Санкт-Петербургское государственное бюджетное профессиональное образовательное учреждение

«Радиотехнический колледж»

К защите допущен

заместитель директора по УПР

\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Бессчетнова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_ г.мп

**ДИПЛОМНЫЙ ПРОЕКТ**

Тема: «Электронный журнал успеваемости»

**Специальность:** 09.02.06 «Сетевое и системное администрирование»

**Выполнил(а):** обучающийся группы 499 Евгений Андреевич Изотов

**Руководитель:** преподаватель Руслан Андреевич Маслов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

СОДЕРЖАНИЕ

[**Введение**](#_fba8uybfb4la) **3**

[**1. Теоретическая часть**](#_51p1zx3ld07p) **7**

[Параграф 1. Сбор информации.](#_mbgpu0md5qif) 7

[Параграф 2. Разработка базы данных.](#_7zvbv6rr9a0u) 8

[Параграф 3. Требования к приложению](#_yoo06h4aj23p) 13

[Параграф 4. Анализ среды разработки](#_ps1spgcbe9g3) 14

[**2. Проектирование структуры системы**](#_3rdcrjn) **20**

[2.1 Проектирование структуры базы данных](#_rzd5fvx1ahh0) 20

[2.2 Проектирование внешнего пользовательского интерфейса](#_au3wygwn12xm) 28

[**3. Программная реализация**](#_7ew5srj95ax) **31**

# **Введение**

В наше время образование очень важно. Все больше и больше дисциплин затрагиваются по мере изучения какой либо специальности. Устоявшаяся система оценивания знаний в виде оценок является важнейшим аспектом отслеживания успеваемости обучающегося. В связи с тем, что образовательные учреждения все больше и больше переходят на автоматизированную систему контроля успеваемости в виде электронного журнала темой дипломной работы стала разработка электронного журнала для ведения учета успеваемости студентов в Радиотехническом колледже.

Объектом исследования данной работы является приложение для электронного учета успеваемости студентов.

Предметом исследования является разработка приложения учета успеваемости и баз данных для приложения.

desktop приложение - это программа, обрабатываемая на стороне клиента и запускаемая в виде обыкновенного исполняемого файла на устройстве пользователя. В качестве такого устройства может быть компьютер, коммуникатор или смартфон.

Стоит разобраться в том, что представляет из себя электронный журнал.

«Электронный журнал» — электронный аналог печатного журнала, доступный для просмотра на мобильных устройствах или компьютере. Распространяется через Интернет или на любых носителях информации

Как показывается практика внедрение такого подхода в систему образования приводит к ряду преимуществ таких как:

1. Упрощение для преподавателей ведения учета оценок.
2. Надежное хранение информации.
3. Быстрая передача информации.
4. Упрощение организации учебного процесса.
5. Возможность исправлять ошибочно выставленные значения.

Приложения будет основано на устоявшейся системе учета успеваемости. Нужно реализовать интерфейс электронного журнала, для работы с оценками, которые можно добавлять, редактировать и удалять. Для работоспособности программы необходимо разработать базу данных, в которой будет храниться информация об учителях, студентах и оценках.

Использование средств автоматизации учебного процесса поможет выполнять работу значительно быстрее и без разного рода ошибок, которые неизбежны при «ручной» обработке больших объемов информации.

Электронный журнал успеваемости находится в тренде сегодняшнего времени благодаря ряду причин, включающих в себя:

1. Электронный дневник успеваемости позволяет получать и обрабатывать информацию в электронном виде, что значительно повышает эффективность учебного процесса и дает возможность своевременно реагировать на изменения в успеваемости ученика.

2. В электронном журнале учета успеваемости больше нет необходимости тратить время на заполнение бумажных бланков и вносить данные вручную, следовательно, количество ошибок при вводе данных уменьшается.

3.Возможность быстро и точно отслеживать общий прогресс класса и индивидуальный прогресс каждого ученика

4. Сокращение времени, затрачиваемого на ручное внесение данных, что позволяет учителям сконцентрироваться на обучении, не теряя время на административные задачи

Целью выпускной квалификационной работы является моделирование, проектирование и разработка электронного журнала учета успеваемости учащихся колледжа, направленной на повышение эффективности выполнения происходящих при работе с учетом оценок.

Для выполнения цели были поставлены следующие задачи:

1. Проанализировать литературу и интернет-источники по базам данных, веб-разработке, информационным системам.
2. Определение стека разработки.
3. Разработка базы данных для хранения, записи и передачи информации.
4. Разработка программного обеспечения с пользовательским интерфейсом - электронного электронного журнала.

Методами исследования были изучение литературы по разработке программных обеспечения для бизнесса. Изучение материалов по структуризации и разработке баз данных. Ознакомление с законодательством о хранении персональных данных, а также наблюдение за рабочим процессом преподавателей и администрации колледжа.

В ходе выполненной работы был разработан продукт для ведения учета успеваемости, который позволяет в цифровом виде хранить и передавать необходимую информацию. Простой интерфейс позволяет преподавателям быстро освоится с программным обеспечением и начать более продуктивную работу. В связи с появлением такого продукта в образовательном учреждении, практически отпадает нужда в больших объемах хранения информации и повышается скорость получению нужных данных непосредственно из приложения, что является неоспоримым плюсом для образовательного учреждения.

# **Теоретическая часть**

## *Параграф 1. Сбор информации.*

При разработке программного обеспечения необходимым аспектом стало ознакомления с рабочим процессом образовательного учреждения. Я советовался с рабочим персоналом колледжа. У каждого из работников были свои задачи. Для меня первостепенную значимость имело изучение правильного хранения записей. Я интересовался у сотрудников особенностями их работы. Таким образом меня ознакомили с большинством тонкостей.

Конечно немало важно было узнать, что преподавательский состав хотел видеть в приложении, ведь упрощение понимания программы пользователем является одним из первостепенных аспектов. Интересуясь у преподавателей тем , что они хотели бы увидеть, я вывел некоторую формулу разработки и придерживался ее.

Информация должны быть структурирована и расставлена так сказать по полочкам. Это повлечет за собой легкость поиска необходимых данных, что позволит быстрее ориентироваться в хранимой информации и выполнять поставленные задачи, что повлечет за собой увеличение эффективности рабочего процесса.

## *Параграф 2. Разработка базы данных.*

**База данных** — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных

За последние 30 лет в области теории систем баз данных была проведен ряд исключительно продуктивных исследований. Полученные результаты вполне можно считать наиболее важным достижением информатики за этот период. Базы данных стали основой информационных систем и в корне изменили методы работы многих организаций. В частности, в последние годы развитие технологии баз данных привело к созданию весьма мощных и удобных в эксплуатации систем. Благодаря этому системы баз данных стали доступными широкому кругу пользователей.

В базе данных сведения из каждого источника сохраняются в отдельной таблице. При работе с данными из нескольких таблиц устанавливаются связи между таблицами. Для поиска и отбора данных, удовлетворяющих определенным условиям, создается запрос. Запросы также позволяют обновить или удалить одновременно несколько записей, выполнить встроенные и специальные сообщения.

Для анализа данных или распечатки их определенным образом используются отчёты. Например, можно создать отчет, группирующий данные и подводящий итоги, или отчет для распечатки итоговых оценок.

В окне базы данных можно работать со всеми ее объектами. Для просмотра объектов определённого типа следует выбрать соответствующую вкладку. С помощью кнопок можно открывать и изменять существующие записи и создавать новые.

Разработанная база данных позволяет быстро и эффективно работать с данными. Удобный интерфейс программы, с одной стороны, позволяет легко ориентироваться в программе, не требуя от пользователя каких-либо специальных навыков работы с электронно-вычислительными машинами, с другой стороны предоставляет пользователю полную информацию обо всех заказах, а также личных данных учащихся и преподавателях.

Одна из главных проблем при выборе базы данных — это выбор между структурами данных SQL (реляционная модель) и NoSQL (нереляционная модель). Обе они обладают хорошей производительностью, но есть некоторые ключевые различия, о которых следует помнить

Базы данных SQL. Реляционная база данных — это набор таблиц, между которыми установлены определенные взаимосвязи. Для обслуживания реляционной базы данных и создания запросов к ней система управления базой данных использует язык структурированных запросов (Structured Query Language, SQL) — обычное пользовательское приложение, предоставляющее простой интерфейс программирования для взаимодействия с базой данных.

Реляционные базы данных состоят из строк, называемых кортежами, и столбцов, называемых атрибутами. Кортежи в таблице используют одни и те же атрибуты.

Реляционная база данных идеально подходит для хранения структурированных данных (почтовых кодов, номеров кредитных карточек, дат, идентификационных номеров). SQL — испытанная технология: она хорошо задокументирована, имеет прекрасную поддержку и отлично работает с большинством современных структур и библиотек. Наиболее яркими примерами баз данных SQL служат PostgreSQL и MySQL. Обе зарекомендовали себя как стабильные и безопасные.

Еще одно большое преимущество реляционных баз данных — их безопасность. Реляционные базы данных поддерживают разрешения на доступ, определяющие, кто может читать и редактировать данные. Администратор базы данных может предоставить конкретному пользователю права на доступ, выбор, вставку или удаление данных. Это защищает информацию от кражи третьими лицами.

Использование системы управления реляционными базами данных (РСУБД) защищает данные от потери и повреждения благодаря четырем свойствам ACID: атомарности, согласованности, изолированности и прочности. Чтобы лучше понять, что это значит, предположим, что две покупательницы одновременно пытаются купить красное платье одинакового размера. При соблюдении принципов ACID эти транзакции гарантированно друг друга не перекроют.

В Реляционных базах данных есть и недостатки.

* Им не хватает гибкости. Реляционные базы данных не могут эффективно работать с полуструктурированными и неструктурированными данными, поэтому они не очень подходят для больших нагрузок и аналитики Интернета вещей.
* По мере усложнения структуры данных становится все труднее передавать информацию из одного программного решения, ориентированного на большие данные, в другое. В крупных организациях реляционные базы данных часто растут независимо в отдельных подразделениях.
* Реляционные базы данных работают только на одном сервере. Следовательно, чтобы ваша СУБД справлялась с еще большим объемом данных, вам придется потратиться на дорогостоящее физическое оборудование.

Эти недостатки вынудили разработчиков искать альтернативы реляционным базам данных. В результате появились базы данных NoSQL и NewSQL.

Базы данных NoSQL, называемые также нереляционными или распределенными базами данных, служат альтернативой реляционным базам данных. Они предоставляют разработчикам большую степень гибкости и масштабируемости, поскольку в них можно хранить и обрабатывать неструктурированные данные (данные из социальных сетей, фотографии, MP3-файлы и т. д.).

Данные в нереляционных базах можно изменять динамически, не затрагивая существующие данные. Кроме того, базы данных NoSQL могут работать на нескольких серверах, поэтому масштабировать их дешевле и проще, чем базы данных SQL.

И поскольку базы данных NoSQL не зависят от одного-единственного сервера, они более отказоустойчивы. Это значит, что в случае отказа одного из компонентов база данных может продолжить работу.

Однако базы данных NoSQL еще не достигли такого уровня развития, как базы данных SQL, и среда NoSQL не так хорошо определена. К тому же базы данных NoSQL ради доступности и гибкости зачастую жертвуют соблюдением принципов ACID.

Как уже было сказано, для эффективности базы данных нужно строго фильтровать и записывать информацию.

Моя база данных состоит из таблицы преподавателей, где хранится информация о личных данных преподавателя, его предмете и подконтрольных ему групп учащихся. В свою очередь таблица студентов располагает информацией о студенте и необходимыми колоннами для связи таблицы с другими таблицами такими как таблица группы и оценок. Для того, чтобы сгруппировать студентов была создана таблица групп. Это позволяет связать студентов вместе по специальности и прочим аспектам. Что дает возможность объединения данных о обучающихся вместе студентах в группы, для простоты поиска и редактирования информации.

Для хранения оценок мною была сделана таблица оценок, которая связана с таблицей студентов. Таки образом таблица студентов освобождается от излишней перегруженности данными, что позволяет более эффективно организовать последующие работы.

## *Параграф 3. Требования к приложению*

Основной задачей проекта является автоматизация деятельности учета оценок, и упрощение работы учителей. Перед началом работы необходимо предъявить к системе конкретные требования, а также определить ее границы исходя из технического задания.

Главный документом, регламентирующим разработку электронных журналов, является письмо Минобрнауки России от 15.02.2012 № АП147/07 «О методических рекомендациях по внедрению систем ведения журналов успеваемости в электронном виде» [18]. В нем описываются полные методические рекомендации по внедрению систем ведения журналов успеваемости в электронном виде. В письме подробно описано, что обучающиеся и их родители (законные представители) должны получить доступ к актуальной и достоверной информации:

о ходе и содержании образовательной деятельности;

• сведения о результатах текущего контроля успеваемости обучающихся;

• сведения о результатах промежуточной аттестации обучающихся;

На основании этого письма были сформулированы требования к разрабатываемому продукту.

Так как электронный журнал хранит личные данные учеников, учителей и родителей (законных представителей), информацию об уроках и оценках, данные должны быть защищены по стандартам Федерального закона РФ № 152-ФЗ «О персональных данных» [23]. Обоснование выбора состава комплекса технических средств основывается на имеющихся сведениях об эксплуатации аналогичных систем. А также на требованиях, предъявляемых к техническим средствам производителями стороннего программного обеспечения используемого при функционировании системы.

## *Параграф 4. Анализ среды разработки*

Анализ выбора среды разработки является важным аспектом, который должен быть учтен при выполнении дипломной работы по специальности «Информационные системы и программирование». Это особенно важно при разработке электронного журнала успеваемости, так как выбор среды разработки может повлиять на качество и производительность приложения.

Начнем с рассмотрения основных критериев, которые должны быть учтены при выборе среды разработки. Важными факторами являются требования программиста, доступность и надежность среды разработки, поддержка используемых технологий, стоимость и уровень поддержки со стороны сообщества разработчиков.

Первым вариантом для разработки электронного дневника успеваемости может быть C#.NET, который является одним из самых популярных языков программирования в мире. Эта среда разработки обеспечивает широкие возможности по созданию и развертыванию многопоточных приложений, что может быть полезно при разработке электронного дневника успеваемости. Более того, C#.NET имеет качественную среду разработки, в которую входит интегрированная среда разработки (IDE), компилятор и отладчик.

Второй вариант можно предложить на базе Java, которая является одной из наиболее популярных сред разработки приложений. Она имеет широкий спектр устройств, на которых можно запускать Java-приложения, а также продвинутую платформу для управления памятью, что делает ее более надежной. Java имеет крупное сообщество разработчиков, что дает большое число ресурсов и ответов на вопросы.

Третий вариант может быть связан с использованием Python, который является интерпретируемым и легковесным языком программирования. Python отлично подходит для быстрого прототипирования и написания программ с небольшой потребностью в ресурсах. Он имеет широкий спектр функций и библиотек, что может быть полезным при разработке электронного дневника успеваемости. Более того, Python имеет крупное сообщество разработчиков и постоянно обновляется.

Изучив вышеприведенные варианты, можно сделать вывод, что каждая среда разработки имеет свои сильные и слабые стороны, которые следует учесть при выборе среды разработки электронного дневника успеваемости.

Для разработки электронного журнала использовался фреймворк AvaloniaUI для разработки графического интерфейса пользователя (GUI) на C#/.NET

Язык C# и платформа .NET вышли из под крыла компании Microsoft, которая никогда не отличалась уважением к мультиплатформенности. Данный язык был тепло принят разработчиками и энтузиасты проекта Mono даже сделали аналог фреймворка под Linux/Unix/Mac Os X.

Аналогично с Java, приложения на C# имеют большой минус — необходимость устанавливать фреймворк.

В остальном — сплошные плюсы, на мой взгляд.

Плюсы: мультиплатформенность, большое количество кадров, хорошая производительность, развитость фреймворка.

Минусы: необходимость установки фреймворка.

Что же нам нужно для создания первого приложения?

Во-первых, нужен текстовый редактор, в котором мы будем писать исходный код.

Во-вторых, нам понадобится компилятор, который преобразует исходный код в исполняемый файл (например, .exe-файл в операционной системе Windows).

В-третьих, нужен фреймворк .NET, который необходим для компиляции и запуска приложения.

Все элементы, необходимые для создания первого приложения, объединены в специальной программе IDE (англ. Integrated Development Environment), а именно:

* Кодовый редактор
* Транслятор (компилятор и/или интерпретатор)
* Средства автоматизированной сборки
* Отладчик

Также IDE может включать в себя средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. И это ещё не всё. Как правило, каждая IDE предназначена для нескольких языков программирования. Для разработки на языке C# можно выделить следующие среды разработки:

* Visual Studio. Разработана компанией Microsoft (так же, как и язык C#), поэтому является основной для создания приложений на C#.

Плюсы:

- Обширные возможности

- Удобство использования

- Отказоустойчивость

- Интеграция с другими системами

- Расширяемость

Минусы:

- Неэффективное использование ресурсов

- Высокая стоимость

- Сложность использования

- Не подходит для работы со всеми языками программирования

- Зависимость от платформы

* Project Rider. Это кроссплатформенная интегрированная среда разработки программного обеспечения для платформы .NET, разрабатываемая компанией JetBrains.

Плюсы:

- Кроссплатформенность

- Полная совместимость с Roslyn языков C#

- Набор готовых шаблонов и проектов

- Различные удобные функции, такие как автоматическое завершение кода, отладка и анализ кода

Минусы:

- Отсутствие некоторых функций из Visual Studio, таких как редактор WPF или [ASP.NET](http://asp.net/) Forms designer

- Большой размер загрузочного файла

- Высокая стоимость

* Sharp Developer. Свободная среда разработки для языка C#, альтернатива Visual Studio .NET.

Плюсы:

- Простота использования

- Множество инструментов и функций, которые облегчают работу программистов

- Огромное количество плагинов, которые можно использовать для расширения возможностей IDE

Минусы:

- Отсутствие поддержки нескольких языков программирования, которые поддерживаются другими free IDE

- Существуют проблемы с отображением кода на монохромных эранах

* Eclipse. Данная IDE предназначена в основном для разработки Java-приложений, но и на C# также можно создавать приложения.

Плюсы:

- Бесплатный исходный код разработки.

- Поддерживает многие языки программирования (Java, C++, Python и др.).

- Хорошая интеграция со сторонними инструментами и плагинами.

- Мощный набор функций для управления проектами.

Минусы:

- Иногда требуется дополнительная настройка для работы с некоторыми языками программирования.

- Необязательно удобен для новичков.

- Может немного тормозить при работе с крупными проектами.

* Visual Studio Code. Редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

Плюсы:

- Бесплатный исходный код разработки.

- Поддержка многих языков программирования.

- Хорошая интеграция с инструментами Git и другими.

- Небольшой и легковесный.

- Хорошие возможности расширения.

Минусы:

- Нет нативной поддержки для .NET Framework и .NET Core.

- Не столь мощный, как полная версия Visual Studio.

- Могут возникать проблемы из-за использования множества плагинов.

# **2. Проектирование структуры системы**

## **2.1 Проектирование структуры базы данных**

Первостепенной задачей при разработке журнала стало создание базы данных. Базы данных являются одной из самых важных компонент при разработке приложений. Вот почему:

1. Хранение данных. Базы данных предоставляют удобное и эффективное хранение данных. Это позволяет приложениям быстро обрабатывать и извлекать необходимую информацию.

2. Безопасность данных. Базы данных обеспечивают защиту данных с помощью системы авторизации и аутентификации, а также шифрования информации. Без баз данных приложение становится уязвимым для атак и взлома.

3. Интеграция с другими приложениями. Базы данных могут использоваться для обмена данными между различными приложениями, что упрощает интеграцию и повышает эффективность.

4. Управление информацией. Базы данных помогают управлять информацией, позволяя мониторить и анализировать данные. Это позволяет повысить эффективность приложения и ускорить процесс принятия решений.

5. Масштабируемость. С помощью баз данных приложения могут масштабироваться по мере роста требований к хранению и обработке данных. Без баз данных приложения сталкиваются с ограничениями скорости и объема работы.

6. Резервное копирование. Базы данных позволяют создавать резервные копии данных, что обеспечивает сохранность информации в случае сбоя системы. Без баз данных приложение может потерять ценную информацию.

В целом, базы данных являются неотъемлемой частью разработки приложений. Они обеспечивают хранение, защиту и управление данными, а также упрощают интеграцию с другими приложениями и повышают эффективность приложения.

На рисунке 1 представлена ERD-схема базы данных для приложения электронный журнал.

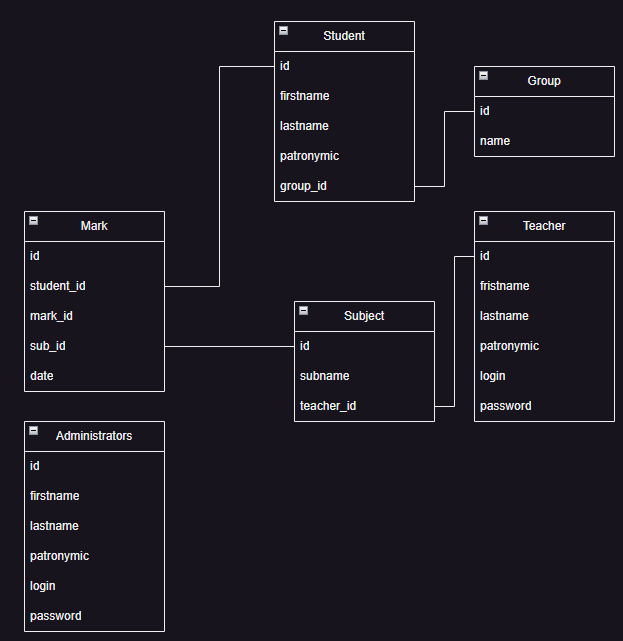


Рисунок 1 - ERD-схема базы данных

Моя база данных состоит из следующих таблиц:

В таблице 1 представлена структура хранения данных о студенте.

Таблица включает в себя ФИО студента, колонну “group\_id” необходимую для того, чтобы включить студента в соответствующую группу. “Role\_id” - это колонна для определения роли, под которой будет авторизовываться студент.

Таблица 1. Student

| Идентификатор | Тип данных | Описание |
| --- | --- | --- |
| Id | Serial4 | Идентификационный номер студента |
| Firstname | varchar | Фамилия студента |
| Lastname | varchar | Имя студента |
| patronymic | varchar | Отчество студента |
| group\_id | int4 | Идентификационный номер группы, в которой состоит студент |
| role\_id | int4 | Идентификационный номер роли |
| login | varchar | Логин для авторизации в журнале |
| password | varchar | Пароль для авторизации в журнале |

Serial4 - представляет автоинкрементирующееся числовое значение, которое занимает 4 байта и может хранить числа от 1 до 2147483647. Значение данного типа образуется путем автоинкремента значения предыдущей строки. Поэтому, как правило, данный тип используется для определения идентификаторов строки.

varchar - это строковый тип данных переменной длины, поэтому он содержит только те символы, которые вы ему назначите.

int4 - это сокращение от Integer (целый с английского), что как бы намекает, что этот тип позволяет хранить целые числа.

Переменные типа int способны хранить целые числа в диапазоне от -2 миллиарда до +2 миллиарда. Или, если быть более точным, то от -2,147,483,648 до 2,147,483,647.

В таблице 2 представлена структура хранения данных таблицы «Groups». Таблица будет включать в себя данные о группе.

Таблица 2. Group

| Идентификатор | Тип данных | Описание |
| --- | --- | --- |
| Id | Serial4 | Идентификационный номер группы |
| name | varchar | Название группы |

В таблице 3 представлена структура хранения данных таблицы «Teacher».

Таблица включает в себя ФИО преподавателя. Логин и пароль для авторизации в журнале.

Таблица 3. Teacher

| Идентификатор | Тип данных | Описание |
| --- | --- | --- |
| Id | Serial4 | Идентификационный номер преподавателя |
| Firstname | varchar | Фамилия преподавателя |
| Lastname | varchar | Имя преподавателя |
| Patronymic | varchar | Отчество преподавателя |
| Login | varchar | Логин для авторизации в журнале |
| Password | varchar | Пароль для авторизации в журнале |

В таблице 4 представлена структура хранения данных таблицы “Subject”. В этой таблице представлена информация о предметах, которые изучают учащиеся Эта таблица используется для связи с таблицей Marks, где хранятся оценки учащихся по каждому предмету. Эти таблицы связаны между собой ключами, что обеспечивает целостность и согласованность данных в базе.

Таблица 4. Subject

| Идентификатор | Тип данных | Описание |
| --- | --- | --- |
| Id | Serial4 | Идентификационный номер дисциплины |
| name | varchar | Название дисциплины |
| teacher\_id | int4 | Идентификационный номер прподавателя. |

В таблице 5 представлена структура хранения данных таблицы “Mark”

Таблица Mark позволяет хранить информацию о том, какую оценку получил каждый учащийся в каждом предмете. Это позволяет анализировать успеваемость каждого ученика, а также сравнивать результаты по предметам и классам. Таблица связана с таблицами Student и Subject, чтобы получить информацию о конкретных учениках и предметах.

Таким образом колонка “student\_id” указывает на студента, которому была выставлена оценка, в колонку “mark\_id” записывается оценка, которую выставили студенту, а колонка “Id”, которая является PRIMARY\_KEY задает уникальный номер выставленной оценки, нужно это для того, чтобы при одинаковых оценках у одного студента они не читались, как одно значение.

Таблица 5. Mark

| Идентификатор | Тип данных | Описание |
| --- | --- | --- |
| id | Serial4 | Идентификационный номер оценки |
| student\_id | int4 | Идентификационный номер студента |
| mark\_id | int4 | Оценка |
| sub\_id | int4 | Идентификационный номер предмета |
| date | timestamp | Дата поставленной оценки |

В таблице 6 представлена структура хранения данных таблицы “Administratos”

Таблица “Administratos” предназначена для хранения данных об администраторах.

Таблица 7. Assoc

| Идентификатор | Тип данных | Описание |
| --- | --- | --- |
| Id | serial4 | Идентификационный номер |
| Firstname | varchar | Фамилия администратора |
| Lastname | varchar | Имя администратора |
| Patronymic | varchar | Отчество администратора |
| login | varchar | логин для входа администратора |
| password | varchar | пароль для входа администратора |

## **2.2 Проектирование внешнего пользовательского интерфейса**

Основной внешний пользовательский интерфейс программного приложения будет представлять себя в виде окон, разработанных при помощи AvaloniaUI и языка XAML.

Первым окном в приложении является окно авторизации. Окно состоит из полей для ввода данных, выполненных при помощи TextBox

TextBox - Элемент графического интерфейса пользователя, предназначенный для ввода небольшого объёма текста без переноса строк. А также кнопки входа.

После авторизации, преподаватель попадет на окно со списком электронного журнала. В окне расположена таблица в которой находятся следующие колонки:

1. ФИО студента.

* В колонки предназначенные для фио студента выводится информация о студенте из базы данных.

1. Рейтинг ученика.

* После колонок ФИО идет колонка с рейтингом ученика, в которой происходит расчет среднего балла учащегося, так же в колонке расположения линия, которая меняет свой цвет в зависимости от среднего балла.

1. Черный - балл ниже 2(Очень плохо)
2. Серый - балл ниже 3(Плохо)
3. Красный - балл ниже 4(Удовлетворительно)
4. Желтый - балл ниже 4.5(Хорошо)
5. Зеленый - балл выше 4.5(Отлично)
6. Сетка оценок по дням недели.

* После колонки рейтинга идут 5 колонок, соответствующие 5 дням недели от Понедельника до Пятница. В зависимости от дня недели и самой недели, о расчете которой я расскажу далее, поставленная оценка сохраняться в бд в соответствии с датой.

Над сеткой учеников расположены элементы управления, которые помогают перемещаться между группами, предметами и неделями.

Фильтрация по группе и предмету выполнены с помощью элемента ComboBox.

Фильтрация по группе осуществляет обновление списка учеников в соответствии с группой в которой они состоят. Фильтрация по предмету позволяет выбрать предмет, по которому будут осуществляться действия с оценками.

ComboBox - Элемент управления который предоставляет пользователям список вариантов. Список отображается и скрывается при разворачивании и сворачивании элемента управления соответственно. В состоянии по умолчанию список свернут, отображается только один вариант. Для просмотра полного списка вариантов пользователю нужно нажать сам элемент.

В окне также система перемещения между неделями. Выполненная с помощью 2 кнопок (Назад и Вперед) и элемента TextBlock, в который выводится даты в диапазоне 7 дней.

Последним элементом в это окне преподавателя является кнопка выхода из аккаунта.

При авторизации администратора открывается окно с меню добавления новых данных. Например нового студента, новой группы, нового преподавателя. Ориентирование по этим атрибутам выполнено с помощью кнопок, при нажатии на одну из которых, открывается окно добавления. Задний фон выполнен с помощью функции установки заднего фона “Background” в DataGrid - наиболее мощном и часто используемом контейнере, напоминающем обычную таблицу. Он содержит столбцы и строки, количество которых задает разработчик.

# **3. Программная реализация**

При открытии приложения отображается страница авторизации. Страница авторизации представлена на рисунке 2. Пользователь вводит свой логин и пароль, которые могут состоять из букв, цифр и других символов после чего получает доступ к дальнейшей работе с приложением. В зависимости от роли пользователя(администратор или учитель) он переходит на соответствующее окно.

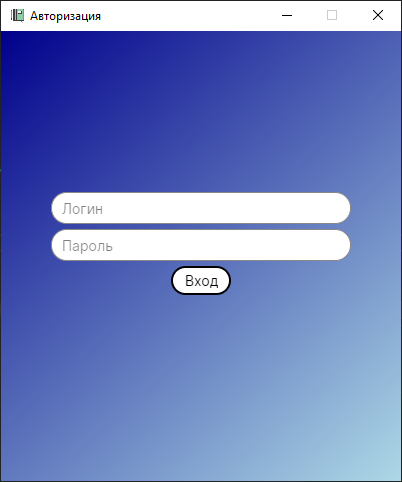


Рисунок 2 - Авторизация.

**Интерфейс Администратора**

На рисунке 3 показано окно администратора. После авторизации администратор попадает в окно администратора. В данном окне представлены кнопки добавления новых учителей, преподавателей и групп, а также выводится фамилия и имя администратора. Окно администратора нужно для того, чтобы вносить изменения в базу данных без непосредственного взаимодействия с ней, что сокращает время работы и повышает продуктивность. При нажатии кнопку “Добавить студента” откроется окошко, в котором расположены поля для ввода данных нового студента. Кроме добавления студента, администратор имеет функцию добавления новых учителей и групп. Кнопка “Выйти из аккаунта” позволяет администратору разлогиниться, после нажатия на кнопку пользователя перенесет на страницу авторизации.

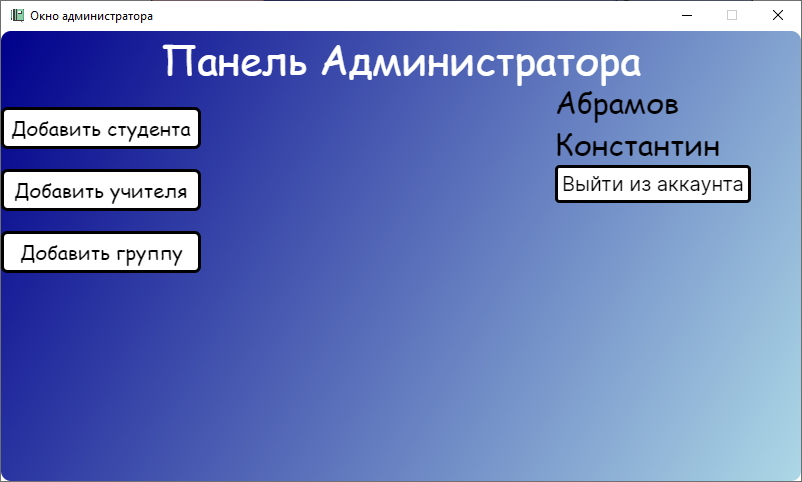


Рисунок 3 - Окно Администратора

На рисунке 4 показан функционал окна добавления студентов.

После заполнения полей данными и нажатия на кнопку сохранения, новый студент вносится в базу данных. После сохранения студента не нужно заново нажимать на кнопку “Добавить студента”. Окно не закрывается после сохранения, что позволяет быстро добавлять новых студентов.

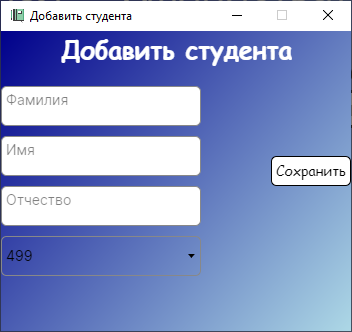


Рисунок 4 - Окно добавления студентов

На рисунке 5 показан функционал окна добавления преподавателей.

После заполнения полей данными и нажатия на кнопку сохранения, новый преподаватель вносится в базу данных. После сохранения студента не нужно заново нажимать на кнопку “Добавить учителя”. Окно не закрывается после сохранения, что позволяет быстро добавлять новых учителей.

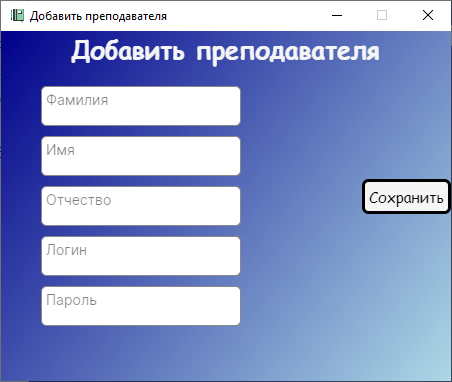


Рисунок 5 - Окно добавления преподавателя

На рисунке 6 показан функционал окна добавления группы.

После заполнения полей данными и нажатия на кнопку сохранения, новая группа вносится в базу данных.

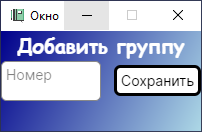


Рисунок 6 - Окно добавления группы

**Интерфейс преподавателя**

Интерфейс преподавателя является основой электронного журнала, ведь ради этого окна и делалось приложение. Итак, после авторизации преподаватель попадает на окно преподавателя, где он может совершать действия по добавлению, редактированию и удалению оценок. Кроме того, приложение позволяет быстро ознакомиться с текущей успеваемости студента или группы в целом, что дает возможность быстрее ориентироваться при составлении отчетов.

Сетка студентов с оценками выполнена с помощью элемента DataGrid.

Элемент DataGrid — это элемент управления, отображающий данные в виде сетки. Колонки студентов не подлежат редактированию, в то время как с колонками предназначенными для оценок можно взаимодействовать. При нажатии на ячейку для оценок, в нее можно поставить оценку.

На рисунке 7 представлено окно преподавателя. В левом верхнем углу расположена фильтрация по группе и предмету, при нажатии на элемент фильтрации выпадает список групп или предметов, при нажатии на элемент списка обновляется таблица оценок в зависимости от выбранного элемента.

В правом верхнем углу расположен элемент смены недели, при нажати на кнопку “**←**” и “**→**” происходит смена недели, вместе с этим меняются оценки на в зависимости от даты, когда стоит оценка.

Далее переходим к таблице с студентами и оценками. ФИО студентов выводится из бд в зависимости от установленной группы. После ФИО идет колонка со средним баллом, в нее выводится средний балл студента за весь промежуток учебного года, что позволяет быстрее сориентироваться с итоговой оценкой, цвет полосы меняется в зависимости от среднего балла от черного(Очень плохо) до зеленого(Отлично). Следующие 5 колонок это колонки с оценками. При нажатии на ячейку она становится текстовым полем куда оценки можно ставить удалять и изменять. Оценки ставятся только в диапазоне от 1 до 5, при попытке написать другие цифры или буквы, содержимое ячейки не будет обновляться.

При выставлении оценки она сохраняется в базу данных, а так как в базе данных есть поле с датой, то программа рассчитывает стоящую неделю и день, в который внесли оценку и сохраняет соответствующую дату.

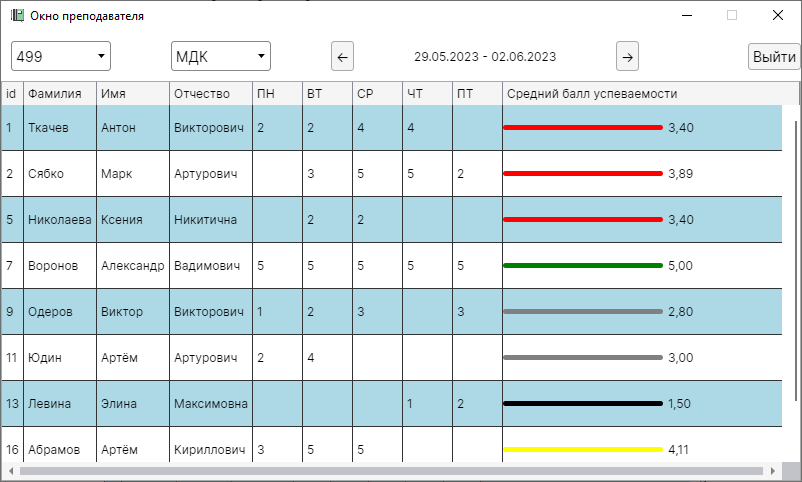


Рисунок 7 - Окно электронного журнала.**Заключение**

В данной работе была решена задача разработки электронного журнала учета успеваемости для учебного заведения. Была проведена аналитическая работа, изучены существующие аналоги и определены требования к системе.

На основе этой информации была разработана архитектура приложения и выбраны технологии для его реализации. Была произведена разработка модулей электронного журнала и реализация его основных функций, таких как взаимодействие с интерфейсом для добавления оценок, добавление учеников и учителей в базу данных.

Результат работы может быть использован в образовательном учреждении для автоматизации процессов учета успеваемости и упрощения жизни учителям. В целом, данная работа является актуальной и достаточно полной. Она имеет большой потенциал для дальнейшего развития, так как есть возможность расширять функционал и добавлять новые модули для улучшения пользовательского опыта.

В рамках выпускной квалификационной работы был разработана система электронный журнал учета успеваемости учащихся колледжа. В результате работы были разработаны:

• структура информационной системы;

• элементы графического интерфейса: кнопки, меню и т.д.;

• вид главной страницы, страницы просмотра и выставления оценок,

• инструкция по работе.

В результате проделанной работы были решены следующие задачи:

1. Проанализирована литература и интернет-источники по базам данных, веб-разработке, информационным системам.

2. Проанализированы нормативные документы по разработке и внедрению электронных журналов.

3. Разработан электронный журнал учета успеваемости учащихся колледжа.

4. Разработана инструкция по использованию электронного журнала. В ходе решения задачи по анализу литературы и интернет-источников по базам данных, веб-разработке с целью формирования умений реализации информационных систем были проанализированы учебники и интернет источники по разработке на языке C# и кроссплатформенной платформе .NET - Avalonia

СПИСОК ЛИТЕРАТУРЫ

Учебная литература:

1. Джоэл Мертц. Programming Windows Store Apps with HTML, CSS, and JavaScript. O'Reilly Media, 2013. – 479 с.
2. Джоэл Мертц. Programming Windows: Writing Windows 8 Apps with C# and XAML. O'Reilly Media, 2013. – 680 с.
3. Джозеф Албахари, Бен Албахари. C# 7.0 in a Nutshell: The Definitive Reference. O'Reilly Media, 2017. – 1088 с.
4. Джон Скит. SQL in a Nutshell: A Desktop Quick Reference. O'Reilly Media, 2008. – 594 с.
5. Бен Форта. Sams Teach Yourself SQL in 10 Minutes. Sams Publishing, 2012. – 256 с.
6. Джон Скит. C# 5.0 in a Nutshell: The Definitive Reference. O'Reilly Media, 2012. – 1064 с.
7. Джоэл Мертц. Programming C# 5.0: Building Windows 8, Web, and Desktop Applications for the .NET 4.5 Framework. O'Reilly Media, 2012. – 824 с.
8. Джоэл Мертц. Programming Entity Framework: DbContext. O'Reilly Media, 2012. – 176 с.
9. Джоэл Мертц. Programming ASP.NET MVC 4: Developing Real-World Web Applications with ASP.NET MVC. O'Reilly Media, 2012. – 322с.
10. Джоэл Мертц. Programming Windows Store Apps with C#. O'Reilly Media, 2013. – 508 с.

**Листинг**

1. Листинг - исходный код окна авторизации

public partial class Auth : Window

{

public Auth()

{

InitializeComponent();

var gradient = new LinearGradientBrush

{

GradientStops = new GradientStops

{

new GradientStop(Colors.DarkBlue, 0.0),

new GradientStop(Colors.LightBlue, 1.0)

},

StartPoint = new RelativePoint(0, 0, RelativeUnit.Relative),

EndPoint = new RelativePoint(1, 1, RelativeUnit.Relative)

};

this.Background = gradient;

EnterButton.Click += EnterButton\_Click;

}

private void EnterButton\_Click(object? sender, Avalonia.Interactivity.RoutedEventArgs e)

{

var admins = Helper.db.Administratos.FirstOrDefault(x => x.Login == logb.Text);

var users = Helper.db.Teachers.FirstOrDefault(x => x.Login == logb.Text);

if (users != null && users.Password == passb.Text)

{

MainWindow win = new MainWindow();

win.Show();

this.Close();

}

else if (users == null && admins != null && admins.Password == passb.Text)

{

int names = admins.Id;

adminname.name = names;

AdminWindow adminWindow = new AdminWindow();

adminWindow.Show();

this.Close();

}

}

}

2) Листинг - исходный код окна администратора

public partial class AdminWindow : Window

{

public Administrato admin;

public AdminWindow()

{

InitializeComponent();

WindowStyle();

Loaddata();

admin = new Administrato();

DataContext = admin;

addstud.Click += Addstud\_Click;

addteach.Click += Addteach\_Click;

addgroup.Click += Addgroup\_Click;

exit.Click += Exit\_Click;

}

private void Addgroup\_Click(object? sender, RoutedEventArgs e)

{

AddGroup addGroup = new AddGroup();

addGroup.Show();

}

private void Exit\_Click(object? sender, RoutedEventArgs e)

{

Auth win = new Auth();

win.Show();

this.Close();

}

private void Addteach\_Click(object? sender, RoutedEventArgs e)

{

AddTeacher teacher = new AddTeacher();

teacher.Show();

}

private void Addstud\_Click(object? sender, RoutedEventArgs e)

{

AddStudent win = new AddStudent();

win.Show();

}

public void WindowStyle()

{

//Первый бордер

var gradient = new LinearGradientBrush

{

GradientStops = new GradientStops

{

new GradientStop(Colors.DarkBlue, 0.0),

new GradientStop(Colors.LightBlue, 1.0)

},

StartPoint = new RelativePoint(0, 0, RelativeUnit.Relative),

EndPoint = new RelativePoint(1, 1, RelativeUnit.Relative)

};

bord.Background = gradient;

}

public void Loaddata()

{

try

{

int admin = adminname.name;

var query = Helper.db.Administratos.Find(admin);

surname.Text = query.Firstname;

name.Text = query.Lastname;

}

catch

{

throw;

}

addstud.BorderBrush = new SolidColorBrush(Colors.Red);

addstud.Background = new SolidColorBrush(Colors.Pink);

addteach.BorderBrush = new SolidColorBrush(Colors.Red);

addteach.Background = new SolidColorBrush(Colors.Pink);

}

}

3) Листинг - исходный код окна добавления студентов

public partial class AddStudent : Window

{

public Student student;

public AddStudent()

{

InitializeComponent();

var gradient = new LinearGradientBrush

{

GradientStops = new GradientStops

{

new GradientStop(Colors.DarkBlue, 0.0),

new GradientStop(Colors.LightBlue, 1.0)

},

StartPoint = new RelativePoint(0, 0, RelativeUnit.Relative),

EndPoint = new RelativePoint(1, 1, RelativeUnit.Relative)

};

maingrid.Background = gradient;

student = new Student();

DataContext = student;

Save.Click += Save\_Click;

group.SelectionChanged += Group\_SelectionChanged;

List<Group> gr = new List<Group>();

gr = Helper.db.Groups.ToList();

group.Items = gr.Select(x => x.Name);

}

private void Group\_SelectionChanged(object? sender, SelectionChangedEventArgs e)

{

}

private void Save\_Click(object? sender, Avalonia.Interactivity.RoutedEventArgs e)

{

try

{

if (surname.Text == null || name.Text == null || patr.Text == null)

{

error.Text = "Кажется... Вы заполнили не все поля";

}

else

{

Helper.db.Students.Add(student);

Helper.db.SaveChanges();

}

}

catch

{

}

}

}

4) Листинг - исходный код окна добавления учителя

public Teacher teacher;

public AddTeacher()

{

InitializeComponent();

teacher = new Teacher();

DataContext = teacher;

Save.Click += Save\_Click;

List<Subject> subs = new List<Subject>();

subs = Helper.db.Subjects.ToList();

var gradient = new LinearGradientBrush

{

GradientStops = new GradientStops

{

new GradientStop(Colors.DarkBlue, 0.0),

new GradientStop(Colors.LightBlue, 1.0)

},

StartPoint = new RelativePoint(0, 0, RelativeUnit.Relative),

EndPoint = new RelativePoint(1, 1, RelativeUnit.Relative)

};

maingrid.Background = gradient;

}

private void Save\_Click(object? sender, Avalonia.Interactivity.RoutedEventArgs e)

{

try

{

if (surname.Text == null || name.Text == null || patr.Text == null)

{

error.Text = "Кажется... Вы заполнили не все поля";

}

else

{

Helper.db.Teachers.Add(teacher);

Helper.db.SaveChanges();

}

}

catch

{

}

}

5) Листинг - исходный код окна добавления группы.

public Group group;

public AddGroup()

{

InitializeComponent();

Save.Click += Save\_Click;

group = new Group();

DataContext = group;

var gradient = new LinearGradientBrush

{

GradientStops = new GradientStops

{

new GradientStop(Colors.DarkBlue, 0.0),

new GradientStop(Colors.LightBlue, 1.0)

},

StartPoint = new RelativePoint(0, 0, RelativeUnit.Relative),

EndPoint = new RelativePoint(1, 1, RelativeUnit.Relative)

};

maingrid.Background = gradient;

}

private void Save\_Click(object? sender, Avalonia.Interactivity.RoutedEventArgs e)

{

Helper.db.Groups.Add(group);

Helper.db.SaveChanges();

}

6) Листинг - исходный код окна преподавателя

using Avalonia.Controls;

using dict.db;

using System;

using System.Collections.Generic;

using System.Linq;

using Npgsql;

using System.Runtime.Intrinsics.Arm;

using Avalonia;

using static System.Net.Mime.MediaTypeNames;

using Avalonia.Media;

namespace dict

{

public partial class MainWindow : Window

{

DateTime pn = new DateTime();

DateTime pt = new DateTime();

public MainWindow()

{

InitializeComponent();

Init();

Data();

LoadData();

}

public void Init()

{

var groups = Helper.db.Groups.ToList();

Group.Items = groups.Select(x => x.Name).ToList();

Group.SelectedIndex = 0;

var subs = Helper.db.Subjects.ToList();

Sub.Items = subs.Select(x => x.Subname).ToList();

Sub.SelectedIndex = 0;

Group.SelectionChanged += Group\_SelectionChanged;

Sub.SelectionChanged += Sub\_SelectionChanged;

btnleft.Click += Btnleft\_Click;

btnright.Click += Btnright\_Click;

StudentDG.CellEditEnding += StudentDG\_CellEditEnding;

}

private void StudentDG\_CellEditEnding(object? sender, DataGridCellEditEndingEventArgs e)

{

if (e.EditAction == DataGridEditAction.Commit)

{

int rowIndex = e.Row.GetIndex();

string ColumnIndex = e.Column.GetValue(DataGridColumn.HeaderProperty).ToString();

var item = StudentDG.Items.Cast<Myclass>().ElementAt(rowIndex);

int qwe = 0;

string str = (e.EditingElement as TextBox).Text;

if (str == "")

{

err.Text = "123";

}

else if(!int.TryParse(str, out qwe))

{

return;

}

if (str == "")

{

}

else if(qwe > 5 || qwe < 1)

{

(e.EditingElement as TextBox).Text = "";

return;

}

if (ColumnIndex.Trim().ToLower() == "пн")

{

Mark mk = Helper.db.Marks.FirstOrDefault(x => x.StudentId == item.StudentId && x.SubId == item.SubId && x.Date == item.Datemarkpn);

if (mk != null)

{

if (str == "")

{

mk.MarkId = null;

Helper.db.SaveChanges();

}

else

{

mk.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

Helper.db.SaveChanges();

}

}

else

{

Mark zxc = new Mark();

zxc.StudentId = item.StudentId;

zxc.SubId = Sub.SelectedIndex + 1;

zxc.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

zxc.Date = pn.ToUniversalTime();

Helper.db.Marks.Add(zxc);

Helper.db.SaveChanges();

}

}

else if (ColumnIndex.Trim().ToLower() == "вт")

{

Mark mk = Helper.db.Marks.FirstOrDefault(x => x.StudentId == item.StudentId && x.SubId == item.SubId && x.Date == item.Datemarkvt);

if (mk != null)

{

if (str == "")

{

mk.MarkId = null;

Helper.db.SaveChanges();

}

else

{

mk.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

Helper.db.SaveChanges();

}

}

else

{

Mark zxc = new Mark();

zxc.StudentId = item.StudentId;

zxc.SubId = Sub.SelectedIndex + 1;

zxc.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

zxc.Date = pn.AddDays(1).ToUniversalTime();

Helper.db.Marks.Add(zxc);

Helper.db.SaveChanges();

}

}

else if (ColumnIndex.Trim().ToLower() == "ср")

{

Mark mk = Helper.db.Marks.FirstOrDefault(x => x.StudentId == item.StudentId && x.SubId == item.SubId && x.Date == item.Datemarksr);

if (mk != null)

{

if (str == "")

{

mk.MarkId = null;

Helper.db.SaveChanges();

}

else

{

mk.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

Helper.db.SaveChanges();

}

}

else

{

Mark zxc = new Mark();

zxc.StudentId = item.StudentId;

zxc.SubId = Sub.SelectedIndex + 1;

zxc.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

zxc.Date = pt.AddDays(-2).ToUniversalTime();

Helper.db.Marks.Add(zxc);

Helper.db.SaveChanges();

}

}

else if (ColumnIndex.Trim().ToLower() == "чт")

{

Mark mk = Helper.db.Marks.FirstOrDefault(x => x.StudentId == item.StudentId && x.SubId == item.SubId && x.Date == item.Datemarkth);

if (mk != null)

{

if (str == "")

{

err.Text = "zxc";

mk.MarkId = null;

Helper.db.SaveChanges();

}

else

{

mk.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

Helper.db.SaveChanges();

}

}

else

{

Mark zxc = new Mark();

zxc.StudentId = item.StudentId;

zxc.SubId = Sub.SelectedIndex + 1;

zxc.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

zxc.Date = pt.AddDays(-1).ToUniversalTime();

Helper.db.Marks.Add(zxc);

Helper.db.SaveChanges();

}

}

else

{

Mark mk = Helper.db.Marks.FirstOrDefault(x => x.StudentId == item.StudentId && x.SubId == item.SubId && x.Date == item.Datemarkft);

if (mk != null)

{

if (str == "")

{

mk.MarkId = null;

Helper.db.SaveChanges();

}

else

{

mk.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

Helper.db.SaveChanges();

}

}

else

{

Mark zxc = new Mark();

zxc.StudentId = item.StudentId;

zxc.SubId = Sub.SelectedIndex + 1;

zxc.MarkId = Convert.ToInt16((e.EditingElement as TextBox).Text);

zxc.Date = pt.ToUniversalTime();

Helper.db.Marks.Add(zxc);

Helper.db.SaveChanges();

}

}

LoadData();

}

}

private void Btnright\_Click(object? sender, Avalonia.Interactivity.RoutedEventArgs e)

{

pn = pn.AddDays(7);

pt = pt.AddDays(7);

Datetxt.Text = pn.ToShortDateString() + " - " + pt.ToShortDateString();

LoadData();

}

private void Btnleft\_Click(object? sender, Avalonia.Interactivity.RoutedEventArgs e)

{

pn = pn.AddDays(-7);

pt = pt.AddDays(-7);

Datetxt.Text = pn.ToShortDateString() + " - " + pt.ToShortDateString();

LoadData();

}

private void Data()

{

string day = DateTime.Today.DayOfWeek.ToString().ToLower().Trim();

switch (day)

{

case "monday":

pn = DateTime.Today;

pt = DateTime.Today.AddDays(5);

Datetxt.Text = pn.ToShortDateString() + " - " + pt.ToShortDateString();

break;

case "tuesday":

pn = DateTime.Today.AddDays(-1);

pt = DateTime.Today.AddDays(3);

Datetxt.Text = pn.ToShortDateString() + " - " + pt.ToShortDateString();

break;

case "wednesday":

pn = DateTime.Today.AddDays(-2);

pt = DateTime.Today.AddDays(2);

Datetxt.Text = pn.ToShortDateString() + " - " + pt.ToShortDateString();

break;

case "thursday":

pn = DateTime.Today.AddDays(-3);

pt = DateTime.Today.AddDays(1);

Datetxt.Text = pn.ToShortDateString() + " - " + pt.ToShortDateString();

break;

case "friday":

pn = DateTime.Today.AddDays(-4);

pt = DateTime.Today;

Datetxt.Text = pn.ToShortDateString() + " - " + pt.ToShortDateString();

break;

case "saturday":

pn = DateTime.Today.AddDays(-5);

pt = DateTime.Today.AddDays(-1);

Datetxt.Text = pn.ToShortDateString() + " - " + pt.ToShortDateString();

break;

case "sunday":

pn = DateTime.Today.AddDays(-6);

pt = DateTime.Today.AddDays(-2);

Datetxt.Text = pn.ToShortDateString() + " - " + pt.ToShortDateString();

break;

}

}

private void Sub\_SelectionChanged(object? sender, SelectionChangedEventArgs e)

{

LoadData();

}

private void Group\_SelectionChanged(object? sender, SelectionChangedEventArgs e)

{

LoadData();

}

public async void LoadData()

{

string connectionString = "Host=localhost;Database=diplomdb;userid=postgres;password=1q3e2w4r";

string sqlExpression = $"Select mark.id, mark.mark\_id, mark.sub\_id, mark.\"date\", s.id, s.firstname, s.lastname, s.patronymic, s.group\_id \r\nfrom mark \r\nfull join student s on s.id = mark.student\_id\r\nwhere ((mark.\"date\" BETWEEN \'{pn}\' and \'{pt}\') or mark.\"date\" is null) and mark.sub\_id ={Sub.SelectedIndex + 1} and s.group\_id = {Group.SelectedIndex + 1}\r\norder by s.id, mark.\"date\"\r\n";

List<Myclass> students = Helper.db.Students.Where(y => y.GroupId == Group.SelectedIndex + 1).Select(x => new Myclass()

{

StudentId = x.Id,

Firstname = x.Firstname,

Lastname = x.Lastname,

Patronymic = x.Patronymic,

GroupId = x.GroupId,

cbox = Sub.SelectedIndex + 1

}).ToList();

using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))

{

await connection.OpenAsync();

NpgsqlCommand command = new NpgsqlCommand(sqlExpression, connection);

using (NpgsqlDataReader reader = await command.ExecuteReaderAsync())

{

if (reader.HasRows) // если есть данные

{

int i = 0;

while (await reader.ReadAsync()) // построчно считываем данные

{

if (i != reader.GetInt16(4))

{

i = reader.GetInt16(4);

Myclass stud = students.FirstOrDefault(x => x.StudentId == i);

if (reader.IsDBNull(0))

{

stud.Id = null;

stud.SubId = null;

stud.cbox = Sub.SelectedIndex + 1;

}

else

{

stud.Id = reader.GetInt16(0);

stud.SubId = reader.GetInt16(2);

stud.cbox = Sub.SelectedIndex + 1;

DateTime dt = reader.GetDateTime(3);

switch (dt.DayOfWeek.ToString().Trim().ToLower())

{

case "monday":

if (reader.IsDBNull(1))

{

stud.markpn = null;

}

else

{

stud.markpn = reader.GetInt16(1);

}

stud.Datemarkpn = reader.GetDateTime(3).ToUniversalTime();

break;

case "tuesday":

if (reader.IsDBNull(1))

{

stud.markvt = null;

}

else

{

stud.markvt = reader.GetInt16(1);

}

stud.Datemarkvt = reader.GetDateTime(3).ToUniversalTime();

break;

case "wednesday":

if (reader.IsDBNull(1))

{

stud.marksr = null;

}

else

{

stud.marksr = reader.GetInt16(1);

}

stud.Datemarksr= reader.GetDateTime(3).ToUniversalTime();

break;

case "thursday":

if (reader.IsDBNull(1))

{

stud.markth = null;

}

else

{

stud.markth = reader.GetInt16(1);

}

stud.Datemarkth= reader.GetDateTime(3).ToUniversalTime();

break;

case "friday":

if (reader.IsDBNull(1))

{

stud.markft = null;

}

else

{

stud.markft = reader.GetInt16(1);

}

stud.Datemarkft = reader.GetDateTime(3).ToUniversalTime();

break;

}

}

}

else

{

Myclass stud = students.FirstOrDefault(x => x.StudentId == i);

DateTime dt = reader.GetDateTime(3);

stud.cbox = Sub.SelectedIndex + 1;

switch (dt.DayOfWeek.ToString().Trim().ToLower())

{

case "monday":

if (reader.IsDBNull(1))

{

stud.markpn = null;

}

else

{

stud.markpn = reader.GetInt16(1);

}

stud.Datemarkpn = reader.GetDateTime(3).ToUniversalTime();

break;

case "tuesday":

if (reader.IsDBNull(1))

{

stud.markvt = null;

}

else

{

stud.markvt = reader.GetInt16(1);

}

stud.Datemarkvt = reader.GetDateTime(3).ToUniversalTime();

break;

case "wednesday":

if (reader.IsDBNull(1))

{

stud.marksr = null;

}

else

{

stud.marksr = reader.GetInt16(1);

}

stud.Datemarksr = reader.GetDateTime(3).ToUniversalTime();

break;

case "thursday":

if (reader.IsDBNull(1))

{

stud.markth = null;

}

else

{

stud.markth = reader.GetInt16(1);

}

stud.Datemarkth = reader.GetDateTime(3).ToUniversalTime();

break;

case "friday":

if (reader.IsDBNull(1))

{

stud.markft = null;

}

else

{

stud.markft = reader.GetInt16(1);

}

stud.Datemarkft = reader.GetDateTime(3).ToUniversalTime();

break;

}

}

}

}

}

await connection.CloseAsync();

}

StudentDG.Items = students;

}

}

}