Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных,**

**принятия решений и финансовых технологий**

**Пояснительная записка к курсовой работе**

на тему:

**Игровая познавательная программа «Сленг английского языка»**

Выполнила:

студентка группы ПИ19-3

Попова Софья Александровна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Научный руководитель:

Петросов Давид Арегович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

2021

**Содержание**

|  |  |
| --- | --- |
| Введение………………………………………………………………… | 3 |
| 1. Постановка задачи……………………………………………………… | 4 |
| 2. Описание предметной области………………………………………… | 6 |
| 3. Актуальность автоматизации………………………………………….. | 7 |
| 3.1. Алгоритмические решения………………………………………….  3.1.1. Сервер…………………………………………………………...  3.1.2. Клиент…………………………………………………………... | 8  8  9 |
| 3.2. Описание интерфейса программы………………………………….  3.2.1. Сервер…………………………………………………………...  3.2.2. Клиент…………………………………………………………... | 11  11  11 |
| 3.3. Состав приложения…………………………………………………..  3.3.1. База данных……………………………………………………..  3.3.2. Сервер…………………………………………………………...  3.3.3. Клиент…………………………………………………………... |  |
| 4. Назначение и состав классов программы……………………………… |  |
| 5. Тестирование работы приложения…………………………………….. |  |
| Заключение………………………………………………………………… |  |
| Список литературы………………………………………………………... |  |
| Приложение………………………………………………………………... |  |

**Введение**

В современном мире всё развивается, начиная от языков, заканчивая технологиями. Однако стоит понимать, кто же является двигателем прогресса. Ответ очевиден – это человечество. Не только каждый человек по отдельности, но и все мы вместе: одни люди создают потребности, другие находят решения для их удовлетворения. С технологиями это работает понятно: изначально была изобретена самая маленькая, ограниченная по функционалу технология, и затем по мере потребностей её развивали, улучшали, делали масштабнее. Например, интернет изначально служил для загрузки текстовых страниц, а сейчас это высокоскоростная сеть, поддерживающая всевозможные виды мультимедии. Но что же происходит с языками, на которых мы разговариваем? Мы как будто не замечаем их развития, вроде, говорим на одном и том же языке, но, если оглянуться назад, можно увидеть, как он изменился за 5, за 10 лет. На самом деле, с языками происходит всё то же самое, что и с технологиями. Требования людей к языку меняются: нужно сокращать слова, чтобы доносить больший объём информации в единицу времени, нужно придумывать новые выражения для постоянно развивающегося мира. И это нормальный процесс, противиться ему бесполезно.

Процесс развития сам по себе является невероятно увлекательным, особенно если это касается повседневных вещей: вспомните наш русский язык год назад и посмотрите на него сейчас – он невероятно изменился: появились новые слова, новый сленг. Это просто колоссальное развитие для языка, на котором говорят несколько миллионов человек. А теперь представьте, что происходит с языком, на котором говорит более миллиарда человек. Даже представить страшно, насколько быстро он изменяется. Тем не менее, пример такого языка – английский язык, развивающийся так быстро, что не успеваешь запомнить слова, как на смену им приходят новые. Именно развитие английского языка и современных технологий являются целью моей работы.

**Постановка задачи**

К счастью, языки можно изучать с помощью современных технологий, приложений. Благодаря грамотно выстроенному процессу обучения, оно происходит с удовольствием и незаметно. В соответствии с выбранной темой я поставила перед собой задачу реализовать приложение для изучения сленга английского языка с использованием трёхзвенной архитектуры: клиента, серверы и базы данных. Для этого мне необходимо было спроектировать базу данных, удовлетворяющую моим потребностям, написать сервер, который будет получать запросы от клиента и в соответствии с ними возвращать требуемые данные, и написать графический интерфейс клиента, состоящий из нескольких взаимосвязанных окон и осуществляющий первичную валидацию данных.

Для написания базы данных для моего приложения мне пригодились знания о проектировании реляционных баз данных. Для реализации серверной части необходимо было освоить технологию ORM, позволяющую имплементировать базу данных с концепциями объектно-ориентированного программирования, а также реализовать выполнение запросов к базе данных. Для реализации клиентской части необходимо было продумать GUI (графический интерфейс) и реализовать его.

После тщательного анализа требований я поставила перед собой следующие задачи:

* разработать информационную модель предметной области с помощью таблиц реляционной базы данных и взаимодействие с ней при помощи сервера;
* разработать несколько форм пользовательского интерфейса для клиента;
* отображать в разработанной базе данных данные изучаемой мной предметной области – сленга английского языка;
* создать такое приложение, которое принесёт пользователю знания сленга английского языка, для этого я выбрала модель «урок-тестирование», где урок – набор теории, которую можно изучить, а тест – практическое применение изученного материала, при этом тест нужно было организовать таким образом, чтобы это было не просто проверкой знаний, а ещё и продолжением изучения темы;
* для реализации предыдущего пункта я должна была добавить в приложение такие функции как добавление новых объектов, получение информации по уже созданным данным, их обновление и удаление, фильтрацию записей, полезное отображение статистических и исторических данных;
* обрабатывать все получаемые от клиента данные таким образом, чтобы программа не завершалась аварийно, а выдавала клиенту соответствующие сообщения;
* писать комментарии таким способом, чтобы по завершении работы над приложением, можно было сгенерировать JavaDoc документацию;
* выстроить структуру приложения таким образом, чтобы она соответствовала модели MVC (model – view – controller).

**Описание предметной области**

Предметная область состоит в изучении живого английского языка. В процессе исследования мне приходилось сталкиваться с различными приложениями для изучение английского языка, но все они были однотипными и не до конца отражали скорость развития этого языка, они скорее были созданы для повседневного, например, иллюстрировали пользователю, что рука – это hand, друг – это friend, но они не учитывали специфику самого живого языка, разных акцентов и культур. Например, не было сказано, что в Великобритании друзей обычно называют mate, в Соединённых Штатах Америки – bro, body или dude. И это достаточно безобидный пример – есть такие слова, которые в американском акценте значат одно, а в британском – совсем другое (например, те же pants), а это, по моему мнению, необходимо знать, чтобы избежать неловких ситуаций.

Моей целью было отразить накопленные мной в течение нескольких лет знания о сленге английского языка, полученные при просмотре видеороликов, прочтении тематических статей, в одном приложении так, чтобы это было максимально просто и понятно пользователю, чтобы изучение чего-то нового было не в тягость, а с интересом.

Таким образом, я проектировала базу данных и серверную часть так, чтобы существовала концепция пользователя и администратора, где администратор формирует для пользователя полезную информацию – обобщает теорию для уроков, создаёт тесты, пишет пояснение к каждому вопросу так и реализует другие базовые функции (такие как редактирование пользователей, просмотр статистики), а пользователь на основе информации, предоставленной администратором, изучает сленг английского языка и также может смотреть результаты своего обучения.

Более подробно реализацию я опишу в соответствующих главах.

**Актуальность автоматизации**

Актуальность автоматизации моего приложения состоит в том, что пользователь не чувствует себя «обделённым» - он сразу же имеет доступ к информации, опубликованной администратором.

Сама программа устроена таким образом, что коэффициент её полезного действия в изучении английского языка для пользователя максимален по сравнению с некоторыми тестовыми системами. Если тест – это тоже получение знаний, то как их можно получать, если система закрыта, если не показывается, верный ли вариант ответа выбрал пользователь, если нет пояснения, почему верный вариант ответа именно такой, как он есть, а не другой?

Это мне не нравилось в большинстве тестовых систем, с которыми мне приходилось иметь дело, поэтому я создала собственное приложение, отвечающее моим требованиям в получении новой информации.

**Алгоритмические решения**

**Сервер**

Сервер написан с помощью фреймворка SpringBoot. Назначение сервера в том, чтобы посылать к базе данных запросы, основанные на потребностях пользователей и возвращать ответы клиенту. Это происходит благодаря тому, что при поступлении того или иного запроса от клиента он смотрит по своим контроллерам, что от него хотят получить, тем самым осуществляя переадресацию запросов на нужные url, ведь каждая url отвечает за конкретное действие. Например, если сервер получит от пользователя запрос на изменение существующих данных, но не сделает никаких изменений, а просто вернёт клиенту список всех записей из таблицы, это будет считаться за неправильную работу. Для помощи серверу в выполнении запросов посложнее, чем просто получение всех записей (или добавление записи, или обновление) служит такой интерфейс как JpaRepository. Благодаря этому интерфейсу сервер может обрабатывать запросы по нескольким критериям, например, получать все записи из таблицы, у которых внешний ключ соответствует передаваемому от клиента значению. Благодаря встроенным методам можно даже искать записи по нескольким внешним ключам, что я активно использую в своей работе. Ну и главные плюсы JpaRepository – это, во-первых, быстрая обработка запросов и быстрое получение ответов, и во-вторых, сокрытие реализации от разработчиков, то есть это невероятно удобно – просто послать требуемый запрос и получить ответ, а не самому заниматься отбором нужных записей, что привело бы к загромождению кода программы. Также в серверной части я использую фреймворк SpringSecurity, позволяющий осуществлять механизмы авторизации и аутентификации на основе роли и статуса пользователя (как я раньше уже описывала, у меня есть две роли – пользователь и администратор). Благодаря этому я защищаю некоторые таблицы от изменения пользователем, изменять данные в них может только администратор. Также для передачи прав доступа человека, от имени которого клиент посылает запрос на сервер, я использую такой стандарт как JSON Web Token, который хранит данные о роли пользователя, логине, времени создании токена и времени прекращении его действия. Иными словами, в небольшой строчке, зашифрованной с помощью секретного ключа, хранится вся необходимая информация – кто этот пользователь, осуществляющий запрос, и что он может делать, а чего не может. Стоит отметить, что этот токен передаётся при каждом запросе с заголовком «Authorization», что является довольно распространённой практикой. Благодаря небольшой строчке токена можно обеспечить безопасность приложения, ведь расшифровать его без знания секретного ключа невозможно.

**Клиент**

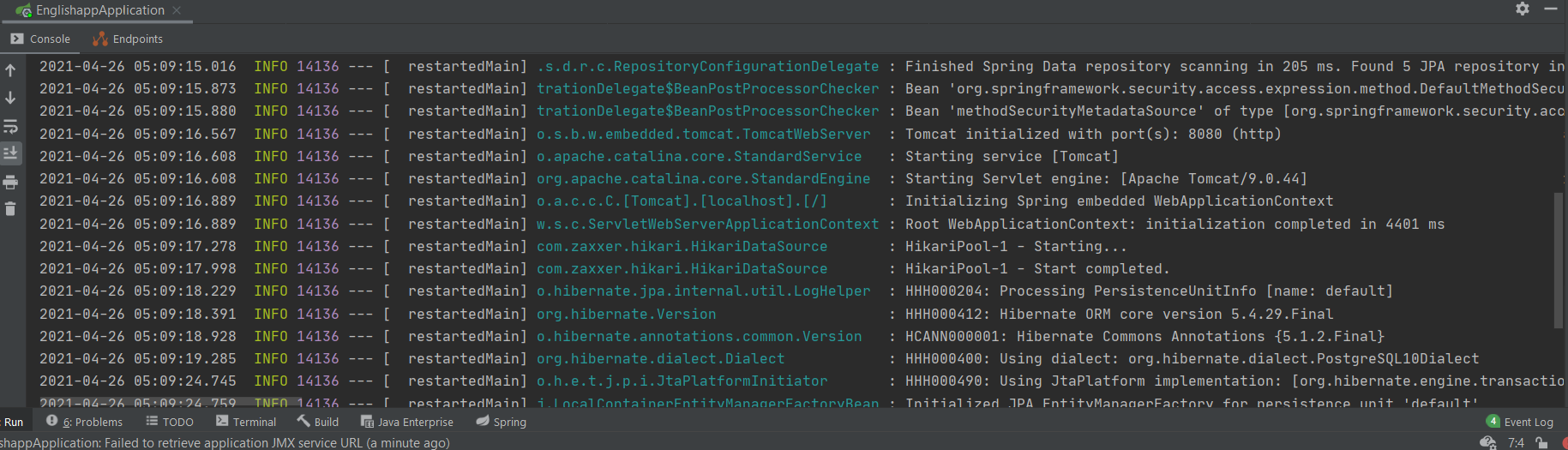
При авторизации в системе с помощью графического интерфейса, написанного на JavaFX – платформе на основе Java, как уже мною было описано в предыдущем пункте, для пользователя формируется токен, хранящий все необходимые данные. На основе данных, возвращаемых с сервера на запрос, человеку открывается окно, соответствующее его роли: если это администратор зашёл в систему, он имеет больший функционал, чем обычный пользователь. После появление формы клиент может перемещаться по её вкладкам, при нажатии определённых кнопок, появляются логически обособленные формы в другом окне. Стоит отметить, что запросы от клиента на сервер посылаются с помощью специально созданных для этого классов. Ещё одно отличие, которое указывает на наличие клиент-серверной архитектуры – это созданный в модели пользователя метод, который позволяет конвертировать Java объект (у меня это объект класса пользователя) в понятную серверу json-строку, которая уже будет отправляться ему в виде тела, а он будет выполнять с ней необходимые действия.

Также из особенностей моего клиента стоит отметить то, что для существования анимации выцветания мне приходится усыплять поток программы на некоторое незначительное время, чтобы эффект исчезания одной формы не перебивался эффектом появления другой формы в том же самом окне. При этом при таком подходе программа продолжает работать исправно и плавно, никакие данные при этом не теряются, быстродействие не снижается.

**Описание интерфейса программы**

**Сервер**

У сервера не существует отдельного графического интерфейса – он представлен в виде отображения информации в консоли, это проиллюстрировано на рис. 3.2.1.1.



*рис. 3.2.1.1*

**Клиент**

Графический интерфейс реализован для клиента с помощью упомянутой выше библиотеки JavaFX.

Всего мой клиент имеет 12 форм, при этом стоит отметить, что некоторые из них используются по несколько раз за счёт сокрытия одних полей формы и открытия других, а также установки элементам контроля каждый раз своего значения – именно того, которое нужно отображать для конкретных целей.

Формы, используемые в моём приложении, хранятся в виде файлов с разрешением fxml и представляют собой:

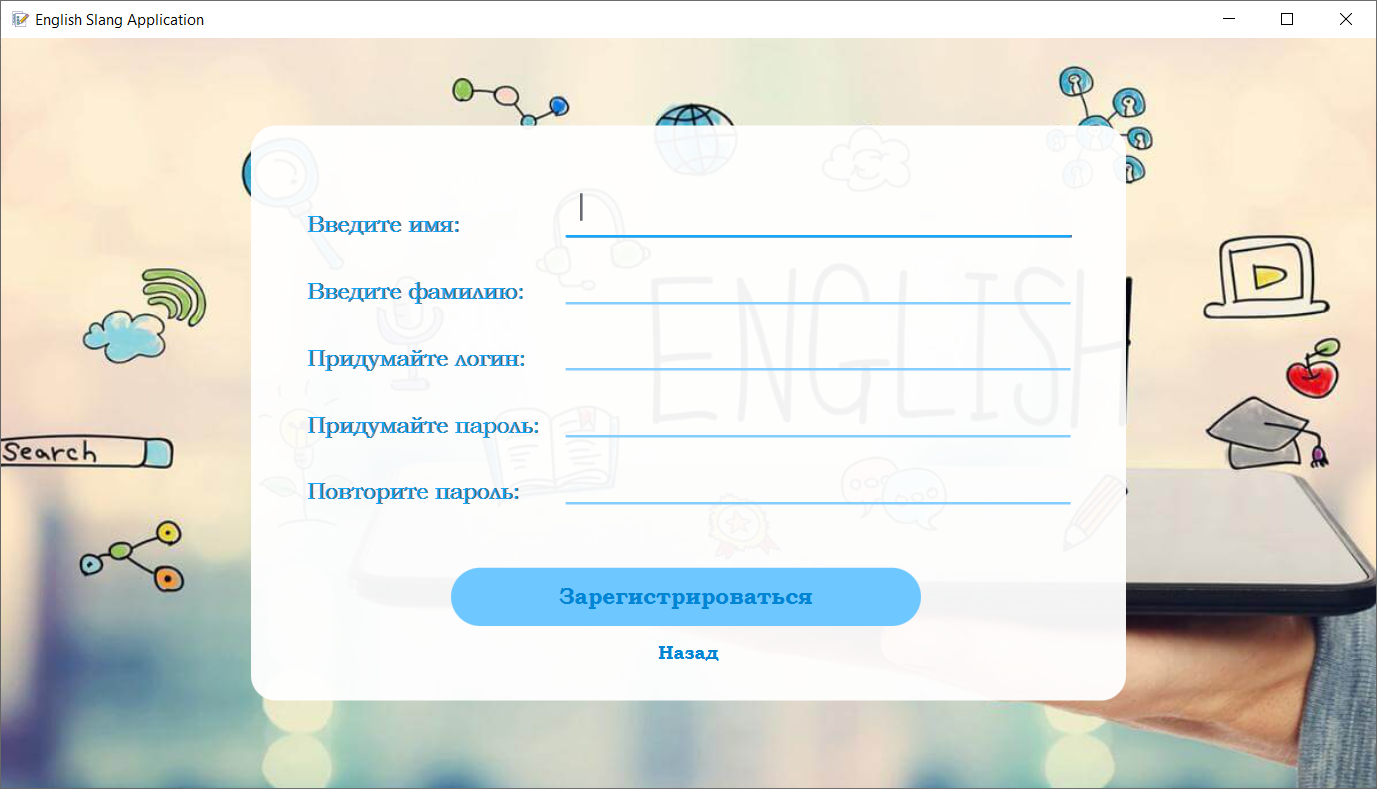
1. Главный фон, открытый в течение всей работы приложения;
2. Форма авторизации, которая накладывается поверх главного фона при открытии приложении, а также при её вызове при совершении определённых действий (рис. 3.2.2.1);



*рис. 3.2.2.1*

С помощью этой формы пользователь может войти в систему (или получить сообщение об ошибке), открыть окно регистрации или окно «Об авторе».

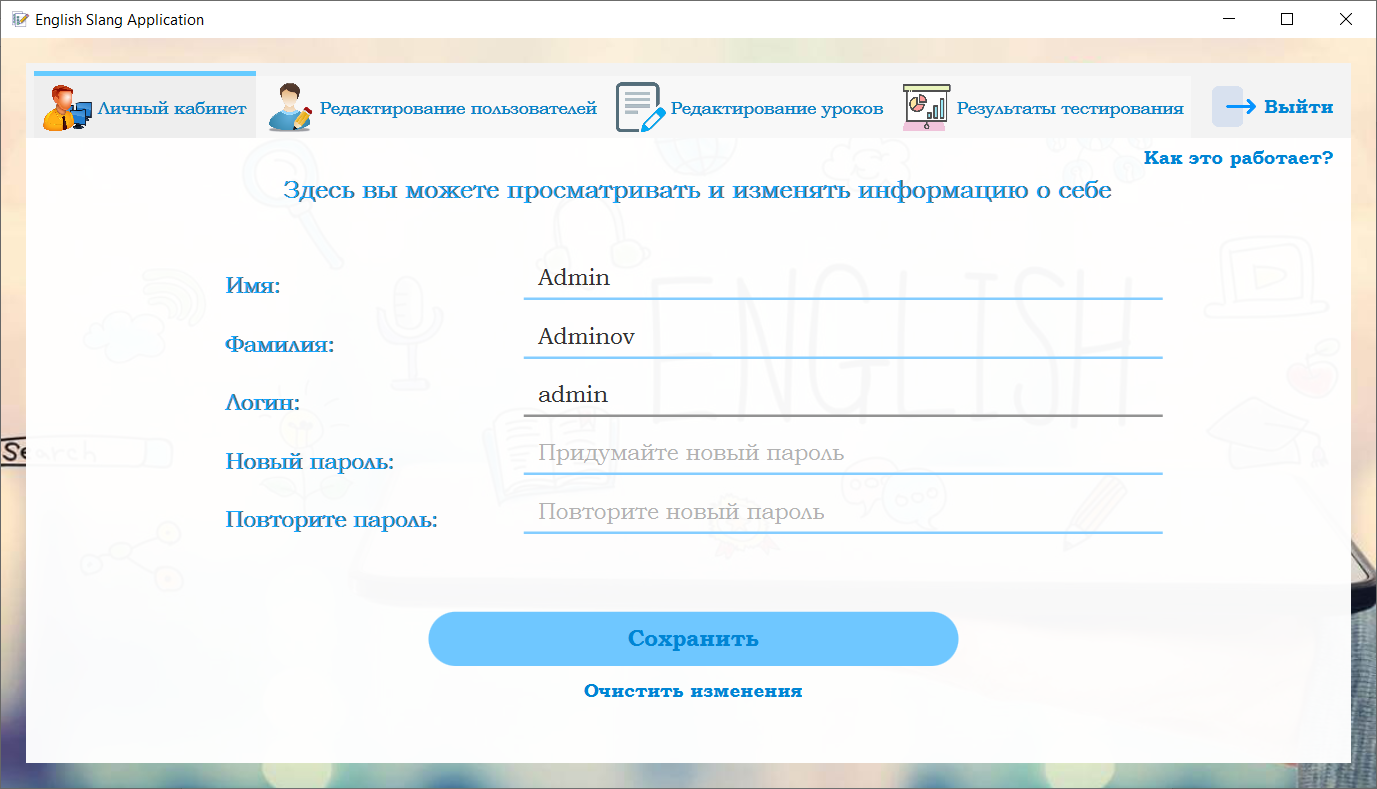
1. Форма регистрации так же располагается поверх заднего фона (рис. 3.2.2.2);



*рис. 3.2.2.2*

С помощью данной формы пользователь может зарегистрироваться в системе (при учёте успешной валидации введённых им регистрационных данных, иначе некорректные поля подсвечиваются красным и появляется сообщение об ошибке), в случае успешной регистрации пользователю присваивается роль пользователя и активный статус, при этом форма регистрации закрывается, вместо неё снова появляется форма авторизации, приглашающая пользователя войти в систему.

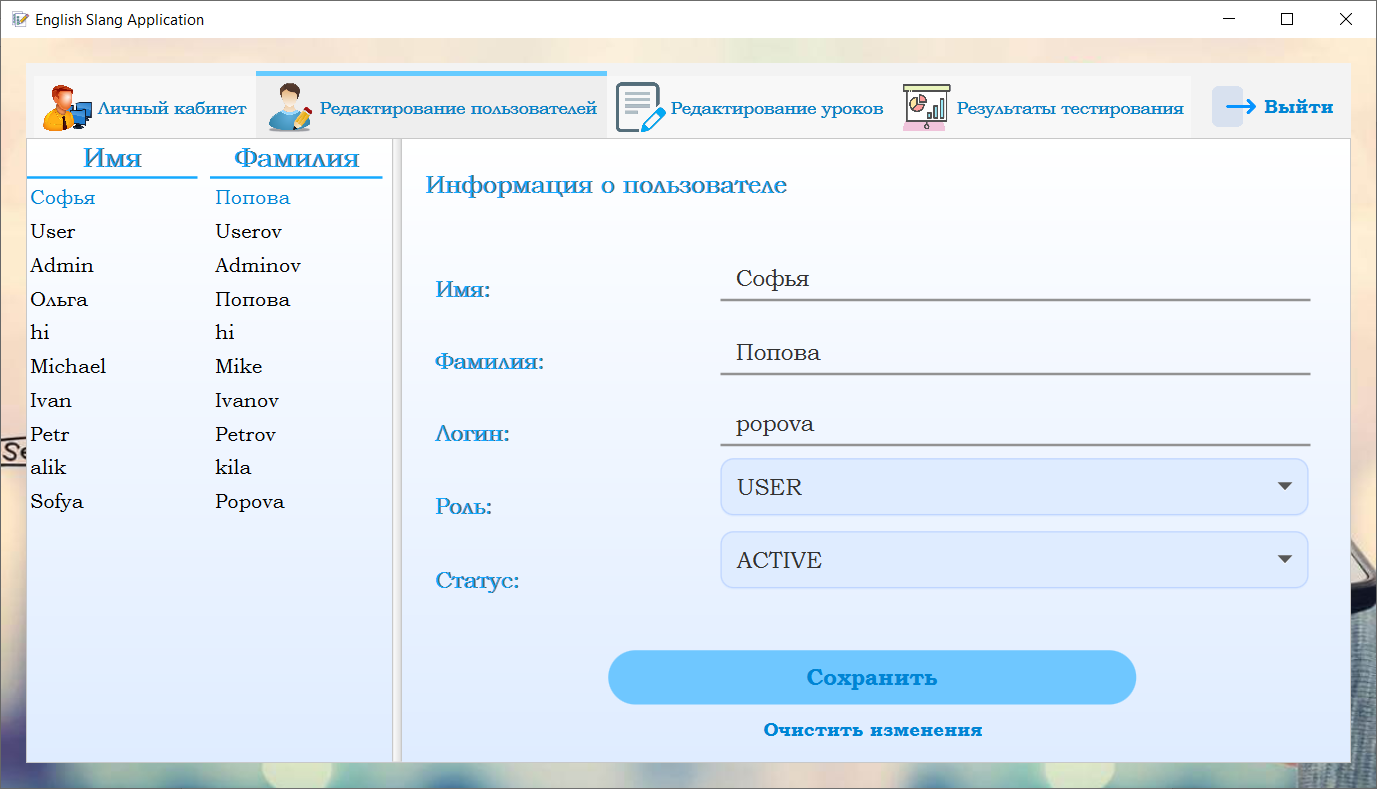
1. Появление следующей формы зависит от того, какая роль у авторизовавшегося пользователя. Сначала проиллюстрирую форму администратора (рис. 3.2.2.3);



*рис. 3.2.2.3*

При входе в систему администратор сразу попадает в свой личный кабинет, где может изменить свои данные (кроме логина, соответственно, это поле подсвечено серым цветом). Сбоку расположена кнопка для отображения справки о том, кто такое администратор и чем он занимается. Сверху расположены вкладки, по которым администратор может спокойно перемещаться. Также есть кнопка выхода из учётной записи.

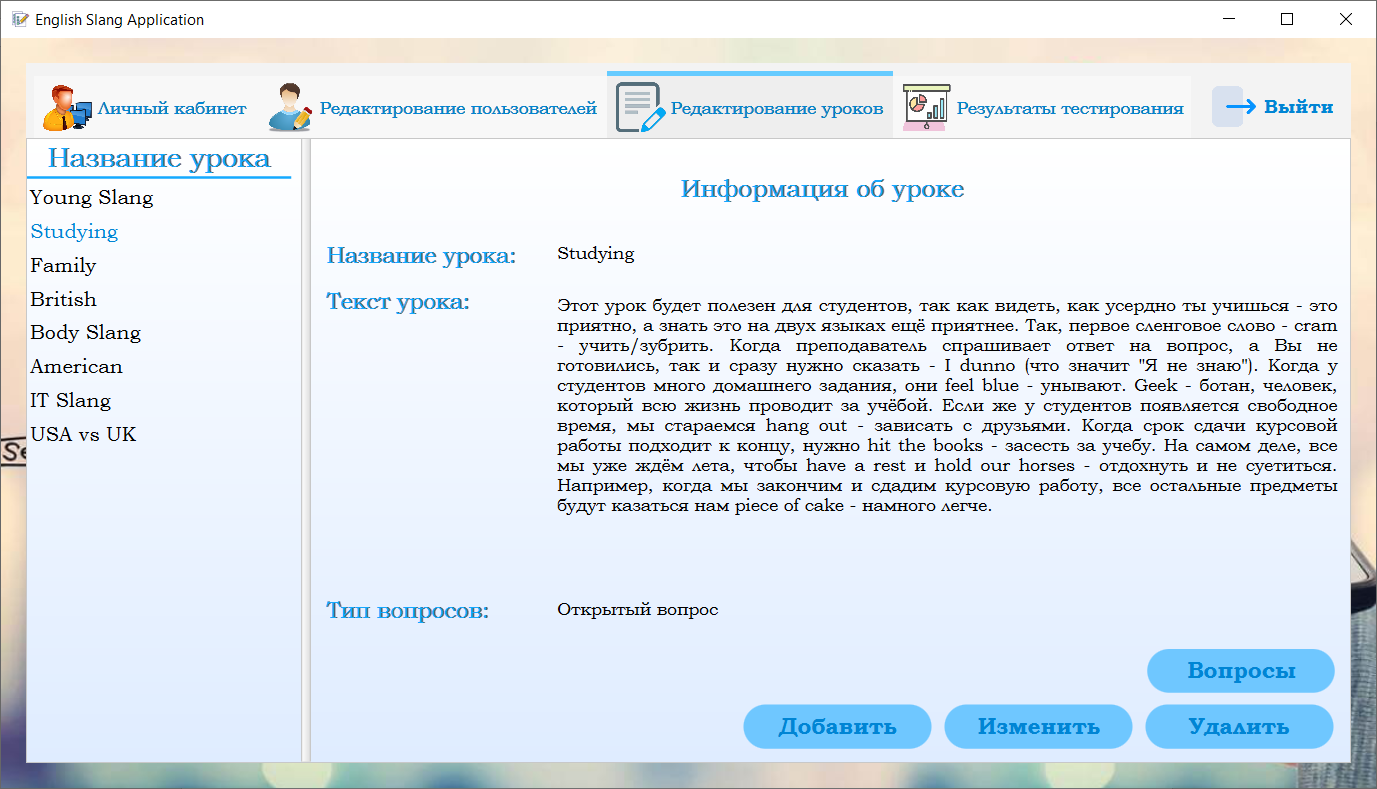
Следующая вкладка администратора – окно редактирования пользователей (рис. 3.2.2.4):



*рис. 3.2.2.4*

Здесь администратор может смотреть информацию о пользователях (кроме паролей) и редактировать им роль (например, при приёме на работу человеку присвоится роль администратора) или статус (если с аккаунта будет замечена подозрительная активность).

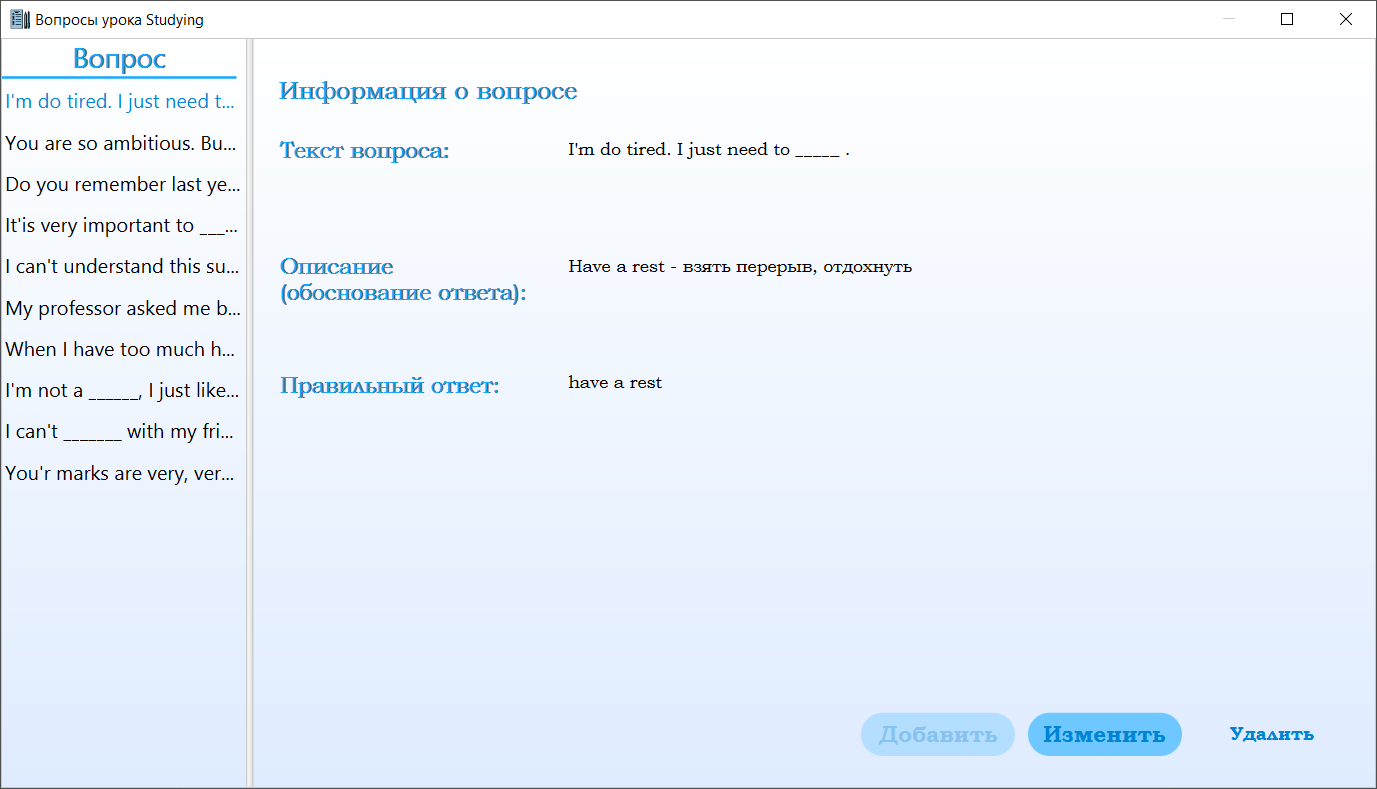
Следующая вкладка – вкладка уроков (рис. 3.2.2.5):



*рис. 3.2.2.5*

Здесь администратор может добавлять/изменять/уроки, а также просматривать вопросы к выбранному уроку.

1. Так, при выборе урока из таблицы и нажатии кнопки «Вопросы», загружается форма вопросов, и администратор может просматривать каждый вопрос (рис. 3.2.2.6);



*рис. 3.2.2.6*

Стоит отметить, что в этой форме кнопка «Добавить» неактивна, так как к этому уроку уже добавлено максимальное количество вопросов – 10 штук. Если удалить одни из них, то кнопка добавления станет активной. Также всегда доступна форма редактирование вопросов (она одна и та же и на добавление, и на изменение, разница лишь в устанавливаемых значениях полей, о чём я говорила ранее).

1. Форма добавления/изменения вопроса (рис. 3.2.2.7);



*рис. 3.2.2.7*

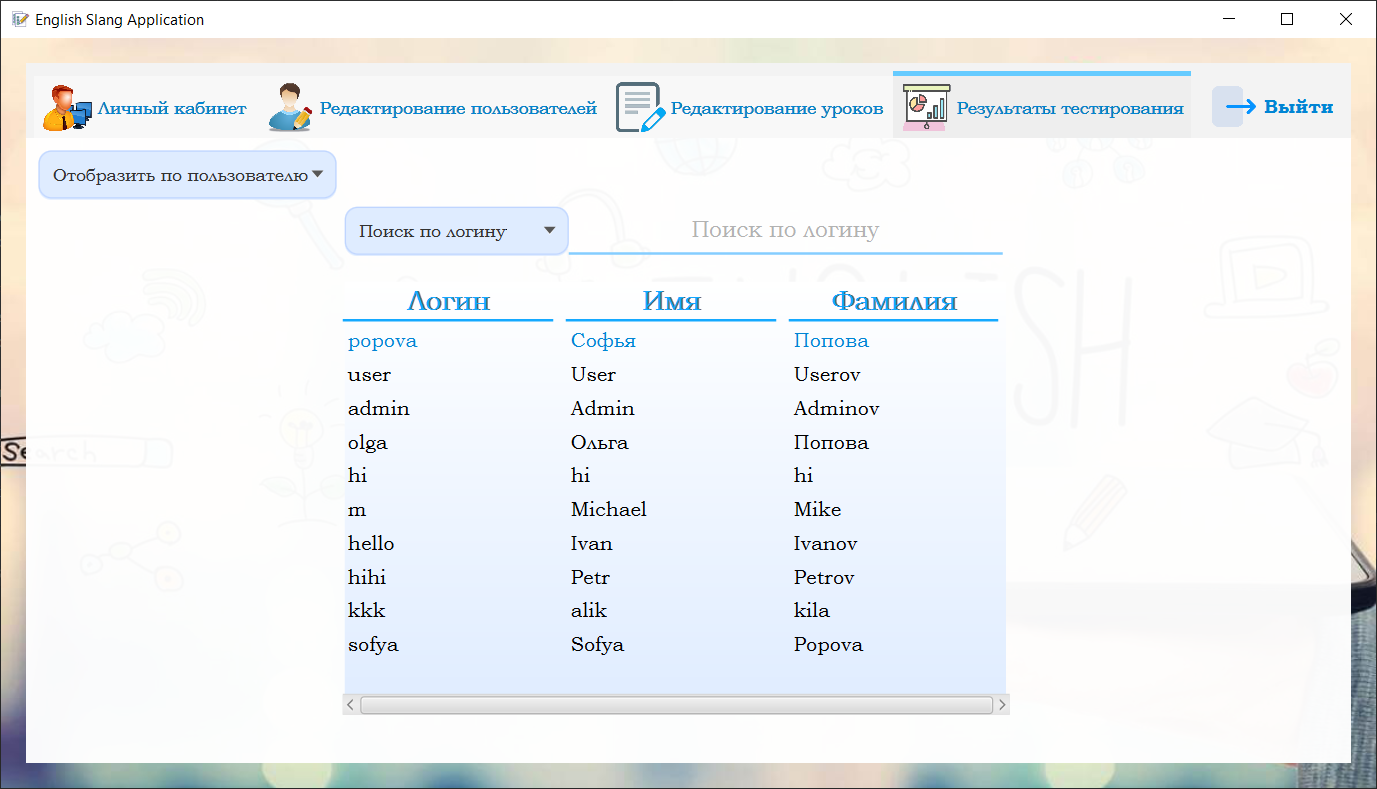
Это форма примечательна тем, что она тоже одна и та же для разных типов вопросов, разница лишь в заполняемых полях. На рис. 3.2.2.8 эта же форма открыта, но для вопроса с четырьмя вариантами ответа:



*рис. 3.2.2.8*

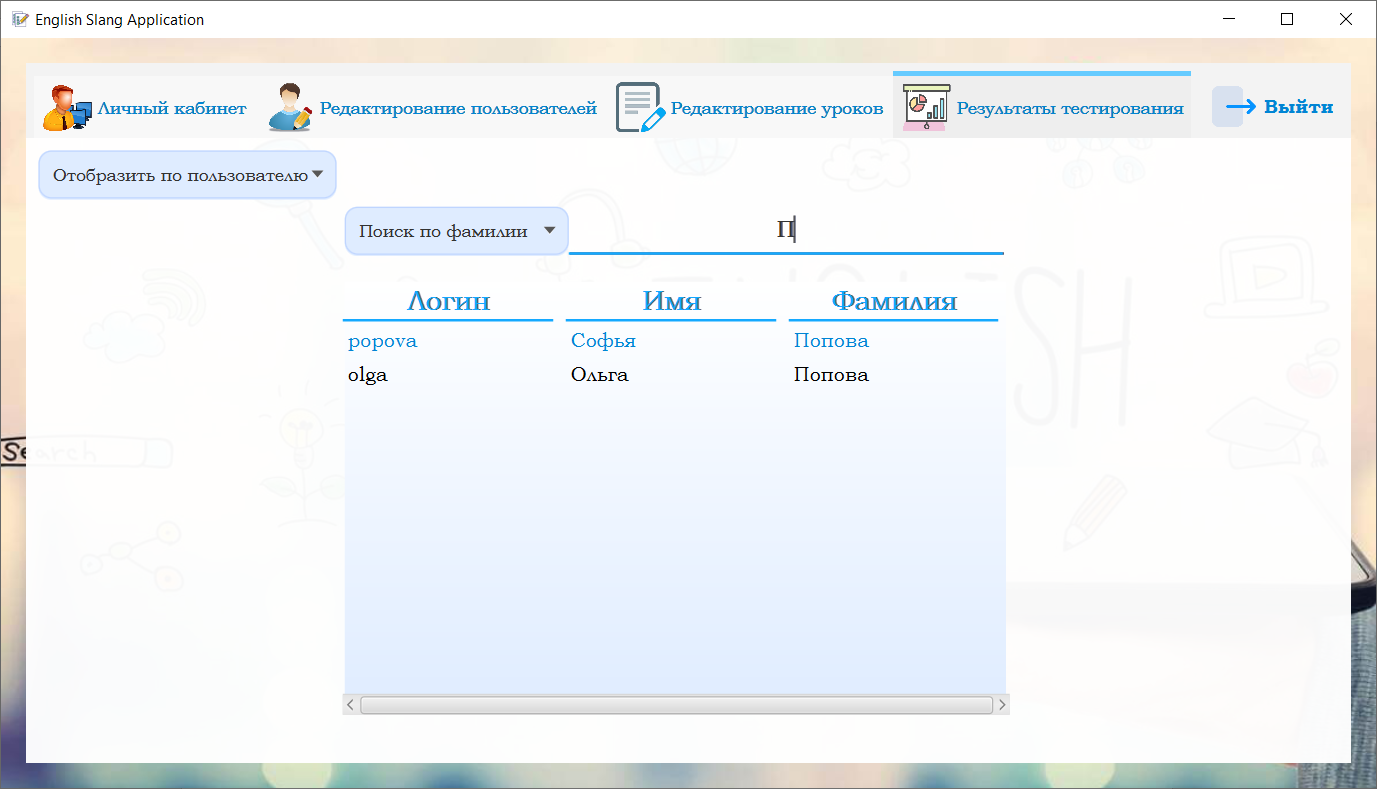
Соответственно, для вопроса с двумя вариантами ответа, форма будет та же самая, но в ней будет на два поля меньше.

Теперь закроем форму редактирования вопросов 3.2.2.8 и форму вопросов урока 3.2.2.6 и вернёмся к окну администратора, к последней вкладке просмотра результатов (рис. 3.2.2.9):



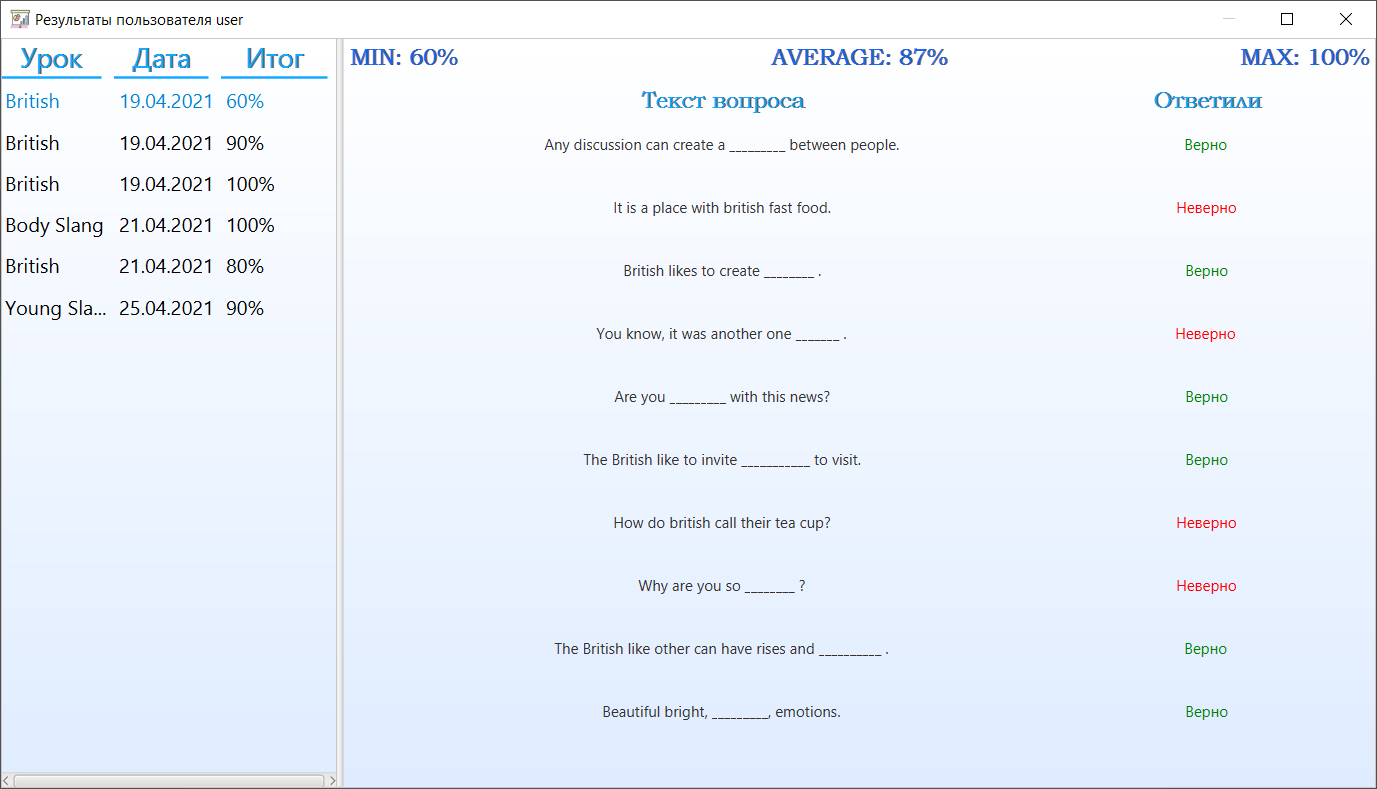
*рис. 3.2.2.9*

На данной вкладке можно просматривать статистику либо по пользователю, либо по уроку, при это реализован динамический поиск по таблице (рис. 3.2.2.10), поиск можно производить по любому выбранному полю:



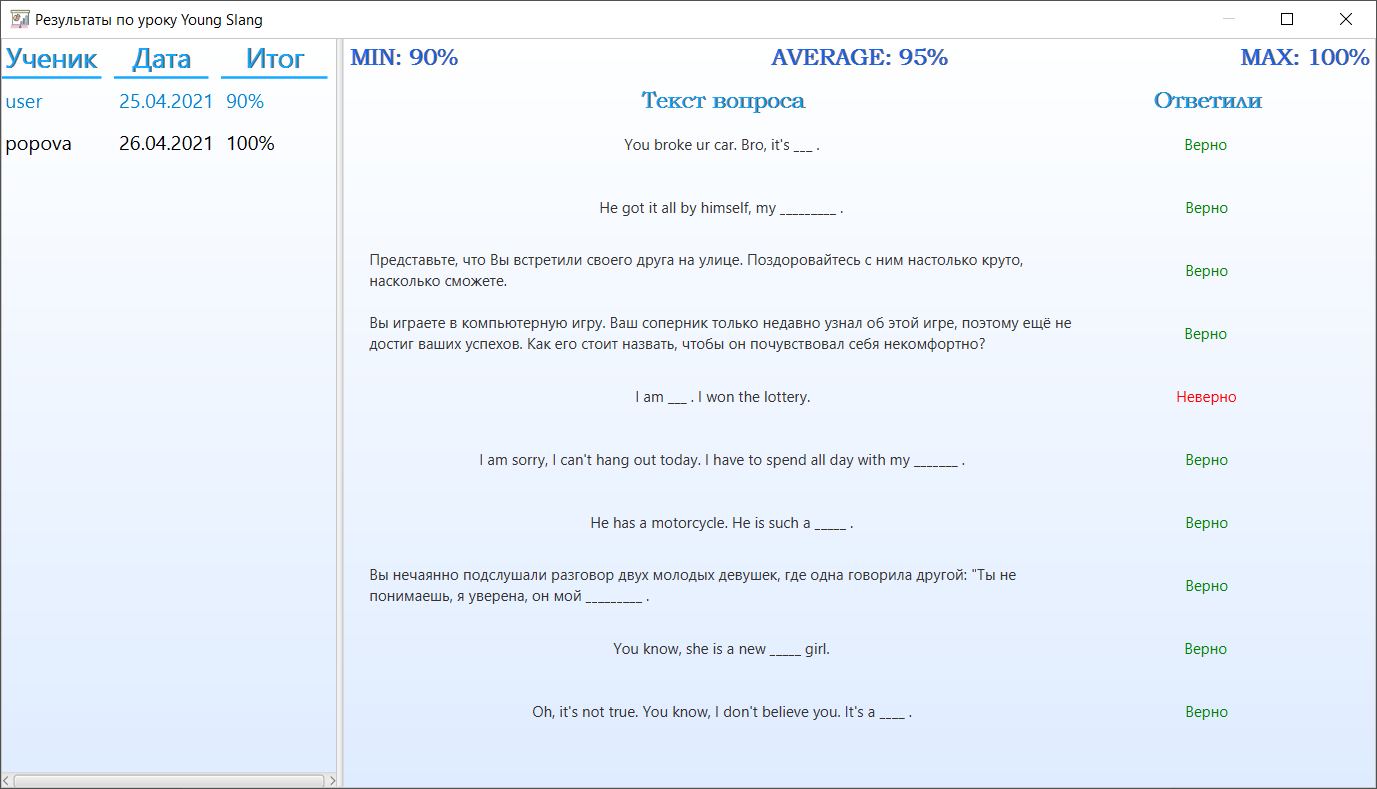
*рис. 3.2.2.10*

Просмотр статистики по пользователю выглядит следующим образом (рис. 3.2.2.11), при этом стоит отметить, что появившиеся сбоку при клике по интересуемому уроку записи – история ответов выбранного пользователя на выбранный урок:



*рис. 3.2.2.11*

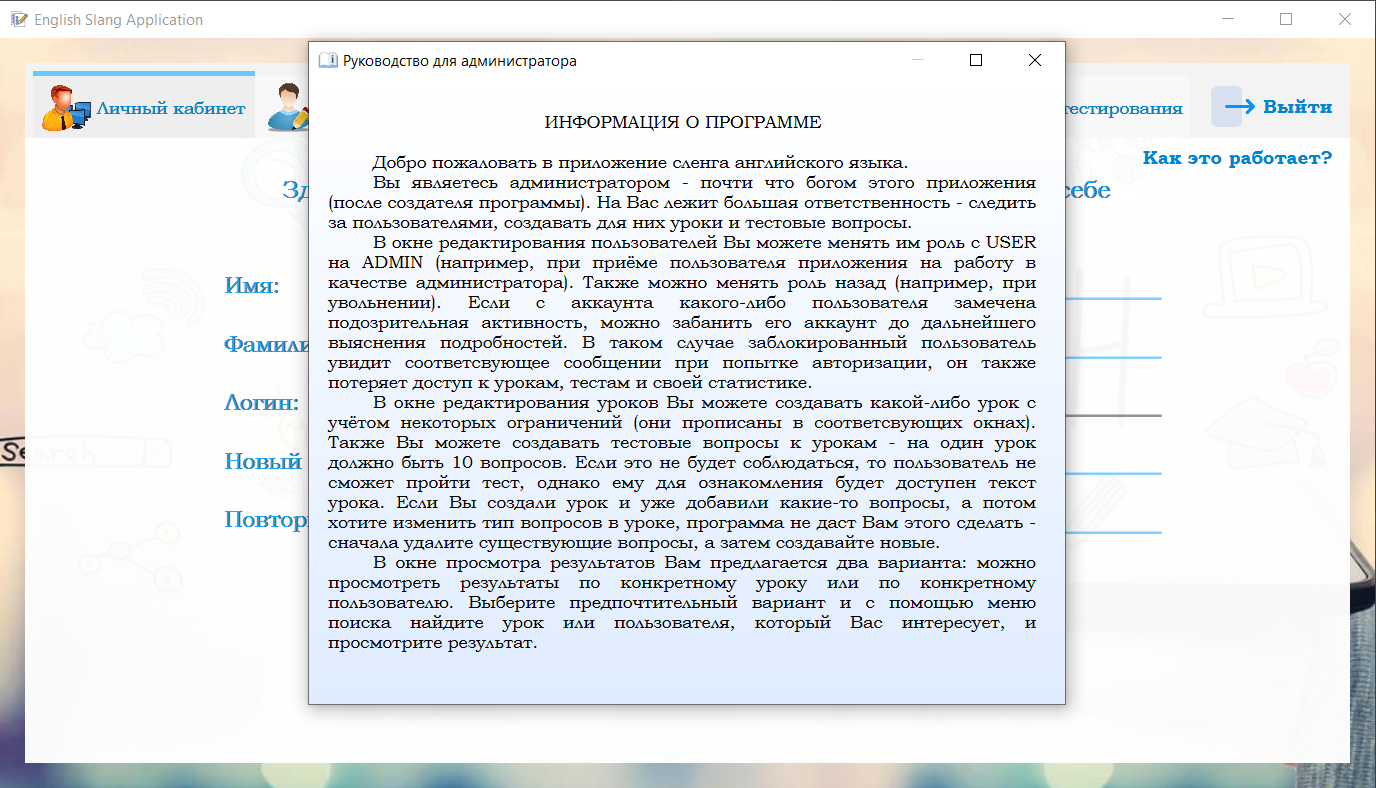
При просмотре статистики по уроку всё то же самое, если выбрать конкретного пользователя, сбоку отобразится его история ответов (рис. 3.2.2.12):



*рис. 3.2.2.12*

При этом в обоих случаях, как можно было заметить, есть возможность отображения статистических данных (минимального результата по пользователю или уроку, среднего и максимального).

1. И последнее, что мы не рассмотрели в окне администратора – открытие справки о программе (стоит отметить, что это одна из многозадачных форм, отображаемых в разных местах с разными данными). Она изображена на рис. 3.2.2.13:



*рис. 3.2.2.12*

На этом окна администратора заканчиваются, перейдём к окну пользователя.