Федеральное государственной образовательное учреждение

Высшего профессионального образования

Национальный исследовательский технологический университет «МИСиС»

Институт информационных технологий и компьютерных наук (ИТКН)

Кафедра автоматизированных систем управления (АСУ)

Курсовая работа

по дисциплине «Разработка клиент-серверных приложений»

Тема:

«Создание сайта для изучения английского языка»

Выполнил: студент 2 курса,

гр. БИВТ-21-6 Тягельский А.А.

Руководитель: Рзазаде У. А.

**Оглавление**

[Введение 3](#_Toc136228457)

[Постановка задачи 4](#_Toc136228458)

[Описание используемых технических средств разработки 6](#_Toc136228459)

[Описание архитектуры проекта 9](#_Toc136228460)

[Описание структуры базы данных проекта 11](#_Toc136228461)

[Описание серверной части проекта 14](#_Toc136228462)

[Описание клиентской части проекта 20](#_Toc136228463)

[Заключение 21](#_Toc136228464)

[Список литературы 22](#_Toc136228465)

# Введение

В настоящее время изучение английского языка стало неотъемлемой частью базового образования, ведь данный язык является международным и используется для коммуникации по всему миру. И это не удивительно, ведь для 360 миллионов человек английский язык является родным, а 20% от всего населения Земли владеют данным языком. Помимо всего прочего около 90% всех научных статей опубликованы именно на английском языке, что делает английский язык чрезвычайно необходимым для изучения, поскольку профессиональный рост невозможен без чтения специфической технической литературы.

Все приведённые выше аспекты сталкивают нас с необходимостью совершенствовать свои навыки владения английским языком, для чего на данный момент существует огромное количество различных способов. Одним из наиболее интересных является игровая форма запоминания информации, в процессе игры клиент не успевает соскучиться и устать, при этом для запоминания и воспроизведения информации используется и зрительная и моторная виды памяти. Клиент-серверные приложения представляют собой одно из наиболее эффективных решений для создания сайта/приложения по изучению английского (или любого другого) языка, а также мониторинга и изучению статистики успешности прохождения того или иного курса/раздела по изучению второго языка.

Клиент-серверные приложения позволяют распределить нагрузку между клиентскими и серверными компонентами, что обеспечивает более быстрое и надежное выполнение операций. Кроме того, такие приложения позволяют централизованно управлять данными, что обеспечивает более высокий уровень безопасности и защиты данных.

В данной курсовой работе рассматривается разработка клиент-серверного приложения для изучения английского языка. Основной целью работы является создание системы, которая поспособствует приятному и эффективному изучению английского языка, а также мониторингу статистики и сбору данных об успехах клиента. Клиенту будет достаточно интересно и полезно отслеживать свои статистические данные, например, для того, чтобы узнать темы, в которых у него возникает наибольшее число затруднений.

Для достижения поставленных целей был проанализирован рынок похожих/смежных в данной области знаний приложений, были рассмотрены основные принципы разработки клиент-серверных приложений, а также технологии, используемые при создании таких приложений. В результате проведённой работы была разработана система, позволяющая эффективно и интересно изучать английский язык, а также отслеживать свои успехи/неудачи посредством изучения получаемой статистики. На данном этапе работы, клиентская часть сервиса не была разработана в полной мере, поскольку это и не являлось главной целью проекта, однако всё взаимодействие с сервером и данными предоставлено в полном объёме.

# Постановка задачи

В качестве главной задачи данного проекта выступает разработка компонентов клиент-серверного приложения для изучения английского языка на базе объектно-ориентированного языка программирования TypeScript, с использованием фреймворка NestJS и хранением данных в реляционной базе данных СУБД Postgres. Для успешного выполнения этой задачи необходимо решить следующие подзадачи:

1. Анализ рынка сайтов с игровой системой изучения английского языка. Необходимо проанализировать существующие решения, определить их достоинства и недостатки, а также определиться со стратегией и особенностями разработки приложения.
2. Анализ требований к приложению. Необходимо определить и составить функциональные и нефункциональные требования к приложению, основываясь на результатах анализа предыдущего пункта. Помимо этого. необходимо отсортировать данный список требований и определиться с первостепенными задачами проекта.
3. Изучение технической части исполнения проекта. Для реализации требований, предъявляемых к приложению, необходимо определить основной стэк используемых технологий и архитектуру приложения.
4. Проектирование базы данных. На основе составления технической части исполнения требования необходимо разработать структуру базы данных для хранения информации в виде взаимосвязанной системы таблиц, в которой будет содержаться информация по оговоренной ранее теме.
5. Разработка серверной части приложения. Необходимо разработать серверную часть приложения, которая будет обеспечивать операции вставки, удаления, модификации, добавления данных в базу. Также сервер необходим для обработки данных, и доступу к этим данным через интуитивно-простой API.
6. Разработка клиентской части приложения. Необходимо продумать (в нашем случае разработка является дополнительной опцией, а не отдельнойзадачей) клиентскую часть приложения, которая будет обеспечивать удобный интерфейс для работы с данными, которые использует клиент.
7. Тестирование приложения. Необходимо провести тестирование приложения на соответствие заявленным ранее требованиям и выявить возможные ошибки и недоработки, которые в последующих версиях приложения можно будет исправить или улучшить.
8. Оформление документации. Необходимо оформить документацию по разработке приложения, включая описание требований, архитектуры, базы данных и тестирования.

# Описание используемых технических средств разработки

Рассмотрим технологии, которые были применены для реализации данного проекта

1). TypeScript и объектно-ориентированный подход

При разработке приложения были применены знания из области базовых средств объектно-ориентированного языка программирования TypeScript. Понятие «объектно-ориентированный» относится к способу написания структурного кода, а именно: разделение кода на так называемые «классы», которые запускаются вместе, чтобы обеспечить согласованное порождение объектов. Класс представляет шаблон для создания объектов и инкапсулирует функциональность, которую должен иметь объект. Класс определяет состояние и поведение, которыми обладает объект. Такая структура программы приводит к универсальному и организованному коду, который легко редактировать и воспринимать. Кроме обычных методов классы могут определять специальные методы, которые называются конструкторами. Конструкторы вызываются при создании нового объекта данного класса и выполняют его инициализацию.

**TypeScript** - это язык программирования, который является надмножеством JavaScript. TypeScript реализует объектно-ориентированный подход, в нем есть полноценная поддержка классов. Он позволяет разработчикам писать более структурированный и понятный код, а также обеспечивает статическую типизацию, что позволяет выявлять ошибки на этапе компиляции. TypeScript является одним из основных инструментов для разработки клиент-серверных приложений. TypeScript не интерпретируется во время выполнения, а компилирует файлы TypeScript (.ts) в файлы JavaScript (.js).

2). PostgreSQL и реляционная модель представления данных

Совокупность данных формирует базу данных — массив, в котором собраны и организованы по некоторому принципу данные из какой-либо области знаний. Принцип построения базы данных определяется ее моделью. Наиболее известны иерархическая, сетевая, объектная и реляционная модели.

PostgreSQL относится именно к реляционной модели базы данных, про которую необходимо упомянуть в данном разделе. **Реляционная база данных** — это составленная по реляционной модели база данных, в которой данные, занесенные в таблицы, имеют изначально заданные отношения. Сами таблицы в такой базе данных также соотносятся друг с другом строго определенным образом. Реляционные базы данных используют целый комплекс инструментов, которые обеспечивают целостность данных, т. е. их точность, полноту и единообразие.

Для взаимодействия с любой реляционной базой данных используется SQL (Structured Query Language) — язык структурированных запросов. Это основа интерфейса систем управления базами данных.

**PostgreSQL** - это реляционная база данных с открытым исходным кодом, которая обладает высокой производительностью и надежностью. Она используется для хранения данных пользователей, а также пула данных, связанных с изучением языка и основным особенностей приложения, например, информация о словах и их переводе, идиомах, статистике пользователей и т.д. PostgreSQL обладает богатым набором функций и поддерживает многопоточность, что делает его идеальным выбором для разработки приложений с большим объемом данных.

3). NestJS, важнейшие зависимости и принципы

**NestJS** - это фреймворк для разработки эффективных и масштабируемых серверных приложений на Node.js, в котором используется прогрессивный JavaScript с полной поддержкой TypeScript (использование последнего является опциональным) и сочетает в себе элементы объектно-ориентированного, функционального и реактивного функционального программирования. Данный фреймворк предоставляет разработчикам множество инструментов и функций для простого и удобного процесса создания классических клиент-серверных приложений. NestJS использует модульную архитектуру и предоставляет множество интеграций с другими технологиями, такими как TypeORM и PostgreSQL.

Важнейшими элементами NestJS и некоторых других фреймворков (например, Spring Web) является модель создания приложения на основе MVC. MVC.

**Model-View-Controller** (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо. Простая и удобная картинка, характеризующая MVC расположена на рис.1.

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок №1

Ещё одним важным компонентом является TypeORM.

**ORM** (Object-Relational Mapping) - технология программирования, которая позволяет преобразовывать несовместимые типы моделей в ООП, в частности, между хранилищем данных и объектами программирования. ORM используется для упрощения процесса сохранения объектов в реляционную базу данных и их извлечения, при этом ORM сама заботится о преобразовании данных между двумя несовместимыми состояниями.

**TypeORM** - это ORM для TypeScript и JavaScript, который позволяет разработчикам работать с базами данных, используя объектно-ориентированный подход. Он предоставляет множество функций, таких как миграции баз данных, связи между таблицами и т.д. TypeORM поставляется со следующими функциями :

* Автоматическое создание схем таблиц базы данных на основе ваших моделей.
* Простая вставка, обновление и удаление объекта в базе данных.
* Создайте сопоставление (один к одному, один ко многим и многие ко многим) между таблицами.
* Предоставляет простые команды командной строки.

Преимущества TypeORM:

* Высококачественные и слабо связанные приложения.
* Масштабируемые приложения.
* Простая интеграция с другими модулями.

TypeORM интегрируется с NestJS и PostgreSQL, и идеально подходит для любой архитектуры - от небольших приложений до корпоративных.

4). Git и система контроля версий

Система контроля версий - это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов (типичный пример: выложили версию приложения в продакшен, начали работу над новыми особенностями и вдруг обнаружились ошибки. Нужно не потеряв новых наработок вернуться к рабочей версии, исправить ошибки).

Git - это система контроля версий, которая позволяет разработчикам отслеживать изменения в коде, вносить исправления и улучшения, а также сотрудничать в команде. Git является одним из наиболее популярных инструментов для управления исходным кодом и используется для разработки клиент-серверных приложений.

# Описание архитектуры проекта

Архитектура проекта для курсовой работы представляет собой реализацию клиент-серверного приложения, состоящего из определённого набора компонентов и технологий, многие из которых уже были перечислены в теоретической части данной пояснительной записки: TypeScript, PostgreSQL, NestJS (на основе NodeJS.), TypeORM и Git.

В первую очередь стоит более подробно разобраться в клиент-серверной архитектуре, после чего описать используемые компоненты.

Клиент-серверную архитектура - вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. В нашем случае, клиент и сервер находятся на одном устройстве.

Схему работы клиент-серверной архитектуры можно увидеть на рис.2.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описаниеРисунок №2

Преимущество клиент-серверных приложений:

* Отсутствие дублирования кода программы-сервера программами-клиентами.
* Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
* Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще организовать контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа.

Недостатки клиент-серверных приложений:

* Неработоспособность сервера может сделать неработоспособной всю вычислительную сеть. Неработоспособным сервером следует считать сервер, производительности которого не хватает на обслуживание всех клиентов, а также сервер, находящийся на ремонте, профилактике и т. п.
* Поддержка работы данной системы требует отдельного специалиста — системного администратора.
* Высокая стоимость оборудования.

**Компоненты**

Архитектура проекта основана на принципе MVC (Model-View-Controller), который позволяет разделить приложение на три основных компонента: модель, представление и контроллер. Более подробно данный принцип расписан в главе с описанием используемых технических средств разработки.

Для работы с базой данных PostgreSQL посредством написания кода в TypeScript используется ORM TypeORM, который позволяет работать с данными посредством написания кода в объектно-ориентированном языке программирования.

Для контроля версий в проекте используется распределённая система контроля версий Git, что позволяет работать в команде и контролировать изменения, которые вносят другие участники проекта. В нашем случае Git позволяет откатиться на более старую версию проекта, избежав ошибок на стадии деплоя.

Для клиентской части приложения используется фреймворк NestJS, который позволяет создавать масштабируемые и модульные приложения на NodeJS. Приложение разбито на модули, что позволяет разделить его на логические части и упростить масштабирование.

# Описание структуры базы данных проекта

Структура базы данных для изучения английского языка содержит пять таблиц: Words, Arts, Idioms, Students и Progress. Более подробно рассмотрим назначение каждой из созданных таблиц.

Таблица Words является самой большой и информационно-полезной среди всех здесь представленных. Она содержит идентификатор (Id), слово на английском языке (name), предметную область слова (topic), часть речи (partOfSpeech), перевод на русский язык (translation), уровень (level) и частоту встречи в речи на 1000 слов(rapidness).

Таблица Arts представляет собой набор информации о какой-либо книге/песне и самом распространённом слове в этом произведении. Она содержит идентификатор (Id), название (name), тип произведения (книга/фильм/песня) (type), автора (author), самое встречающееся слово (topWord), и частоту его употребления(frequency).

Таблица Idioms представляет собой набор информации об идиоме. Она содержит идентификатор (Id), текст идиомы (text), предметную область данной идиомы (topic), и длину данной идиомы (length).

Таблица Students представляет собой набор информации о пользователях приложения. Она содержит идентификатор (student\_id), имя клиента (student\_name), его возраст (student\_age), и уровень владения языком (student\_grade).

Таблица Students представляет собой набор информации о пользователях приложения. Она содержит идентификатор (id), область знаний (subject) и результат прохождения (score).

Для удобства работы с базой данных, а также для расширения функциональности мы создали некоторое число триггеров, функций, представлений и хранимых процедур. Не имеет смысла приводить в данном проекте все вышеизложенные структуры, но отобразить некоторые из них стоит.

* ProgressStatistics, который представлен на рис. 3, содержит информацию о успеваемости каждого клиента, включая их уникальный идентификатор (Id), минимальный результат (MIN), максимальный результат (MAX), средний результат (AVG), название предметной области (subject).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок №3

* TopStudents, представленный на рисунке 4, содержит информацию о, успеваемости студентов, средняя оценка которых превышает 90 баллов. Данное представление содержит уникальный идентификатор (student\_id), имя студента (student\_name), среднюю оценку (AVG).

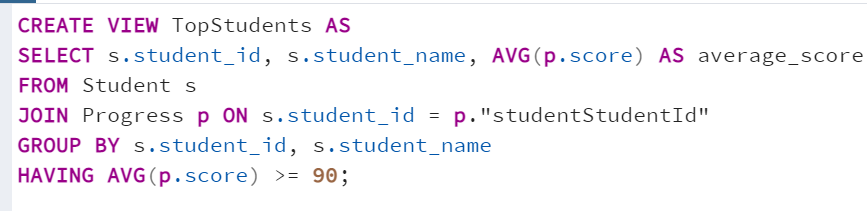


Рисунок №4

* UpdateStudentAge, представленный на рисунке 5, - функция, которая позволяет обновлять в таблице Students поле возраста для пользователя. На вход берёт (student\_id), после чего обновляет поле (student\_age)? добавляя к прежнему числу единицу.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок №5

Структура всей базы данных представлена на рисунке 6.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок №6. Таблицы БД.

Таким образом, структура базы данных для изучения английского языка и сбора статистики успехов пользователей позволит удобно хранить и обрабатывать информацию о пользователях, словах и идиомах, а также отображать ее в удобном виде для пользователей системы.

# Описание серверной части проекта

Серверная часть приложения представляет собой RESTful API, которое обрабатывает запросы от клиентской части и взаимодействует с базой данных PostgreSQL.

Архитектура серверной части представлена на рис.7

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок №7. Архитектура сервера.

Далее рассмотрим на примере сущности Words структуру создания и обработки информации, связанной с конкретной сущностью, а также заглянем внутрь файла app.module.ts.

На рис.8 изображено устройство класса CreateWordDto, который отвечает за создание новых слов.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок №8

Половина полей остаётся незаполненными при добавлении нового слова, и это один из примеров использования DTO (Data Transfer Object). Теперь нам необязательно заполнять все поля, а можно заполнить только данные 3.

На рис.9 изображена структура контроллера для класса Word, который отвечает за взаимодействие клиента и сервера (обработку действий пользователя).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок №9

Можно увидеть, что реализованы все CRUD операции, а также дополнительно получение неполной информации из таблицы. CRUD включает в себя операции добавления, получения, изменения и удаления данных с сервера.

На рис.10 изображена часть структуры кода сервиса для класса Word, который отвечает за бизнес логику обработки данных.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок №10

На данном рисунке предоставлены для обозрения 2 функции, отвечающие за обработку операций создания и нахождения слова по id. Именно данные методы вызываются в контроллере при обработке одноимённых событий.

Теперь рассмотрим app.module.ts, изображённый на рис.11.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок №11

В данном файле мы подтягиваем все компоненты проекта, а помимо этого конфигурируем подключение к базе данных, указывая тип базы данных, порт сервера, имя пользователя, пароль и всё остальное. Также не забываем про опцию autoLoadEntities: true, чтобы TypeScript подтянул все ссылки правильно.

Для удобства контроля и отслеживания запросов была использована документация Swagger UI, представленная на рис. 12.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 12. Документация Swagger UI.

По сути, в данном проекте именно Swagger UI позволяет определить правильность выполнения поставленной задачи, отследить ошибки и исправить их. Поскольку клиентская часть не была соединена с серверной частью в полной мере, то Swagger UI является в некотором смысле площадкой, где клиент общается с сервером, запрашивая какие-либо данные (или отправляя их на сервер).

В приложении реализованы следующие функции: просмотр всех англоязычных слов, находящихся в базе, просмотр идиом и самых часто употребляемых словах в популярных произведениях, просмотр статистики учёбы клиентов. Для каждой из этих функций разработаны соответствующие контроллеры и сервисы.

TypeORM используется для создания таблиц и связей между ними, а также для выполнения запросов к базе данных напрямую через TypeScript без необходимости взаимодействия с Postgres через систему управления (pgAdmin к примеру). На рисунках 13 и 14 изображены примеры обращения к базе данных через Swagger UI.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 13. Результат выполнения запроса к серверу.

Запрос на получение информации из таблицы Words. Как мы видим операция выполнена успешно, о чём символизирует код 200 и полученное в ответ тело запроса.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок №14

Запрос на удаление данных также работает корректно, о чём свидетельствует код выполнения и полученный заголовок ответа.

В целом, серверная часть проекта представляет собой надежную и масштабируемую систему, которая позволяет пользователю погрузиться в изучение английского языка и следить за своей статистикой. Помимо этого, приложение имеет потенциал для дальнейшей разработки и добавления новых фишек.

# Описание клиентской части проекта

Клиентская часть проекта на данный момент не имеет полноценной связи с серверной, однако, с помощью Swagger UI и оформленных Web-страниц можно эмулировать клиентскую часть данного приложения.

Сайт представляет собой выстроенную иерархию, где из Main Menu можно зайти в Game Palletes, откуда открыт доступ к выбору нужной игры. Смотреть рис.15

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок №15

На данном этапе разработки клиент-серверного приложения клиентская часть была разработана в отрыве от серверной, что делает проверку работоспособности всей системы невозможной. Тем не менее, клиентская часть при дальнейшей доработке программы будет представлять полноценную и удобную систему для изучения английского языка. Swagger UI эмулирует клиентские окна по отправке и получению необходимой информации, а HTML страницы сайта дают представление о front части данного проекта.

# Заключение

Разработка клиент-серверных приложений для изучения английского языка является важным шагом для облегчения учебного процесса и способствует развитию данной отрасли. Такие приложения позволяют ускорить обучение, используя игровой подход и удобную структуру хранения, обработки, взаимодействия с информацией. Сбор статистики по каждому пользователю нагружает систему, но даёт ясное представление об успехах того или иного студента в изучаемой области.

При разработке таких приложений необходимо проанализировать рынок схожих приложений, выявить все недостатки и плюсы таковых. После чего необходимо выстроить стратегию и требования к создаваемому продукту. Для успеха приложения также требуется удобный UI, быстрое взаимодействие с базой данных и уникальные фишки.

По итогу разработки приложения, были проведены тестирования, которые показали его высокую производительность и надежность. В результате, разработанное приложение для изучения английского языка может быть использовано, чтобы повысить уровень знания английского языка у студентов, учеников школ и любопытствующих.

# Список литературы

1. Даков В. "HTML и CSS. Разработка и дизайн веб-сайтов". Москва: ДМК Пресс, 2019.с.
2. Корнелюк А. "PostgreSQL. Проектирование и разработка баз данных". Москва: ДМК Пресс, 2019.
3. Кузнецов В. "NestJS. Разработка масштабируемых веб-приложений на Node.js". Москва: ДМК Пресс, 2020.
4. Git https://git-scm.com/book/ru/v2/Введение-О-системе-контроля-версий
5. Чакрыгин М. "Git. Система контроля версий". Москва: ДМК Пресс, 2019.
6. Карпов А.А. "Разработка клиент-серверных приложений на языке TypeScript". Москва: ДМК Пресс, 2019.
7. Козлов А. "TypeORM. Разработка баз данных на TypeScript". Москва: ДМК Пресс, 2020.
8. Wikipedia “Модель MVC” https://ru.wikipedia.org/wiki/Model-View-Controller
9. Макфарланд А., Бутстрап Т. "Bootstrap 4. Руководство разработчика". Москва: ДМК Пресс, 2018.

Git репозиторий проекта находится по адресу: