

## Содержание

|   |    |
|---|----|
| Введение .....  | 3  |
| 1 Обзор аналогов и постановка задачи .....                    | 4  |
| 1.1 Основные сведения по теме «Веб-приложение» .....          | 4  |
| 1.2 Обзор аналогичных решений .....                           | 4  |
| 1.2.1 Веб-приложение Duolingo.com .....                       | 4  |
| 1.2.2 Веб-приложение Poliglot16.ru .....                      | 5  |
| 1.2.3 Веб-приложение Puzzel-english.com .....                 | 6  |
| 1.3 Анализ прототипов .....                                   | 7  |
| 1.4 Постановка задачи .....                                   | 7  |
| 2 Проектирование веб-приложения .....                         | 9  |
| 2.1 Диаграмма использования .....                             | 9  |
| 2.2 Логическая схема базы данных .....                        | 13 |
| 2.3 Архитектура веб-приложения .....                          | 19 |
| 2.4 Вывод по разделу .....                                    | 20 |
| 3 Реализация веб-приложения .....                             | 21 |
| 3.1 Программная платформа Node.js .....                       | 21 |
| 3.2 СУБД PostgreSQL .....                                     | 21 |
| 3.3 Средство моделирования объектного документа .....         | 21 |
| 3.4 Библиотеки и фреймворки .....                             | 22 |
| 3.4 Структура серверной части .....                           | 24 |
| 3.8 Структура клиентской части .....                          | 27 |
| 3.9 Вывод по разделу .....                                    | 29 |
| 4 Тестирование программного продукта .....                    | 30 |
| 4.1 Функциональное тестирование для роли Пользователя .....   | 30 |
| 4.2 Функциональное тестирование для роли Администратора ..... | 33 |
| 4.3 Функциональное тестирование для роли Гостя .....          | 37 |
| 4.4 Вывод по разделу .....                                    | 37 |
| 5 Руководство пользователя .....                              | 38 |
| 5.1 Руководство Гостя .....                                   | 38 |
| 5.2 Руководство Пользователя .....                            | 39 |
| 5.3 Руководство администратора .....                          | 42 |
| Заключение .....  | 45 |
| Список использованных источников .....                        | 46 |
| Приложение А .....  | 49 |
| Приложение Б .....  | 53 |
| Приложение В .....  | 55 |
| Приложение Г .....  | 57 |
| Приложение Д .....  | 61 |
| Приложение Е .....  | 65 |
| Приложение Ж .....  | 71 |

## Введение

Целью данного курсового проекта является разработка веб-приложения для изучения испанского языка. Оно будет позволять пользователям улучшать свои навыки и знания в любое удобное время и в любом месте. В рамках проекта будет рассмотрено несколько аспектов, необходимых для создания и успешного функционирования веб-приложения.

Для достижения цели были поставлены следующие задачи:

1. Провести анализ существующих интернет-магазинов электроники и определить ключевые требования к разрабатываемому веб-приложению (раздел 1);
2. Разработать архитектуру веб-приложения (раздел 2);
3. Реализовать функционал веб-приложения с учетом поставленных требований (раздел 3);
4. Провести тестирование для выявления и устранения ошибок, а также для проверки соответствия требованиям (раздел 4);
5. Разработать руководство пользователя веб-приложения (раздел 5)

Целевая аудитория веб-приложения — люди с доходом от среднего и выше среднего, которые ищут надежную электронику для повседневной жизни, работы, учебы и досуга.

В качестве программной платформы было решено использовать Node.js [1]

## 1 Обзор аналогов и постановка задачи

В ходе выполнения курсового проекта были проанализированы аналоги приложения для самостоятельного изучения испанского языка, выявлены их отличительные особенности и поставлены задачи курсового проекта.

### 1.1 Основные сведения по теме «Веб-приложение»

Веб-приложение представляет собой веб-сайт, на котором размещены страницы с частично либо полностью несформированным содержанием. Окончательное содержание формируется только после того, как посетитель сайта запросит страницу с веб-сервера. В связи с тем, что окончательное содержание страницы зависит от запроса, созданного на основе действий посетителя, такая страница называется динамической.

Спектр использования веб-приложений очень широк. Использование веб-приложений приносит определенную пользу как посетителям веб-сайтов, так и их разработчикам. Например, посетители могут быстро и легко находить требуемую информацию на веб-сайтах с большим объемом информации, а разработчики сохранять и анализировать данные, полученные от посетителей сайта.

Долгое время использовался метод, при котором данные, введенные в *HTML*-формы, отсылались для обработки *CGI*-приложениям или специально назначенным работникам в виде сообщений электронной почты. Веб-приложение позволяет сохранять данные непосредственно в базе данных, а также получать данные и формировать отчеты на основе полученных данных для анализа. В качестве примера можно привести интерактивные страницы банков, страницы для контроля товарных запасов, социологические исследования и опросы, а также формы для обратной связи с пользователями.

Веб-приложение может использоваться для обновления веб-сайтов с периодически меняющимся содержанием.

### 1.2 Обзор аналогичных решений

Одним из ключевых моментов в разработке программного обеспечения является изучение аналогов, выявление достоинств и недостатков в них. С помощью анализа аналогов можно выделить функционал, который обязательно должен присутствовать в веб-приложении. Это необходимо для построения каркаса будущего приложения. Также анализ помогает выделить недостатки и избежать их в собственной реализации.

#### 1.2.1 Веб-приложение Duolingo.com

*Duolingo*[2] — самая популярная платформа для самостоятельного изучения языков. Данное веб-приложение имеет следующие возможности:

- изучение грамматических правил и слов по определенным темам в виде игрового процесса;

- тематические диалоги, которые позволяют приблизиться к жизненным ситуациям;
  - личная статистика прогресса и статистика друзей;
  - система достижений и бонусов за получение этих достижений.
- Интерфейс веб-приложения представлен на рисунке 1.1.

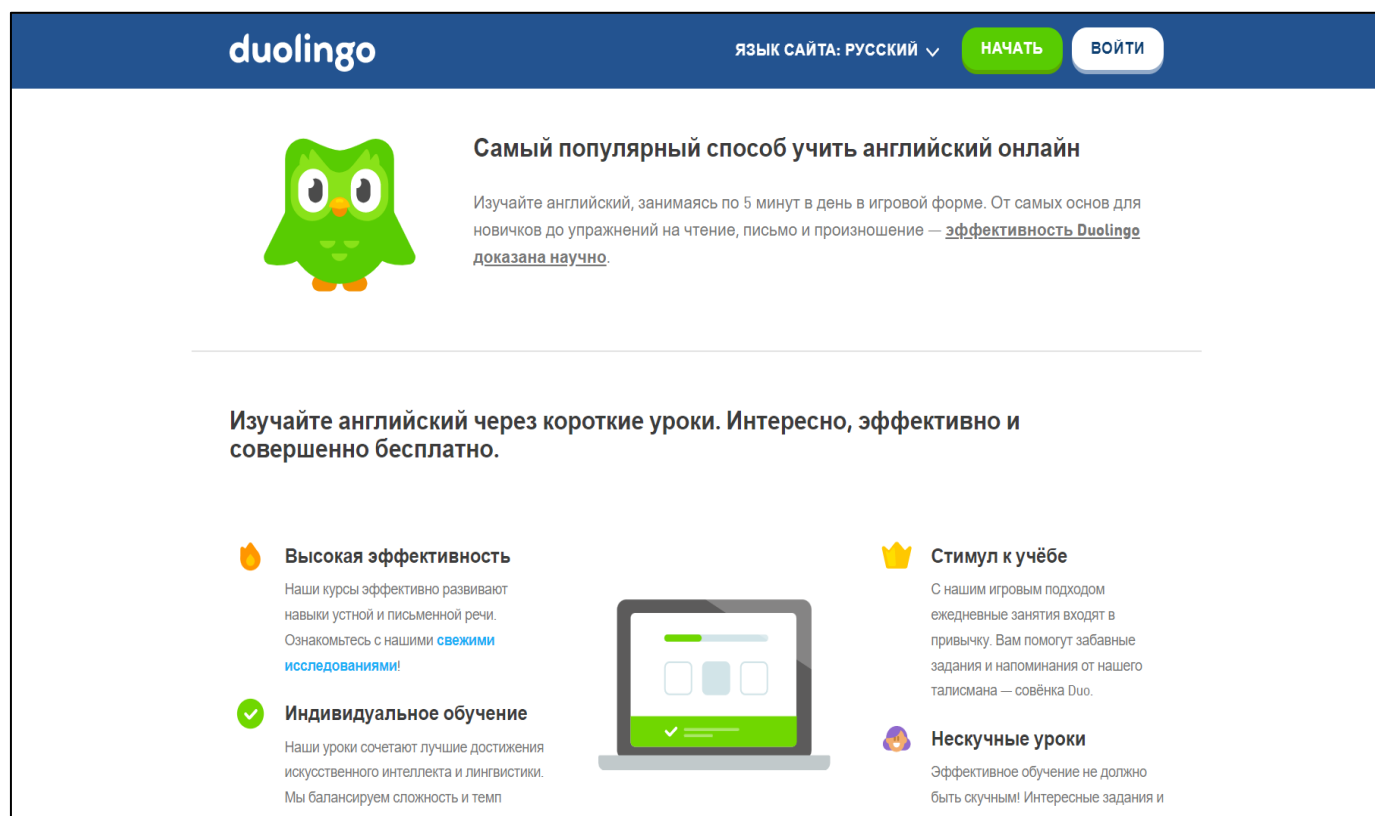


Рисунок 1.1 – Веб-приложение *duolingo.com*

Из недостатков необходимо отметить то, что имеющийся контент позволяет получить только начальные навыки овладения языком. Также нет возможности начать обучение определённых тем или уроков, не пройдя до этого предыдущие. Также невозможно настроить приложение под свои навыки. У приложения есть и плюсы, например, за различные достижения пользователю вручаются награды (за количество выученных слов, за ежедневные посещения, за верные ответы подряд и так далее). Отслеживая статистику других пользователей, повышается мотивация изучения и прохождения уровней дальше.

### 1.2.2 Веб-приложение Poliglot16.ru

*Poliglot16*[3] — самоучитель английского языка с бесплатными уроками [2]. Приложение имеет следующие возможности:

- практические и тестовые задания по грамматике;
- изучение слов по тематическим словарям;
- добавление собственных словарей;
- сохранение данных приложения для переноса;
- настройка режимов обучения.

Интерфейс веб-приложения представлен на рисунке 1.2.

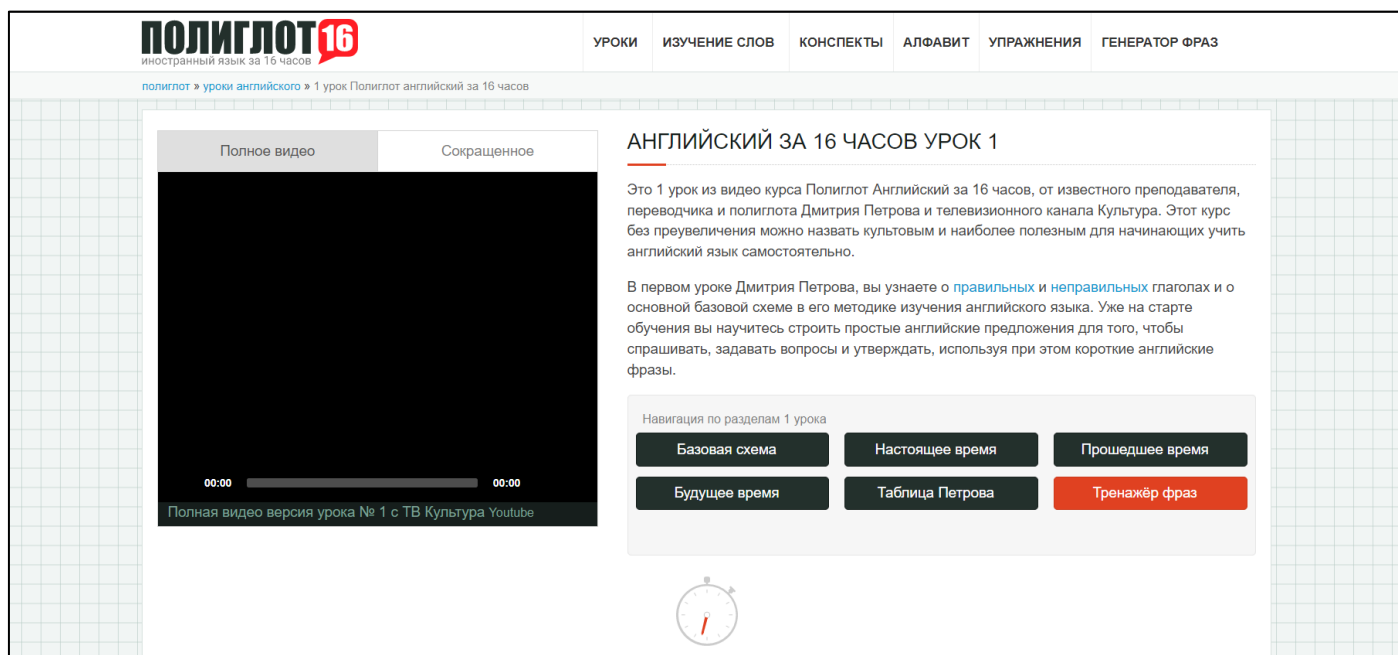


Рисунок 1.2 – Веб-приложение poliglot16.ru

Довольно устаревший дизайн и не обновляемая учебная база, что делает веб-приложение быстро выходящим из актуальности.

### 1.2.3 Веб-приложение Puzzel-english.com

Puzzel-english.com[4] – это приложение так же было разработано под мобильные операционные системы *IOS* и *Android* но имеет и веб-версию. Имеет следующие возможности:

- уроки по грамматическим правилам и расширение словарного запаса;
- конспект с грамматическими правилами;
- тематические истории в процессе прохождения, которых хорошо запоминаются слова за счёт образования ассоциаций;
- видео-уроки с текстовым сопровождением;
- возможность установить напоминание о занятии на любое время.

Интерфейс веб-приложения представлен на рисунке 1.3.

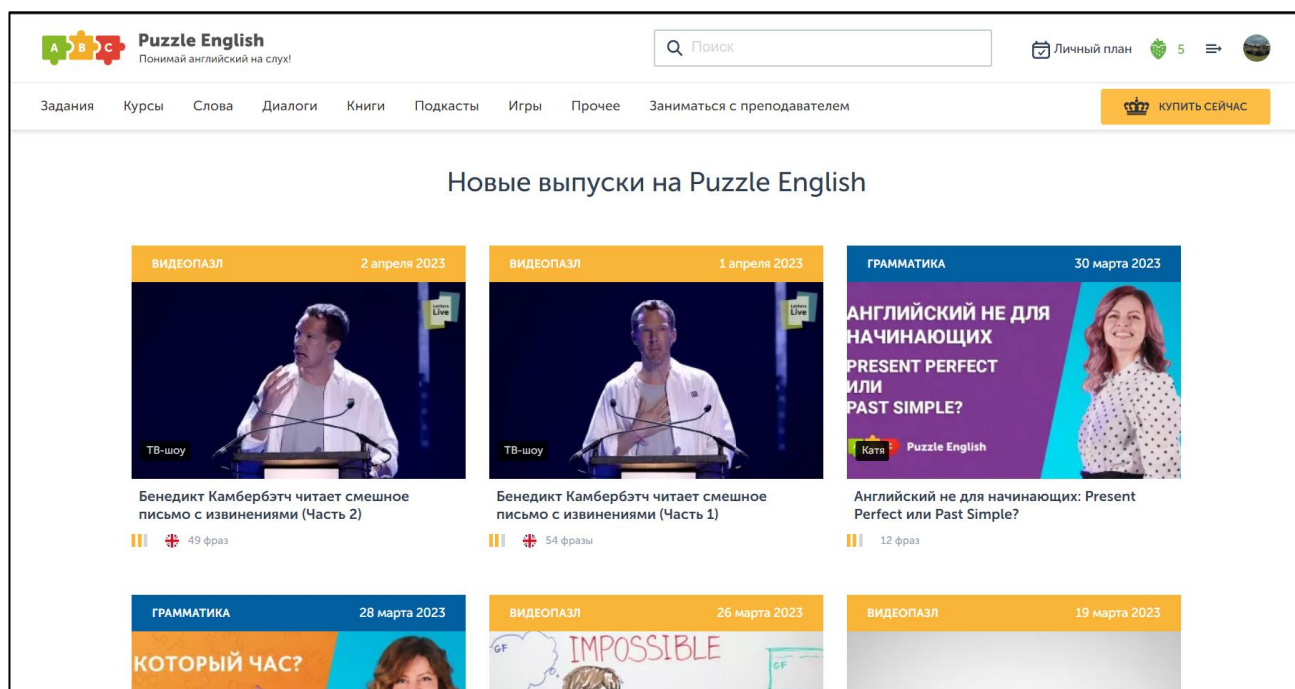


Рисунок 1.3 – Веб-приложение *puzzle-english.com*

К недостаткам данного веб-приложения можно отнести:

- основная часть контента является платной;
- невозможность изучать грамматику и лексику отдельно;
- довольно короткие уроки, за которые сложно усвоить тему.

### 1.3 Анализ прототипов

Схожесть рассмотренных программных средств заключается в том, что все они имеют как грамматические задания, так и задания на расширение лексикона. Также эти приложения содержат контент, который позволяет получить лишь начальные знания языка.

Главные отличия программ кроются в их функциональности. Так, например, в приложениях *Duolingo* и *Puzzle-English* изучение грамматики и словарного запаса проходит в модульной форме, в отличие от приложения *Poliglot16*. Только в приложении *Duolingo* имеется статистика прогресса, в приложении *Poliglot16* возможность добавлять собственные словари, а в приложении *Puzzle-English* есть грамматические конспекты. Также приложение *Puzzle-English* основную часть своего контента предлагает на платной основе.

Таким образом, можно сделать вывод о том, что в каждом приложении есть свои плюсы и минусы. Но, разумеется, каждому разработчику хочется, чтобы его приложение не имело отрицательных качеств, так как это сильно влияет рейтинг приложения, от чего напрямую зависит прибыль и статус самого разработчика.

### 1.4 Постановка задачи

Анализ веб-приложений, в основном, сводится к удобству пользовательского интерфейса, а также реализации всех функций необходимых пользователю.

В данном разделе был произведен обзор аналогов приложений для самостоятельного изучения испанского языка, рассмотрены их плюсы и минусы. Каждый аналог был подробно описан, и было показано, для чего предназначено то или иное программное обеспечение.

Исходя из сделанных выводов при анализе аналоговых приложений было решено реализовать следующие ключевые функциональные возможности:

- регистрация и авторизация пользователя;
- реализация ролей Гостя, Пользователя и Администратора;
- тренировка грамматики и расширение словарного запаса;
- база грамматических заданий и словарей;
- настройки обучения;
- статистика пользователя.

Программа должна быть предназначена для различной аудитории пользователей. Это значит, что приложение должно быть простое и иметь доступный дизайн.

Все эти пункты были учтены при выполнении курсового проектирования.

## 2 Проектирование веб-приложения

### 2.1 Диаграмма использования

Приложение поддерживает 3 роли: «Гость» «Пользователь» и «Администратор». Диаграмма вариантов использования приложения указана на рисунке 2.1

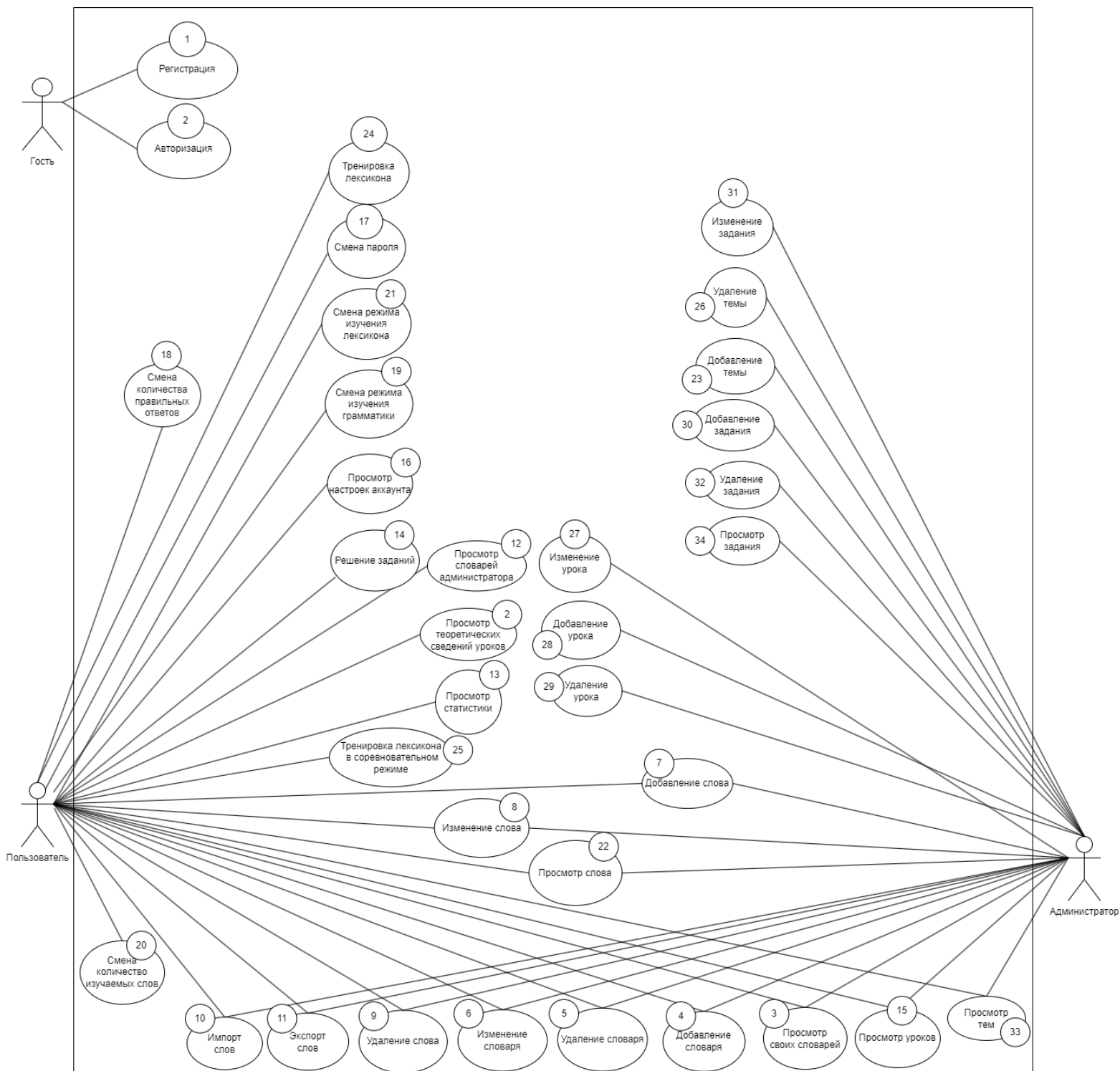


Рисунок 2.1 - Диаграмма вариантов использования



Диаграммы вариантов использования точно определяет функциональность веб-приложения, а также какие возможности будут доступны клиенту с определенной ролью.

Список ролей и их предназначение определены в таблице 2.1

Таблица 2.1 – назначение ролей пользователей

| Роль          | Назначение  |
|---------------|---|
| Гость         | Регистрация, авторизация  |
| Пользователь  | Просмотр своих словарей, просмотр словарей Администратора, добавление своего словаря, изменение своего словаря, удаление своего словаря, импорт своих слов, экспорт своих слов, просмотр слов своего словаря, удаление своих слов, изменение своих слов, импорт и экспорт слов, просмотр уроков, решение заданий урока, просмотр теоретических сведений урока, просмотр статистики, просмотр настроек аккаунта, смена пароля, смена режима изучения грамматики, смена количества изучаемых слов, смена режима изучения лексикона, Смена количества правильных ответов, тренировка грамматики. |
| Администратор | Просмотр своих словарей, добавление словаря, изменение словаря, удаление словаря, импорт и экспорт слов, просмотр слов словаря, удаление слов, изменение слов, просмотр уроков, добавление уроков, удаление уроков, изменение уроков, просмотр заданий урока, добавление заданий, изменение заданий, удаление заданий,  |

Функциональные возможности согласно диаграмме вариантов использования для ролей и их описание представлены в таблице 2.2

Таблица 2.2 – Функциональные возможности с описанием для каждой роли

| Номер | Вариант использования | Роли  | Описание  |
|-------|-----------------------|-------|---|
| 1     | Регистрироваться      | Гость | Возможность создать учетную запись для получения дополнительных возможностей. |
| 2     | Авторизация           | Гость | Возможность входить в учетную запись.   |

Продолжение таблицы 2.2

|    |                                  |                             |  |
|----|----------------------------------|-----------------------------|--|
| 3  | Просмотр своих словарей          | Пользователь, Администратор | Возможность просматривать словари.   |
| 4  | Добавление словаря               | Пользователь, Администратор | Возможность создавать словарь  |
| 5  | Удаление своего словаря          | Пользователь, Администратор | Возможность удалять словарь  |
| 6  | Изменение своего словаря         | Пользователь, Администратор | Возможность изменять свой словарь  |
| 7  | Добавление своего слова          | Пользователь, Администратор | Возможность добавления слова в свой словарь  |
| 8  | Изменение своего слова           | Пользователь, Администратор | Возможность изменять слово в своем словаре   |
| 9  | Удаление своего слова            | Пользователь, Администратор | Возможность удалять слово из своего словаря  |
| 10 | Импорт слов                      | Пользователь, Администратор | Возможность импортировать слова  |
| 11 | Экспорт слов                     | Пользователь, Администратор | Возможность экспортировать слова   |
| 12 | Просмотр словарей Администратора | Пользователь                | Возможность просмотреть словари созданные Администратором  |
| 13 | Просмотр статистики              | Пользователь                | Возможность просмотреть статистику по выученным словам, решенным заданиям, пройденных слов в квизе |
| 14 | Решение заданий                  | Пользователь                | Возможность решать грамматические задания  |
| 15 | Просмотр уроков                  | Пользователь, Администратор | Возможность просмотра всех доступных уроков  |
| 16 | Просмотр настроек аккаунта       | Пользователь                | Возможность просмотра персональных настроек аккаунта   |
| 17 | Смена пароля                     | Пользователь                | Возможность сменить пароль   |

Продолжение таблицы 2.2

| Номер | Вариант использования                        | Роли                        | Описание  |
|-------|--|-----------------------------|---|
| 18    | Смена количества правильных ответов          | Пользователь                | Возможность изменить количество правильных ответов, следующих подряд, после которого слово считается выученным и исключается из списка, но остаётся в словаре |
| 19    | Смена режима изучения грамматики             | Пользователь                | Возможность изменить способ изучения грамматики   |
| 20    | Смена количества изучаемых слов              | Пользователь                | Возможность изменить количество слов изучаемых одновременно   |
| 21    | Смена режима изучения лексики                | Пользователь                | Возможность изменить способ изучения лексики  |
| 22    | Просмотр слова                               | Пользователь, Администратор | Возможность просмотреть слова в словаре   |
| 23    | Добавление темы                              | Администратор               | Возможность добавлять новую тему  |
| 24    | Тренировка лексики                           | Пользователь                | Возможность переводить случайные слова  |
| 25    | Тренировка лексики в соревновательном режиме | Пользователь                | Возможность переводить случайные слова на скорость  |
| 26    | Удаление темы                                | Администратор               | Возможность удалять существующую тему   |

Продолжение таблицы 2.2

| Номер | Вариант использования | Роли                        | Описание                                    |
|-------|-----------------------|-----------------------------|---|
| 27    | Изменение урока       | Администратор               | Возможность добавлять новый урок            |
| 28    | Добавление урока      | Администратор               | Возможность редактировать существующий урок |
| 29    | Удаление урока        | Администратор               | Возможность удалять урок                    |
| 30    | Добавление задания    | Администратор               | Возможность добавлять задание к уроку       |
| 31    | Изменение задания     | Администратор               | Возможность изменять задание                |
| 32    | Удаление задания      | Администратор               | Возможность удалять задание из урока        |
| 33    | Просмотр тем          | Администратор, Пользователь | Возможность просматривать список тем        |
| 34    | Просмотр заданий      | Администратор               | Возможность смотреть задания уроков         |

## 2.2 Логическая схема базы данных

Диаграмма базы данных таблиц (*Database Table Diagram*) – это визуальное представление структуры базы данных и отношений между таблицами, которые хранятся в этой базе данных. Диаграмма базы данных представлена на рисунке 2.2.

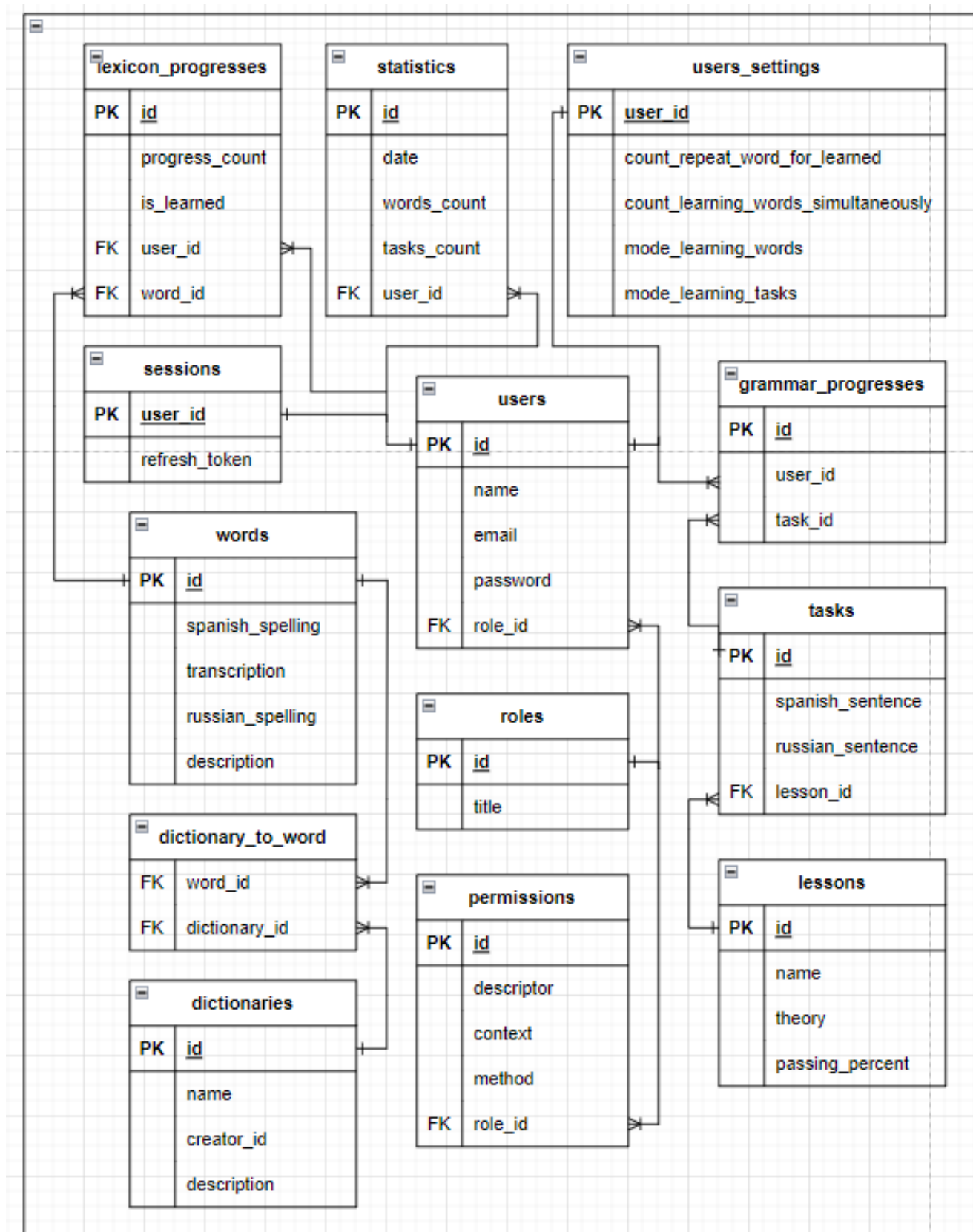


Рисунок 2.2 – Логическая схема базы данных

Таблица dictionaries содержит словари. Перечень полей таблицы dictionaries приведен в таблице 2.3.

Таблица 2.3 – Описание полей таблицы dictionaries

| Поле        | Тип                        | Назначение   |
|-------------|----------------------------|--|
| id          | int, identity, primary key | Идентификатор словаря. Является первичным ключом таблицы |
| name        | varchar(30), not null      | Название словаря   |
| description | varchar(255)               | Краткое описание словаря                                 |
| creator_id  | int, not null              | Идентификатор создателя словаря                          |

Таблица dictionary\_to\_word служит для связи таблиц dictionaries и words отношением многие ко многим. Перечень полей таблицы dictionary\_to\_word приведен в таблице 2.4.

Таблица 2.4 – Описание полей таблицы dictionary\_to\_word

| Поле          | Тип                        | Назначение   |
|---------------|----------------------------|--|
| dictionary_id | int, not null, foreign key | Внешний ключ, ссылающийся на идентификатор (id) таблицы dictionaries |
| word_id       | int, not null, foreign key | внешний ключ, ссылающийся на идентификатор (id) таблицы words        |

Таблица grammar\_progresses служит для связи таблиц users и tasks отношением многие ко многим и каждая запись в таблице информирует о том, что определённый пользователь успешно выполнил определённое задание.

Перечень полей таблицы grammar\_progresses приведен в таблице 2.5.

Таблица 2.5 – Описание полей таблицы grammar\_progresses

| Поле    | Тип                        | Назначение  |
|---------|----------------------------|---|
| id      | int, identity, primary key | Идентификатор связи, является первичным ключом таблицы        |
| task_id | int, not null, foreign key | Внешний ключ, ссылающийся на идентификатор (id) таблицы tasks |
| user_id | int not null, foreign key  | Внешний ключ, ссылающийся на идентификатор (id) таблицы users |

Таблица lessons хранит информацию о каждом уроке и имеет связь с таблицей tasks один ко многим, так как в одном уроке может быть несколько заданий.

Перечень полей таблицы lessons приведен в таблице 2.5.

Таблица 2.5 – Описание полей таблицы lessons

| Поле            | Тип                        | Назначение   |
|-----------------|----------------------------|--|
| id              | int, identity, primary key | Идентификатор урока, является первичным ключом таблицы   |
| name            | varchar(30), not null      | Название урока   |
| theory          | varchar(max), not null     | Теоретические сведения по теме урока   |
| passing_percent | int, not null              | Минимальный процент правильно выполненных заданий урока, для того, чтобы тема урока считалась достаточно изученной |

Таблица *lexicon\_progresses* служит для связи таблиц *users* и *words* отношением многие ко многим и каждая запись в таблице хранит информацию о прогрессе пользователя в изучении слов.

Перечень полей таблицы *lexicon\_progresses* приведен в таблице 2.6.

Таблица 2.7 – Описание полей таблицы *lexicon\_progresses*

| Поле           | Тип                        | Назначение  |
|----------------|----------------------------|---|
| id             | int, identity, primary key | Идентификатор связи, является первичным ключом таблицы                |
| progress_count | int, not null              | Количество правильных переводов слова пользователем, сделанных подряд |
| is_learned     | Boolean, not null          | Указывает, выучено слово ( <i>true</i> ) или нет ( <i>false</i> )     |
| user_id        | int, not null, foreign key | Внешний ключ, ссылающийся на идентификатор (id) таблицы <i>users</i>  |
| word_id        | int, not null, foreign key | Внешний ключ, ссылающийся на идентификатор (id) таблицы <i>words</i>  |

Таблица *permissions* хранит разрешения ролей для авторизации.

Перечень полей таблицы *permissions* приведен в таблице 2.7.

Таблица 2.7 – Описание полей таблицы *permissions*

| Поле       | Тип                        | Назначение   |
|------------|----------------------------|--|
| id         | int, identity, primary key | Идентификатор разрешения, является первичным ключом таблицы          |
| context    | varchar(255)               | Контекст выполняемого запроса  |
| method     | varchar(255), not null     | Метод http запроса   |
| role_id    | int, not null, foreign key | Внешний ключ, ссылающийся на идентификатор (id) таблицы <i>roles</i> |
| descriptor | int, not null              | Наименование контроллера   |

Таблица *roles* хранит данные о каждой роли.

Перечень полей таблицы *roles* приведен в таблице 2.8.

Таблица 2.9 – Описание полей таблицы *roles*

| Поле  | Тип                        | Назначение  |
|-------|----------------------------|---|
| id    | int, identity, primary key | Идентификатор роли, является первичным ключом таблицы |
| title | varchar(255), not null     | Название роли   |

Таблица *sessions* хранит данные о сессиях пользователей и связан с таблицей *users* связью один к одному.

Перечень полей таблицы *sessions* приведен в таблице 2.9.

Таблица 2.9 – Описание полей таблицы *sessions*

| Поле          | Тип                                 | Назначение  |
|---------------|-------------------------------------|---|
| user_id       | int, unique, foreign key            | Внешний ключ, ссылающийся на идентификатор (id) таблицы users |
| refresh_token | varchar(max), not null, foreign key | <i>refresh</i> -токен пользователя                            |

Таблица *statistics* хранит статистические данные об обучении пользователей за каждый день: количество правильных переводов слов, заданий и квиз очков.

Перечень полей таблицы *statistics* приведен в таблице 2.10.

Таблица 2.10 – Описание полей таблицы *statistics*

| Поле        | Тип                        | Назначение  |
|-------------|----------------------------|---|
| id          | int, identity, primary key | Идентификатор статистики, является первичным ключом таблицы                         |
| date        | Date, not null             | Дата создания записи  |
| words       | int                        | Количество правильных переводов слова пользователем                                 |
| tasks       | int                        | Количество правильно сделанных заданий пользователем                                |
| quiz_points | int                        | Количество квиз очков, полученных пользователем за правильные переводы слов в квизе |
| user_id     | int, not null, foreign key | Внешний ключ, ссылающийся на идентификатор (id) таблицы roles                       |

Таблица *tasks* хранит задания уроков.

Перечень полей таблицы *tasks* приведен в таблице 2.11.

Таблица 2.11 – Описание полей таблицы *tasks*

| Поле             | Тип                      | Назначение   |
|------------------|--------------------------|--|
| id               | int unique foreign key   | Идентификатор задания, является первичным ключом таблицы       |
| spanish_sentence | varchar(300) not null    | Задание, написанное на испанском языке                         |
| russian_sentence | varchar(300) not null    | Перевод задания из поля spanish_sentence на русский язык       |
| lesson_id        | int not null foreign key | Внешний ключ, ссылающийся на идентификатор (id) таблицы lesson |

Таблица *users* хранит данные пользователей.

Перечень полей таблицы *users* приведен в таблице 2.12.

Таблица 2.12 – Описание полей таблицы *users*

| Поле  | Тип                      | Назначение  |
|-------|--------------------------|---|
| id    | int, unique, primary key | Идентификатор пользователя, является первичным ключом таблицы |
| name  | varchar(255), not null   | Имя пользователя  |
| email | varchar(255), not null   | Электронная почта пользователя                                |



Продолжение таблицы 2.12

| Поле             | Тип                        | Назначение  |
|------------------|----------------------------|---|
| password         | varchar(255), not null     | Захэшированный пароль пользователя                            |
| role_id          | int, not null, foreign key | Внешний ключ, ссылающийся на идентификатор (id) таблицы roles |
| id_enable_lesson | int, not null              | Идентификатор (id) урока, доступного пользователю             |

Таблица *user\_settings* хранит настройки обучения пользователей.  
Перечень полей таблицы *user\_settings* приведен в таблице 2.13.

Таблица 2.13 – Описание полей таблицы *user\_settings*

| Поле                                | Тип                    | Назначение  |
|-------------------------------------|------------------------|---|
| user_id                             | int unique foreign key | Внешний ключ, ссылающийся на идентификатор (id) таблицы users         |
| count_repeat_word_for_learned       | int not null           | Количество правильных ответов подряд, чтобы слово считалось выученным |
| count_learning_words_simultaneously | int not null           | Количество слов изучаемых одновременно                                |
| mode_learning_words                 | varchar(255) not null  | Способ изучения лексики   |
| mode_learning_tasks                 | varchar(255) not null  | Способ изучения грамматики  |

Таблица *words* хранит слова, добавленные администратором и пользователями.

Перечень полей таблицы *words* приведен в таблице 2.14

Таблица 2.14 – Описание полей таблицы *words*

| Поле             | Тип                    | Назначение   |
|------------------|------------------------|--|
| id               | int unique foreign key | Идентификатор слова, является первичным ключом таблицы |
| spanish_spelling | int not null           | Испанское написание слова                              |
| transcription    | int not null           | Транскрипция слова на испанском                        |
| russian_spelling | varchar(255) not null  | Русский перевод слова из поля spanish_spelling         |
| description      | varchar(255) not null  | Описание слова (для уточнения перевода)                |

Таблица *topics* хранит темы уроков.

Перечень полей таблицы *topics* приведен в таблице 2.15

Таблица 2.15 – Описание полей таблицы *topics*

| Поле | Тип                    | Назначение  |
|------|------------------------|---|
| id   | int unique foreign key | Идентификатор темы, является первичным ключом таблицы |
| name | varchar(255) not null  | Название темы для уроков                              |

Таким образом, каждая таблица содержит определённые поля, связывающие

одну таблицу с другой для эффективного управления данными.

## 2.3 Архитектура веб-приложения

Архитектура веб-приложения включает клиентскую часть для взаимодействия с пользователем, сервер для обработки запросов и базу данных для хранения информации. Эти компоненты обмениваются данными через сеть, обеспечивая работу системы.

Архитектура приложения детально представлена на рисунке 2.3

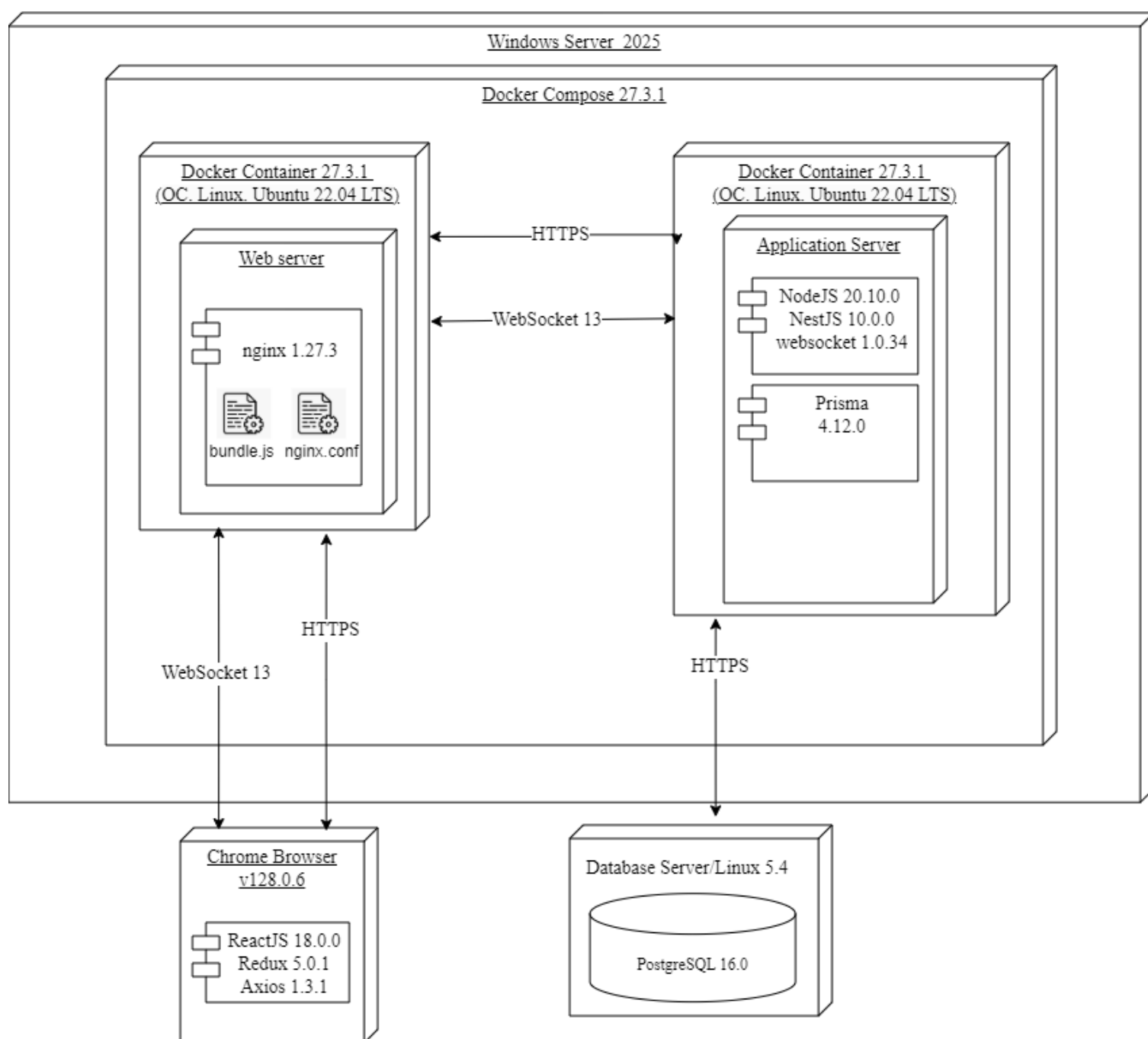


Рисунок 2.3 – Архитектура web-приложения

В таблице 2.16 подробно описаны основные элементы системы, их назначение и роль в работе приложения.

Таблица 2.16 – Назначение элементов архитектурной схемы веб-приложения

| Элемент  | Назначение   |
|--|--|
| <i>Web Server</i><br>( <i>nginx</i> [5])           | Предоставляет доступ к статическим ресурсам пользовательского интерфейса веб-приложения.                               |
| <i>Database Server</i><br>( <i>PostgreSQL</i> [6]) | Используется для хранения, управления и предоставления доступа к данным, которые необходимы для работы веб-приложения. |
| <i>Application Server</i>                          | Обрабатывать запросы пользователя, запрашивать данные из базы данных.  |

Описание протоколов, используемых при работе веб-приложений, представлено в таблице 2.17.

Таблица 2.17 – Описание используемых протоколов.

| Протокол      | Назначение  |
|---------------|---|
| HTTP[7]       | Основной протокол для обмена данными.   |
| HTTPS[8]      | Обмен данными между Web Server и Application Server, Web Server и Chrome Browser. Обеспечить безопасную передачу данных путём использования криптографического протокола TLS. Используется транспортный протокол TCP[9] |
| WebSocket[10] | Обмен данными между Web Server и Application Server, Web Server и Chrome Browser. Обеспечивать постоянное соединения используя TCP-соединение. <i>Sec-WebSocketVersion</i> 13   |

Слаженная работа всех элементов архитектуры обеспечивает надёжную работу веб-приложения.

## 2.4 Вывод по разделу

1. Рассмотрены три роли веб-приложения: гость, пользователь и администратор, двадцать две функциональные возможности с их описанием.

2. Определена структура базы данных, она включает четырнадцать таблиц. Описано содержание таблиц, их связи между друг другом.

## 3 Реализация веб-приложения

### 3.1 Программная платформа Node.js

Для серверной части проекта выбрали Node.js — высокопроизводительную платформу с событийной архитектурой, позволяющей обрабатывать множество запросов одновременно. Основным фреймворком стал Nest.js[11], который использует TypeScript[12] для статической проверки типов, упрощённой отладки и повышения читаемости кода. TypeScript обеспечивает совместимость с любыми средами JavaScript[13], включая Node.js, и поддерживает современные стандарты ECMAScript[14].

### 3.2 СУБД PostgreSQL

Для проекта выбрали PostgreSQL — мощную реляционную СУБД, известную надёжностью, расширяемостью и активным сообществом. Она обеспечивает хранение данных, поддержку ACID-транзакций и эффективную обработку сложных запросов, что делает её отличным выбором для систем с большими объёмами данных и сложной бизнес-логикой.

### 3.3 Средство моделирования объектного документа

*Prisma*[15] — инструмент для работы с реляционными базами данных в проектах на TypeScript и JavaScript. Он реализует ORM нового поколения, упрощая взаимодействие с базой данных через Prisma Schema и минимизируя необходимость в сложных SQL-запросах. Интеграция с TypeScript обеспечивает автодополнение, проверку типов и безопасную работу с данными. Также Prisma поддерживает миграции схемы, упрощая управление структурой данных и командную работу.

Соответствие моделей Prisma и СУБД *PostgreSQL* представлено в таблице 3.1.

Таблица 3.1 – Сопоставление моделей, используемых в Prisma

| Prisma название модели таблицы | <i>PostgreSQL</i> название таблицы |
|--------------------------------|------------------------------------|
| User                           | user                               |
| UserSettings                   | usersettings                       |
| Topic                          | topic                              |
| User                           | user                               |
| Lesson                         | lesson                             |
| Task                           | task                               |
| GrammarProgress                | grammarprogress                    |
| Dictionary                     | dictionary                         |
| Word                           | word                               |
| DictionaryToWord               | dictionarytoword                   |

Продолжение таблицы 3.1

|                                |                             |
|--------------------------------|-----------------------------|
| Prisma название модели таблицы | PostgreSQL название таблицы |
| LexiconProgress                | lexiconprogress             |
| Statistics                     | statistics                  |
| Session                        | session                     |
| Role                           | role                        |
| Permission                     | permission                  |

Исходный код моделей на Prisma представлен в приложении А.

### 3.4 Библиотеки и фреймворки

В процессе разработки серверной части веб-приложения для обеспечения её функциональности и повышения эффективности работы системы были использованы программные библиотеки, представленные в таблице 3.2.

Таблица 3.2 – Программные библиотеки серверной части

| Название              | Версия | Описание  |
|-----------------------|--------|---|
| @nestjs[16]           | 9.0.0  | Общие модули и утилиты фреймворка <i>NestJS</i> для построения приложений.                |
| @prisma/client[17]    | 4.11.0 | Генерируемый клиент для работы с базой данных, использующей <i>Prisma ORM</i> .           |
| bcrypt[18]            | 5.1.0  | Библиотека для хеширования паролей с использованием алгоритма <i>bcrypt</i> .             |
| class-transformer[19] | 0.5.1  | Инструмент для преобразования объектов, например, при использовании DTO в приложении.     |
| class-validator[20]   | 0.14.0 | Библиотека для валидации данных на основе аннотаций в <i>TypeScript</i> -классах.         |
| cookie[21]            | 0.5.0  | Библиотека для обработки и работы с <i>HTTP</i> -куками.                                  |
| cookie-parser[22]     | 1.4.6  | <i>Middleware</i> для анализа <i>cookie</i> в <i>HTTP</i> -запросах.                      |
| cors[23]              | 2.8.5  | <i>Middleware</i> для настройки <i>Cross-Origin Resource Sharing (CORS)</i> в приложении. |
| helmet[24]            | 7.0.0  | <i>Middleware</i> для повышения безопасности <i>HTTP</i> -запросов в приложении.          |
| https[25]             | 1.0.0  | Модуль для создания <i>HTTPS</i> -серверов и работы с защищёнными соединениями.           |
| joi[26]               | 17.9.0 | Библиотека для валидации данных, удобная для настройки схем проверки.                     |
| nodemailer[27]        | 6.7.8  | Библиотека для отправки электронных писем.  |

Продолжение таблицы 3.2

| Название                     | Версия | Описание  |
|------------------------------|--------|---|
| <i>passport</i> [28]         | 0.6.0  | Основная библиотека <i>Passport</i> для аутентификации. |
| <i>passport-jwt</i> [29]     | 4.0.1  | <i>Passport</i> -стратегия для работы с <i>JWT</i> .    |
| <i>reflect-metadata</i> [30] | 0.1.13 | Библиотека для работы с метаданными                     |
| <i>rxjs</i> [31]             | 7.8.1  | Библиотека для работы с реактивными потоками данных.    |
| <i>websocket</i> [32]        | 1.0.34 | Библиотека для работы с <i>WebSocket</i> -соединениями. |

В процессе разработки клиентской части веб-приложения были задействованы программные библиотеки, представленные в таблице 3.3.

Таблица 3.3 – Программные библиотеки клиентской части

| Название                                   | Версия | Описание  |
|--|--------|---|
| <i>@fortawesome</i> [33]                   | 6.3.0  | библиотека <i>FontAwesome</i> для работы с <i>SVG</i> -иконками.                        |
| <i>@fortawesome/react-fontawesome</i> [34] | 0.2.0  | Компоненты для интеграции <i>FontAwesome</i> с <i>React</i> -приложениями.              |
| <i>@testing-library/jest-dom</i> [35]      | 5.16.4 | Библиотека для тестирования <i>React</i> -компонентов с использованием <i>Jest</i>      |
| <i>@testing-library/react</i> [36]         | 13.5.0 | Библиотека для эмуляции пользовательских событий в тестах.                              |
| <i>axios</i> [37]                          | 1.3.3  | HTTP-клиент для отправки запросов к серверу.  |
| <i>bootstrap</i> [38]                      | 5.2.3  | CSS[39]-фреймворк для создания адаптивных и стилизованных пользовательских интерфейсов. |
| <i>chart.js</i> [39]                       | 4.3.0  | Библиотека для визуализации данных с использованием интерактивных графиков и диаграмм.  |
| <i>date-fns</i> [40]                       | 2.29.3 | Библиотека для работы с датами в JavaScript.  |
| <i>dotenv</i> [41]                         | 16.0.3 | Загрузка переменных окружения из файла <i>.env</i>                                      |
| <i>js-cookie</i> [42]                      | 3.0.1  | Библиотека для работы с HTTP-куками в браузере.   |
| <i>jwt-decode</i> [43]                     | 3.1.2  | Утилита для декодирования <i>JWT</i> -токенов.  |
| <i>mobx</i> [44]                           | 6.0.5  | Библиотека для управления состоянием в приложениях.                                     |
| <i>primeicons</i> [45]                     | 6.0.1  | Иконки для библиотеки компонентов <i>PrimeReact</i> .                                   |
| <i>primereact</i> [46]                     | 9.2.3  | Библиотека компонентов пользовательского интерфейса для <i>React</i> .                  |

Продолжение таблицы 3.3

|                              |        |   |
|------------------------------|--------|---|
| <i>react</i> [47]            | 18.0.0 | JavaScript-библиотека для создания пользовательских интерфейсов.                  |
| <i>react-bootstrap</i> [48]  | 2.7.4  | Реализация компонентов Bootstrap для React.                                       |
| <i>react-date-range</i> [49] | 1.4.0  | Компонент для выбора диапазона дат в React-приложениях.                           |
| <i>react-dom</i> [50]        | 18.2.0 | Пакет для рендеринга React-компонентов в DOM.                                     |
| <i>react-router-dom</i> [51] | 6.10.0 | Библиотека для работы с маршрутизацией в React-приложениях.                       |
| <i>react-scripts</i> [52]    | 5.0.1  | Скрипты и конфигурация для работы с приложениями на Create React App.             |
| <i>rxjs</i> [53]             | 7.8.1  | Библиотека для работы с реактивными потоками данных.                              |
| <i>socket.io-client</i> [54] | 4.6.1  | Клиентская библиотека для работы с WebSocket через Socket.IO.                     |
| <i>web-vitals</i> [55]       | 2.1.4  | Библиотека для измерения и анализа показателей производительности веб-приложений. |
| <i>ws</i> [56]               | 8.13.0 | Библиотека для работы с WebSocket-соединениями.                                   |

### 3.4 Структура серверной части

Разработка серверной части веб-приложения выполнена с использованием модульно-слоистой архитектуры (Modular Layered Architecture) [57], которая обеспечивает чёткое разделение ответственности между компонентами системы. Такой подход упрощает масштабирование проекта, его поддержку и дальнейшее расширение. В архитектуре приложения выделяются следующие основные слои:

1. Контроллеры (Controllers): обрабатывают входящие HTTP-запросы, управляют входящими данными, вызывают бизнес-логику из сервисов и возвращают ответ клиенту.

2. Сервисы (Services): реализуют бизнес-логику приложения и обращаются к моделям для взаимодействия с базой данных.

3. Репозитории (Repository): инкапсулирует операции взаимодействия с базой данных. Модели данных, определённые с использованием Prisma, предоставляют абстракцию для доступа к таблицам.

4. Модули (Modules): они определяют зависимости, провайдеры и экспортируемые компоненты.

Файлы проекта организованы таким образом, чтобы они были сгруппированы в директории по их назначению. Так, каждая директория отвечает за определённый слой или функционал приложения. В таблице 3.4 приведён список директорий проекта разработки приложения и назначение файлов, хранящихся в этих директориях.

Таблица 3.4 – Директории проекта и их назначение

| Директория        | Назначение файлов директории   |
|-------------------|--|
| src               | Главная рабочая директория проекта, содержит основную бизнес-логику приложения.                            |
| src/modules       | Файлы модулей, отвечающие за обработку HTTP-запросов, взаимодействие с сервисами и возврат ответов клиенту |
| src/configuration | Содержит файлы конфигурации приложения   |
| src/helpers       | Хранит вспомогательные модули сторонних сервисов   |
| prisma            | Содержит файлы, связанные с ORM Prisma, служащие для описания схемы базы данных, файлы миграций            |

В каждой директории содержатся файлы, выполняющие строго определённые задачи. В таблице 3.5 приведен список файлов директории *modules* на примере *dictionary*.

Таблица 3.5 – список файлов директории *dictionary* в *modules*

| Директория               | Назначение файлов директории  |
|--------------------------|---|
| dto                      | Здесь хранятся Data Transfer Objects, которые определяют структуру данных, передаваемых между клиентом и сервером       |
| response                 | Папка с типами или интерфейсами для описания структуры ответов API, связанных со словарём                               |
| dictionary.controller.ts | Контроллер для обработки запросов, связанных со словарём (например, добавление, обновление или получение слов)          |
| dictionary.module.ts     | Файл, который связывает все элементы модуля. В NestJS он определяет зависимости, провайдеры и экспортируемые компоненты |
| dictionary.repository.ts | файл для работы с базой данных. Обычно отвечает за взаимодействие с таблицей или коллекцией, связанной со словарём      |
| dictionary.service.ts    | Содержит бизнес-логику для работы с данными словаря. Используется контроллером и может обращаться к repository          |

Пример реализации модуля предоставлен в Приложениях Г по Приложение Е. Всего разработано 11 модулей, каждый из которых методы, обрабатывающие входящие запросы. Для взаимодействия с частью методов необходима авторизация:

- *auth* для регистрации и авторизации пользователей;
- *dictionary* для работы со словарями;
- *grammar-progress* для работы со статистикой по заданиям;
- *lesson* для работы с уроками;
- *lexicon-progress* для работы со статистикой по словам;
- *session* для работы с *refresh*-токеном пользователей;
- *statistics* для работы со статистикой пользователей;
- *task* для работы с заданиями;



- *topic* для работы с темами уроков;
- *user* для смены пароля аккаунта и получения данных пользователей;
- *user-settings* для работы с настройками аккаунта пользователя;
- *word* для работы со словами.

Всего в проекте 33 *API* метода. Описание доступных модулей, адресов и методов приведено в таблице 3.6.

Таблица 3.6 – Описание методов *API*

| Адрес                                    | Метод  | Описание   | Модуль                  |
|--|--------|--|-------------------------|
| /api/user/{id}                           | GET    | Получение данных пользователя по его <i>id</i>                       | <i>user</i>             |
| /api/user                                | PATCH  | Обновление пароля пользователя                                       | <i>user</i>             |
| /api/auth/registration                   | POST   | Регистрация нового пользователя                                      | <i>auth</i>             |
| /api/auth/login                          | POST   | Вход в аккаунт   | <i>auth</i>             |
| /api/auth/refresh                        | POST   | Обновление токенов по <i>refresh</i> -токену                         | <i>auth</i>             |
| /api/auth/logout                         | POST   | Выход из аккаунта  | <i>auth</i>             |
| /api/dictionary/                         | POST   | Создание словаря   | <i>dictionary</i>       |
| /api/dictionary/admin                    | GET    | Получение списка словарей, созданных администратором                 | <i>dictionary</i>       |
| /api/dictionary/user                     | GET    | Получение списка словарей, созданных пользователем                   | <i>dictionary</i>       |
| /api/dictionary/review                   | GET    | Получение списка словарей, созданных пользователем и администратором | <i>dictionary</i>       |
| /api/dictionary/learn                    | GET    | Получение списка словарей для обучения                               | <i>dictionary</i>       |
| /api/dictionary/{id}                     | PATCH  | Обновление словаря по <i>id</i>                                      | <i>dictionary</i>       |
| /api/dictionary/{id}                     | DELETE | Удаление словаря по <i>id</i>  | <i>dictionary</i>       |
| /api/word/{id}                           | POST   | Создание слова   | <i>word</i>             |
| /api/word/import/{id}                    | POST   | Импорт слов в словарь по <i>id</i> словаря                           | <i>word</i>             |
| /api/word/{id}                           | PATCH  | Обновление слова по <i>id</i>  | <i>word</i>             |
| /api/word/{id}/dictionary/{dictionaryId} | DELETE | Удаление слова из словаря по <i>id</i> слова и <i>id</i> словаря     | <i>word</i>             |
| /api/lesson                              | POST   | Создание урока   | <i>lesson</i>           |
| /api/lesson/admin                        | GET    | Получение всех уроков  | <i>lesson</i>           |
| /api/lesson/learn                        | GET    | Получение доступных уроков для изучения пользователем                | <i>lesson</i>           |
| /api/lesson/{id}                         | PATCH  | Обновление урока по <i>id</i>  | <i>lesson</i>           |
| /api/lesson/{id}                         | DELETE | Удаление урока по <i>id</i>  | <i>lesson</i>           |
| /api/grammar-progress                    | POST   | Создание записи изученного урока                                     | <i>grammar-progress</i> |
| /api/task                                | POST   | Создать задания  | <i>task</i>             |
| /api/task/{id}                           | PATCH  | Обновление задания по <i>id</i>                                      | <i>task</i>             |
| /api/task/{id}                           | DELETE | Удаление задания по <i>id</i>  | <i>task</i>             |
| /api/statistics                          | POST   | Создание статистики  | <i>statistics</i>       |
| /api/statistics/all                      | GET    | Получение всех статистик   | <i>statistics</i>       |
| /api/statistics/user-all                 | GET    | Получение всех статистик пользователя                                | <i>statistics</i>       |
| /api/statistics/user-to-day              | GET    | Получение статистики пользователя за текущий день                    | <i>statistics</i>       |
| /api/user-settings                       | GET    | Получение настроек пользователя                                      | <i>user-settings</i>    |
| /api/user-settings                       | PATCH  | Обновление настроек пользователя                                     | <i>user-settings</i>    |
| /api/lexicon-progress                    | POST   | Создание записи изученного слова                                     | <i>lexicon-progress</i> |

### 3.8 Структура клиентской части

Клиентская часть приложения реализована с использованием компонентного подхода. Основная логика и элементы пользовательского интерфейса размещены в директории `src`. Директории представлены в таблице 3.7

Таблица 3.7 – Основные директории проекта в папке `src` их назначение

| Директория                | Назначение   |
|---------------------------|--|
| <code>api-requests</code> | Хранит файлы, связанные с запросами к API.   |
| <code>components</code>   | Содержит повторно используемые React-компоненты, такие как интерфейсные элементы (Input, NavBar), маршрутизаторы или другие строительные блоки приложения. |
| <code>pages</code>        | Хранит страницы приложения   |
| <code>stores</code>       | Используется для управления состоянием приложения  |
| <code>styles</code>       | Хранит CSS стили проекта   |
| <code>utils</code>        | Хранит вспомогательные утилиты и функции, которые используются в разных частях приложения  |

Таблица соответствия маршрутов и страниц представлена в таблице 3.8.

Таблица 3.8 – Маршруты и страницы

| Страница         | Маршрут                             | Роль                               | Назначение страницы   |
|------------------|-------------------------------------|------------------------------------|---|
| Dictionary       | <code>/api/dictionary</code>        | Администратор, гость, пользователь | Просмотр, добавление, изменение, удаление своих словарей. Просмотр, добавление, изменение, удаление своих слов.   |
| DictionaryLearn  | <code>/api/lexicon-progress</code>  | Пользователь                       | Выбор словаря для изучения, изучение лексикона, тренировка лексикона в соревновательном режиме  |
| DictionaryReview | <code>/api/dictionary/review</code> | Пользователь                       | Просмотр словаря и слов Администратора, экспорт словаря Администратора, сортировка словаря по названию или описанию. Сортировка слов по испанскому переводу, транскрипции, русскому переводу, описанию. |

Продолжение таблицы 3.8

| Страница        | Маршрут               | Роль                               | Назначение страницы  |
|-----------------|-----------------------|------------------------------------|--|
| Lesson          | /api/lesson/admin     | Администратор                      | Просмотр уроков, создание уроков, поиск уроков по названию, удаление уроков, изменение уроков, просмотр заданий урока, создание заданий, изменение заданий, удаление заданий, поиск задания по русскому и испанскому переводу. |
| LessonLearn     | /api/grammar-progress | Пользователь                       | Просмотр уроков, выполнение заданий урока  |
| Login           | /api/auth/login       | Гость                              | Авторизация Гостя  |
| NotFound        | /                     | Администратор, Пользователь, Гость | Страница для несуществующего маршрута  |
| PersonalCabinet | /api/                 | Пользователь                       | Страница пользователя с его настройками.   |
| Registration    | /api/registration     | Гость                              | Страница регистрации Гостя   |
| Statistics      | /api/statistics       | Пользователь                       | Страница для просмотра статистики пройденных материалов.   |

Помимо маршрутов и страниц, приложение включает множество компонентов, которые обеспечивают функциональность и удобство использования клиентской части.

В таблице 3.9 представлено описание всех остальных компонентов приложения и их назначение.

Таблица 3.9 – Описание компонентов

| Компонент | Назначение   |
|-----------|--|
| AppRouter | Этот компонент отвечает за маршрутизацию в приложении. Распределяет доступ к страницам в зависимости от роли пользователя. |
| Navbar    | Используется как навигационная панель по web-приложению.   |

|       |  |
|-------|--|
| Input | Универсальный входной элемент с валидацией, стилизованной ошибкой и гибкой настройкой. |
|-------|--|

### 3.9 Вывод по разделу

1. Определена программная платформа Node.js, СУБД PostgreSQL и объектно-реляционное отображение для реализации веб-приложения.
2. Определена структура серверной части, реализация функциональности для ролей, структура клиентской части.

## 4 Тестирование программного продукта

В этой главе рассмотрены и протестированы основные элементы интерфейса веб-приложения.

### 4.1 Функциональное тестирование для роли Пользователя

Таблица 4.1 – Описание тестирования функций Пользователя

| Название функции         | Описание тестирования  | Итог тестирования функции |
|--------------------------|--|---------------------------|
| Просмотр своих словарей  | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть»<br>Ожидаемый результат: отображение списка словарей  | Успешно.                  |
| Создание своих словарей  | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть», заполнить форму создания слова, нажать на кнопку «Добавить»<br>Ожидаемый результат: создание и отображение созданного словаря | Успешно.                  |
| Изменение своего словаря | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть», нажать на «карандаш», изменить данные в форме словаря<br>Ожидаемый результат: изменение словаря.                              | Успешно.                  |
| Удаление своего словаря  | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть», нажать на «мусорный бак» у словаря<br>Ожидаемый результат: удаление словаря.  | Успешно.                  |

Продолжение таблицы 4.1

| Название функции | Описание тестирования  | Итог тестирования функции |
|------------------|--|---------------------------|
| Просмотр слова   | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть», нажать на словарь<br>Ожидаемый результат: отображение слов  | Успешно.                  |
| Добавление слова | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть», нажать на словарь, заполнить форму добавления слова<br>Ожидаемый результат: добавление и отображение добавленного слова | Успешно.                  |
| Удаление слова   | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть», нажать на словарь, нажать на «мусорный бак» у слова<br>Ожидаемый результат: удаление слова                              | Успешно.                  |
| Изменение слова  | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть», нажать на словарь, нажать на «карандаш», изменить данные в форме слова<br>Ожидаемый результат: изменение слова          | Успешно.                  |
| Экспорт слов     | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть», нажать на «Экспорт» рядом со словарем<br>Ожидаемый результат: скачивание <i>JSON</i> файла со словами                   | Успешно.                  |

Продолжение таблицы 4.1

| Название функции                       | Описание тестирования  | Итог тестирования функции |
|--|--|---------------------------|
| Импорт слов                            | <p>Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Изменить» или «Просмотреть», нажать на «Импорт» рядом со словарем, выбрать подходящий <i>JSON</i> файл</p> <p>Ожидаемый результат: добавление слов с <i>JSON</i> файла.</p> | Успешно.                  |
| Просмотр настроек аккаунта             | <p>Действие: нажать на кнопку «Личный кабинет»</p> <p>Ожидаемый результат: переход на страницу настроек аккаунта</p>   | Успешно.                  |
| Смена пароля                           | <p>Действие: нажать на кнопку «Личный кабинет», ввести в форму старый пароль, ввести новый пароль, ввести новый пароль повторно, нажать кнопку «Сменить пароль»</p> <p>Ожидаемый результат: переход на страницу настроек аккаунта</p>                          | Успешно.                  |
| Смена режима изучения лексики          | <p>Действие: нажать на кнопку «Личный кабинет», в форме выбрать другой способ, нажать на кнопку «Сохранить»</p> <p>Ожидаемый результат: сохранение настроек в базе данных</p>  | Успешно.                  |
| Просмотр уроков                        | <p>Действие: нажать на кнопку «Уроки»,</p> <p>Ожидаемый результат: отображение уроков на странице</p>  | Успешно.                  |
| Просмотр теоретических сведений уроков | <p>Действие: нажать на кнопку «Уроки», нажать на нужный урок</p> <p>Ожидаемый результат: отображение теоретических сведений</p>  | Успешно.                  |

Продолжение таблицы 4.1

| Название функции                               | Описание тестирования  | Итог тестирования функции |
|--|--|---------------------------|
| Прохождение заданий                            | Действие: нажать на кнопку «Уроки», нажать на нужный урок, ответить на задания урока<br>Ожидаемый результат: задание засчитано   | Успешно.                  |
| Тренировка лексикона                           | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Учить», выбрать словарь, правильно перевести слово<br>Ожидаемый результат: слово решено успешно.   | Успешно.                  |
| Тренировка лексикона в соревновательном режиме | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Учить», правильно перевести слово на скорость, получить очки. В навигационной панели нажать на «Статистика»<br>Ожидаемый результат: увидеть как решили другие. | Успешно.                  |
| Просмотр статистики                            | В навигационной панели нажать на «Статистика»<br>Ожидаемый результат: увидеть статистику Пользователей.  | Успешно.                  |

## 4.2 Функциональное тестирование для роли Администратора

В таблице 4.2 представлено тестирование функций Администратора

Таблица 4.2 – Описание тестирования функций Администратора

| Название функции        | Описание тестирования   | Итог тестирования функции |
|-------------------------|---|---------------------------|
| Просмотр своих словарей | Действие: нажать на кнопку «Словари и слова» в навигационной панели<br>Ожидаемый результат: отображение списка словарей | Успешно.                  |



Продолжение таблицы 4.2

| Название функции         | Описание тестирования  | Итог тестирования функции |
|--------------------------|--|---------------------------|
| Создание своих словарей  | Действие: нажать на кнопку «Словари и слова» в навигационной панели, заполнить форму создания слова, нажать на кнопку «Добавить»<br>Ожидаемый результат: создание и отображение созданного словаря | Успешно.                  |
| Изменение своего словаря | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «карандаш», изменить данные в форме словаря<br>Ожидаемый результат: изменение словаря.                              | Успешно.                  |
| Удаление своего словаря  | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «мусорный бак» у словаря<br>Ожидаемый результат: удаление словаря.  | Успешно.                  |
| Просмотр слова           | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на словарь<br>Ожидаемый результат: отображение слов  | Успешно.                  |
| Добавление слова         | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на словарь, заполнить форму добавления слова<br>Ожидаемый результат: добавление и отображение добавленного слова       | Успешно.                  |
| Удаление слова           | Действие: нажать на кнопку «Словари и слова», нажать на словарь, нажать на «мусорный бак» у слова<br>Ожидаемый результат: удаление слова   | Успешно.                  |

Продолжение таблицы 4.2

| Название функции | Описание тестирования   | Итог тестирования функции |
|------------------|---|---------------------------|
| Изменение слова  | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на словарь, нажать на «карандаш», изменить данные в форме слова<br>Ожидаемый результат: изменение слова                       | Успешно.                  |
| Экспорт слов     | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Экспорт» рядом со словом<br>Ожидаемый результат: скачивание <i>JSON</i> файла со словами                                  | Успешно.                  |
| Импорт слов      | Действие: нажать на кнопку «Словари и слова» в навигационной панели, нажать на «Импорт» рядом со словом, выбрать подходящий <i>JSON</i> файл<br>Ожидаемый результат: добавление слов с <i>JSON</i> файла. | Успешно.                  |
| Просмотр тем     | Действие: нажать на кнопку «Уроки» в навигационной панели<br>Ожидаемый результат: отображение тем.  | Успешно.                  |
| Добавление темы  | Действие: нажать на кнопку «Уроки» в навигационной панели, заполнить форму для создания урока, в форме создания урока указать новое название темы<br>Ожидаемый результат: создание темы и урока.          | Успешно.                  |
| Удаление темы    | Действие: нажать на кнопку «Уроки» в навигационной панели, удалить все уроки в теме<br>Ожидаемый результат: удаление темы.  | Успешно.                  |

Продолжение таблицы 4.3

| Название функции   | Описание тестирования   | Итог тестирования функции |
|--------------------|---|---------------------------|
| Добавление урока   | Действие: нажать на кнопку «Уроки» в навигационной панели, заполнить форму для создания урока<br>Ожидаемый результат: создание урока.   | Успешно.                  |
| Удаление урока     | Действие: нажать на кнопку «Уроки» в навигационной панели, нажать на кнопку «Мусорки» рядом с уроком<br>Ожидаемый результат: удаление урока.  | Успешно.                  |
| Просмотр задания   | Действие: нажать на кнопку «Уроки» в навигационной панели, нажать на урок<br>Ожидаемый результат: получение заданий урока.  | Успешно.                  |
| Удаление задания   | Действие: нажать на кнопку «Уроки» в навигационной панели, нажать на урок, нажать на кнопку «Мусорки» рядом с уроком.<br>Ожидаемый результат: удаление задания.                       | Успешно.                  |
| Добавление задания | Действие: нажать на кнопку «Уроки» в навигационной панели, нажать на урок, заполнить форму для добавления заданий, нажать кнопку «Добавить»<br>Ожидаемый результат: создание задания. | Успешно.                  |

### 4.3 Функциональное тестирование для роли Гостя

В таблице 4.3 представлены результаты тестирования функций Гостя

Таблица 4.3 - Описание тестирования функций Гостя

| Название функции | Описание тестирования  | Итог тестирования функции |
|------------------|--|---------------------------|
| Регистрация      | <p>Действие: ввести корректные данные(имя, фамилию, <i>email</i>, пароль) и отправить форму. Проверь письмо с паролем на почте</p> <p>Ожидаемый результат: учетная запись создана, пользователь получил письмо на почту.</p> | Успешно.                  |
| Авторизация      | <p>Действие: ввести корректные данные(<i>email</i>, пароль) и отправить форму.</p> <p>Ожидаемый результат: пользователь входит в учетную запись .</p>  | Успешно.                  |

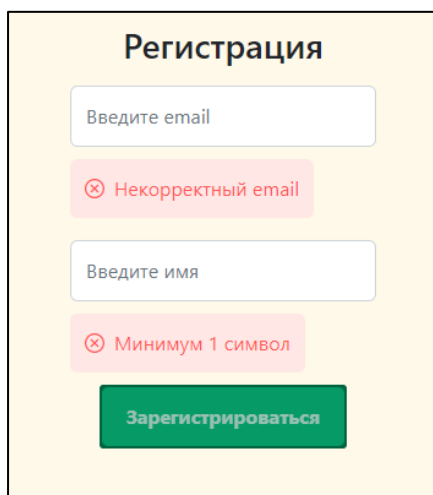
### 4.4 Вывод по разделу

1. Выполнено функциональное тестирование для каждой функции, представленной в диаграмме вариантов использования.
2. Все этапы пройдены успешно, web-приложение работает стабильно.
3. Количество тестов составило 39, покрытие тестами – 95%.

## 5 Руководство пользователя

### 5.1 Руководство Гостя

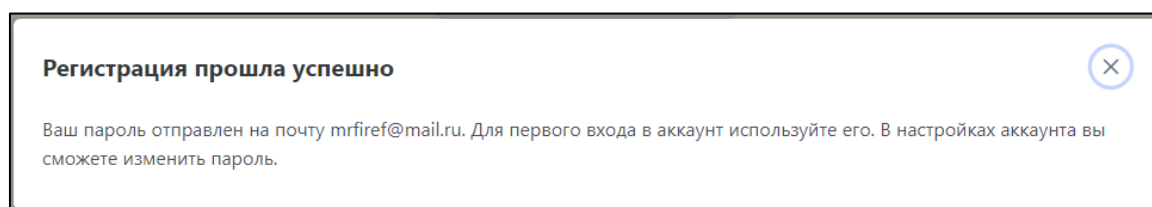
При первом переходе к приложению у неавторизованного пользователя есть возможность зарегистрироваться либо авторизоваться. Форма со страницы регистрации приведена на рисунке 5.1.



The registration form is titled "Регистрация" (Registration). It contains two input fields: "Введите email" (Enter email) and "Введите имя" (Enter name). Below the email field is a red error message: "✗ Некорректный email" (Incorrect email). Below the name field is a red error message: "✗ Минимум 1 символ" (Minimum 1 symbol). At the bottom of the form is a green button labeled "Зарегистрироваться" (Register).

Рисунок 5.1 – Форма для регистрации пользователя

При заполнении формы корректными данными и последующем нажатии на кнопку регистрации, пользователь получит сообщение, показанное на рисунке 5.2.



The success message dialog box is titled "Регистрация прошла успешно" (Registration successful). It contains the text: "Ваш пароль отправлен на почту mrfiref@mail.ru. Для первого входа в аккаунт используйте его. В настройках аккаунта вы сможете изменить пароль." (Your password has been sent to the email mrfiref@mail.ru. For the first login, use it. In the account settings, you will be able to change the password.) There is a close button (✗) in the top right corner.

Рисунок 5.2 – Сообщение об успешной регистрации

После закрытия диалогового окна пользователь перенаправляется на страницу входа, представленную на рисунке 5.3.

**Авторизация**

mrfiref@mail.ru

.....

**Войти**

Рисунок 5.3 – Форма для авторизации Гостя

## 5.2 Руководство Пользователя

После введения корректных данных и нажатия кнопки входа, пользователь перенаправляется на страницу со статистикой. На странице статистики находится таблица со статистикой всех пользователей за всё время, личная статистика за всё время и за текущий день в виде диаграмм. Страница представлена на рисунке 5.4.

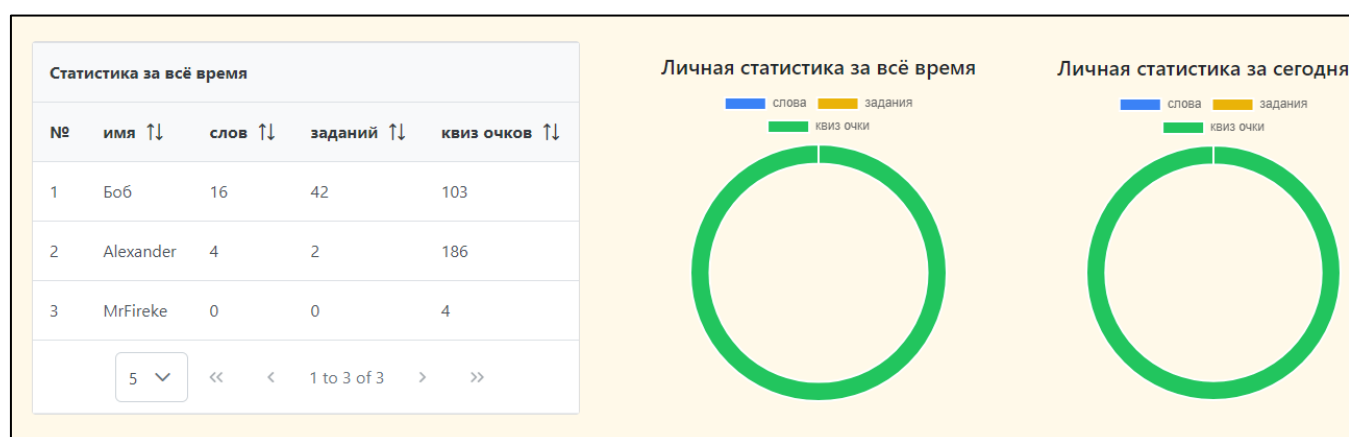


Рисунок 5.4 – Страница статистики

Перемещение по приложению осуществляется с помощью навигационной панели вверху страницы, которую можно увидеть в верху рисунка 5.4. Для изменения настроек аккаунта или обучения, пользователю необходимо перейти на страницу «Личный кабинет», которая представлена на рисунке 5.5.

Рисунок 5.5 – Страница «Личный кабинет»

На данной странице можно изменить пароль пользователя, заполнив все 3 поля верными и валидными данными. Чтобы изменить настройки обучения, необходимо выставить нужные параметры на ползунках и выпадающих списках и нажать на кнопку «Сохранить».

Для перехода на страницу просмотра словарей и слов с текущим прогрессом необходимо в выпадающем меню «Словари и слов» нажать на кнопку «Просмотреть». Страница представлена на рисунке 5.6.

| Словари     |                   |                | Слова        |                 |            |                          |            |                       |
|-------------|-------------------|----------------|--------------|-----------------|------------|--------------------------|------------|-----------------------|
| название ↑↓ | описание ↑↓       | экспортировать | испанский ↑↓ | транскрипция ↑↓ | русский ↑↓ | описание ↑↓              | выучено ↑↓ | правильных ответов ↑↓ |
| Дом         | Слова для дома    |                | casa         | 'ka.sa          | дом        | Жилое помещение          | нет        | 0                     |
| Животные    | Слова по животным |                | puerta       | 'pwer.ta        | дверь      | Вход в помещение         | нет        | 0                     |
|             |                   |                | ventana      | ben'ta.na       | окно       | Проем в стене для света  | нет        | 0                     |
|             |                   |                | pared        | pa.'red         | стена      | Вертикальная конструкция | нет        | 0                     |
|             |                   |                | techo        | 'te.tʃo         | потолок    | Верхняя часть помещения  | нет        | 0                     |

Рисунок 5.6 – Страница просмотра словарей и слов

На данной странице в таблицах отображены словари пользователя и администратора. Для каждого слова дополнительно к основной информации показан текущий прогресс изучения и выучено слово или нет.

В выпадающем меню «Словари и слова», нажав на кнопку «Изменить», пользователю откроется страница, изображенная на рисунке 5.7.

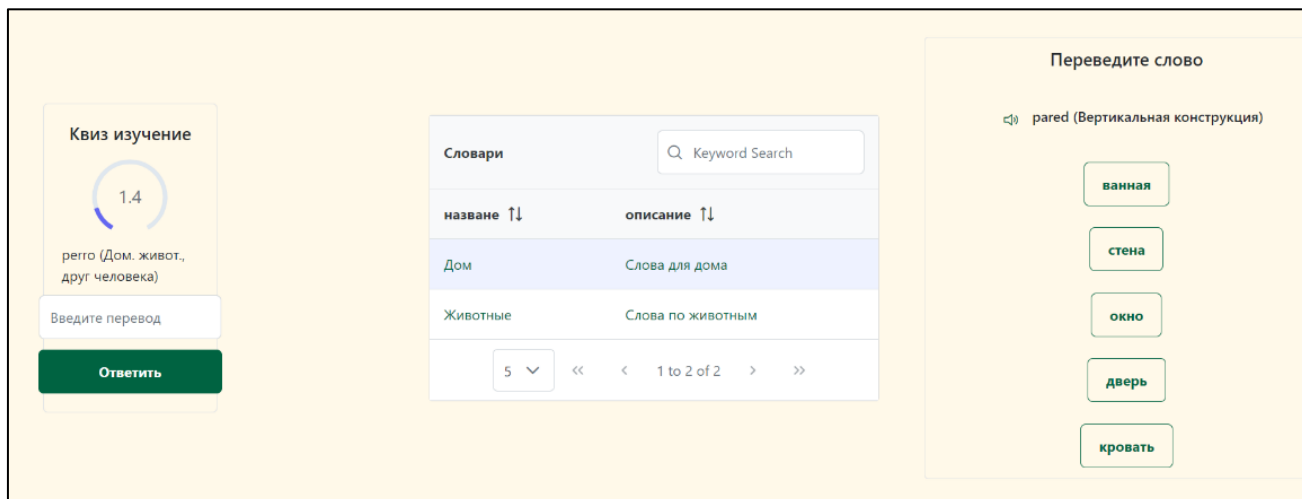


Рисунок 5.7 – Страница изучения слов

На странице доступны два варианта изучения слов: выбор из пяти вариантов и перевод слова в виде квиза. Пользователю в случайном порядке подбирается слово и 5 вариантов перевода, один из которых правильный. Также доступна функция прослушивания текущего слова. При правильном ответе слово меняется, в обратном случае пользователь получит сообщение, представленное на рисунке 5.8.

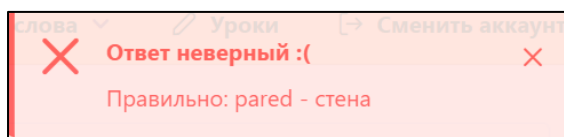


Рисунок 5.8 – Сообщение при неверном переводе слова

Второй вариант изучения подразумевает под собой генерацию случайного слова раз в 10 секунд, перевод которого нужно ввести в текстовое поле. При неверном переводе пользователь получит сообщение, аналогичное сообщению на рисунке 5.8, иначе страница изменится так, как показано на рисунке 5.9.

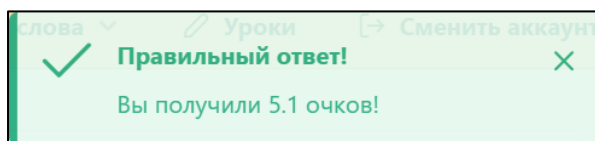


Рисунок 5.9 – Верный перевод слова в квизе

После верного ответа пользователь получает квиз очки, попадёт в таблицу лидеров и кнопка отправления ответа становится неактивна. Таблица лидеров обновляется для каждого слова.

В навигационном меню, нажав на кнопку «Уроки», пользователь будет направлен на одноименную страницу. Интерфейс данной страницы представлен на рисунке 5.10.



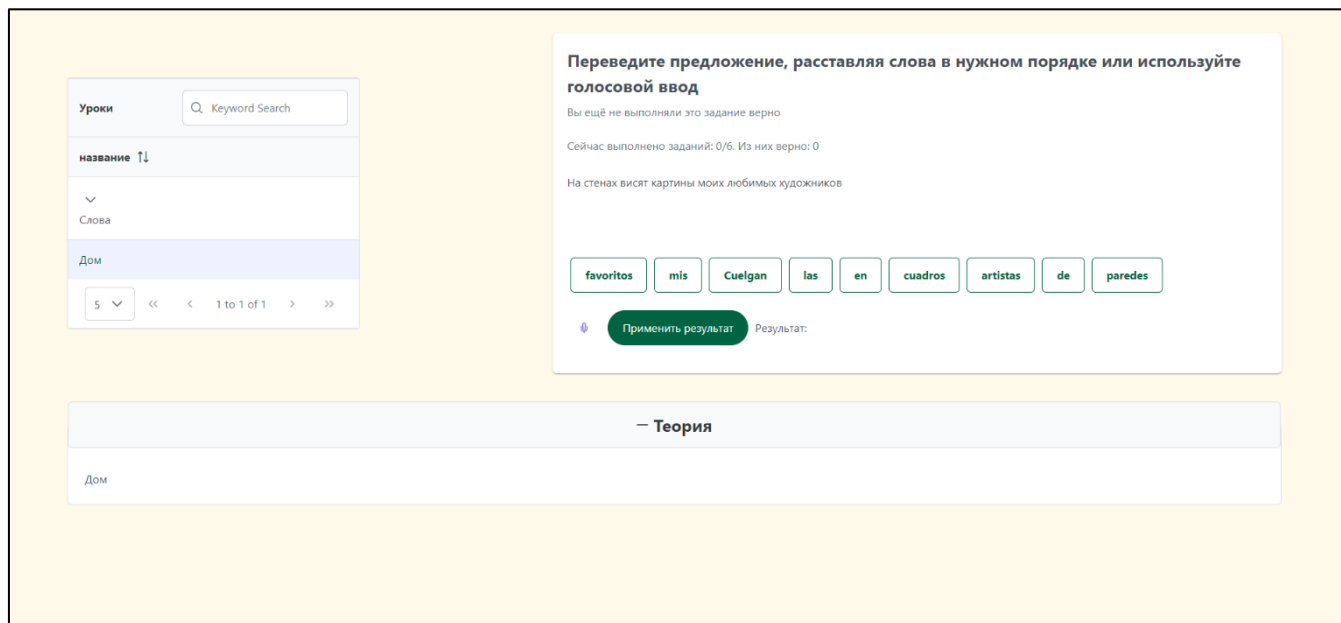


Рисунок 5.10 – Страница «Уроки»

Выбрав один из доступных уроков, форма с заданиями будет заполнена случайным заданием из выбранного урока. Из слов, находящихся в кнопках, нужно составить перевод предложения, выбирая в нужном порядке необходимые слова. В случае правильного перевода предложения будет выбрано следующее задание, иначе пользователь получит ошибку, которая представлена на рисунке 5.11.

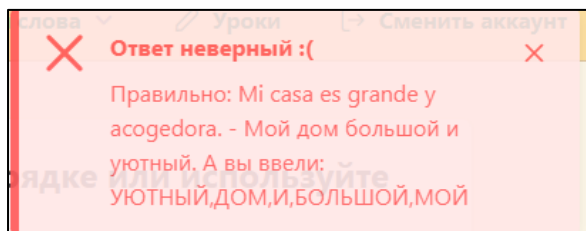


Рисунок 5.11 – Сообщение при неверном переводе предложения

Когда задания в уроке закончатся, пользователю будет предложено выбрать новый урок или пройти урок ещё раз.

### 5.3 Руководство администратора

В выпадающем меню «Словари и слова», нажав на кнопку «Изменить», администратору откроется страница, изображенная на рисунке 5.12. На этой странице представлены 2 таблицы и 2 формы для добавления словарей и слов. При введении некорректных данных, администратор получит сообщение об ошибке. Для добавления слова, дополнительно необходимо выбрать словарь, нажав на необходимый в таблице словарей.

Словари

Keyword Search

название ↑↓

описание ↑↓

импорт

|          |                   |  |  |  |
|----------|-------------------|--|--|--|
| Дом      | Слова для дома    |  |  |  |
| Животные | Слова по животным |  |  |  |

5

<<

<

1 to 2 of 2

>

>>

Слова

Keyword Search

испанский ↑↓

транскрипция ↑↓

русский ↑↓

описание ↑↓

|         |           |         |                          |  |
|---------|-----------|---------|--------------------------|--|
| casa    | 'ka.sa    | дом     | Жилое помещение          |  |
| puerta  | 'pwer.ta  | дверь   | Вход в помещение         |  |
| ventana | ben'ta.na | окно    | Проем в стене для света  |  |
| pared   | pa.'red   | стена   | Вертикальная конструкция |  |
| techo   | 'te.tʃo   | потолок | Верхняя часть помещения  |  |

5

<<

<

1 to 5 of 10

>

>>

Форма для добавления словарей

Название

Описание

необязательное поле

Добавить

Форма для добавления слов

испанский

транскрипция

русский

описание

необязательное поле

Добавить

Рисунок 5.12 – Страница изменения словарей и слов

Для обновления и удаления словарей и слов есть соответствующие кнопки в каждой строке таблиц. Пример изменения слова изображен на рисунке 5.13.

| испанский ↑↓                      | транскрипция ↑↓                     | русский ↑↓                       | описание ↑↓                                  |
|-----------------------------------|-------------------------------------|----------------------------------|--|
| <input type="text" value="casa"/> | <input type="text" value="'ka.sa"/> | <input type="text" value="дом"/> | <input type="text" value="Жилое помещение"/> |

Рисунок 5.13 – Изменение слова

После ввода всех изменения необходимо нажать на «галочку», чтобы изменения применились или на «крестик», чтобы изменения не были сохранены. Также в таблицах доступны сортировки по всем столбцам, при нажатии на название столбца, и поиск по таблице с помощью поля ввода в шапке таблицы.

Для импорта и экспорта слов из определённых словарей есть соответствующие кнопки в каждой строке таблицы. На рисунке 5.14 можно видеть кнопки импорта и экспорта для словаря «Семья».

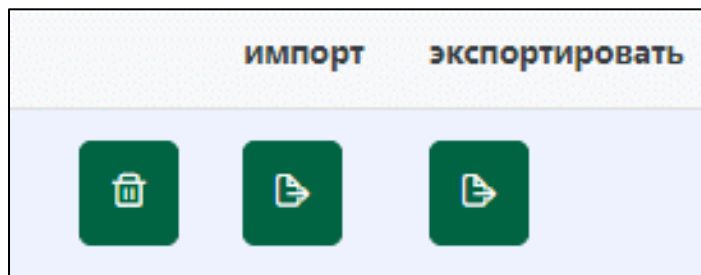


Рисунок 5.14 – Экспорт и импорт словарей

После ввода всех изменения необходимо нажать на «галочку», чтобы изменения применились или на «крестик», чтобы изменения не были сохранены. Также в таблицах доступны сортировки по всем столбцам, при нажатии на название столбца, и поиск по таблице с помощью поля ввода в шапке таблицы.

Данная страница доступна только для администратора. Для перехода на неё, администратору необходимо нажать на кнопку «Уроки», находящуюся в навигационной панели.

На странице находятся две таблицы для уроков и заданий и две формы для добавления уроков и заданий. Для удаления урока в его строке нужно нажать на кнопку со значком мусорки. Удаление задания происходит аналогично. Выбрав урок или задание, форма будет заполнена данными, как показано на рисунке 5.15.

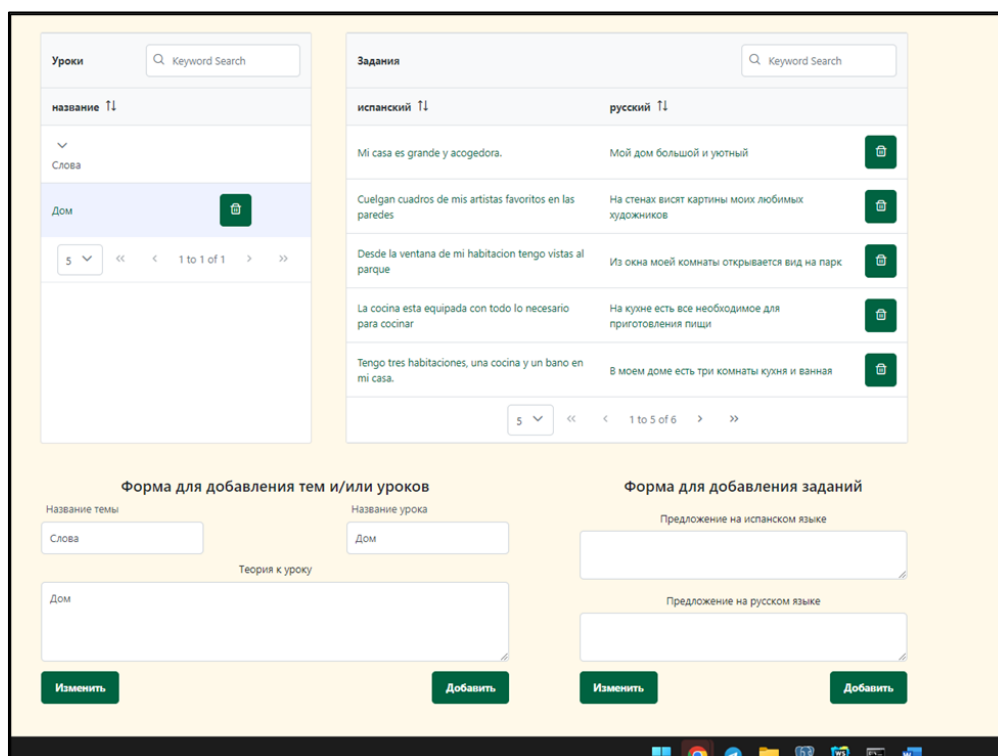


Рисунок 5.15 – Страница «Уроки» для администратора

Для изменения выбранного урока или задания необходимо исправить данные в форме и нажать на кнопку «Изменить».

## Заключение

В ходе данного курсового проекта было реализовано веб-приложение для изучения испанского языка. Основные результаты разработки:

1. Web-приложение поддерживает 3 роли: Гость, Пользователь, Администратор.

2. В ходе рассмотрения аналогов разрабатываемого приложения были выявлены основные функции и элементы

3. Реализовано 35 функций.

4. База данных включает 14 таблиц.

5. Использована REST-архитектура, обеспечивающая удобное взаимодействие клиента и сервера через HTTP. Основные компоненты: Модели данных: описывают таблицы базы данных и их связи, разработаны с помощью PostgreSQL.

Поддержка двусторонней связи через WebSocket для обновлений в реальном времени.

7. Было разработано 39 тестов, которые покрыли 95% функционала веб-приложения.

На основе полученных результатов работы web-приложения можно заключить, что цель проекта достигнута, а все требования технического задания были полностью выполнены.

### Список использованных источников

- 1 node [Электронный ресурс]. – Режим доступа: <https://nodejs.org/> – Дата доступа: 12.12.2024.
- 2 Главная страница сайта Duolingo [Электронный ресурс]. – Режим доступа: <https://www.duolingo.com/> – Дата доступа: 12.12.2024.
- 3 Главная страница сайта Poliglot16 [Электронный ресурс]. – Режим доступа: <https://poliglot16.ru/> – Дата доступа: 12.12.2024.
- 4 Главная страница сайта Puzzle English [Электронный ресурс]. – Режим доступа: <https://puzzle-english.com/> – Дата доступа: 12.12.2024.
- 5 nginx [Электронный ресурс]. – Режим доступа: <https://nginx.org/> – Дата доступа: 12.12.2024.
- 6 PostgreSQL [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/> – Дата доступа: 13.12.2024.
- 7 HTTP [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/HTTP> – Дата доступа: 14.10.2024.
- 8 HTTPS [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#https> – Дата доступа: 14.10.2024.
- 9 TCP [Электронный ресурс]. – Режим доступа: <https://www.rfc-editor.org/rfc/rfc793> – Дата доступа: 14.10.2024.
- 10 WebSocket [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/WebSocket> – Дата доступа: 14.10.2024.
- 11 Nest [Электронный ресурс]. – Режим доступа: <https://nestjs.com/> – Дата доступа: 12.12.2024.
- 12 TypeScript [Электронный ресурс]. – Режим доступа: <https://www.typescriptlang.org/> – Дата доступа: 12.12.2024.
- 13 JavaScript [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> – Дата доступа: 12.12.2024.
- 14 ECMAScript [Электронный ресурс]. – Режим доступа: <https://tc39.es/ecma262/> – Дата доступа: 12.12.2024.
- 15 Prisma [Электронный ресурс]. – Режим доступа: <https://www.prisma.io/> – Дата доступа: 12.12.2024.
- 16 @nestjs [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/@nestjs/core> – Дата доступа: 12.12.2024.
- 17 @prisma/client [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/@prisma/client> – Дата доступа: 12.12.2024.
- 18 bcrypt [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/bcrypt> – Дата доступа: 12.12.2024.
- 19 class-transformer [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/class-transformer> – Дата доступа: 12.12.2024.
- 20 class-validator [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/class-validator> – Дата доступа: 12.12.2024.

- 21 cookie [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/cookie> – Дата доступа: 12.12.2024.
- 22 cookie-parser [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/cookie-parser> – Дата доступа: 12.12.2024.
- 23 cors [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/cors> – Дата доступа: 12.12.2024.
- 24 helmet [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/helmet> – Дата доступа: 12.12.2024.
- 25 https [Электронный ресурс]. – Режим доступа: <https://nodejs.org/api/https.html> – Дата доступа: 12.12.2024.
- 26 joi [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/joi> – Дата доступа: 12.12.2024.
- 27 nodemailer [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/nodemailer> – Дата доступа: 12.12.2024.
- 28 passport [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/passport> – Дата доступа: 12.12.2024.
- 29 passport-jwt [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/passport-jwt> – Дата доступа: 12.12.2024.
- 30 reflect-metadata [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/reflect-metadata> – Дата доступа: 12.12.2024.
- 31 rxjs [Электронный ресурс]. – Режим доступа: <https://rxjs.dev/> – Дата доступа: 12.12.2024.
- 32 websocket [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/ws> – Дата доступа: 12.12.2024.
- 33 @fortawesome [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/@fortawesome/fontawesome-svg-core> – Дата доступа: 12.12.2024.
- 34 @fortawesome/react-fontawesome [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/@fortawesome/react-fontawesome> – Дата доступа: 12.12.2024.
- 35 @testing-library/jest-dom [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/@testing-library/jest-dom> – Дата доступа: 12.12.2024.
- 36 @testing-library/react [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/@testing-library/react> – Дата доступа: 12.12.2024.
- 37 axios [Электронный ресурс]. – Режим доступа: <https://axios-http.com/> – Дата доступа: 12.12.2024.
- 38 bootstrap [Электронный ресурс]. – Режим доступа: <https://getbootstrap.com/> – Дата доступа: 12.12.2024.
- 39 chart.js [Электронный ресурс]. – Режим доступа: <https://www.chartjs.org/> – Дата доступа: 12.12.2024.
- 40 date-fns [Электронный ресурс]. – Режим доступа: <https://date-fns.org/> – Дата доступа: 12.12.2024.
- 41 dotenv [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/dotenv> – Дата доступа: 12.12.2024.

- 42 js-cookie [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/js-cookie> – Дата доступа: 12.12.2024.
- 43 jwt-decode [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/jwt-decode> – Дата доступа: 12.12.2024.
- 44 mobx [Электронный ресурс]. – Режим доступа: <https://mobx.js.org/> – Дата доступа: 12.12.2024.
- 45 primeicons [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/primeicons> – Дата доступа: 12.12.2024.
- 46 primereact [Электронный ресурс]. – Режим доступа: <https://www.primefaces.org/primereact/> – Дата доступа: 12.12.2024.
- 47 react [Электронный ресурс]. – Режим доступа: <https://reactjs.org/> – Дата доступа: 12.12.2024.
- 48 react-bootstrap [Электронный ресурс]. – Режим доступа: <https://react-bootstrap.github.io/> – Дата доступа: 12.12.2024.
- 49 react-date-range [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/react-date-range> – Дата доступа: 12.12.2024.
- 50 react-dom [Электронный ресурс]. – Режим доступа: <https://reactjs.org/docs/react-dom.html> – Дата доступа: 12.12.2024.
- 51 react-router-dom [Электронный ресурс]. – Режим доступа: <https://reactrouter.com/> – Дата доступа: 12.12.2024.
- 52 react-scripts [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/react-scripts> – Дата доступа: 12.12.2024.
- 53 rxjs [Электронный ресурс]. – Режим доступа: <https://rxjs.dev/> – Дата доступа: 12.12.2024.
- 54 socket.io-client [Электронный ресурс]. – Режим доступа: <https://socket.io/> – Дата доступа: 12.12.2024.
- 55 web-vitals [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/web-vitals> – Дата доступа: 12.12.2024.
- 56 ws [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/ws> – Дата доступа: 12.12.2024.
- 57 Modular Layered Architecture [Электронный ресурс]. – Режим доступа: <https://martinfowler.com/eaaCatalog/layers.html> – Дата доступа: 12.12.2024.

## Приложение А

### Листинг схемы базы данных в ORM Prisma

```

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model User {
  id            int           @id @default(autoincrement())
  name          String
  email         String        @unique
  password      String
  roleId        int           @default(1) @map("role_id")
  idEnableLesson int          @default(1) @map("id_enable_lesson")
  role          Role          @relation(fields: [roleId], references: [id], onDelete: Cascade)
  session       Session?
  grammarProgress GrammarProgress[]
  settings      UserSettings?
  dictionary    Dictionary[]
  lexiconProgress LexiconProgress[]
  statistics    Statistics[]

  @@map("users")
}

model UserSettings {
  userId      int           @unique @map("user_id")
  user        User          @relation(fields: [userId], references: [id], onDelete: Cascade)
  countRepeatWordForLearned int @map("count_repeat_word_for_learned")
  countRepeatWordsSimultaneously int @map("count_learning_words_simultaneously")
  learningModeWords LearningMode
  learningModeTasks LearningMode

  @@map("users_settings")
}

model Topic {
  id            int           @id @default(autoincrement())
  name          String        @unique
  lessons       Lesson[]

  @@map("topic")
}

```



```

model Lesson {
    id            int            @id @default(autoincrement())
    name          String
    theory        String
    topicId       int            @map("topic_id")
    topic         Topic          @relation(fields: [topicId], refer-
ences: [id], onDelete: Cascade)
    tasks         Task[]

    @@map("lessons")
}

model Task {
    id            int            @id @default(autoincrement())
    spanishSentence String      @map("spanish_sentence")
    russianSentence String      @map("russian_sentence")
    lessonId      int            @map("lesson_id")
    lesson        Lesson        @relation(fields: [lessonId],
references: [id], onDelete: Cascade)
    grammarProgress GrammarProgress[]

    @@unique([spanishSentence, russianSentence, lessonId])
    @@map("tasks")
}

model GrammarProgress {
    id            int            @id @default(autoincrement())
    userId        int            @map("user_id")
    user          User           @relation(fields: [userId], references: [id], onDelete:
Cascade)
    taskId        int            @map("task_id")
    task          Task           @relation(fields: [taskId], references: [id], onDelete:
Cascade)

    @@unique([userId, taskId])
    @@map("grammar_progresses")
}

model Dictionary {
    id            int            @id @default(autoincrement())
    name          String
    description    String?
    creatorId     int            @map("creator_id")
    user          User           @relation(fields: [creatorId],
references: [id], onDelete: Cascade)
    dictionaryToWord DictionaryToWord[]
    words         Word[]

    @@map("dictionaries")
}

model Word {

```

```

    id            int            @id @default(autoincrement())
    spanishSpelling String        @map("spanish_spelling")
    transcription String
    russianSpelling String        @map("russian_spelling")
    description   String?
    lexiconProgress LexiconProgress[]
    dictionaryToWord DictionaryToWord[]
    dictionaries  Dictionary[]

    @@unique([spanishSpelling, russianSpelling])
    @@map("words")
}

model DictionaryToWord {
    id            int            @id @default(autoincrement())
    dictionaryId  int
    wordId        int
    dictionary     Dictionary @relation(fields: [dictionaryId], references: [id], onDelete: Cascade)
    word           Word         @relation(fields: [wordId], references: [id], onDelete: Cascade)

    @@unique([dictionaryId, wordId])
    @@map("dictionary_to_word")
}

model LexiconProgress {
    id            int            @id @default(autoincrement())
    progressCount int            @map("progress_count")
    isLearned     Boolean        @map("is_learned")
    userId         int            @map("user_id")
    user           User          @relation(fields: [userId], references: [id], onDelete: Cascade)
    wordId         int            @map("word_id")
    word           Word          @relation(fields: [wordId], references: [id], onDelete: Cascade)

    @@unique([userId, wordId])
    @@map("lexicon_progresses")
}

model Statistics {
    id            int            @id @default(autoincrement())
    date          DateTime
    words          int?
    tasks          int?
    quizPoints     Float?        @map("quiz_points")
    userId         int            @map("user_id")
    user           User          @relation(fields: [userId], references: [id], onDelete: Cascade)

    @@unique([date, userId])
    @@map("statistics")
}

```

```

}

model Session {
    userId      int      @unique @map("user_id")
    user        User     @relation(fields: [userId], references: [id],
onDelete: Cascade)
    refreshToken String @map("refresh_token")

    @@map("sessions")
}

model Role {
    id          int          @id @default(autoincrement())
    title       String
    permissions Permission[]
    users       User[]

    @@map("roles")
}

model Permission {
    id          int          @id @default(autoincrement())
    descriptor  String
    context     String?
    method      Method
    roleId      int          @map("role_id")
    role        Role        @relation(fields: [roleId], references: [id],
onDelete: Cascade)

    @@map("permissions")
}

enum Method {
    ALL
    GET
    POST
    DELETE
    PUT
    PATCH
}

enum LearningMode {
    TRANSLATE_FROM_spanish
    TRANSLATE_FROM_RUSSIAN
    COMBINED
}

```

## Приложение Б

### Листинг файла package.json серверной части приложения

```
{
  "name": "spanish-api",
  "version": "1.0.0",
  "description": "",
  "author": "",
  "private": true,
  "license": "UNLICENSED",
  "scripts": {
    "build": "nest build",
    "format": "prettier --write \"src/**/*.ts\" \"test/**/*.ts\"",
    "start": "nest start",
    "start:dev": "nest start --watch",
    "start:debug": "nest start --debug --watch",
    "start:prod": "node dist/main",
    "lint": "eslint \"{src,apps,libs,test}/**/*.ts\" --fix",
    "test": "jest",
    "test:watch": "jest --watch",
    "test:cov": "jest --coverage",
    "test:debug": "node --inspect-brk -r tsconfig-paths/register -r ts-node/register node_modules/.bin/jest --runInBand",
    "test:e2e": "jest --config ./test/jest-e2e.json"
  },
  "dependencies": {
    "@nestjs/common": "^9.0.0",
    "@nestjs/config": "^2.2.0",
    "@nestjs/core": "^9.0.0",
    "@nestjs/jwt": "^10.0.2",
    "@nestjs/mapped-types": "*",
    "@nestjs/passport": "^9.0.3",
    "@nestjs/platform-express": "^9.0.0",
    "@nestjs/platform-socket.io": "^9.4.0",
    "@nestjs/swagger": "^6.1.4",
    "@nestjs/websockets": "^9.4.0",
    "@prisma/client": "^4.11.0",
    "bcrypt": "^5.1.0",
    "class-transformer": "^0.5.1",
    "class-validator": "^0.14.0",
    "cookie": "^0.5.0",
    "cookie-parser": "^1.4.6",
    "cors": "^2.8.5",
    "helmet": "^7.0.0",
    "https": "^1.0.0",
    "joi": "^17.9.0",
    "nodemailer": "^6.7.8",
    "passport": "^0.6.0",
    "passport-jwt": "^4.0.1",
    "passport-local": "^1.0.0",
    "reflect-metadata": "^0.1.13",
    "rxjs": "^7.8.1",
    "websocket": "^1.0.34"
  }
}
```

```

},
"devDependencies": {
  "@nestjs/cli": "^9.0.0",
  "@nestjs/schematics": "^9.0.0",
  "@nestjs/testing": "^9.0.0",
  "@types/bcrypt": "^5.0.0",
  "@types/cookie-parser": "^1.4.3",
  "@types/express": "^4.17.13",
  "@types/jest": "29.2.4",
  "@types/node": "^18.11.18",
  "@types/passport-jwt": "^3.0.8",
  "@types/passport-local": "^1.0.35",
  "@types/supertest": "^2.0.11",
  "@types/ws": "^8.5.4",
  "@typescript-eslint/eslint-plugin": "^5.0.0",
  "@typescript-eslint/parser": "^5.0.0",
  "eslint": "^8.0.1",
  "eslint-config-prettier": "^8.3.0",
  "eslint-plugin-prettier": "^4.0.0",
  "jest": "29.3.1",
  "prettier": "^2.3.2",
  "prisma": "^4.12.0",
  "source-map-support": "^0.5.20",
  "supertest": "^6.1.3",
  "ts-jest": "29.0.3",
  "ts-loader": "^9.2.3",
  "ts-node": "^10.9.1",
  "tsconfig-paths": "4.1.1",
  "typescript": "^4.9.5"
},
"jest": {
  "moduleFileExtensions": [
    "js",
    "json",
    "ts"
  ],
  "rootDir": "src",
  "testRegex": ".*\\.spec\\.ts$",
  "transform": {
    "^.+\\.?(t|j)s$": "ts-jest"
  },
  "collectCoverageFrom": [
    "**/*.?(t|j)s"
  ],
  "coverageDirectory": "../coverage",
  "testEnvironment": "node"
},
"prisma": {
  "seed": "ts-node prisma/seed.ts"
}
}

```

## Приложение В

### Листинг файла package.json клиентской части приложения

```
{
  "name": "spanish-front",
  "proxy": "https://localhost:5000/",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@fortawesome/fontawesome-svg-core": "^6.3.0",
    "@fortawesome/free-regular-svg-icons": "^6.3.0",
    "@fortawesome/free-solid-svg-icons": "^6.3.0",
    "@fortawesome/react-fontawesome": "^0.2.0",
    "@testing-library/jest-dom": "^5.16.4",
    "@testing-library/react": "^13.1.1",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.3.3",
    "bootstrap": "^5.2.3",
    "chart.js": "^4.3.0",
    "date-fns": "^2.29.3",
    "dotenv": "^16.0.3",
    "js-cookie": "^3.0.1",
    "jwt-decode": "^3.1.2",
    "mobx": "^6.0.5",
    "mobx-react-lite": "^3.1.7",
    "primeicons": "^6.0.1",
    "primereact": "^9.2.3",
    "react": "^18.0.0",
    "react-bootstrap": "^2.7.4",
    "react-date-range": "^1.4.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.10.0",
    "react-scripts": "5.0.1",
    "rxjs": "^7.8.1",
    "socket.io-client": "^4.6.1",
    "web-vitals": "^2.1.4",
    "ws": "^8.13.0"
  },
  "scripts": {
    "start": "set HTTPS=true& set\nSSL_CERT_FILE=C:\\\\My\\\\OpenSSL\\\\CW.crt& set\nSSL_KEY_FILE=C:\\\\My\\\\OpenSSL\\\\CW.key& react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
```

```
"production": [  
  ">0.2%",  
  "not dead",  
  "not op_mini all"  
],  
"development": [  
  "last 1 chrome version",  
  "last 1 firefox version",  
  "last 1 safari version"  
]  
}
```

## Приложение Г

### Код контроллера dictionary

```

import {
  Controller,
  Get,
  Post,
  Body,
  Patch,
  Param,
  Delete,
  UseGuards,
  Req,
  Res,
} from '@nestjs/common';
import { ApiBearerAuth, ApiOkResponse, ApiTags } from '@nestjs/swagger';
import { swaggerType } from 'src/helpers/swagger/utils';
import { JwtAuthGuard } from '../auth/guard/jwt-auth.guard';
import RequestWithUser from '../auth/interface/request-with-user.interface';
import { DictionaryService } from '../dictionary.service';
import { CreateDictionaryDto } from '../dto/create-dictionary.dto';
import { UpdateDictionaryDto } from '../dto/update-dictionary.dto';
import { DictionaryReviewResponse } from '../response/dictionary-review.response';
import { DictionaryResponse } from '../response/dictionary.response';
import * as fs from 'fs';
import { Response as ExpressResponse } from 'express';

@ApiTags('dictionary')
@Controller('dictionary')
export class DictionaryController {
  constructor(private readonly dictionaryService: DictionaryService) {}

  @ApiBearerAuth()
  @UseGuards(JwtAuthGuard)
  @Post()
  public createDictionary(
    @Req() req: RequestWithUser,
    @Body() createDictionaryDto: CreateDictionaryDto,
  ): Promise<DictionaryResponse> {
    return this.dictionaryService.createDictionary(
      +req.user.id,
      createDictionaryDto,
    );
  }
}

```



```

    );
}

@ApiBearerAuth()
@UseGuards(JwtAuthGuard)
@Get('export/:id')
public async export(@Param('id') id: string, @Res() res: ExpressRe-
sponse) {
    const fileName = 'dictionary_' + Date.now() + '.json';
    const fileContent = await this.dictionaryService.exportDiction-
ary(+id);

    fs.writeFileSync(fileName, JSON.stringify(fileContent), 'utf-8');

    res.setHeader('Content-Type', 'application/json');
    res.setHeader('Content-Disposition', `attachment; file-
name=${fileName}`);

    const fileStream = fs.createReadStream(fileName);
    fileStream.pipe(res);

    fileStream.on('end', () => {
        fs.unlinkSync(fileName);
    });
}

@ApiBearerAuth()
@ApiOkResponse(swaggerType(DictionaryResponse))
@UseGuards(JwtAuthGuard)
@Get('admin')
public getAdminDictionaries(): Promise<DictionaryResponse[]> {
    return this.dictionaryService.getAdminDictionaries();
}

@ApiBearerAuth()
@ApiOkResponse(swaggerType(DictionaryResponse))
@UseGuards(JwtAuthGuard)
@Get('learn')
public getDictionariesLearn(
    @Req() req: RequestWithUser,
): Promise<DictionaryReviewResponse[]> {
    return this.dictionaryService.getDictionariesLearn(+req.user.id);
}

@ApiBearerAuth()
@ApiOkResponse(swaggerType(DictionaryResponse))
@UseGuards(JwtAuthGuard)

```

```

@Get('review')
public getDictionariesReview(
    @Req() req: RequestWithUser,
): Promise<DictionaryReviewResponse[]> {
    return this.dictionaryService.getDictionariesReview(+req.user.id);
}

@ApiBearerAuth()
@ApiOkResponse(swaggerType(DictionaryResponse))
@UseGuards(JwtAuthGuard)
@Get('user')
public getUserDictionaries(
    @Req() req: RequestWithUser,
): Promise<DictionaryResponse[]> {
    return this.dictionaryService.getUserDictionaries(+req.user.id);
}

@ApiBearerAuth()
@UseGuards(JwtAuthGuard)
@Delete('/:id')
public remove(
    @Param('id') id: string,
    @Req() req: RequestWithUser,
): Promise<void> {
    return this.dictionaryService.removeDictionary(+id,
+req.user.id);
}

@ApiBearerAuth()
@ApiOkResponse(swaggerType(DictionaryResponse))
@UseGuards(JwtAuthGuard)
@Patch('/:id')
public updateDictionary(
    @Param('id') id: string,
    @Req() req: RequestWithUser,
    @Body() updateDictionaryDto: UpdateDictionaryDto,
): Promise<DictionaryResponse> {
    return this.dictionaryService.updateDictionary(
        +id,
        +req.user.id,
        updateDictionaryDto,
    );
}

@ApiBearerAuth()
@UseGuards(JwtAuthGuard)

```

```
@Post('import/:id')
public async importDictionary(
    @Param('id') id: string,
    @Req() req: RequestWithUser,
    @Body() body: { property: any[] }, // Указываем ожидаемую
структуру
): Promise<void> {
    const wordData = body.property; // Извлекаем массив из property
    await this.dictionaryService.importDictionary(+id, wordData);
}
}
```

## Приложение Д

### Код сервиса dictionary

```
import {
  BadRequestException,
  Injectable,
  NotFoundException,
} from '@nestjs/common';
import { Dictionary } from '@prisma/client';
import { DictionaryRepository } from '../dictionary.repository';
import { CreateDictionaryDto } from '../dto/create-dictionary.dto';
import { UpdateDictionaryDto } from '../dto/update-dictionary.dto';
import { DictionaryReviewResponse } from '../response/dictionary-review.response';
import { DictionaryResponse } from '../response/dictionary.response';
import { WordForExportResponse } from '../word/response/word-for-export.response';

@Injectable()
export class DictionaryService {
  constructor(private readonly dictionaryRepository: DictionaryRepository) {}

  public async checkDictionaryOwner(
    dictionaryId: number,
    userId: number,
  ): Promise<void> {
    const dictionary: Dictionary =
      await this.dictionaryRepository.getDictionaryById(dictionaryId);

    if (!dictionary) {
      throw new NotFoundException('Dictionary not found');
    }
  }
}
```

```

        if (dictionary.creatorId !== userId) {
            throw new BadRequestException('Not owner try to update dictionary');
        }
    }

    public async importDictionary(
        dictionaryId: number,
        wordData: WordForExportResponse[],
    ): Promise<void> {
        const dictionary = await this.dictionaryRepository.getDictionaryById(
            dictionaryId,
        );

        if (!dictionary) {
            throw new NotFoundException('Dictionary not found');
        }

        await this.dictionaryRepository.importWords(dictionaryId, wordData);
    }

    public async createDictionary(
        creatorId: number,
        createDictionaryDto: CreateDictionaryDto,
    ): Promise<DictionaryResponse> {
        return await this.dictionaryRepository.createDictionary(
            creatorId,
            createDictionaryDto,
        );
    }

```

```

public async exportDictionary(
    dictionaryId: number,
): Promise<WordForExportResponse[]> {
    return await this.dictionaryRepository.exportDictionaryById(dictionaryId);
}

public async getAdminDictionaries(): Promise<DictionaryResponse[]>
{
    return await this.dictionaryRepository.getAdminDictionaries();
}

public async getDictionariesLearn(
    id: number,
): Promise<DictionaryReviewResponse[]> {
    const data = await this.dictionaryRepository.getDictionariesLearn(id);

    data.forEach((dictionary) => {
        dictionary.words = dictionary.words.filter(
            (word) =>
                word.lexiconProgress[0] === undefined ||
                word.lexiconProgress[0].isLearned === false,
        );
    });

    return data;
}

public async getDictionariesReview(
    id: number,
): Promise<DictionaryReviewResponse[]> {
    return await this.dictionaryRepository.getDictionariesReview(id);
}

```

```
}

    public async getUserDictionaries(id: number): Promise<DictionaryRe-
sponse[]> {
        return await this.dictionaryRepository.getUserDictionaries(id);
    }

    public async removeDictionary(
        dictionaryId: number,
        userId: number,
    ): Promise<void> {
        await this.checkDictionaryOwner(dictionaryId, userId);
        await this.dictionaryRepository.removeDictionaryById(diction-
aryId);
    }

    public async updateDictionary(
        dictionaryId: number,
        userId: number,
        updateDictionaryDto: UpdateDictionaryDto,
    ): Promise<DictionaryResponse> {
        await this.checkDictionaryOwner(dictionaryId, userId);

        return await this.dictionaryRepository.updateDictionaryById(
            dictionaryId,
            updateDictionaryDto,
        );
    }
}
```

## Приложение Е

### Код репозитория dictionary

```
import { Injectable } from '@nestjs/common';

import { DatabaseService } from 'src/database/database.service';
import { CreateDictionaryDto } from '../dto/create-dictionary.dto';
import { UpdateDictionaryDto } from '../dto/update-dictionary.dto';
import { DictionaryResponse } from '../response/dictionary.response';
import { WordForExportResponse } from '../word/response/word-for-export.response';

@Injectable()
export class DictionaryRepository {
  private dictionariesReviewSelect = {
    id: true,
    name: true,
    description: true,
    creatorId: true,
    words: {
      select: {
        id: true,
        spanishSpelling: true,
        transcription: true,
        russianSpelling: true,
        description: true,
        lexiconProgress: {
          select: {
            progressCount: true,
            isLearned: true,
          },
        },
      },
    },
  },
};
private fullDictionarySelect = {
  id: true,
  name: true,
  description: true,
  creatorId: true,
  words: {
    select: {
      id: true,
      spanishSpelling: true,
      transcription: true,
      russianSpelling: true,
```



```

        description: true,
    },
},
};

constructor(private readonly db: DatabaseService) {}

public async createDictionary(
    creatorId: number,
    createDictionaryDto: CreateDictionaryDto,
): Promise<DictionaryResponse> {
    return await this.db.dictionary.create({
        data: {
            creatorId,
            ...createDictionaryDto,
        },
        select: this.fullDictionarySelect,
    });
}

public async exportDictionaryById(
    id: number,
): Promise<WordForExportResponse[]> {
    const dictionary = await this.db.dictionary.findMany({
        where: {
            id,
        },
        select: {
            words: {
                select: {
                    spanishSpelling: true,
                    transcription: true,
                    russianSpelling: true,
                    description: true,
                },
            },
        },
    });

    return dictionary[0].words;
}

public async getAdminDictionaries(): Promise<DictionaryResponse[]>
{
    return await this.db.dictionary.findMany({
        where: {
            user: {

```

```

        roleId: 2,
    },
},
select: this.fullDictionarySelect,
});
}

public async getDictionariesLearn(creatorId: number): Promise<any[]> {
    return await this.db.dictionary.findMany({
        where: {
            OR: [{ creatorId }, { user: { roleId: 2 } }],
        },
        select: {
            id: true,
            name: true,
            description: true,
            creatorId: true,
            words: {
                select: {
                    id: true,
                    spanishSpelling: true,
                    transcription: true,
                    russianSpelling: true,
                    description: true,
                    lexiconProgress: {
                        where: {
                            userId: creatorId,
                        },
                        select: {
                            id: true,
                            progressCount: true,
                            isLearned: true,
                        },
                        take: 1,
                    },
                },
            },
        },
    });
}

public async getDictionariesReview(creatorId: number): Promise<any[]> {
    return await this.db.dictionary.findMany({
        where: {
            OR: [{ creatorId }, { user: { roleId: 2 } }],

```

```

    },
    select: {
      id: true,
      name: true,
      description: true,
      creatorId: true,
      words: {
        select: {
          id: true,
          spanishSpelling: true,
          transcription: true,
          russianSpelling: true,
          description: true,
          lexiconProgress: {
            where: {
              userId: creatorId,
            },
            select: {
              progressCount: true,
              isLearned: true,
            },
            take: 1,
          },
        },
      },
    },
  },
});
}

public async getDictionaryById(id: number): Promise<DictionaryRe-
sponse> {
  return await this.db.dictionary.findUnique({
    where: {
      id,
    },
    select: this.fullDictionarySelect,
  });
}

public async getUserDictionaries(
  creatorId: number,
): Promise<DictionaryResponse[]> {
  return await this.db.dictionary.findMany({
    where: {
      creatorId,
    },
    select: this.fullDictionarySelect,
  });
}

```

```

    });
}

public async removeDictionaryById(id: number): Promise<void> {
    await this.db.dictionary.delete({
        where: {
            id,
        },
    });
}

public async updateDictionaryById(
    id: number,
    updateDictionaryDto: UpdateDictionaryDto,
): Promise<DictionaryResponse> {
    return await this.db.dictionary.update({
        where: {
            id,
        },
        data: {
            ...updateDictionaryDto,
        },
        select: this.fullDictionarySelect,
    });
}

public async importWords(
    dictionaryId: number,
    wordData: WordForExportResponse[],
): Promise<void> {
    try {
        console.log('Data to insert:', wordData);

        // Шаг 1: Создать записи в таблице Word
        const createdWords = await this.db.word.createMany({
            data: wordData.map((word) => ({
                spanishSpelling: word.spanishSpelling,
                transcription: word.transcription,
                russianSpelling: word.russianSpelling,
                description: word.description,
            })),
            skipDuplicates: true,
        });

        console.log('Words created:', createdWords);

        // Шаг 2: Получить слова из базы данных (с их ID)
        const words = await this.db.word.findMany({

```

```

    where: {
      OR: wordData.map((word) => ({
        spanishSpelling: word.spanishSpelling,
        russianSpelling: word.russianSpelling,
      })),
    },
  });

  // Шаг 3: Создать записи в таблице DictionaryToWord
  const dictionaryToWordData = words.map((word) => ({
    dictionaryId: dictionaryId,
    wordId: word.id,
  }));

  await this.db.dictionaryToWord.createMany({
    data: dictionaryToWordData,
    skipDuplicates: true,
  });

  console.log('DictionaryToWord relationships created.');
```

```

} catch (error) {
  console.error('Error during importWords:', error);
  throw error;
}
}
}

```

## Приложение Ж

### Код страницы PersonalCabinet

```

import { useContext, useEffect, useRef, useState } from "react";
import { Button } from "primereact/button";
import { Toast } from "primereact/toast";
import { Slider } from "primereact/slider";
import { Dropdown } from "primereact/dropdown";

//theme
import "primereact/resources/themes/lara-light-indigo/theme.css";

//core
import "primereact/resources/primereact.min.css";

//icons
import "primeicons/primeicons.css";

import { observer } from "mobx-react-lite";
import Input from "../components/Input";
import { REGEXES } from "../utils/regexes";

import { changePassword, updateUserSettings } from "../api-requests/user-api";

import "../styles/personal-cabinet.css";
import { Context } from "..";
import { LearningMode } from "../utils/learn-settings";

const PersonalCabinet = observer(() => {
  const toast = useRef(null);
  const { userSettings } = useContext(Context);
  const { user } = useContext(Context);

  useEffect(() => {
    setLearningSettings({
      countRepeatWordForLearned: userSettings.getCountRepeatWordFor-
Learned(),
      countRepeatWordsSimultaneously:
        userSettings.getCountRepeatWordsSimultaneously(),
      learningModeWords: userSettings.getLearningModeWords(),
      learningModeTasks: userSettings.getLearningModeTasks(),
    });
  }, [userSettings]);

  const showSuccess = (message) => {

```

```

toast.current.show({
  severity: "success",
  summary: "Успешно",
  detail: message,
  life: 3000,
});
};

const showError = (message) => {
  toast.current.show({
    severity: "error",
    summary: "Ошибка",
    detail: message,
    life: 3000,
  });
};

const [password, setPassword] = useState({
  oldPassword: "",
  newPassword: "",
  repitNewPassword: "",
});
const [isValidPassword, setIsValidPassword] = useState(null);
const [isValidPasswordRepit, setIsValidPasswordRepit] = use-
eState(null);

const changePasswordHandler = ({ id, value }) => {
  setPassword((prev) => ({ ...prev, [id]: value }));
  console.log(user.getRoleId());

  if (id === "newPassword") {
    setIsValidPassword(REGEXES.PASSWORD_REGEX.test(value));
    setIsValidPasswordRepit(password.repitNewPassword === value);
  }

  if (id === "repitNewPassword") {
    setIsValidPasswordRepit(password.newPassword === value);
  }
};

const changePasswordRequest = async () => {
  try {
    await changePassword(password.oldPassword, password.newPass-
word);
    setPassword({
      oldPassword: "",
      newPassword: "",

```

```

        repitNewPassword: "",
    });
    showSuccess("Пароль изменён");
} catch (error) {
    showError("Введён неверный текущий пароль");
}
};

const [learningSettings, setLearningSettings] = useState({
    countRepeatWordForLearned: userSettings.getCountRepeatWordFor-
Learned(),
    countRepeatWordsSimultaneously:
        userSettings.getCountRepeatWordsSimultaneously(),
    learningModeWords: userSettings.getLearningModeWords(),
    learningModeTasks: userSettings.getLearningModeTasks(),
});

const changeSettingsHandler = (id, value) => {
    if (learningSettings[id] !== value) {
        setLearningSettings((prev) => ({ ...prev, [id]: value }));
    }
};

const learningModes = [
    {
        name: "Перевод с испанского",
        value: LearningMode.TRANSLATE_FROM_SPANISH,
    },
    { name: "Перевод с русского", value: LearningMode.TRANSLATE_FROM_RUSSIAN },
    { name: "Комбинированный вариант", value: LearningMode.COMBINED },
],

];

const updateUserSettingsRequest = async () => {
    try {
        await updateUserSettings(learningSettings);
        userSettings.setUserSettings(learningSettings);
        showSuccess("Настройки обучения обновлены.");
    } catch (error) {
        showError();
    }
};

return (
    <div className="container">
        <Toast ref={toast} />

```



```

<div className="account-settings">
  <h4 className="">Обновление данных аккаунта</h4>
  <Input
    id="oldPassword"
    value={password.oldPassword}
    labelText="Текущий пароль"
    isValidValue={true}
    inputType="password"
    setDataHandler={changePasswordHandler}
  />
  <Input
    id="newPassword"
    value={password.newPassword}
    labelText="Новый пароль"
    isValidValue={isValidPassword}
    inputType="password"
    setDataHandler={changePasswordHandler}
    errorMessage="От 4 до 16 символов"
  />
  <Input
    id="repiteNewPassword"
    value={password.repiteNewPassword}
    labelText="Повторите новый пароль"
    isValidValue={isValidPasswordRepite}
    inputType="password"
    setDataHandler={changePasswordHandler}
    errorMessage="Пароли не совпадают"
  />
  <Button
    className="change-password-button"
    label="Сменить пароль"
    onClick={changePasswordRequest}
    disabled={!isValidPassword || !isValidPasswordRepite}
  />
</div>
<div className="learning-settings">
  <h4>Настройки обучения</h4>
  <div className="learning-settings__block">
    <label className="learning-settings__label">
      Количество правильных ответов, следующих подряд, после
которого
      слово считается выученным и исключается из списка, но
остаётся в
      словаре
    </label>
    <Slider
      className="learning-settings__input"

```

```

        value={learningSettings.countRepeatWordForLearned}
        min={2}
        max={10}
        onChange={ (e) =>
            changeSettingsHandler("countRepeatWordForLearned",
e.value)
        }
    />
    <label>{learningSettings.countRepeatWordForLearned}</label>
</div>
<div className="learning-settings__block">
    <label className="learning-settings__label">
        Количество слов изучаемых одновременно
    </label>
    <Slider
        className="learning-settings__input"
        value={learningSettings.countRepeatWordsSimultaneously}
        min={10}
        max={50}
        onChange={ (e) =>
            changeSettingsHandler("countRepeatWordsSimultaneously",
e.value)
        }
    />
    <label>{learningSettings.countRepeatWordsSimultane-
ously}</label>
</div>
<div className="learning-settings__block">
    <label className="learning-settings__label">
        Способ изучения лексики
    </label>
    <Dropdown
        value={learningSettings.learningModeWords}
        onChange={ (e) =>
            changeSettingsHandler("learningModeWords", e.value)
        }
        options={learningModes}
        optionLabel="name"
        placeholder="Выберите вариант"
        className="w-full"
    />
</div>
<div className="learning-settings__block">
    <label className="learning-settings__label">
        Способ изучения грамматики
    </label>
    <Dropdown

```

```

        value={learningSettings.learningModeTasks}
        onChange={ (e) =>
            changeSettingsHandler("learningModeTasks", e.value)
        }
        options={learningModes}
        optionLabel="name"
        placeholder="Выберите вариант"
        className="w-full"
    />
</div>
<Button
    className="update-user-settings-button"
    label="Сохранить"
    onClick={updateUserSettingsRequest}
/>
</div>
</div>
);
});

export default PersonalCabinet;

```