECMAScript. Эта версия официально называется ECMAScript 2015, которую чаще всего называют ECMAScript 2015 или просто ES2015. С недавнего времени стандарты ECMAScript выпускаются ежегодно. Эта документация относится к последней версии черновика, которой является ECMAScript 2018.

Не следует путать JavaScript с языком программирования Java. И "Java", и "JavaScript" являются торговыми марками или зарегистрированными торговыми марками Oracle в США и других странах. Однако, у обоих языков различный синтаксис, семантика и применение.

2.11. Firebase

Firebase – это платформа для разработки мобильных приложений и веб-приложений, разработанная Firebase, Inc. в 2011 году, а затем приобретенная Google в 2014 году. По состоянию на март 2020 года платформа Firebase насчитывает 19 продуктов, которые используются более чем 1,5 миллионами приложений.

Первым продуктом Firebase был Realtime Firebase, который синхронизирует данные приложений на iOS, Android и веб-устройствах и сохраняет их в облаке Firebase. Продукт помогает разработчикам программного обеспечения в создании приложений для совместной работы в реальном времени.

В 2014 году Firebase выпустила два продукта. Firebase Hosting и Firebase Authentication. Компания стала позиционироваться как мобильный бэкэнд как сервис.

В октябре 2014 года Firebase была приобретена Google. Год спустя, в октябре 2015 года, Google приобрела Divshot, платформу веб-хостинга HTML5, чтобы объединить ее с командой Firebase.

В мае 2016 года на ежегодной конференции разработчиков Google I/O компания Firebase представила Firebase Analytics и объявила, что расширяет свои сервисы, чтобы стать единой платформой back-end-as-a-service (BaaS) для мобильных разработчиков. Firebase теперь интегрируется с различными другими службами Google, включая Google Cloud Platform, AdMob и Google Ads, чтобы предлагать более широкие продукты для разработчиков. Google Cloud Messaging, служба Google для отправки push-уведомлений на устройства Android, была заменена продуктом Firebase, Firebase Cloud Messaging, который добавил функциональность для доставки push-уведомлений как на iOS, так и на веб-устройства. В январе 2017 года Google приобрела Fabric и Crashlytics у Twitter, чтобы добавить эти сервисы в Firebase.

В октябре 2017 года Firebase запустил Cloud Firestore, базу данных документов в режиме реального времени, как преемник оригинальной базы данных Firebase Realtime.

2.12. Visual Studio

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода.

Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

3. Моделирование предметной области

3.1. Описание backend части

На рисунке ниже изображено дерево проекта backend части.

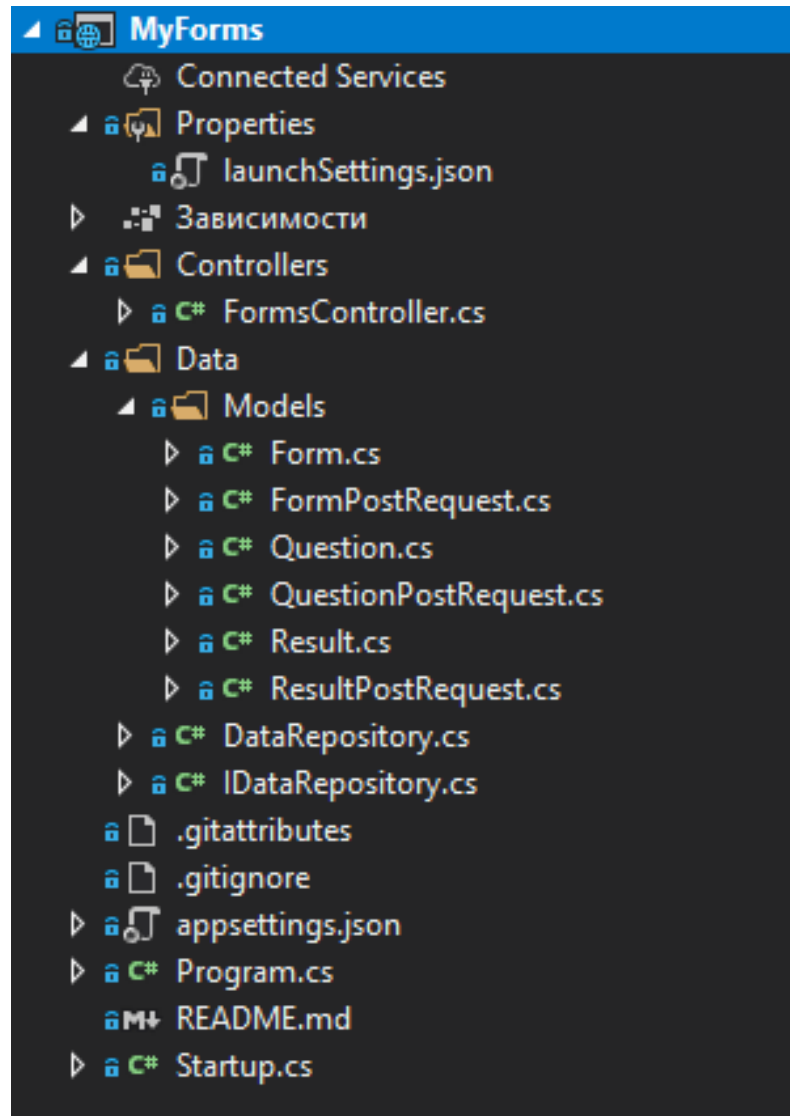


Рисунок 3.1. Структура проекта backend части

В папке Properties содержится файл launchSettings.json, в котором описываются профили запуска приложения и настройки сервера IIS.

В папке Controllers находится единственный контроллер FormsController.cs, который отвечает за создание и отправку форм, сохранение результатов прохождения форм.

В папке Data лежат интерфейс IRepository.cs и класс DataRepository.cs, в которых содержатся методы для взаимодействия с базой данных.

В папке Models находятся модели для связи с сущностями базы данных, а также модели для биндинга с данными с frontend части.

Файл appsettings.json содержит различные настройки, такие как строки подключения к БД и настройки логгирования.

Файл Startup.cs предназначен для настройки маршрутизации, Dependency Injection, подключения сервисов.

Файл Program.cs содержит инструкции по запуску веб-приложения.

3.2. Описание front-end части

На рисунке ниже изображено дерево проекта frontend части.

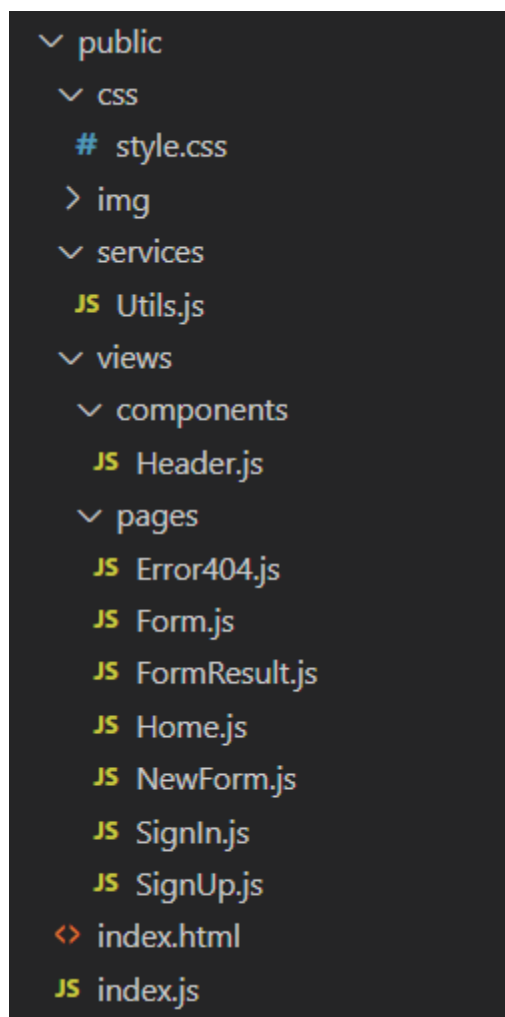


Рисунок 3.2. Структура проекта frontend части

Файл index.html содержит шаблон всех страниц сайта, именно в эту страницу динамически подгружается содержимое. Кроме того, в файле подключаются сервисы Firebase.

В index.js находится маршрутизатор и всех страниц приложения, этот файл является точкой входа всего приложения. В нем происходит динамический рендер всех страниц и компонентов.

В папке css расположен всего один файл – style.css. Он отвечает за внешний вид всего сайта, в нем прописаны стили всех элементов, анимации кнопок и других интерактивных элементов.

В папке img находятся используемые в приложении изображения.

В папке services тоже расположен всего один файл – Utils.js. В нем написаны две функции. Одна разбивает строку запроса на три элемента (resource, id, verb), другая отвечает за вызов так называемого Snackbar – небольшого возникающего на несколько секунд уведомления о чем-либо внизу страницы.

В папке views две папки – components и pages. В папке components один файл – Header.js. Он отвечает за рендер шапки сайта. В папке pages находятся файлы всех страниц приложения.

3.3. Описание используемой базы данных

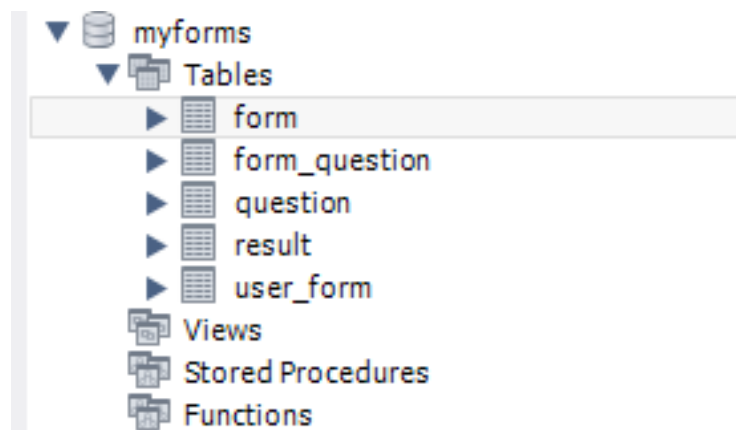


Рисунок 3.3. Список таблиц базы данных myforms

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 name	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 description	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рисунок 3.4. Структура таблицы form

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 fid	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 qid	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рисунок 3.5. Структура таблицы form_question





Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 name	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 type	ENUM('single choice',...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 options	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рисунок 3.6. Структура таблицы question




Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 fid	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 answers	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рисунок 3.7. Структура таблицы result




Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 uid	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 fid	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рисунок 3.8. Структура таблицы user_form

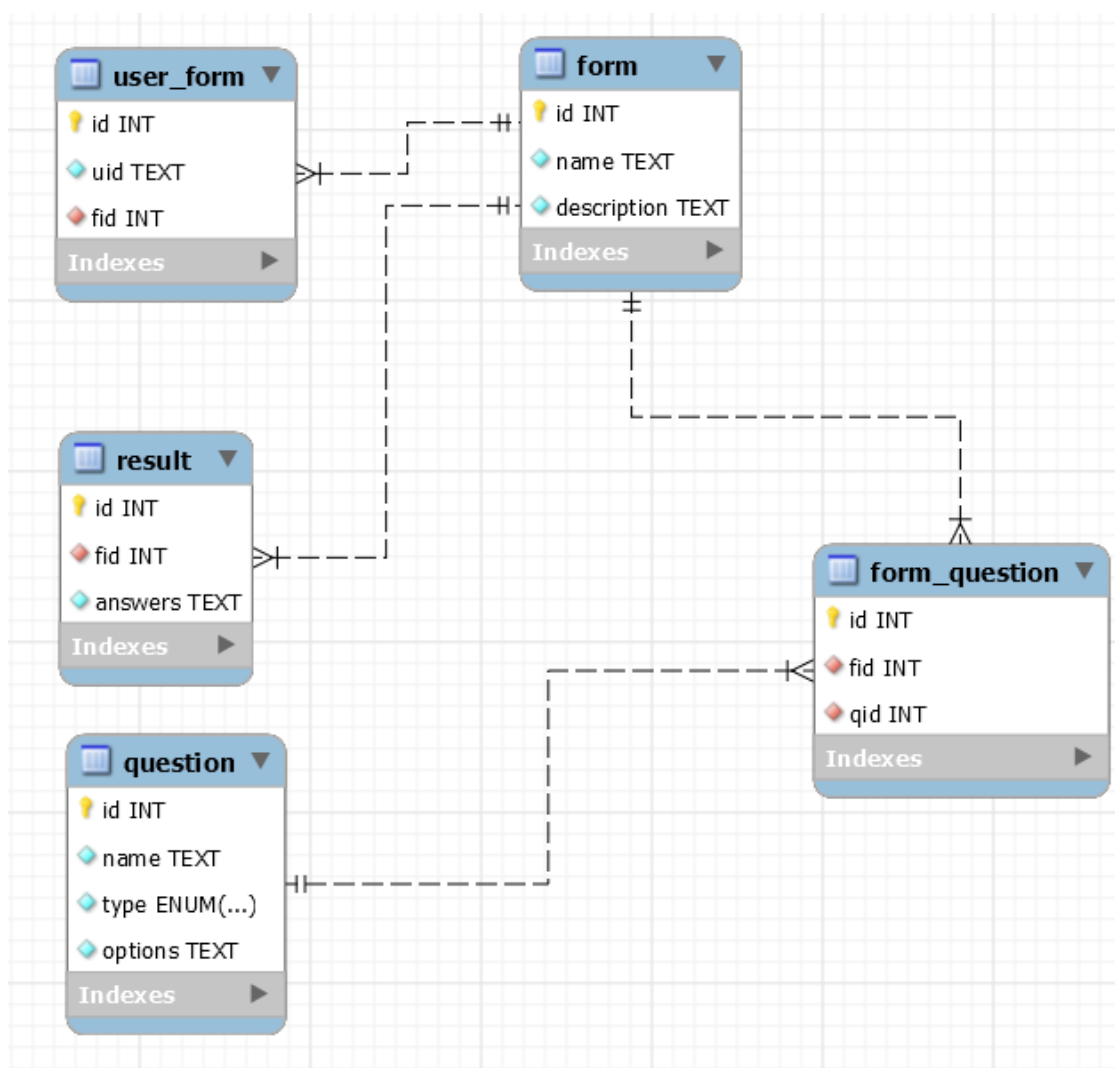


Рисунок 3.9. Диаграмма базы данных

4. Реализация приложения

При открытии веб-приложения пользователя встречает главная страница сайта:

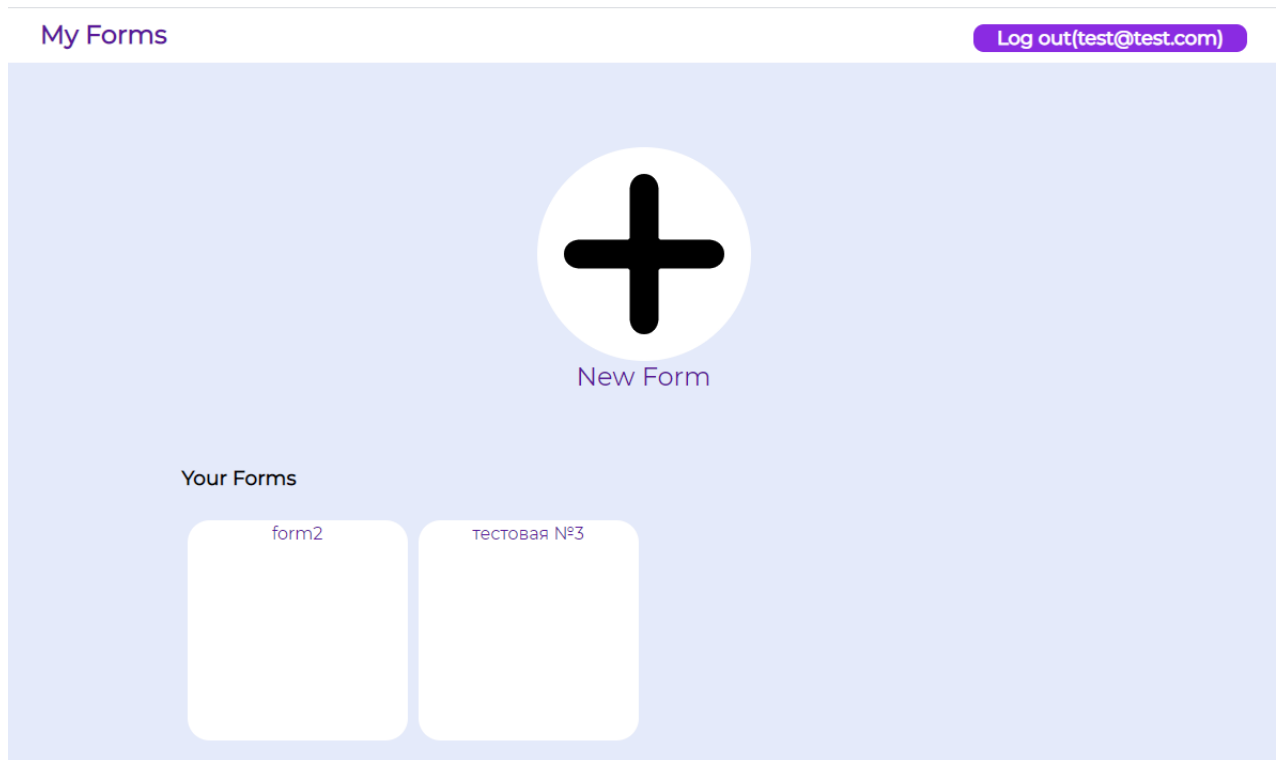


Рисунок 4.1. Главная страница приложения

Здесь можно увидеть формы, которые создал пользователь, а также шапку сайта, общую для всех страниц. В шапке слева находится ссылка на главную страницу сайта, справа – на страницу авторизации пользователя

The image shows a web interface for user authentication. At the top left, there is a link labeled "My Forms". At the top right, there is a link labeled "Sign In". The main content area has a light blue background. In the center, there is a white rounded rectangle containing the "Sign in" title. Below the title are two input fields: "E-mail" and "Password". Below these fields is a purple button labeled "Login". At the bottom of the white rectangle, there is a link that says "Don't have account? Sign up!".

Рисунок 4.2. Страница авторизации пользователя

Если пользователь не имеет аккаунт, он не может создать форму. Чтобы зарегистрироваться, нужно перейти по ссылке ниже кнопки «Login». Для регистрации необходимо ввести свой адрес электронной почты, а также придумать пароль. Аутентификация пользователей реализована с помощью сервиса Firebase Authentication.

My Forms

Sign In

Sign up

Sign up

Рисунок 4.3. Страница регистрации пользователя

После создания аккаунта пользователь может создать форму. Для этого нужно нажать на кнопку с изображением «плюса», после чего откроется страница с редактором форм:

My Forms

Log out(test@test.com)

Собрание друзей

Давно не виделись, есть желание встретиться

Где? text answer

место, адрес

Когда? single choice

19:00 X

19:30 X

Add option

Save

Рисунок 4.4. Редактор форм

Чтобы сохранить форму, нужно нажать на кнопку «Save». После чего форма станет доступна на главной странице приложения. Автор формы может отправить ссылку на форму. Кликнув по этой ссылке человек увидит страницу прохождения формы.

My Forms Log out(test@test.com)

Собрание друзей

Давно не виделись, есть желание встретиться

Где?

место, адрес

Когда?

☐ 19:00

☐ 19:30

Send

Рисунок 4.5. Страница прохождения формы

После заполнения формы следует нажать на кнопку «Send», чтобы отправить форму. Автор формы может посмотреть результаты опроса, кликнув по квадрату с формой на главной странице приложения.

Собрание друзей (1 answers)

Давно не виделись, есть желание встретиться

Share link to form: [/#/form/5](#)

Где?

1. кафе "Мара"

Когда?

☒ 19:00 (1 times)

☐ 19:30 (0 times)

Рисунок 4.6. Страница результатов формы

Заключение

В ходе курсовой работы было рассмотрено множество различных технологий разработки программного обеспечения. Были получены необходимые навыки по работе, настройке, управлению базами данными. Также были изучены различные подходы и шаблоны по проектированию, прототипированию и разработке программных продуктов.

Результатом работы является готовый функциональный интернет-сервис для создания и прохождения опросов. Приложение соответствует всем заявленным ранее требованиям.

Список используемых источников

1. Firebase Docs: <https://firebase.google.com/docs?authuser=0>
2. HTML Academy: <https://htmlacademy.ru/study>
3. Современный учебник JavaScript: <https://learn.javascript.ru/>
4. Документация по MySQL: <https://dev.mysql.com/doc/>

ПРИЛОЖЕНИЕ А. Исходный код приложения

Backend часть

1) FormsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using MyForms.Data;
using MyForms.Data.Models;
using System.Text.Json;
using System.Text.Json.Serialization;

namespace MyForms.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class FormsController : ControllerBase
    {
        private readonly IDataRepository _dataRepository;

        public FormsController(IDataRepository dataRepository)
        {
            _dataRepository = dataRepository;
        }

        [HttpGet("{formId}")]
        public ActionResult<Form> GetForm(int formId)
        {
            Form form = _dataRepository.GetForm(formId);
            return form;
        }

        [HttpGet("user-forms/{uid}")]
        public IEnumerable<Form> GetUserForms(string uid)
        {
            return _dataRepository.GetUserForms(uid);
        }

        [HttpGet("{formId}/results")]
        public IEnumerable<Result> GetFormResults(int formId)
        {
            return _dataRepository.GetFormResults(formId);
        }

        [HttpPost]
        public ActionResult<Form> PostForm([FromBody] FormPostRequest form)
```

```

    {
        var savedForm = _dataRepository.PostForm(form);
        return CreatedAtAction("PostForm", savedForm);
    }

    [HttpPost("results")]
    public ActionResult<Result> PostResult([FromBody] ResultPostRequest result)
    {
        var savedResult = _dataRepository.PostResult(result);
        return CreatedAtAction("PostResult", savedResult);
    }
}
}

```

2) IRepository.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using MyForms.Data.Models;

namespace MyForms.Data
{
    public interface IRepository
    {
        Form GetForm(int formId);
        IEnumerable<Form> GetUserForms(string userId);
        Question[] GetFormQuestions(int formId);
        IEnumerable<Result> GetFormResults(int formId);
        string GetFormUserId(int formId);

        Form PostForm(FormPostRequest formPostRequest);
        Question PostQuestion(QuestionPostRequest questionPostRequest);
        Result PostResult(ResultPostRequest resultPostRequest);
    }
}

```

3) DataRepository.cs

```

using Microsoft.Extensions.Configuration;
using MySql.Data.MySqlClient;
using Dapper;
using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Threading.Tasks;
using MyForms.Data;
using MyForms.Data.Models;
using System.Text.Json.Serialization;
using System.Text.Json;

namespace MyForms.Data
{
    public class DataRepository : IDataRepository
    {
        private readonly string _connectionString;

        public DataRepository(IConfiguration configuration)
        {
            _connectionString = configuration["ConnectionStrings:MySQLConnection"];
        }

        public Form GetForm(int formId)
        {
            using (var connection = new MySqlConnection(_connectionString))
            {
                connection.Open();

                var form = connection.QueryFirstOrDefault<Form>(
                    @"SELECT id, name AS fname, description FROM form
                    WHERE id=@FormId;", new { FormId = formId }
                );

                form.Questions = GetFormQuestions(formId);
                form.Uid = GetFormUserId(formId);
                return form;
            }
        }

        public IEnumerable<Form> GetUserForms(string userId)
        {
            using (var connection = new MySqlConnection(_connectionString))
            {
                connection.Open();

                var forms = connection.Query<Form>(
                    @"SELECT form.id, user_form.uid as uid, form.name AS fname, form.descripti
on FROM form
                    JOIN user_form ON (form.id = user_form.fid)
                    WHERE user_form.uid=@UserId;", new { UserId = userId }
                );

                return forms;
            }
        }
    }
}

```

```

    }
}

public Question[] GetFormQuestions(int formId)
{
    using (var connection = new MySqlConnection(_connectionString))
    {
        connection.Open();

        var questions = connection.Query<Question>(
            @"SELECT question.id, question.name AS qname, question.type, question.opti
ons
            FROM form JOIN form_question ON(form_question.fid = form.id)
            JOIN question ON(form_question.qid = question.id)
            WHERE form.id=@FormId;", new { FormId = formId }
        );

        foreach (var question in questions)
        {
            question.Options = JsonSerializer.Deserialize<string[]>(question.Options);
        }
        return questions.ToArray();
    }
}

public string GetFormUserId(int formId)
{
    using (var connection = new MySqlConnection(_connectionString))
    {
        connection.Open();

        string uid = connection.QueryFirstOrDefault<string>(
            @"SELECT uid FROM user_form
            WHERE fid=@FormId;", new { FormId = formId }
        );

        return uid;
    }
}

public IEnumerable<Result> GetFormResults(int formId)
{
    using (var connection = new MySqlConnection(_connectionString))
    {
        connection.Open();

        var results = connection.Query<Result>(
            @"SELECT id, fid, answers FROM result
            WHERE fid=@FormId;", new { FormId = formId }
        );
    }
}

```

```

        return results;
    }
}

public Form PostForm(FormPostRequest formPostRequest)
{
    using (var connection = new MySqlConnection(_connectionString))
    {
        connection.Open();

        connection.Execute(
            @"INSERT INTO form(name, description) VALUES (@Name, @Description);",
            new
            {
                formPostRequest.Name,
                formPostRequest.Description
            }
        );

        var form = connection.QueryFirstOrDefault<Form>(
            @"SELECT id, name, description FROM form ORDER BY id DESC LIMIT 1;"
        );

        connection.Execute(
            @"INSERT INTO user_form(uid, fid) VALUES(@Uid, @Fid);",
            new
            {
                Uid = formPostRequest.UserId,
                Fid = form.Id
            }
        );

        foreach (var questionPostRequest in formPostRequest.Questions)
        {
            var question = PostQuestion(questionPostRequest);

            connection.Execute(
                @"INSERT INTO form_question(fid, qid) VALUES(@Fid, @Qid);",
                new
                {
                    Fid = form.Id,
                    Qid = question.Id
                }
            );
        }

        return form;
    }
}

```



```

public Question PostQuestion(QuestionPostRequest questionPostRequest)
{
    using (var connection = new MySqlConnection(_connectionString))
    {
        connection.Open();

        connection.Execute(
            @"INSERT INTO question(name, type, options) VALUES (@QName, @Type, @Options)",
            new {
                questionPostRequest.QName,
                questionPostRequest.Type,
                Options = JsonSerializer.Serialize(questionPostRequest.Options)
            }
        );

        return connection.QueryFirstOrDefault<Question>(
            @"SELECT id, name, type, options FROM question ORDER BY id DESC LIMIT 1;"
        );
    }
}

public Result PostResult(ResultPostRequest resultPostRequest)
{
    using (var connection = new MySqlConnection(_connectionString))
    {
        connection.Open();

        connection.Execute(
            @"INSERT INTO result(fid, answers) VALUES (@Fid, @Answers);",
            new {
                resultPostRequest.Fid,
                resultPostRequest.Answers
            }
        );

        return connection.QueryFirstOrDefault<Result>(
            @"SELECT id, fid, answers FROM result ORDER BY id DESC LIMIT 1;"
        );
    }
}
}
}
}

```

4) Form.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MyForms.Data.Models
{
    public class Form
    {
        public int Id { get; set; }
        public string Fname { get; set; }
        public string Description { get; set; }
        public string Uid { get; set; }

        public Question[] Questions { get; set; }
    }
}

```

5) FormPostRequest.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MyForms.Data.Models
{
    public class FormPostRequest
    {
        public string Name { get; set; }
        public string Description { get; set; }
        public string UserId { get; set; }
        public QuestionPostRequest[] Questions { get; set; }
    }
}

```

6) Question.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MyForms.Data.Models
{
    public class Question
    {

```

```

        public int Id { get; set; }
        public string Qname { get; set; }
        public string Type { get; set; }
        public dynamic Options { get; set; }
    }
}

```

7) QuestionPostRequest.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MyForms.Data.Models
{
    public class QuestionPostRequest
    {
        public string QName { get; set; }
        public string Type { get; set; }
        public string[] Options { get; set; }
    }
}

```

8) Result.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MyForms.Data.Models
{
    public class Result
    {
        public int Id { get; set; }
        public int Fid { get; set; }
        public string Answers { get; set; }
    }
}

```

9) ResultPostRequest.cs

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Threading.Tasks;

namespace MyForms.Data.Models
{
    public class ResultPostRequest
    {
        public int Fid { get; set; }
        public string Answers { get; set; }
    }
}

```

10) Program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace MyForms
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                })
        }
    }
}

```

11) Startup.cs

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using MyForms.Data;

namespace MyForms
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddScoped<IDataRepository, DataRepository>();

            services.AddControllers();

            services.AddCors(options =>
            {
                options.AddPolicy("CorsPolicy", builder =>
                    builder.AllowAnyMethod()
                        .AllowAnyHeader()
                        .WithOrigins("http://localhost:5000")
                        .AllowCredentials()
                );
            });

            // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
            public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
            {
                if (env.IsDevelopment())
                {
                    app.UseDeveloperExceptionPage();
                }

                app.UseCors("CorsPolicy");
            }
        }
    }
}

```

```

        app.UseHttpsRedirection();

        app.UseRouting();

        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
}

```

Frontend часть

1) index.html

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="css/style.css">
    <script type="module" src="/index.js"></script>
</head>

<body>
    <header id="header-content">

    </header>
    <main id="main-content">

    </main>

    <script defer src="/__/firebase/7.14.5/firebase-app.js"></script>
    <!-- include only the Firebase features as you need -->
    <script defer src="/__/firebase/7.14.5/firebase-auth.js"></script>
    <script defer src="/__/firebase/7.14.5/firebase-database.js"></script>
    <!-- initialize the SDK after all desired features are loaded -->
    <script defer src="/__/firebase/init.js"></script>
</body>

</html>

```

2) index.js

```

"use strict";

import Home from './views/pages/Home.js';
import SignIn from './views/pages/SignIn.js';
import SignUp from './views/pages/SignUp.js';
import NewForm from './views/pages/NewForm.js';
import Form from './views/pages/Form.js';
import FormResult from './views/pages/FormResult.js';
import Error404 from './views/pages/Error404.js';

import Header from './views/components/Header.js';

import Utils from './services/Utils.js';

const routes = {
  '/': Home,
  '/signin': SignIn,
  '/signup': SignUp,
  '/newform': NewForm,
  '/form/:id': Form,
  '/formresult/:id': FormResult
}

const router = async () => {
  const header = null || document.getElementById('header-content');
  const main_content = null || document.getElementById("main-content");

  header.innerHTML = await Header.render();
  await Header.after_render();

  let request = Utils.parseRequestURL()

  let parsedURL = (request.resource ? '/' + request.resource : '/') + (request.id ? '/:id' :
  '') + (request.verb ? '/' + request.verb : '')
  console.log(parsedURL)

  let page = routes[parsedURL] ? routes[parsedURL] : Error404
  main_content.innerHTML = await page.render();

  await page.after_render();
}

// Listen on hash change:
window.addEventListener('hashchange', router);

// Listen on page load:
window.addEventListener('load', router);

```

3) Utils.js

```
const Utils = {
  // Parse a url and break it into resource, id and verb
  parseRequestURL : () => {

    let url = location.hash.slice(1) || '/';
    let r = url.split("/");

    let request = {
      resource: null,
      id: null,
      verb: null
    };

    request.resource = r[1];
    request.id = r[2];
    request.verb = r[3];

    return request;
  },

  createSnackbar: (function() {
    let queue = [];

    return function (message) {
      let snackbar = document.createElement('div');
      snackbar.className = 'snackbar';

      let text = document.createTextNode(message);
      snackbar.appendChild(text);

      snackbar.addEventListener('transitionend', function (event, elapsed) {
        if (event.propertyName === 'opacity' && this.style.opacity == 0) {
          this.parentNode.removeChild(this);
          queue.shift();

          if (queue[0]) {
            setSnackabrBehavior(queue[0]);
          }
        }
      }).bind(snackbar));

      if (queue.length == 0) {
        setSnackabrBehavior(snackbar);
      }

      queue.push(snackbar);
    };
  })()
```



```
}

function setSnackBarBehavior(snackbar) {
  setTimeout(function () {
    this.style.opacity = 0;

  }.bind(snackbar), 3000);

  document.body.appendChild(snackbar);

  getComputedStyle(snackbar).bottom;
  snackbar.style.bottom = '0px';
  snackbar.style.opacity = 1;
}

export default Utils;
```