СОДЕРЖАНИЕ

[Введение 4](#_Toc103561876)

[1. АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ 5](#_Toc103561877)

[**1.1** **Постановка задачи** 5](#_Toc103561878)

[2. ПРОЕКТИРОВАНИЕ Структуры приложения 7](#_Toc103561879)

[**2.1 Архитектура приложения** 7](#_Toc103561880)

[**2.1.1 Архитектура клиентской части** 7](#_Toc103561881)

[**2.1.1.1 Инструменты разработки клиентской части** 8](#_Toc103561882)

[**2.1.1.2 Структура проекта клиентской части приложения** 8](#_Toc103561883)

[**2.1.2 Архитектура серверной части** 11](#_Toc103561884)

[**2.1.2.1 Инструменты разработки серверной части** 11](#_Toc103561885)

[**2.1.2.2 Структура проекта серверной части приложения** 11](#_Toc103561886)

[**2.1.3 База данных** 12](#_Toc103561887)

[**2.1.3.1 Структура базы данных** 13](#_Toc103561888)

[3. РАЗработка алгоритмов функционирования и структур данных 14](#_Toc103561889)

[**3.1 Аутентификация и авторизация пользователя** 14](#_Toc103561890)

[**3.2 Создание теста** 15](#_Toc103561891)

[**3.3 Удаление теста** 15](#_Toc103561892)

[**3.4 Алгоритм прохождения теста учеником** 15](#_Toc103561893)

[**3.5 Алгоритм вывода тестов ученику** 16](#_Toc103561894)

[**3.6 Алгоритм вывода результатов теста преподавателю** 16](#_Toc103561895)

[4. Тестирование 17](#_Toc103561896)

[**4.1 Описание входных и выходных данных** 17](#_Toc103561897)

[**4.2 Результаты тестирования** 17](#_Toc103561898)

[Тест 1: Логин и регистрация пользователя администратором: 17](#_Toc103561899)

[Тест 2: Создание теста учителем 18](#_Toc103561900)

[Тест 3: прохождение теста учеником 20](#_Toc103561901)

[Тест 4: удаление теста 21](#_Toc103561902)

[ЗАКЛЮЧЕНИЕ 22](#_Toc103561903)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc103561904)

ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ

ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ

# Введение

Веб-приложение – клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети.

При построении архитектуры веб приложений, как правило, используется клиент-серверный подход. Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабат ывает ответы от него. Серверная часть получает запросы от клиента, выполняет их обработку, после этого формирует веб-страницу и отправляет ее клиенту по сети с использованием протокола HTTP. Однако в наше время редко встретишь приложение, которое после каждого запроса к серверу полностью обновляет страницу. Все чаще используется подход, при котором по сети отправляются только необходимые в рамках текущего запроса данные. Само веб-приложение может выступать в качестве клиента для других служб, например, серверная часть может выступать в качестве клиента для базы данных.

Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами. Из этого преимущества вытекают и другие, такие как доступ с различных устройств и отсутствие клиентского ПО.

### 

# АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ

В ходе выполнения данного курсового проекта необходимо разработать систему тестирования компетенций учащихся. Данная система должна иметь следующий функционал:

1. Разные уровни доступа к системе (обучающийся, администратор, преподаватель).
2. Добавление, изменение и удаление тестов.
3. Формирование сводной документации (отчет по прохождению теста).
4. Различные типы вопросов для включения в тест.
5. Конфигурация теста (набор настроек для определения процедуры прохождения тестирования: время, отводимое на тест, возможность получения обучающимся информации о правильных/неправильных ответах и их количестве и т.д.)

Несмотря на то, что данные требования являются довольно тривиальными в рамках разрабатываемой системы, необходимо предаварительно тщательно разработать архитектуру на разных уровнях приложения, также стоит уделить особое внимание 4 пунтку требований, а также распределить права между различными типами пользователей (например, какие права должны быть у администратора и учителя), и то, как должны выводится данные о правильных и неправильных ответах, а также то, как данные о проходимом тесте должны отправляться на сервер.

### **Постановка задачи**

1. Система должная иметь несолько уровней доступа:
   1. Обучающийся – базовый уровень. Может проходит назначенные на него тесты и получать результаты пройденных тестов.
   2. Учитель – может создавать, удалять и редактировать тесты, а также получать информацию о прохождении теста учениками.
   3. Администратор – может регистрировать и менять уровни доступа пользователей.
2. Добавление/изменение и удаление тестов – учитель должен иметь возможность удалять, добавлять, а также изменять существующие тесты. Под изменением теста подразумевается возможность удалять либо добавлять ответы.
3. Формирование сводной документации
   1. Ученик должен иметь возможность просматривать информацию о пройденном тесте (список вопросов и правильные ответы, процент правильных ответов).
   2. Учитель должен иметь возможность просматривать информацию о прохждении учениками созданного им тестов (процент правильных ответов для каждого ученика).
4. Различные типы вопросов для включения в тест:
   1. “Радио баттон” c одним правильным ответом.
   2. “Чекбокс” с несколькими правильными ответами.
   3. Произвольный ответ
5. Конфигурация теста – время, отводимое на тест.

# 2. ПРОЕКТИРОВАНИЕ Структуры приложения

### **2.1 Архитектура приложения**

Система тестирования студентов – веб-приложение с клиент-серверной архитекрутой (Рисунок 2.1.1)

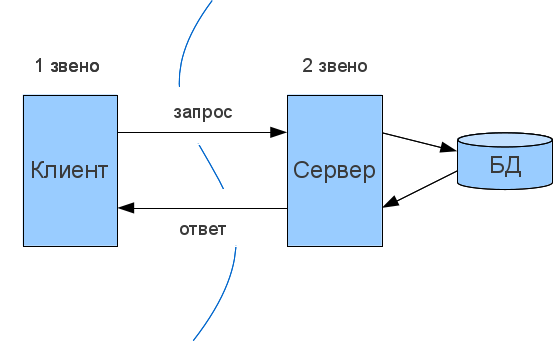


Рисунок 2.1.1 (Клиент-серверная архитектура)

### **2.1.1 Архитектура клиентской части**

Клиентская часть – веб-приложение, разработанное с помощью таких технологий как:

1. TypeScript 4.6.3 – язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript. Является обратно-совместимым с JavaScript и компилируется в него.
2. Angular 13.3.3 – платформа для разработки веб-приложений, разрабатываемая командой из Google, написанная на языке TypeScript.
3. PrimeNG – проект с открытым исходным кодом, предоставляющий большой набор UI компонентов для Angular.
4. HTML 5 – стандартизированный язык разметки документов для просмотра веб-страниц в браузере.
5. CSS 3 – формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

### **2.1.1.1 Инструменты разработки клиентской части**

Клиентская часть разрабатывалась с использованием таких инструментов как:

1. NPM (Node Packet Manager) – менеджер пакетов, позволяющий легко устанавливать, обновлять и восстанавливать программные пакеты, требуемые для разработки приложения.
2. JetBrains WebStorm – превосходная IDE для разработки веб-приложений от компании JetBrains.
3. Google Chrome – веб-браузер от компании Google

### **2.1.1.2 Структура проекта клиентской части приложения**

Структура проекта клиентской части приложения выглядит следующим образом (Рисунок 2.1.1.2.1):

Text

Description automatically generated

а Рисунок 2.1.1.2.1 (Структура проекта клиентской части)

1. node\_modules – содержит программные зависимости проекта, описанные в файле package.json либо package-lock.json.
   1. src – содержит исходный код приложения.
      1. index.html – страница веб-приложения. Клиент – SPA, поэтому index.html – единственная html страница которая нам нужна. Далее в нее будет подгружаться контент из отдельных компонентов приложения.
      2. main.ts – входная точка приложения. В нем происходит загрузка основного модуля приложения, через который инициализируются вспомогательные модули и комоненты.
      3. polyfills.ts – содержит код, позволяющий Angular приложению быть совместимым с различными браузерами.
      4. core – основное смысловое ядро приложения, в котором реализована основная логика приложения.
         1. configuration – содержит сервисы, обрабатывающие конфигурационные файлы и предоставляющие их данные остальной части приложения.
         2. guards – содержит классы, которые защищают пути и функционал приложения от неаутентифицированных пользователей, либо от пользователей не имеющих определенны привилегий.
         3. models – содержит модели представлений, либо модели для отправки серверу.
         4. services – содержит классы-сервисы приложений. Как правило это сервисы для взаимодействия с серверной частью.
      5. ui – содержит компоненты приложения. В Angular компоненты – кирпичики для создаения пользовательского интерфейса. Каждый комопненты обладает своей разметкой, стилями и логикой.
         1. componetns – папка с компонентами.
         2. components-list.ts – экспортирует список компонентов.
         3. ui.module.ts – модуль UI части приложения. Инициализирует и предоставляет остальным модулям приложения компоненты.
      6. app-routing.module.ts – модуль, в котором происходит конфигурация маршрутизации приложения.
      7. app.module.ts – основной модуль приложения, в котором происходит инициализация всех остальных модулей приложения.
      8. assets – содержит статичекские файлы приложения.
   2. angular.json – содержит конфигурацию проекта, а также инструментов разработки и доставки приложения.
   3. package.json – Angular приложение можно рассматривать как пакет NPM. package.json предоставляет информацию NPM для идентификации пакета, а также о его зависимостях
   4. tsconfig.json – содержит конфигурацию TypeScript компилятора.

### **2.1.2 Архитектура серверной части**

Серверная часть – REST API приложение, разработанное с помощью таких технологий, как

1. .NET 6 – модульная плафторма для разработки программного обеспечения с открытым исходным кодом от компании Microsoft.
2. C# 10 – объектно-ориентированны язык программирования от компании Microsoft.
3. ASP.NET Core 6 – кроссплатформенный фреймворк для создания веб-приложений на платформе .NET с открытым исходным от компании Microsoft.
4. Entity Framework Core – технология доступа к данным с открытым исходным данным от компании Microsoft.
5. Hangfire – Библиотека для конфигурации и запуска фоновых процессов.

### **2.1.2.1 Инструменты разработки серверной части**

Серверная часть разрабатывалась с использованием таких технологий как:

1. JetBrains Rider – превосходная IDE для разработки приложений на платформе .NET от компании JetBrains.
2. Postman – API плафторма для создания и использования API.

### **2.1.2.2 Структура проекта серверной части приложения**

Серверное приложение разрабатывалась с использование классической слоеной архитекруты. В приложении есть несколько слоев:

1. Tests.Domain - Слой доступа к данным
2. Tests.Application - Слой бизнес логики
3. Tests.Common - Общая библиотека – содержит классы представляющие собой конфигурацию проекта, классы-утилиты и логгер.
4. Tests.Api - Слой API

.Структура проекта серверной части приложения выглядит следующим образом (Рисунок 2.1.2.2.1):

Text

Description automatically generated

Рисунок 2.1.2.2.1 (Структура серверной части)

Каждый слой имеет набор интерфейсов, их имплементаций и моделей. Помимо этого слой API имеет набор API контроллеров, а слой бизнес логики имеет набор процессов, которые запускаются в параллельном потоке приложения.

### **2.1.3 База данных**

В данном проекте используется база данных SQLite – компатная встраиваемая СУБД. Для доступа к данным используется технлогия Entity Framework Core.

### **2.1.3.1 Структура базы данных**

Graphical user interface

Description automatically generated

Рисунок 2.1.3.1.1 – структура базы данных.

# 3. РАЗработка алгоритмов функционирования и структур данных

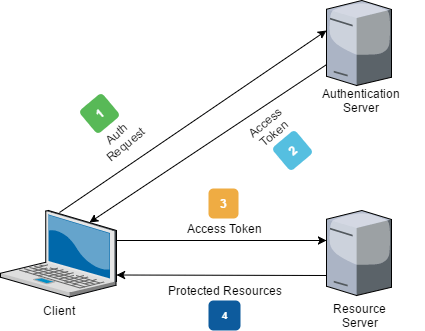
### **3.1 Аутентификация и авторизация пользователя**

Пользователь не может сам зарегистироваться в приложении, для этого Адмиинистратор должен зарегистрировать нового пользователя в панели Адмиинистратора и предоставить логин и пароль новому пользователю, используя которые он сможет пройти аутентификацию. В данном приложении используется авторизация и аутентификация на основе JWT (JSON Web Token) токена. JWT – отктытый стандарт (RFC 7519) который определяет компактный и автономный способ безопасной передачи информации между сторонами в виде объекте JSON. Передаваемой информации можно доверять, так как она имеет цифровую подпись. JWT токены могут быть подписаны с помошью HMAC алгоритма.

JWT токены состоят из трех частей:

1. Заголовок
2. Полезная нагрузка
3. Цифровая подпись

И обычно выглядят следующим образом: xxxx.yyyy.zzzz



РРисунок 3.1.1 (Алгоритм аутентификации на основе JWT)

В текущей системе зарегистрировать пользователей может только пользователь, обладающий правами админситратора. Также администратор может менять права пользователей.

### **3.2 Создание теста**

В текущей системе создать тест может только пользователь, обладающий правами учителя. Для создания теста используется следующий алгоритм:

1. Пользователь переходит на страницу создания теста.
2. Пользователь создает необходимые вопросы.
3. Пользователь устанавливает время отведенное на тест.
4. Пользователь выбирает список учеников, которые получат данный тест.
5. Пользователь публикует тест.
6. Тест отправляется на сервер в формате JSON.
7. Тест попадает в базу данных.
8. Вопросы в тесте попадают в базу данных.

### **3.3 Удаление теста**

В текущей системе удалить тест может только пользователь, обладающий правами учителя. Для удаления теста используется следующий алгоритм:

1. Пользователь переходит на страницу со списком опубликованных им тестов.
2. Пользователь выбирает и удаляет выбранный им тест.
3. Id теста отправляется на сервер.
4. Если тест существует.
   1. Тест удаляется из базы данных и страницы клиента.
5. Иначе.
   1. Сообщение об ошибке отображается на клиенте.

### **3.4 Алгоритм прохождения теста учеником**

В теущей система проходить тесты могут только пользователи, обладающие правами ученика.

1. Пользователь выбирает тест со статусом “Assigned” в списке доступных ему тестов.
2. На экране пользователя отображается список доступных ему вопросов
3. Пользователь отвечает на доступные вопросы
4. Пользователь нажимает кнопку Submit
5. На сервер отправляется Id теста и список ответов. Каждый ответ содержит Id вопроса и ответ пользователя.
6. Сервер достает из БД тест с указанным Id, а также всю необходимую информацию, относящуюся к тесту: список вопросов. Ответы пользователя заносятся в базу данных, статус теста становится Done.

### **3.5 Алгоритм вывода тестов ученику**

1. Пользовтель заходит на главную страницу.
2. Сервер получает GET запрос.
3. Сервер достает тесты из базы данных, назначенные текущему пользователю.
   1. Если статус теста “Done”, вычисляется резльтат теста
4. Сервер отправляет данные о тестах клиенту в формате JSON.

### **3.6 Алгоритм вывода результатов теста преподавателю**

1. Учитель заходит на страницу созданных им тестов.
2. Учитель выбирает определенный тест.
3. Id выбранного теста отправляется на сервер.
4. Сервер достает все записи из таблицы TestResults для заданного теста.
5. Сервер отправляет записи серверу в формате JSON.

# 4. Тестирование

### **4.1 Описание входных и выходных данных**

Данные, передаваемые клиентом серверу и серверу клиенту отправляются в формате JSON. Передаваемые данные, как правило, являются объектом определенного класса, перед передачей, объект сериализуется в JSON формат и вставляются в тело HTTP запроса, после того, как они получены, данные извлекаются из тела запроса и десериализуются в объект.

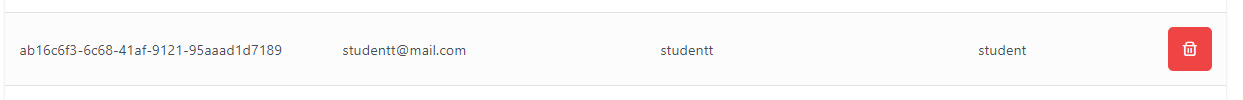
### **4.2 Результаты тестирования**

### Тест 1: Логин и регистрация пользователя администратором:

Ожидаемый результат: пользователь с правами администратора создает нового пользователя, после чего созданный пользователь может войти в систему.

Graphical user interface, application

Description automatically generated



Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Вывод: как видим, администратор регистрирует нового пользователя, после чего пользователь появляется в списке администратора. Созданный пользователь может войти в систему и увидеть список назначенных ему тестов.

### Тест 2: Создание теста учителем

Ожидаемый результат: Учитель может создать новый тест, назначить на него учеников, создать различные типы вопросов. Ученики, назначенные на тест смогут увидеть этот тест.

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

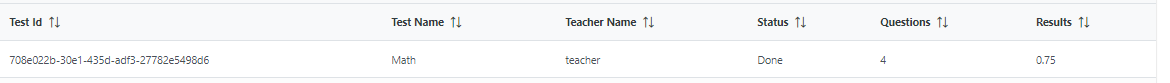
Вывод: учитель может создать новый тест, назначить на него учеников, создать различные типы вопросов, после чего учитель может увидеть созданный тест в таблице, а назначенные ученики пройти его.

### Тест 3: прохождение теста учеником

Ожидаемый результат: Ученик выбирает назначенный ему тест, проходит его и отправляет ответы, после чего учитель и ученик могут увидеть результаты.

Application

Description automatically generated



Graphical user interface, text, application, chat or text message

Description automatically generated

Вывод: ученик может пройти назначенный ему тест, после чего он и его учитель смогут увидеть результаты.

### Тест 4: удаление теста

Ожидаемый результат: учитель может удалить тест после чего он пропадает с его таблицы и таблицы тестов, тех учеников, которым он был назначен.

A screenshot of a computer

Description automatically generated

Graphical user interface, application

Description automatically generated

Вывод: тесты удаляются корректно.

# ЗАКЛЮЧЕНИЕ

В результате курсового проекта было разработанно программное обеспеченеие которое позволяет учителям проводить контроль компетенции учащихся. А именно создавать тесты и проверять результаты их прохождения.

Подобные системы широко используются в различных учебных заведениях и позволяют сэкономить время и ресурсы как преподователям, так и учащимся.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам.
2. ГОСТ 19.504-79. Единая система программной документации ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.
4. ГОСТ 19.005-85. ЕСПД.

Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.

1. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.
2. ГОСТ 19.102-77. ЕСПД. Стадии разработки.
3. ГОСТ 19.103-77. ЕСПД. Обозначения программ и программных документов.
4. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.
5. ГОСТ 19.402-78. ЕСПД. Описание программы.
6. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.