БЕЛКООПСОЮЗ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОЛОДЕЧНЕНСКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ»

Информационно-справочная система «Дипломное проектирование»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Дисциплина: «База данных и системы управления базами данных»

КП.2400101.0428259.ПЗ-21

Руководитель О.Ч. Цыганок

Разработчик М.В. Буйновский

Молодечно 2021

СОДЕРЖАНИЕ

[Введение 4](#_Toc89886788)

[1 Анализ предметной области 6](#_Toc89886789)

[2 Постановка задачи 7](#_Toc89886790)

[3 Разработка информационного обеспечения задачи 9](#_Toc89886791)

[3.1 Информационный анализ входной информации необходимой для решения задачи и выделение информационных объектов 9](#_Toc89886792)

[3.2 Определение связей информационных объектов и построение информационной логической модели 11](#_Toc89886793)

[3.3 Определение логической структуры базы данных 11](#_Toc89886794)

[3.4 Разработка исходных данных контрольного примера и создание базы данных 12](#_Toc89886795)

[4 Описание алгоритмов решения задачи и их математических реализаций 29](#_Toc89886796)

[5 Разработка диалогов приложения пользователя 34](#_Toc89886797)

[6 Руководство пользователя 39](#_Toc89886798)

[Заключение 45](#_Toc89886799)

[Список использованных источников 46](#_Toc89886800)

[Приложение А 47](#_Toc89886801)

13, 35-44

# ВВЕДЕНИЕ

Дипломное проектирование является заключительным этапом обучения студентов в вузе и имеет своей целью закрепление и углубление теоретических и практических знаний по избранной специальности и применение их для решения конкретных задач.

Задачи дипломного проектирования:

* формирование навыков ведения самостоятельной проектно-конструкторской или исследовательской работы и овладение методикой проектирования или научного исследования и эксперимента;
* приобретение навыков обобщения и анализа результатов, полученных другими разработчиками или исследователями;
* выявление степени подготовленности студента к самостоятельной работе в системе образования или в условиях современного производства.

Дипломный проект является квалификационной работой выпускника. По уровню выполнения дипломного проекта и результатам его защиты Государственной экзаменационной комиссией (ГЭК) делается заключение о возможности присвоения выпускнику соответствующей квалификации. Настоящее методическое пособие содержит методические указания к выполнению дипломных проектов, которые разработаны в соответствии с Инструкцией по подготовке, оформлению и представлению к защите дипломных проектов в высших учебных заведениях, утвержденной приказом Министерства образования Республики Беларусь от 27.06.1997 г. № 356, а также Инструкцией по оформлению диссертации, автореферата и публикаций по теме диссертации, введенной в действие Постановлением президиума государственного высшего аттестационного комитета Республики Беларусь 24.12.1997 № 178 (в редакции постановления Высшей аттестационной комиссии Республики Беларусь 15.08.2007 г. № 4) с целью определения условий и сроков выполнения, правил оформления дипломных проектов. Среди видов периодических печатных изданий самый распространенный и популярный — это газеты. Это издание, которое выпускается на нескольких печатных листах через строго определенные короткие промежутки времени.

Основой современных информационных технологий являются данные. Практически все информационные системы в той или иной степени связаны с функциями долговременного хранения и обработки данных. Информация является фактором, определяющим эффективность любой сферы деятельности.

Данные, предназначенные для компьютерной обработки, целесообразно структурировать, т. е. организовывать определенным образом. Именно структурирование позволяет пользователю оптимизировать работу с данными (сократить время поиска, обеспечить эффективное хранение, исключить ошибки при обработке).

База данных – совокупность взаимосвязанных и организованных определенным образом данных.

Тема курсового проекта ­– Информационно-справочная система «Дипломное проектирование».

Целью данного проекта является создание информационно-справочной системы по дипломному проектированию. Система позволяет автоматизировать повседневные процессы в деятельности преподавателей, такие как:

* просмотр прогресса выполнения дипломного проекта учащимися:
* просмотр и подтверждение прогресса выполнения дипломных проектов преподавателями;
* создание отчетности о прогрессе выполнения дипломного проекта;
* присваивание тем дипломных проектов учащимся;
* изменение данных тем дипломных проектов учащихся.

Внедрение программного средства позволит удобно контролировать информацию о прогрессе выполнения дипломных проектов учащимися и легко присваивать темы дипломных проектов учащимся преподавателю. Позволит просматривать прогресс выполнения дипломного проекта учащемуся. Позволит администратору управлять данными о группах, преподавателях и учащихся.

Достоинством программного средства является современный, удобный и интуитивно понятный пользовательский интерфейс, также понятный и удобный ввод и вывод информации из базы данных.

1 Анализ предметной области

Дипломное проектирование является заключительным этапом обучения студентов в вузе и имеет своей целью закрепление и углубление теоретических и практических знаний по избранной специальности и применение их для решения конкретных задач. Каждый дипломный проект разделяется на этапы.

Тематика, классификация и содержание дипломных проектов

Тематика дипломных проектов разрабатывается выпускающими кафедрами ежегодно с учетом актуальности и практической значимости. Тема дипломного проекта является продолжением исследований, проводимых в процессе написания курсовых работ и прохождения преддипломной практики.

Дипломный проект выполняется на протяжении времени, отведенного для этого учебным планом специальности, и времени нахождения студента на преддипломной практике. Закрепление за студентами тем дипломных проектов и их руководителей ежегодно по представлению кафедры оформляется приказом ректора перед направлением студентов в соответствии с учебным планом на преддипломную практику.

Структура дипломного проекта

Дипломный проект состоит из расчетно-пояснительной записки и графической части.

Расчетно-пояснительная записка дипломного проекта технолого-методического направления должна содержать несколько глав. Это могут быть:

* введение;
* аналитический раздел;
* технологический раздел;
* конструкторский раздел (расчет и проектирование средств
* технологического оснащения);
* экономическая часть;
* анализ ресурсо- и энергосберегающих технологий технологического
* процесса обработки детали (сборки машины или механизма);
* охрана труда на производственном участке;
* экологический раздел;
* методическая часть;
* заключение;
* литература;
* приложения.

По согласованию с руководителем проекта могут быть включены и другие главы и/или разделы.

## 2 Постановка задачи

Для автоматизации отслеживания прогресса выполнения дипломных проектов нужно разработать информационно-справочную систему «Дипломное проектирование».

При разработке этого программного средства необходимо было выполнить следующие задачи:

База данных должна содержать информацию о дипломниках, руководителях, дипломных проектов и темах, которые они предлагают. Необходимо организовать закрепление тем за учащимися. Учесть, что каждый руководитель может иметь не более 8 дипломников. Не могут быть две одинаковые темы у разных учащихся. Также назначается руководитель по охране труда и экономическому разделу. Организовать выдачу задания на дипломное проектирование (техконтроль, нормоконтроль).

Для реализации этих задач требуется система управления базами данных, поддерживающая реляционную модель базы данных. Постановка задачи предусматривает разработку приложения «клиент–сервер», поэтому в качестве системы управления базами данных было выбрана среда PostgreSQL 14 и язык программирования C#.

Microsoft Visual Studio – линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. VisualStudio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (например, Subversion и VisualSourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода).

Главным преимуществом Visual Studio 2022 является удобство. Обеспечивает возможность создания разнообразных приложений на основе одного набора навыков.

C# – объектно-ориентированный язык программирования.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Основные достоинства С#:

– C# создавался параллельно с каркасом Framework .Net и в полной мере учитывает все его возможности — как FCL, так и CLR;

– C# является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;

– C# является мощным объектным языком с возможностями наследования и универсализации;

– C# является наследником языков C/C++, сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от С++ к C#;

– сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются такие опасные свойства С++ как указатели, адресация, разыменование, адресная арифметика;

– благодаря каркасу Framework .Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;

– мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;

– реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.

3 Разработка информационного обеспечения задачи

3.1 Информационный анализ входной информации необходимой для решения задачи и выделение информационных объектов

Входная информация для разрабатываемой программы определяется задачей разработки информационно-справочной системы «Дипломное проектирование».

Сведения о преподавателях содержат следующую информацию:

* тип преподавателя;
* Ф.И.О. преподавателя;
* адрес;
* телефонный номер.

Сведения о типах преподавателя содержат названия типов преподавателя.

Сведения о группах содержат следующую информацию:

* куратор группы;
* наименование группы;
* год формирования группы.

Сведения о студентах содержат следующую информацию:

* группа;
* Ф.И.О. студента;
* телефонный номер.

Сведения о пользователях содержат следующую информацию:

* код учителя;
* код пользователя;
* пароль пользователя.

Сведения о темах дипломных проектах следующую информацию:

* студент, которому присвоен дипломный проект;
* тема дипломного проекта;
* преподаватель по основному разделу;
* преподаватель по экономическому разделу;
* преподаватель по разделу охраны труда.

Сведения о преподавателях содержат следующую информацию:

* наименование этапа;
* дипломный проект, к которому присвоен данный этап;
* преподаватель, проверяющий данный этап;
* процент выполнения этапа;
* дата начала этапа;
* дата окончания этапа.

Сведения о названиях этапов содержат названия этапов.

В предметной области имеется множество разнообразных объектов. Обычно под объектом понимают некую сущность (реальную или абстрактную), о которой собирается какая-то информация. Информационный объект *–* это описание некоторой сущности (реального объекта, явления, процесса, события) в виде совокупности логически связанных реквизитов (информационных элементов). Процесс построения информационной модели состоит из следующих шагов:

* определение сущностей;
* определение зависимостей между сущностями;
* задание первичных и альтернативных ключей;
* определение атрибутов сущностей;
* приведение модели к требуемому уровню нормальной формы;
* переход к физическому описанию модели: назначение соответствий имя сущности - имя таблицы, атрибут сущности - атрибут таблицы; задание триггеров, процедур и ограничений;
* генерация базы данных.

В результате анализа предметной области было выделено 8 информационных объектов, их реквизиты приведены в таблице 3.1.1.

Таблица 3.1.1 - Информационные объекты и их реквизиты

|  |  |
| --- | --- |
| Информационные объекты | Реквизиты |
| Типы преподавателей | Код типа – первичный ключ  Наименование типа |
| Преподаватели | Код преподавателя – первичный ключ  Код типа преподавателя – внешний ключ  Ф.И.О. преподавателя  Адрес  Телефонный номер |
| Группы | Код группы – первичный ключ  Код преподавателя – внешний ключ  Наименование группы  Год формирования группы |
| Студенты | Код студента – первичный ключ  Код группы – внешний ключ  Ф.И.О. студента  Телефонный номер |
| Пользователи | Код пользователя – первичный ключ  Код преподавателя – внешний ключ  Код студента – внешний ключ  Хэш пароля пользователя |
| Дипломные проекты | Код дипломного проекта – первичный ключ  Код студента – внешний ключ  Тема дипломного проекта  Код учителя по основному разделу – внешний ключ  Код учителя по экономическому разделу – внешний ключ  Код учителя по разделу охраны труда – внешний ключ |
| Наименования этапов | Код наименования этапа – первичный ключ  Наименование этапа |
| Этапы дипломного проекта | Код этапа – первичный ключ  Код наименования этапа – внешний ключ  Код дипломного проекта – внешний ключ  Код преподавателя – внешний ключ  Процент выполнения этапа  Дата начала этапа  Дата окончания этапа |

### 3.2 Определение связей информационных объектов и построение информационной логической модели

Связи между выявленными информационными объектами определяются реаль­ными отношениями между парами объектов. Связи информационных объектов представлены в виде таблицы 3.2.1.

Таблица 3.2.1 – Связи информационных объектов

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ связи | Главный информационный объект | Подчиненный информационный объект | Тип отношения |
| Код типа преподавателя | Преподаватели | Типы преподавателей | 1:М |
| Код преподавателя | Группы | Преподаватели | 1:М |
| Код группы | Студенты | Группы | 1:М |
| Код преподавателя | Пользователи | Преподаватели | 1:М |
| Код студента | Пользователи | Студенты | 1:М |
| Код студента | Дипломные проекты | Студенты | 1:М |
| Код преподавателя | Дипломные проекты | Преподаватели | 1:М |
| Код наименования этапа | Этапы дипломного проекта | Наименования этапов | 1:М |
| Код дипломного проекта | Этапы дипломного проекта | Дипломные проекты | 1:М |
| Код преподавателя | Этапы дипломного проекта | Преподаватели | 1:М |

### 3.3 Определение логической структуры базы данных

Модель данных — это концептуальное представление для выражения и передачи бизнес-требований. Она наглядно показывает характер данных, бизнес-правила, управляющие данными, и то, как данные будут организованы в базе данных.

При разработке базы данных обычно выделяется несколько уровней моделирования, при помощи которых происходит переход от предметной области к конкретной реализации базы данных средствами конкретной «Система управления базами данных». Можно выделить следующие уровни: сама предметная область, концептуальная модель предметной области, логическая модель данных, физическая модель данных, собственно база данных и приложения.

Логическая структура реляционной базы данных определяется совокупностью логически взаимосвязанных реляционных таблиц. Каждая реляционная таблица имеет структуру, определяемую реквизитным составом одного из информационных объектов. Логические связи таблиц соответствуют структурным связям между объектами.

Физическая модель представляет собой список всех сущностей, атрибутов и ключевых полей, только в таком виде, в каком они находятся в самой базе данных. т. е. отображаются наименования полей, их типы и параметры, а также связи между сущностями.

Одним из инструментов функционального моделирования является CASE-средства верхнего уровня Entity Relationship Diagram (ERD). В ERD существуют два уровня представления и моделирования - логический и физический. Логический уровень означает прямое отображение фактов из реальной жизни. Например, люди, столы, отделы, собаки и компьютеры являются реальными объектами. Они именуются на естественном языке, с любыми разделителями слов (пробелы, запятые и т.д.). На логическом уровне не рассматривается использование конкретной СУБД, не определяются типы данных (например, целое или вещественное число) и не определяются индексы для таблиц. Целевая СУБД, имена объектов и типы данных, индексы составляют второй (физический) уровень модели ERD.

Логическая модель базы данных, созданная с помощью ERD представлена на рисунке 3.3.1.

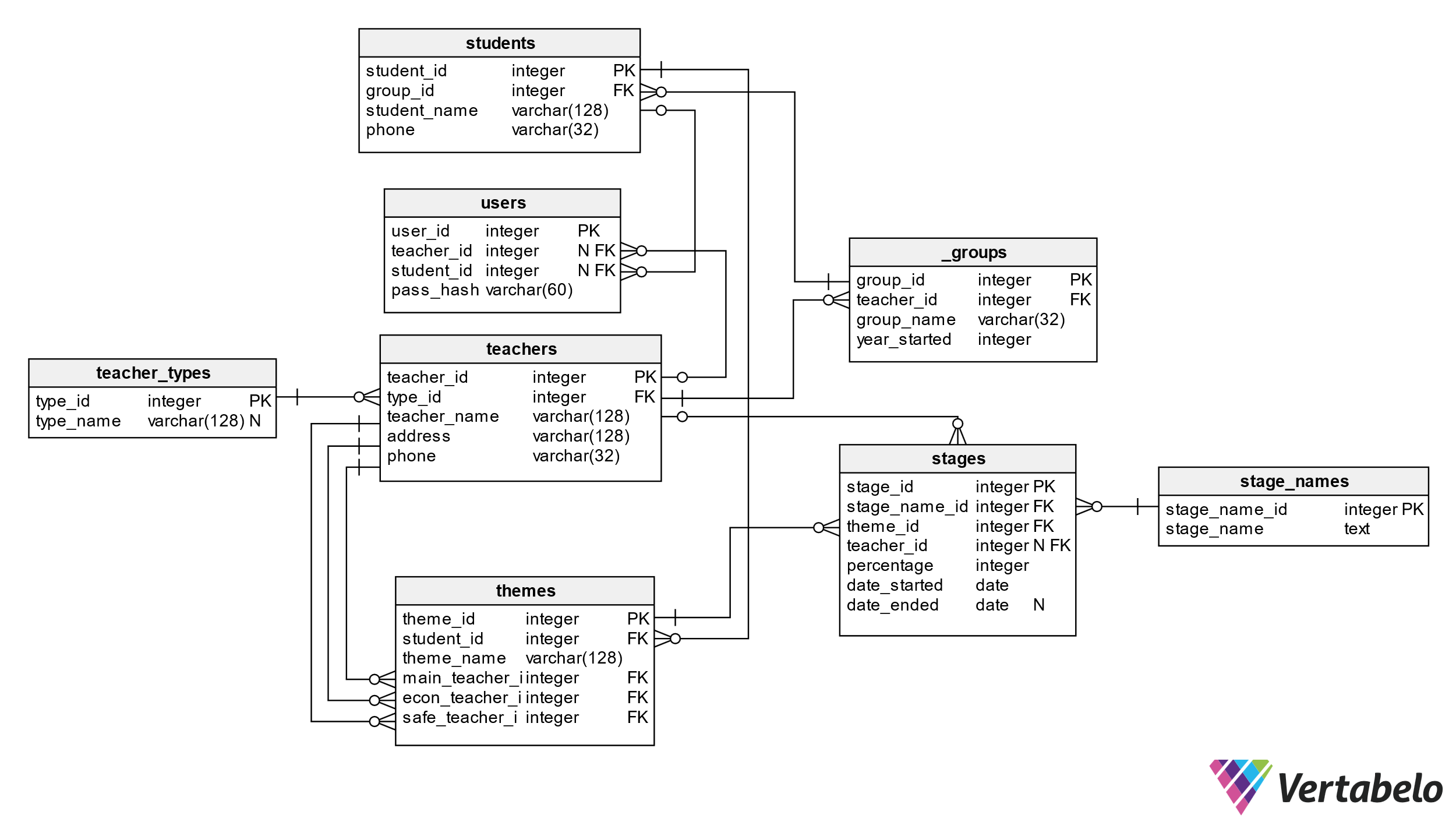


Рисунок 3.3.1 – Логическая модель базы данных

### 3.4 Разработка исходных данных контрольного примера и создание базы данных

Требования к данным контрольного примера - их представительность, учитыва­ющая особенности информации, указанные в описании предметной области. Такие данные должны обеспечить отладку алгоритма на компьютере и подтвердить ра­ботоспособность реализации алгоритма. В данных контрольного примера для рассматриваемой задачи должно быть предусмотрено, что в учебном плане содержится несколько предметов, на один предмет могут быть установлены два преподавателя. Данные контрольного примера, предназначенные для тестиро­вания, отладки и демонстрации программного средства, приведены на рисунках 3.4.1.

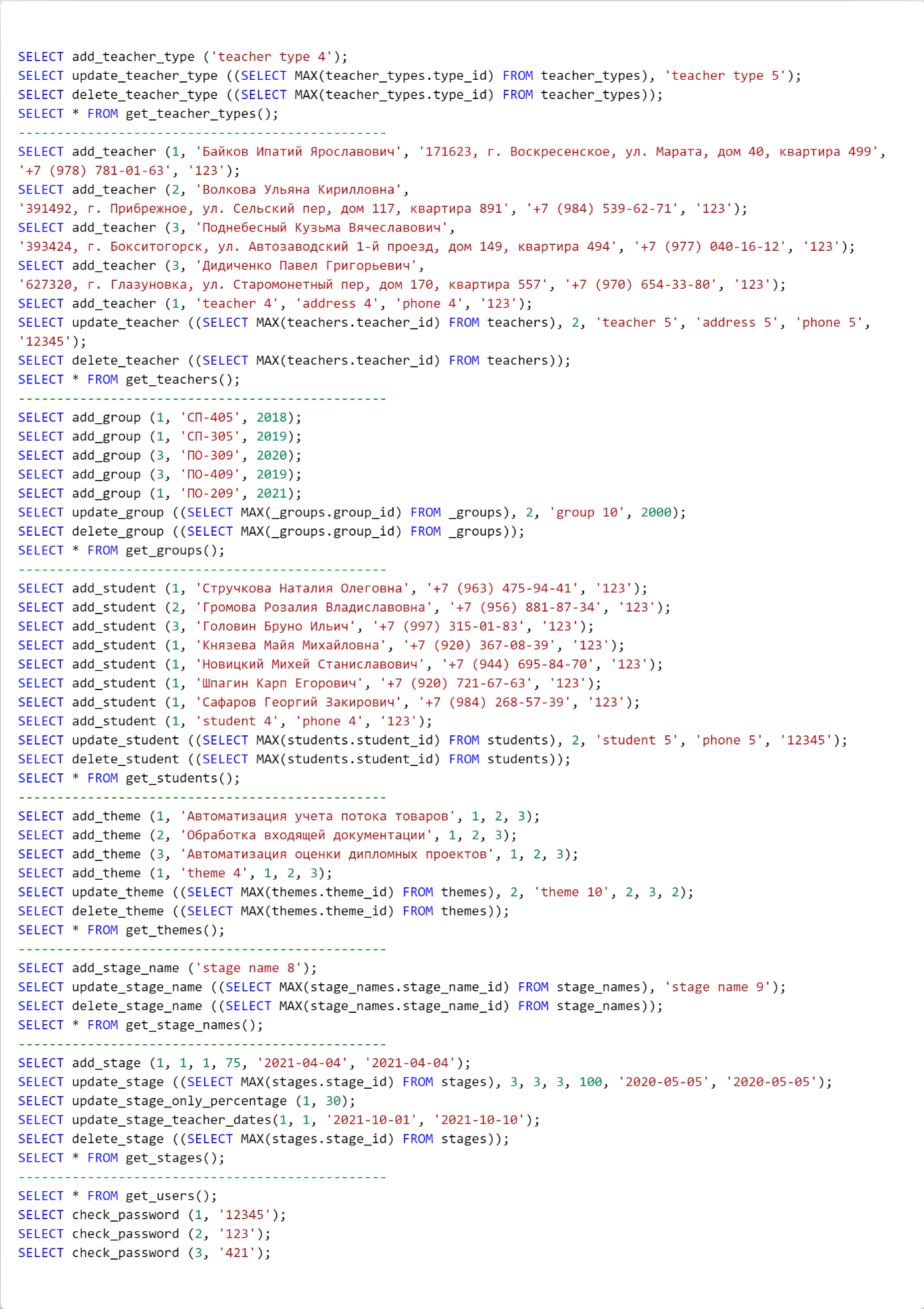


Рисунок 3.4.1 – Данные контрольного примера

В рамках данного курсового проекта для решения поставленной задачи была создана база данных «DiplomaProjects». Так как при разработке приложения используется клиент-серверная архитектура, то для создания базы данных было выбрано СУБД PostgreSQL. Ниже, в таблицах 3.4.1 – 3.4.8 представлены таблицы базы данных с типами данных столбцов и предлагаемыми ограничения­ми целостности.

Таблица 3.4.1 – Структура таблицы «teacher\_types»

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Значение поля | Название поля в БД | Тип поля в БД |
| Код типа | Числовой, первичный ключ | type\_id | integer PK |
| Наименование типа | Строковый | type\_name | varchar(128) |

Таблица 3.4.2 – Структура таблицы «teachers»

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Значение поля | Название поля в БД | Тип поля в БД |
| Код преподавателя | Числовой, первичный ключ | teacher\_id | integer PK |
| Код типа преподавателя | Числовой, внешний ключ | type\_id | integer FK |
| Ф.И.О. преподавателя | Строковый | teacher\_name | varchar(128) |
| Адрес | Строковый | address | varchar(128) |
| Телефонный номер | Строковый | phone | varchar(32) |

Таблица 3.4.3 – Структура таблицы «\_groups»

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Значение поля | Название поля в БД | Тип поля в БД |
| Код группы | Числовой, первичный ключ | group\_id | integer PK |
| Код преподавателя | Числовой, внешний ключ | teacher\_id | integer FK |
| Наименование группы | Строковый | group\_name | varchar(32) |
| Год формирования группы | Числовой | year\_started | integer |

Таблица 3.4.4 – Структура таблицы «students»

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Значение поля | Название поля в БД | Тип поля в БД |
| Код студента | Числовой, первичный ключ | student\_id | integer PK |
| Код группы | Числовой, внешний ключ | group\_id | integer FK |
| Ф.И.О. студента | Строковый | student\_name | varchar(128) |
| Телефонный номер | Строковый | phone | varchar(32) |

Таблица 3.4.5 – Структура таблицы «users»

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Значение поля | Название поля в БД | Тип поля в БД |
| Код пользователя | Числовой, первичный ключ | user\_id | integer PK |
| Код преподавателя | Числовой, внешний ключ | teacher\_id | integer FK |
| Код студента | Числовой, внешний ключ | student\_id | integer FK |
| Хэш пароля пользователя | Строковый | pass\_hash | varchar(60) |

Таблица 3.4.6 – Структура таблицы «themes»

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Значение поля | Название поля в БД | Тип поля в БД |
| Код дипломного проекта | Числовой, первичный ключ | theme\_id | integer PK |
| Код студента | Числовой, внешний ключ | student\_id | integer FK |
| Тема дипломного проекта | Строковый | theme\_name | varchar(128) |
| Код учителя по основному разделу | Числовой, внешний ключ | main\_teacher\_id | integer FK |
| Код учителя по экономическому разделу | Числовой, внешний ключ | econ\_teacher\_id | integer FK |
| Код учителя по разделу охраны труда | Числовой, внешний ключ | safe\_teacher\_id | integer FK |

Таблица 3.4.7 – Структура таблицы «stage\_names»

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Значение поля | Название поля в БД | Тип поля в БД |
| Код наименования этапа | Числовой, первичный ключ | stage\_name\_id | integer PK |
| Наименование этапа | Текстовый | stage\_name | text |

Таблица 3.4.8 – Структура таблицы «stages»

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Значение поля | Название поля в БД | Тип поля в БД |
| Код этапа | Числовой, первичный ключ | stage\_id | integer PK |
| Код наименования этапа | Числовой, внешний ключ | stage\_name\_id | integer FK |
| Код дипломного проекта | Числовой, внешний ключ | theme\_id | integer FK |
| Код преподавателя | Числовой, внешний ключ | teacher\_id | integer FK |
| Процент выполнения этапа | Числовой | percentage | integer |
| Дата начала этапа | Дата | date\_started | date |
| Дата окончания этапа | Дата | date\_ended | date |

Описание схемы БД на языке SQL приведено ниже:

CREATE EXTENSION IF NOT EXISTS pgcrypto;

DROP TABLE IF EXISTS stages, stage\_names, themes, students, \_groups, teachers, teacher\_types, users;

CREATE TABLE teacher\_types (

type\_id integer GENERATED ALWAYS AS IDENTITY NOT NULL,

type\_name varchar(128)

);

ALTER TABLE teacher\_types

ADD CONSTRAINT pk\_teacher\_types PRIMARY KEY (type\_id);

CREATE TABLE teachers (

teacher\_id integer GENERATED ALWAYS AS IDENTITY NOT NULL,

type\_id integer NOT NULL,

teacher\_name varchar(128) NOT NULL,

address varchar(128) NOT NULL,

phone varchar(32) NOT NULL

);

ALTER TABLE teachers

ADD CONSTRAINT pk\_teachers PRIMARY KEY (teacher\_id),

ADD CONSTRAINT fk\_teachers\_teacher\_types\_type\_id FOREIGN KEY (type\_id) REFERENCES teacher\_types;

CREATE TABLE \_groups (

group\_id integer GENERATED ALWAYS AS IDENTITY NOT NULL,

teacher\_id integer NOT NULL,

group\_name varchar(32) NOT NULL,

year\_started integer NOT NULL

);

ALTER TABLE \_groups

ADD CONSTRAINT pk\_groups PRIMARY KEY (group\_id),

ADD CONSTRAINT fk\_groups\_teachers\_teacher\_id FOREIGN KEY (teacher\_id) REFERENCES teachers;

CREATE TABLE students (

student\_id integer GENERATED ALWAYS AS IDENTITY NOT NULL,

group\_id integer NOT NULL,

student\_name varchar(128) NOT NULL,

phone varchar(32) NOT NULL

);

ALTER TABLE students

ADD CONSTRAINT pk\_students PRIMARY KEY (student\_id),

ADD CONSTRAINT fk\_students\_groups\_group\_id FOREIGN KEY (group\_id) REFERENCES \_groups;

CREATE TABLE users (

user\_id integer GENERATED ALWAYS AS IDENTITY NOT NULL,

teacher\_id integer,

student\_id integer,

pass\_hash varchar(60) NOT NULL

);

ALTER TABLE users

ADD CONSTRAINT pk\_users PRIMARY KEY (user\_id),

ADD CONSTRAINT fk\_users\_teachers\_teacher\_id FOREIGN KEY (teacher\_id) REFERENCES teachers,

ADD CONSTRAINT fk\_users\_students\_student\_id FOREIGN KEY (student\_id) REFERENCES students,

ADD CONSTRAINT valid\_user CHECK ((student\_id IS NULL) != (teacher\_id IS NULL)),

ADD CONSTRAINT u\_teacher\_id UNIQUE (teacher\_id),

ADD CONSTRAINT u\_student\_Id UNIQUE (student\_id);

CREATE TABLE themes (

theme\_id integer GENERATED ALWAYS AS IDENTITY NOT NULL,

student\_id integer NOT NULL,

theme\_name varchar(128) NOT NULL,

main\_teacher\_id integer NOT NULL,

econ\_teacher\_id integer NOT NULL,

safe\_teacher\_id integer NOT NULL

);

ALTER TABLE themes

ADD CONSTRAINT pk\_themes PRIMARY KEY (theme\_id),

ADD CONSTRAINT fk\_themes\_students\_student\_id FOREIGN KEY (student\_id) REFERENCES students,

ADD CONSTRAINT fk\_themes\_teachers\_main\_teacher\_id FOREIGN KEY (main\_teacher\_id) REFERENCES teachers (teacher\_id),

ADD CONSTRAINT fk\_themes\_teachers\_econ\_teacher\_id FOREIGN KEY (econ\_teacher\_id) REFERENCES teachers (teacher\_id),

ADD CONSTRAINT fk\_themes\_teachers\_safe\_teacher\_id FOREIGN KEY (safe\_teacher\_id) REFERENCES teachers (teacher\_id),

ADD CONSTRAINT u\_theme\_name UNIQUE (theme\_name);

CREATE TABLE stage\_names (

stage\_name\_id integer GENERATED ALWAYS AS IDENTITY NOT NULL,

stage\_name text NOT NULL

);

ALTER TABLE stage\_names

ADD CONSTRAINT pk\_stage\_names PRIMARY KEY (stage\_name\_id);

CREATE TABLE stages (

stage\_id integer GENERATED ALWAYS AS IDENTITY NOT NULL,

stage\_name\_id integer NOT NULL,

theme\_id integer NOT NULL,

teacher\_id integer,

percentage integer NOT NULL DEFAULT (0),

date\_started date NOT NULL DEFAULT(CURRENT\_DATE),

date\_ended date

);

ALTER TABLE stages

ADD CONSTRAINT pk\_stages PRIMARY KEY (stage\_id),

ADD CONSTRAINT fk\_stages\_stage\_names\_stage\_name\_id FOREIGN KEY (stage\_name\_id) REFERENCES stage\_names,

ADD CONSTRAINT fk\_stages\_themes\_theme\_id FOREIGN KEY (theme\_id) REFERENCES themes,

ADD CONSTRAINT fk\_stages\_teachers\_teacher\_id FOREIGN KEY (teacher\_id) REFERENCES teachers;

DROP FUNCTION IF EXISTS

add\_teacher\_type, update\_teacher\_type, delete\_teacher\_type,

add\_teacher, update\_teacher, delete\_teacher,

add\_group, update\_group, delete\_group,

add\_student, update\_student, delete\_student,

add\_theme, update\_theme, delete\_theme,

add\_stage\_name, update\_stage\_name, delete\_stage\_name,

add\_stage, update\_stage, update\_stage\_only\_percentage, update\_stage\_teacher\_dates, delete\_stage,

get\_teacher\_types, get\_teachers, get\_groups, get\_students, get\_users, get\_themes, get\_stage\_names, get\_stages,

check\_password;

CREATE FUNCTION add\_teacher\_type (\_type\_name varchar(128))

RETURNS void

AS $$

BEGIN

INSERT INTO teacher\_types (type\_name)

VALUES (\_type\_name);

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION update\_teacher\_type (\_type\_id integer, \_type\_name varchar(128))

RETURNS void

AS $$

BEGIN

UPDATE teacher\_types

SET type\_name = \_type\_name

WHERE type\_id = \_type\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION delete\_teacher\_type (\_type\_id integer)

RETURNS void

AS $$

BEGIN

DELETE FROM teacher\_types

WHERE type\_id = \_type\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE OR REPLACE FUNCTION add\_teacher (

\_type\_id integer,

\_teacher\_name varchar(128),

\_address varchar(128),

\_phone varchar(32),

\_pass varchar(128)

)

RETURNS void

AS $$

DECLARE

new\_teacher\_id integer;

BEGIN

INSERT INTO teachers (type\_id, teacher\_name, address, phone)

VALUES (\_type\_id, \_teacher\_name, \_address, \_phone)

RETURNING teacher\_id INTO new\_teacher\_id;

INSERT INTO users (teacher\_id, pass\_hash)

VALUES (new\_teacher\_id, crypt(\_pass, gen\_salt('bf', 8)));

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION update\_teacher (

\_teacher\_id integer,

\_type\_id integer,

\_teacher\_name varchar(128),

\_address varchar(128),

\_phone varchar(32),

\_pass varchar(128)

)

RETURNS void

AS $$

DECLARE

old\_pass varchar(60);

BEGIN

old\_pass = (SELECT pass\_hash FROM users WHERE teacher\_id = \_teacher\_id);

IF crypt(\_pass, old\_pass) != old\_pass THEN

UPDATE users

SET pass\_hash = crypt(\_pass, gen\_salt('bf', 8))

WHERE teacher\_id = \_teacher\_id;

END IF;

UPDATE teachers

SET

type\_id = \_type\_id, teacher\_name = \_teacher\_name,

address = \_address, phone = \_phone

WHERE teacher\_id = \_teacher\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION delete\_teacher (\_teacher\_id integer)

RETURNS void

AS $$

BEGIN

DELETE FROM users

WHERE teacher\_id = \_teacher\_id;

DELETE FROM teachers

WHERE teacher\_id = \_teacher\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION add\_group (

\_teacher\_id integer,

\_group\_name varchar(32),

\_year\_started integer

)

RETURNS void

AS $$

BEGIN

INSERT INTO \_groups (teacher\_id, group\_name, year\_started)

VALUES (\_teacher\_id, \_group\_name, \_year\_started);

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION update\_group (

\_group\_id integer,

\_teacher\_id integer,

\_group\_name varchar(32),

\_year\_started integer

)

RETURNS void

AS $$

BEGIN

UPDATE \_groups

SET

teacher\_id = \_teacher\_id, group\_name = \_group\_name,

year\_started = \_year\_started

WHERE group\_id = \_group\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION delete\_group (\_group\_id integer)

RETURNS void

AS $$

BEGIN

DELETE FROM \_groups

WHERE group\_id = \_group\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION add\_student (

\_group\_id integer,

\_student\_name varchar(128),

\_phone varchar(32),

\_pass varchar(128)

)

RETURNS void

AS $$

DECLARE

new\_student\_id integer;

BEGIN

INSERT INTO students (group\_id, student\_name, phone)

VALUES (\_group\_id, \_student\_name, \_phone)

RETURNING student\_id INTO new\_student\_id;

INSERT INTO users (student\_id, pass\_hash)

VALUES (new\_student\_id, crypt(\_pass, gen\_salt('bf', 8)));

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION update\_student (

\_student\_id integer,

\_group\_id integer,

\_student\_name varchar(128),

\_phone varchar(32),

\_pass varchar(128)

)

RETURNS void

AS $$

DECLARE

old\_pass varchar(60);

BEGIN

old\_pass = (SELECT pass\_hash FROM users WHERE student\_id = \_student\_id);

IF crypt(\_pass, old\_pass) != old\_pass THEN

UPDATE users

SET pass\_hash = crypt(\_pass, gen\_salt('bf', 8))

WHERE student\_id = \_student\_id;

END IF;

UPDATE students

SET

group\_id = \_group\_id, student\_name = \_student\_name, phone = \_phone

WHERE student\_id = \_student\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION delete\_student (\_student\_id integer)

RETURNS void

AS $$

BEGIN

DELETE FROM users

WHERE student\_id = \_student\_id;

DELETE FROM students

WHERE student\_id = \_student\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION add\_theme (

\_student\_id integer,

\_theme\_name varchar(128),

\_main\_teacher\_id integer,

\_econ\_teacher\_id integer,

\_safe\_teacher\_id integer

)

RETURNS integer

AS $$

DECLARE

new\_theme\_id integer;

\_stage\_names record;

BEGIN

-- CODES:

-- 0 - success

-- -1 - teacher already has 8 themes

IF (SELECT COUNT(\*) FROM themes WHERE main\_teacher\_id = \_main\_teacher\_id) >= 8 THEN

RETURN -1;

END IF;

INSERT INTO themes (student\_id, theme\_name, main\_teacher\_id, econ\_teacher\_id, safe\_teacher\_id)

VALUES (\_student\_id, \_theme\_name, \_main\_teacher\_id, \_econ\_teacher\_id, \_safe\_teacher\_id)

RETURNING theme\_id INTO new\_theme\_id;

FOR \_stage\_names IN

SELECT stage\_names.stage\_name\_id

FROM stage\_names

LOOP

EXECUTE add\_stage(\_stage\_names.stage\_name\_id, new\_theme\_id, NULL, 0, CURRENT\_DATE, CURRENT\_DATE);

END LOOP;

RETURN new\_theme\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION update\_theme (

\_theme\_id integer,

\_student\_id integer,

\_theme\_name varchar(128),

\_main\_teacher\_id integer,

\_econ\_teacher\_id integer,

\_safe\_teacher\_id integer

)

RETURNS integer

AS $$

BEGIN

-- CODES:

-- 0 - success

-- -1 - teacher already has 8 themes

IF (SELECT COUNT(\*) FROM themes WHERE main\_teacher\_id = \_main\_teacher\_id) >= 8 THEN

RETURN -1;

END IF;

UPDATE themes

SET

student\_id = \_student\_id, theme\_name = \_theme\_name,

main\_teacher\_id = \_main\_teacher\_id,

econ\_teacher\_id = \_econ\_teacher\_id,

safe\_teacher\_id = \_safe\_teacher\_id

WHERE theme\_id = \_theme\_id;

RETURN 0;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION delete\_theme (\_theme\_id integer)

RETURNS void

AS $$

DECLARE

\_stages record;

BEGIN

FOR \_stages IN

SELECT stages.stage\_id

FROM stages

WHERE stages.theme\_id = \_theme\_id

LOOP

EXECUTE delete\_stage(\_stages.stage\_id);

END LOOP;

DELETE FROM themes

WHERE theme\_id = \_theme\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION add\_stage\_name(\_stage\_name text)

RETURNS void

AS $$

BEGIN

INSERT INTO stage\_names (stage\_name)

VALUES (\_stage\_name);

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION update\_stage\_name (\_stage\_name\_id integer, \_stage\_name text)

RETURNS void

AS $$

BEGIN

UPDATE stage\_names

SET stage\_name = \_stage\_name

WHERE stage\_name\_id = \_stage\_name\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION delete\_stage\_name (\_stage\_name\_id integer)

RETURNS void

AS $$

BEGIN

DELETE FROM stage\_names

WHERE stage\_name\_id = \_stage\_name\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION add\_stage (

\_stage\_name\_id integer,

\_theme\_id integer,

\_teacher\_id integer,

\_percentage integer,

\_date\_started date,

\_date\_ended date

)

RETURNS void

AS $$

BEGIN

INSERT INTO stages (stage\_name\_id, theme\_id, teacher\_id, percentage, date\_started, date\_ended)

VALUES (\_stage\_name\_id, \_theme\_id, \_teacher\_id, \_percentage, \_date\_started, \_date\_ended);

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION update\_stage (

\_stage\_id integer,

\_stage\_name\_id integer,

\_theme\_id integer,

\_teacher\_id integer,

\_percentage integer,

\_date\_started date,

\_date\_ended date

)

RETURNS void

AS $$

BEGIN

UPDATE stages

SET

stage\_name\_id = \_stage\_name\_id, theme\_id = \_theme\_id,

teacher\_id = \_teacher\_id, percentage = \_percentage,

date\_started = \_date\_started, date\_ended = \_date\_ended

WHERE stage\_id = \_stage\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION update\_stage\_teacher\_dates (

\_stage\_id integer,

\_teacher\_id integer,

\_date\_started date,

\_date\_ended date

)

RETURNS void

AS $$

BEGIN

UPDATE stages

SET teacher\_id = \_teacher\_id, date\_started = \_date\_started, date\_ended = \_date\_ended

WHERE stage\_id = \_stage\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION update\_stage\_only\_percentage (\_stage\_id integer, \_percentage integer)

RETURNS void

AS $$

BEGIN

UPDATE stages

SET percentage = \_percentage

WHERE stage\_id = \_stage\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION delete\_stage (\_stage\_id integer)

RETURNS void

AS $$

BEGIN

DELETE FROM stages

WHERE stage\_id = \_stage\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION get\_teacher\_types ()

RETURNS TABLE (

type\_id integer,

type\_name varchar

)

AS $$

BEGIN

RETURN QUERY

SELECT teacher\_types.type\_id, teacher\_types.type\_name

FROM teacher\_types;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION get\_teachers ()

RETURNS TABLE (

teacher\_id integer,

type\_id integer,

type\_name varchar,

teacher\_name varchar,

address varchar,

phone varchar

)

AS $$

BEGIN

RETURN QUERY

SELECT

teachers.teacher\_id, teachers.type\_id, teacher\_types.type\_name,

teachers.teacher\_name, teachers.address, teachers.phone

FROM teachers

JOIN teacher\_types USING (type\_id);

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION get\_groups ()

RETURNS TABLE (

group\_id integer,

teacher\_id integer,

teacher\_name varchar,

group\_name varchar,

year\_started integer

)

AS $$

BEGIN

RETURN QUERY

SELECT

\_groups.group\_id, \_groups.teacher\_id, teachers.teacher\_name,

\_groups.group\_name, \_groups.year\_started

FROM \_groups

JOIN teachers USING (teacher\_id);

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION get\_students ()

RETURNS TABLE (

student\_id integer,

group\_id integer,

group\_name varchar,

student\_name varchar,

phone varchar,

theme\_id integer,

theme\_name varchar,

percentage integer

)

AS $$

BEGIN

RETURN QUERY

SELECT

students.student\_id, students.group\_id, \_groups.group\_name,

students.student\_name, students.phone, COALESCE(themes.theme\_id, 0) AS theme\_name,

COALESCE(themes.theme\_name, 'Отсутствует') AS theme\_name,

COALESCE((

SELECT AVG(CASE WHEN stages.percentage > 100 THEN 100 ELSE stages.percentage END)::int AS percentage

FROM stages

WHERE stages.theme\_id = themes.theme\_id

), 0)

FROM students

JOIN \_groups USING (group\_id)

LEFT JOIN themes USING (student\_id);

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION get\_users ()

RETURNS TABLE (

user\_id integer,

is\_teacher boolean,

id integer,

name varchar

)

AS $$

BEGIN

RETURN QUERY

SELECT

users.user\_id, users.teacher\_id IS NOT NULL AS is\_teacher,

COALESCE(users.teacher\_id, users.student\_id) AS id,

COALESCE(teachers.teacher\_name, students.student\_name) AS name

FROM users

LEFT JOIN teachers USING (teacher\_id)

LEFT JOIN students USING (student\_id);

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION get\_themes ()

RETURNS TABLE (

theme\_id integer,

student\_id integer,

student\_name varchar,

theme\_name varchar,

main\_teacher\_id integer,

econ\_teacher\_id integer,

safe\_teacher\_id integer

)

AS $$

BEGIN

RETURN QUERY

SELECT

themes.theme\_id, themes.student\_id, students.student\_name, themes.theme\_name,

themes.main\_teacher\_id, themes.econ\_teacher\_id, themes.safe\_teacher\_id

FROM themes

JOIN students USING (student\_id);

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION get\_stage\_names ()

RETURNS TABLE (

stage\_name\_id integer,

stage\_name text

)

AS $$

BEGIN

RETURN QUERY

SELECT stage\_names.stage\_name\_id, stage\_names.stage\_name

FROM stage\_names;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION get\_stages ()

RETURNS TABLE (

stage\_id integer,

stage\_name\_id integer,

stage\_name text,

theme\_id integer,

teacher\_id integer,

teacher\_name varchar,

percentage integer,

date\_started date,

date\_ended date

)

AS $$

BEGIN

RETURN QUERY

SELECT

stages.stage\_id, stages.stage\_name\_id, stage\_names.stage\_name,

stages.theme\_id, COALESCE(stages.teacher\_id, 0), COALESCE(teachers.teacher\_name, ''),

stages.percentage, stages.date\_started, stages.date\_ended

FROM stages

JOIN stage\_names USING (stage\_name\_id)

LEFT JOIN teachers USING (teacher\_id)

ORDER BY theme\_id, stage\_name\_id;

END;

$$ LANGUAGE 'plpgsql';

CREATE FUNCTION check\_password (\_user\_id integer, \_pass varchar(128))

RETURNS integer

AS $$

DECLARE

\_pass\_hash varchar(60);

BEGIN

-- CODES:

-- 0 - success

-- 1 - fail

\_pass\_hash = (SELECT pass\_hash FROM users WHERE users.user\_id = \_user\_id);

IF crypt(\_pass, \_pass\_hash) = \_pass\_hash THEN

RETURN 0;

ELSE

RETURN 1;

END IF;

END;

$$ LANGUAGE 'plpgsql';

SELECT add\_teacher\_type('Преподаватель по основному отделу');

SELECT add\_teacher\_type('Преподаватель по экономическому отделу');

SELECT add\_teacher\_type('Преподаватель по отделу охраны труда');

SELECT add\_stage\_name('Разработка постановки задачи, организационно-экономическая сущность программы, определение входной, постоянной, выходной информации, выбор математической модели.');

SELECT add\_stage\_name('Выбор операционной системы, выбор языка программирования и системы управления базой данных.');

SELECT add\_stage\_name('Разработка и отладка программного продукта. Испытание программы.');

SELECT add\_stage\_name('Оформление проектной документации.');

SELECT add\_stage\_name('Работа над экономическим разделом.');

SELECT add\_stage\_name('Работа над разделом охраны труда и окружающей среды.');

SELECT add\_stage\_name('Оформление пояснительной записки и графической части.');

## 4 Описание алгоритмов решения задачи и их математических реализаций

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. VisualStudio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (например, Subversion и VisualSourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода).

Список компонентов, которые использовались при написании приложения, представлен в таблице 4.1.

Таблица 4.1 – Компоненты форм

|  |  |
| --- | --- |
| Компонент | Описание |
| Label | Метка для размещения в окне не очень длинных однострочных данных. |
| TextBox | Компонент для ввода данных. |
| DateTimePicker | Компонент для выбора даты и времени |
| ComboBox | Выпадающий список элементов. |
| ListBox | Список элементов. |
| Button | Компонент представляет собой стандартную кнопку и широко используется для управления программами. |
| TrackBar | Компонент, который с помощью перемещения ползунка позволяет вводить числовые значения |
| CheckBox | Компонент предназначен для установки одного из двух значений: отмечен или не отмечен. |
| DataGridView | Компонент, который инкапсулирует двухмерную таблицу, в которой строки представляют собой набор записей, а столбцы – поля набора данных. Обеспечивает визуализацию и редактирование полей из набора данных. |
| UserControl | Компонент, который позволяет сгруппировать другие элементы и используете их все вместе. |
| MenuStrip | Элемент представляет панель инструментов. |
| Timer | Является компонентом для запуска действий, повторяющихся через определенный промежуток времени. |

Описание процедур созданного приложения перечислены в следующих таблицах.

Таблица 4.2 – Методы формы «MainForm»

|  |  |
| --- | --- |
| Процедура | Описание |
| private void logoutToolStripMenuItem\_Click(object sender, EventArgs e) | Метод для перезагрузки приложения. Используется для MenuStripItem. |
| private void exitToolStripMenuItem\_Click(object sender, EventArgs e) | Метод для закрытия приложения. Используется для MenuStripItem. |
| private void teacherProjectsToolStripMenuItem\_Click(object sender, EventArgs e) | Метод для открытия teacherProjectsControl приложения. Используется для MenuStripItem. |
| private void teacherGroupsToolStripMenuItem\_Click(object sender, EventArgs e) | Метод для открытия teacherGroupsControl приложения. Используется для MenuStripItem. |
| private void studentProjectsToolStripMenuItem\_Click(object sender, EventArgs e) | Метод для открытия studentProjectControl приложения. Используется для MenuStripItem. |
| private void setActive(object sender) | Метод для активации MenuStripItem. |
| public void hideAllControls() | Метод для скрытия всех UserControl. |
| private void MainForm\_Load(object sender, EventArgs e) | Метод для инициализации MenuStrip, строки соединения и пользователя. |
| private void teacherGroupsTimer\_Tick(object sender, EventArgs e) | Метод используется для открытия teacherGroupsThemesControl по команде. |

Таблица 4.3 – Методы формы «ConnectionForm»

|  |  |
| --- | --- |
| Процедура | Описание |
| private void showPassCheck\_CheckedChanged(object sender, EventArgs e) | Метод для скрытия либо показа пароля в passwordBox. |
| private void cancelButton\_Click(object sender, EventArgs e) | Метод для закрытия приложения при нажатии на cancelButton. |
| private void connectButton\_Click(object sender, EventArgs e) | Метод для попытки инициализации строки подключения при нажатии на connectButton. |

Таблица 4.4 – Методы формы «LoginForm»

|  |  |
| --- | --- |
| Процедура | Описание |
| private void LoginForm\_Load(object sender, EventArgs e) | Метод для инициализации Items в usernameBox. |
| private void clearButton\_Click(object sender, EventArgs e) | Метод для закрытия приложения при нажатии на cancelButton. |
| private void showPassCheck\_CheckedChanged(object sender, EventArgs e) | Метод для скрытия либо показа пароля в passwordBox. |
| private void loginButton\_Click(object sender, EventArgs e) | Метод для попытки инициализации строки пользователя при нажатии на loginButton. |

Таблица 4.5 – Методы пользовательского компонента «StudentProjectControl»

|  |  |
| --- | --- |
| Процедура | Описание |
| public void ItemsLoad() | Метод для инициализации всех компонентов. |
| private void stagesList\_SelectedIndexChanged(object sender, EventArgs e) | Метод для обновления полей при изменении выбора в stagesList. |

Таблица 4.6 – Методы пользовательского компонента «TeacherGroupsControl»

|  |  |
| --- | --- |
| Процедура | Описание |
| public void ItemsLoad() | Метод для инициализации всех компонентов. |
| private void groupsListBox\_SelectedIndexChanged(object sender, EventArgs e) | Метод для обновления полей при изменении выбора в listBox. |
| private void groupsGrid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e) | Метод для открытия компонента TeacherGroupsThemesControl с нужными параметрами при нажатии на поле таблицы. |

Таблица 4.7 – Методы пользовательского компонента «TeacherGroupsThemesControl»

|  |  |
| --- | --- |
| Процедура | Описание |
| public void ItemsLoad(int studentId, string studentName, int themeId) | Метод для инициализации всех компонентов. |
| public void UpdateGrid() | Метод для обновления таблицы stagesGrid. |
| public void ShowStages() | Метод для отображения компонентов, относящихся к stages. |
| public void HideStages() | Метод для скрытия компонентов, относящихся к stages. |
| private void updateButton\_Click(object sender, EventArgs e) | Метод для изменения значений дипломного проекта. |
| private void addButton\_Click(object sender, EventArgs e) | Метод для создания дипломного проекта. |
| private void deleteButton\_Click(object sender, EventArgs e) | Метод для удаления дипломного проекта. |
| private void stageEditButton\_Click(object sender, EventArgs e) | Метод для изменения этапа дипломного проекта. |
| private void stagesGrid\_SelectionChanged(object sender, EventArgs e) | Метод для смены значений в полях, относящихся к stages при изменении выбранного поля в stagesGrid. |

Таблица 4.8 – Методы пользовательского компонента «TeacherProjectsControl»

|  |  |
| --- | --- |
| Процедура | Описание |
| public void ItemsLoad() | Метод для инициализации всех компонентов . |

Продолжение таблицы 4.8

|  |  |
| --- | --- |
| Процедура | Описание |
| private void stagesList\_SelectedIndexChanged(object sender, EventArgs e) | Метод для обновления полей при изменении выбора в stagesList. |
| private void studentsList\_SelectedIndexChanged(object sender, EventArgs e) | Метод для обновления полей при изменении выбора в studentsList. |
| public void UpdateStages() | Метод для обновления полей дипломных проектов. |
| public void UpdateStages2() | Метод для обновления полей этапов дипломных проектов. |
| private void submitButton\_Click(object sender, EventArgs e) | Метод для изменения статуса этапа при нажатии. |
| private void cancelButton\_Click(object sender, EventArgs e) | Метод для изменения статуса этапа при нажатии. |
| private void percentageBar\_Scroll(object sender, EventArgs e) | Метод для изменения процента выполнения этапа при перемещении курсора percentageBar. |
| private void percentageBox\_TextChanged(object sender, EventArgs e) | Метод для изменения процента выполнения этапа при изменении текста в percentageBox. |
| private void updateTimer\_Tick(object sender, EventArgs e) | Метод для обновления процента выполнения этапа в базе данных с задержкой. |
| private void updatePercentage() | Метод для обновления процента выполнения этапа в базе данных. |
| private void reportButton\_Click(object sender, EventArgs e) | Метод для формирования Excel отчета. |

Таблица 4.9 – Методы класса «Settings»

|  |  |
| --- | --- |
| Процедура | Описание |
| public static bool LoadConStr() | Метод для загрузки строки подключения из файла. |
| public static void SaveConStr() | Метод для сохранения строки подключения к базе данных. |

Таблица 4.10 – Методы класса «DataBaseGet»

|  |  |
| --- | --- |
| Процедура | Описание |
| public static List<T> Get<T>(string cmd, string custom\_query\_parameters, Func<NpgsqlDataReader, T> handler) | Generic метод для получения данных из базы данных. |
| public static List<TeacherType> TeacherTypes(string custom\_query\_parameters) | Метод для получения данных о типах учителя. |
| public static List<Teacher> Teachers(string custom\_query\_parameters) | Метод для получения данных об учителях. |
| public static List<Group> Groups(string custom\_query\_parameters) | Метод для получения данных о группах. |

Продолжение таблицы 4.10

|  |  |
| --- | --- |
| Процедура | Описание |
| public static List<Student> Students(string custom\_query\_parameters) | Метод для получения данных о студентах. |
| public static List<User> Users(string custom\_query\_parameters) | Метод для получения данных о пользователях. |
| public static List<Theme> Themes(string custom\_query\_parameters) | Метод для получения данных о дипломных проектах. |
| public static List<StageName> StageNames(string custom\_query\_parameters) | Метод для получения данных о названиях этапов. |
| public static List<Stage> Stages(string custom\_query\_parameters) | Метод для получения данных об этапах дипломного проекта. |
| public static bool PasswordIsValid(int id, string pass) | Метод для проверки пароля и логина пользователя. |

Таблица 4.11 – Методы класса «DataBaseAdd»

|  |  |
| --- | --- |
| Процедура | Описание |
| public static string TeacherType(string type\_name) | Метод для добавления записи тип учителя в базу данных. |
| public static string Teacher(int type\_id, string teacher\_name, string address, string phone, string pass) | Метод для добавления записи учитель в базу данных. |
| public static string Group(int teacher\_id, string group\_name, int year\_started) | Метод для добавления записи группа в базу данных. |
| public static string Student(int group\_id, string student\_name, string phone, string pass) | Метод для добавления записи студент в базу данных. |
| public static string Theme(int student\_id, string theme\_name, int main\_teacher\_id, int econ\_teacher\_id, int safe\_teacher\_id) | Метод для добавления записи дипломный проект в базу данных. |
| public static string StageName(string stage\_name) | Метод для добавления записи имя этапа в базу данных. |
| public static string Stage(int stage\_name\_id, int theme\_id, int teacher\_id, int percentage, DateTime date\_started, DateTime date\_ended) | Метод для добавления записи этап дипломного проекта в базу данных. |

Таблица 4.12 – Методы класса «DataBaseAdd»

|  |  |
| --- | --- |
| Процедура | Описание |
| public static string Delete(int id, string cmd) | Метод для удаления записи из базы данных. |
| public static string TeacherType(int id) | Метод для удаления записи учитель из базы данных. |
| public static string Teacher(int id) | Метод для удаления записи группа из базы данных. |

Продолжение таблицы 4.12

|  |  |
| --- | --- |
| public static string Student(int id) | Метод для удаления записи студент из базы данных. |
| public static string Theme(int id) | Метод для удаления записи дипломный проект из базы данных. |
| public static string StageName(int id) | Метод для удаления записи имя этапа из базы данных. |
| public static string Stage(int id) | Метод для удаления записи этап дипломного проекта из базы данных. |

Таблица 4.13 – Методы класса «DataBaseUpdate»

|  |  |
| --- | --- |
| Процедура | Описание |
| public static string TeacherType(int teacher\_type\_id, string type\_name) | Метод для изменения записи тип учителя в базе данных. |
| public static string Teacher(int teacher\_id, int type\_id, string teacher\_name, string address, string phone, string pass) | Метод для изменения записи учитель в базе данных. |
| public static string Group(int group\_id, int teacher\_id, string group\_name, int year\_started) | Метод для изменения записи группа в базе данных. |
| public static string Student(int student\_id, int group\_id, string student\_name, string phone, string pass) | Метод для изменения записи студент в базе данных. |
| public static string Theme(int theme\_id, int student\_id, string theme\_name, int main\_teacher\_id, int econ\_teacher\_id, int safe\_teacher\_id) | Метод для изменения записи дипломный проект в базе данных. |
| public static string StageName(int stage\_name\_id, string stage\_name) | Метод для изменения записи имя этапа в базе данных. |
| public static string Stage(int stage\_id, int stage\_name\_id, int theme\_id, int teacher\_id, int percentage, DateTime date\_started, DateTime date\_ended) | Метод для изменения записи этап дипломного проекта в базу данных. |
| public static string StagePercentageOnly(int stage\_id, int percentage) | Метод для изменения процента выполнения в записи этап дипломного проекта в базу данных. |
| public static string StageTeacherDates(int stage\_id, int teacher\_id, DateTime started\_date, DateTime ended\_date) | Метод для изменения дат и учителя в записи этап дипломного проекта в базу данных. |

## 5 Разработка диалогов приложения пользователя

Основное назначение создаваемой программы хранение справочной информации по дипломным проектам.

Структуру меню программ можно описать основными блоками – «Форма подключения к базе данных», «Форма Авторизации», «Форма со статусом дипломного проекта», «Форма со списком учащихся», «Форма создания дипломного проекта», «Форма со статусами дипломных проектов учащихся».

Работа программы осуществляется по диалоговому и событийному режиму, при этом под диалогом понимается предоставление пользователю нескольких альтернатив и обработка его выбора. В диалоговую систему входят: главное меню с соответствующими всплывающими подменю, а также диалоговые окна. Под событиями понимаются процессы, активизируемые пользователем, а также программные события – получение определенным полем фокуса редактирование или потеря фокуса ввода. На основании данных событий активизируются процедуры контроля допустимости данных. В компьютерных системах баз данных пользователи для ввода, просмотра и распечатки квитанций с информацией базы данных могут применять формы.

«Форма подключения к базе данных» позволяет подключиться к базе данных. Форма представлена на рисунке 5.1.

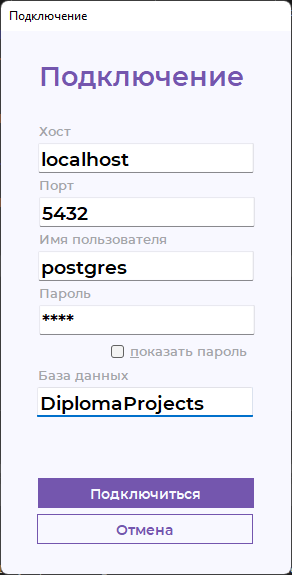


Рисунок 5.1 – «Форма подключения к базе данных»

«Форма Авторизации» позволяет авторизоваться пользователю. Форма представлена на рисунке 5.2.

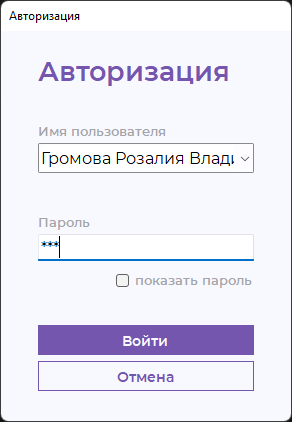


Рисунок 5.2 – «Форма Авторизации»

В зависимости от пользователя могут быть представлены разные пользовательские компоненты. В начале будут описаны компоненты для пользователя типа «Преподаватель».

«Форма со списком учащихся» предназначена для просмотра куратором списка учащихся, а также добавления дипломных проектов учащимся. Представлена на рисунке 5.3.

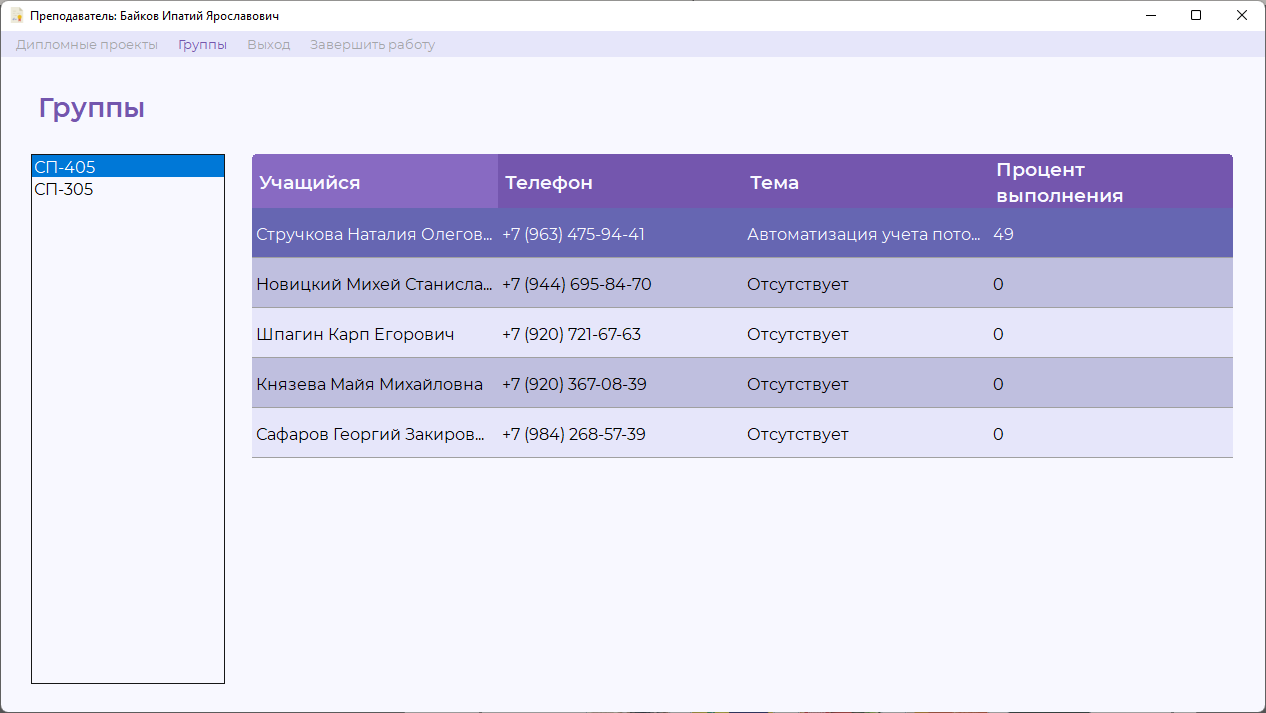


Рисунок 5.3 – «Форма со списком учащихся»

При двойном нажатии по учащемуся на «Форма со списком учащихся» (Рисунок 5.3) откроется «Форма создания дипломного проекта» (Рисунок 5.4). Форма позволяет управлять информацией о дипломных проектах.

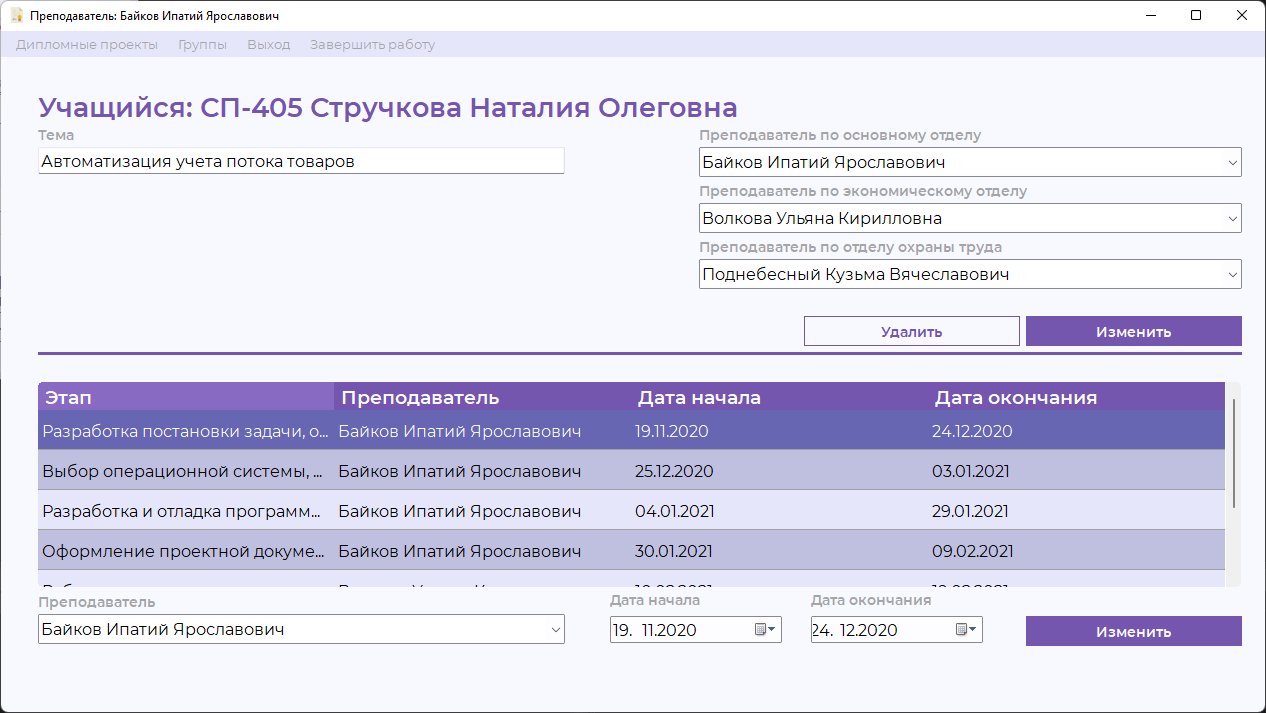


Рисунок 5.4 – «Форма создания дипломного проекта»

«Форма со списком учащихся» предназначена для просмотра куратором списка учащихся, а также добавления дипломных проектов учащимся. Представлена на рисунке 5.5.

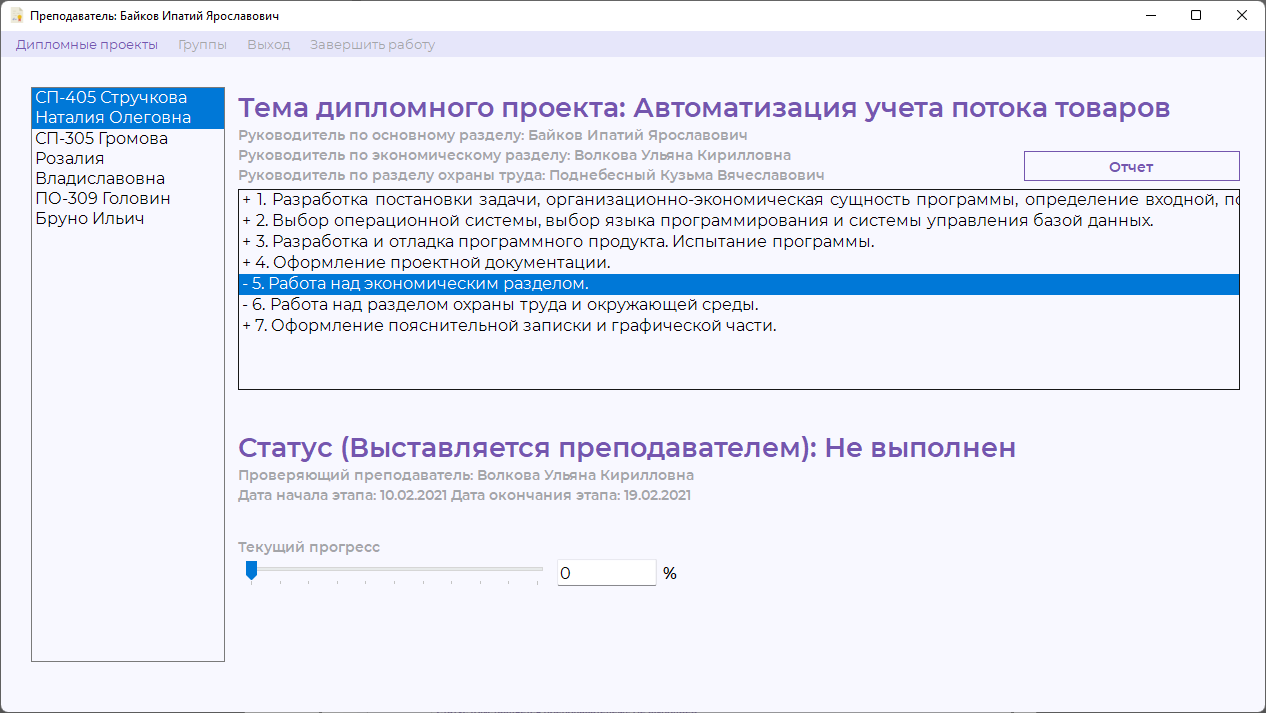


Рисунок 5.5 – «Форма со статусами дипломных проектов учащихся»

При авторизации за учащегося будет отображена «Форма со статусом дипломного проекта» (Рисунок 5.6). Она предназначена для отображения статуса дипломного проекта учащемуся.

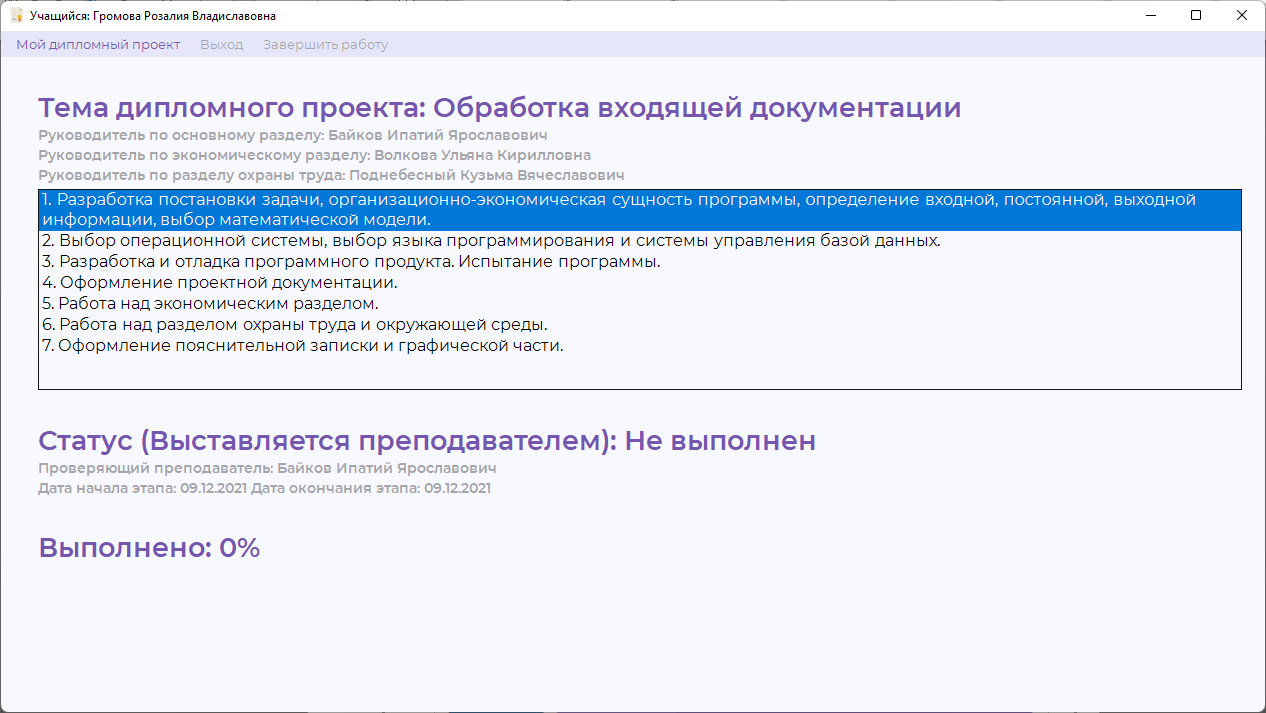


Рисунок 5.6 – «Форма со статусом дипломного проекта»

## 6 Руководство пользователя

Программа включает в себя удобный и понятный пользователю интерфейс, формы для ввода информации в базу данных, получение отчета на печать. В программе разработанная двухуровневая система пользователей:

* программа для учащихся позволит просмотреть статус дипломного проекта;
* программа для преподавателя позволяет просматривать из изменять статус дипломных проектов, которые они ведут, добавлять дипломные проекты учащимся их групп.

Выполнение программы:

* перед началом работы с программой необходимо проверить соответствие аппаратных и программных средств компьютера указанным требованиям;
* после первого запуска будет предложено подключиться к базе данных (Рисунок 6.1). Если программа успешно подключится к базе данных, то она сохранит введенные данные, что пользователю не приходилось каждый раз вводить их;
* при запуске программы появится окно авторизации (Рисунок 6.2). После успешной авторизации откроется главная форма программы.

Подключение к базе данных:

Чтобы подключиться к базе данных нужно ввести данные в поля «Хост», «Порт», «Имя пользователя», «Пароль» и «База данных». После этого нужно нажать на кнопку «Подключиться» либо на клавишу «Enter». В случае успешного подключение будет выдано сообщение об успешном подключении, в противном случае будет выдано сообщение с ошибкой. Поле «Показать пароль» скрывает либо показывает поле «Пароль». Чтобы выйти из программы нужно нажать на кнопку «Отмена» либо на клавишу «Esc».

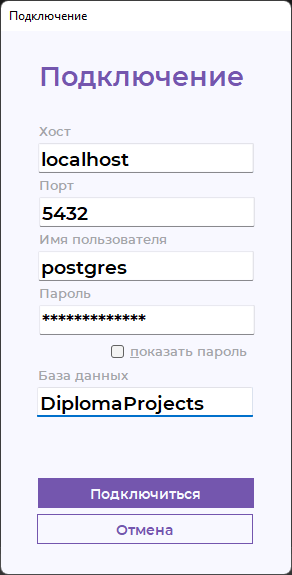


Рисунок 6.1 – Окно подключения к базе данных

Авторизация:

Для авторизации пользователя нужно выбрать существующее имя пользователя и ввести пароль. После нажатия на кнопку «Войти» либо на клавишу «Enter»: будет выдана ошибка, если имя пользователя и/или пароль неверны, данное окно закроется и откроется главная форма в случае успешной авторизации. Поле «Показать пароль» скрывает либо показывает поле «Пароль». Чтобы выйти из программы нужно нажать на кнопку «Отмена» либо на клавишу «Esc».

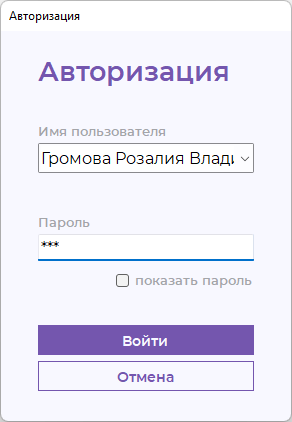


Рисунок 6.2 – Окно авторизации

Авторизация за учащегося:

После авторизации за учащегося появится главное окно с формой «Мой дипломный проект». Сверху в панели кнопка «Мой дипломный проект» (ctrl + w) перезагрузит форму, «Выход» (ctrl + r) выйдет из аккаунта и покажет окно авторизации, «Завершить работу» (ctrl + t) выйдет из приложения. На форме отображены следующий элементы: тема дипломного проекта учащегося, руководители по различным разделам данного дипломного проекта, список этапов дипломного проекта, статус выбранного этапа дипломного проекта, проверяющий преподаватель, дата начала и окончания выбранного этапа и процент выполнения выбранного этапа.

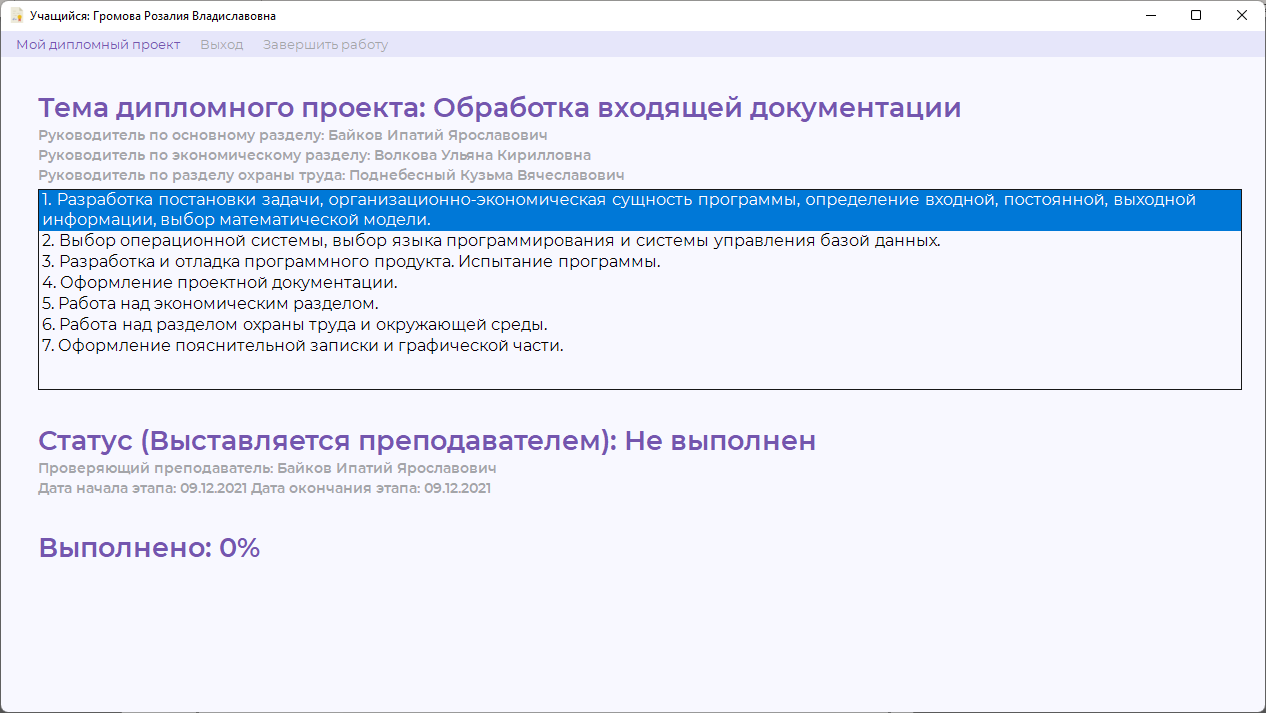


Рисунок 6.3 – Окно дипломных проектов учащихся

Авторизация за преподавателя:

После авторизации за преподавателя появится главное окно с формой «Группы преподавателя». Сверху в панели кнопка «Дипломные проекты» (ctrl + w) откроет форму «Список дипломных проектов», «Группы» (ctrl + r) откроет форму «Группы преподавателя», «Выход» (ctrl + r) выйдет из аккаунта и покажет окно авторизации, «Завершить работу» (ctrl + t) выйдет из приложения. На форме «Группы» (Рисунок 6.4) отображены следующий элементы: список групп преподавателя, список учащихся выбранной группы.

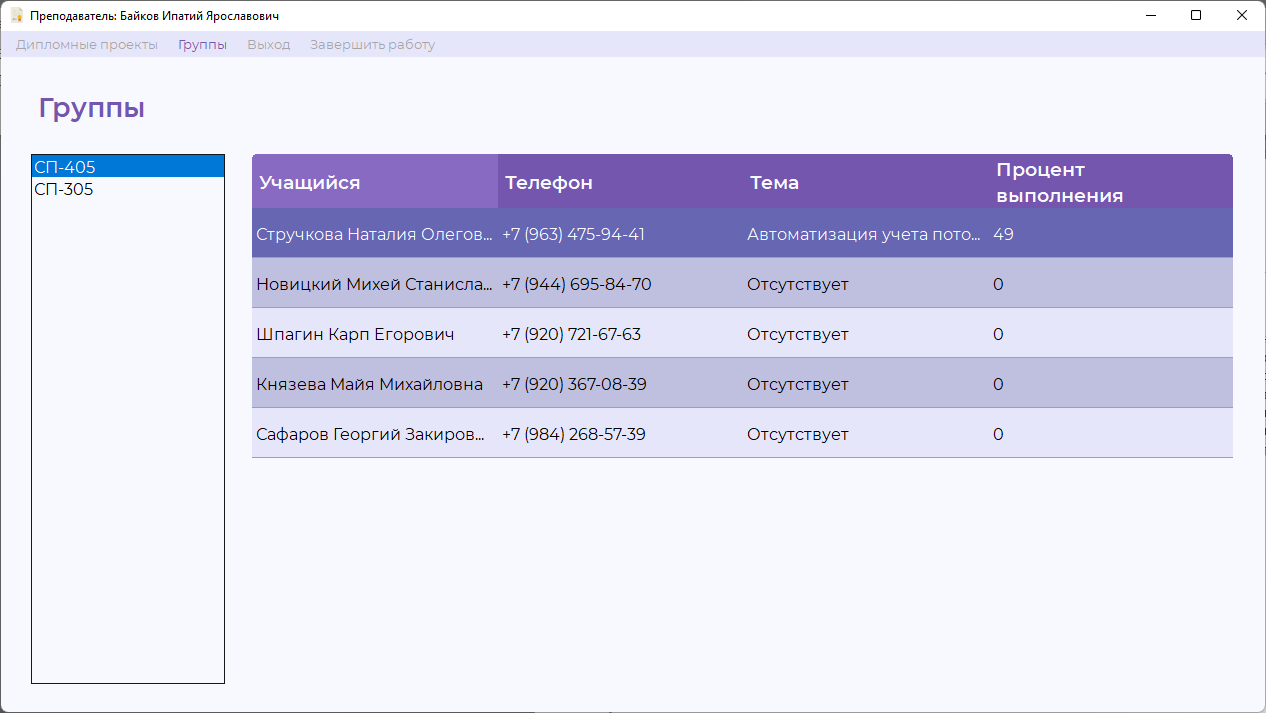


Рисунок 6.4 – Форма «Группы преподавателя»

При двойном нажатии на учащегося откроется форма «Дипломный проект учащегося» (Рисунок 6.5 либо Рисунок 6.6). В случае, если у учащегося темы еще нет, будет отображена форма без этапов проекта (Рисунок 6.5), если же тема есть, то будет отображена форма с этапами проекта (Рисунок 6.6). Чтобы добавить проект нужно заполнить поля «Тема», «Преподаватель по основному отделу», «Преподаватель по экономическому разделу», «Преподаватель по отделу охраны труда», затем нажать на кнопку «Создать». В случае успешного создания отобразятся этапы проекта (Рисунок 6.6). Чтобы изменить данные проекта нужно нажать на кнопку «Изменить», чтобы удалить проект нужно нажать на кнопку «Удалить». Чтобы изменить данные этапов проекта нужно выбрать этап в таблице, заменить данные в полях этапов и нажать на кнопку «Изменить» в поле этапов.

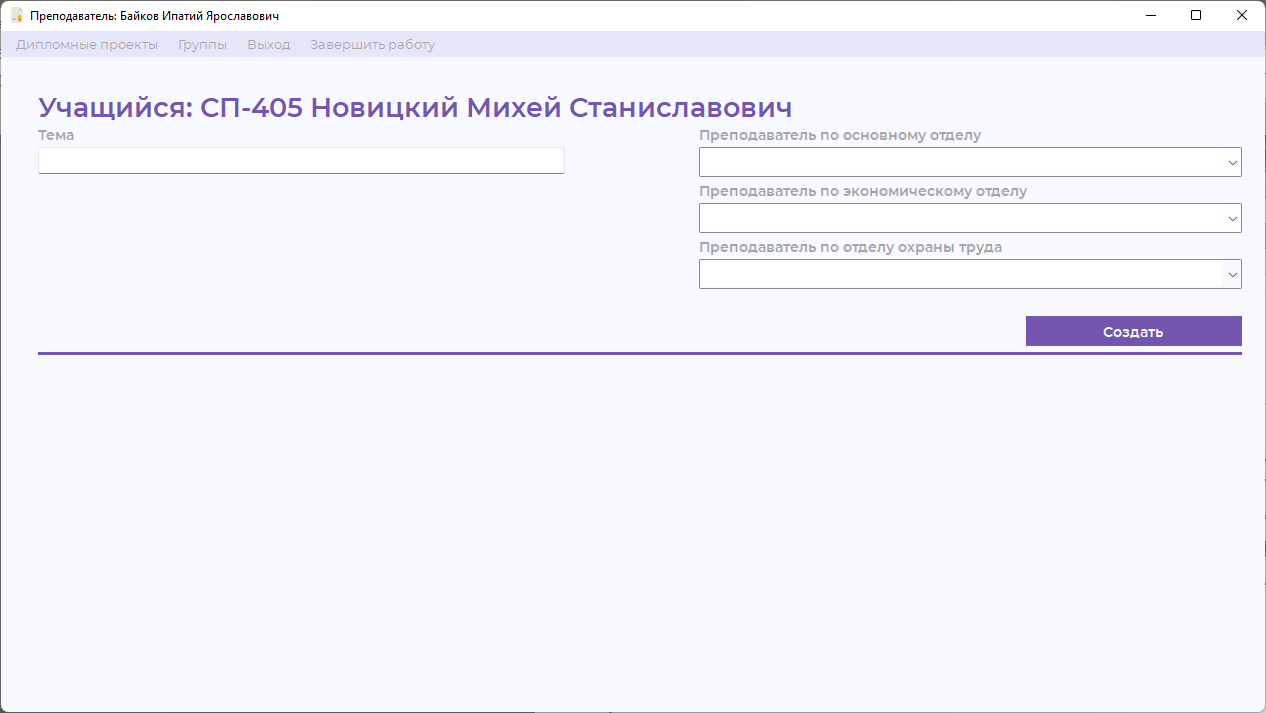


Рисунок 6.5 – Форма «Дипломный проект учащегося» (еще нету проекта)

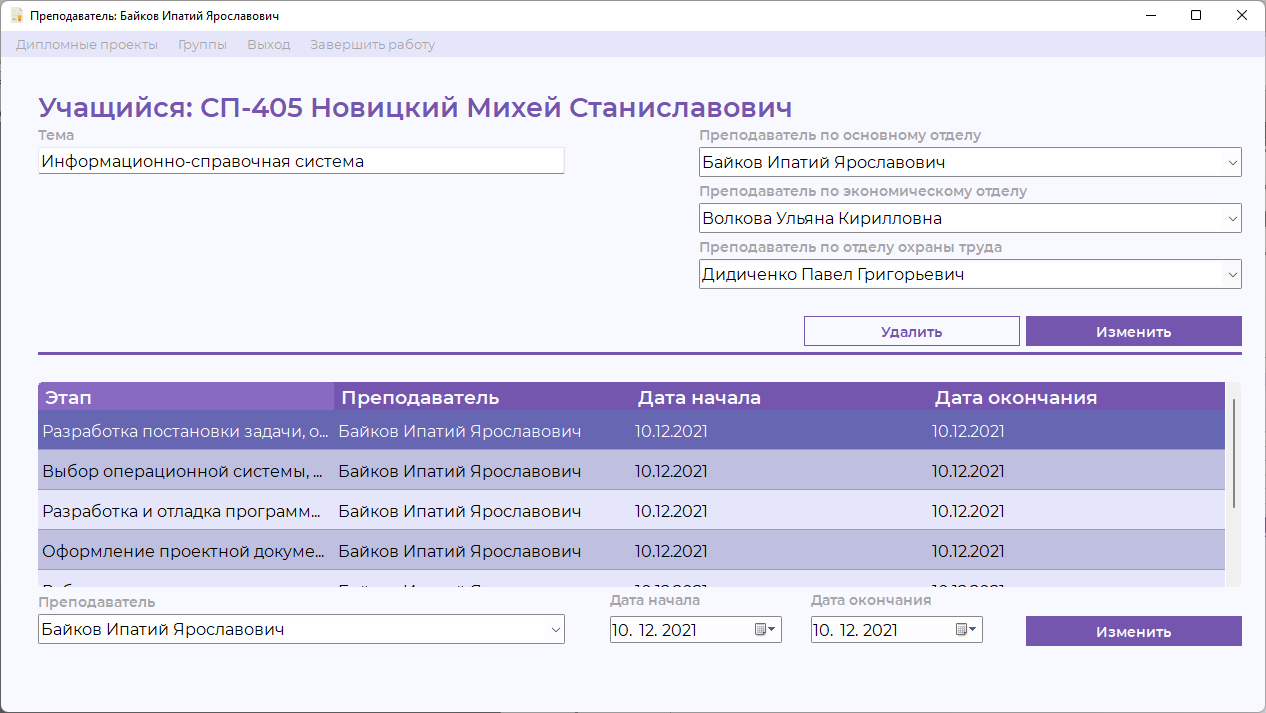


Рисунок 6.6 – Форма «Дипломный проект учащегося» (есть проект)

Дипломные проекты:

Форма «Дипломные проекты» делится на 2 части: список дипломных проектов у данного преподавателя и статус дипломного проекта. Статус дипломного проекта отображает статус выбранного дипломного проекта в списке. В статусе дипломного проекта отображены следующий элементы: тема дипломного проекта учащегося, руководители по различным разделам данного дипломного проекта, список этапов дипломного проекта, статус выбранного этапа дипломного проекта, проверяющий преподаватель, дата начала и окончания выбранного этапа и процент выполнения выбранного этапа. В случае, если статус этапа «Выполнен» процент выполнения будет скрыт (Рисунок 6.7-6.8). Преподаватель может изменять статус выполнения только тех этапов, слева от которых в списке есть «Плюс». Чтобы сформировать отчет нужно выбрать в списке дипломных проектов проект, по которому нужно сделать отчет, затем нажимаете на кнопку «Отчет» справа. После нажатия сформируется отчет (Рисунок 6.8).

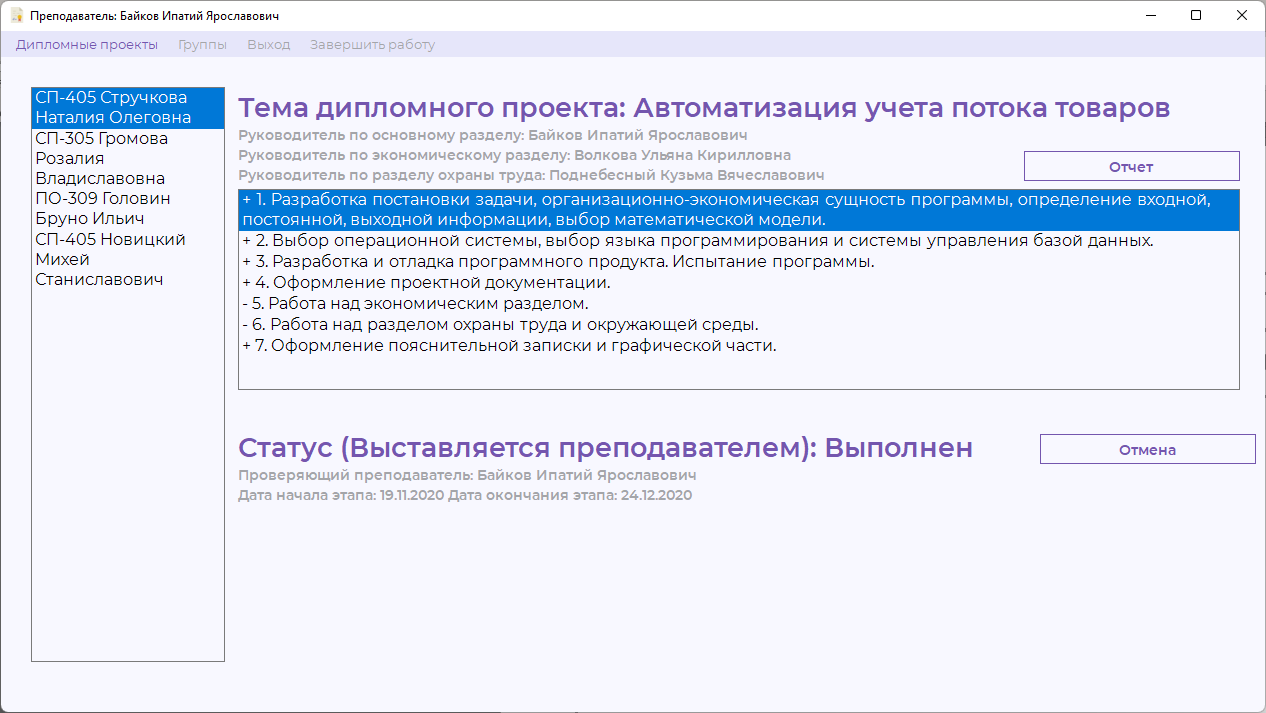


Рисунок 6.7 – Форма «Дипломные проекты» (Этап выполнен)

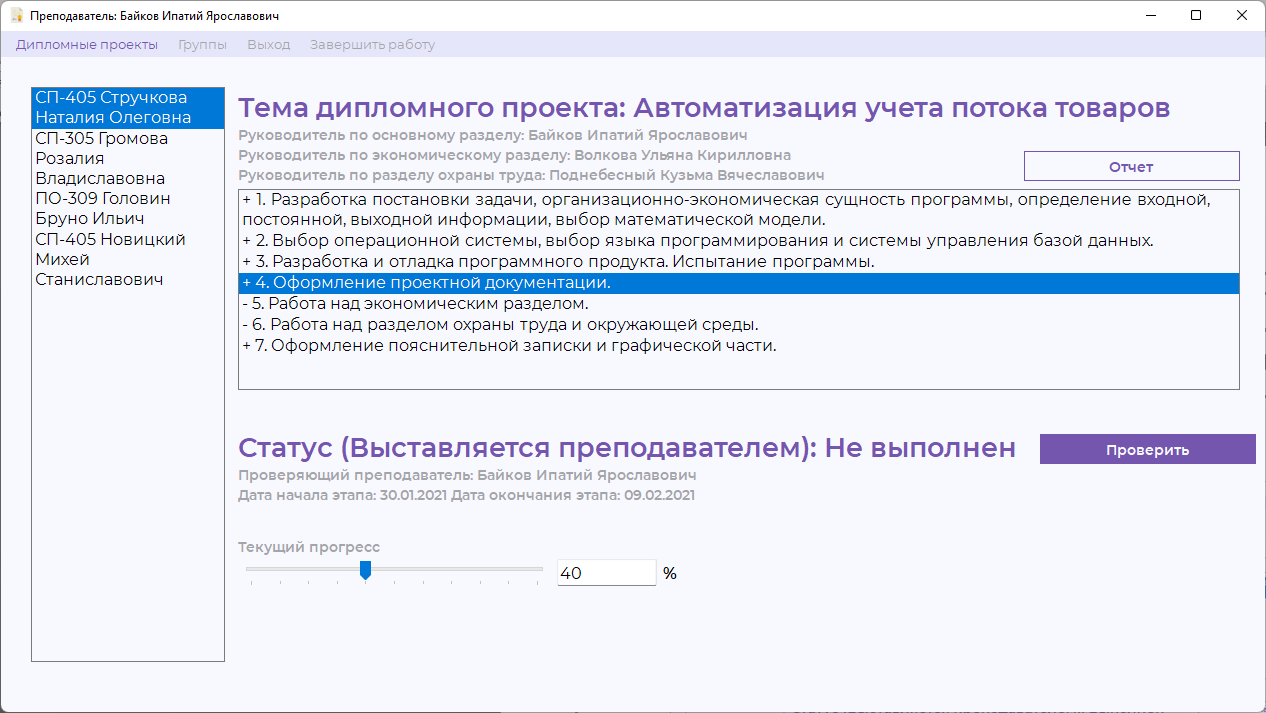


Рисунок 6.8 – Форма «Дипломные проекты» (Этап не выполнен)

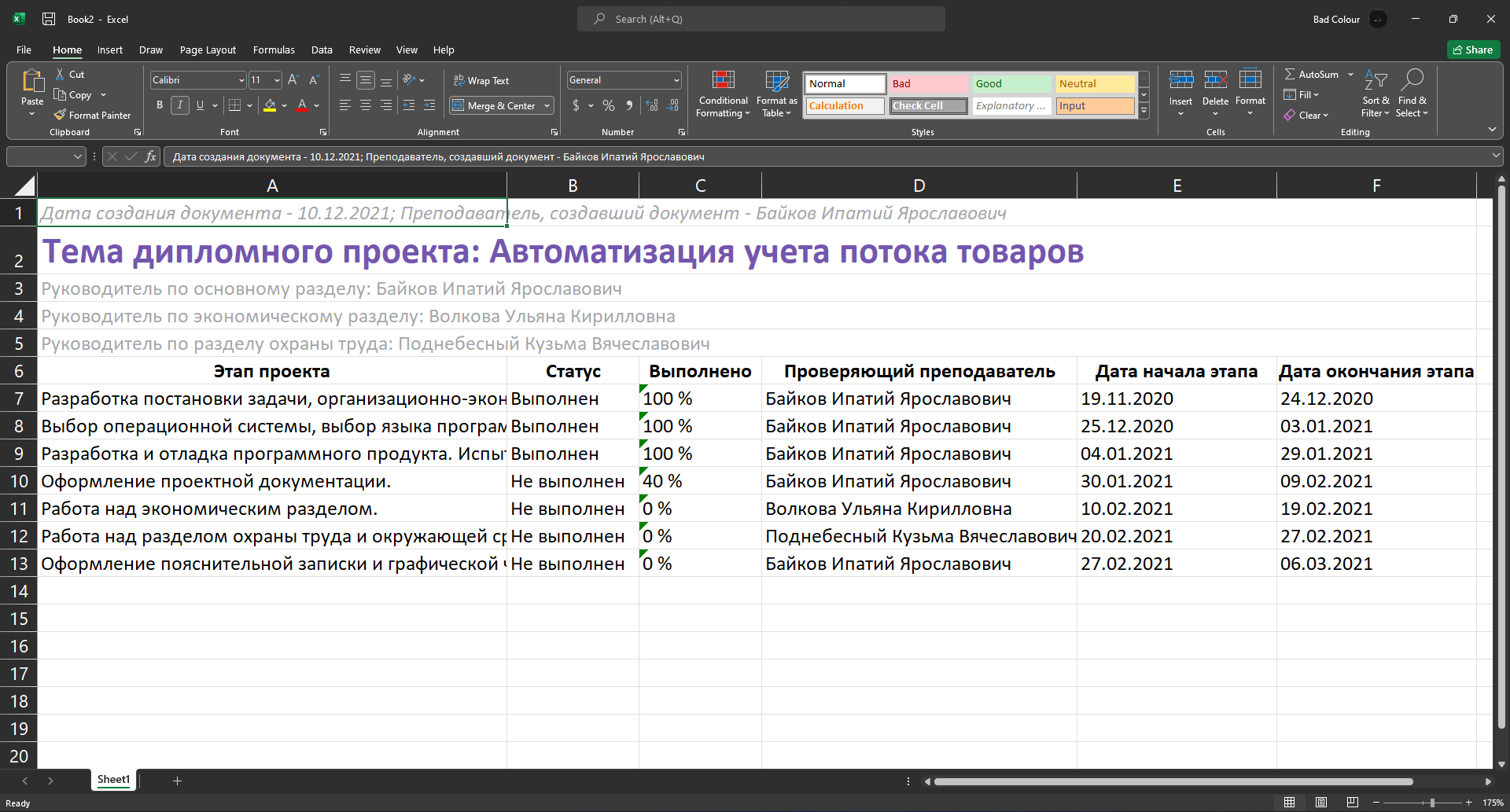


Рисунок 6.9 – Отчет по дипломному проекту

## ЗАКЛЮЧЕНИЕ

В ходе работы над курсовым проектом были применены на практике изученные методы и приемы программирования, закреплены навыки и умения по проектированию и программированию учебных задач, закреплены практические навыки в области визуального программирования.

Итогом курсового проектирования стало создание информационно-справочной системы «Дипломное проектирование», с приемлемым графическим интерфейсом пользователя, в среде программирования «Microsoft Visual Studio 2022» на языке «C#». Эта среда разработки является наиболее выгодной при создании таких сравнительно несложных и небольших по размеру программ, в частности приложений с базами данных.

Данная программа является завершённым программным продуктом, хотя возможна ее дальнейшая доработка, для осуществления которой необходимо сначала в течение некоторого периода времени применять ее в реальных жизненных условиях, а потом исправлять возникшие недочеты, которые допустил разработчик.

В ходе проектирования автоматизации построена логическая и физическая модели данных и даны их описания, функциональная модель.

Дана характеристика и описание входной и оперативной информации, а также описана их структура. Здесь также отражена структура каждого файла оперативной, постоянной информации полученных в результате решения задач.

При разработке структуры диалога была спроектирована работа с первичными документами, со справочниками и отчетами.

Используя дружественный, проработанный интерфейс с использованием многочисленных справочников, увеличилась скорость обработки информации по учету подписок на издания и снизилась утомляемость людей, ответственных за этот участок работы.

В целом задача была выполнена в полной мере, и программное средство может быть использована на практике. Проекта были усовершенствованы знания в области объектно-ориентированного программирования, разработке баз данных, приобретены навыки использования научно-технической и экономической информации.

|  |
| --- |
|  |

Программное средство имеет простой и интуитивно понятный интерфейс для удобной и простой работы в ней, реализована возможность взаимодействия с данными из базы данных и формирование отчётов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Нэш, Т. C# 2008: Ускоренный курс для профессионалов / Т. Нэш. - М.: «Вильямс», 2018.

2. Пауэрс, Л. Microsoft Visual Studio / Л. Пауэрс, Снелл М.- СПб.: БХВ-Петербург, 2019.

3. Петцольд, Ч. Программирование для Microsoft Windows на C#. В 2 томах. Том 2. / Ч. Петцольд. - М.: Издательско-торговый дом «Русская Редакция», 2002.

4. Троелсен, Э. C# и платформа .NET. Библиотека программиста / Э. Троелсен. - СПб.: Питер, 2008.

5. Кирилова, Г.И. Информационные технологии и компьютерные средства в образовании // Educational technology & Society., 2011

6. Немнюгин, С. Программирование на языке высокого уровня : Учебник для вузов – 2-е издание / С. Немнюгин. – Санкт-Петербург: ООО «Питер Пресс»,2017. – 544с.

7. Ник Рендольф, Дэвид Гарднер, Майкл Минутилло, Крис Андерсон. Visual Studio 2010 для профессионалов = Professional Visual Studio 2010. — М.: «Диалектика», 2011. — С. 1184.

8. Джон, Скит. C# для профессионалов: тонкости программирования, 3-е издание, новый перевод = C# in Depth, 3rd ed.. — М.: «Вильямс», 2014. — 608 с.

9. Кристиан, Н. C# 5.0 и платформа .NET 4.5 для профессионалов = Professional C# 5.0 and .NET 4.5. — М.: «Диалектика», 2013. — 1440 с.

10. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. Язык программирования C#. Классика Computers Science. 4-е издание = C# Programming Language (Covering C# 4.0), 4th Ed. — СПб.: «Питер», 2012. — 784 с.

11. Э. Стиллмен, Дж. Грин. Изучаем C#. 2-е издание = Head First C#, 2ed. — СПб.: «Питер», 2012. — 704 с.

12. Герберт, Ш. C# 4.0: полное руководство = C# 4.0 The Complete Reference. — М.: «Вильямс», 2010. — С. 1056.

## ПРИЛОЖЕНИЕ А

(Обязательное)

Листинг программы

**Settings.cs**

using System;

using System.Data;

using System.IO;

using Npgsql;

using Newtonsoft.Json;

namespace CourseProject.Classes

{

public static class Settings

{

public static NpgsqlConnectionStringBuilder conStr = new NpgsqlConnectionStringBuilder();

public static User user;

public static bool LoadConStr()

{

if (!Directory.Exists(".\\settings"))

{

return false;

}

if (!File.Exists(".\\settings\\connection\_string.json"))

{

return false;

}

string json = "";

using (StreamReader sr = new StreamReader(@".\\settings\\connection\_string.json"))

{

json = sr.ReadToEnd();

}

conStr = JsonConvert.DeserializeObject<NpgsqlConnectionStringBuilder>(json);

if (!CheckDatabaseConnection(conStr))

{

conStr = null;

return false;

}

else

return true;

}

public static void SaveConStr()

{

JsonSerializer serializer = new JsonSerializer();

serializer.NullValueHandling = NullValueHandling.Ignore;

if (!Directory.Exists(".\\settings"))

{

Directory.CreateDirectory(".\\settings");

}

using (StreamWriter sw = new StreamWriter(@".\\settings\\connection\_string.json"))

using (JsonWriter writer = new JsonTextWriter(sw))

{

serializer.Serialize(writer, conStr);

}

}

public static bool CheckDatabaseConnection(NpgsqlConnectionStringBuilder temp)

{

using (NpgsqlConnection conn = new NpgsqlConnection(temp.ConnectionString))

{

try

{

conn.Open();

if (conn.State == ConnectionState.Open)

return true;

}

catch (Exception) { }

finally

{

conn.Close();

}

return false;

}

}

}

}

**DataBaseTables.cs**

using System;

namespace CourseProject.Classes

{

public class TeacherType

{

public int type\_id { get; set; }

public string type\_name { get; set; }

public TeacherType (int type\_id, string type\_name)

{

this.type\_id = type\_id;

this.type\_name = type\_name;

}

public TeacherType ()

{

this.type\_id = 0;

this.type\_name = "";

}

}

public class Teacher : TeacherType

{

public int teacher\_id { get; set; }

public string teacher\_name { get; set; }

public string address { get; set; }

public string phone { get; set; }

public Teacher(int teacher\_id, int type\_id, string type\_name, string teacher\_name, string address, string phone)

{

this.teacher\_id = teacher\_id;

this.type\_id = type\_id;

this.type\_name = type\_name;

this.teacher\_name = teacher\_name;

this.address = address;

this.phone = phone;

}

public Teacher()

{

this.teacher\_id = 0;

this.type\_id = 0;

this.type\_name = "";

this.teacher\_name = "";

this.address = "";

this.phone = "";

}

public override string ToString()

{

return this.teacher\_name;

}

}

public class Group

{

public int group\_id { get; set; }

public int teacher\_id { get; set; }

public string teacher\_name { get; set; }

public string group\_name { get; set; }

public int year\_started { get; set; }

public Group (int group\_id, int teacher\_id, string teacher\_name, string group\_name, int year\_started)

{

this.group\_id = group\_id;

this.teacher\_id = teacher\_id;

this.teacher\_name = teacher\_name;

this.group\_name = group\_name;

this.year\_started = year\_started;

}

public Group()

{

this.group\_id = 0;

this.teacher\_id = 0;

this.teacher\_name = "";

this.group\_name = "";

this.year\_started = 0;

}

}

public class Student

{

public int student\_id { get; set;}

public int group\_id { get; set; }

public string group\_name { get; set; }

public string student\_name { get; set; }

public string phone { get; set; }

public int theme\_id { get; set; }

public string theme\_name { get; set; }

public int percentage { get; set; }

public Student (int student\_id, int group\_id, string group\_name, string student\_name, string phone, int theme\_id, string theme\_name, int percentage)

{

this.student\_id = student\_id;

this.group\_id = group\_id;

this.group\_name = group\_name;

this.student\_name = student\_name;

this.phone = phone;

this.theme\_id = theme\_id;

this.theme\_name = theme\_name;

this.percentage = percentage;

}

public Student ()

{

this.student\_id = 0;

this.group\_id = 0;

this.group\_name = "";

this.student\_name = "";

this.phone = "";

this.theme\_id = 0;

this.theme\_name = "";

this.percentage = 0;

}

}

public class User

{

public int user\_id { get; set; }

public bool is\_teacher { get; set; }

public int id { get; set; }

public string name { get; set; }

public User(int user\_id, bool is\_teacher, int id, string name)

{

this.user\_id = user\_id;

this.is\_teacher = is\_teacher;

this.id = id;

this.name = name;

}

public User ()

{

this.user\_id = 0;

this.is\_teacher = false;

this.id = 0;

this.name = "";

}

public override string ToString()

{

return this.name;

}

}

public class Theme

{

public int theme\_id { get; set;}

public int student\_id { get; set; }

public string student\_name { get; set; }

public string theme\_name { get; set;}

public int main\_teacher\_id { get; set; }

public int econ\_teacher\_id { get; set; }

public int safe\_teacher\_id { get; set; }

public string main\_teacher\_name { get; set; }

public string econ\_teacher\_name { get; set; }

public string safe\_teacher\_name { get; set; }

public Theme (int theme\_id, int student\_id, string student\_name, string theme\_name, int main\_teacher\_id, int econ\_teacher\_id, int safe\_teacher\_id)

{

this.theme\_id = theme\_id;

this.student\_id = student\_id;

this.student\_name = student\_name;

this.theme\_name = theme\_name;

this.main\_teacher\_id = main\_teacher\_id;

this.econ\_teacher\_id = econ\_teacher\_id;

this.safe\_teacher\_id = safe\_teacher\_id;

this.main\_teacher\_name = "";

this.econ\_teacher\_name = "";

this.safe\_teacher\_name = "";

}

public Theme ()

{

this.theme\_id = 0;

this.student\_id = 0;

this.student\_name = "";

this.theme\_name = "";

this.main\_teacher\_id = 0;

this.econ\_teacher\_id = 0;

this.safe\_teacher\_id = 0;

this.main\_teacher\_name = "";

this.econ\_teacher\_name = "";

this.safe\_teacher\_name = "";

}

public void AddTeacherNames(string main\_teacher\_name, string econ\_teacher\_name, string safe\_teacher\_name)

{

this.main\_teacher\_name = main\_teacher\_name;

this.econ\_teacher\_name = econ\_teacher\_name;

this.safe\_teacher\_name = safe\_teacher\_name;

}

}

public class StageName

{

public int stage\_name\_id { get; set;}

public string stage\_name { get; set;}

public StageName (int stage\_name\_id, string stage\_name)

{

this.stage\_name\_id = stage\_name\_id;

this.stage\_name = stage\_name;

}

public StageName ()

{

this.stage\_name\_id = 0;

this.stage\_name = "";

}

}

public class Stage : StageName

{

public int stage\_id { get; set;}

public int theme\_id { get; set;}

public int teacher\_id { get; set;}

public string teacher\_name { get; set;}

public int percentage { get; set;}

public DateTime date\_started { get; set;}

public DateTime date\_ended { get; set;}

public Stage (int stage\_id, int stage\_name\_id, string stage\_name, int theme\_id, int teacher\_id, string teacher\_name, int percentage, DateTime date\_started, DateTime date\_ended)

{

this.stage\_id = stage\_id;

this.stage\_name\_id = stage\_name\_id;

this.stage\_name = stage\_name;

this.theme\_id = theme\_id;

this.teacher\_id = teacher\_id;

this.teacher\_name = teacher\_name;

this.percentage = percentage;

this.date\_started = date\_started;

this.date\_ended = date\_ended;

}

public Stage ()

{

this.stage\_id = 0;

this.theme\_id = 0;

this.teacher\_id = 0;

this.teacher\_name = "";

this.percentage = 0;

this.date\_started = DateTime.Now;

this.date\_ended = DateTime.Now;

}

}

}

**DataBaseGet.cs**

using System;

using System.Collections.Generic;

using Npgsql;

namespace CourseProject.Classes

{

public static class DataBaseGet

{

public static List<T> Get<T>(string cmd, string custom\_query\_parameters, Func<NpgsqlDataReader, T> handler)

{

List<T> buf = new List<T>();

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand(cmd + " " + custom\_query\_parameters, conn))

{

command.Parameters.Clear();

command.Prepare();

using (NpgsqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

buf.Add(handler(reader));

}

}

}

}

catch (Exception)

{

return null;

}

finally

{

conn.Close();

}

}

return buf;

}

public static List<TeacherType> TeacherTypes(string custom\_query\_parameters)

{

return Get("SELECT \* FROM get\_teacher\_types()", custom\_query\_parameters, (reader) =>

{

return new TeacherType(reader.GetInt32(0), reader.GetString(1));

});

}

public static List<Teacher> Teachers(string custom\_query\_parameters)

{

return Get("SELECT \* FROM get\_teachers()", custom\_query\_parameters, (reader) =>

{

return new Teacher(

reader.GetInt32(0), reader.GetInt32(1), reader.GetString(2),

reader.GetString(3), reader.GetString(4), reader.GetString(5)

);

});

}

public static List<Group> Groups(string custom\_query\_parameters)

{

return Get("SELECT \* FROM get\_groups()", custom\_query\_parameters, (reader) =>

{

return new Group(

reader.GetInt32(0), reader.GetInt32(1), reader.GetString(2),

reader.GetString(3), reader.GetInt32(4)

);

});

}

public static List<Student> Students(string custom\_query\_parameters)

{

return Get("SELECT \* FROM get\_students()", custom\_query\_parameters, (reader) =>

{

return new Student(

reader.GetInt32(0), reader.GetInt32(1), reader.GetString(2),

reader.GetString(3), reader.GetString(4), reader.GetInt32(5),

reader.GetString(6), reader.GetInt32(7)

);

});

}

public static List<User> Users(string custom\_query\_parameters)

{

return Get("SELECT \* FROM get\_users()", custom\_query\_parameters, (reader) =>

{

return new User(reader.GetInt32(0), reader.GetBoolean(1), reader.GetInt32(2), reader.GetString(3));

});

}

public static List<Theme> Themes(string custom\_query\_parameters)

{

return Get("SELECT \* FROM get\_themes()", custom\_query\_parameters, (reader) =>

{

return new Theme(

reader.GetInt32(0), reader.GetInt32(1), reader.GetString(2),

reader.GetString(3), reader.GetInt32(4), reader.GetInt32(5),

reader.GetInt32(6)

);

});

}

public static List<StageName> StageNames(string custom\_query\_parameters)

{

return Get("SELECT \* FROM get\_stage\_names()", custom\_query\_parameters, (reader) =>

{

return new StageName(reader.GetInt32(0), reader.GetString(1));

});

}

public static List<Stage> Stages(string custom\_query\_parameters)

{

return Get("SELECT \* FROM get\_stages()", custom\_query\_parameters, (reader) =>

{

return new Stage(

reader.GetInt32(0), reader.GetInt32(1), reader.GetString(2),

reader.GetInt32(3), reader.GetInt32(4), reader.GetString(5),

reader.GetInt32(6), reader.GetDateTime(7), reader.GetDateTime(8)

);

});

}

public static bool PasswordIsValid(int id, string pass)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT \* FROM check\_password(:\_id, :\_pass);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_id", id);

command.Parameters.AddWithValue("\_pass", pass);

command.Prepare();

return (int)command.ExecuteScalar() == 0;

}

}

catch (Exception)

{

return false;

}

finally

{

conn.Close();

}

}

}

}

}

**DataBaseAdd.cs**

using System;

using Npgsql;

namespace CourseProject.Classes

{

public static class DataBaseAdd

{

public static string TeacherType(string type\_name)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT add\_teacher\_type(:\_type\_name);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_type\_name", type\_name);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Teacher(int type\_id, string teacher\_name, string address, string phone, string pass)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT add\_teacher(:\_type\_id, :\_teacher\_name, :\_address, :\_phone, :\_pass);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_type\_name", type\_id);

command.Parameters.AddWithValue("\_teacher\_name", teacher\_name);

command.Parameters.AddWithValue("\_address", address);

command.Parameters.AddWithValue("\_phone", phone);

command.Parameters.AddWithValue("\_pass", pass);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Group(int teacher\_id, string group\_name, int year\_started)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT add\_group(:\_teacher\_id, :\_group\_name, :\_year\_started);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_teacher\_id", teacher\_id);

command.Parameters.AddWithValue("\_group\_name", group\_name);

command.Parameters.AddWithValue("\_year\_started", year\_started);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Student(int group\_id, string student\_name, string phone, string pass)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT add\_student(:\_group\_id, :\_student\_name, :\_phone, :\_pass);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_group\_id", group\_id);

command.Parameters.AddWithValue("\_student\_name", student\_name);

command.Parameters.AddWithValue("\_phone", phone);

command.Parameters.AddWithValue("\_pass", pass);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Theme(int student\_id, string theme\_name, int main\_teacher\_id, int econ\_teacher\_id, int safe\_teacher\_id)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT add\_theme(:\_student\_id, :\_theme\_name, :\_main\_teacher\_id, :\_econ\_teacher\_id, :\_safe\_teacher\_id);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_student\_id", student\_id);

command.Parameters.AddWithValue("\_theme\_name", theme\_name);

command.Parameters.AddWithValue("\_main\_teacher\_id", main\_teacher\_id);

command.Parameters.AddWithValue("\_econ\_teacher\_id", econ\_teacher\_id);

command.Parameters.AddWithValue("\_safe\_teacher\_id", safe\_teacher\_id);

command.Prepare();

int ret = (int)command.ExecuteScalar();

if (ret < 0)

return "Превышено количество тем на преподавателя!";

else

return ret.ToString();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

}

public static string StageName(string stage\_name)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT add\_stage\_name(:\_stage\_name);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_stage\_name", stage\_name);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Stage(int stage\_name\_id, int theme\_id, int teacher\_id, int percentage, DateTime date\_started, DateTime date\_ended)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT add\_stage(:\_stage\_name\_id, :\_theme\_id, :\_teacher\_id, :\_percentage, :\_date\_started, :\_date\_ended);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_stage\_name\_id", stage\_name\_id);

command.Parameters.AddWithValue("\_theme\_id", theme\_id);

command.Parameters.AddWithValue("\_teacher\_id", teacher\_id);

command.Parameters.AddWithValue("\_percentage", percentage);

command.Parameters.AddWithValue("\_date\_started", date\_started.ToString(@"yyyy-MM-dd"));

command.Parameters.AddWithValue("\_date\_ended", date\_ended.ToString(@"yyyy-MM-dd"));

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

}

}

**DataBaseDelete.cs**

using System;

using Npgsql;

namespace CourseProject.Classes

{

public static class DataBaseDelete

{

public static string Delete(int id, string cmd)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT " + cmd + "(:\_id);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue(":\_id", id);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string TeacherType(int id)

{

return Delete(id, "delete\_teacher\_type");

}

public static string Teacher(int id)

{

return Delete(id, "delete\_teacher");

}

public static string Group(int id)

{

return Delete(id, "delete\_group");

}

public static string Student(int id)

{

return Delete(id, "delete\_student");

}

public static string Theme(int id)

{

return Delete(id, "delete\_theme");

}

public static string StageName(int id)

{

return Delete(id, "delete\_stage\_name");

}

public static string Stage(int id)

{

return Delete(id, "delete\_stage");

}

}

}

**DataBaseUpdate.cs**

using System;

using Npgsql;

namespace CourseProject.Classes

{

public static class DataBaseUpdate

{

public static string TeacherType(int teacher\_type\_id, string type\_name)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT update\_teacher\_type(:\_teacher\_type\_id, :\_type\_name);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_teacher\_type\_id", teacher\_type\_id);

command.Parameters.AddWithValue("\_type\_name", type\_name);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Teacher(int teacher\_id, int type\_id, string teacher\_name, string address, string phone, string pass)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT update\_teacher(:\_teacher\_id, :\_type\_id, :\_teacher\_name, :\_address, :\_phone, :\_pass);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_teacher\_id", teacher\_id);

command.Parameters.AddWithValue("\_type\_name", type\_id);

command.Parameters.AddWithValue("\_teacher\_name", teacher\_name);

command.Parameters.AddWithValue("\_address", address);

command.Parameters.AddWithValue("\_phone", phone);

command.Parameters.AddWithValue("\_pass", pass);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Group(int group\_id, int teacher\_id, string group\_name, int year\_started)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT update\_group(:\_group\_id, :\_teacher\_id, :\_group\_name, :\_year\_started);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_group\_id", group\_id);

command.Parameters.AddWithValue("\_teacher\_id", teacher\_id);

command.Parameters.AddWithValue("\_group\_name", group\_name);

command.Parameters.AddWithValue("\_year\_started", year\_started);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Student(int student\_id, int group\_id, string student\_name, string phone, string pass)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT update\_student(:\_student\_id, :\_group\_id, :\_student\_name, :\_phone, :\_pass);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_student\_id", student\_id);

command.Parameters.AddWithValue("\_group\_id", group\_id);

command.Parameters.AddWithValue("\_student\_name", student\_name);

command.Parameters.AddWithValue("\_phone", phone);

command.Parameters.AddWithValue("\_pass", pass);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Theme(int theme\_id, int student\_id, string theme\_name, int main\_teacher\_id, int econ\_teacher\_id, int safe\_teacher\_id)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT update\_theme(:\_theme\_id, :\_student\_id, :\_theme\_name, :\_main\_teacher\_id, :\_econ\_teacher\_id, :\_safe\_teacher\_id);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_theme\_id", theme\_id);

command.Parameters.AddWithValue("\_student\_id", student\_id);

command.Parameters.AddWithValue("\_theme\_name", theme\_name);

command.Parameters.AddWithValue("\_main\_teacher\_id", main\_teacher\_id);

command.Parameters.AddWithValue("\_econ\_teacher\_id", econ\_teacher\_id);

command.Parameters.AddWithValue("\_safe\_teacher\_id", safe\_teacher\_id);

command.Prepare();

if ((int)command.ExecuteScalar() < 0)

return "Превышено количество тем на учителя!";

else

return "";

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

}

public static string StageName(int stage\_name\_id, string stage\_name)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT update\_stage\_name(:\_stage\_name\_id, :\_stage\_name);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_stage\_name\_id", stage\_name\_id);

command.Parameters.AddWithValue("\_stage\_name", stage\_name);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string Stage(int stage\_id, int stage\_name\_id, int theme\_id, int teacher\_id, int percentage, DateTime date\_started, DateTime date\_ended)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT update\_stage(:\_stage\_id, :\_stage\_name\_id, :\_theme\_id, :\_teacher\_id, :\_percentage, :\_date\_started, :\_date\_ended);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_stage\_id", stage\_id);

command.Parameters.AddWithValue("\_stage\_name\_id", stage\_name\_id);

command.Parameters.AddWithValue("\_theme\_id", theme\_id);

command.Parameters.AddWithValue("\_teacher\_id", teacher\_id);

command.Parameters.AddWithValue("\_percentage", percentage);

command.Parameters.AddWithValue("\_date\_started", date\_started.ToString(@"yyyy-MM-dd"));

command.Parameters.AddWithValue("\_date\_ended", date\_ended.ToString(@"yyyy-MM-dd"));

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string StagePercentageOnly(int stage\_id, int percentage)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT update\_stage\_only\_percentage(:\_stage\_id, :\_percentage);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_stage\_id", stage\_id);

command.Parameters.AddWithValue("\_percentage", percentage);

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

public static string StageTeacherDates(int stage\_id, int teacher\_id, DateTime started\_date, DateTime ended\_date)

{

using (NpgsqlConnection conn = new NpgsqlConnection(Settings.conStr.ConnectionString))

{

try

{

conn.Open();

using (NpgsqlCommand command = new NpgsqlCommand("SELECT update\_stage\_teacher\_dates(:\_stage\_id, :\_teacher\_id, :\_started\_date::date, :\_ended\_date::date);", conn))

{

command.Parameters.Clear();

command.Parameters.AddWithValue("\_stage\_id", stage\_id);

command.Parameters.AddWithValue("\_teacher\_id", teacher\_id);

command.Parameters.AddWithValue("\_started\_date", started\_date.ToString(@"yyyy-MM-dd"));

command.Parameters.AddWithValue("\_ended\_date", ended\_date.ToString(@"yyyy-MM-dd"));

command.Prepare();

command.ExecuteScalar();

}

}

catch (Exception exc)

{

return exc.Message;

}

finally

{

conn.Close();

}

}

return "";

}

}

}

**MainForm.cs**

using System;

using System.Drawing;

using System.Windows.Forms;

using CourseProject.Classes;

using CourseProject.Forms;

namespace CourseProject

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

Settings.LoadConStr();

}

private void logoutToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Restart();

}

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

Environment.Exit(0);

}

private void teacherProjectsToolStripMenuItem\_Click(object sender, EventArgs e)

{

setActive(sender);

// Show TeacherProjectsControl and hide others

hideAllControls();

teacherProjectsControl.Show();

teacherProjectsControl.ItemsLoad();

}

private void teacherGroupsToolStripMenuItem\_Click(object sender, EventArgs e)

{

setActive(sender);

// Show TeacherGroupsControl and hide others

hideAllControls();

teacherGroupsControl.Show();

teacherGroupsControl.ItemsLoad();

}

private void studentProjectsToolStripMenuItem\_Click(object sender, EventArgs e)

{

setActive(sender);

// Show StudentProjectControl and hide others

hideAllControls();

studentProjectControl.Show();

studentProjectControl.ItemsLoad();

}

private void setActive(object sender)

{

foreach (ToolStripMenuItem item in menuStrip.Items)

item.ForeColor = Color.FromArgb(164, 165, 169);

if (sender != null)

((ToolStripMenuItem)sender).ForeColor = Color.FromArgb(116, 86, 174);

}

public void hideAllControls()

{

studentProjectControl.Hide();

teacherGroupsControl.Hide();

teacherGroupsThemesControl.Hide();

teacherProjectsControl.Hide();

}

private void MainForm\_Load(object sender, EventArgs e)

{

teacherGroupsTimer.Start();

// Initialize database connection

ConnectionForm connection = new ConnectionForm();

if (!Settings.LoadConStr())

{

if (connection.ShowDialog() != DialogResult.OK)

{

Environment.Exit(0);

}

}

// Authorize user

LoginForm login = new LoginForm();

if (login.ShowDialog() != DialogResult.OK)

{

Environment.Exit(0);

}

if (Settings.user.is\_teacher)

{

// TEACHER

// Set tool bar name

this.Text = "Преподаватель: " + Settings.user.name;

// Add menu strip items

ToolStripMenuItem project = new ToolStripMenuItem("projects")

{

Text = "Дипломные проекты",

ShortcutKeys = Keys.Control | Keys.Q

};

project.Click += teacherProjectsToolStripMenuItem\_Click;

menuStrip.Items.Add(project);

ToolStripMenuItem groups = new ToolStripMenuItem("groups")

{

Text = "Группы",

ShortcutKeys = Keys.Control | Keys.W

};

groups.Click += teacherGroupsToolStripMenuItem\_Click;

menuStrip.Items.Add(groups);

// Show TeacherGroupsControl and hide others

hideAllControls();

teacherGroupsControl.Show();

teacherGroupsControl.ItemsLoad();

// Activate menu strip button

setActive(menuStrip.Items[1]);

}

else

{

// STUDENT

// Set tool bar name

this.Text = "Учащийся: " + Settings.user.name;

// Add menu strip items

ToolStripMenuItem project = new ToolStripMenuItem("projects")

{

Text = "Мой дипломный проект",

ShortcutKeys = Keys.Control | Keys.Q

};

project.Click += studentProjectsToolStripMenuItem\_Click;

menuStrip.Items.Add(project);

// Show StudentProjectControl and hide others

hideAllControls();

studentProjectControl.Show();

studentProjectControl.ItemsLoad();

// Activate menu strip button

setActive(menuStrip.Items[0]);

}

// Add additional menu strip items

ToolStripMenuItem logout = new ToolStripMenuItem("logout")

{

Text = "Выход",

ShortcutKeys = Keys.Control | Keys.R

};

logout.Click += logoutToolStripMenuItem\_Click;

menuStrip.Items.Add(logout);

ToolStripMenuItem exit = new ToolStripMenuItem("exit")

{

Text = "Завершить работу",

ShortcutKeys = Keys.Control | Keys.T

};

Click += exitToolStripMenuItem\_Click;

menuStrip.Items.Add(exit);

menuStrip.ForeColor = Color.FromArgb(164, 165, 169);

}

private void teacherGroupsTimer\_Tick(object sender, EventArgs e)

{

if (teacherGroupsControl.studentId > 0)

{

hideAllControls();

teacherGroupsThemesControl.Show();

teacherGroupsThemesControl.ItemsLoad(teacherGroupsControl.studentId, teacherGroupsControl.studentName, teacherGroupsControl.themeId);

teacherGroupsControl.studentId = 0;

teacherGroupsControl.studentName = null;

teacherGroupsControl.themeId = 0;

setActive(null);

}

if (teacherGroupsThemesControl.exit)

{

hideAllControls();

teacherGroupsControl.Show();

teacherGroupsControl.ItemsLoad();

teacherGroupsThemesControl.exit = false;

}

}

}

}

**ConnectionForm.cs**

using System;

using System.Windows.Forms;

using Npgsql;

using CourseProject.Classes;

namespace CourseProject.Forms

{

public partial class ConnectionForm : Form

{

public ConnectionForm()

{

InitializeComponent();

}

private void showPassCheck\_CheckedChanged(object sender, EventArgs e)

{

if (!showPassCheck.Checked)

passwordBox.PasswordChar = '\*';

else

passwordBox.PasswordChar = char.MinValue;

}

private void cancelButton\_Click(object sender, EventArgs e)

{

this.DialogResult = DialogResult.Abort;

}

private void connectButton\_Click(object sender, EventArgs e)

{

int port = 0;

if (!int.TryParse(portBox.Text, out port))

{

MessageBox.Show("Порт может содержать только цифры!", "Ошибка подключения");

}

else

{

NpgsqlConnectionStringBuilder temp = new NpgsqlConnectionStringBuilder();

temp.Host = hostBox.Text;

temp.Port = port;

temp.Username = usernameBox.Text;

temp.Password = passwordBox.Text;

temp.Database = databaseBox.Text;

if (Settings.CheckDatabaseConnection(temp))

{

Settings.conStr = temp;

Settings.SaveConStr();

MessageBox.Show("Успешно подключено!", "Успешное подключение");

this.DialogResult = DialogResult.OK;

}

else

{

MessageBox.Show("Невозможно подключиться к базе данных!\nУбедитесь, что данные введены верно и база данных активна.", "Ошибка подключения");

}

}

}

}

}

**LoginForm.cs**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using CourseProject.Classes;

namespace CourseProject.Forms

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

}

List<User> users = new List<User>();

private void LoginForm\_Load(object sender, EventArgs e)

{

users = DataBaseGet.Users("");

foreach (User val in users)

{

usernameBox.Items.Add(val);

}

}

private void clearButton\_Click(object sender, EventArgs e)

{

this.DialogResult = DialogResult.Abort;

}

private void showPassCheck\_CheckedChanged(object sender, EventArgs e)

{

if (!showPassCheck.Checked)

passwordBox.PasswordChar = '\*';

else

passwordBox.PasswordChar = char.MinValue;

}

private void loginButton\_Click(object sender, EventArgs e)

{

if (usernameBox.SelectedIndex < 0)

MessageBox.Show("Логин либо пароль не верны", "Ошибка авторизации");

else

{

if (DataBaseGet.PasswordIsValid(((User)usernameBox.SelectedItem).user\_id, passwordBox.Text))

{

Settings.user = (User)usernameBox.SelectedItem;

DialogResult = DialogResult.OK;

}

else

{

MessageBox.Show("Логин либо пароль не верны", "Ошибка авторизации");

MessageBox.Show(((User)usernameBox.SelectedItem).id.ToString(), passwordBox.Text);

}

}

}

}

}

**StudentProjectControl.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using CourseProject.Classes;

namespace CourseProject.Forms

{

public partial class StudentProjectControl : UserControl

{

List<Theme> lastTheme = new List<Theme>();

List<Stage> list = new List<Stage>();

private void lst\_MeasureItem(object sender, MeasureItemEventArgs e)

{

e.ItemHeight = (int)e.Graphics.MeasureString(stagesList.Items[e.Index].ToString(), stagesList.Font, stagesList.Width).Height;

}

private void lst\_DrawItem(object sender, DrawItemEventArgs e)

{

e.DrawBackground();

e.DrawFocusRectangle();

e.Graphics.DrawString(stagesList.Items[e.Index].ToString(), e.Font, new SolidBrush(e.ForeColor), e.Bounds);

}

public StudentProjectControl()

{

InitializeComponent();

stagesList.DrawMode = DrawMode.OwnerDrawVariable;

stagesList.MeasureItem += lst\_MeasureItem;

stagesList.DrawItem += lst\_DrawItem;

}

public void ItemsLoad()

{

lastTheme = DataBaseGet.Themes("WHERE student\_id = " + Settings.user.id + " ORDER BY theme\_id DESC LIMIT 1");

if (lastTheme.Count == 0)

{

themeNameLabel.Text = "У вас нету дипломных проектов.\nЕсли у вас должен быть дипломный проект, то обратитесь к преподавателю.";

stagesList.Hide();

checkedLabel.Hide();

datesLabel.Hide();

percentageLabel.Hide();

}

else

{

themeNameLabel.Text = "Тема дипломного проекта: " + lastTheme[0].theme\_name;

lastTheme[0].AddTeacherNames(

DataBaseGet.Teachers("WHERE teacher\_id = " + lastTheme[0].main\_teacher\_id)[0].teacher\_name,

DataBaseGet.Teachers("WHERE teacher\_id = " + lastTheme[0].econ\_teacher\_id)[0].teacher\_name,

DataBaseGet.Teachers("WHERE teacher\_id = " + lastTheme[0].safe\_teacher\_id)[0].teacher\_name

);

mainTeacherName.Text = "Руководитель по основному разделу: " + lastTheme[0].main\_teacher\_name;

econTeacherName.Text = "Руководитель по экономическому разделу: " + lastTheme[0].econ\_teacher\_name;

safeTeacherName.Text = "Руководитель по разделу охраны труда: " + lastTheme[0].safe\_teacher\_name;

list = DataBaseGet.Stages("WHERE theme\_id = " + lastTheme[0].theme\_id);

stagesList.Items.Clear();

for (int i = 0; i < list.Count; i++)

{

stagesList.Items.Add((i + 1) + ". " + list[i].stage\_name);

}

stagesList.SelectedIndex = 0;

}

}

private void stagesList\_SelectedIndexChanged(object sender, EventArgs e)

{

if (list[stagesList.SelectedIndex].percentage > 100)

{

checkedLabel.Text = "Статус (Выставляется преподавателем): Выполнен";

percentageLabel.Hide();

}

else

{

checkedLabel.Text = "Статус (Выставляется преподавателем): Не выполнен";

percentageLabel.Show();

percentageLabel.Text = "Выполнено: " + list[stagesList.SelectedIndex].percentage + "%";

}

stageTeacherLabel.Text = "Проверяющий преподаватель: " + list[stagesList.SelectedIndex].teacher\_name;

datesLabel.Text =

"Дата начала этапа: " + list[stagesList.SelectedIndex].date\_started.ToString("dd.MM.yyyy") +

" Дата окончания этапа: " + list[stagesList.SelectedIndex].date\_ended.ToString("dd.MM.yyyy");

}

}

}

**TeacherGroupsControl.cs**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using CourseProject.Classes;

namespace CourseProject.Forms

{

public partial class TeacherGroupsControl : UserControl

{

List<Group> groups;

List<Student> students;

public int studentId;

public string studentName;

public int themeId;

public TeacherGroupsControl()

{

InitializeComponent();

}

public void ItemsLoad()

{

groups = DataBaseGet.Groups("WHERE teacher\_id = " + Settings.user.id);

groupsListBox.Items.Clear();

if (groups.Count == 0)

{

groupLabel.Text = "Вы не являетесь куратором ни одной группы.\nЕсли вы являетесь куратором и группы нету, то обратитесь к администраторам.";

groupsListBox.Hide();

groupsGrid.Hide();

}

else

{

foreach (Group val in groups)

{

groupsListBox.Items.Add(val.group\_name);

}

groupsGrid.Rows.Clear();

groupsListBox.SelectedIndex = 0;

}

}

private void groupsListBox\_SelectedIndexChanged(object sender, EventArgs e)

{

if (students != null)

students.Clear();

groupsGrid.Rows.Clear();

students = DataBaseGet.Students("WHERE group\_id = " + groups[groupsListBox.SelectedIndex].group\_id);

if (students != null)

{

foreach (Student val in students)

{

groupsGrid.Rows.Add(val.student\_id, val.student\_name, val.phone, val.theme\_id, val.theme\_name, val.percentage);

}

}

}

private void groupsGrid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

this.studentId = Convert.ToInt32(groupsGrid.Rows[e.RowIndex].Cells[0].Value);

this.studentName = groups[groupsListBox.SelectedIndex].group\_name + " " + groupsGrid.Rows[e.RowIndex].Cells[1].Value.ToString();

this.themeId = Convert.ToInt32(groupsGrid.Rows[e.RowIndex].Cells[3].Value);

}

}

}

**TeacherGroupsThemesControl.cs**

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Threading;

using System.Windows.Forms;

using CourseProject.Classes;

namespace CourseProject.Forms

{

public partial class TeacherGroupsThemesControl : UserControl

{

public bool exit;

public int studentId;

public string studentName;

public int themeId;

public Theme currentTheme;

List<Teacher> teachers;

List<Stage> stages;

public TeacherGroupsThemesControl()

{

InitializeComponent();

}

public void ItemsLoad(int studentId, string studentName, int themeId)

{

this.studentId = studentId;

this.studentName = studentName;

this.themeId = themeId;

currentTheme = null;

teachers = null;

stages = null;

themeBox.Text = "";

mainTeacherBox.Text = "";

econTeacherBox.Text = "";

safeTeacherBox.Text = "";

studentLabel.Text = "Учащийся: " + this.studentName;

mainTeacherBox.Items.Clear();

econTeacherBox.Items.Clear();

safeTeacherBox.Items.Clear();

teachers = DataBaseGet.Teachers("");

foreach (Teacher val in teachers)

{

switch (val.type\_id)

{

case 1: mainTeacherBox.Items.Add(val); break;

case 2: econTeacherBox.Items.Add(val); break;

case 3: safeTeacherBox.Items.Add(val); break;

}

}

if (themeId < 1)

{

addButton.Show();

deleteButton.Hide();

updateButton.Hide();

HideStages();

}

else

{

addButton.Hide();

deleteButton.Show();

updateButton.Show();

ShowStages();

currentTheme = DataBaseGet.Themes("WHERE theme\_id = " + themeId)[0];

themeBox.Text = currentTheme.theme\_name;

for (int i = 0; i < mainTeacherBox.Items.Count; i++)

{

if (((Teacher)mainTeacherBox.Items[i]).teacher\_id == currentTheme.main\_teacher\_id)

{

mainTeacherBox.SelectedIndex = i;

break;

}

}

for (int i = 0; i < econTeacherBox.Items.Count; i++)

{

if (((Teacher)econTeacherBox.Items[i]).teacher\_id == currentTheme.econ\_teacher\_id)

{

econTeacherBox.SelectedIndex = i;

break;

}

}

for (int i = 0; i < safeTeacherBox.Items.Count; i++)

{

if (((Teacher)safeTeacherBox.Items[i]).teacher\_id == currentTheme.safe\_teacher\_id)

{

safeTeacherBox.SelectedIndex = i;

break;

}

}

}

}

public void UpdateGrid()

{

stagesGrid.Rows.Clear();

foreach (Stage val in stages)

{

stagesGrid.Rows.Add(val.stage\_id, val.stage\_name, val.teacher\_id, val.teacher\_name, val.date\_started.ToString(@"dd.MM.yyyy"), val.date\_ended.ToString(@"dd.MM.yyyy"));

}

}

public void ShowStages()

{

stagesGrid.Show();

stages = DataBaseGet.Stages("WHERE theme\_id = " + themeId);

UpdateGrid();

stageTeacherBox.Show();

stageTeacherBox.Items.Clear();

teachers = DataBaseGet.Teachers("");

foreach (Teacher val in teachers)

{

stageTeacherBox.Items.Add(val);

}

int teacher\_id = Convert.ToInt32(stagesGrid.Rows[0].Cells[2].Value);

for (int i = 0; i < stageTeacherBox.Items.Count; i++)

{

if (((Teacher)stageTeacherBox.Items[i]).teacher\_id == teacher\_id)

stageTeacherBox.SelectedIndex = i;

}

stageTeacherLabel.Show();

stageDateEndedLabel.Show();

stageDateEndedPicker.Show();

stageDateStartedLabel.Show();

stageDateStartedPicker.Show();

stageEditButton.Show();

}

public void HideStages()

{

stagesGrid.Hide();

stageTeacherBox.Hide();

stageTeacherLabel.Hide();

stageDateEndedLabel.Hide();

stageDateEndedPicker.Hide();

stageDateStartedLabel.Hide();

stageDateStartedPicker.Hide();

stageEditButton.Hide();

}

private void updateButton\_Click(object sender, EventArgs e)

{

if (mainTeacherBox.SelectedIndex < 0 || econTeacherBox.SelectedIndex < 0 || safeTeacherBox.SelectedIndex < 0)

{

MessageBox.Show("Введены не верные данные", "Ошибка");

}

else

{

int main\_teacher\_id = ((Teacher)mainTeacherBox.SelectedItem).teacher\_id;

int econ\_teacher\_id = ((Teacher)econTeacherBox.SelectedItem).teacher\_id;

int safe\_teacher\_id = ((Teacher)safeTeacherBox.SelectedItem).teacher\_id;

string err = DataBaseUpdate.Theme(themeId, studentId, themeBox.Text, main\_teacher\_id, econ\_teacher\_id, safe\_teacher\_id);

if (!string.IsNullOrEmpty(err))

{

MessageBox.Show(err, "Ошибка");

}

}

}

private void addButton\_Click(object sender, EventArgs e)

{

if (mainTeacherBox.SelectedIndex < 0 || econTeacherBox.SelectedIndex < 0 || safeTeacherBox.SelectedIndex < 0)

{

MessageBox.Show("Введены не верные данные", "Ошибка");

}

else

{

int main\_teacher\_id = ((Teacher)mainTeacherBox.SelectedItem).teacher\_id;

int econ\_teacher\_id = ((Teacher)econTeacherBox.SelectedItem).teacher\_id;

int safe\_teacher\_id = ((Teacher)safeTeacherBox.SelectedItem).teacher\_id;

string err = DataBaseAdd.Theme(studentId, themeBox.Text, main\_teacher\_id, econ\_teacher\_id, safe\_teacher\_id);

if (!string.IsNullOrEmpty(err))

{

if (!int.TryParse(err, out themeId))

MessageBox.Show(err, "Ошибка");

else

{

Thread.Sleep(1000);

ShowStages();

addButton.Hide();

deleteButton.Show();

updateButton.Show();

}

}

else

{

Thread.Sleep(1000);

ShowStages();

addButton.Hide();

deleteButton.Show();

updateButton.Show();

}

}

}

private void deleteButton\_Click(object sender, EventArgs e)

{

DataBaseDelete.Theme(themeId);

exit = true;

}

private void stageEditButton\_Click(object sender, EventArgs e)

{

int stage\_id = Convert.ToInt32(stagesGrid.SelectedRows[0].Cells[0].Value);

int teacher\_id = ((Teacher)stageTeacherBox.SelectedItem).teacher\_id;

DataBaseUpdate.StageTeacherDates(stage\_id, teacher\_id, stageDateStartedPicker.Value, stageDateEndedPicker.Value);

ShowStages();

}

private void stagesGrid\_SelectionChanged(object sender, EventArgs e)

{

if (stagesGrid.SelectedRows.Count > 0)

{

int teacher\_id = Convert.ToInt32(stagesGrid.SelectedRows[0].Cells[2].Value);

for (int i = 0; i < stageTeacherBox.Items.Count; i++)

{

if (((Teacher)stageTeacherBox.Items[i]).teacher\_id == teacher\_id)

stageTeacherBox.SelectedIndex = i;

}

stageDateStartedPicker.Value = DateTime.ParseExact(

stagesGrid.SelectedRows[0].Cells[4].Value.ToString(),

"dd.MM.yyyy", CultureInfo.InvariantCulture

);

stageDateEndedPicker.Value = DateTime.ParseExact(

stagesGrid.SelectedRows[0].Cells[5].Value.ToString(),

"dd.MM.yyyy", CultureInfo.InvariantCulture

);

}

}

}

}

**TeacherProjectsControl.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

using CourseProject.Classes;

namespace CourseProject.Forms

{

public partial class TeacherProjectsControl : UserControl

{

private void stagesList\_MeasureItem(object sender, MeasureItemEventArgs e)

{

e.ItemHeight = (int)e.Graphics.MeasureString(stagesList.Items[e.Index].ToString(), stagesList.Font, stagesList.Width).Height;

}

private void stagesList\_DrawItem(object sender, DrawItemEventArgs e)

{

e.DrawBackground();

e.DrawFocusRectangle();

e.Graphics.DrawString(stagesList.Items[e.Index].ToString(), e.Font, new SolidBrush(e.ForeColor), e.Bounds);

}

private void studentsList\_MeasureItem(object sender, MeasureItemEventArgs e)

{

e.ItemHeight = (int)e.Graphics.MeasureString(studentsList.Items[e.Index].ToString(), studentsList.Font, studentsList.Width).Height;

}

private void studentsList\_DrawItem(object sender, DrawItemEventArgs e)

{

e.DrawBackground();

e.DrawFocusRectangle();

e.Graphics.DrawString(studentsList.Items[e.Index].ToString(), e.Font, new SolidBrush(e.ForeColor), e.Bounds);

}

public TeacherProjectsControl()

{

InitializeComponent();

stagesList.DrawMode = DrawMode.OwnerDrawVariable;

stagesList.MeasureItem += stagesList\_MeasureItem;

stagesList.DrawItem += stagesList\_DrawItem;

studentsList.DrawMode = DrawMode.OwnerDrawVariable;

studentsList.MeasureItem += studentsList\_MeasureItem;

studentsList.DrawItem += studentsList\_DrawItem;

}

List<Student> students = new List<Student>();

List<Stage> stages = new List<Stage>();

List<Theme> lastTheme = new List<Theme>();

int tick = -1;

public void ItemsLoad()

{

updateTimer.Start();

studentsList.Hide();

stagesList.Hide();

studentsList.Items.Clear();

stagesList.Items.Clear();

students.Clear();

stages.Clear();

lastTheme.Clear();

students = DataBaseGet.Students("WHERE theme\_id IN (SELECT themes.theme\_id FROM themes WHERE themes.main\_teacher\_id = " + Settings.user.id + " OR themes.econ\_teacher\_id = " + Settings.user.id + " OR themes.safe\_teacher\_id = " + Settings.user.id + ")");

if (students.Count > 0)

{

foreach (Student val in students)

{

studentsList.Items.Add(val.group\_name + " " + val.student\_name);

}

studentsList.Show();

stagesList.Show();

studentsList.SelectedIndex = 0;

}

else

{

themeNameLabel.Text = "Вы не являетесь руководителем ни одного проекта.";

studentsList.Hide();

stagesList.Hide();

checkedLabel.Hide();

datesLabel.Hide();

percentageLabel.Hide();

percentageBox.Hide();

percentageBar.Hide();

percentLabel.Hide();

cancelButton.Hide();

submitButton.Hide();

mainTeacherName.Hide();

econTeacherName.Hide();

safeTeacherName.Hide();

}

}

private void stagesList\_SelectedIndexChanged(object sender, EventArgs e)

{

UpdateStages2();

}

private void studentsList\_SelectedIndexChanged(object sender, EventArgs e)

{

UpdateStages();

}

public void UpdateStages()

{

int old\_selected\_index = stagesList.SelectedIndex == -1 ? 0 : stagesList.SelectedIndex;

lastTheme = DataBaseGet.Themes("WHERE theme\_id = " + students[studentsList.SelectedIndex].theme\_id + " ORDER BY theme\_id DESC LIMIT 1");

if (lastTheme.Count == 0)

{

themeNameLabel.Text = "У данного учащегося нету дипломных проектов.";

stagesList.Hide();

checkedLabel.Hide();

datesLabel.Hide();

percentageLabel.Hide();

percentageBox.Hide();

percentageBar.Hide();

percentLabel.Hide();

cancelButton.Hide();

submitButton.Hide();

mainTeacherName.Hide();

econTeacherName.Hide();

safeTeacherName.Hide();

}

else

{

themeNameLabel.Text = "Тема дипломного проекта: " + lastTheme[0].theme\_name;

stages = DataBaseGet.Stages("WHERE theme\_id = " + lastTheme[0].theme\_id);

stagesList.Items.Clear();

for (int i = 0; i < stages.Count; i++)

{

stagesList.Items.Add((Settings.user.id == stages[i].teacher\_id ? "+ " : "- ") + (i + 1) + ". " + stages[i].stage\_name);

}

lastTheme[0].AddTeacherNames(

DataBaseGet.Teachers("WHERE teacher\_id = " + lastTheme[0].main\_teacher\_id)[0].teacher\_name,

DataBaseGet.Teachers("WHERE teacher\_id = " + lastTheme[0].econ\_teacher\_id)[0].teacher\_name,

DataBaseGet.Teachers("WHERE teacher\_id = " + lastTheme[0].safe\_teacher\_id)[0].teacher\_name

);

mainTeacherName.Text = "Руководитель по основному разделу: " + lastTheme[0].main\_teacher\_name;

econTeacherName.Text = "Руководитель по экономическому разделу: " + lastTheme[0].econ\_teacher\_name;

safeTeacherName.Text = "Руководитель по разделу охраны труда: " + lastTheme[0].safe\_teacher\_name;

stagesList.SelectedIndex = old\_selected\_index;

UpdateStages2();

}

}

public void UpdateStages2()

{

int old\_percentage = stages[stagesList.SelectedIndex].percentage;

if (old\_percentage > 100)

{

checkedLabel.Text = "Статус (Выставляется преподавателем): Выполнен";

percentageLabel.Hide();

percentageLabel.Hide();

percentageBox.Hide();

percentageBar.Hide();

percentLabel.Hide();

if (Settings.user.id == stages[stagesList.SelectedIndex].teacher\_id)

{

submitButton.Hide();

cancelButton.Show();

}

else

{

submitButton.Hide();

cancelButton.Hide();

}

}

else

{

checkedLabel.Text = "Статус (Выставляется преподавателем): Не выполнен";

percentageLabel.Show();

percentageLabel.Show();

percentageBox.Show();

percentageBar.Show();

percentLabel.Show();

if (Settings.user.id == stages[stagesList.SelectedIndex].teacher\_id)

{

submitButton.Show();

cancelButton.Hide();

}

else

{

submitButton.Hide();

cancelButton.Hide();

}

percentageBar.Value = old\_percentage;

percentageBox.Text = old\_percentage.ToString();

}

stageTeacherLabel.Text = "Проверяющий преподаватель: " + stages[stagesList.SelectedIndex].teacher\_name;

datesLabel.Text =

"Дата начала этапа: " + stages[stagesList.SelectedIndex].date\_started.ToString("dd.MM.yyyy") +

" Дата окончания этапа: " + stages[stagesList.SelectedIndex].date\_ended.ToString("dd.MM.yyyy");

}

private void submitButton\_Click(object sender, EventArgs e)

{

int current\_stage\_id = stages[stagesList.SelectedIndex].stage\_id;

int old\_percentage = stages[stagesList.SelectedIndex].percentage;

DataBaseUpdate.StagePercentageOnly(current\_stage\_id, old\_percentage + 200);

UpdateStages();

}

private void cancelButton\_Click(object sender, EventArgs e)

{

int current\_stage\_id = stages[stagesList.SelectedIndex].stage\_id;

int old\_percentage = stages[stagesList.SelectedIndex].percentage;

DataBaseUpdate.StagePercentageOnly(current\_stage\_id, old\_percentage - 200);

UpdateStages();

}

private void percentageBar\_Scroll(object sender, EventArgs e)

{

percentageBox.Text = percentageBar.Value.ToString();

}

private void percentageBox\_TextChanged(object sender, EventArgs e)

{

if (!int.TryParse(percentageBox.Text, out int val) || val > 100 || val < 0)

{

percentageBox.Text = "";

percentageBar.Value = 0;

tick = -1;

}

else

{

int new\_percentage = Convert.ToInt32(percentageBox.Text);

int old\_percentage = stages[stagesList.SelectedIndex].percentage;

percentageBar.Value = new\_percentage;

if (new\_percentage != old\_percentage) tick = 1;

}

}

private void updateTimer\_Tick(object sender, EventArgs e)

{

if (tick >= 0) tick--;

if (tick == 0) updatePercentage();

}

private void updatePercentage()

{

int current\_stage\_id = stages[stagesList.SelectedIndex].stage\_id;

if (stages[stagesList.SelectedIndex].percentage != Convert.ToInt32(percentageBox.Text))

{

DataBaseUpdate.StagePercentageOnly(current\_stage\_id, Convert.ToInt32(percentageBox.Text));

stages = DataBaseGet.Stages("WHERE theme\_id = " + lastTheme[0].theme\_id);

}

}

private void reportButton\_Click(object sender, EventArgs e)

{

if (studentsList.SelectedIndex > -1)

{

Excel.Application oXL;

Excel.\_Workbook oWB;

Excel.\_Worksheet oSheet;

try

{

oXL = new Excel.Application();

oXL.Visible = true;

oWB = oXL.Workbooks.Add();

oSheet = oWB.ActiveSheet as Excel.Worksheet;

oSheet.Cells[1, 1].Value2 = "Дата создания документа - " + DateTime.Now.ToString("dd.MM.yyyy") + "; Преподаватель, создавший документ - " + Settings.user.name;

oSheet.Cells[2, 1].Value2 = themeNameLabel.Text;

oSheet.Cells[3, 1].Value2 = mainTeacherName.Text;

oSheet.Cells[4, 1].Value2 = econTeacherName.Text;

oSheet.Cells[5, 1].Value2 = safeTeacherName.Text;

oSheet.get\_Range("A1", "F1").Merge();

oSheet.get\_Range("A2", "F2").Merge();

oSheet.get\_Range("A3", "F3").Merge();

oSheet.get\_Range("A4", "F4").Merge();

oSheet.get\_Range("A5", "F5").Merge();

oSheet.Cells[6, 1] = "Этап проекта";

oSheet.Cells[6, 2] = "Статус";

oSheet.Cells[6, 3] = "Выполнено";

oSheet.Cells[6, 4] = "Проверяющий преподаватель";

oSheet.Cells[6, 5] = "Дата начала этапа";

oSheet.Cells[6, 6] = "Дата окончания этапа";

oSheet.get\_Range("A1", "F5").Font.Color = Color.FromArgb(164, 165, 169);

oSheet.get\_Range("A2", "F2").Font.Bold = true;

oSheet.get\_Range("A2", "F2").Font.Size = 20;

oSheet.get\_Range("A2", "F2").Font.Color = Color.FromArgb(116, 86, 174);

oSheet.get\_Range("A1", "F1").Font.Italic = true;

oSheet.get\_Range("A6", "F6").Font.Bold = true;

oSheet.get\_Range("A1", "F6").VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;

oSheet.get\_Range("A6", "F6").HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

oSheet.get\_Range("A1", "A1").ColumnWidth = 48;

oSheet.get\_Range("B1", "B1").ColumnWidth = 13;

oSheet.get\_Range("C1", "C1").ColumnWidth = 12;

oSheet.get\_Range("D1", "D1").ColumnWidth = 32;

oSheet.get\_Range("E1", "E1").ColumnWidth = 20;

oSheet.get\_Range("F1", "F1").ColumnWidth = 20;

string[,] list = new string[stages.Count, 6];

for (int i = 0; i < stages.Count; i++)

{

list[i, 0] = stages[i].stage\_name;

list[i, 1] = stages[i].percentage > 100 ? "Выполнен" : "Не выполнен";

list[i, 2] = stages[i].percentage > 100 ? "100 %" : stages[i].percentage.ToString() + " %";

list[i, 3] = stages[i].teacher\_name;

list[i, 4] = stages[i].date\_started.ToString(@"dd.MM.yyyy");

list[i, 5] = stages[i].date\_ended.ToString(@"dd.MM.yyyy");

}

int last\_cell = 6 + list.GetLength(0);

oSheet.get\_Range("A7", "F" + last\_cell).Value2 = list;

oXL.Visible = true;

oXL.UserControl = true;

}

catch (Exception exc)

{

MessageBox.Show("Невозможно создать отчет:\n" + exc.Message, "Ошибка");

}

}

}

}

}