Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Северо-Осетинский государственный университет

имени Коста Левановича Хетагурова»

**КУРСОВАЯ РАБОТА**

«Создание парсеров для загрузки первичных документов на информационный портал факультета»

Выполнил:

студент 3 курса направления

«Прикладная математика и информатика»

Гутнов Сослан Таймуразович

Научный руководитель:

старший преподаватель кафедры

прикладной математики и информатики

Гагиев .

**«Работа допущена к защите»**

Заведующий кафедрой

к. ф-м. н., доцент

Басаева Е.К. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Владикавказ 2023

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ............................................................................................................. 3

ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ................................................................ 6

1.1 Целочисленное программирование...................................................... 6

1.2 Постановка задачи и методы решения............................................... 13

1.3 Пример решения задачи целочисленного программирования ........ 19

ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ................................................................ 26

2.1. Инструкция пользователя................................................................... 26

2.2 Инструкция системного администратора.......................................... 35

2.3 Инструкция программиста ................................................................. 43

ЗАКЛЮЧЕНИЕ ................................................................................................... 74

СПИСОК ЛИТЕРАТУРЫ....................................................................................77

**Введение**

Парсинг - это процесс извлечения данных из структурированных или неструктурированных источников данных, например, веб-страниц, документов PDF или файлов Excel.

Создание парсеров для загрузки первичных документов на информационный портал факультета может помочь автоматизировать процессы загрузки документов и обеспечить более эффективную работу с ними. Например, парсер может извлекать данные из документов и заполнять поля на портале автоматически, что позволит сократить время, затрачиваемое на ручное заполнение данных.

Однако, при создании парсера необходимо учитывать особенности и форматы конкретных документов, которые будут загружаться на портал. Также стоит обратить внимание на безопасность и конфиденциальность данных, которые будут обрабатываться парсером, чтобы избежать возможных утечек информации.

Цель курсовой работы - создать программу с использованием C# и Windows Forms, которая будет парсить документы «Кадровая справка» и «Нагрузки преподавателей» и загружать полученную информацию в базу данных университета СОГУ.

Для достижения цели необходимо выполнить следующие задачи:

* Изучить формат и структуру документов "Кадровая справка" и "Нагрузки" университета СОГУ.
* Создать программное обеспечение, которое автоматически будет извлекать и структурировать информацию из этих документов, а затем загружать ее в базу данных университета.

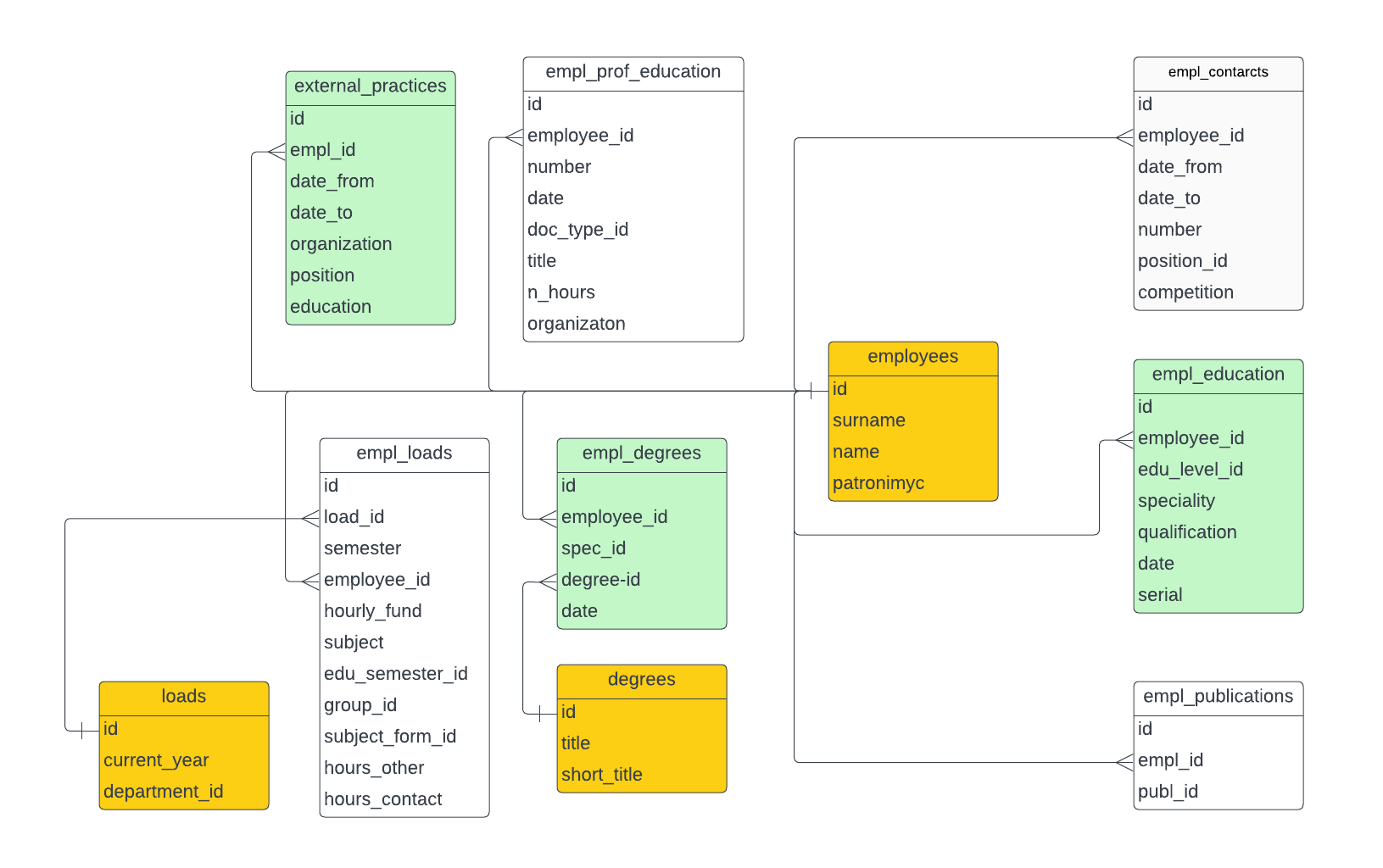
Программа работает в 3 этапа:

1. Выбираем Excel документ, программа выведет данные с этого документа на форму
2. Обработка данных с формы
3. Данные закидываются в базу

**ГЛАВА 1.**

**1.1 База данных**

Ниже представлены таблицы базы данных с которыми проводится работа:



Таблицы отмеченные желтым цветом считаются справочными, зеленые и желтые таблицы заполняются программой, белые не заполняются.

В базе данных отсутствуют связи, поиск информации реализуется через выборку по id.

**1.2. Этапы разработки**

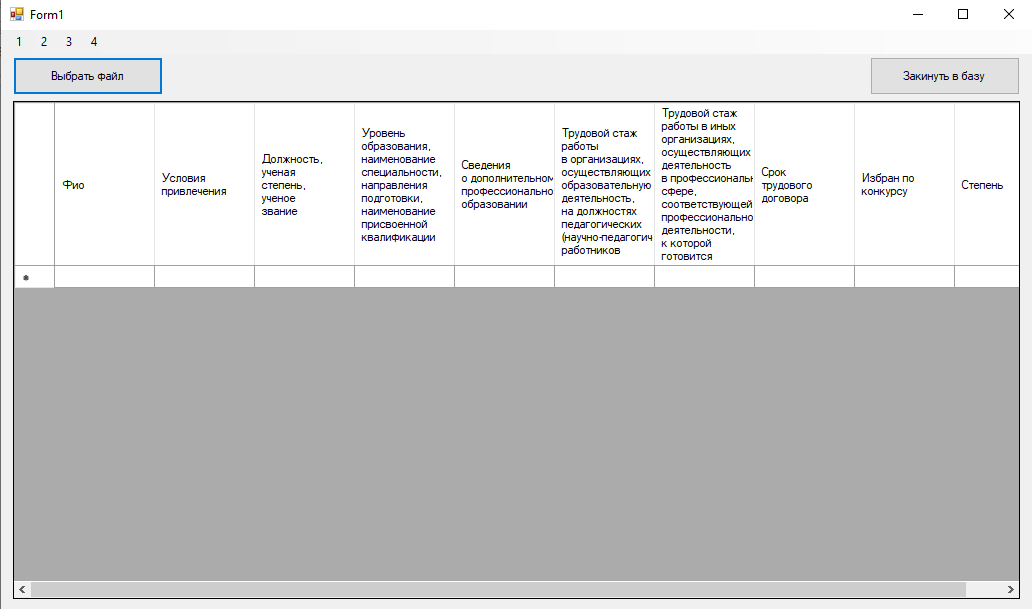
Программа разрабатывалась несколько этапов. Первым делом необходимо было сделать визуальную часть, т.е. форму в которой можно было бы выбрать документ, который будет парситься, проверить данные и закинуть их в базу по нажатию кнопки. Было принято решение не особо концентрироваться на этом этапе т.к. в этом не было необходимости, ведь главная цель это считать данные, а времени было немного. Далее я разбирался с документами, какие листы, какие в них столбцы мне нужно парсить, какие типы ошибок имеются и как их исправить. Потом составлялись запросы в БД с уже обработанными и очищенными данными.

Важно отметить что на данный момент приложение работает только с документом «Кадровая справка».

**ГЛАВА 2**

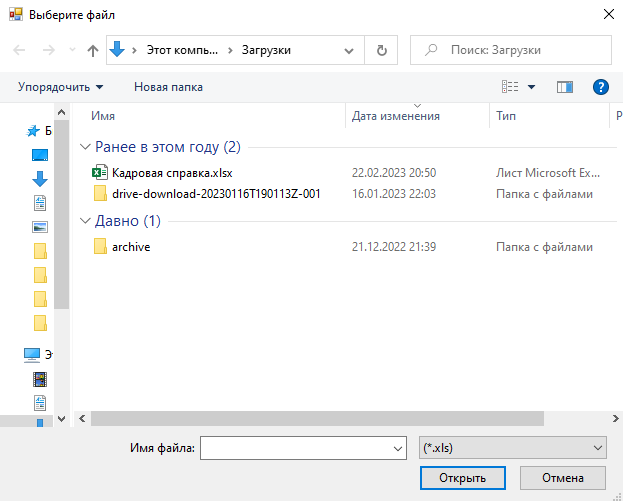
В данной главе подробнее рассмотрим этапы разработки и непосредственно сам код.

**2.1. Руководство пользователя**

****

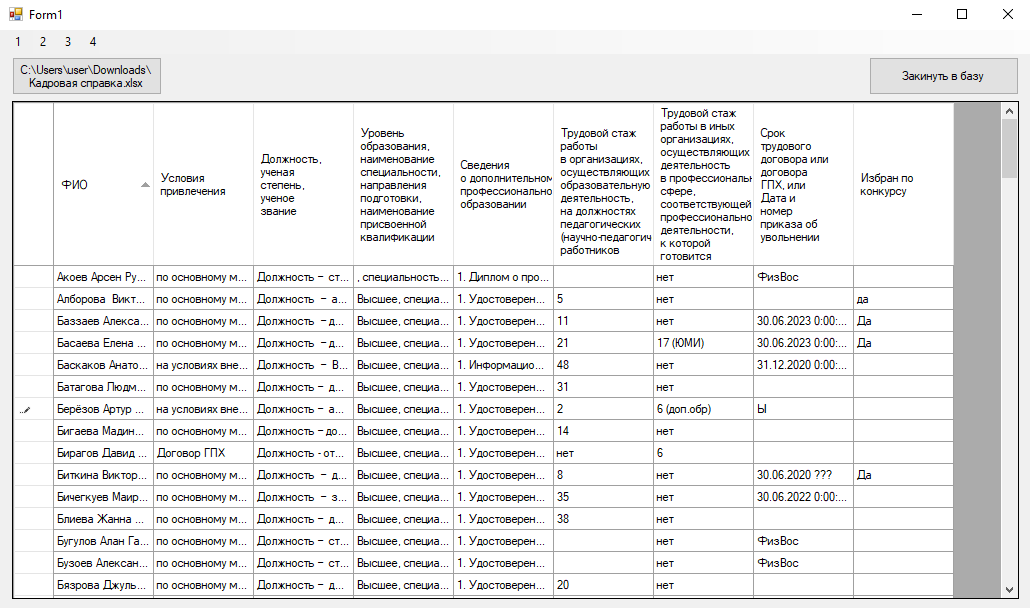
Так визуально выглядит приложение. Т.к. пользователями данного приложения 100% будут являться люди работающие с информационным порталом факультета красоту наводить не считается необходимым. Главная красота в результате работы приложения!

Далее необходимо выбрать документ, это можно сделать нажав кнопку «Выбрать файл», откроется проводник Windows где можно найти и выбрать свой файл.

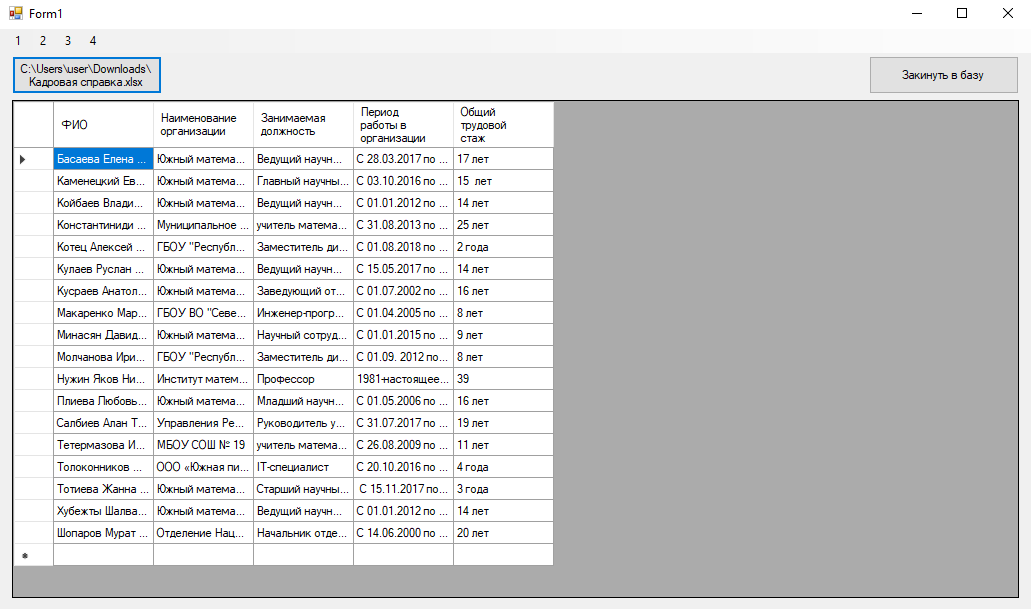


Выбираем файл..

Далее программа считает данные с документа и выведет их в таблицу на форме, где мы можем их просмотреть и проверить



Нажимая на цифры слева сверху можно переключать листы документа.



Проверяем правильно ли все считалось, также можно подредактировать или вписать новые данные. (но лучше так не делать)

Далее нажимаем кнопку «Закинуть в базу» и программа после обработки данных с формы отправит их в базу.

**2.2. Руководство системного администратора**

Для использования приложения необходимо наличие на компьютере SQL server 2019. Системные требования к программному продукту: ОС Windows 7 и выше, .Net Framework 4.7.2 и выше

**2.3. Руководство разработчика**

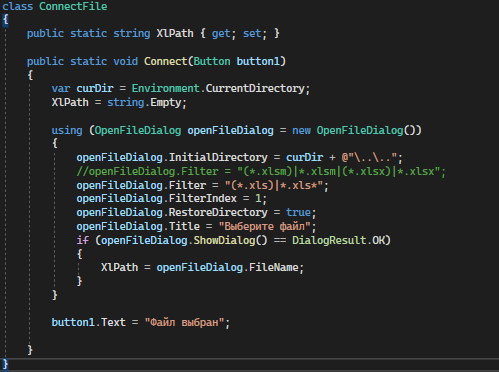
**2.3.1. Excel файл**

Так как программа начинается с выбора файла, начнем оттуда.

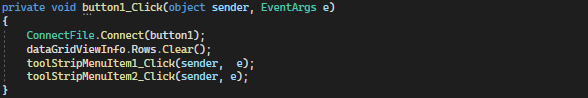
Для выбора файла был создан отдельный класс - ConnectFile,

Есть публичное статическое поле XlPath чтобы при необходимости в другой части программы можно было получить путь к файлу.

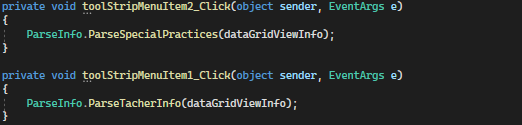
Основной код находится в функции Connect которая открывает диалоговое окно где пользователь может выбрать свой файл. Она принимает на вход кнопку чтобы после выбора поменять текст на ней.



Данная функция вызывается при нажатии на кнопку «Выбрать файл»,



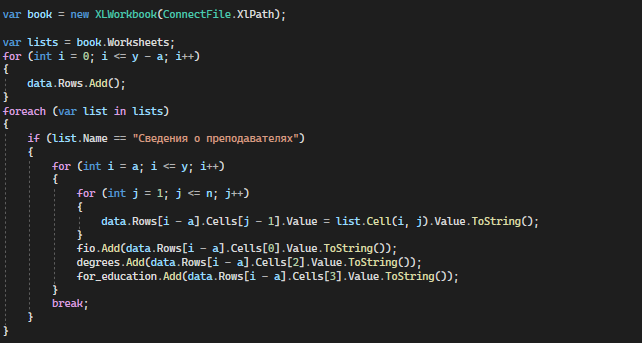
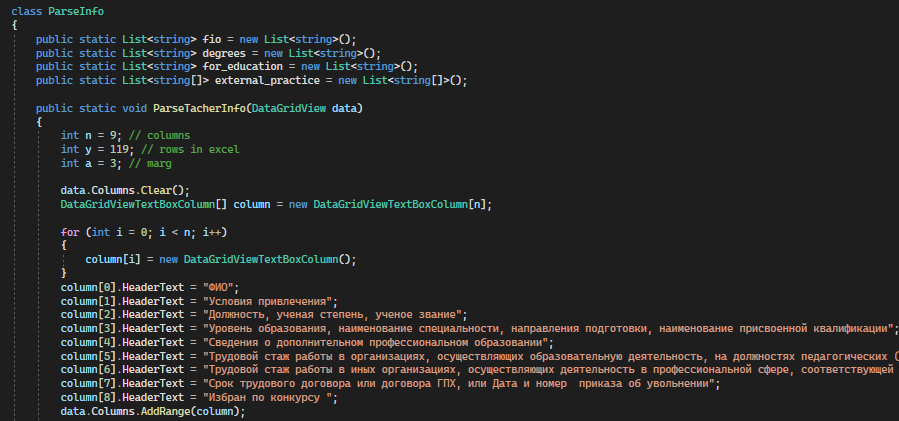
После этого вызываются функции, которые срабатывают при нажатии кнопок(тех самых циферок которые отвечают за листы)



Те в свою очередь вызывают функции класса ParseInfo которые отвечают за заполнение данными DataGridViewInfo ( DataGridViewInfo – таблица на форме) и сохранение данных.

**2.3.2. ParseInfo**

Class ParseInfo:



Данный класс как было сказано ранее отвечает за считывание данных с excel файла. Содержит в себе 4 листа в которых и будет хранится информация. Также есть 2 статичные функции которые записывают в эти листы данные

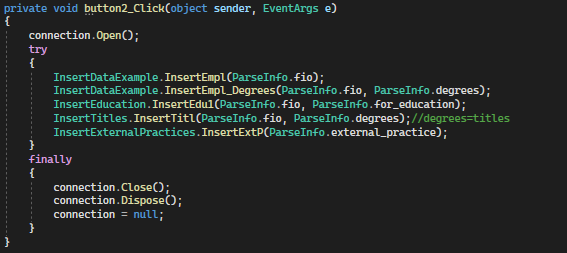
Алгоритм проходит по листам документа, ищет нужный лист, далее проходится по столбцам и строкам - заполняет datagrid и листы данными.

Для работы с Excel документами я использовал библиотеку ClosedXML.Excel, благодаря ей можно проводить различные манипуляции с excel документами, начиная с целых книг и заканчивая свойствами ячеек.

Как только эти методы завершат работу, цикл функций начавшихся с кнопки «Выбрать файл» также завершится и программа будет ждать дальнейших указаний.

**2.3.3. Insert Info**

Когда пользователь нажимает кнопку «Закинуть в базу» вызывается данная функция:



Она открывает соединение с БД и вызывает методы из классов типа Insert..

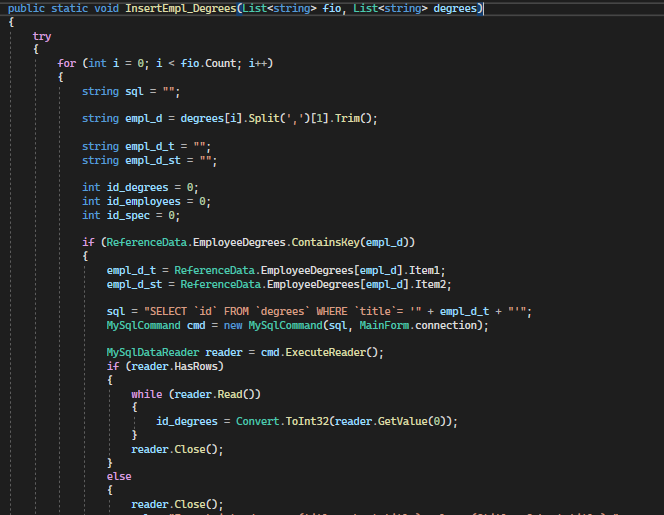
Каждый из этих классов отвечает за отдельные таблицы в БД

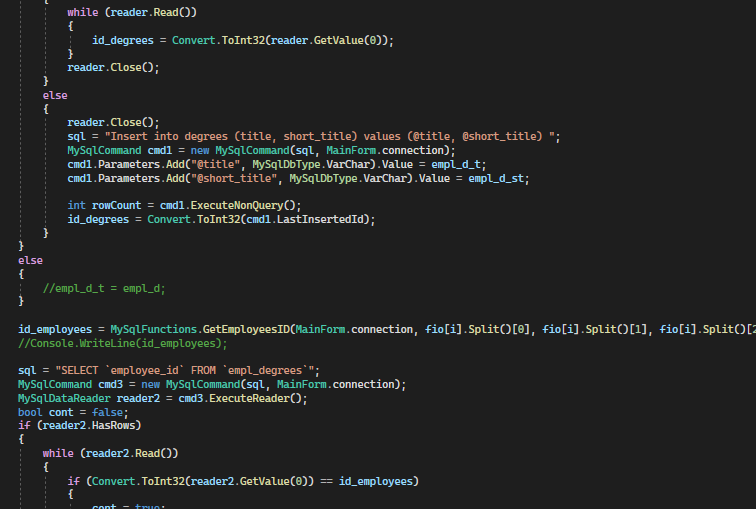
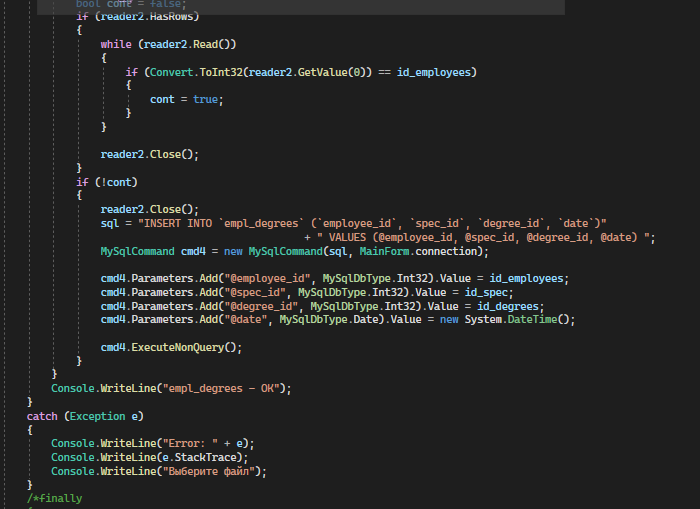
|  |  |
| --- | --- |
| Класс | Таблицы |
| InsertDegrees | employees, degrees, empl\_degrees |
| InsertEducation | empl\_education |
| InsertTitles | empl\_titles |
| InsertExternalPractices | external\_practices |
|  |  |

Каждый класс содержит функцию в которых и реализуются алгоритмы обработки данных и заполнения БД, приведем пример:

1. Цикл по Преподавателям (ParseInfo.fio)
   1. Берем данные из других листов по тому же индексу
      1. Если встречаются разные форматы ячеек разделяем их ifом или
      2. Если встречаются «массивы данных» делаем по ним for и закидываем каждый по отдельности
         1. Проверка на пустоту
         2. Может быть еще другие проверки
         3. Забираем данные с ячейки (отделяем ненужные слова и т.п.)
   2. Собираем все поля данных, которые будем закидывать в БД
      1. Запрос к базе где получаем id поля (если надо)
   3. Проверяем нет ли уже точно такой же строки в таблице
   4. Закидываем в БД
2. Конец цикла
3. Отчет о успешности

Приведем пример кода с функции InsertEmpl\_Degrees из класса InsertDegrees



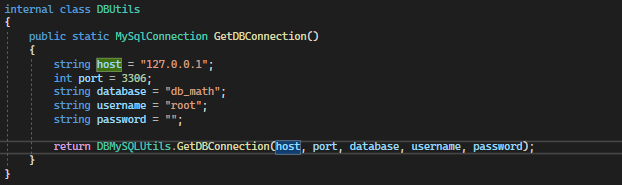


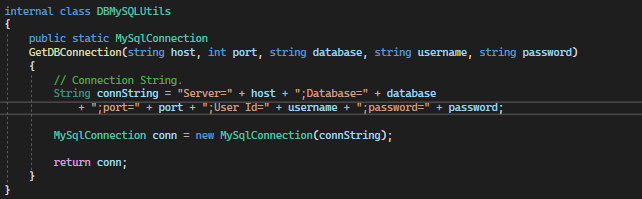
После заполнения таблиц соединение закроется.

**2.3.4. SQL запросы**

Для работы с Базой данных я использовал библиотеку MySql.Data

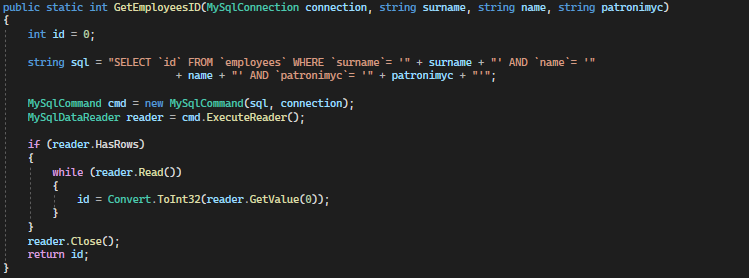
Подключение к БД производится следующим образом:



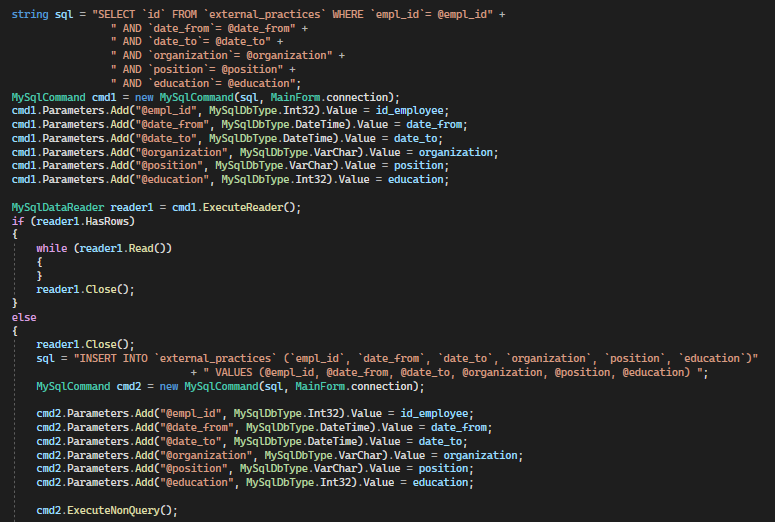


В программе я в основном использовал запросы двух типов “SELECT” и “INSERT”, select для проверки есть ли такая запись в базе или для получения id некоторых строк, insert для заполнения.

Для упрощения жизни я вынес некоторые запросы в отдельный класс MySqlFunctions, так как они часто повторялись, например получение id преподавателя из таблицы employees.

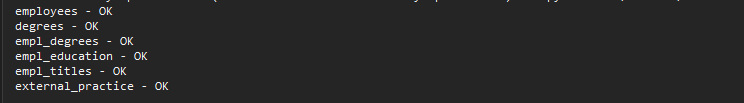


Или же заполнение external\_practices, вынес так как в одной функции не помещалась вся обработка данных, sql запросы:

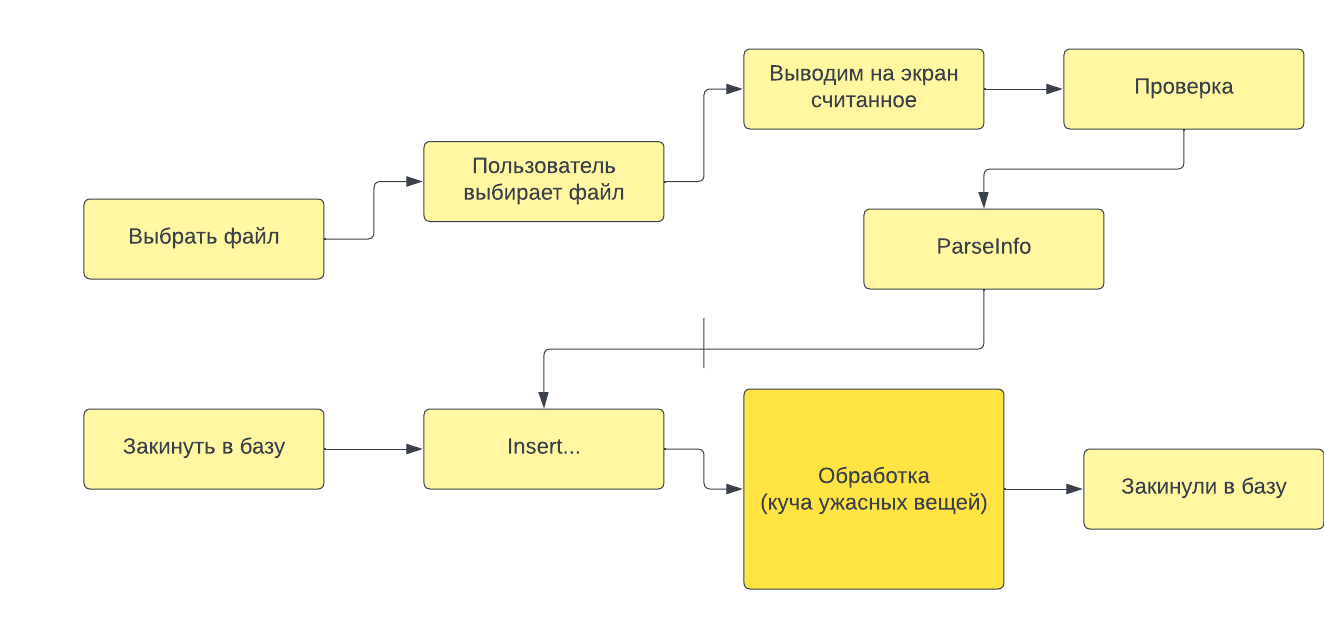


**2.3.4. Выводы и общий алгоритм**

Во многих функциях применяется конструкция try {} catch {} для вывода ошибок в консоль во время выполнения программы, если же проблем не возникло программа должна вывести в консоль отчет о заполнении таблиц типа: “table\_name”-OK



Общий алгоритм:



**Заключение**

В дальнейших планах добавить в этот проект парсер для «Нагрузок» и может быть еще каких-нибудь других документов

По итогу работы над курсовым проектом:

1. Было написано десктоп приложение и сгенерирована база данных, которая предназначена для сотрудников факультета математики и компьютерных наук.
2. Все участники получили незаменимые знания в таких областях, как базы данных, SQL запросы, работа с файлами Excel, система контроля версий GitHub