МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Разработка базы данных для программного средства «Тестирование»

Выполнил студент Харевич К. В.

(Ф.И.О.)

Руководитель проекта ст. п. Блинова Е. А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В. В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Реферат

Лит.

Листов

1

БГТУ

Реферат

Пояснительная записка курсового проекта содержит 50 страниц пояснительной записки, 9 источников литературы, 5 приложений.

.NET Framework, WPF, MATERIAL DESIGN, MAHAPPS, ORACLE DATABASE 12c, ORACLE SQL DEVELOPR, ODP.NET, ADO.NET, MD5 HASH.

Цель курсового проекта – разработка базы данных и приложения, которые обеспечат простое и удобное взаимодействие между собой, что позволит администратору создавать любые темы для тестов, а также создавать, изменять и удалять тесты. Пользователь же может проходить тесты, просматривать результаты пройденных тестов, свою статистику за всё время и редактировать свой профиль.

В первой главе проводится аналитический обзор схожих приложений по тематике курсового проекта.

Вторая глава посвящена процессу проектирования системы и описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования проекта.

В четвёртой главе описано тестирование, проверка работоспособности и анализ данных системы.

В заключении приведены результаты проделанной работы.

Содержание

[Реферат 2](#_Toc58858708)

[Введение 4](#_Toc58858710)

[1. Аналитический обзор схожих приложений 5](#_Toc58858711)

[2. Проектирование и разработка базы данных 7](#_Toc58858712)

[2.1. Спецификация функциональных требований 7](#_Toc58858713)

[2.2. Разработка модели базы данных 9](#_Toc58858714)

[2.3. Пользователи 10](#_Toc58858715)

[2.4. Представления 10](#_Toc58858716)

[2.5. Хранимые процедуры 11](#_Toc58858717)

[2.5.1 Выборка данных для представлений 12](#_Toc58858718)

[2.5.2 Проверка корректности введенных пароля и логина 12](#_Toc58858719)

[2.5.3 Проверка существования пользователя 13](#_Toc58858720)

[2.5.4 Добавление данных в таблицы 13](#_Toc58858721)

[2.5.5 Изменение данных в таблицах 13](#_Toc58858722)

[2.5.6 Добавление дополнительных таблиц при создании пользователя 14](#_Toc58858723)

[2.5.7 Экспорт и импорт таблиц в формат xml 14](#_Toc58858724)

[2.6. Индексы 15](#_Toc58858725)

[2.7. Триггеры 16](#_Toc58858726)

[3. Технология полнотекстового поиска 16](#_Toc58858727)

[4. Проектирование и разработка программного средства 19](#_Toc58858728)

[4.1 Проверка работоспособности и анализ данных 21](#_Toc58858729)

[4.2 Тестирование приложения 27](#_Toc58858730)

[Заключение 31](#_Toc58858731)

[Список литературы 32](#_Toc58858732)

[Приложение А 33](#_Toc58858733)

[Приложение Б 34](#_Toc58858734)

[Приложение В 38](#_Toc58858735)

[Приложение Г 47](#_Toc58858736)

[Приложение Д 50](#_Toc58858737)

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Введение

Лит.

Листов

1

БГТУ

## **Введение**

В данной записке приведено описание реляционной базы данных и десктопного приложения, разработанных в соответствии с заданием на курсовое проектирование по теме «Тестирование» по дисциплине «Базы данных».

Задачей данной курсовой работы является разработка базы данных, предназначенной для приложения «Blue Fox Tests». Приложение будет обладать следующими функционалом:

− авторизация и регистрация пользователей;

− разделение пользователя на админа и пользователя;

− создание, выполнение, изменение, удаление тестов из списка;

− изменение профиля пользователя;

− просмотр результатов пройденного теста;

− просмотр статистики выполненных тестов.

Система управления базами данных (СУБД) — комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными. Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

Для разработки и управления базой данных курсового проекта использовалась система управления реляционными базами данных Oracle Database 12c. Из её преимуществ: поддерживает самые большие базы данных, хорошо обрабатывает транзакции. Система сохраняет высокую производительность, в результате чего пользователи не страдают от низкой скорости обработки.

## **Аналитический обзор схожих приложений**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

1. Аналитический обзор схожих приложений

Лит.

Листов

1

БГТУ

Для составления технических требований к проекту был проведен анализ похожих программных средств. Из открытого источника (интернет) были найдены web-приложения для организации тестирования. Одно из них – leetcode.com [1].

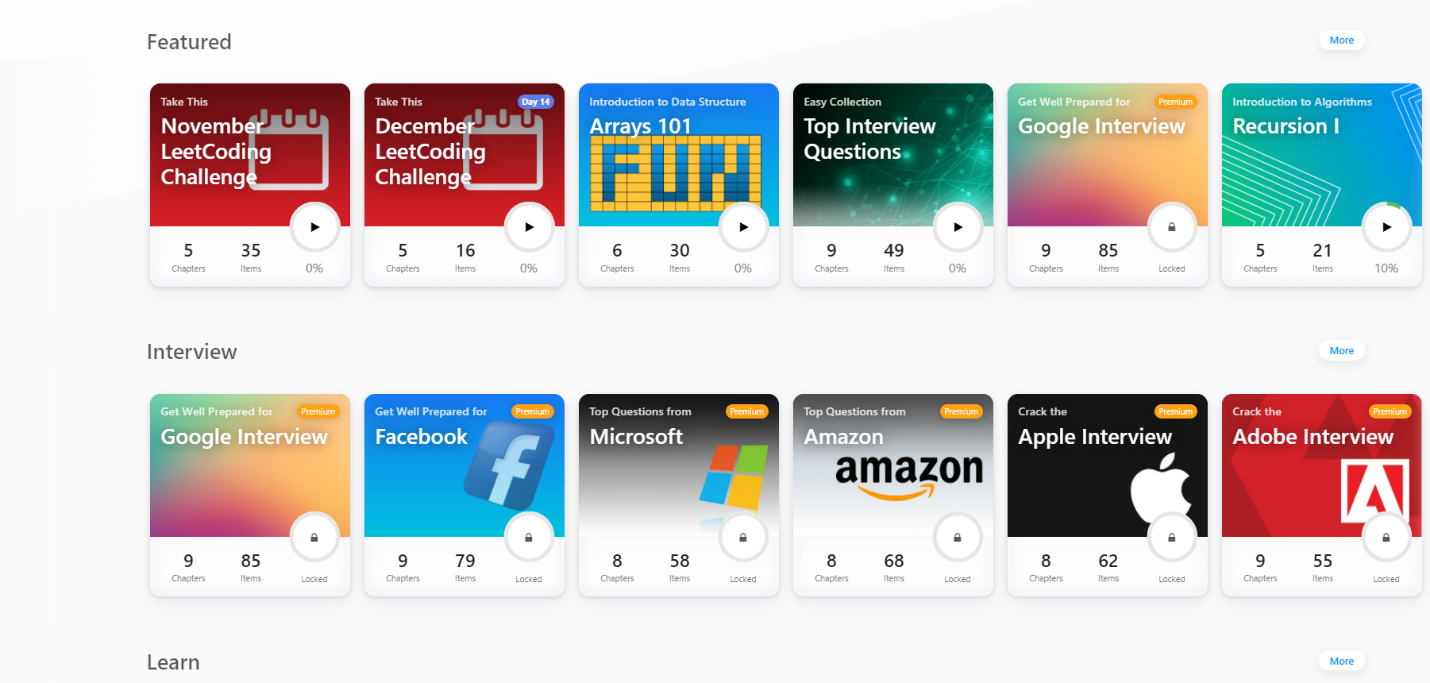
[Leetcode](https://leetcode.com/) – это сайт, где вы можете решать задачи, похожие на те, что задают на собеседованиях. Главное, чему вы можете научиться здесь – это искусство решения проблем и определения закономерностей.

Рисунок 1.1 – Web-приложение leetcode.com

Второй аналог, который я хочу представить, это SunRav. В пакет SunRav TestOfficePro[2] входят программы для создания тестов, проведения тестирования и обработки результатов тестирования. С его помощью можно организовать и провести тестирования и экзамены в образовательных учреждениях (вузы, колледжи, школы), а предприятия и организации могут осуществлять аттестацию и сертификацию своих сотрудников.

Пакет позволяет создавать тесты по предметам школьной программы, вузовским дисциплинам, тесты для профессионального тестирования, психологические тесты и т.д. Окно приложения представлено на рисунке (Рисунок 1.2).

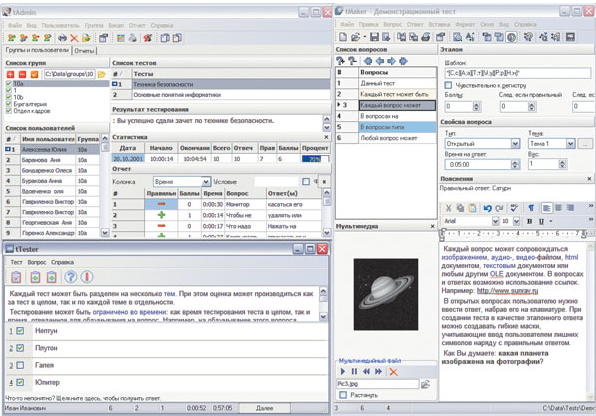


Рисунок 1.2 – «SunRav»

## **Проектирование и разработка базы данных**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

2. Проектирование и разработка базы данных

Лит.

Листов

1

БГТУ

При разработке курсового проекта понадобились следующие объекты:

1. Таблицы;
2. Представления

3. Хранимые процедуры;

4. Индексы.

5. Триггеры

# **Спецификация функциональных требований**

Рассмотрим возможности, которые предоставляет разработанное программное средство. Для наглядности демонстрации использовались UML -диаграммы.

UML [3] – уникальный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

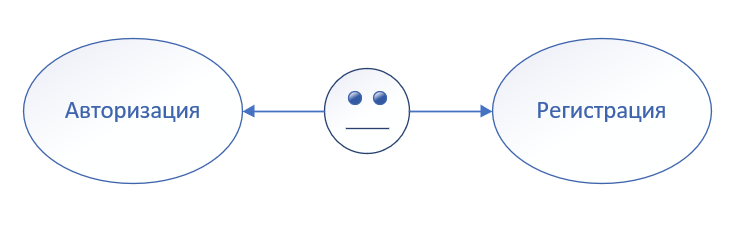
На рисунке 2.1 показана UML-диаграмма, отображающая возможности приложения с точки зрения неавторизованного пользователя.

Рисунок 2.1.1 – UML диаграмма вариантов использования для неавторизованного пользователя

Неавторизованный пользователь имеет только две возможности: авторизоваться и зарегистрироваться.

Если пользователь авторизуется, то будет иметь возможности, представленные на рисунке 2.1.2.



Рисунок 2.1.2 – UML диаграмма вариантов использования для авторизованного пользователя

Авторизованный пользователь может выбирать тему для теста, решать тесты, просматривать их результаты, просматривать личный профиль, статистику по пройденным тестам, меня цветовую схему приложения.

Также в приложение можно войти как администратор. Администратор имеет такие же функции, как и пользователь, а также и недоступные для пользователя, представленные на рис. 2.1.3.



Рисунок 2.1.3 – UML диаграмма вариантов использования для администратора

# **Разработка модели базы данных**

Для реализации базы данных приложения «Blue Fox Tests» было разработано 10 таблиц. Диаграмма базы со структурой связей представлена в приложении А, а в приложении Б представлен скрипт создания БД.

Таблица Users представляет список пользователей:

− User\_Id – идентификатор пользователя,

− Username – имя пользователя, под которым производится вход,

− Password\_Hash– хэшированный пароль,

Таблица Admins представляет список админов, состоит из столбцов:

− Admin\_Id – идентификатор админа,

− User\_Id – идентификатор пользователя

Таблица User\_Info представляет информацию о пользователях:

− User\_Id – идентификатор пользователя,

− Name – имя пользователя,

− Gender – пол (gender),

− Location – место проживания,

− Birthday – дата день рождения,

− Summary – 'информация о себе',

− Education – информация об месте учёбы,

− Work – информация об месте работы.

Таблица User\_Stats представляет информацию о рейсах, состоит из столбцов:

− User\_Id – идентификатор пользователя,

− Right\_Answered – кол-во правильно отвеченных вопросов,

− Total\_Answered – всего отвеченных вопросов,

− Avg\_Score – среднее значение результатов тестов в процентах,

− Finished\_Tests\_Count – кол-во завершённых тестов,

− Passed\_Tests\_Count – кол-во положительно пройденных тестов.

Таблица Themes\_For\_Tests представляет информацию о рейсах, состоит из столбцов:

− Theme\_Id – идентификатор темы,

− Theme\_Namee – название темы.

Таблица Tests представляет информацию о рейсах, состоит из столбцов:

− Test\_Id – идентификатор теста,

− Admin\_Id – идентификатор админа,

− Test\_Name – название теста,

− Theme\_Id – идентификатор темы,

− Time\_Limit\_In\_Minutes – ограничение по времени в минутах,

− Passing\_Score – проходной бал для теста в процентах,

− Is\_Enabled – открыт ли тест для решения.

Таблица Questions\_For\_Tests представляет информацию о рейсах, состоит из столбцов:

− Question\_Id – идентификатор вопроса,

− Test\_Id – идентификатор теста,

− Question\_Number – номер вопроса в тесте,

− Question – текст вопроса.

Таблица Answers\_For\_Tests представляет информацию о рейсах, состоит из столбцов:

− Answer\_Id – идентификатор ответа,

− Answer – текст ответа,

− Is\_Right – правильный ли ответ,

− Question\_Id – идентификатор вопроса.

Таблица Test\_Result представляет информацию о рейсах, состоит из столбцов:

− Result\_Id – идентификатор результата,

− User\_Id – идентификатор пользователя,

− Test\_Id – идентификатор теста,

− Try\_Count – номер попытки для теста,

− Is\_Passed – положительно ли пройден тест,

− Score – баллы в процентах,

− Right\_Answers\_Count – кол-во правильно отвеченных вопросов,

− Questions\_Count – кол-во вопросов,

− Start\_Date – дата начала теста,

− End\_Date – дата конца теста.

Таблица User\_Answers представляет информацию о рейсах, состоит из столбцов:

− Result\_Id – идентификатор результата,

− Question\_Id – идентификатор вопроса,

− User\_Answer – идентификатор ответа.

# **Пользователи**

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации.

При проектировании базы данных были использованы 2 пользователя. Первый пользователь – bluefox\_user – обычный пользователь, имеет доступ для чтения таблиц, права для вставки в некоторые таблицы. Ограниченный список доступных процедур.

Второй пользователь – bluefox\_admin – обладает более обширным перечнем прав. Имеет права для чтения, изменения и записи во всех таблица. Полный список доступных процедур.

# **Представления**

Представление[4], или виртуальная таблица в базе данных Oracle Database – это специфический образ таблицы или набора таблиц, определенный оператором SELECT. Представление не существует физически как обычная таблица, являющаяся частью табличного пространства. Фактически представление создает виртуальную таблицу или подтаблицу только с теми строками и/или столбцами, которые нужно показать пользователю.

Представление Oracle – результат хранимого запроса, поэтому в словаре данных сохраняется только определение представления.

Представления Oracle используются в приложениях по нескольким причинам, включая перечисленные ниже:

* уменьшение сложности;
* повышение безопасности;
* повышение удобства;
* переименование столбцов таблицы;
* настройка данных для пользователей;
* защита целостности данных.

В процессе выполнения курсового проекта были созданы представления для каждой таблицы.

Представления базы данных: USERS\_VIEW, ADMINS\_VIEW, STATS\_VIEW, INFO\_VIEW, THEMES\_VIEW, TESTS\_VIEW, QUESTIONS\_VIEW, ANSWERS\_VIEW, TEST\_RESULT\_VIEW, USER\_ANSWERS\_VIEW (рисунок 2.1).

Скрипты для создания всех представлений базы данных находятся в приложении Б.

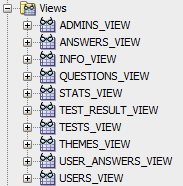


Рисунок 2.4 – Представления

# **Хранимые процедуры**

Хранимая процедура[5] – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Процедура представляет собой модуль, выполняющий одно или несколько действий.

При разработке курсового проекта было создано 34 процедуры для следующих целей:

1. Выборка данных из представлений;

2. Проверка корректности введенных пароля и логина;

3. Проверка существования пользователя;

4. Добавление данных в таблицы;

5. Изменение данных в таблицах;

6. Добавление дополнительных таблиц при создании пользователя;

7. Экспорт и импорт таблиц в формат xml;

Весь перечень созданных процедур будет представлен в Приложении В.

# **Выборка данных для представлений**

Для вывода данных из представлений были написаны следующие процедуры: GET\_USERS, GET\_USER\_BY\_USERNAME, GET\_USER\_BY\_ID, GET\_USER\_WITH\_INFO, GET\_USER\_INFO, GET\_THEMES, GET\_THEME\_BY\_NAME и другие. Поскольку вывод таблицы размером более 100 000 строк занимает довольно много времени, было принято решение создать индексы, что позволило уменьшить время вывода таблицы до минимума.

# **Проверка корректности введенных пароля и логина**

При регистрации нового пользователя с помощью процедуры AddUser, ему необходимо заполнить все необходимые поля. Для увеличения безопасности учетной записи пользователя его пароль находится в захэшированном виде. Хеширование пароля происходит за счет вызова функции MD5.ComputeHash в приложении.

Процедура CHECK\_USER\_LOGIN проверяет корректность логина и пароля, и, в случае корректно введенных данных, возвращает курсор на найденного пользователя. Данная процедура представлена ниже на рисунке 2.5.2.

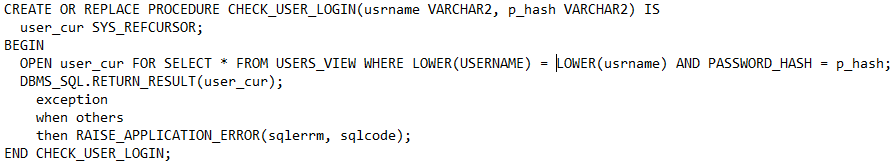


Рисунок 2.5.2 – Пример создания процедуры CHECK\_USER\_LOGIN

# **Проверка существования пользователя**

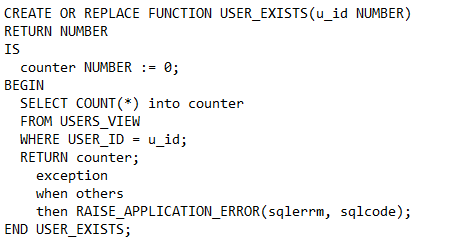
При заполнении полей на этапе регистрации пользователя существует возможность отобразить сообщение об уже существующем имени пользователя. Для этого используется процедура USER\_EXISTS. Данная процедура представлена ниже на рисунке 2.5.3.

Рисунок 2.5.3 – Пример создания процедуры USER\_EXISTS

# **Добавление данных в таблицы**

Для управления базой данных были созданы процедуры добавления новых строк: ADD\_USER, ADD\_USER\_INFO, ADD\_THEME, ADD\_TEST, ADD\_QUESTION, ADD\_ANSWER, ADD\_USER\_ANSWER. Они были разработаны для заполнения соответствующих таблиц.

# **Изменение данных в таблицах**

Для управления базой данных были созданы процедуры обновления: UPDATE\_USER\_INFO и UPDATE\_TEST\_RESULT\_TEST\_END. Они были разработаны для обновления данных в соответствующих таблицах.

# **Добавление дополнительных таблиц при создании пользователя**

При регистрации пользователя должны создаваться прилегающие к нему таблиц информации и статистики пройденных тестов. Для этого была разработана процедура INITIALIZE\_USER\_INFO\_AND\_STATS (рисунок 2.5.6). Данная процедуры используется в купе с триггером USER\_INFO\_INIT, о котором будет рассказано подробнее в следующих пунктах.

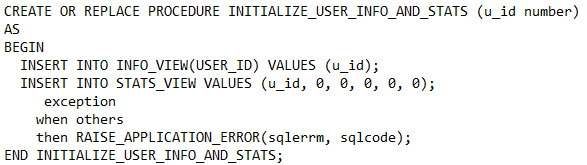


Рисунок 2.5.6 – Пример создания процедуры INITIALIZE\_USER\_INFO\_AND\_STATS

# **Экспорт и импорт таблиц в формат xml**

XML — это формат данных, используемый для обмена данными в форме, которая может быть легко использована и распространена. Часто возникает необходимость импортировать и экспортировать XML-файлы в Oracle, в данной курсовой работе используются пакеты DBMS\_XMLPARSER[6] и DBMS\_XSLPROCESSOR для импорта xml, и DBMS\_XMLDOM для экспорта данных в xml формат. Для работы с файлами так же использовался пакет DBMS\_LOB. Данные пакеты были выбраны, потому что они обладают очень широким функционалом и гибкой настройкой xml документа. В данном курсовом проекте функции экспорта, импорта используются для таблиц Users, User\_Info и User\_Stats. Процедуры для экспорта и импорта представлены в приложении Г. Пример выходного xml файла можно посмотреть на рисунке 2.5.7.

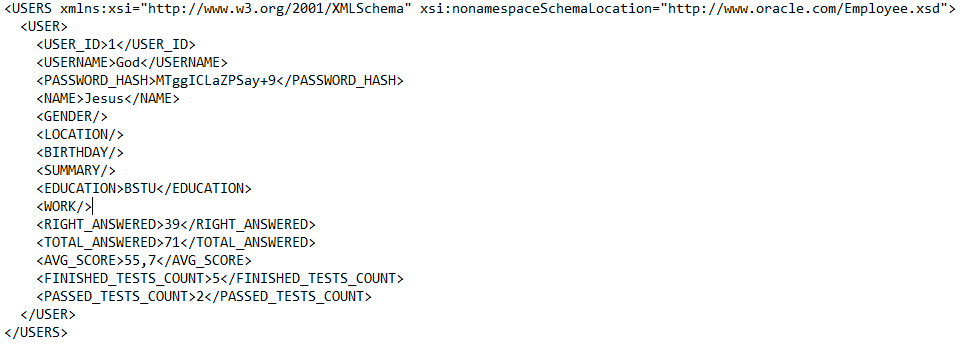


Рисунок 2.5.7 – полученный xml файл после экспорта.

# **Индексы**

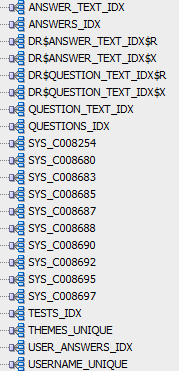
Индекс [7] – объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. В связи с необходимостью выборки таблиц, заполненных минимум на 100 000 строк, было принято решение разработать индексы (рисунок 2.6).

Рисунок 2.6 – Индексы базы данных

# **Триггеры**

Триггер[8] — [хранимая процедура](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0) особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определённом столбце заданной таблицы [реляционной базы данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Триггеры применяются для обеспечения целостности данных и реализации сложной [бизнес-логики](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0). Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в [транзакции](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

В данном курсовом проекте для таблиц были реализованы AFTER триггеры USER\_INFO\_INIT и UPDATE\_USER\_STATS\_ON\_TEST\_END, реагирующие на операции INSERT и UPDATE соответственно. USER\_INFO\_INIT триггер срабатывает при добавлении пользователя в таблицу Users, он вызывает процедуру INITIALIZE\_USER\_INFO\_AND\_STATS, которая инициализирует таблицы User\_Info и User\_Stats для нового пользователя. UPDATE\_USER\_STATS\_ON\_TEST\_END триггер срабатывает при изменении строки в таблице User\_Result и обновляет статистику пользователя в соответствии с полученным результатом теста. Все скрипты триггеров находятся в приложении Д.

## **Технология полнотекстового поиска**

Полнотекстовый поиск [9] – поиск документов в базе данных текстов с использованием полнотекстового индекса. Полнотекстовый индекс – это индекс, в котором перечислены все слова, встречающиеся в тексте, и указаны позиции, на которых эти слова встречаются. Полнотекстовый индекс позволяет производить поиск очень быстро, в отличие, например от поиска при помощи оператора LIKE. Кроме того, системы полнотекстового поиска обычно включают в себя функционал, расширяющий возможности поиска, например поиск с учетом морфологии языка, синонимов, созвучных слов, с учетом меры близости между словами в запросе и другие.

Для реализации данной технологии была использована компонента ORACLE TEXT.

Oracle Text[9] – это бесплатная компонента, входящая в состав Oracle Database, предназначенная для создания приложений с использованием полнотекстового поиска. В предыдущих версиях Oracle имела также названия ConText и interMedia Text. Начиная с 11-ой версии входит во все редакции СУБД.

Oracle Text позволяет строить полнотекстовые индексы 4-ёх типов:

* CONTEXT - основной тип индекса. Применяется для индексирования содержимого одной или нескольких колонок таблицы;
* CTXCAT - применяется для индексации небольших текстовых фрагментов, не требует периодической переиндексации;
* CTXRULE - применяется при создании приложений для классификации документов;
* CTXXPATH - предназначен для ускорения работы с полями типа XMLType.

В данном курсовом проекте был выполнен полнотекстовый поиск с применением индекса CTXCAT для текста вопроса и ответа, так как этот индекс идеально подходит для не слишком больших строк. Процесс создания индекса представлен на рисунке 3.1

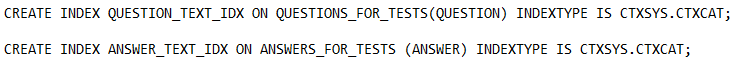


Рисунок 3.1 – Создание индексов CTX

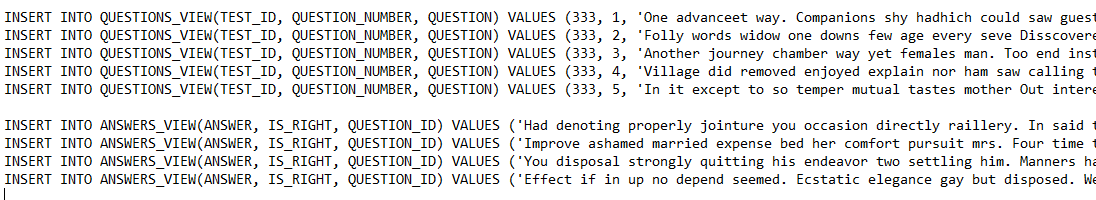
Также следует добавить записи с текстом в таблицы вопросов и ответов (Рисунок 3.2).

Рисунок 3.2 – Добавление записей в таблицы вопросов и ответов

Теперь сравним время поиска, использую функцию CATSEARCH (рисунок 3.3) и LIKE (рисунок 3.4).

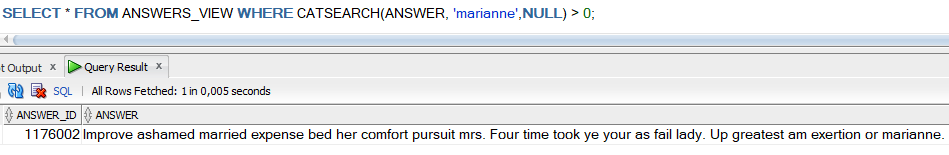


Рисунок 3.3 – Поиск через CATSEARCH

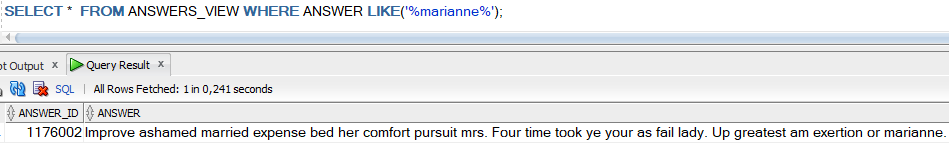


Рисунок 3.4 – Поиск через LIKE

Из полученных результатов мы можем сделать вывод о том, что полнотекстовый поиск работает быстрее, что поможет нам оптимизировать работу нашей базы данных.

Также стоит отменить то, что полнотекстовый поиск будет находить похожие слова, в отличии от LIKE, что позволит пользователю находить похожие слова, если он забыл точную формулировку.

## **Проектирование и разработка программного средства**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

3. Проектирование и разработка программного средства

Лит.

Листов

1

БГТУ

В процессе выполнения курсового проекта было разработано десктопное приложение, предназначенное для взаимодействия пользователя с базой данных. Приложение было разработано на языке программирования C# с использованием таких технологий как .NET Framework, WPF, MATERIAL DESIGN, MAHAPPS, ODP.NET, ADO.NET, MD5 HASH. Структура разработанного программного средства представлена на рисунке 4.1.

Проект включает в себя следующие окна:

* SignUpWindow – окно регистрации пользователя;
* LoginWindow – окно авторизации;
* MainWindow – основное окно, в которое пользователь попадает при успешной авторизации. Здесь находится весь функционал, распределённый по разным вкладкам.

Для взаимодействия через приложения с данными из БД была применена технология ADO.NET [10] совместно с драйвером ODP.NET [11]. Oracle Data Provider for .NET (ODP.NET) обеспечивает оптимизированный доступ к данным ADO.NET в базе данных Oracle. ODP.NET позволяет разработчикам использовать преимущества расширенной функциональности базы данных Oracle, включая реальные кластеры приложений, самонастраивающийся кэш операторов и быстрое переключение при отказе соединения. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций. Схематичная архитектура ADO.NET представлена на рисунке 4.2.

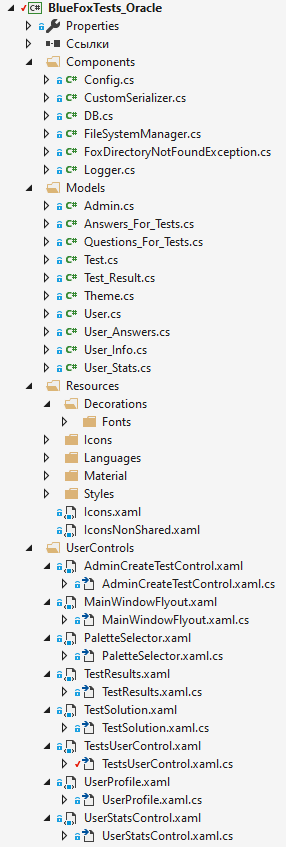
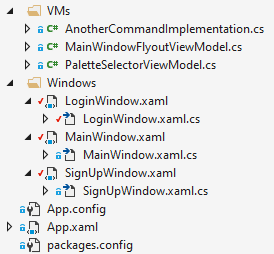


Рисунок 4.1 – Структура проекта



Рисунок 4.2 – Архитектура ADO.NET

Для установки соединения с нашей базой данных нам необходимо было указать в конфигурации проекта строку подключения, куда мы передаем такие параметры как Data Source, User ID, Password.

# **Проверка работоспособности и анализ данных**

Установка программы.

Перед установкой программы прежде всего нужно установить Oracle Database и выполнить команды создания таблиц.

После удачного выполнения команд можно приступать к запуску приложения. Окно после запуска представлено на рисунке 4.1.1.

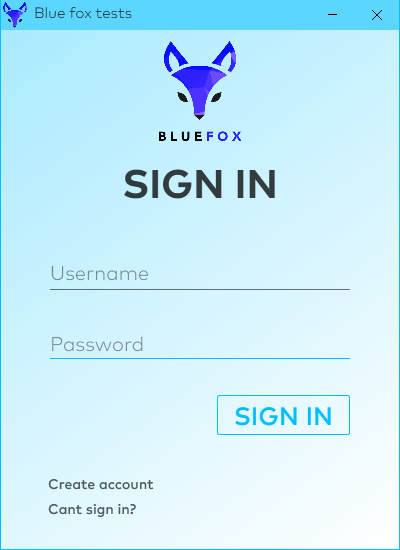


Рисунок 4.1.1 – Авторизация

Для регистрации следует перейти в окно Registration (Рисунок 4.1.2).

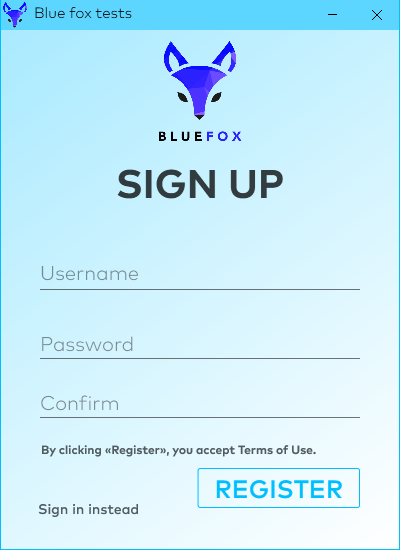


Рисунок 4.1.2 – Регистрация

После регистрации и авторизации пользователь попадает в главное окно программы “Тесты” (Рисунок 4.1.3). Для того, чтобы выбрать тесты из определённой темы надо выбрать тему и нажать “Show Tests”.

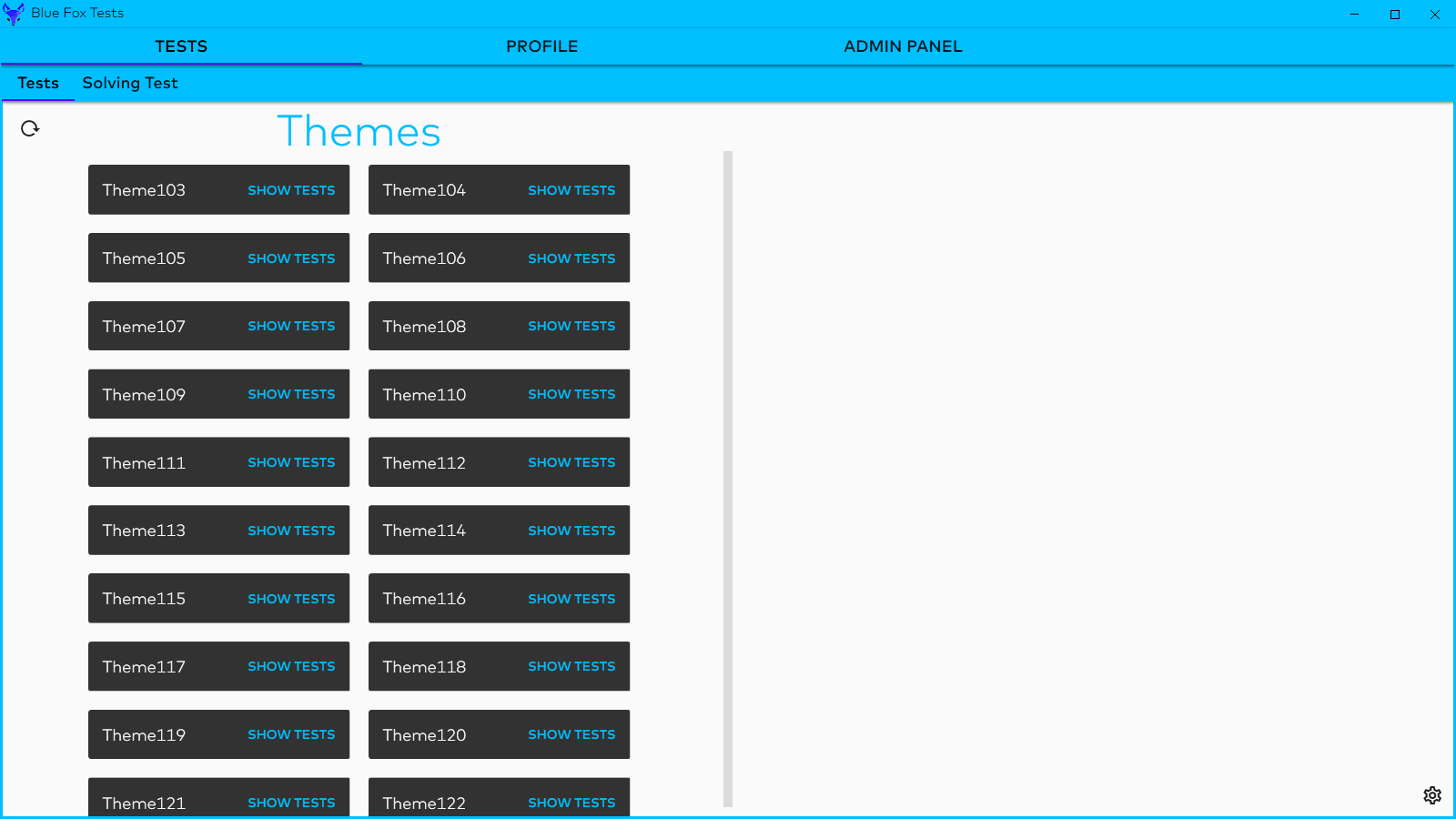


Рисунок 4.1.3 – Вкладка Тесты

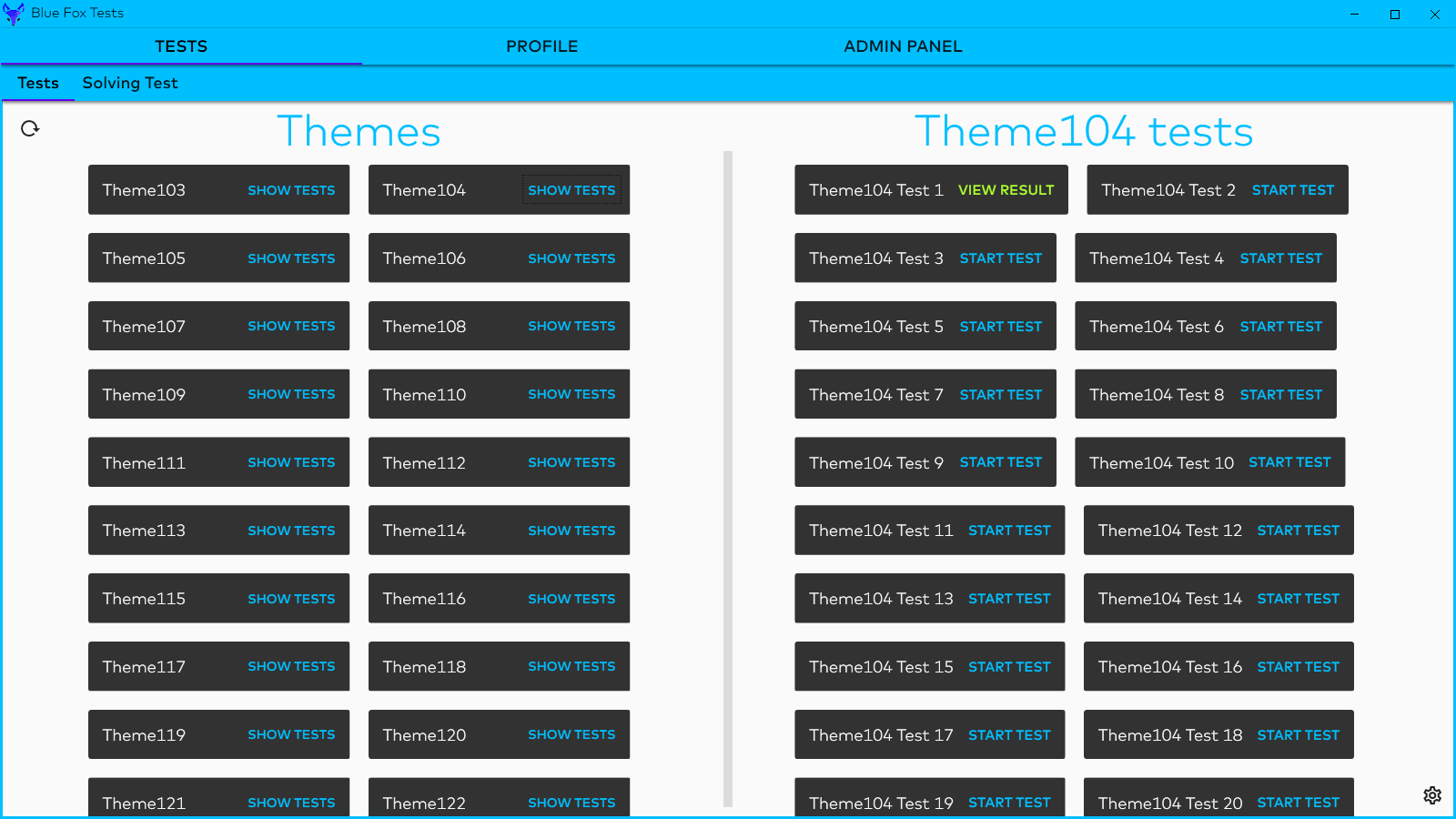
Для загрузки теста нажимаем “Start Test” и начинаем выполнение теста во вкладке “Solving Test” (Рисунок 4.1.4).

Рисунок 4.1.4 – Выбор теста

После загрузки теста во вкладке “Solving Test” появляется информация о загруженном тесте (Рисунок 4.1.5). Нажав кнопку “Start”, вы начнёте тест (Рисунок 4.1.6).

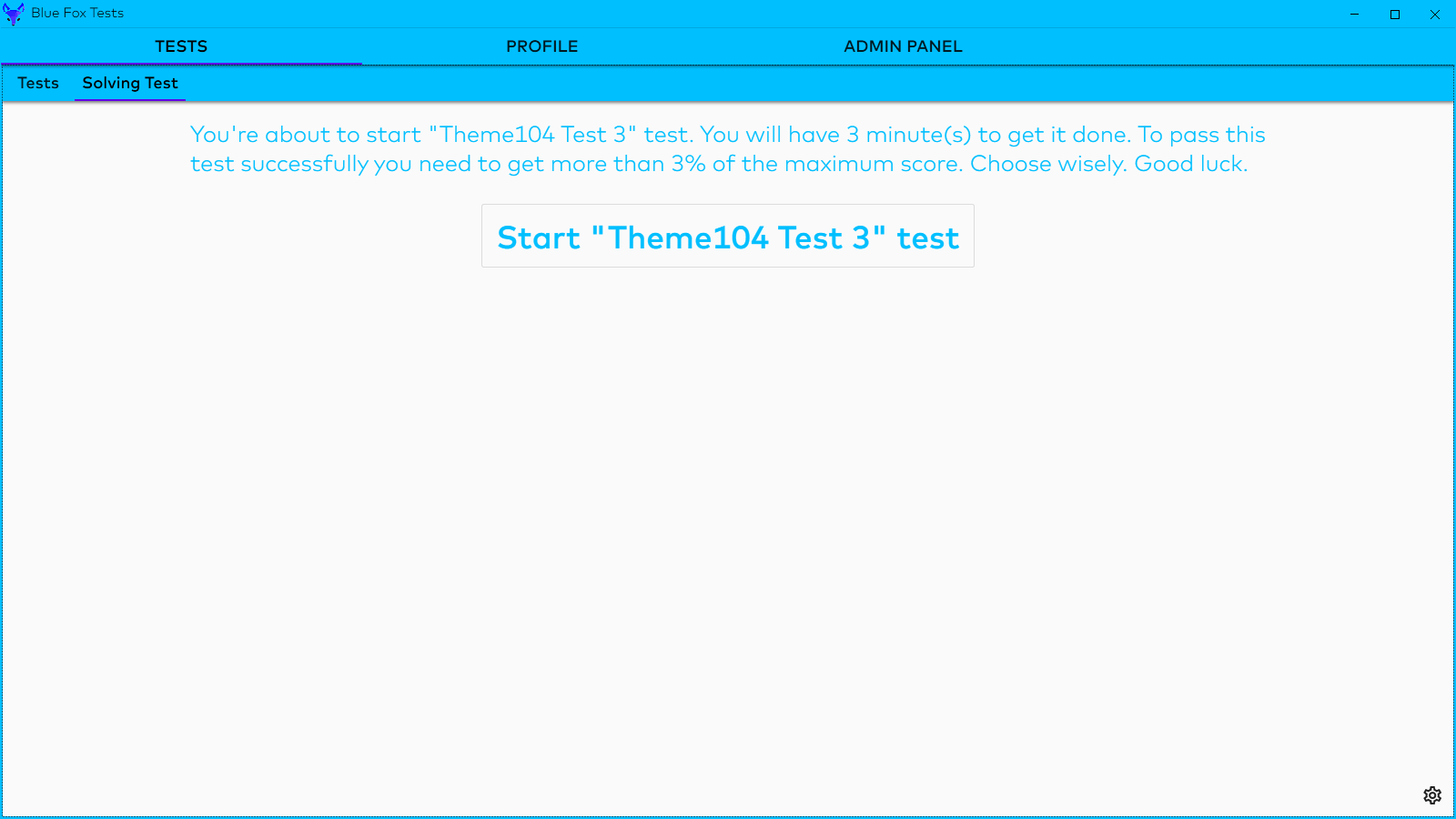


Рисунок 4.1.5 – Превью теста

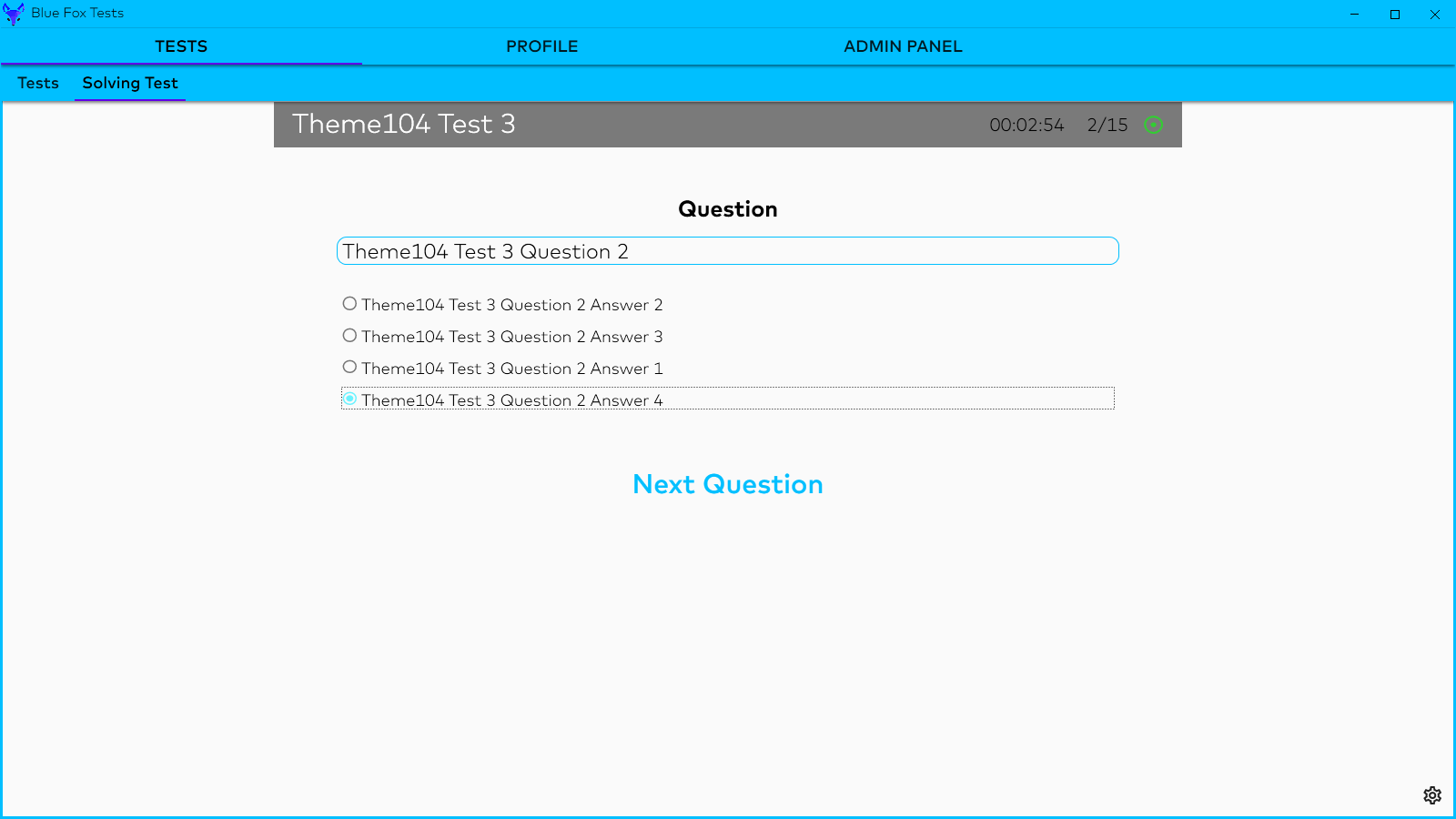


Рисунок 4.1.6 – Решение теста

После окончания теста появляется окно результатов. (Рисунок 4.1.7).

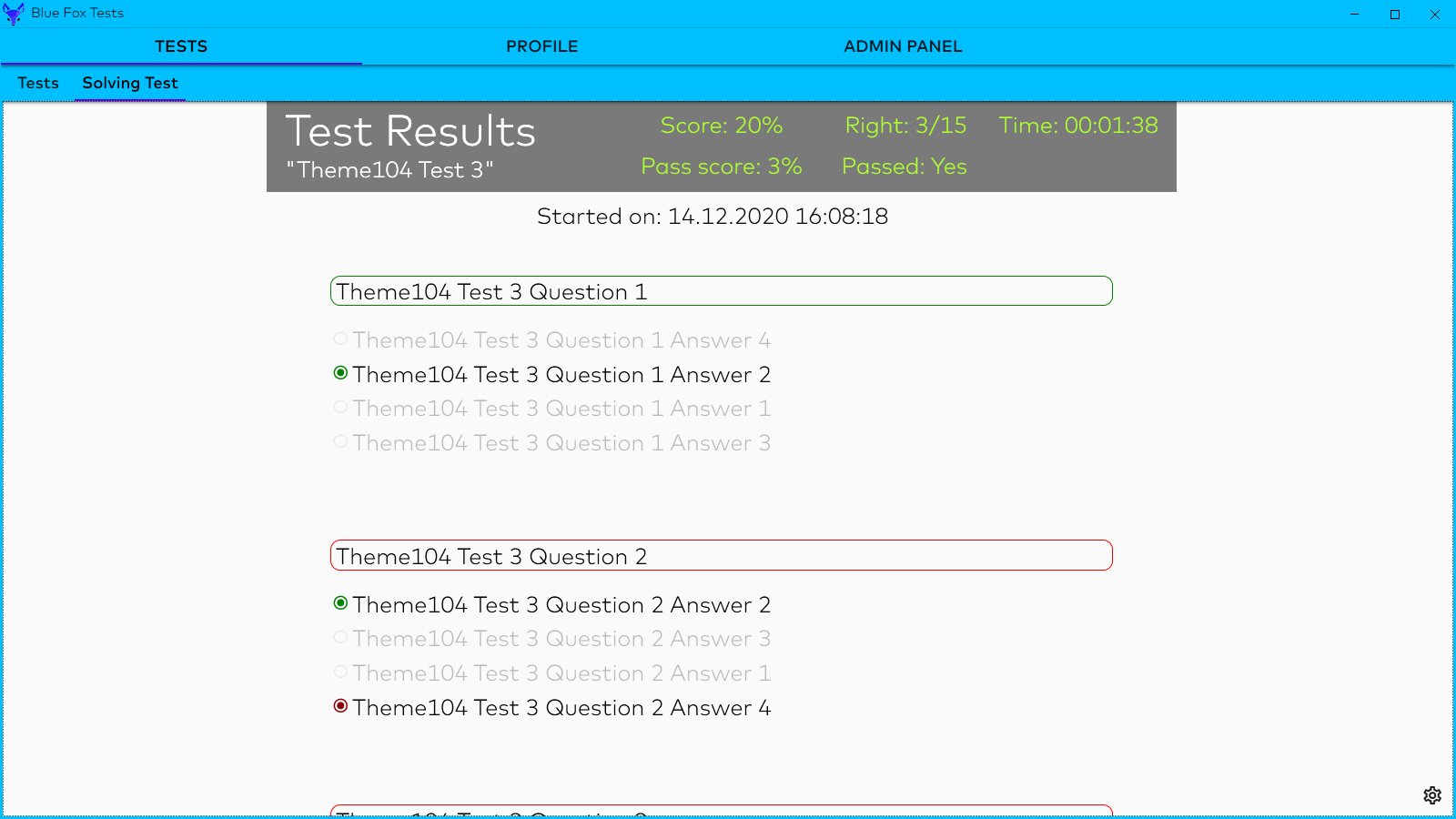


Рисунок 4.1.7 – Результат теста

Пункт меню «Profile».

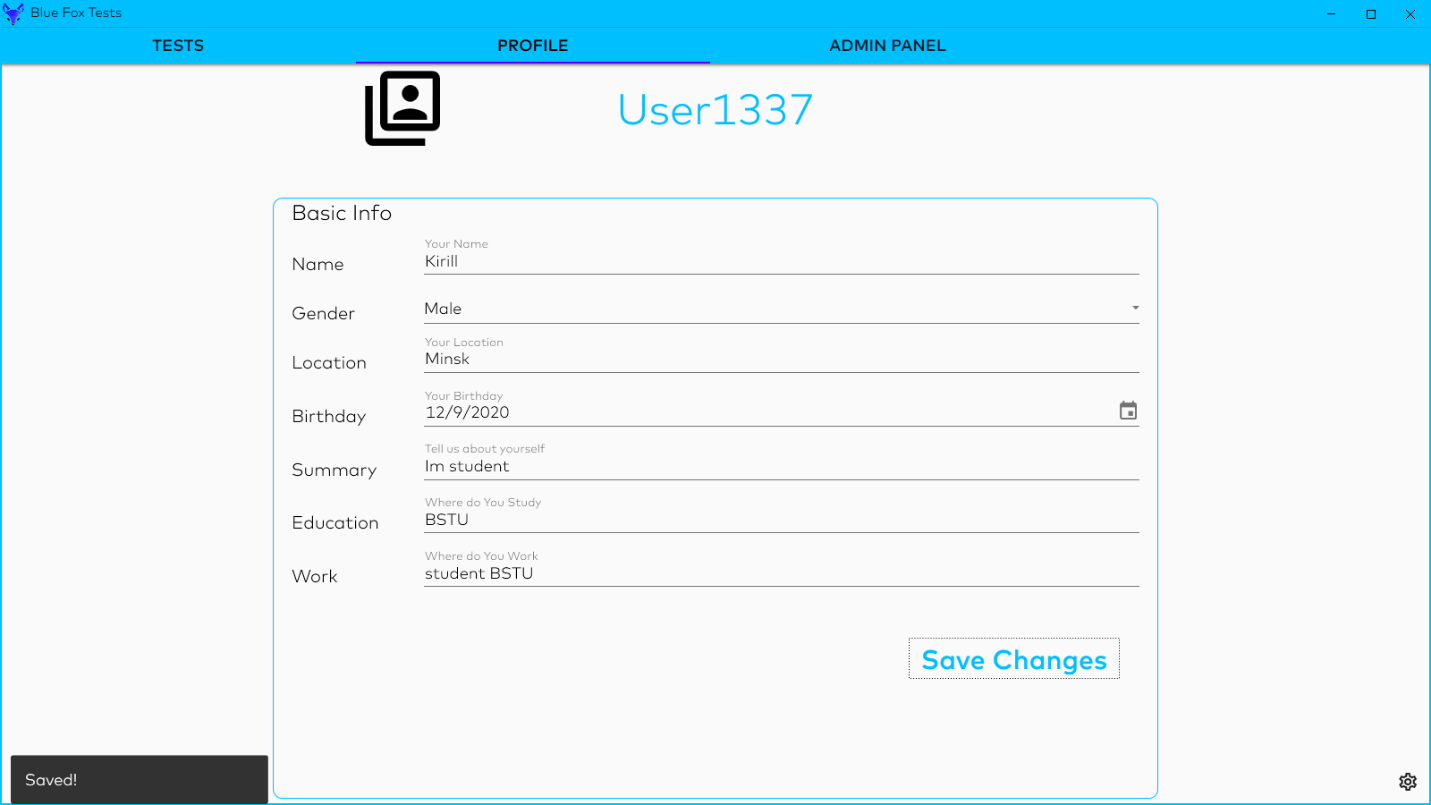
При открытии пункта меню можно просмотреть текущие данные о пользователи. Также их можно редактировать с последующим сохранением в базу данных (Рисунок 4.1.8).

Рисунок 4.1.8 – Профиль пользователя

Выпадающее меню.

В этом меню мы можем поменять тему приложения между темной/светлой, выйти из учётной записи (рисунок 4.1.9) и просмотреть статистику текущего пользователя (рисунок 4.1.10).

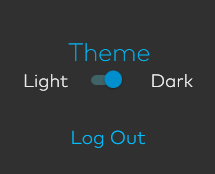
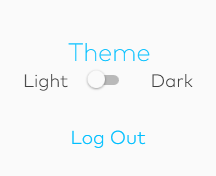


Рисунок 4.1.9 – Дополнительное меню

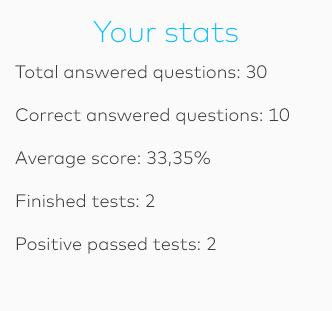
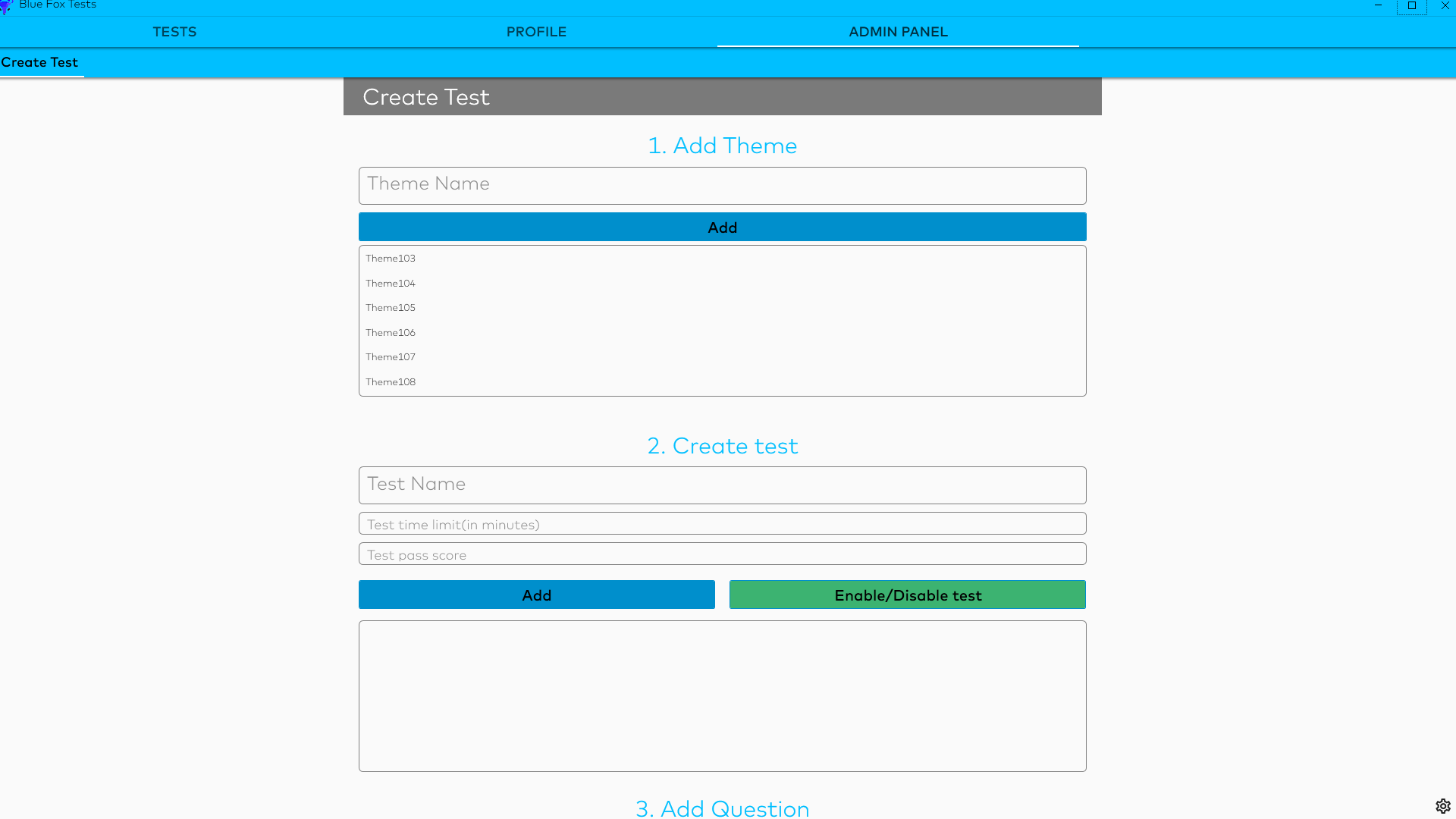
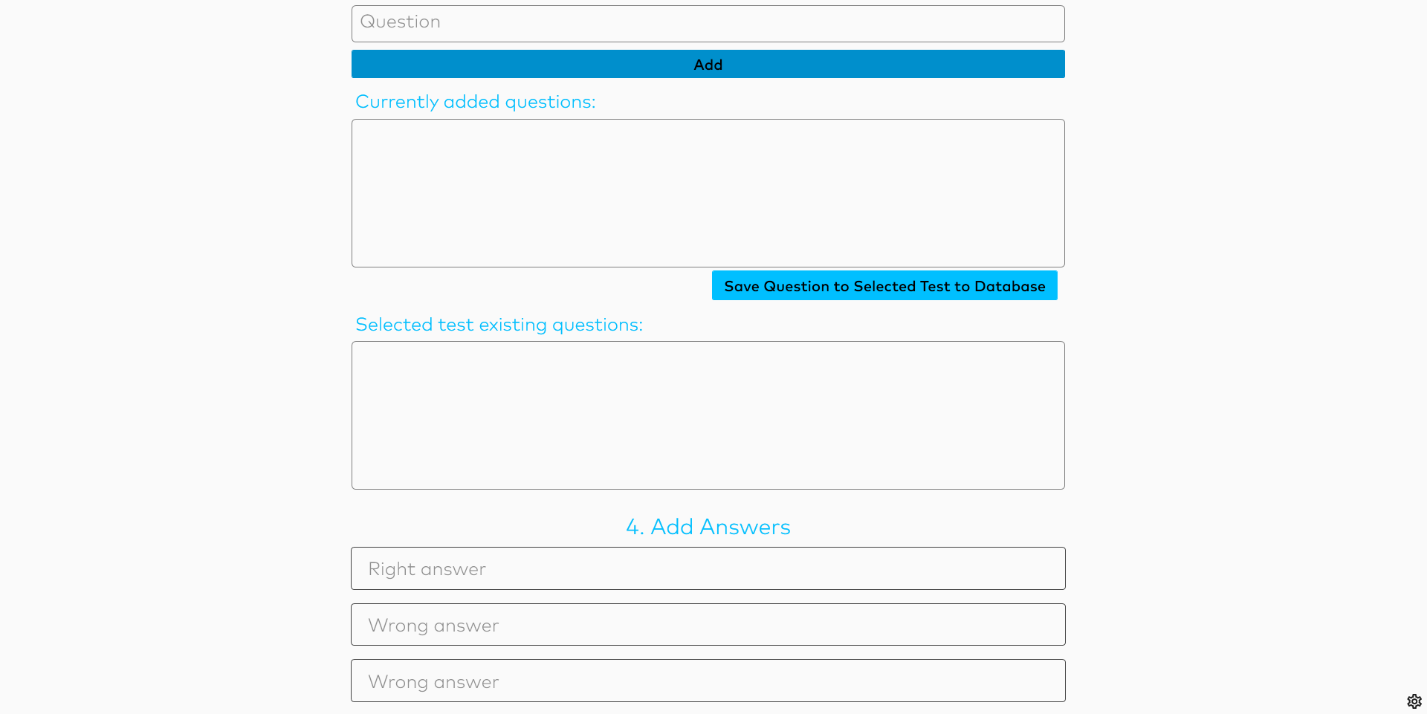


Рисунок 4.1.10 – Статистика пользователя

Вкладка Admin Panel служит для создания/редактирования тем/тестов/вопросов/ответов к тестам (Рисунок 4.1.11).



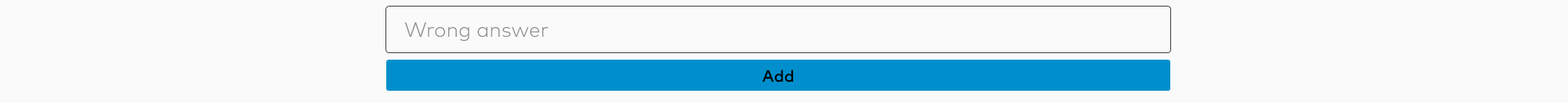


Рисунок 4.1.11 – Создание тестов

# **Тестирование приложения**

Для обеспечения корректности работы программы, обрабатываются различные ошибки, возникающие в процессе работы. Данное программное средство использует подключение к базе данных, следовательно, неправильно введенные данные или же их отсутствие может повлечь за собой неработоспособность приложения.

На рисунках 4.2.1, 4.2.2, 4.2.3 представлен результат обработки исключения при регистрации и авторизации. Так же такие исключения возникнут, если попытаться сохранить изменения логина и пароля пользователя без введённых данных.

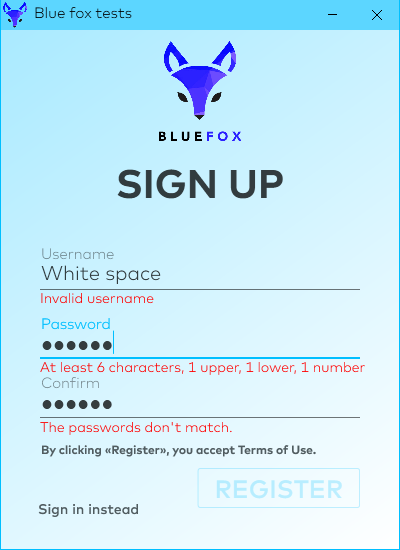


Рисунок 4.2.1 – Некорректный логин и пароль

На рисунке 4.2.2 представлен результат попытки зайти не введя имя пользователя.

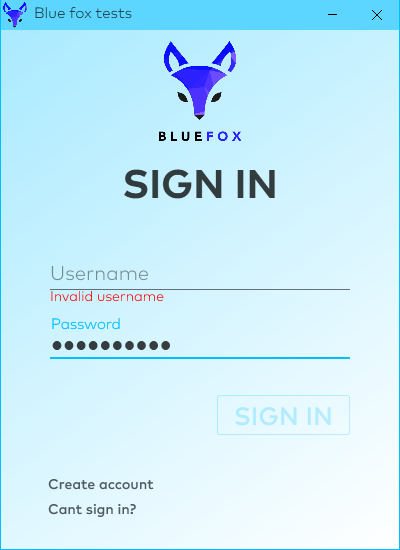


Рисунок 4.2.2 – Не введены данные

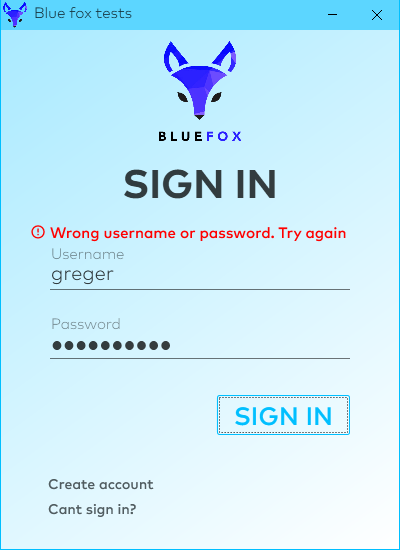
Также возникают ошибки, если пользователь пытается ввести неверные данные в поле логин или пароль (рисунок 4.2.3).

Рисунок 4.2.3 – Неверные логин и пароль

На рисунках 4.2.4, 4.2.5, представлена обработка неверно введённых данных при создании тестов.

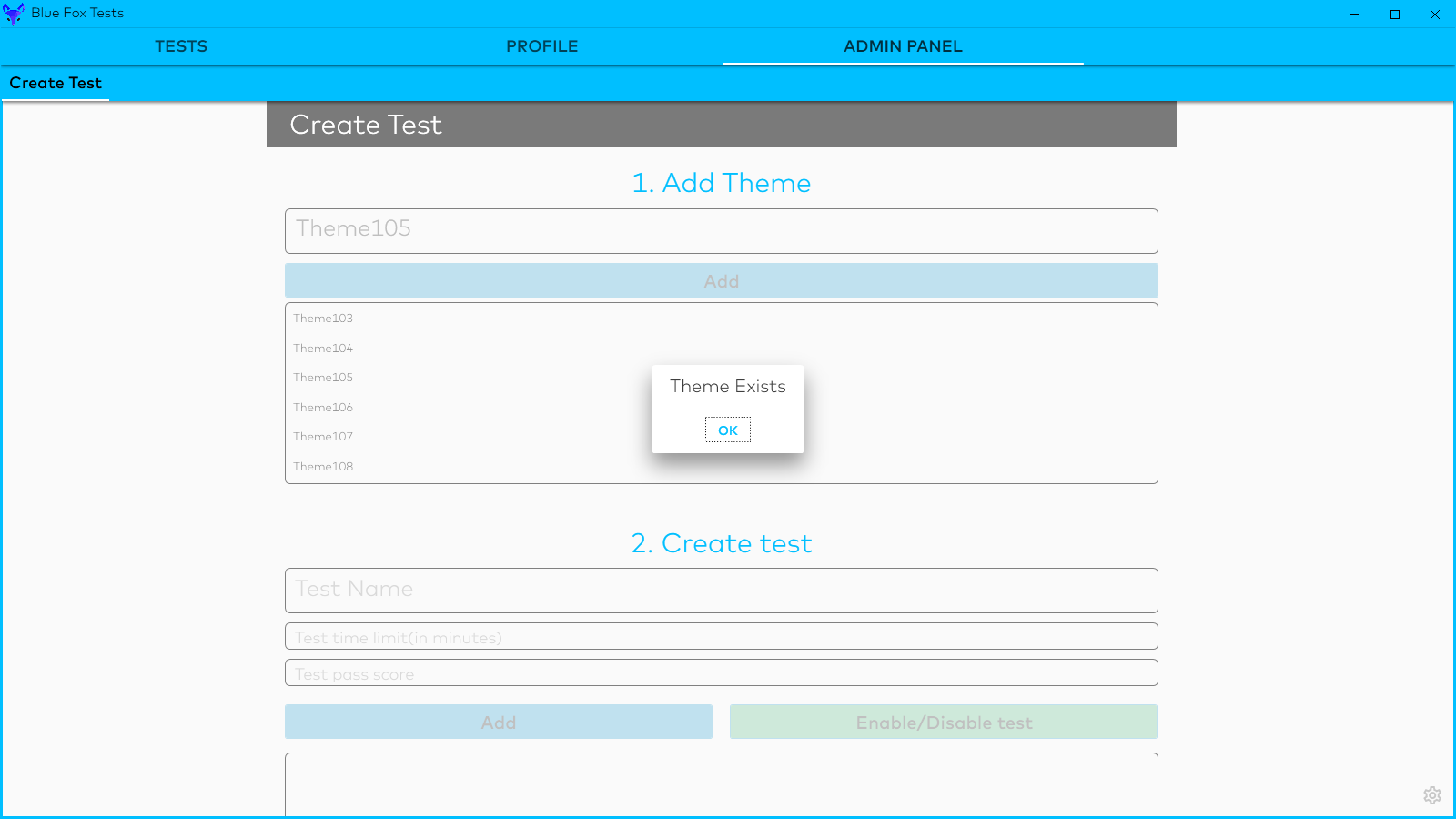


Рисунок 4.2.4 – Тема уже существует

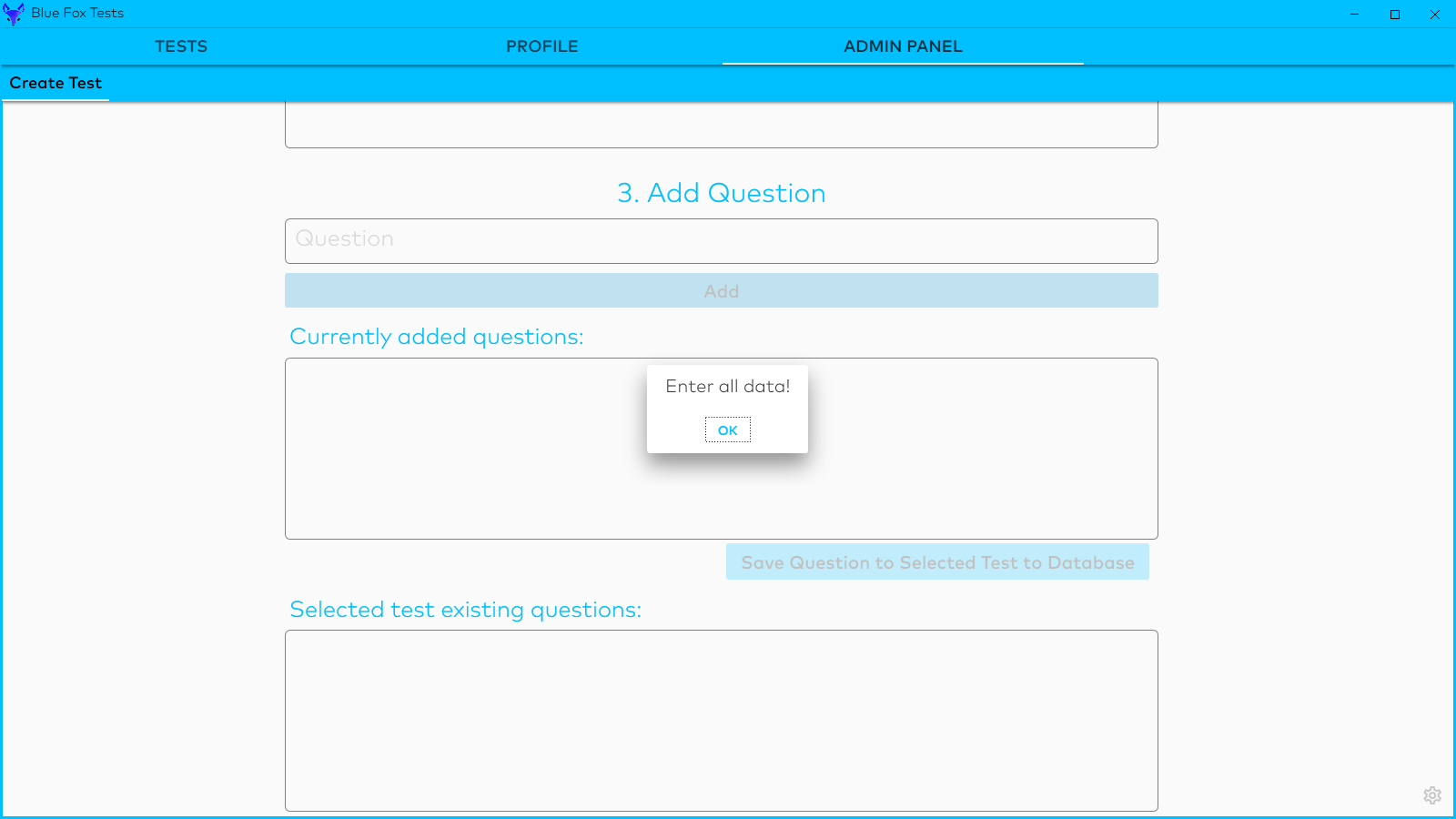


Рисунок 4.2.5 – Указаны не все параметры при создании теста

**Заключение**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Заключение

Лит.

Листов

1

БГТУ

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных «Тестирование», которая в купе с приложением формирует полноценное десктопное приложение для управления и решения тестов. Основной целью курсового проекта стало проектирование базы данных для дальнейшей интеграции с приложением, которое помогло облегчить взаимодействие с базой данных посредством программного интерфейса. При разработке базы данных использовались следующие объекты: таблицы, представления, хранимые процедуры, функции, индексы, триггеры, синонимы. Взаимодействие между приложением и базой данных было реализовано с помощью технологии [ADO.NET](https://vk.com/away.php?utf=1&to=http%3A%2F%2FADO.NET). При разработке выполнены следующие пункты:

* авторизация и регистрация пользователей;
* разделение пользователя на админа и пользователя;
* cоставление интерфейса для взаимодействия с БД;
* создание, выполнение, изменение, удаление тестов из списка;
* управление профилями пользователей;
* шифрование данных клиента с использованием MD5 хэша.
* просмотр результатов пройденного теста;
* просмотр статистики выполненных тестов.
* технология полнотекстового поиска с использованием Oracle Text;
* импорт из XML файла в таблицы БД и экспорт в XML файл таблиц БД данных о пользователях с использованием пакета DBMS\_XMLPARSER.

В результате выполнения курсового проекта было получено рабочее десктопное приложение «BlueFox Tests».

Приложение прошло тестирование при использовании в БД большого количество данных.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

## **Список литературы**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Список литературы

Лит.

Листов

1

БГТУ

1. Leetcode [Электронный ресурс]. – Режим доступа: <https://leetcode.com> – Дата доступа: 15.12.2020.

2. SunRav TestOfficePro – Программа для создания тестов [Электронный ресурс]. – Режим доступа: <https://sunrav.ru/testofficepro.html> – Дата доступа: 15.12.2020.

3. UML – диаграммы классов [Электронный ресурс]. – Режим доступа: <https://prog-cpp.ru/uml-classes> – Дата доступа: 15.12.2020.

4. Представления в Oracle [Электронный ресурс]. – Режим доступа: <https://oracle-patches.com/oracle/begin/3016-представления-oracle-database> – Дата доступа: 15.12.2020.

5. Хранимая процедура [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Хранимая_процедура> – Дата доступа: 15.12.2020.

6. XML-парсер для Oracle PL/SQL [Электронный ресурс]. – Режим доступа: <https://o7planning.org/ru/10497/xml-parser-for-oracle-pl-sql>– Дата доступа: 15.12.2020.

7. Индекс (базы данных) [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Индекс_(базы_данных)> – Дата доступа: 15.12.2020.

8. Триггер (базы данных) [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Триггер_(базы_данных)> – Дата доступа: 15.12.2020.

9. Полнотекстовый поиск при помощи Oracle Text [Электронный ресурс]. – Режим доступа: <http://oraclestart.blogspot.com/2013/08/oracle-text.html> – Дата доступа: 15.12.2020.

10. ActiveX Data Object для .NET [Электронный ресурс]. – Режим доступа: <https://www.oracle.com/database/technologies/appdev/dotnet/odp.html> – Дата доступа: 15.12.2020.

11. Oracle Data Provider for .NET [Электронный ресурс]. – Режим доступа: <https://www.oracle.com/database/technologies/appdev/dotnet/odp.html> – Дата доступа: 15.12.2020.

## **Приложение А**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение А

Лит.

Листов

1

БГТУ

## **Приложение Б**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение Б

Лит.

Листов

1

БГТУ

CREATE TABLE Users (

User\_Id number(10) NOT NULL ,

Username nvarchar2(20) NOT NULL,

Password\_Hash nvarchar2(15) NOT NULL,

PRIMARY KEY (User\_Id)

);

CREATE TABLE Admins (

Admin\_Id number(10) NOT NULL ,

User\_Id number(10) NOT NULL,

PRIMARY KEY (Admin\_Id)

);

CREATE TABLE User\_Stats (

User\_Id number(10) NOT NULL,

Right\_Answered number(10),

Total\_Answered number(10),

Avg\_Score number(4,2),

Finished\_Tests\_Count number(10),

Passed\_Tests\_Count number(10),

PRIMARY KEY (User\_Id)

);

CREATE TABLE User\_Info (

User\_Id number(10) NOT NULL,

Name nvarchar2(30),

Gender nvarchar2(15),

Location nvarchar2(50),

Birthday timestamp(3),

Summary nvarchar2(100),

Education nvarchar2(60),

Work nvarchar2(60),

PRIMARY KEY (User\_Id)

);

CREATE TABLE Test\_Result (

Result\_Id number(10) ,

User\_Id number(10),

Test\_Id number(10),

Try\_Count number(10),

Is\_Passed number(1),

Score number(4,2),

Right\_Answers\_Count number(10),

Questions\_Count number(10),

Start\_Date date,

End\_Date date,

PRIMARY KEY (Result\_Id)

);

CREATE TABLE Tests (

Test\_Id number(10) NOT NULL ,

Admin\_Id number(10),

Test\_Name nvarchar2(50),

Theme\_Id number(10),

Time\_Limit\_In\_Minutes number(10),

Passing\_Score number(4,2),

Is\_Enabled number(1),

PRIMARY KEY (Test\_Id)

);

CREATE TABLE Questions\_For\_Tests (

Question\_Id number(10) NOT NULL ,

Test\_Id number(10),

Question\_Number number(10),

Question nvarchar2(200),

PRIMARY KEY (Question\_Id)

);

CREATE TABLE Answers\_For\_Tests (

Answer\_Id number(10) NOT NULL ,

Answer nvarchar2(200),

Is\_Right number(1) NOT NULL,

Question\_Id number(10),

PRIMARY KEY (Answer\_Id)

);

CREATE TABLE Themes\_For\_Tests (

Theme\_Id number(10) NOT NULL ,

Theme\_Name nvarchar2(70),

PRIMARY KEY (Theme\_Id)

);

CREATE TABLE User\_Answers (

Result\_Id number(10) NOT NULL,

Question\_Id number(10),

User\_Answer number(10)

);

-- Generate ID using sequence and trigger

CREATE SEQUENCE Users\_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER Users\_seq\_tr

BEFORE INSERT ON Users FOR EACH ROW

WHEN (NEW.User\_Id IS NULL)

BEGIN

SELECT Users\_seq.NEXTVAL INTO :NEW.User\_Id FROM DUAL;

END;

/

CREATE SEQUENCE Admins\_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER Admins\_seq\_tr

BEFORE INSERT ON Admins FOR EACH ROW

WHEN (NEW.Admin\_Id IS NULL)

BEGIN

SELECT Admins\_seq.NEXTVAL INTO :NEW.Admin\_Id FROM DUAL;

END;

/

CREATE SEQUENCE Test\_Result\_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER Test\_Result\_seq\_tr

BEFORE INSERT ON Test\_Result FOR EACH ROW

WHEN (NEW.Result\_Id IS NULL)

BEGIN

SELECT Test\_Result\_seq.NEXTVAL INTO :NEW.Result\_Id FROM DUAL;

END;

/

CREATE SEQUENCE Tests\_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER Tests\_seq\_tr

BEFORE INSERT ON Tests FOR EACH ROW

WHEN (NEW.Test\_Id IS NULL)

BEGIN

SELECT Tests\_seq.NEXTVAL INTO :NEW.Test\_Id FROM DUAL;

END;

/

CREATE SEQUENCE Questions\_For\_Tests\_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER Questions\_For\_Tests\_seq\_tr

BEFORE INSERT ON Questions\_For\_Tests FOR EACH ROW

WHEN (NEW.Question\_Id IS NULL)

BEGIN

SELECT Questions\_For\_Tests\_seq.NEXTVAL INTO :NEW.Question\_Id FROM DUAL;

END;

/

CREATE SEQUENCE Answers\_For\_Tests\_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER Answers\_For\_Tests\_seq\_tr

BEFORE INSERT ON Answers\_For\_Tests FOR EACH ROW

WHEN (NEW.Answer\_Id IS NULL)

BEGIN

SELECT Answers\_For\_Tests\_seq.NEXTVAL INTO :NEW.Answer\_Id FROM DUAL;

END;

CREATE SEQUENCE Themes\_For\_Tests\_seq START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER Themes\_For\_Tests\_seq\_tr

BEFORE INSERT ON Themes\_For\_Tests FOR EACH ROW

WHEN (NEW.Theme\_Id IS NULL)

BEGIN

SELECT Themes\_For\_Tests\_seq.NEXTVAL INTO :NEW.Theme\_Id FROM DUAL;

END;

/

ALTER TABLE Users ADD CONSTRAINT Username\_Unique UNIQUE (username);

ALTER TABLE Themes\_For\_Tests ADD CONSTRAINT Themes\_Unique UNIQUE (Theme\_Name);

ALTER TABLE Admins ADD FOREIGN KEY (User\_Id) REFERENCES Users (User\_Id);

ALTER TABLE User\_Stats ADD FOREIGN KEY (User\_Id) REFERENCES Users (User\_Id);

ALTER TABLE User\_Info ADD FOREIGN KEY (User\_Id) REFERENCES Users (User\_Id);

ALTER TABLE Test\_Result ADD FOREIGN KEY (User\_Id) REFERENCES Users (User\_Id);

ALTER TABLE Test\_Result ADD FOREIGN KEY (Test\_Id) REFERENCES Tests (Test\_Id);

ALTER TABLE Tests ADD FOREIGN KEY (Admin\_Id) REFERENCES Admins (Admin\_Id);

ALTER TABLE Tests ADD FOREIGN KEY (Theme\_Id) REFERENCES Themes\_For\_Tests (Theme\_Id);

ALTER TABLE Questions\_For\_Tests ADD FOREIGN KEY (Test\_Id) REFERENCES Tests (Test\_Id);

ALTER TABLE Answers\_For\_Tests ADD FOREIGN KEY (Question\_Id) REFERENCES Questions\_For\_Tests (Question\_Id) ON DELETE CASCADE;

ALTER TABLE User\_Answers ADD FOREIGN KEY (Result\_Id) REFERENCES Test\_Result (Result\_Id);

ALTER TABLE User\_Answers ADD FOREIGN KEY (Question\_Id) REFERENCES Questions\_For\_Tests (Question\_Id);

ALTER TABLE User\_Answers ADD FOREIGN KEY (User\_Answer) REFERENCES Answers\_For\_Tests (Answer\_Id);

CREATE OR REPLACE VIEW USERS\_VIEW AS SELECT \* FROM USERS;

CREATE OR REPLACE VIEW ADMINS\_VIEW AS SELECT \* FROM ADMINS;

CREATE OR REPLACE VIEW STATS\_VIEW AS SELECT \* FROM USER\_STATS;

CREATE OR REPLACE VIEW INFO\_VIEW AS SELECT \* FROM USER\_INFO;

CREATE OR REPLACE VIEW THEMES\_VIEW AS SELECT \* FROM THEMES\_FOR\_TESTS;

CREATE OR REPLACE VIEW TESTS\_VIEW AS SELECT \* FROM TESTS;

CREATE OR REPLACE VIEW QUESTIONS\_VIEW AS SELECT \* FROM QUESTIONS\_FOR\_TESTS;

CREATE OR REPLACE VIEW ANSWERS\_VIEW AS SELECT \* FROM ANSWERS\_FOR\_TESTS;

CREATE OR REPLACE VIEW TEST\_RESULT\_VIEW AS SELECT \* FROM TEST\_RESULT;

CREATE OR REPLACE VIEW USER\_ANSWERS\_VIEW AS SELECT \* FROM USER\_ANSWERS;

## **Приложение В**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение В

Лит.

Листов

1

БГТУ

----------GET ALL USERS FROM USERS

CREATE OR REPLACE PROCEDURE GET\_USERS IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM USERS\_VIEW u LEFT JOIN ADMINS\_VIEW a ON a.User\_Id = u.User\_Id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_USERS;

--------------------

----------GET USER BY USERNAME

CREATE OR REPLACE PROCEDURE GET\_USER\_BY\_USERNAME(usrname USERS.USERNAME%TYPE) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM USERS\_VIEW WHERE LOWER(USERNAME) = LOWER(usrname);

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_USER\_BY\_USERNAME;

--------------------

----------GET USER BY ID

CREATE OR REPLACE PROCEDURE GET\_USER\_BY\_ID(u\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM USERS\_VIEW WHERE USER\_ID = u\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_USER\_BY\_ID;

--------------------

----------GET USER WITH INFO

CREATE OR REPLACE PROCEDURE GET\_USER\_WITH\_INFO(u\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM USERS\_VIEW u LEFT JOIN USER\_INFO info ON u.user\_id = info.user\_id WHERE u.user\_id = u\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_USER\_WITH\_INFO;

--------------------

----------GET USER INFO BY ID

CREATE OR REPLACE PROCEDURE GET\_USER\_INFO(u\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM INFO\_VIEW WHERE user\_id = u\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_USER\_INFO;

--------------------

----------GET THEMES

CREATE OR REPLACE PROCEDURE GET\_THEMES IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM THEMES\_VIEW;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_THEMES;

--------------------

----------GET THEME BY NAME

CREATE OR REPLACE PROCEDURE GET\_THEME\_BY\_NAME(th\_name VARCHAR2) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM THEMES\_VIEW WHERE LOWER(THEME\_NAME) = LOWER(th\_name);

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_THEME\_BY\_NAME;

--------------------

----------GET TESTS BY THEME ID

CREATE OR REPLACE PROCEDURE GET\_TESTS\_BY\_THEME\_ID(th\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM TESTS\_VIEW WHERE THEME\_ID = th\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_TESTS\_BY\_THEME\_ID;

--------------------

----------GET TEST BY NAME

CREATE OR REPLACE PROCEDURE GET\_TEST\_BY\_NAME(t\_name VARCHAR2) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM TESTS\_VIEW WHERE LOWER(TEST\_NAME) = LOWER(t\_name);

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_TEST\_BY\_NAME;

--------------------

----------GET QUESTIONS BY TEST ID

CREATE OR REPLACE PROCEDURE GET\_QUESTIONS\_BY\_TEST\_ID(t\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM QUESTIONS\_VIEW WHERE TEST\_ID = t\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_QUESTIONS\_BY\_TEST\_ID;

--------------------

----------GET ANSWERS BY QUESTION ID

CREATE OR REPLACE PROCEDURE GET\_ANSWERS\_BY\_QUESTION\_ID(q\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM ANSWERS\_VIEW WHERE QUESTION\_ID = q\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_ANSWERS\_BY\_QUESTION\_ID;

--------------------

----------GET USER ANSWERS BY RESULT ID

CREATE OR REPLACE PROCEDURE GET\_USER\_ANSWERS\_BY\_RESULT\_ID(res\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM USER\_ANSWERS\_VIEW WHERE RESULT\_ID = res\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_USER\_ANSWERS\_BY\_RESULT\_ID;

--------------------

----------FIND TEST RESULT

CREATE OR REPLACE PROCEDURE FIND\_TEST\_RESULT(u\_id NUMBER, t\_id NUMBER, try\_c NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM TEST\_RESULT\_VIEW WHERE TEST\_ID = t\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END FIND\_TEST\_RESULT;

--------------------

----------GET TEST RESULT BY RESULT ID

CREATE OR REPLACE PROCEDURE GET\_TEST\_RESULT\_BY\_RESULT\_ID(res\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM TEST\_RESULT\_VIEW WHERE RESULT\_ID = res\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_TEST\_RESULT\_BY\_RESULT\_ID;

--------------------

----------GET TEST RESULTS BY USER ID

CREATE OR REPLACE PROCEDURE GET\_TEST\_RESULTS\_BY\_USER\_ID(u\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM TEST\_RESULT\_VIEW WHERE USER\_ID = u\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_TEST\_RESULTS\_BY\_USER\_ID;

--------------------

----------GET USER STATS BY USER ID

CREATE OR REPLACE PROCEDURE GET\_USER\_STATS(u\_id NUMBER) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM STATS\_VIEW WHERE USER\_ID = u\_id;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END GET\_USER\_STATS;

----------ADD USER

CREATE OR REPLACE PROCEDURE ADD\_USER (

username VARCHAR2,

password\_hash VARCHAR2) AS

BEGIN

INSERT INTO USERS(USERNAME, PASSWORD\_HASH) VALUES (username, password\_hash);

commit;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END;

--------------------

----------ADD USERINFO

CREATE OR REPLACE PROCEDURE ADD\_USER\_INFO (

u\_id number, u\_name VARCHAR2, u\_gender VARCHAR2, u\_location VARCHAR2, u\_birthday TIMESTAMP,

u\_summary VARCHAR2, u\_education VARCHAR2, u\_work VARCHAR2) AS

BEGIN

INSERT INTO INFO\_VIEW VALUES (u\_id, u\_name, u\_gender, u\_location, u\_birthday, u\_summary, u\_education, u\_work);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END ADD\_USER\_INFO;

--------------------

----------ADD THEME

CREATE OR REPLACE PROCEDURE ADD\_THEME (th\_name VARCHAR2)

AS

BEGIN

INSERT INTO THEMES\_VIEW(THEME\_NAME) VALUES (th\_name);

commit;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END ADD\_THEME;

--------------------

----------ADD TEST

CREATE OR REPLACE PROCEDURE ADD\_TEST (a\_id NUMBER, t\_name VARCHAR2, th\_id NUMBER, time\_limit NUMBER, pass\_score NUMBER, is\_enable NUMBER)

AS

BEGIN

INSERT INTO TESTS\_VIEW(ADMIN\_ID, TEST\_NAME, THEME\_ID, TIME\_LIMIT\_IN\_MINUTES, PASSING\_SCORE, is\_enabled) VALUES (a\_id, t\_name, th\_id, time\_limit, pass\_score, is\_enable);

commit;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END ADD\_TEST;

--------------------

----------ADD QUESTION

CREATE OR REPLACE PROCEDURE ADD\_QUESTION (t\_id NUMBER, q\_number NUMBER, qn VARCHAR2)

AS

BEGIN

INSERT INTO QUESTIONS\_VIEW(TEST\_ID, QUESTION\_NUMBER, QUESTION) VALUES (t\_id, q\_number, qn);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END ADD\_QUESTION;

--------------------

----------ADD ANSWER

CREATE OR REPLACE PROCEDURE ADD\_ANSWER (answ VARCHAR2, is\_r NUMBER, q\_id NUMBER)

AS

BEGIN

INSERT INTO ANSWERS\_VIEW(ANSWER, IS\_RIGHT, QUESTION\_ID) VALUES (answ, is\_r, q\_id);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END ADD\_ANSWER;

--------------------

----------ADD USER ANSWER

CREATE OR REPLACE PROCEDURE ADD\_USER\_ANSWER (res\_id NUMBER, q\_id NUMBER, u\_answ NUMBER)

AS

BEGIN

INSERT INTO USER\_ANSWERS\_VIEW VALUES (res\_id, q\_id, u\_answ);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END ADD\_USER\_ANSWER;

--------------------

----------INITIALIZE USER INFO AND STATS

CREATE OR REPLACE PROCEDURE INITIALIZE\_USER\_INFO\_AND\_STATS (u\_id number)

AS

BEGIN

INSERT INTO INFO\_VIEW(USER\_ID) VALUES (u\_id);

INSERT INTO STATS\_VIEW VALUES (u\_id, 0, 0, 0, 0, 0);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END INITIALIZE\_USER\_INFO\_AND\_STATS;

--------------------

----------INITIALIZE TEST RESULT AND RETURN RESULT\_ID

CREATE OR REPLACE FUNCTION FUNC\_INITIALIZE\_TEST\_RESULT(u\_id NUMBER, t\_id NUMBER, try\_c NUMBER, q\_count NUMBER, st\_date TEST\_RESULT.START\_DATE%TYPE)

RETURN NUMBER

IS

BEGIN

INSERT INTO TEST\_RESULT\_VIEW(USER\_ID, TEST\_ID, TRY\_COUNT, QUESTIONS\_COUNT, START\_DATE, RIGHT\_ANSWERS\_COUNT, SCORE, IS\_PASSED) VALUES (u\_id, t\_id, try\_c, q\_count, st\_date, 0, 0, 0);

RETURN Test\_Result\_seq.CURRVAL;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END FUNC\_INITIALIZE\_TEST\_RESULT;

----------CHECK USER LOGIN

CREATE OR REPLACE PROCEDURE CHECK\_USER\_LOGIN(usrname VARCHAR2, p\_hash VARCHAR2) IS

user\_cur SYS\_REFCURSOR;

BEGIN

OPEN user\_cur FOR SELECT \* FROM USERS\_VIEW WHERE LOWER(USERNAME) = LOWER(usrname) AND PASSWORD\_HASH = p\_hash;

DBMS\_SQL.RETURN\_RESULT(user\_cur);

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END CHECK\_USER\_LOGIN;

--------------------

----------CHECK IF USER EXISTS

CREATE OR REPLACE FUNCTION USER\_EXISTS(u\_id NUMBER)

RETURN NUMBER

IS

counter NUMBER := 0;

BEGIN

SELECT COUNT(\*) into counter

FROM USERS\_VIEW

WHERE USER\_ID = u\_id;

RETURN counter;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END USER\_EXISTS;

--------------------

----------CHECK IF USER EXISTS BY USERNAME

CREATE OR REPLACE FUNCTION USER\_EXISTS\_USERNAME(usrname VARCHAR2)

RETURN NUMBER

IS

counter NUMBER := 0;

BEGIN

SELECT COUNT(\*) into counter

FROM USERS\_VIEW

WHERE LOWER(USERNAME) = LOWER(usrname);

RETURN counter;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END USER\_EXISTS\_USERNAME;

--------------------

----------CHECK IF USER IS ADMIN

CREATE OR REPLACE FUNCTION USER\_IS\_ADMIN(u\_id NUMBER)

RETURN NUMBER

IS

counter NUMBER := 0;

BEGIN

SELECT COUNT(\*) into counter

FROM ADMINS\_VIEW

WHERE USER\_ID = u\_id;

if counter = 0 then return 0;

else SELECT ADMIN\_ID into counter FROM ADMINS\_VIEW WHERE USER\_ID = u\_id;

RETURN counter;

END IF;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END USER\_IS\_ADMIN;

----------UPDATE USERINFO

CREATE OR REPLACE PROCEDURE UPDATE\_USER\_INFO (

u\_id number, u\_name VARCHAR2, u\_gender VARCHAR2, u\_location VARCHAR2, u\_birthday VARCHAR2,

u\_summary VARCHAR2, u\_education VARCHAR2, u\_work VARCHAR2)

AS

BEGIN

UPDATE USER\_INFO

SET NAME = u\_name, GENDER = u\_gender, LOCATION = u\_location, BIRTHDAY = TO\_TIMESTAMP(u\_birthday, 'DD.MM.YYYY'), SUMMARY = u\_summary, EDUCATION = u\_education, WORK = u\_work

WHERE USER\_ID = u\_id;

commit;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END UPDATE\_USER\_INFO;

--------------------

----------UPDATE TEST\_RESULT ON TEST END

CREATE OR REPLACE PROCEDURE UPDATE\_TEST\_RESULT\_TEST\_END (

res\_id NUMBER, scor TEST\_RESULT.SCORE%TYPE, q\_count NUMBER, right\_ans\_count NUMBER, is\_pass TEST\_RESULT.IS\_PASSED%TYPE,

end\_d TEST\_RESULT.END\_DATE%TYPE)

AS

BEGIN

UPDATE TEST\_RESULT

SET SCORE = scor, QUESTIONS\_COUNT = q\_count, RIGHT\_ANSWERS\_COUNT = right\_ans\_count, IS\_PASSED = is\_pass, END\_DATE = end\_d

WHERE RESULT\_ID = res\_id;

commit;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END UPDATE\_TEST\_RESULT\_TEST\_END;

--------------------

----------ENABLE/DISABLE TEST

CREATE OR REPLACE PROCEDURE ENABLE\_TEST (t\_id number, is\_enable number)

AS

BEGIN

UPDATE TESTS

SET IS\_ENABLED = is\_enable WHERE TEST\_ID = t\_id;

commit;

exception

when others

then RAISE\_APPLICATION\_ERROR(sqlerrm, sqlcode);

END ENABLE\_TEST;

--------------------

## **Приложение Г**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение Г

Лит.

Листов

1

БГТУ

CREATE OR REPLACE PROCEDURE EXPORT\_USERS\_TO\_XML

IS

DOC DBMS\_XMLDOM.DOMDocument;

XDATA XMLTYPE;

CURSOR XMLCUR IS

SELECT XMLELEMENT("USERS",

XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi",

'http://www.oracle.com/Employee.xsd' AS "xsi:nonamespaceSchemaLocation"),

XMLAGG(XMLELEMENT("USER",

XMLELEMENT("USER\_ID",U.USER\_ID),

XMLELEMENT("USERNAME",U.USERNAME),

XMLELEMENT("PASSWORD\_HASH",U.PASSWORD\_HASH),

XMLELEMENT("NAME",I.NAME),

XMLELEMENT("GENDER",I.GENDER),

XMLELEMENT("LOCATION",I.LOCATION),

XMLELEMENT("BIRTHDAY",I.BIRTHDAY),

XMLELEMENT("SUMMARY",I.SUMMARY),

XMLELEMENT("EDUCATION",I.EDUCATION),

XMLELEMENT("WORK",I.WORK),

XMLELEMENT("RIGHT\_ANSWERED",S.RIGHT\_ANSWERED),

XMLELEMENT("TOTAL\_ANSWERED",S.TOTAL\_ANSWERED),

XMLELEMENT("AVG\_SCORE",S.AVG\_SCORE),

XMLELEMENT("FINISHED\_TESTS\_COUNT",S.FINISHED\_TESTS\_COUNT),

XMLELEMENT("PASSED\_TESTS\_COUNT",S.PASSED\_TESTS\_COUNT)

))

) FROM USERS\_VIEW U LEFT JOIN INFO\_VIEW I ON U.USER\_ID = I.USER\_ID LEFT JOIN USER\_STATS S ON I.USER\_ID = S.USER\_ID;

BEGIN

OPEN XMLCUR;

LOOP

FETCH XMLCUR INTO XDATA;

EXIT WHEN XMLCUR%NOTFOUND;

END LOOP;

CLOSE XMLCUR;

DOC := DBMS\_XMLDOM.NewDOMDocument(XDATA);

DBMS\_XMLDOM.WRITETOFILE(DOC, 'XML\_PATH/users.xml');

END;

CREATE OR REPLACE PROCEDURE IMPORT\_USERS\_FROM\_XML

IS

L\_CLOB CLOB;

L\_BFILE BFILE := BFILENAME('XML\_PATH', 'users.xml');

L\_DEST\_OFFSET INTEGER := 1;

L\_SRC\_OFFSET INTEGER := 1;

L\_BFILE\_CSID NUMBER := 0;

L\_LANG\_CONTEXT INTEGER := 0;

L\_WARNING INTEGER := 0;

P DBMS\_XMLPARSER.PARSER;

V\_DOC DBMS\_XMLDOM.DOMDOCUMENT;

V\_ROOT\_ELEMENT DBMS\_XMLDOM.DOMELEMENT;

V\_CHILD\_NODES DBMS\_XMLDOM.DOMNODELIST;

V\_CURRENT\_NODE DBMS\_XMLDOM.DOMNODE;

U USERS\_VIEW%ROWTYPE;

INF INFO\_VIEW%ROWTYPE;

S USER\_STATS%ROWTYPE;

BEGIN

DBMS\_LOB.CREATETEMPORARY (L\_CLOB, TRUE);

DBMS\_LOB.FILEOPEN(L\_BFILE, DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADCLOBFROMFILE (DEST\_LOB => L\_CLOB, SRC\_BFILE => L\_BFILE, AMOUNT => DBMS\_LOB.LOBMAXSIZE,

DEST\_OFFSET => L\_DEST\_OFFSET, SRC\_OFFSET => L\_SRC\_OFFSET, BFILE\_CSID => L\_BFILE\_CSID,

LANG\_CONTEXT => L\_LANG\_CONTEXT, WARNING => L\_WARNING);

DBMS\_LOB.FILECLOSE(L\_BFILE);

COMMIT;

-- Create XML Parser.

P := DBMS\_XMLPARSER.NEWPARSER;

-- Parse XML into DOM object

DBMS\_XMLPARSER.PARSECLOB(P,L\_CLOB);

-- Document Element

V\_DOC := DBMS\_XMLPARSER.GETDOCUMENT(P);

-- Root element (<USERS>)

V\_ROOT\_ELEMENT := DBMS\_XMLDOM.Getdocumentelement(v\_Doc);

--- return Dbms\_Xmldom.Domnodelist

V\_CHILD\_NODES := DBMS\_XMLDOM.GETCHILDRENBYTAGNAME(V\_ROOT\_ELEMENT,'\*');

FOR i IN 0 .. DBMS\_XMLDOM.GETLENGTH(V\_CHILD\_NODES) - 1

LOOP

-- <USER> Node.

V\_CURRENT\_NODE := DBMS\_XMLDOM.ITEM(V\_CHILD\_NODES,i);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'USER\_ID/text()',U.USER\_ID);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'USERNAME/text()',U.USERNAME);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'PASSWORD\_HASH/text()',U.PASSWORD\_HASH);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'NAME/text()',INF.NAME);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'GENDER/text()',INF.GENDER);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'LOCATION/text()',INF.LOCATION);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'BIRTHDAY/text()',INF.BIRTHDAY);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'SUMMARY/text()',INF.SUMMARY);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'EDUCATION/text()',INF.EDUCATION);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'WORK/text()',INF.WORK);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'RIGHT\_ANSWERED/text()',S.RIGHT\_ANSWERED);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'TOTAL\_ANSWERED/text()',S.TOTAL\_ANSWERED);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'AVG\_SCORE/text()',S.AVG\_SCORE);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'FINISHED\_TESTS\_COUNT/text()',S.FINISHED\_TESTS\_COUNT);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE, 'PASSED\_TESTS\_COUNT/text()',S.PASSED\_TESTS\_COUNT);

INSERT INTO USERS(USERNAME,PASSWORD\_HASH)

VALUES( U.USERNAME,U.PASSWORD\_HASH);

INSERT INTO USER\_INFO(USER\_ID, NAME,GENDER,LOCATION,BIRTHDAY,SUMMARY,EDUCATION,WORK)

VALUES(bluefox\_admin.Users\_seq.CURRVAL,INF.NAME,INF.GENDER,INF.LOCATION,INF.BIRTHDAY,INF.SUMMARY,INF.EDUCATION,INF.WORK);

INSERT INTO USER\_STATS(USER\_ID, RIGHT\_ANSWERED,TOTAL\_ANSWERED,AVG\_SCORE,FINISHED\_TESTS\_COUNT,PASSED\_TESTS\_COUNT)

VALUES(bluefox\_admin.Users\_seq.CURRVAL,S.RIGHT\_ANSWERED,S.TOTAL\_ANSWERED,S.AVG\_SCORE,S.FINISHED\_TESTS\_COUNT,S.PASSED\_TESTS\_COUNT);

END LOOP;

DBMS\_LOB.FREETEMPORARY(L\_CLOB);

DBMS\_XMLPARSER.FREEPARSER(P);

DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);

COMMIT;

END;

## **Приложение Д**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Харевич К. В.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение Д

Лит.

Листов

1

БГТУ

create or replace trigger USER\_INFO\_INIT

after insert on USERS for each row

begin

INITIALIZE\_USER\_INFO\_AND\_STATS(Users\_seq.CURRVAL);

end;

create or replace trigger UPDATE\_USER\_STATS\_ON\_TEST\_END

after update on TEST\_RESULT for each row

begin

UPDATE USER\_STATS SET

AVG\_SCORE = (AVG\_SCORE \* FINISHED\_TESTS\_COUNT + :NEW.SCORE) / (FINISHED\_TESTS\_COUNT + 1),

TOTAL\_ANSWERED = TOTAL\_ANSWERED + :NEW.QUESTIONS\_COUNT,

RIGHT\_ANSWERED = RIGHT\_ANSWERED + :NEW.RIGHT\_ANSWERS\_COUNT,

FINISHED\_TESTS\_COUNT = FINISHED\_TESTS\_COUNT + 1,

PASSED\_TESTS\_COUNT = PASSED\_TESTS\_COUNT + :NEW.IS\_PASSED

WHERE USER\_ID = :OLD.USER\_ID;

end;