**Содержание**

**Введение**

**1.Постановка задачи5**

1.1 Описание предметной области 5

1.2 Входные и выходные данные 6

1.3 Математическая (вербальная) модель задачи 8

1.4 Требования к программно-информационной системе 9

**2.Проектирование программно-информационной системы13**

2.1 Составление перечня функциональности(Userstories) 13

2.2 Составление сценариев взаимодействия 15

2.3 Разработка интерфейса взаимодействия с пользователем17

**3.Разработка документации программно-информационной системы20**

3.1 Разработка диаграммы классов 20

3.2 Разработка диаграммы деятельности 21

3.3 Разработка основных алгоритмов22

**4.Программная реализация информационной системы26**

4.1 Выбор и обоснование используемых технологий 26

4.2 Описание используемых структур данных 27

4.3 Исходный код программы 28

4.4 Описание основных модулей программы 31

**5.Тестирование программно-информационной системы33**

**Заключение36**

**Список использованных источников37**

**Приложение А**

**Введение**

С приходом компьютера в жизнь человека стало намного проще управлять информацией. Все люди, для хранения каких-то данных, используют носители информации, начиная с флеш-карт, заканчивая серверами. Нередко приходится использовать базы данных.

LINQ to SQL представляет технологию доступа и управления реляционными данными. Данная технология позволяет составлять запросу к базе данных в удобной форме с помощью операторов LINQ, которые затем трансформируются в sql-выражения. Ключевыми объектами здесь являются сущности, которые хранятся в базе данных, контекст данных и запрос LINQ.

Целью курсового проекта является получение практических навыков разработки программ в среде Microsoft Visual Studio при помощи технологии WPF (Windows Presentation Foundation) на примере программно-информационной системы “Школа”.

**1 Постановка задачи**

1.1 Описание предметной области

Требуется разработать программную систему, предназначенную для завуча школы. Она должна обеспечивать хранение сведений о каждом учителе, о предметах, которые он преподает, номере закрепленного за ним кабинета. Об учениках должны храниться следующие сведения: фамилия и имя, в каком классе учится, какую оценку имеет в текущей четверти по каждому предмету. Завуч должен иметь возможность добавить сведения о новом учителе или ученике, внести в базу данных четвертные оценки учеников каждого класса по каждому предмету, удалить данные об уволившемся учителе и отчисленном из школы ученике.

Таблица 1.1 — Таблицы, которые должны содержаться в базе данных

|  |  |  |
| --- | --- | --- |
| **Учитель** | **Учитель** | **Успеваемость** |
| Ключ | Ключ | Ключ |
| ФИО | Ученик | Ученик |
| Предмет | Класс | Предмет 1 |
| № кабинета |  | Предмет 2 |
|  |  | Предмет 3 |

Завучу могут потребоваться следующие сведения о текущем состоянии успеваемости:

* Успеваемость по заданному предмету.
* Количество неуспевающих учеников по всем классам.
* У какого учителя самая низкая успеваемость.
* Средняя оценка по всем предметам в каждом классе.
* Класс с самой высокой успеваемостью по всем предметам.
* Класс с самой низкой успеваемостью по всем предметам.

Завуч может выполнять следующие операции:

* Добавить сведения о новом ученике.
* Добавить сведения о новом учителе.
* Внести в базу данных четвертные оценки учеников каждого класса по каждому предмету.
* Удалить данные об уволившемся учителе.
* Удалить данные об отчисленном из школы ученике.

1.2 Входные и выходные данные

Входные и выходные данные для программной системы предоставленны в таблицах 1.2-1.4

Таблица 1.2 – Структура таблицы «TableTeachers»

| COLUMN NAME | DATATYPE | NOT NULL | COMMENTS |
| --- | --- | --- | --- |
| Id | INT | True | PK\_ Id |
| FullName | VARCHAR(50) | True |  |
| Subject | VARCHAR(50) | True |  |
| ClassRoom | INT | True |  |

Таблица 1.3 – Структура таблицы «TableStudents»

| COLUMN NAME | DATATYPE | NOT NULL | COMMENTS |
| --- | --- | --- | --- |
| Id | INT | True | PK\_Id |
| FullName | VARCHAR(50) | True |  |
| * Class | VARCHAR(50) | True |  |

Таблица 1.4 – Структура таблицы « TablePerformance »

| COLUMN NAME | DATATYPE | NOT NULL | COMMENTS |
| --- | --- | --- | --- |
| Id | INT | True | PK\_Id |
| NameStudent | VARCHAR(50) | True |  |
| Maths | VARCHAR(50) | True |  |
| Physics | VARCHAR(50) | True |  |
| Biology | VARCHAR(50) | True |  |

1.3 Математическая (вербальная) модель задачи

Вербальная модель – это письменное или устное представление информационной модели средствами естественного языка. Примером вербальных моделей является текстовое описание объектов или процессов.

На рисунке 1.1 представлена математическая модель программно-информационной системы, на которой можно увидеть действия, доступные пользователю. После запуска программы пользователю открывается главное окно, на котором доступныразличные действия. Пользователю предоставляется возможность просмотра данных из таблицы базы данных, для этого он должен открыть соответствующую вкладку.

Вкладка “Students” содержит элементы для просмотра данных об учениках из таблицы базы данных,а так же кнопки “Create Student” и “Delete Student”, для добавения и удаления ученика соответственно. Вкладка “Teachers” содержит элементы для просмотра данных об учителях из таблицы базы данных,а так же кнопки “Create Teacher” и “Delete Teacher”, для добавения и удаления учителя соответственно. Вкладка “Performance” содержит элементы для просмотра данных об успеваемости из таблицы базы данных,а так же дополнительные кнопки и поля отвечающие за фильтрацию выводимых сведений.

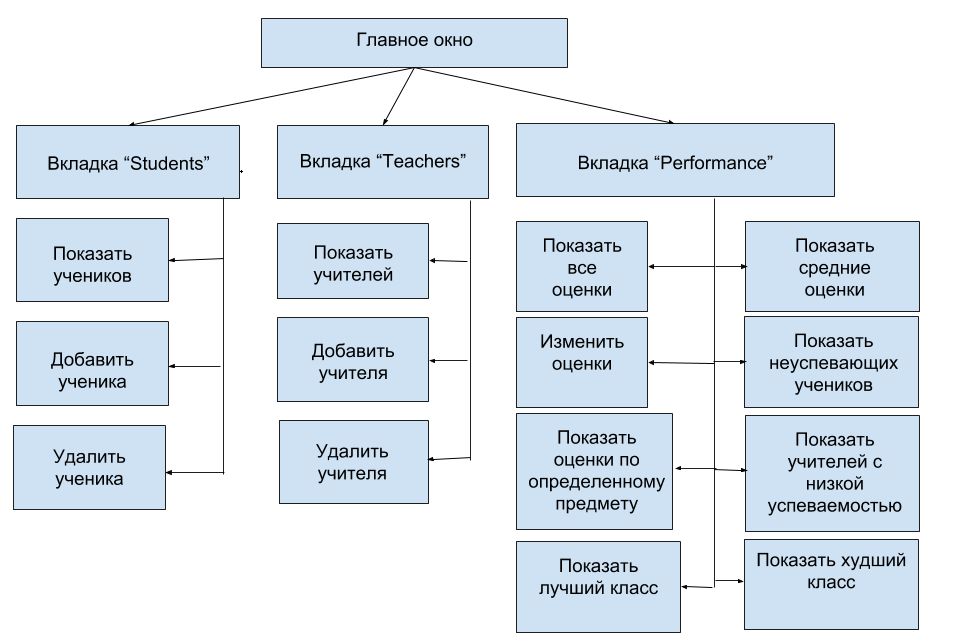


Рисунок 1.1 — Математическая модель программ

1.4 Требования к программно-информационной системе

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на **DirectX**. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML: вы можете создавать насыщенный графический интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C# и VB.NET, либо совмещать и то, и другое.

### Преимущества WPF:

* Использование традиционных языков .NET-платформы - C# и VB.NET для создания логики приложения.
* Возможность декларативного определения графического интерфейса с помощью специального языка разметки XAML, основанном на xml и представляющем альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#/VB.NET
* Независимость от разрешения экрана: поскольку в WPF все элементы измеряются в независимых от устройства единицах, приложения на WPF легко масштабируются под разные экраны с разным разрешением.
* Новые возможности, которых сложно было достичь в WinForms, например, создание трехмерных моделей, привязка данных, использование таких элементов, как стили, шаблоны, темы и др.
* Хорошее взаимодействие с WinForms, благодаря чему, например, в приложениях WPF можно использовать традиционные элементы управления из WinForms.
* Богатые возможности по созданию различных приложений: это и мультимедиа, и двухмерная и трехмерная графика, и богатый набор встроенных элементов управления, а также возможность самим создавать новые элементы, создание анимаций, привязка данных, стили, шаблоны, темы и многое другое
* Аппаратноеускорение графики - вне зависимости от того, работаете ли вы с 2D или 3D, графикой или текстом, все компоненты приложения транслируются в объекты, понятные Direct3D, и затем визуализируются с помощью процессора на видеокарте, что повышает производительность, делает графику более плавной.
* Создание приложений под множество ОС семейства Windows - от Windows XP до Windows 10

В тоже время WPF имеет определенные ограничения. Несмотря на поддержку трехмерной визуализации, для создания приложений с большим количеством трехмерных изображений, прежде всего игр, лучше использовать другие средства - DirectX или специальные фреймворки, такие как Monogame или Unity.

Также стоит учитывать, что по сравнению с приложениями на Windows Forms объем программ на WPF и потребление ими памяти в процессе работы в среднем несколько выше.

LINQ to SQL представляет технологию доступа и управления реляционными данными. Данная технология позволяет составлять запросу к бд в удобной форме в с помощью операторов LINQ, которые затем трансформируются в sql-выражения. Ключевыми объектами здесь являются сущности, которые хранятся в базе данных, контекст данных и запрос LINQ.

SQL — [декларативный](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), применяемый для создания, модификации и управления данными в [реляционной базе данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), управляемой соответствующей [системой управления базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

SQL является прежде всего [информационно-логическим языком](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), предназначенным для описания, изменения и извлечения данных, хранимых в [реляционных базах данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). SQL можно назвать [языком программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), при этом он не является [тьюринг-полным](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B0_%D0%BF%D0%BE_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D1%83), но вместе с тем стандарт языка спецификацией [SQL/PSM](https://ru.wikipedia.org/wiki/SQL/PSM) предусматривает возможность его [процедурных расширений](https://ru.wikipedia.org/wiki/SQL#%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D1%8B%D0%B5_%D1%80%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%B8%D1%8F).

Изначально SQL был основным способом работы пользователя с [базой данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
* изменение структур таблиц.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Язык SQL представляет собой совокупность:

* операторов
* инструкций
* вычисляемых функций

**2 Проектирование программно-информационной системы**

2.1 Составление перечня функциональности(Userstories)

В таблице 2.1 представлен перечень функциональности программы.

Имеются роли “Пользователь”, которая показывает, какие действия производит пользователь при взаимодействии с программой, и “Система” – показывает, какие действия выполняются системой в зависимости от действий “Пользователя”. Например, когда пользователь открываает ввкладку “Students”, система выводит данную таблицу из базы данных в элемент окна “ DataGrid”.

Таблица 2.1 — Перечень функциональности (Userstories)

|  |  |  |  |
| --- | --- | --- | --- |
| Роль | Действие | Тип объекта | Дополнительные возможности |
| Пользователь | Просматривает | Окно | Переход к окнам; выход из приложения |
| Пользователь | Открывает вкладку | Students | - |
| Система | Выводит | Таблицу DataGrid | Просмотр,  удаление,  добавление данных |
| Пользователь | Нажимает кнопку | Show Fabric Data | - |
| Система | Выводит | Таблицу DataGrid | Просмотр,  удаление,  добавление данных |
| Пользователь | Открывает вкладку | Teachers | - |
| Система | Выводит | Таблицу DataGrid | Просмотр,  удаление,  добавление данных |
| Пользователь | Нажимает кнопку | Create Student | - |
| Система | Вывод | Окно добавления | Ввод данных |
| Пользователь | Нажимает кнопку | Create | - |
| Система | Выводит | Начальное окно | - |
| Пользователь | Открывает вкладку | Students / Teachers / Perormance | - |
| Система | Выводит | Таблицу DataGrid | Просмотр,  удаление,  добавление данных |

Диаграмма взаимодействия пользователя с программой представлена на рисунке 2.1



Рисунок 2.1 – Диаграмма взаимодействия пользователя с программой

2.2 Составление сценариев взаимодействия

Программа предлагает пользователю свободные манипуляции с данными. Пользователь может редактировать данные при помощи редактирования информации в таблицах, удаления информации, добавления информации, сохранения изменений в информации.

1.После того, как пользователь запускает программу, открывается окно (Window) (рисунок 2.2), на котором расположены следующие элементы:

Три вкладки “Students”,”Teachers”,”Performance”.На вкладке “Students” расположены DataGrid и две кнопки “Create Student” и “Delete Student”.

На вкладке “Teachers” расположены DataGrid и две кнопки “Create Teacher” и “Delete Teacher”.

На вкладке “Performance” расположены два DataGrid,поля ввода оценок и параметров фильтра, а так же семь кнопок:

“Set Mark”,“Average mark”,”By Subject”,”Low performance stud”,”Low performance teacher”,”Best Class”,”Worst Class”.

2.После нажатия на одну из кнопок “Tab”, на элемент DataGrid выводится соответствующая таблица с соответствующими данными из базы данных. Это могут быть сведения об учениках,учителлях или об оценках.

3.Если пользователь захочет изменить текущие данные добавлением новых, то он нажимает на соответствующую кнопку “Create”. После этого открывается новое окно, в котором содержатся элементы для ввода данных (TextBox) и кнопки: для окончания действия (кнопка “Cancel”) и для подтверждения (сохранения/записи в БД) новых данных (кнопка “Create”) (рисунок 2.3).

4.Далее, после нажатия на кнопку “Create”, система возвращает пользователя на главное окно, где показываеться обновленная таблица.

5.Также пользователю доступно удаление данных. Для этого ему необходимо ввести идентификатор (id) строки таблицы, которую он хочет удалить, в соответствующее поле для ввода, после чего нажать кнопку “Delete” , объект удалиться,а таблица обновиться автоматически.

6.Такие манипуляции пользователь может провести с каждой из существующих таблиц.

7.Пользователь может организовать вывод данных, основанных на одном из критериев вывода, которые он может увидеть на вкладке “Peformance”. Пользователю необходимо ввести значение, которое указано в виде подсказки в поле для ввода или выбрать параметр фильтра из ComdoBox,далее нажать соответствующию кнопку для получения отфильтрованных сведений.

2.3 Разработка интерфейса взаимодействия с пользователем

Графический интерфейс пользователя (GUI) — разновидность пользовательского интерфейса, в котором [элементы интерфейса](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0) (меню, кнопки, значки, списки и т. п.), представленные пользователю на [дисплее](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BF%D0%BB%D0%B5%D0%B9), исполнены в виде [графических](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0) изображений. Также называется графической оболочкой управления.

В отличие от [интерфейса командной строки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8), в GUI пользователь имеет произвольный доступ (с помощью [устройств ввода](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_%D0%B2%D0%B2%D0%BE%D0%B4%D0%B0) — клавиатуры, мыши, джойстика и т. п.) ко всем видимым экранным объектам (элементам интерфейса) и осуществляет непосредственное манипулирование ими. Чаще всего элементы интерфейса в GUI реализованы на основе [метафор](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D1%84%D0%BE%D1%80%D0%B0) и отображают их назначение и свойства, что облегчает понимание и освоение программ неподготовленными [пользователями](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C).

Графический интерфейс пользователя является частью пользовательского интерфейса и определяет взаимодействие с пользователем на уровне визуализированной информации.

На рисунке 2.2 – 2.5 представлены структурные схемы окон программы. На главном окне программы (Рисунок 2.2) находятся такие элементы, как кнопки, текстовые поля, поля ввода и элемент для вывода таблицы из базы данных. На рисунке 2.3 представлено окно программы, которое открывается после нажатия на кнопку “Create”. Здесь содержаться поля для ввода данных и кнопки, которые предназначены для подтверждения добавления элемента или окончания действия и, следовательно, к возвращению к главному окну. Аналогичные элементы находятся на окнах,представленных на рисунках 2.4 и 2.5.



Рисунок 2.2 — Главное окно программы.

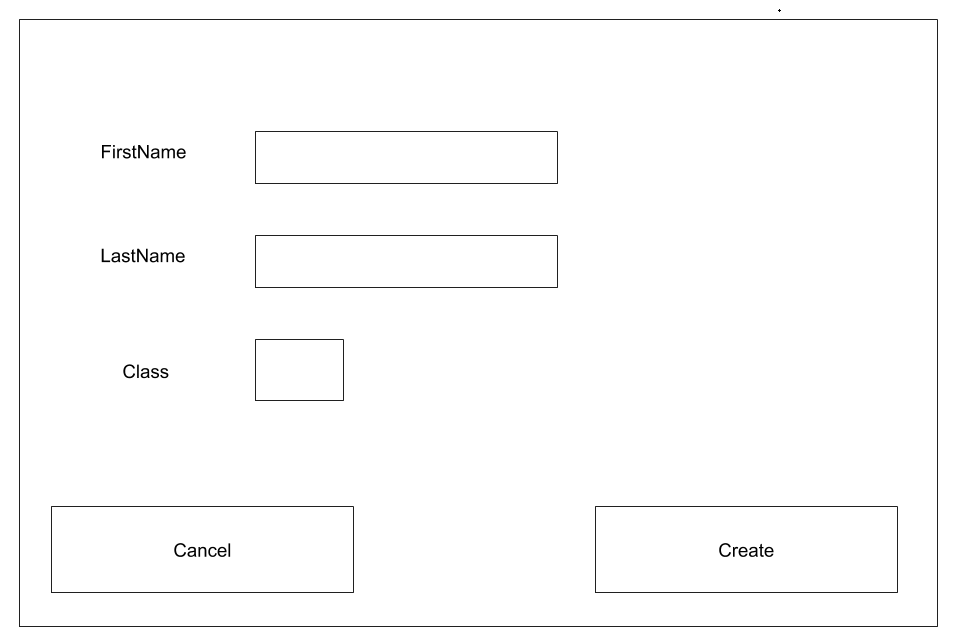


Рисунок 2.3 — Окно после нажатия на кнопку “Create”(Student).

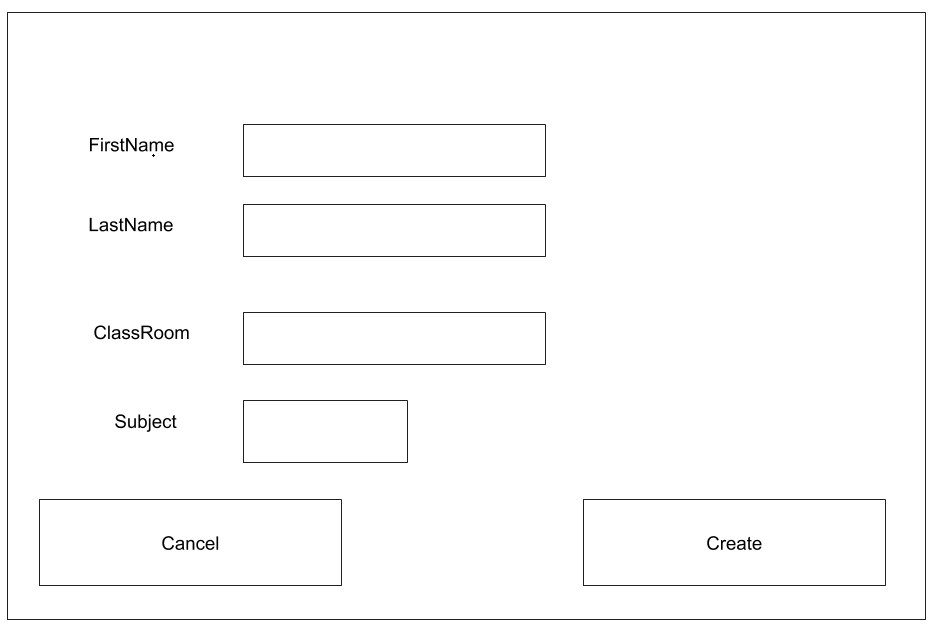


Рисунок 2.4 — Окно после нажатия на кнопку “Create”(Teacher).

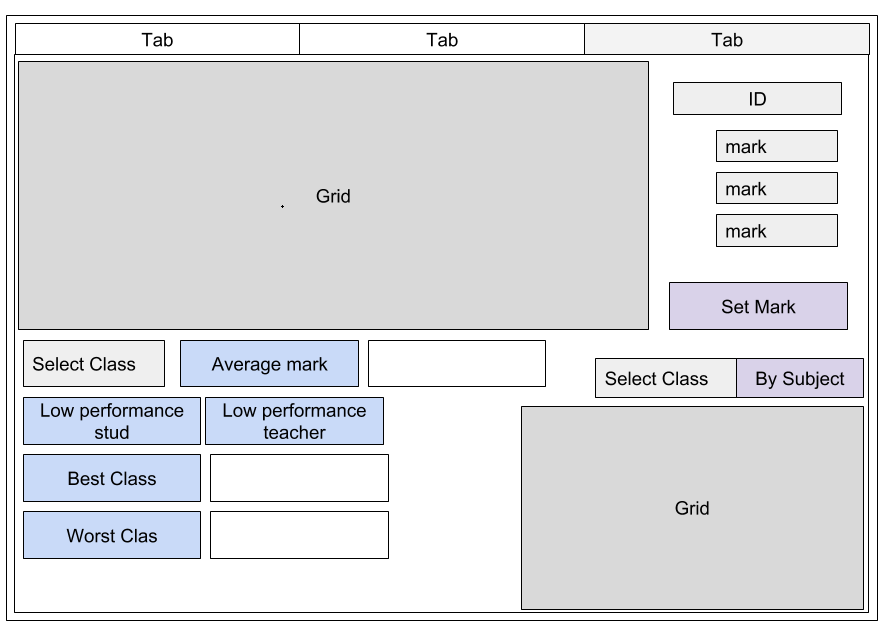


Рисунок 2.5 — Вкладка с фильтрами и оценками.

**3 Разработка документации программно-информационной системы**

3.1 Разработка диаграммы классов

Диаграмма классов  — структурная [диаграмма](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_(UML)) языка моделирования  [UML](https://ru.wikipedia.org/wiki/UML), демонстрирующая общую структуру иерархии [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) системы, их коопераций, [атрибутов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0), [методов](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования. На рисунке 3.1 представлена диаграмма классов данной программно-информационной системы. Диаграмма построена в среде VisualStudio. На диаграмме содержаться классы “Students”, “Teachers”, “Performance.



Рисунок 3.1 —  Диаграмма классов программно-информационной системы

3.2 Разработка диаграммы деятельности

Диаграмма деятельности  — диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграммы деятельности состоят из ограниченного количества фигур, соединённых стрелками. Основные фигуры:

* Прямоугольники с закруглениями — действия;
* Ромбы — решения;
* Широкие полосы — начало (разветвление) и окончание (схождение) ветвления действий;
* Чёрный круг — начало процесса (начальный узел);
* Чёрный круг с обводкой — окончание процесса (финальный узел);

Стрелки идут от начала к концу процесса и показывают потоки управления или потоки объектов (данных).

На рисунке 3.2 графически представлена диаграмма деятельности в форме графа деятельности, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому. После запуска приложение пользователю предлагается выбор действий:

Выбор вкладки “Students”, “Teachers”, “Performance”. После нажатия на вкладку открывается доступ к функциям соответствующей вкладкиa.Дальнейшие действия программы формируются в зависимости от действий пользователя: если тот хочет добавить новую информацию в базу данных, то идут проверка, введены ли значения в поля для ввода. Если значения введены, то программа добавляет их в базу данных, если нет, то пользователю предлагается ввести данные либо отменить текущее действие и перейти к другому шагу.



Рисунок 3.2 — Диаграмма деятельности

3.3 Разработка основных алгоритмов

Алгоритм — конечная совокупность точно заданных правил решения произвольного класса задач или набор [инструкций](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), описывающих порядок действий исполнителя для решения некоторой задачи. Независимые инструкции могут выполняться в произвольном порядке, параллельно, если это позволяют используемые исполнители.

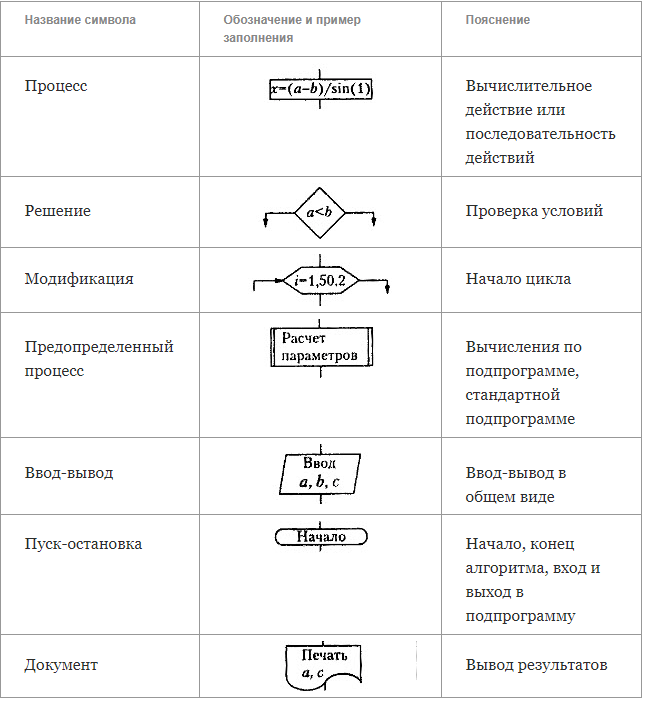
Разработать алгоритм означает разбить задачу на определенную последовательность шагов. От разработчика алгоритма требуется знание особенностей и правил составления алгоритмов.

Основные особенности алгоритмов:

* Наличие ввода исходных данных.
* Наличие вывода результата выполнения алгоритма, поскольку цель выполнения алгоритма – получение результата, имеющего вполне определенное отношение к исходным данным.
* Алгоритм должен иметь дискретную структуру, т.е. алгоритм представляется в виде последовательности шагов, и выполнение каждого очередного шага начинается после завершения предыдущего.
* Однозначность – каждый шаг алгоритма должен быть четко определен и не должен допускать произвольной трактовки исполнителем.
* Конечность – исполнение алгоритма должно закончиться за конечное число шагов.
* Корректность – алгоритм должен задавать правильное решение задачи.
* Массовость (общность) – алгоритм разрабатывается для решения некоторого класса задач, различающихся исходными данными.
* Эффективность – алгоритм должен выполняться за разумное конечное время. При этом выбирается наиболее простой и короткий способ решения задачи при соблюдении, естественно, всех ограничений и требований к алгоритму.

Блок-схема алгоритма — графическое изображение алгоритма в виде связанных между собой с помощью стрелок (линий перехода) и *блоков* — графических символов, каждый из которых соответствует одному шагу алгоритма. Внутри блока дается описание соответствующего действия.

В таблице приведены наиболее часто употребляемые символы.



На рисунке 3.3 представлен основной алгоритм информационной системы. После запуска программы открывается главное окно и выбор действия. Например, после выбора действия “Create”, происходит проверка на корректность выполняемых действий, если всё верно, то действие выполняется и происходит переход к главному окну приложения, если нет, то действие не выполняется.

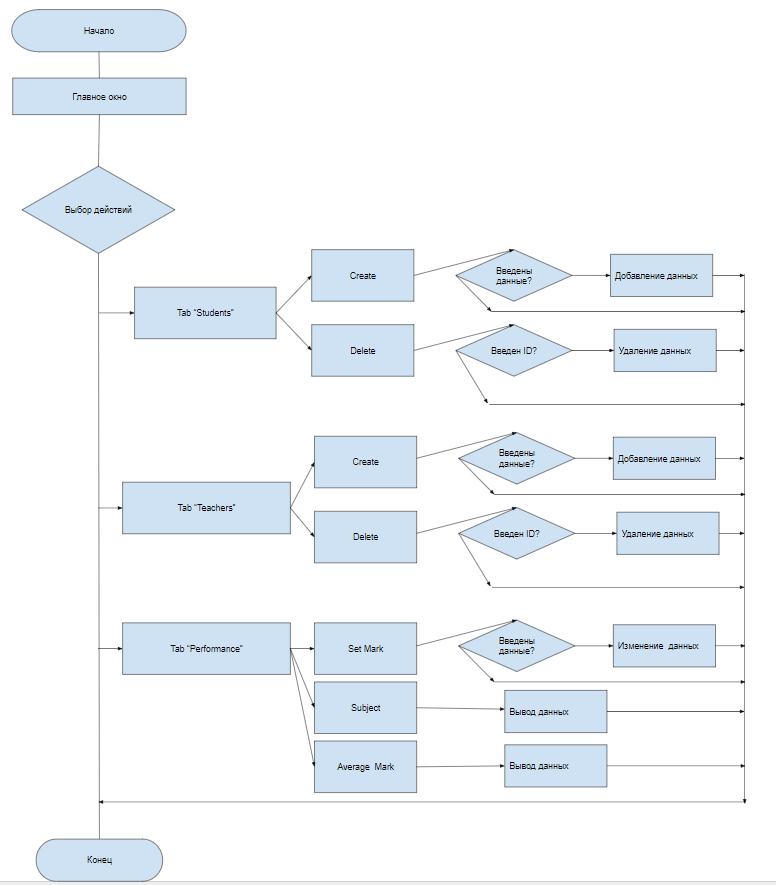


Рисунок 3.3 — Основной алгоритм информационной системы

**4 Программная реализация информационной системы**

4.1 Выбор и обоснование используемых технологий

При создании курсового проекта был выбран язык программирования C#, потому что он является изучаемым языком на данном курсе.

Язык C# появился в июне 2000 г., в результате работы группы разработчиков компании Microsoft, возглавляемой Андерсом Хейлсбергом.

.NET представляет собой новую платформу, новый API для программирования в Windows, а С# есть новый язык, созданный с нуля, для работы с этой платформой.

Авторы C# стремились создать язык, сочетающий простоту и выразительность современных объектно-ориентированных языков (вроде Java) c богатством возможностей и мощью C++. По словам Андерса Хейлсберга, C# позаимствовал большинство своих синтаксических конструкций из C++. В частности, в нем присутствуют такие удобные типы данных, как структуры и перечисления (другой потомок C++ -- Java -- лишен этих элементов, что создает определенные неудобства при программировании). Синтаксические конструкции С# унаследованы не только от C++, но и от VisualBasic. Например, в С#, как и в VisualBasic, используются свойства классов. Как C++, С# позволяет производить перегрузку операторов для созданных вами типов, Java не поддерживает ни ту, ни другую возможность). С# это фактически гибрид разных языков.

При этом С# синтаксически не менее (если не более) чист, чем Java, и так же

прост, как VisualBasic, но обладает практически той же мощью и гибкостью, что и C++.

Особенности С#:

* Полная поддержка классов и объектно-ориентированного программирования, включая наследование интерфейсов и реализаций, виртуальных функций и перегрузки операторов.
* Полный и хорошо определенный набор основных типов.
* Автоматическое освобождение динамически распределенной памяти.
* Возможность отметки классов и методов атрибутами, определяемыми пользователем. Это может быть полезно при документировании и способно воздействовать на процесс компиляции (например, можно пометить методы, которые должны компилироваться только в отладочном режиме).
* Полный доступ к библиотеке базовых классов .NET, а также легкий доступ к Windows API (если это действительно необходимо).
* Указатели и прямой доступ к памяти, если они необходимы. Однако язык разработан таким образом, что практически во всех случаях можно обойтись и без этого.
* Поддержка свойств и событий в стиле VB и др.

4.2 Описание используемых структур данных

Объект SqlConnection – используется для подключения к базе данных;

В конструктор объекту SqlConnection передается строка подключения, которая инициализирует объект. Чтобы использовать этот объект и подключаться к базе данных, нужно выполнить метод Open(), а после завершения работы с базой данных - вызвать метод Close() для закрытия подключения.

После установки подключения можно выполнить команды. Команды представлены объектом интерфейса System.Data.IDbCommand. Провайдер для MS SQL предоставляет его реализацию в виде класса SqlCommand. Этот класс инкапсулирует sql-выражение, которое должно быть выполнено.

Чтобы задействовать LINQ to SQL в проекте, нам надо добавить библиотеку System.Data.Linq.dll.

Атрибут [Table] позволяет выполнить сопоставление таблицы из бд с данной моделью. Однако по умолчанию модель должна называться также, как и таблица, то есть Users. Поэтому, чтобы сопоставление прошло успешно, в атрибуте Table указывается свойство Name с именем таблицы.

Каждое свойство модели соответствует столбцу по типу данных. Чтобы происходило автоматическое сопоставление, над свойствами применяется атрибут [Column]. Без данного атрибута система не сможет связать столбец из таблицы со свойством.

Контекст данных представлен объектом DataContext. В конструктор этого класса передается строка подключения. Через контекст данных можно работать с базой данных.

Чтобы применить фильтрацию и сортировку в LINQ to SQL, мы можем воспользоваться стандартными операторами и выражениями LINQ.

Операторы where и orderby позволяют произвести сортировку.

Для группировки данных применяется оператор group by или метод GroupBy().

Для удаления объекта из базы данных применяется метод DeleteOnSubmit() или DeleteAllOnSubmit(), если удаляется список объектов.

4.3 Исходный код программы

Исходный код программы см. в Приложении А.

Типовой запрос данных из таблиц представлен на рисунке 4.1

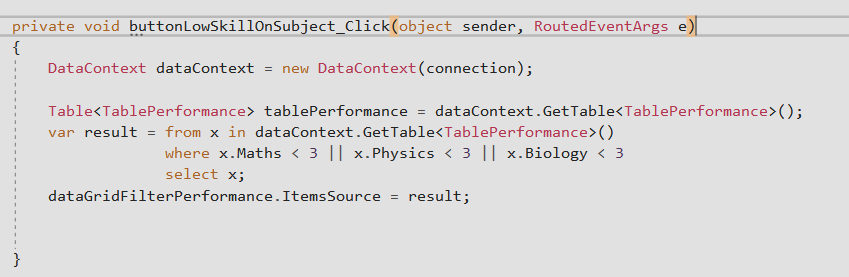


Рисунок 4.1 – Типовой запрос данных из таблицы TablePerformance

В этом фрагменте кода мы получаем данные из таблицы TablePerformance. Для этого мы используем класс DataContext, чтобы получать поля из таблицы. Затем создаем переменную result, с помощью которой осуществляется запрос linq to sql. Из этого кода можно понять, что мы достаем из таблицы значения полей NameStudent,Maths,Physics и Biology при условии, что значение Maths или Physics или Biology являются меньше чем 3. ItemSource говорит о том, что данные из таблицы будут занесены в dataGrid.

Добавление записи сразу в две таблицы реализовано следующим образом:

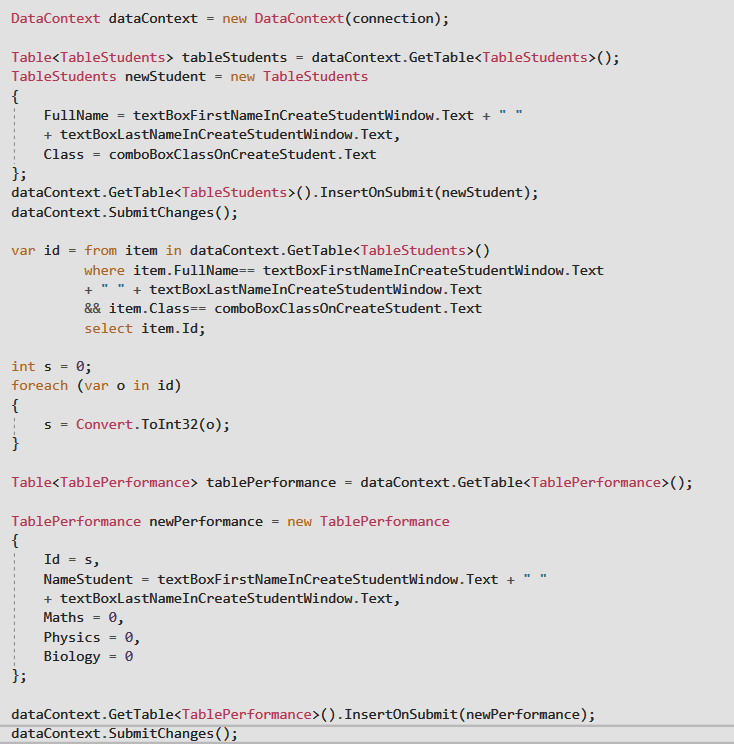


Рисунок 4.2 – Реализация записи в две таблицам сразу

Сначала добавляем запись об ученике в TableStudent,где будет храниться его ФИО и класс в котором он учиться.Далее добавляем запись о нем в TablePerformance.Для этого получаем его ID с помощью Linq запроса.Зная ID добавляем запись с ФИО ученика и его оценками(при создании равные 0).InsertonSubmit добавляет объект класса в табицу.SubmitChanges() выполняет сравнение данных и сохраняет изменения в БД.

4.4 Описание основных модулей программы

Пространство имён System.Data.SqlClient содержит множество классов, вот основные из них:

* [SqlConnection](https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlconnection?view=netframework-4.7.2): Представляет соединение с базой данных SQL Server. Этот класс не может быть унаследован. Данный класс содержит ряд методов:
* SqlConnection.Open - открывает соединение с базой данных с параметрами свойств, указанными в ConnectionString.
* SqlConnection.OpenAsync - асинхронная версия Open (), которая открывает соединение с базой данных с параметрами свойств, указанными в ConnectionString.
* SqlConnection.Close - закрывает соединение с базой данных. Это предпочтительный метод закрытия любого открытого соединения.
* [SqlCommand](https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlcommand?view=netframework-4.7.2): Представляет инструкцию Transact-SQL или хранимую процедуру для выполнения в базе данных SQL Server. Этот класс не может быть унаследован.
* [SqlCommandBuilder](https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlcommandbuilder?view=netframework-4.7.2): Автоматически генерирует однотабличные команды, которые используются для согласования изменений, внесенных в набор данных, со связанной базой данных SQL Server. Этот класс не может быть унаследован.
* [SqlConnectionStringBuilder](https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlconnectionstringbuilder?view=netframework-4.7.2): Предоставляет простой способ создания и управления содержимым строк подключения, используемых классом SqlConnection.
* [SqlDataReader](https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqldatareader?view=netframework-4.7.2): Предоставляет способ чтения потока строк только для пересылки из базы данных SQL Server. Этот класс не может быть унаследован.
* [SqlErrorCollection](https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlerrorcollection?view=netframework-4.7.2): Собирает все ошибки, сгенерированные поставщиком данных .NET Framework для SQL Server. Этот класс не может быть унаследован.

LINQ to SQL — наименование, присвоенное API-интерфейсу IQueryable<T>, который позволяет запросам LINQ работать с базой данных Microsoft SQL Server. Чтобы воспользоваться преимуществами LINQ to SQL в проект понадобится добавить ссылку на сборку System.Data.Linq.dll, а также директиву using System.Data.Linq.

**5 Тестирование программно-информационной системы**

Программно-информационная система “Школа” предназначена для хранения информации о школе в базе данных.

Данная система соответствует всем требованиям, которые были ей приписаны на первых этапах создания. В ней есть список данных об учениках, учителях, успеваемости, также включена возможность их редактирования и сохранения изменений. Система позволяет делать выборку по критериям, которые указаны в условии задания.

Также оформление данной системы не содержит лишних элементов, которые могут усложнить восприятие и взаимодействие программы с пользователем. Интерфейс представлен на английском языке для понимания данной системы большим количеством пользователей. На рисунках 5.1- 5.6 приведен интерфейс системы.

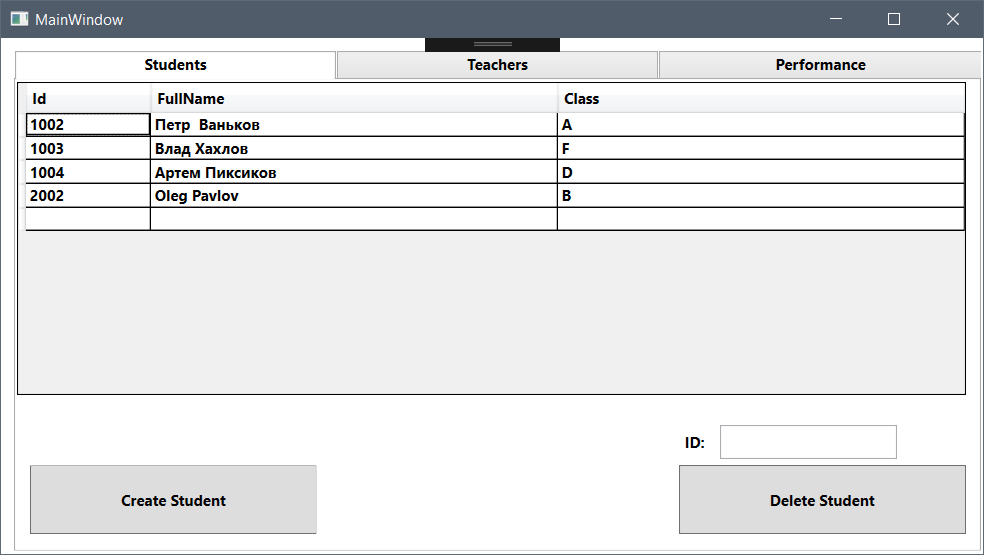


Рисунок 5.1 — Главное окно программы при запуске

Рисунок 5.2 —Главное окно программы после нажатия Tab “Teachers”, которая проводит сортировку данных и выводит список кур с максимальной яйценоскостью

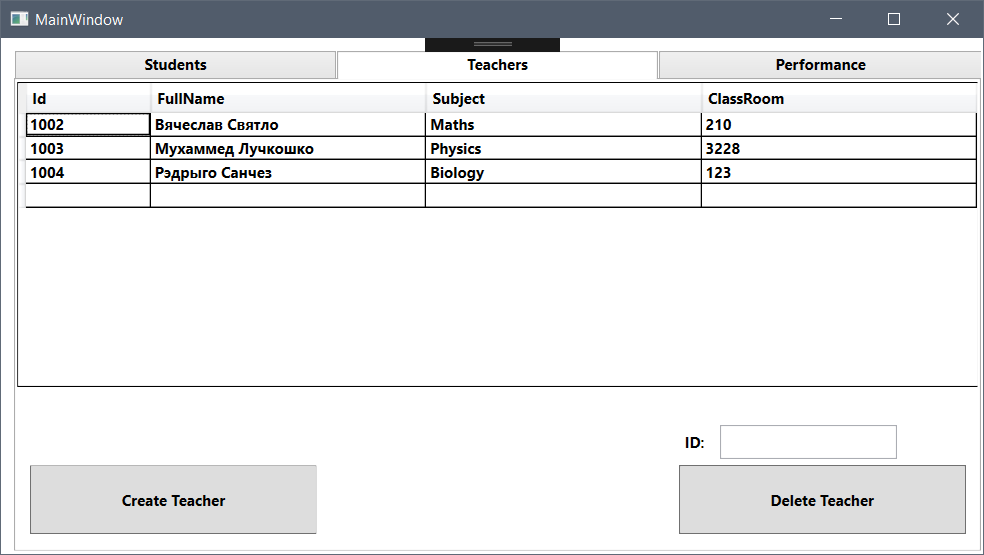


Рисунок 5.3. — Окно программы после нажатия на кнопку “Teachers”

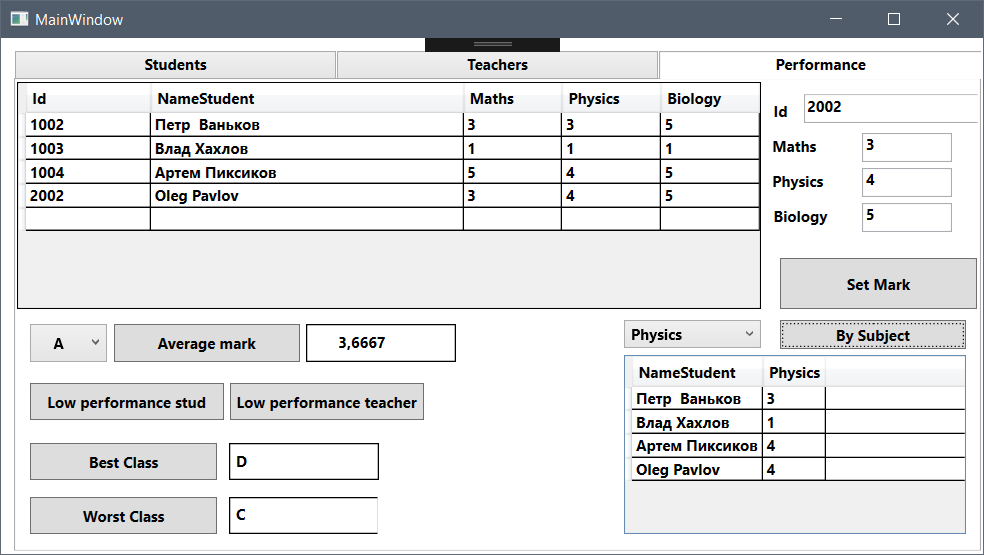


Рисунок 5.4 — Окно программы после нажатия на кнопку “Performance”

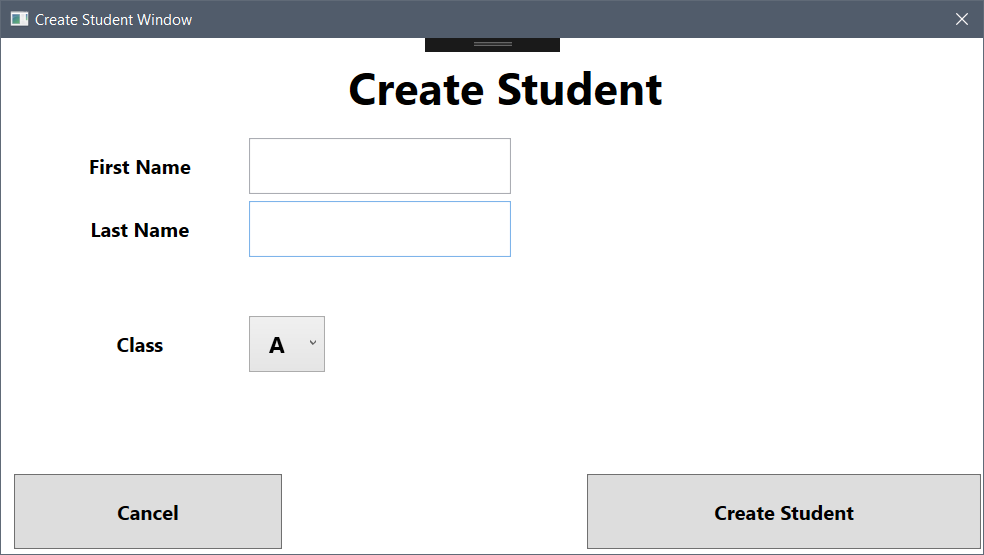


Рисунок 5.5 — Окно программы после нажатия на кнопку “Create Student”

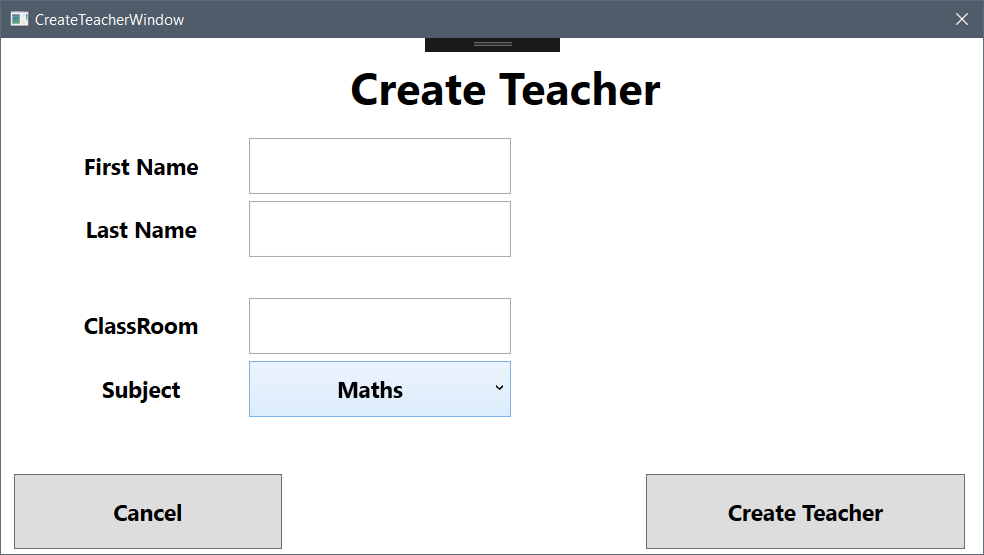


Рисунок 5.6 — Окно программы после нажатия на кнопку “Create Teacher”

**Заключение**

В результате курсового проекта была разработана программно-информационная система “Школа”. Были получены практические навыки в использовании таких технологий, как WPF (Windows Presentation Foundation) и Linq To SQL, что удовлетворяет заданию. Проект разрабатывался в среде Microsoft Visual Studio. Также были получены и закреплены навыки работы с реляционной системой управления базами данных MySQL. Данная система позволяет легко работать с реляционными (табличными) базами данных, осуществлять различные запросы к ним на языке запросов SQL (Structured Query Language).

**Список использованных источников**

1. **Linq To Sql**

<https://metanit.com/sharp/adonet/4.1.php>

1. **MySQL**

<https://metanit.com/sharp/adonet/2.1.php>

1. **SqlDataAdapter**

<https://metanit.com/sharp/adonet/3.1.php>

4. **Руководство по WPF**

https:// metanit.com/sharp/wpf/

1. **Диаграммы классов Visual Studio**

http:// professorweb.ru/my/programs/visual-studio/level3/3\_6.php

1. **C# и MS SQL Server**

<https://metanit.com/sharp/adonet/4.2.php>

**Приложение А**

**Код программы**

Файл TableStudents.cs

using System.Data.Linq.Mapping;

namespace SchoolDateBaseWPF

{

[Table(Name = "TableStudents")]

public class TableStudents

{

[Column(IsPrimaryKey = true, IsDbGenerated = true)]

public int Id { get; set; }

[Column]

public string FullName { get; set; }

[Column]

public string Class { get; set; }

}

}

Файл TableTeachers.cs

using System.Data.Linq.Mapping;

namespace SchoolDateBaseWPF

{

[Table(Name = "TableTeachers")]

public class TableTeachers

{

[Column(IsPrimaryKey = true, IsDbGenerated = true)]

public int Id { get; set; }

[Column]

public string FullName { get; set; }

[Column]

public string Subject { get; set; }

[Column]

public int Classroom { get; set; }

}

}

Файл TablePerformance.cs

using System.Data.Linq.Mapping;

namespace SchoolDateBaseWPF

{

[Table(Name = "TablePerformance")]

public class TablePerformance

{

[Column(IsPrimaryKey = true)]

public int Id { get; set; }

[Column]

public string NameStudent { get; set; }

[Column]

public int Maths { get; set; }

[Column]

public int Physics { get; set; }

[Column]

public int Biology { get; set; }

}

}

Файл App.config

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<configSections>

</configSections>

<connectionStrings>

<add name="SchoolDateBaseWPF.Properties.Settings.ShoolDateBaseConnectionString"

connectionString="Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\SchoolDateBase.mdf;Integrated Security=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

</configuration>

Файл MainWindow.xaml.cs

using System;

using System.Linq;

using System.Windows;

using System.Windows.Input;

using System.Data.SqlClient;

using System.Data;

using System.Data.Linq;

using System.Text.RegularExpressions;

namespace SchoolDateBaseWPF

{

public partial class MainWindow : Window

{

SqlConnection connection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog=SchoolDataBase;Integrated Security=True;

Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");

public MainWindow()

{

InitializeComponent();

}

#region RegionShowTable

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

ShowStudents();

ShowTeachers();

ShowPerformance();

}

public void ShowStudents()

{

DataContext dataContext = new DataContext(connection);

Table<TableStudents> tableStudents = dataContext.GetTable<TableStudents>();

dataGridRigthContentStudents.ItemsSource = tableStudents;

}

public void ShowTeachers()

{

DataContext dataContext = new DataContext(connection);

Table<TableTeachers> tableTeachers = dataContext.GetTable<TableTeachers>();

dataGridRigthContentTechers.ItemsSource = tableTeachers;

}

public void ShowPerformance()

{

DataContext dataContext = new DataContext(connection);

Table<TablePerformance> tablePerformance = dataContext.GetTable<TablePerformance>();

dataGridPerformance.ItemsSource = tablePerformance;

}

#endregion

#region RegionCreateButton

private void createStudent\_Click(object sender, RoutedEventArgs e)

{

CreateStudentWindow createWindow = new CreateStudentWindow(connection);

createWindow.Show();

Close();

}

private void buttonCreateTeacher\_Click(object sender, RoutedEventArgs e)

{

CreateTeacherWindow createTeacherWindow = new CreateTeacherWindow(connection);

createTeacherWindow.Show();

Close();

}

#endregion

#region RegionDeleteButton

private void buttonDeleteStudent\_Click(object sender, RoutedEventArgs e)

{

bool success = Int32.TryParse(textBoxIdDelete.Text, out int number);

if (textBoxIdDelete.Text != " " && success == true)

{

DataContext dataContext = new DataContext(connection);

var deleteStudent =from del in dataContext.GetTable<TableStudents>()

where del.Id == Convert.ToInt32(textBoxIdDelete.Text)

select del;

foreach (var detail in deleteStudent)

{

dataContext.GetTable<TableStudents>().DeleteOnSubmit(detail);

}

dataContext.SubmitChanges();

var deletePerformance = from del in dataContext.GetTable<TablePerformance>()

where del.Id == Convert.ToInt32(textBoxIdDelete.Text)

select del;

foreach (var detail in deletePerformance)

{

dataContext.GetTable<TablePerformance>().DeleteOnSubmit(detail);

}

dataContext.SubmitChanges();

testlabel.Content = "Deleted";

ShowStudents();

ShowPerformance();

}

else

{

testlabel.Content = "Error!!!";

}

}

private void buttonDeleteTeacher\_Click(object sender, RoutedEventArgs e)

{

bool success = Int32.TryParse(textBoxIdDeleteTeacher.Text, out int number);

if (textBoxIdDeleteTeacher.Text != " " && success == true)

{

DataContext dataContext = new DataContext(connection);

var deleteTeachers = from del in dataContext.GetTable<TableTeachers>()

where del.Id == Convert.ToInt32(textBoxIdDeleteTeacher.Text)

select del;

foreach (var detail in deleteTeachers)

{

dataContext.GetTable<TableTeachers>().DeleteOnSubmit(detail);

}

dataContext.SubmitChanges();

testlabelteacher.Content = "Deleted";

ShowTeachers();

}

else

{

testlabelteacher.Content = "Error!!!";

}

}

#endregion

private void buttonSetMark\_Click(object sender, RoutedEventArgs e)

{

DataContext dataContext = new DataContext(connection);

var updateStud = from up in dataContext.GetTable<TablePerformance>()

where up.Id == Convert.ToInt32(textBoxIdStudent.Text)

select up;

foreach (var i in updateStud)

{

i.Maths= (textBoxSetMarkMaths.Text != "") ? Convert.ToInt32(textBoxSetMarkMaths.Text) : i.Maths;

i.Physics = (textBoxSetMarkPhysics.Text != "") ? Convert.ToInt32(textBoxSetMarkPhysics.Text) : i.Physics;

i.Biology = (textBoxSetMarkBiology.Text != "") ? Convert.ToInt32(textBoxSetMarkBiology.Text) : i.Biology;

}

dataContext.SubmitChanges();

ShowPerformance();

}

private void buttonPoPredmety\_Click(object sender, RoutedEventArgs e)

{

DataContext dataContext = new DataContext(connection);

if (comboBoxSubgect.Text == "Maths")

{

Table<TablePerformance> tablePerformance = dataContext.GetTable<TablePerformance>();

var result = from x in dataContext.GetTable<TablePerformance>()

select new { x.NameStudent, x.Maths };

dataGridFilterPerformance.ItemsSource = result;

}

if (comboBoxSubgect.Text == "Physics")

{

Table<TablePerformance> tablePerformance = dataContext.GetTable<TablePerformance>();

var result = from x in dataContext.GetTable<TablePerformance>()

select new { x.NameStudent, x.Physics };

dataGridFilterPerformance.ItemsSource = result;

}

if (comboBoxSubgect.Text == "Biology")

{

Table<TablePerformance> tablePerformance = dataContext.GetTable<TablePerformance>();

var result = from x in dataContext.GetTable<TablePerformance>()

select new { x.NameStudent, x.Biology };

dataGridFilterPerformance.ItemsSource = result;

}

}

private void buttonLowSkillOnSubject\_Click(object sender, RoutedEventArgs e)

{

DataContext dataContext = new DataContext(connection);

Table<TablePerformance> tablePerformance = dataContext.GetTable<TablePerformance>();

var result = from x in dataContext.GetTable<TablePerformance>()

where x.Maths < 3 || x.Physics < 3 || x.Biology < 3

select x;

dataGridFilterPerformance.ItemsSource = result;

}

private void buttonLowMarksSubject\_Click(object sender, RoutedEventArgs e)

{

DataContext dataContext = new DataContext(connection);

Table<TablePerformance> tablePerformance = dataContext.GetTable<TablePerformance>();

var resultOnMatsh = from x in dataContext.GetTable<TablePerformance>()

select x.Maths;

int marksMaths = 0;

foreach (var o in resultOnMatsh)

{

marksMaths += Convert.ToInt32(o);

}

var resultOnPhysics = from x in dataContext.GetTable<TablePerformance>()

select x.Physics;

int marksPhysics = 0;

foreach (var o in resultOnPhysics)

{

marksPhysics += Convert.ToInt32(o);

}

var resultOnBiology = from x in dataContext.GetTable<TablePerformance>()

select x.Biology;

int marksBiology = 0;

foreach (var o in resultOnBiology)

{

marksBiology += Convert.ToInt32(o);

}

string subjectVariant = "";

if (marksMaths < marksPhysics && marksMaths < marksBiology)

{

subjectVariant = "Maths";

};

if (marksPhysics < marksMaths && marksPhysics < marksBiology)

{

subjectVariant = "Physics";

};

if (marksBiology < marksMaths && marksBiology < marksPhysics)

{

subjectVariant = "Biology";

};

var resultTeacher = from x in dataContext.GetTable<TableTeachers>()

where x.Subject == Convert.ToString(subjectVariant)

select new {x.FullName,x.Subject};

dataGridFilterPerformance.ItemsSource = resultTeacher;

}

private void buttonAverageMarks\_Click(object sender, RoutedEventArgs e)

{

DataContext dataContext = new DataContext(connection);

var resultA = from x in dataContext.GetTable<TableStudents>()

where x.Class == comboBoxAverageMarksInClass.Text

select new {x.Id};

var marksA = from x in dataContext.GetTable<TablePerformance>()

select new {x.Id, x.Maths, x.Physics, x.Biology };

double averageA = 0;

int countA = 0;

foreach (var a in resultA)

{

foreach (var o in marksA)

{

if (o.Id == a.Id)

{

averageA += Convert.ToDouble(o.Maths + o.Physics + o.Biology) / 3;

countA++;

}

}

}

labelAverageMarksInClass.Content= String.Format("{0,12:F4}", averageA = Convert.ToDouble(averageA / countA));

}

private void buttonBestClassOnMark\_Click(object sender, RoutedEventArgs e)

{

GetMarkClass("Best");

}

private void buttonTheWorstClassOnMark\_Click(object sender, RoutedEventArgs e)

{

GetMarkClass("Worst");

}

public void GetMarkClass(string variant)

{

DataContext dataContext = new DataContext(connection);

var resultA = from x in dataContext.GetTable<TableStudents>()

where x.Class == "A"

select new { x.Id };

var resultB = from x in dataContext.GetTable<TableStudents>()

where x.Class == "B"

select new { x.Id };

var resultC = from x in dataContext.GetTable<TableStudents>()

where x.Class == "C"

select new { x.Id };

var resultD = from x in dataContext.GetTable<TableStudents>()

where x.Class == "D"

select new { x.Id };

var resultF = from x in dataContext.GetTable<TableStudents>()

where x.Class == "F"

select new { x.Id };

var marksAll = from x in dataContext.GetTable<TablePerformance>()

select new { x.Id, x.Maths, x.Physics, x.Biology };

int summMarkA = 0;

int summMarkB = 0;

int summMarkC = 0;

int summMarkD = 0;

int summMarkF = 0;

foreach (var a in resultA)

{

foreach (var o in marksAll)

{

if (o.Id == a.Id)

{

summMarkA += o.Maths + o.Physics + o.Biology;

}

}

}

foreach (var a in resultB)

{

foreach (var o in marksAll)

{

if (o.Id == a.Id)

{

summMarkB += o.Maths + o.Physics + o.Biology;

}

}

}

foreach (var a in resultC)

{

foreach (var o in marksAll)

{

if (o.Id == a.Id)

{

summMarkC += o.Maths + o.Physics + o.Biology;

}

}

}

foreach (var a in resultD)

{

foreach (var o in marksAll)

{

if (o.Id == a.Id)

{

summMarkD += o.Maths + o.Physics + o.Biology;

}

}

}

foreach (var a in resultF)

{

foreach (var o in marksAll)

{

if (o.Id == a.Id)

{

summMarkF += o.Maths + o.Physics + o.Biology;

}

}

}

int[] massMarks = new int[5] { summMarkA, summMarkB, summMarkC, summMarkD, summMarkF };

string[] massClassNumber = new string[5] { "A", "B", "C", "D", "F" };

int max = massMarks[0];

int min = massMarks[0];

if (variant == "Best")

{

for (int i = 0; i < 5; i++)

{

if (massMarks[i] > max)

{

max = massMarks[i];

labelBestClassOnMark.Content = massClassNumber[i];

}

}

}

if (variant == "Worst")

{

for (int i = 0; i < 5; i++)

{

if (massMarks[i] < min)

{

min = massMarks[i];

labelTheWorstClassOnMark.Content = massClassNumber[i];

}

}

}

}

#region RegionValidationMain

private void textBoxIdStudent\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

string inputSymbol = e.Text.ToString();

if (!Regex.Match(inputSymbol, @"[0-9]|\.|,").Success)

{

e.Handled = true;

}

}

private void textBoxIdDeleteTeacher\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

string inputSymbol = e.Text.ToString();

if (!Regex.Match(inputSymbol, @"[0-9]|\.|,").Success)

{

e.Handled = true;

}

}

private void textBoxSetMarkMaths\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

e.Handled = setMarkText(e);

}

private void textBoxSetMarkPhysics\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

e.Handled = setMarkText(e);

}

private void textBoxSetMarkBiology\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

e.Handled = setMarkText(e);

}

private bool setMarkText(TextCompositionEventArgs e)

{

string inputSymbol = e.Text.ToString();

if (!Regex.Match(inputSymbol, @"[0-5]|\.|,").Success)

{ return true; }

else

{ return false; }

}

#endregion

}

}