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 «Студенты университета»

Выполнил студент Истомин И.О.

(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

Реферат

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И. О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Реферат

Лит.

Листов

1

БГТУ

Пояснительная записка курсового проекта содержит 42 страниц пояснительной записки, 33 иллюстраций, 9 источников литературы, 5 приложений.

.NET Framework, WPF, ORACLE DATABASE 12c, SQL DEVELOPER, ADO.NET

Цель курсового проекта – разработка базы данных и приложения, которые обеспечат простое и удобное взаимодействие между собой, что позволит пользователю следить за своей успеваемостью, а также работать с отработками, пересдачами удаленно, что упрощает учебный процесс. Курсовой проект содержит 4 роли: студент, студент-староста, преподаватель, деканат.

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](#_Toc27830404)

[Введение 4](#_Toc27830405)

[1 Аналитический обзор схожих приложений](#_Toc27830406) 5

[2 Проектирование и разработка базы данных](#_Toc27830407) 7

[2.1 Спецификация функциональных требований](#_Toc27830408) 7

[2.2 Разработка модели базы данных](#_Toc27830409) 9

[2.3 Пользователи](#_Toc27830410) 12

[2.4 Хранимые процедуры](#_Toc27830408) 12

[2.4.1 Выборка данных для таблиц](#_Toc27830409) 12

[2.4.2 Заполнение таблиц 100 000 строк](#_Toc27830410) 13

[2.4.3 Добавление пользователя и проверка корректности](#_Toc27830411) 13

[2.4.4 Добавление отработки/пересдачи](#_Toc27830434) 13

[2.4.5 Удаление данных из таблиц](#_Toc27830435) 14

[2.4.6 Добавление данных в таблицы](#_Toc27830436) 14

[2.4.7 Изменение данных в таблицах](#_Toc27830411) 15

[2.4.8 Эскпорт и импорт таблиц в формат xml](#_Toc27830434) 15

[2.5 Реализация технологий](#_Toc27830408) 17

[2.5.1 Реализация технологии мультимедийные типы данных в БД](#_Toc27830409) 17

[3 Проектирование и разработка программного средства](#_Toc27830436) 18

[4 Тестирование, проверка работоспособности и анализ данных 2](#_Toc27830404)0

[Заключение 29](#_Toc27830405)

[Список литературы](#_Toc27830406) 30

[Приложение А](#_Toc27830407) 31

[Приложение Б 3](#_Toc27830405)2

[Приложение В](#_Toc27830406) 39

[Приложение Г](#_Toc27830407) 40

[Приложение Д](#_Toc27830406) 41

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И. О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Введение

Лит.

Листов

1

БГТУ

## **Введение**

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

В данной работе разработано ПО, которое позволяет студенту выполнить запрос на отработку, пересдачу, просматривать свою успеваемость. В университете не малую роль играют староста группы и преподаватель. Староста следит за работой своей группы и выставляет пропуски для поддержания порядка в учебном процессе, а преподаватель выставляет оценки студентам и обрабатывает отработки, пересдачи в удаленном режиме.

Целью данной работы является создание приложения, которое работает с личными данными пользователя и упрощает работу с деканатом университета.

Задачей данной курсовой работы является разработка базы данных предназначенной для приложения «Студенты университета». Приложение будет обладать следующими функционалом:

* авторизация и регистрация пользователей;
* составление интерфейса для взаимодействия с БД;
* возможность работы с отработками, пересдачами, пропусками;
* разделение ролей пользователей;
* шифрование данных клиента.

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

СУБД существует огромное множество: Oracle, MS SQL Server, Microsoft Access, MySql и так далее. В данной работе будет использовано решение Oracle Database 12c.

## **1 Аналитический обзор схожих приложений**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И. О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

1 Аналитический обзор схожих приложений

Лит.

Листов

1

БГТУ

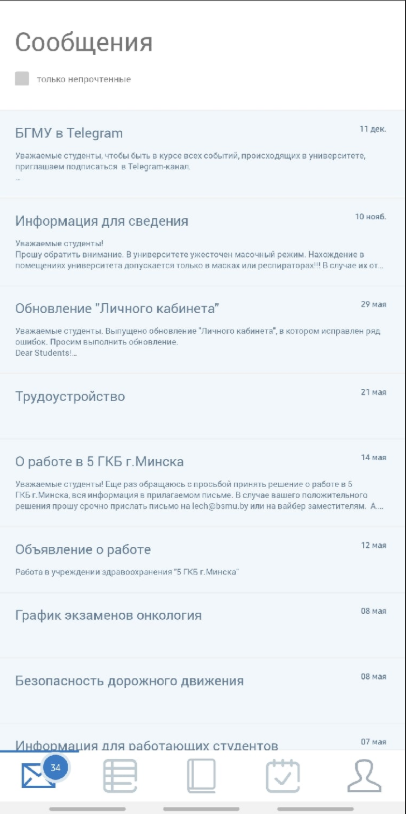
Для составления технических требований к проекту был проведен анализ похожих программных средств. Из открытого источника (интернет) было найдено Android приложение для работы со студентами – Личный кабинет студента БГМУ.

Рисунок 1.1 – Android-приложение ЛК БГМУ

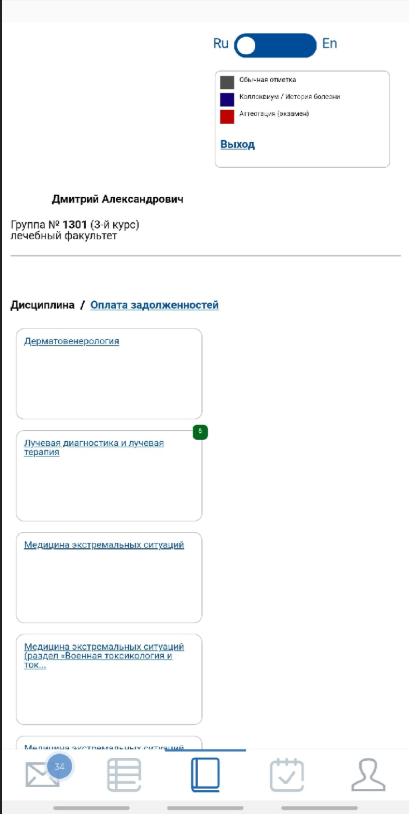


Рисунок 1.2 – Вкладка учебный процесс

Функционально приложение включает в себя:

* Просмотр оценок;
* Просмотр новостей;
* Скачивание расписания;
* Обратная связь.

Вышеназванное приложение обладает схожим функционалом для облегчения выполнения необходимых операций, связанных с учебным процессом. Однако StudentHub позволяет работать с отработками, пересдачами, оценками, пропусками удаленно.

**2 Проектирование и разработка базы данных**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И.О.

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. Хранимые процедуры;

3. Индексы;

4. Функции;

5. Последовательности;

6. Триггеры;

7. Пакеты.

**2.1 Спецификация функциональных требований**

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

UML – 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. [4]

На рисунке 2.1.1 показана UML-диаграмма, отображающая возможности приложения с точки зрения неавторизованного пользователя.

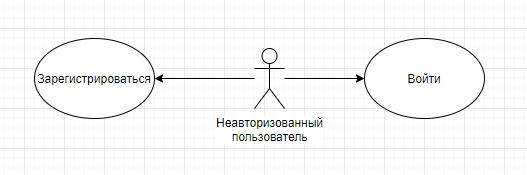


Рисунок 2.1.1 – UML диаграмма вариантов использования для неавторизованного пользователя

Неавторизованный пользователь имеет только две возможности: авторизоваться и зарегистрироваться.

Если пользователь авторизуется, то будет иметь возможности, представленные на рисунке 2.2.2.

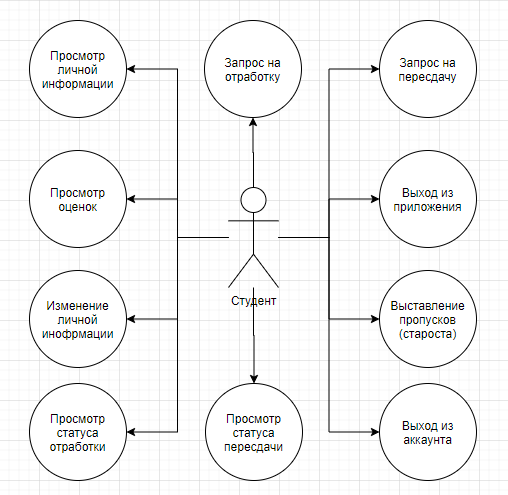


Рисунок 2.2.2 – UML диаграмма вариантов использования для студента

Студент может отправлять запрос на отработку, пересдачу, просматривать и изменять личную информацию, просматривать оценки, статусы отработок, пересдач, выходить из приложения или учетной записи, также если студент является старостой, то он может выставлять пропуски студентам своей группы.

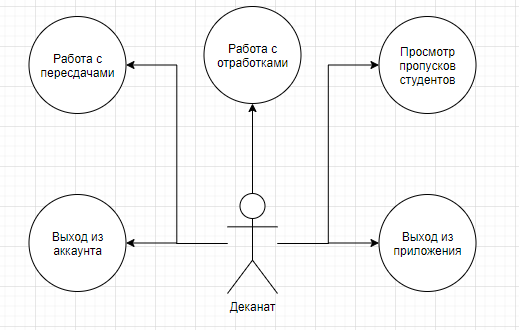
В приложение можно войти как деканат. Тогда открываются следующие возможности: работа с отработками, пересдачами, просмотр количества пропусков студентов, выход из приложения или учетной записи. Диаграмма вариантов использования для деканата отображена на рисунке 2.3.3.

Рисунок 2.2.3 – UML диаграмма вариантов использования для деканата

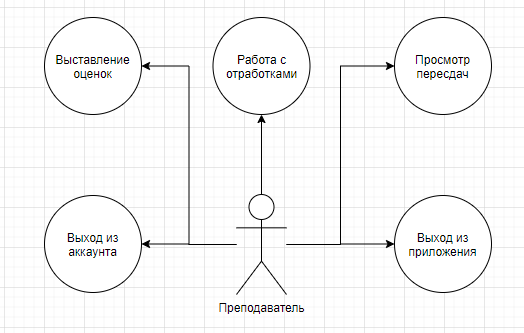
Также в приложении существует роль преподавателя. Преподаватель может просматривать отработки, ставить дату для проведения их, просматривать предстоящие пересдачи, их статусы и даты, выставлять студентам оценки по предметам, а также выйти из приложения или учетной записи. Диаграмма вариантов использования для деканата отображена на рисунке 2.3.4.

Рисунок 2.2.4 – UML диаграмма вариантов использования для преподавателя

**2.2 Разработка модели базы данных**

Для реализации базы данных для StudentHub было разработано 12 таблиц. Диаграмма базы со структурой связей представлена в приложении А, а в приложении Б представлен скрипт создания БД.

Логически можно вывести 5 основных таблиц: users, adjustments, retakes, student\_progress и gaps.

Таблица users представляет перечень пользователей, состоит из столбцов:

−id – идентификатор пользователя, тип number, первичный ключ;

− login – имя пользователя, тип varchar2;

− password – пароль пользователя, тип varchar2;

− role – роль пользователя, тип varchar2.

Таблица adjustments представляет информацию об отработках, состоит из столбцов:

−id – идентификатор отработки, тип number, первичный ключ;

− user\_id – идентификатор пользователя, тип number, вторичный ключ;

− teacher\_id – идентификатор преподавателя, тип number, вторичный ключ;

− subject – название предмета, тип char;

− status – статус отработки, тип varchar2;

− adjustment\_date –дата отработки, тип date;

− access\_date –дата принятия отработки, тип date;

− filing\_date – дата подачи запроса на отработку, тип date;

− img ­– изображение с заявлением, тип blob.

Таблица retakes представляет информацию о пересдачах, состоит из столбцов:

−id – идентификатор пересдачи, тип number, первичный ключ;

− user\_id – идентификатор пользователя, тип number, вторичный ключ;

− teacher\_id – идентификатор преподавателя, тип number, вторичный ключ;

− subject – название предмета, тип char;

− status – статус отработки, тип varchar2;

− retake\_date –дата пересдачи, тип date;

− img ­– изображение с заявлением, тип blob.

Таблица student\_progress представляет информацию о успеваемости студентов, состоит из столбцов:

−id – идентификатор успеваемости, тип number, первичный ключ;

− user\_id – идентификатор пользователя, тип number, вторичный ключ;

− subject – название предмета, тип char;

− progress\_date –дата выставления оценки, тип date;

− feedback –комментарий, тип varchar2.

Таблица gaps представляет информацию о пропусках студентов, состоит из столбцов:

−id – идентификатор записи, тип number, первичный ключ;

− user\_id – идентификатор пользователя, тип number, вторичный ключ;

− subject – название предмета, тип char;

− gap\_date –дата выставления пропуска, тип date;

− gap\_count – количество пропусков, тип number.

Так же есть еще 7 таблиц для поддержания всей остальной логической структуры базы данных.

Таблица student\_info представляет информацию о студенте, состоит из столбцов:

−id – идентификатор студента, тип number, первичный ключ;

− user\_id – идентификатор пользователя, тип number, вторичный ключ;

− student\_name – фамилия, имя, отчество студента, тип varchar2;

− status – статус в группе, тип varchar2;

− course – курс, на котором учится студент, тип number;

− num\_group – номер группы, в которой учится студент;

− specialization – специальность, на которой учится студент, тип varchar2;

− faculty – факультет, на котором учится студент, тип char;

– birthday – день рождения студента, тип date.

Таблица deanery\_info представляет информацию о деканате, состоит из столбцов:

−id – идентификатор деканата, тип number, первичный ключ;

− user\_id – идентификатор пользователя, тип number, вторичный ключ;

− deanery\_name – название деканата;

− telephone – контактная информация, тип varchar2;

− faculty – название факультета, тип char.

Таблица teacher\_info представляет информацию о преподавателе, состоит из столбцов:

−id – идентификатор преподавателя, тип number, первичный ключ;

− user\_id – идентификатор пользователя, тип number, вторичный ключ;

− teacher\_name – фамилия, имя, отчество преподавателя;

− telephone – контактная информация, тип varchar2;

− faculty – название факультета, тип char.

Таблица subjects представляет информацию о предметах, состоит из столбцов:

−subject – аббревиатура предмета, тип char, первичный ключ;

− subject\_name – название предмета, тип varchar2;

− faculty – название факультета, тип char, вторичный ключ.

Таблица faculty представляет информацию о факультетах, состоит из столбцов:

−faculty – аббревиатура факультета, тип char, первичный ключ;

− faculty\_name – название факультета, тип varchar2.

Таблица specialization представляет информацию о специальностях, состоит из столбцов:

−specialization – аббревиатура специальности, тип char, первичный ключ;

− name – название специальности, тип varchar2.

Таблица faculty\_specialization представляет связку факультета и специальности, состоит из столбцов:

– id – идентификатор записи, тип number;

−faculty – аббревиатура факультета, тип char, вторичный ключ;

− specialization – аббревиатура специальности, тип char, вторичный ключ.

# **2.3 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 было использовано 2 пользователя. Первый пользователь – StudentHubUser – обычный пользователь, имеет доступ для чтения таблиц, связанных с отработками, пересдачами, оценками, пропусками, права для вставки в таблицу запросов на отработку, пересдачу, обновление таблиц.

Второй пользователь – StudentHubAdmin – обладает более обширным перечнем прав. Имеет права для чтения, изменения и записи во всех таблицах.

**2.4 Хранимые процедуры**

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

При разработке курсового проекта было создано 18 процедур для следующих целей:

1. Выборка данных для таблиц;

2. Заполнение таблиц 100 000 строк;

3. Добавление пользователя и проверка корректности;

4. Добавление отработки/пересдачи;

5. Удаление данных из таблиц;

6. Добавление данных в таблицы;

7. Изменение данных в таблицах;

8. Экспорт и импорт таблиц в формат xml;

Весь перечень созданных процедур будет представлен в Приложении В.

**2.4.1 Выборка данных для таблиц**

Для вывода данных из таблиц были написаны следующие процедуры: FindUser, FindStudent, FindDeanery, FindTeacher и другие. Поскольку вывод таблицы размером в 100000 строк занимает довольно много времени, было принято решение создать индексы, что позволило уменьшить время вывода таблицы до минимума.

**2.4.2 Заполнение таблиц 100 000 строк**

Для заполнения таблиц были разработаны процедуры для генерации случайных строк и процедуры заполнения таблиц этими строками. Процедура заполнения таблицы users: generateUsers.

**2.4.3 Добавление пользователя и проверка корректности**

При регистрации нового пользователя с помощью процедуры addUser, ему необходимо заполнить необходимые поля: login, password. Для увеличения безопасности учетной записи пользователя его пароль находится в захэшированном виде. Хеширование пароля происходит за счет работы с пакетом DBMS\_CRYPTO. Также проверяется корректность ввода с помощью select запроса. Данная процедура представлена ниже на рисунке 2.4.1.

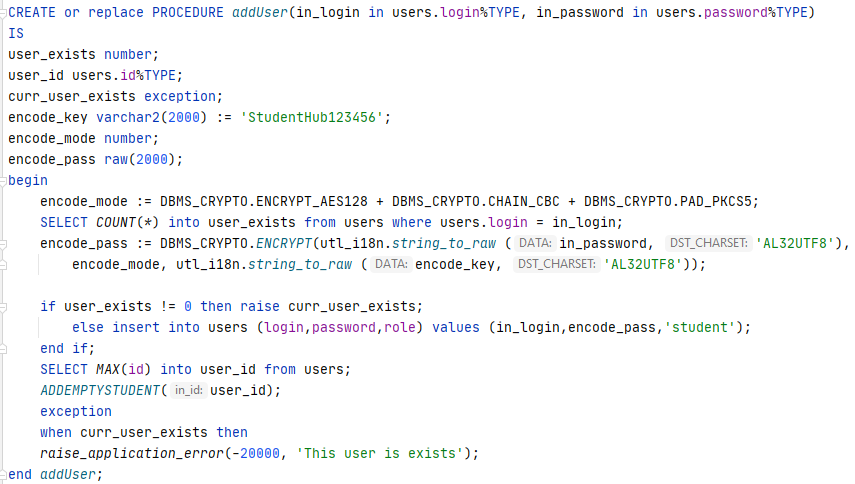


Рисунок 2.4.1 – Пример создания процедуры addUser

**2.4.4 Добавление отработки/пересдачи**

Для добавления отработки/пересдачи были разработаны процедуры addAdjustment, addRetake соответственно. Данные процедуры представлены ниже на рисунке 2.4.2.

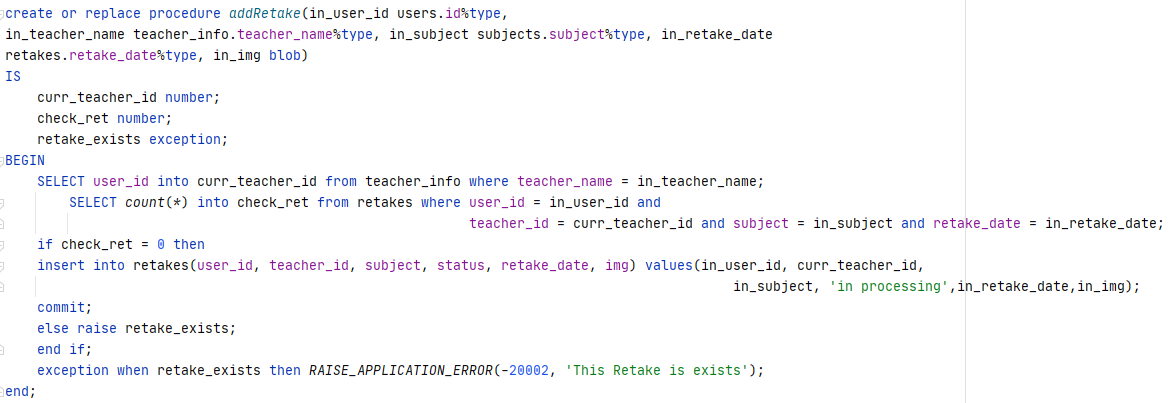
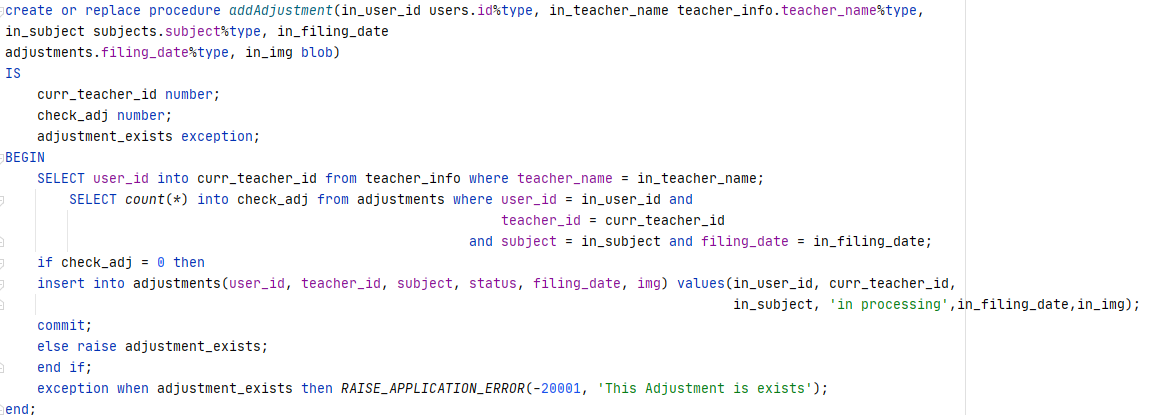
 

Рисунок 2.4.2 – Пример создания процедур addAdjustment, addRetake

**2.4.5 Удаление данных из таблиц**

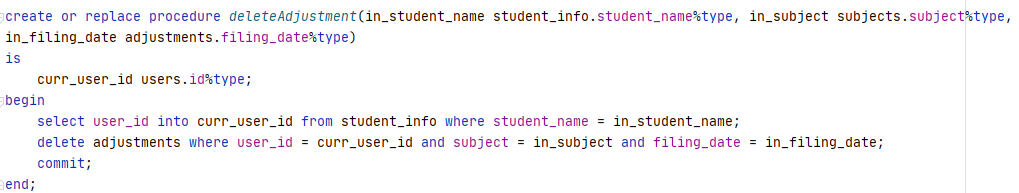
Для управления базой данных была создана процедура для удаления отработок deleteAdjustment. Она удаляет строки из таблицы adjustments. Данная процедура представлена ниже на рисунке 2.4.3.

Рисунок 2.4.3 – Пример создания процедуры deleteAdjustment

**2.4.6 Добавление данных в таблицы**

Для управления базой данных были созданы процедуры добавления новых строк: addUser, addEmptyStudent, addAdjustment, addRetake, setGaps, setRatings. Они были разработаны для заполнения соответствующих таблиц.

Процедура setRatings позволяет преподавателю поставить оценки студентам. Данная процедура представлена ниже на рисунке 2.4.4.

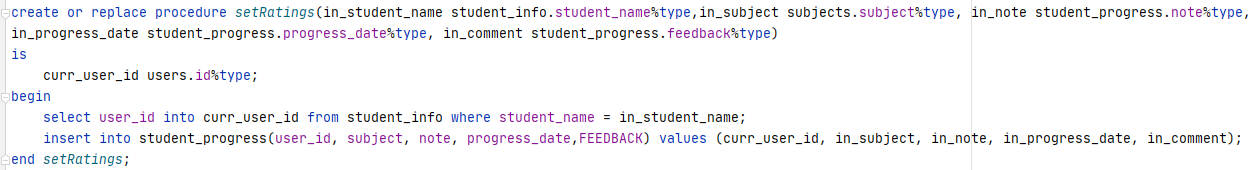
****

Рисунок 2.4.4 – Пример создания процедуры setRatings

**2.4.7 Изменение данных в таблицах**

Для управления базой данных были созданы процедуры обновления: updateStudent, accept\_decline\_Adjustment, accept\_decline\_Retake, accept\_decline\_Adjustment\_Teacher. Они были разработаны для обновления данных в соответствующих таблицах.

**2.4.8 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 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 таблицы Users, так как эта таблица является важнейшей в базе данных.Пример создания процедур экспорта, импорта можно посмотреть на рисунках 2.4.5, 2.4.6.

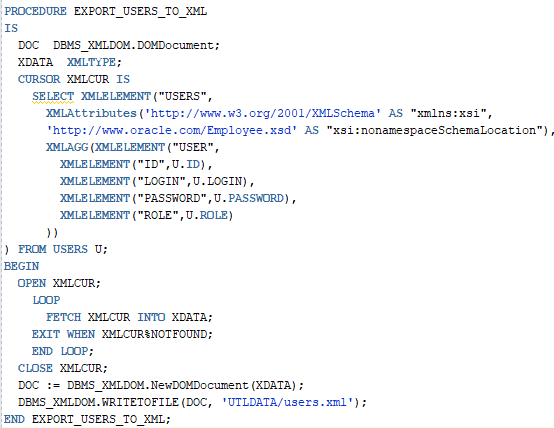


Рисунок 2.4.5 – Пример создания процедуры EXPORT\_USERS\_TO\_XML

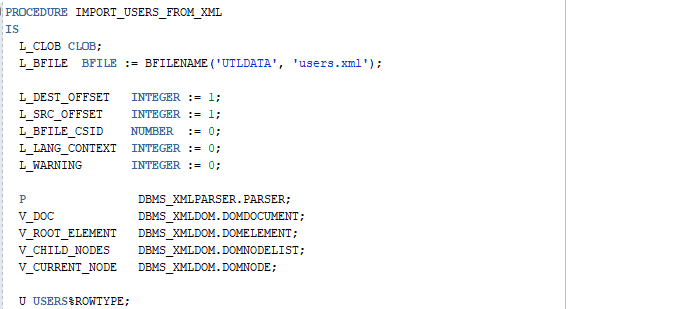
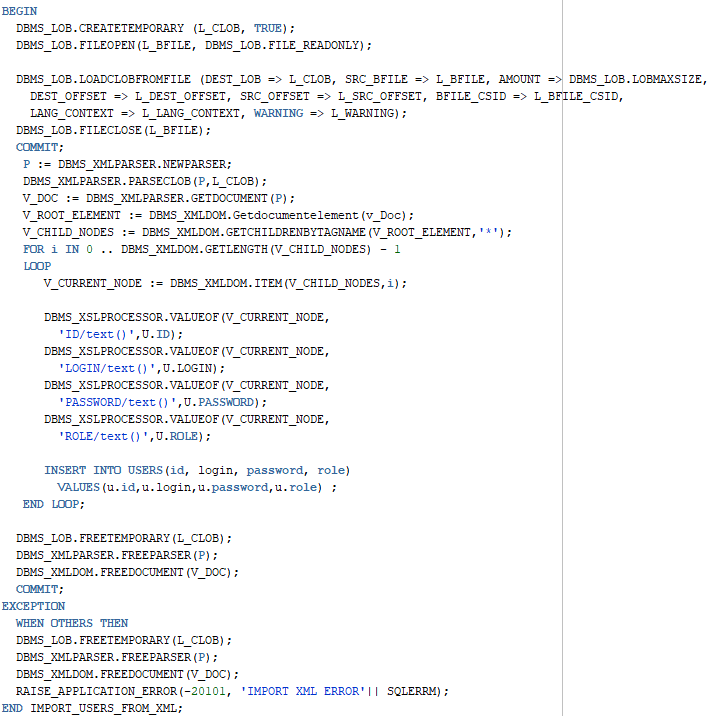


Рисунок 2.4.6 – Пример создания процедуры IMPORT\_USERS\_FROM\_XML

**2.5 Реализация технологий**

В процессе разработки проекта была разработана технология: мультимедийные типы данных в БД.

**2.5.1** **Реализация технологии мультимедийные типы данных в БД**

Oracle database 12c поддерживает несколько мультимедийных типов данных: ORDAudio, ORDImage, ORDVideo, ORDDoc, а также можно осуществить запись с помощью blob (большой плюс - транзакционный контроль). Экземляры ORDAudio, ORDDoc, ORDImage и ORDVideo состоят из атрибутов (включает метаданные и медиаданные) и методов. Медиаданные - это реальное видео, аудио или графическое изображение. Метаданные - это информация о данных: длина объекта, метод компрессии, формат. Методы - это процедуры, которые могут быть выполнены для объекта, такие как getContent() и setProperties().

В данной курсовой работе мультимедийные типы данных использовались в таблицах adjustments, retakes. Для хранения использовался тип данных blob. Пример процедуры, где была реализована данная технология, представлен ниже на рисунке 2.5.1

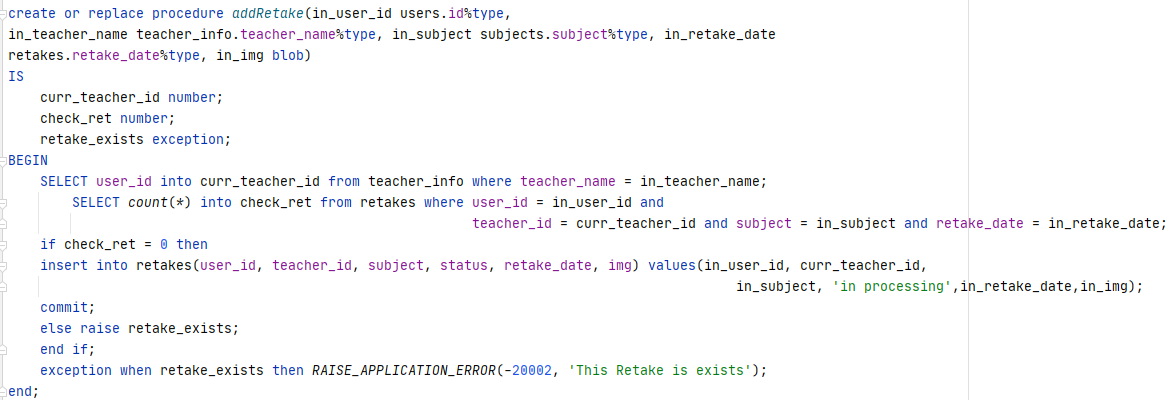


Рисунок 2.5.1 – Пример реализации технологии мультимедийные типы данных

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И.О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

3 Проектирование и разработка программного средства

Лит.

Листов

1

БГТУ

**3 Проектирование и разработка программного средства**

В процессе выполнения курсового проекта было разработано desktop-приложение, предназначенное для взаимодействия пользователя с базой данных. Приложение было разработано на языке программирования C# с использованием таких технологий как .NET Framework, WPF и ADO.NET. Структура разработанного программного средства представлена ниже.

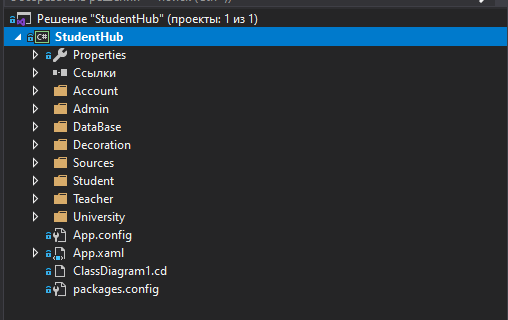


Рисунок 3.1 – Структура проекта

Описание обобщенной структуры проекта представлено в таблице 3.1.

Таблица 3.1 – Обобщенная структура проекта

|  |  |
| --- | --- |
| Объект | Описание |
| Properties | Свойства проекта, содержит информацию о сборке, используемых ресурсах и настройках |
| Ссылки | Перечень сборок, используемых в проекте |
| Account | Содержит в себе окна для работы с аккаунтом, а именно авторизация и регистрация |
| Admin | Содержит в себе окна для администратора |

|  |  |
| --- | --- |
| DataBase | Содержит в себе класс для работы с базой данных |
| Decoration | Содержит в себе словари ресурсов (XAML) для оформления ЭУ и не только |
| Sources | Содержит в себе 3 подпапки, в которых находятся необходимые объекты для оформления приложения (иконки, рисунки, словари ресурсов) |
| Student | Содержит в себе окна для студента |
| Teacher | Содержит в себе окна для преподавателя |

Продолжение таблицы 3.1

Для взаимодействия через приложения с данными из БД была применена технология ADO.NET. ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. 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. Схематично архитектуру ADO.NET можно представить следующим образом [9].



Рисунок 3.2 – Архитектура ADO.NET

Для установки соединения с нашей базой данных нам необходимо было создать класс OracleDataBaseConnection, который хранил в себе необходимые параметры для подключения к базе данных Oracle.

**4 Тестирование, проверка работоспособности и анализ данных**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И.О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

4 Тестирование, проверка работоспособности и анализ данных

Лит.

Листов

1

БГТУ

Разработанное программное средство полностью соответствует созданному для него техническому заданию, также оно использует дополнительно разработанный функционал.

Для проверки корректности работы программного средства, продемонстрируем добавление нового пользователя.

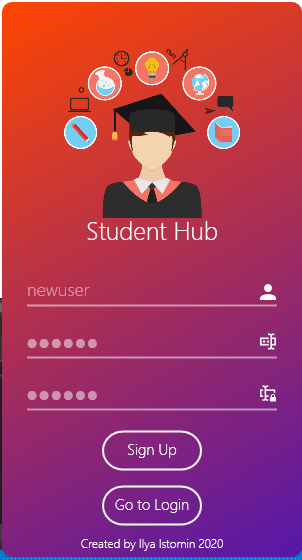


Рисунок 4.1 – Окно регистрации нового пользователя

После нажатия на кнопку «Sign Up» клиент заносится в базу данных, при этом пароль хэшируется, так как в программе имеется функция хэширования описанная выше.



Рисунок 4.2 – Запись только что зарегистрированного пользователя

После того, как пользователь зарегистрировался, открывается начальная страница, показанная на рисунке 4.3.

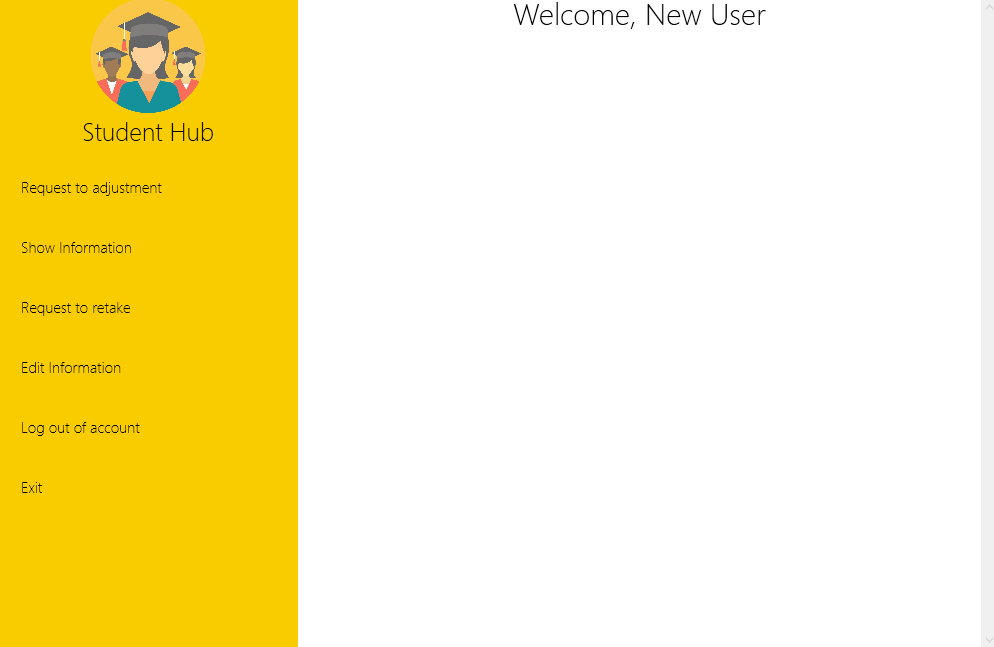


Рисунок 4.3 – Начальная страница

У пользователя доступны вкладки Request to adjustment, Show Information, Request to retake, Edit information, Log out of account, Exit. При переходе во вкладку Request to adjustment открывается окно с полями для заполнения запроса на отработку.

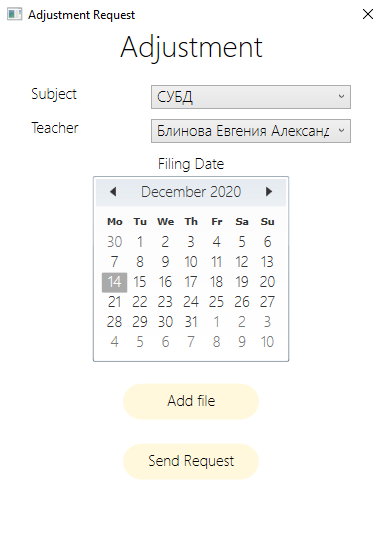


Рисунок 4.4 – Окно для запроса на отработку

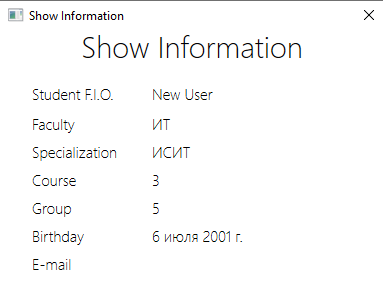
 Пользователь может просмотреть личную информацию.

Рисунок 4.5 – Просмотр информации

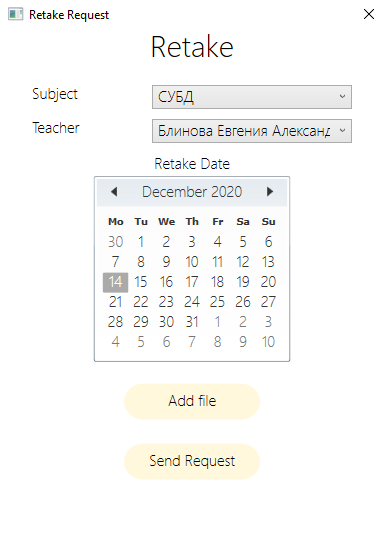
При переходе во вкладку Request to retake открывается окно с полями для заполнения запроса на пересдачу.

Рисунок 4.6 – Окно для запроса на пересдачу

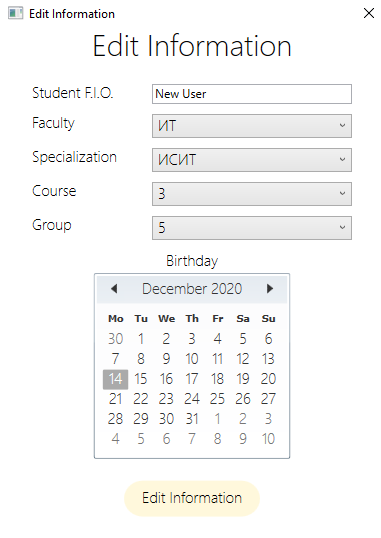
Пользователь может изменять личную информацию.

Рисунок 4.6 – Окно для изменения личной информации

При отправке запроса на отработку/пересдачу запись записывается в таблицу adjustments/retakes соответственно и отображается в главном окне.

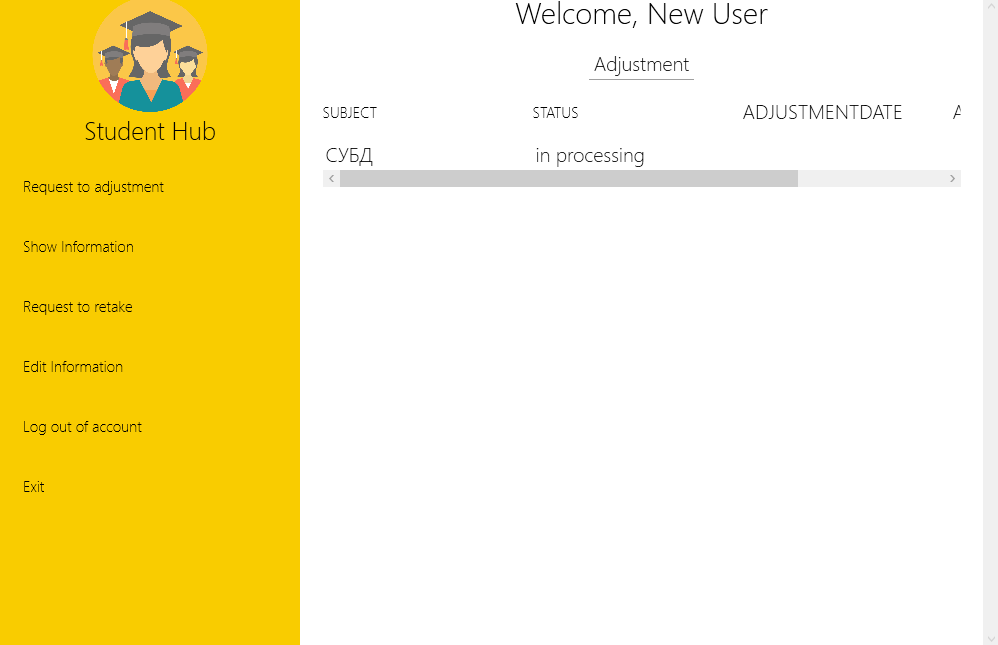


Рисунок 4.7 – Главное окно с отображением отработки

После чего деканат должен рассмотреть запрос и вынести решение – отклонить или принять, в зависимости от того, как введены данные. Далее преподаватель должен в личном кабинете, если запрос принят, выставить день для отработки занятий.

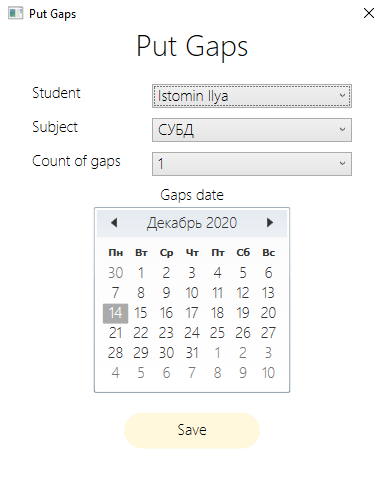
Также в приложении есть роль старосты, который может выставлять пропуски своим одногруппникам.

Рисунок 4.8 – Окно для выставления пропусков

Для завершения работы или для переключения на сотрудника деканата необходимо выйти из приложения, нажав на кнопку Log out of account или Exit.

При входе в приложение под профилем администратора открывается совершенно другое окно предоставляющее другой функционал.

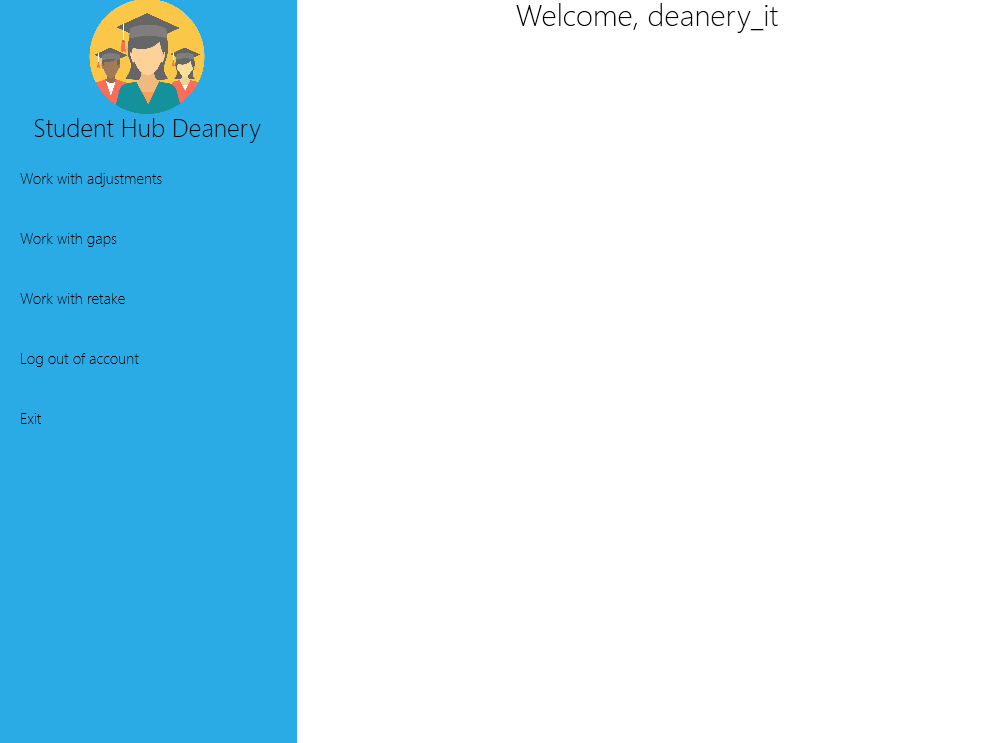


Рисунок 4.9 – Начальное окно для администратора

Рисунок 4.10 – Окно для работы с отработками

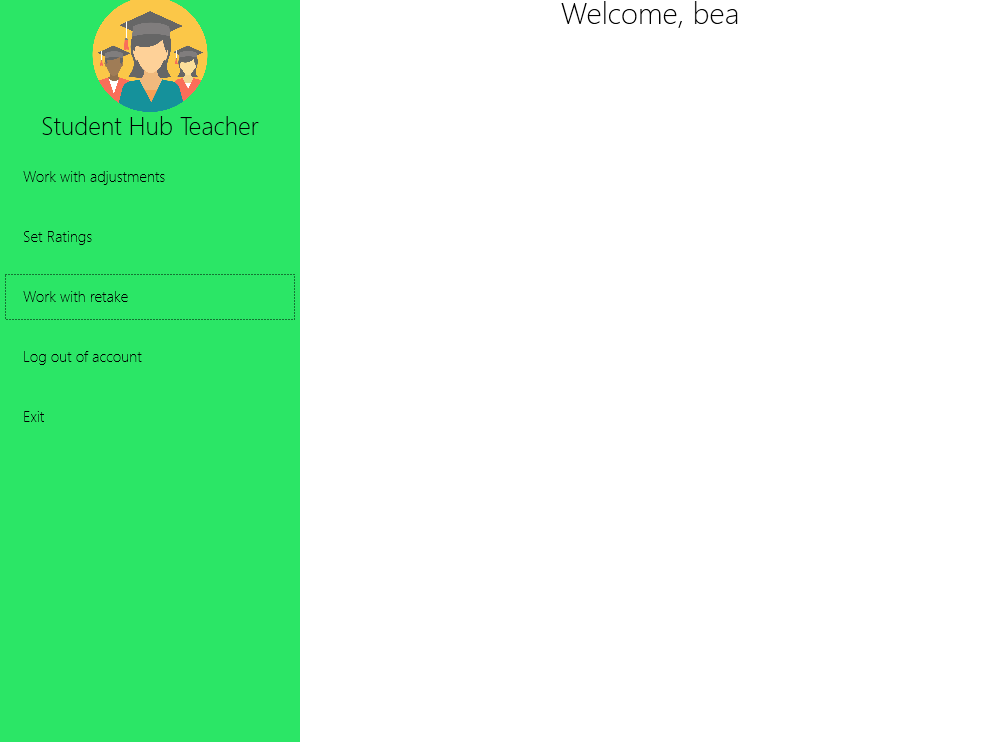


Рисунок 4.11 – Окно просмотра количества пропусков

Рисунок 4.12 – Окно для работы с пересдачами

Для завершения работы или для переключения на преподавателя необходимо выйти из приложения, нажав на кнопку Log out of account или Exit.

При входе в приложение под профилем преподавателя открывается совершенно другое окно предоставляющее другой функционал.



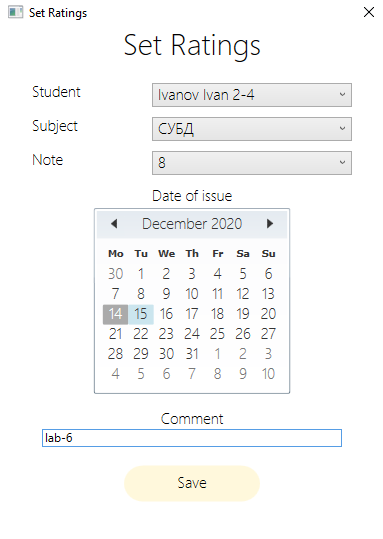
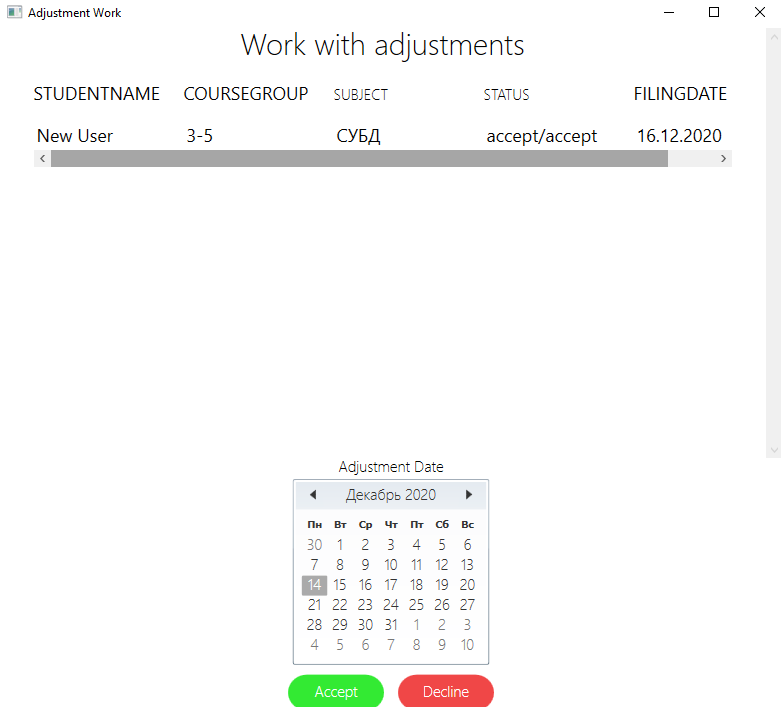
Рисунок 4.13 – Начальное окно для преподавателя

Рисунок 4.14 – Окно для выставления оценок



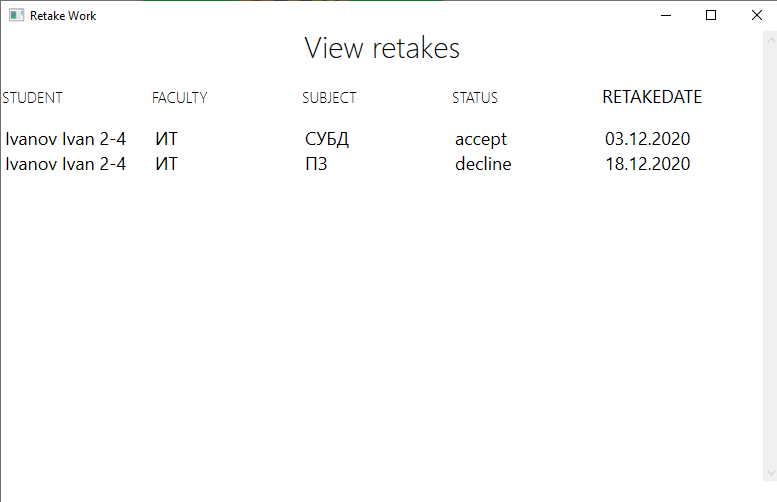
Рисунок 4.15 – Окно для работы с отработками

Рисунок 4.16 – Окно для просмотра пересдач

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И.О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Заключение

Лит.

Листов

1

БГТУ

**Заключение**

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных для программного средства «Студенты университета», которая в совокупности с приложением формирует полноценное desktop-приложение для управления учебным процессом. 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 выполнены следующие пункты:

* авторизация и регистрация пользователей;
* составление интерфейса для взаимодействия с БД;
* возможность работы с отработками, пересдачами, пропусками;
* разделение ролей пользователей;
* шифрование данных клиента.

Приложение прошло тестирование при использовании в БД большого количество данных.

Данный проект сложно сравнить с аналогами, ведь для каждого университета система разрабатывается индивидуально. По сути, данный проект является показательным примером базы данных для различных университетов, позволяя охватить необходимый минимум требований для их администрирования.

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

Пакеты,индексы, заключение, литература, ссылки на литературу []

**Список литературы**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И.О..

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Список литературы

Лит.

Листов

1

БГТУ

1. Сайт ДО belstu.by – https://dist.belstu.by.

2. Интернет ресурс для проектирования и создания UML-диаграмм – https://app.diagrams.net

3. Сайт o7planning.org – https://o7planning.org/ru/10519/working-with-oracle-database-using-csharp

4. Oracle docs – https://docs.oracle.com/en/database/

5. Сайт interface.ru – http://www.interface.ru/home.asp?artId=20889

6. METANIT – https://metanit.com/sharp/adonet/1.1.php

7. METANIT SQL – https://metanit.com/sql/.

8. METANIT C# – https://metanit.com/sharp/.

9. MSDN C# – https://docs.microsoft.com/ru-ru/dotnet/csharp/.

**Приложение А**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И. О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

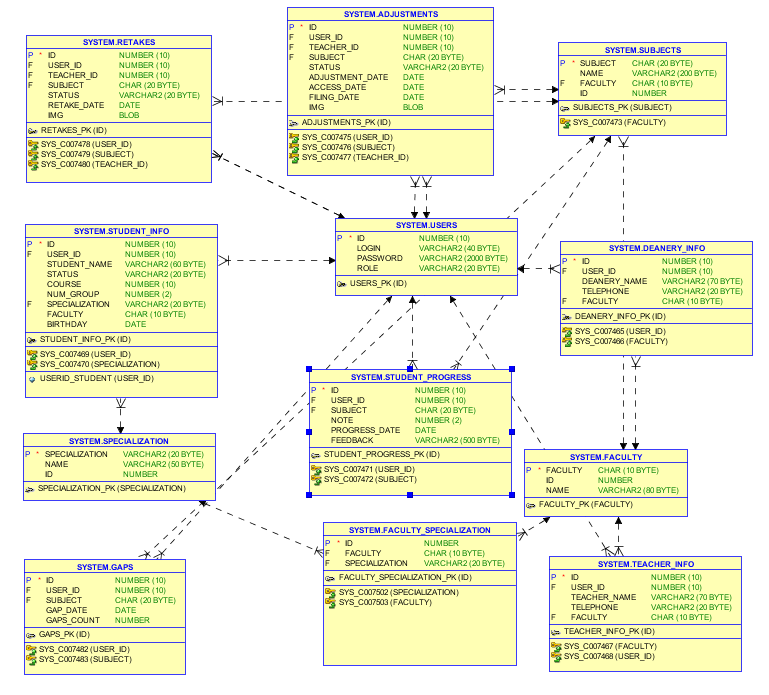
Приложение А

Лит.

Листов

1

БГТУ

****

**Приложение Б**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И. О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение Б

Лит.

Листов

1

БГТУ

*---  
-- Table 'users'  
--  
---*BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE users';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE users (  
id NUMBER(10) DEFAULT NULL NULL,  
login VARCHAR2(40) DEFAULT NULL NULL,  
password VARCHAR2(2000) DEFAULT NULL NULL,  
role VARCHAR2(20) DEFAULT NULL NULL,  
PRIMARY KEY (id)  
);  
alter table users add constraint role check ( role in ('student','deanery','teacher') );  
  
*-- 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.id IS NULL)  
BEGIN  
SELECT users\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
  
  
*--student info*BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE student\_info';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE student\_info (  
id NUMBER(10) DEFAULT NULL NULL,  
user\_id NUMBER(10) DEFAULT NULL NULL,  
student\_name VARCHAR2(60) DEFAULT NULL NULL,  
status varchar2(20) DEFAULT NULL NULL,  
course NUMBER(10) DEFAULT NULL NULL,  
num\_group NUMBER(2) DEFAULT NULL NULL,  
specialization VARCHAR2(20) DEFAULT NULL NULL,  
faculty CHAR(10) DEFAULT NULL NULL,  
birthday DATE DEFAULT NULL NULL,  
PRIMARY KEY (id)  
);

*-- Generate ID using sequence and trigger*CREATE SEQUENCE student\_info\_seq START WITH 1 INCREMENT BY 1;  
  
CREATE OR REPLACE TRIGGER student\_info\_seq\_tr  
BEFORE INSERT ON student\_info FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT student\_info\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
  
  
*---  
-- Table 'deanery\_info'  
--  
---*BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE deanery\_info';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE deanery\_info (  
id NUMBER(10) DEFAULT NULL NULL,  
user\_id NUMBER(10) DEFAULT NULL NULL,  
deanery\_name varchar2(70) DEFAULT NULL NULL,  
telephone VARCHAR2(20) DEFAULT NULL NULL,  
faculty char(10) default null null,  
PRIMARY KEY (id)  
);  
  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE deanery\_info\_seq START WITH 1 INCREMENT BY 1;  
  
CREATE OR REPLACE TRIGGER deanery\_info\_seq\_tr  
BEFORE INSERT ON deanery\_info FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT deanery\_info\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
  
BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE teacher\_info';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
CREATE TABLE teacher\_info (  
id NUMBER(10) DEFAULT NULL NULL,  
user\_id NUMBER(10) DEFAULT NULL NULL,  
teacher\_name varchar2(70) DEFAULT NULL NULL,  
telephone VARCHAR2(20) DEFAULT NULL NULL,  
faculty char(10) default null null,  
PRIMARY KEY (id)  
);  
  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE teacher\_info\_seq START WITH 1 INCREMENT BY 1;  
  
CREATE OR REPLACE TRIGGER teacher\_info\_seq\_tr  
BEFORE INSERT ON teacher\_info FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT teacher\_info\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE student\_progress';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE  
student\_progress (  
id NUMBER(10) DEFAULT NULL NULL,  
user\_id NUMBER(10) DEFAULT NULL NULL,  
subject CHAR(20) DEFAULT NULL NULL,  
note NUMBER(2) DEFAULT NULL NULL,  
progress\_date DATE DEFAULT NULL NULL,  
PRIMARY KEY (id)  
);  
  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE student\_progress\_seq START WITH 1 INCREMENT BY 1;  
  
CREATE OR REPLACE TRIGGER student\_progress\_seq\_tr  
BEFORE INSERT ON student\_progress FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT student\_progress\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
  
*---  
-- Table 'subjects'  
--  
---*BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE subjects';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE subjects (  
id number,  
subject CHAR(20) DEFAULT NULL NULL,  
name VARCHAR2(200) DEFAULT NULL NULL,  
faculty CHAR(10) DEFAULT NULL NULL,  
PRIMARY KEY (subject)  
);  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE subjects\_seq START WITH 1 INCREMENT BY 1;  
CREATE OR REPLACE TRIGGER subjects\_seq\_tr  
BEFORE INSERT ON subjects FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT subjects\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
  
  
*---  
-- Table 'faculty\_specialization'  
--  
---*BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE faculty\_specialization';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE faculty\_specialization (  
id number DEFAULT NULL NULL,  
faculty char(10) DEFAULT NULL NULL,  
specialization varchar2(20) DEFAULT NULL NULL,  
PRIMARY KEY (id)  
);  
  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE faculty\_specialization\_seq START WITH 1 INCREMENT BY 1;  
  
CREATE OR REPLACE TRIGGER faculty\_specialization\_seq\_tr  
BEFORE INSERT ON faculty\_specialization FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT faculty\_specialization\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
  
*---  
-- Table 'faculty'  
--  
---*BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE faculty';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE faculty (  
id number,  
faculty CHAR(10) DEFAULT NULL NULL,  
name varchar2(80) DEFAULT NULL NULL,  
PRIMARY KEY (faculty)  
);  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE faculty\_seq START WITH 1 INCREMENT BY 1;  
CREATE OR REPLACE TRIGGER faculty\_seq\_tr  
BEFORE INSERT ON faculty FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT faculty\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
  
 *--  
---*BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE specialization';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE specialization (  
specialization VARCHAR2(20) DEFAULT NULL NULL,  
name VARCHAR2(50) DEFAULT NULL NULL,  
PRIMARY KEY (specialization)  
);  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE specialization\_seq START WITH 1 INCREMENT BY 1;  
  
CREATE OR REPLACE TRIGGER specialization\_seq\_tr  
BEFORE INSERT ON specialization FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT specialization\_seq.*NEXTVAL* INTO :NEW.id FROM DUAL;  
END;  
  
  
*---  
-- Table 'adjustments'  
--  
---*BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE adjustments';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
/  
  
CREATE TABLE adjustments (  
id NUMBER(10) DEFAULT NULL NULL,  
user\_id NUMBER(10) DEFAULT NULL NULL,  
teacher\_id NUMBER(10) DEFAULT NULL NULL,  
subject CHAR(20) DEFAULT NULL NULL,  
status varchar2(20) DEFAULT NULL NULL,  
adjustment\_date DATE DEFAULT NULL NULL,  
access\_date DATE DEFAULT NULL NULL,  
filing\_date DATE default null null,  
img blob,  
PRIMARY KEY (id)  
);  
  
  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE adjustments\_seq START WITH 1 INCREMENT BY 1;  
  
CREATE OR REPLACE TRIGGER adjustments\_seq\_tr  
BEFORE INSERT ON adjustments FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT adjustments\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
  
*---  
-- Table 'retakes'  
--  
---*BEGIN  
EXECUTE IMMEDIATE 'DROP TABLE retakes';  
EXCEPTION  
WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE retakes (  
id NUMBER(10) DEFAULT NULL NULL,  
user\_id NUMBER(10) DEFAULT NULL NULL,  
teacher\_id NUMBER(10) DEFAULT NULL NULL,  
subject CHAR(20) DEFAULT NULL NULL,  
status varchar2(20) DEFAULT NULL NULL,  
retake\_date DATE DEFAULT NULL NULL,  
img blob,  
PRIMARY KEY (id)  
);  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE retakes\_seq START WITH 1 INCREMENT BY 1;  
  
CREATE OR REPLACE TRIGGER retakes\_seq\_tr  
BEFORE INSERT ON retakes FOR EACH ROW  
WHEN (NEW.id IS NULL)  
BEGIN  
SELECT retakes\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
*-- ---  
-- Table 'gaps'  
--  
-- ---*BEGIN  
 EXECUTE IMMEDIATE 'DROP TABLE gaps';  
EXCEPTION  
 WHEN OTHERS THEN NULL;  
END;  
  
  
CREATE TABLE gaps (  
 id NUMBER(10) DEFAULT NULL NULL,  
 user\_id NUMBER(10) DEFAULT NULL NULL,  
 subject CHAR(20) DEFAULT NULL NULL,  
 gap\_date DATE DEFAULT NULL NULL,  
 gaps\_count number,  
 PRIMARY KEY (id)  
);  
  
*-- Generate ID using sequence and trigger*CREATE SEQUENCE gaps\_seq START WITH 1 INCREMENT BY 1;  
  
CREATE OR REPLACE TRIGGER gaps\_seq\_tr  
 BEFORE INSERT ON gaps FOR EACH ROW  
 WHEN (NEW.id IS NULL)  
BEGIN  
 SELECT gaps\_seq.*NEXTVAL* INTO :NEW.**id** FROM DUAL;  
END;  
  
*-- ---  
-- Foreign Keys  
-- ---*ALTER TABLE deanery\_info ADD FOREIGN KEY (user\_id) REFERENCES users (id);  
ALTER TABLE deanery\_info ADD FOREIGN KEY (faculty) REFERENCES faculty (faculty);  
ALTER TABLE teacher\_info ADD FOREIGN KEY (faculty) REFERENCES faculty (faculty);  
ALTER TABLE teacher\_info ADD FOREIGN KEY (user\_id) REFERENCES users (id);  
ALTER TABLE student\_info ADD FOREIGN KEY (user\_id) REFERENCES users (id);  
ALTER TABLE student\_info ADD FOREIGN KEY (specialization) REFERENCES specialization (specialization);  
ALTER TABLE student\_progress ADD FOREIGN KEY (user\_id) REFERENCES users (id);  
ALTER TABLE student\_progress ADD FOREIGN KEY (subject) REFERENCES subjects (subject);  
ALTER TABLE subjects ADD FOREIGN KEY (faculty) REFERENCES faculty (faculty);  
ALTER TABLE adjustments ADD FOREIGN KEY (user\_id) REFERENCES users (id);  
ALTER TABLE adjustments ADD FOREIGN KEY (subject) REFERENCES subjects (subject);  
ALTER TABLE adjustments ADD FOREIGN KEY (teacher\_id) REFERENCES users (id);  
ALTER TABLE retakes ADD FOREIGN KEY (user\_id) REFERENCES users (id);  
ALTER TABLE retakes ADD FOREIGN KEY (subject) REFERENCES subjects (subject);  
ALTER TABLE retakes ADD FOREIGN KEY (teacher\_id) REFERENCES users (id);  
ALTER TABLE gaps ADD FOREIGN KEY (user\_id) REFERENCES users (id);  
ALTER TABLE gaps ADD FOREIGN KEY (subject) REFERENCES subjects (subject);  
ALTER TABLE timetable ADD FOREIGN KEY (teacher\_id) REFERENCES users (id);  
ALTER TABLE faculty\_specialization add foreign key (specialization) references specialization(specialization);  
ALTER TABLE faculty\_specialization add foreign key (faculty) references faculty(faculty);

**Приложение В**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И. О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение В

Лит.

Листов

1

БГТУ

|  |  |  |
| --- | --- | --- |
| Название процедуры | Описание | Принимаемые параметры |
| findUser | Процедура поиска пользователя | In\_login in, in\_password in, user\_cur out |
| addUser | Процедура добавления пользователя | In\_login in, in\_password in |
| addEmptyStudent | Процедура добавления пустого студента | in\_id in |
| findStudent | Процедура поиска студента | In\_user\_id in, student out |
| updateStudent | Процедура обновления личной информации студента | @city\_name, @countryName, @distance |
| addAdjustment | Процедура добавления отработки | in\_user\_id in, in\_teacher\_name in, in\_subject in, in\_filing\_date in, in\_img in |
| addRetake | Процедура добавления пересдачи | in\_user\_id in, in\_teacher\_name in, in\_subject in, in\_retake\_date in, in\_img in |
| findDeanery | Процедура поиска сотрудника деканата | In\_user\_id in, deanery out |
| findTeacher | Процедура поиска преподавателя | In\_user\_id in, teacher out |
| setGaps | Процедура добавления информации об пропусках | In\_user\_id in, in\_subject in, in\_gaps\_date in, in\_gaps\_count in |
| accept\_decline\_Adjustment | Процедура для работы с отработками | In\_user\_id in, in\_subject in, in\_filing\_date in, action in |
| accept\_decline\_Retake | Процедура для работы с пересдачами | In\_user\_id in, in\_subject in, in\_retake\_date in, action in |
| setRating | Процедура добавления оценок | In\_student\_name in, in\_subject in, in\_note in, in\_progress\_date in, in\_comment in |
| accept\_decline\_Adjustment\_Teacher | Процедура для работы с отработками (для преподавателя) | In\_student\_name in, in\_subject in, in\_teacher\_id, in\_adjustment\_date in, in\_filing\_date in, action in |
| deleteAdjustment | Процедура удаления отработки | In\_student\_name, in\_subject in, in\_filing\_date in |
| generateUsers | Процедура добавления 100000 строк в юзеров | - |
| XML\_PACKAGE.EXPORT\_USERS  \_TO\_XML | Процедура экспорта пользователей в xml | - |
| XML\_PACKAGE.IMPORT\_USERS  \_FROM\_XML | Процедура импорта пользователей из xml в бд | - |

**Приложение Г**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И. О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

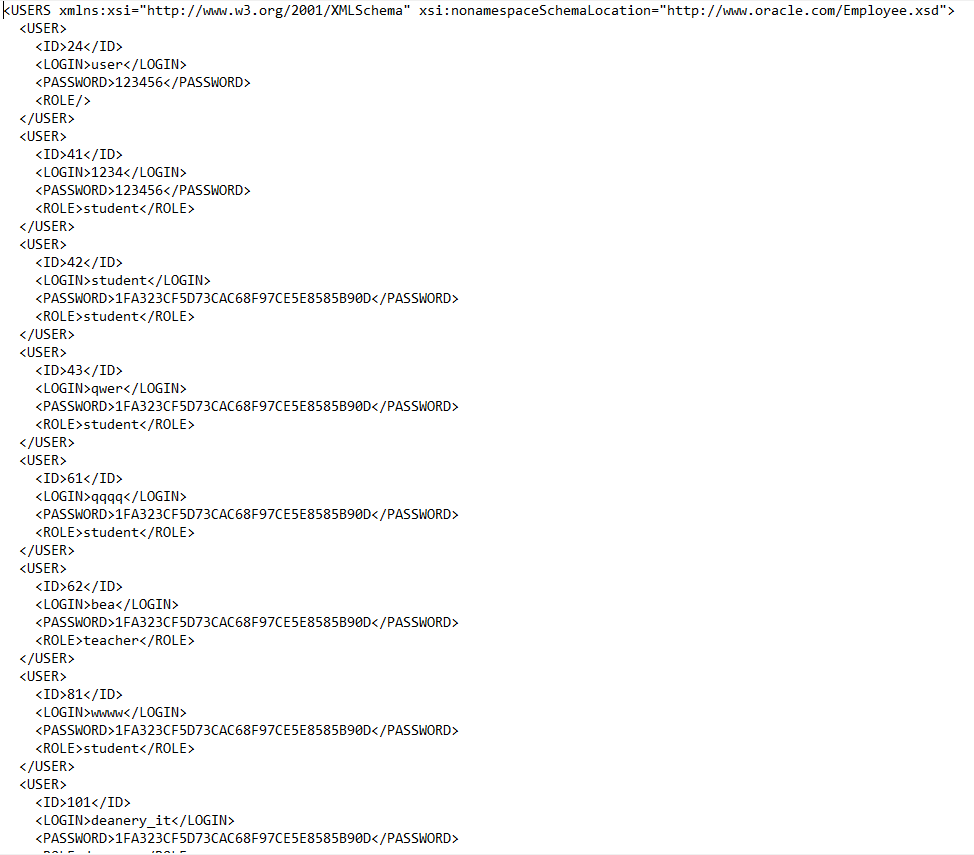
Приложение Г

Лит.

Листов

1

БГТУ



**Приложение Д**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Истомин И. О.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение Д

Лит.

Листов

1

БГТУ

CREATE OR REPLACE DIRECTORY UTLDATA AS 'A:/XML';  
DROP DIRECTORY UTLDATA;  
  
  
CREATE OR REPLACE PACKAGE XML\_PACKAGE IS  
 PROCEDURE *EXPORT\_USERS\_TO\_XML*;  
 PROCEDURE *IMPORT\_USERS\_FROM\_XML*;  
END XML\_PACKAGE;  
  
CREATE OR REPLACE PACKAGE BODY XML\_PACKAGE IS  
  
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*("ID",U.ID),  
 *XMLELEMENT*("LOGIN",U.LOGIN),  
 *XMLELEMENT*("PASSWORD",U.PASSWORD),  
 *XMLELEMENT*("ROLE",U.ROLE)  
 ))  
) FROM USERS U;  
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, 'UTLDATA/users.xml');  
END *EXPORT\_USERS\_TO\_XML*;  
  
PROCEDURE *IMPORT\_USERS\_FROM\_XML*IS  
 L\_CLOB CLOB;  
 L\_BFILE BFILE := *BFILENAME*('UTLDATA', '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%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;  
   
 P := DBMS\_XMLPARSER.NEWPARSER;  
  
 DBMS\_XMLPARSER.PARSECLOB(P,L\_CLOB);  
  
 V\_DOC := DBMS\_XMLPARSER.GETDOCUMENT(P);  
  
 V\_ROOT\_ELEMENT := DBMS\_XMLDOM.Getdocumentelement(v\_Doc);  
  
 V\_CHILD\_NODES := DBMS\_XMLDOM.GETCHILDRENBYTAGNAME(V\_ROOT\_ELEMENT,'\*');  
  
 FOR i IN 0 .. DBMS\_XMLDOM.GETLENGTH(V\_CHILD\_NODES) - 1  
 LOOP  
  
 V\_CURRENT\_NODE := DBMS\_XMLDOM.ITEM(V\_CHILD\_NODES,i);  
  
 DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE,  
 'ID/text()',U.ID);  
 DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE,  
 'LOGIN/text()',U.LOGIN);  
 DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE,  
 'PASSWORD/text()',U.PASSWORD);  
 DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE,  
 'ROLE/text()',U.ROLE);  
  
 INSERT INTO USERS(id, login, password, role)  
 VALUES(u.id,u.login,u.password,u.role) ;  
 END LOOP;  
  
 DBMS\_LOB.*FREETEMPORARY*(L\_CLOB);  
 DBMS\_XMLPARSER.FREEPARSER(P);  
 DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);  
 COMMIT;  
EXCEPTION  
 WHEN OTHERS THEN  
 DBMS\_LOB.*FREETEMPORARY*(L\_CLOB);  
 DBMS\_XMLPARSER.FREEPARSER(P);  
 DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);  
 *RAISE\_APPLICATION\_ERROR*(-20101, 'IMPORT XML ERROR'|| *SQLERRM*);  
END *IMPORT\_USERS\_FROM\_XML*;  
  
END XML\_PACKAGE;