Министерство общего и профессионального образования
Ростовской области
государственное бюджетное профессиональное образовательное учреждение
Ростовской области
«Волгодонский техникум информационных технологий, бизнеса и дизайна
имени В.В. Самарского»

УТВЕРЖДАЮ:
И.о. зам. директора по УР
_____ О.А. Морозова
_____ _____ 2023 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему Разработка программного модуля просмотр оценок для МБОУ «Гимназия «Шанс»

Специальность 09.02.07 Информационные системы и программирование

Студент	_____ (подпись)	Д.Д. Беседа
Руководитель проекта	_____ (подпись)	Ю.В. Зиненко
Консультант по экономической части	_____ (подпись)	Е.А. Галицына
Нормоконтроль	_____ (подпись)	И.Н. Власенко

Защищен с оценкой _____
Протокол № _____
от ___ _____ 2023 г.

Волгодонск
2023

Министерство общего и профессионального образования
Ростовской области
государственное бюджетное профессиональное образовательное учреждение
Ростовской области
«Волгодонский техникум информационных технологий, бизнеса и дизайна
имени В.В. Самарского»

ОДОБРЕНО:
цикловой комиссией профессионального информационного цикла
Протокол № 6 от «23» января 2023 г.

Председатель ЦК _____ / Р.В. Ромашов /

УТВЕРЖДАЮ:
И.о. зам. директора по УР

_____ / О.А. Морозова /

ЗАДАНИЕ
НА ВЫПОЛНЕНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студента Беседа Дмитрия Денисовича

Специальность 09.02.07 Информационные системы и программирование

1. Тема: Разработка программного модуля просмотр оценок для МБОУ «Гимназия «Шанс» утверждена приказом по техникуму №147 (ИСП-19)/148 (ИСП-19К) от «8» февраля 2023 г.

2. Срок сдачи законченной работы: 10 июня 2023 г.

3. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):
- анализ предметной области;
- разработка технического задания;
- проектирование программного продукта;
- разработка программного продукта.

4. Перечень графического материала (с точным указанием обязательных чертежей)

5. Руководитель: преподаватель, Зиненко Ю.В.
(должность, фамилия, инициалы, подпись)

Задание получил «1» марта 2023 г. _____ / Д.Д. Беседа /
(подпись)

ОДОБРЕНО:
цикловой комиссией профессионального информационного цикла
Протокол № 6 от «23» января 2023 г.

Председатель ЦК _____ / Р.В. Ромашов /

УТВЕРЖДАЮ:
И.о. зам. директора по УР
_____ / О.А. Морозова /

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студента Беседа Дмитрия Денисовича

Специальность 09.02.07 Информационные системы и программирование

№	Наименование мероприятия	Срок выполнения	Отметка о выполнении
1	Получение задания на ВКР	01.03.2023	
2	Подбор и проведение анализа источников специальной литературы по теме работы	15.03.2023	
3	Подбор литературы и материалов о деятельности организации (предприятия)	22.03.2023	
4	Выполнение исследования по теме ВКР	29.03.2023	
5	Литературное изложение разделов. Работа над введением: актуальность, цель, задачи и пр.	05.04.2023	
6	Первый просмотр руководителем ВКР	08.04.2023	
7	Работа над теоретической главой	13.04.2023	
8	Второй просмотр руководителем ВКР	20.04.2023	
9	Работа над аналитической главой	27.04.2023	
10	Работа над практической главой	29.04.2023	
11	Описание практической значимости работы. Предложения по внедрению мероприятий	04.05.2023	
12	Третий просмотр руководителем ВКР	06.05.2023	
13	Форматирование работы в соответствии с требованиями нормоконтроля	11.05.2023	
14	Формулировка выводов. Заключение. Оформление списка литературы	13.05.2023	
15	Форматирование работы в соответствии с требованиями нормоконтроля	16.05.2023	
16	Четвертый просмотр руководителем ВКР	20.05.2023	
17	Техническое оформление работы	27.05.2023	
18	Представление работы с отзывом и рецензией	01.06.2023	
19	Подготовка защитного слова, оформление раздаточного материала для комиссии, презентации ВКР	10.06.2023	

Руководитель _____ / Ю.В. Зиненко / «1» марта 2023 г.
(подпись)

Студент _____ / Д.Д. Беседа / «1» марта 2023 г.
(подпись)

Министерство общего и профессионального образования
Ростовской области
государственное бюджетное профессиональное образовательное учреждение
Ростовской области
«Волгодонский техникум информационных технологий, бизнеса и дизайна
имени В.В. Самарского»

ОТЗЫВ
НА ДИПЛОМНЫЙ ПРОЕКТ

Студента *Беседа Дмитрия Денисовича*

Тема дипломного проекта *Разработка программного модуля просмотр оценок для МБОУ «Гимназия «Шанс»*

1. Актуальность темы. *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.*

2. Оценка содержания дипломного проекта. *Vel orci porta non pulvinar neque laoreet suspendisse interdum. Sagittis orci a scelerisque purus semper eget duis at tellus. Sit amet cursus sit amet.*

3. Качество теоретического и расчетного обоснования принятых в дипломном проекте решений (положительные стороны работы, замечания и недостатки). *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Semper viverra nam libero justo laoreet sit. Senectus et netus et malesuada fames ac turpis egestas sed.*

4. Практическая значимость дипломного проекта. *Vel orci porta non pulvinar neque laoreet suspendisse interdum.*

5. Рекомендации по внедрению результатов дипломного проекта. *Vel orci porta non pulvinar neque laoreet suspendisse interdum.*

6. Заключение. *Данный дипломный проект заслуживает оценки «Отлично»*

Руководитель _____ / Ю.В. Зиненко /
(подпись)

С отзывом ознакомлен _____ / Д.Д. Беседа /
(подпись)

«10» июня 2023 года

Министерство общего и профессионального образования
Ростовской области
государственное бюджетное профессиональное образовательное учреждение
Ростовской области
«Волгодонский техникум информационных технологий, бизнеса и дизайна
имени В.В. Самарского»

РЕЦЕНЗИЯ
НА ДИПЛОМНЫЙ ПРОЕКТ

Студента *Беседа Дмитрия Денисовича*

Тема дипломного проекта *Разработка программного модуля просмотр оценок для МБОУ «Гимназия «Шанс»*

1. Актуальность, новизна. *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.*

2. Оценка качества выполнения каждой главы дипломного проекта. *Vel orci porta non pulvinar neque laoreet suspendisse interdum. Sagittis orci a scelerisque purus semper eget duis at tellus. Sit amet cursus sit amet.*

3. Отличительные особенности дипломного проекта. *Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Semper viverra nam libero justo laoreet sit. Senectus et netus et malesuada fames ac turpis egestas sed.*

4. Недостатки дипломного проекта. *Vel orci porta non pulvinar neque laoreet suspendisse interdum.*

5. Практического задания дипломного проекта и рекомендации по ее внедрению. *Vel orci porta non pulvinar neque laoreet suspendisse interdum.*

6. Рекомендуемая оценка. *Данный дипломный проект заслуживает оценки «Отлично»*

Рецензент _____ / Ю.В. Зиненко /
(подпись)

С рецензией ознакомлен _____ / Д.Д. Беседа /
(подпись)

«10» июня 2023 года

СОДЕРЖАНИЕ

Введение.....7
1 Технико-экономическая характеристика объекта.....8
2 Сбор, анализ и формирование требований к программному продукту.....10
2.1 Цели и назначение автоматизированный системы.....11
2.2 Входная и выходная информация программного продукта.....12
2.3 Требования к автоматизированной системе.....13
3 Проектирование программного продукта.....15
3.1 Потоки данных в информационной системе.....16
3.2 Сценарии использования программного продукта.....17
3.3 Архитектура программного модуля.....19
3.4 Проектирование реляционной базы данных.....20
4 Разработка программного модуля.....26
4.1 Инструментальные средства разработки.....28
4.2 Описание алгоритма программы.....33
4.3 Пользовательский интерфейса.....35
4.4 Инструкция по эксплуатации.....37
5 Экономическая часть.....41
6 Охрана труда и техники безопасности.....46
Заключение.....48
Список литературы.....49
Приложение А.....52

Введение

Модуль "Просмотр оценок" - это программный компонент, который позволяет пользователям просматривать свои оценки в системе образования. Он может быть использован как студентами, так и преподавателями для управления процессом обучения.

Приложение может содержать информацию о текущих оценках, результатах тестов и других важных данных.

Разработка требует тщательного планирования и анализа требований пользователей. Важно учитывать все необходимые функции, удобство использования и безопасность данных. Кроме того, необходимо обеспечить совместимость модуля с другими системами образования, чтобы обеспечить его эффективное использование.

Основная цель разработки заключается в упрощении процесса получения информации о своих оценках и результатах обучения для студентов и преподавателей. Оно должно быть удобным в использовании, иметь интуитивно понятный интерфейс и обеспечивать быстрый доступ к необходимой информации.

Также важно учитывать удобство использования приложения и его доступность для всех пользователей. Приложение должно быть интуитивно понятным и простым в использовании, чтобы студенты и преподаватели могли быстро получить необходимую информацию.

1 Технико-экономическая характеристика объекта

Технико-экономическая характеристика предприятия представляет собой универсальную методику анализа, которая позволяет оценить общее положение дел в рамках хозяйствующего субъекта, определить наличие проблемных зон и степень их воздействия, грамотно расставить приоритеты и определить оптимальную траекторию развития.

Особенность технико-экономического анализа состоит в том, что он учитывает не только финансовые моменты и экономическую эффективность, выгоду, но и задействованную в основной деятельности материально-техническую основу, эффективность ее применения и пр. Благодаря расчету и анализу определенных показателей студент или руководитель организации получает информацию относительно ключевых аспектов деятельности исследуемого объекта:

– Какие изменения произошли в деятельности предприятия и в чем они отмечены, характер изменений, укладываются ли они в рамки изначально намеченной траектории и пр.;

– Насколько эффективно используется материально-техническая база предприятия, какая часть приносит больше выгоды, необходимо ли обновление оснащения и оборудование, покрывают ли получаемые доходы обслуживание текущего комплекса и производственного цикла и пр.;

Окупается ли затрачиваемая материально-техническая база при текущих условиях реализации, какие направления деятельности следует оптимизировать или вовсе ликвидировать, возможно ли развитие нового направления и пр [3].

Полное наименование Муниципальное бюджетное образовательное учреждение "Гимназия "Шанс" г. Волгодонска, сокращённое наименование - МБОУ "Гимназия «Шанс» г. Волгодонска.

Здание МБОУ "Гимназия «Шанс» г.Волгодонска построено в 1983 г., по типовому проекту дошкольного образовательного учреждения.

Фактический и юридический адрес: 347383, пр. Мира, 29, г.Волгодонск, Ростовская область, Российская Федерация. Контакты: e-mail: shans-26vdonsk@mail.ru, телефон/факс: 8(8639)234392.

Основной вид деятельности – общеобразовательная (лицензия Министерства общего и профессионального образования Ростовской области 61 № 000120 от 21.12.2010г., действует бессрочно).

МБОУ "Гимназия «Шанс» г.Волгодонска является областной инновационной площадкой по внедрению модели преемственности нравственно-патриотического воспитания учащихся и воспитанников в условиях введения ФГТ и ФГОС в соответствии с приказом минобразования РО «О присвоении статуса областной инновационной площадки и областной пилотной площадки образовательным учреждениям и о прекращении действия статуса областной инновационной площадки» от 03.07.2013г.

2 Сбор, анализ и формирование требований к программному продукту

Анализ требований – это важнейшая часть процесса разработки программного обеспечения (ПО), которая задаёт вектор всей остальной работе. Она включает в себя следующие подпроцессы:

– сбор требований к разрабатываемому ПО;

– систематизацию выявленных требований;

– поиск взаимосвязей и их рационализация;

– документирование полученной информации.

Во время сбора требований необходимо учитывать вероятность возникновения противоречий между тремя заинтересованными сторонами: заказчик, разработчик и конечный пользователь. Успех всего проекта напрямую зависит от качества сбора и анализа требований. Требования к разрабатываемому ПО должны отвечать следующим критериям:

– документируемость (в письменном и/или электронном виде);

– выполнимость;

– тестируемость;

– достаточный (для текущего проекта) уровень детализации.

Анализ требований – это достаточно длительный и трудоёмкий процесс, который состоит из восьми этапов:

– сбор информации;

– общение с заказчиком и конечным пользователем;

– изучение предметной области;

– анализ собранной информации:

– систематизация информации;

– поиск противоречий и неточностей;

– в случае наличия данных проблем, их решение;

– поиск взаимосвязей;

– документирование информации (описание, сценарии, спецификация процессов и т.д.).

Основная задача анализа требований – это получение списка не дублируемых требований к разрабатываемому ПО [1]. Корректная группировка требований позволит определить минимальное и достаточное количество необходимых функций, которые в свою очередь смогут удовлетворить максимально большое количество целей. Таким образом, грамотный подход к этой задаче поможет обозначить чёткие рамки проекта, что сэкономит бюджет и облегчит процесс разработки ПО.

На основе анализа требований можно определить следующие спецификации для программного модуля электронного расписания:

– язык программирования – C#;

– база данных – SQL Server;

– интерфейс пользователя – веб–приложение;

– функциональность модуля должна быть доступна через интернет;

– доступ к системе должен быть защищен паролем и логином;

программный модуль должен поддерживать многопользовательскую работу с возможностью управления правами доступа.

2.1 Цели и назначение автоматизированный системы

Разрабатываемая информационная система должна автоматизировать процессы просмотра оценок учащихся для «Гимназия «Шанс».

Применение программного модуля в деятельности «Гимназия «Шанс» упростит работу с просмотром оценок учащихся.

Разработка приложения просмотра оценок для учащихся для «Гимназия «Шанс» позволит:

– повысить эффективность работы образовательного учреждения;

– повысить эффективность работы с оценками.

Разрабатываемый модуль должен выполнять следующие функции:

– формирование базы данных;

– просмотр оценок;

– сортировка по предметам;

– фильтрация по классам;

– не требовать особых навыков для работы;

– быть понятен для неопытного пользователя;

– иметь интуитивно понятный интерфейс.

К удобствам использования модуля относятся следующие возможности:

– уменьшение временных и трудовых затрат на выполнение функциональных обязанностей, снижение эмоциональной нагрузки;

– возможность автоматической публикации вакансии.

2.2 Входная и выходная информация программного продукта

Входные и выходные данные — это информация, которая передается между различными компонентами программного продукта. Входными данными являются данные, которые поступают в систему извне, а выходные данные - информация, которую система генерирует или выдает наружу. Входные данные могут быть различными, например, пользовательские запросы, данные из баз данных, файлы, графические изображения и т.д. Выходные данные также могут быть различными в зависимости от типа программного продукта, но обычно они включают ответы на запросы пользователя, отчеты, графики, таблицы и т.д.

Для эффективного управления входными и выходными данными необходимо понимать, как они взаимодействуют друг с другом и какие требования предъявляются к ним. Это может включать в себя оптимизацию процесса обработки данных, обеспечение безопасности передачи данных и контроль за их использованием.

Входная информация:

– название класса;

– название предмета.

Выходная информация:

– список оценок учащихся.

2.3 Требования к автоматизированной системе

Требования к автоматизированной системе разработки программного модуля для просмотра оценок могут включать следующие пункты:

- Модуль должен позволять пользователям просматривать оценки по предметам и ученикам.
- Модуль должен предоставлять возможность фильтрации оценок по различным параметрам.
- Система должна обеспечивать безопасность данных пользователей, включая защиту от несанкционированного доступа и изменения.
- Необходимо использовать шифрование данных при передаче между клиентом и сервером.
- Должна быть реализована система аутентификации и авторизации пользователей.
- Система должна быть быстрой и эффективной при обработке запросов пользователей.
- Следует оптимизировать запросы к базе данных для уменьшения времени выполнения операций.
- Рекомендуется использовать кэширование для ускорения доступа к данным.
- Интерфейс должен быть интуитивно понятным и удобным для пользователей.
- Пользователи должны иметь возможность быстро находить нужную информацию.

- Пользовательский интерфейс должен быть адаптивным и поддерживать различные устройства.

- Система должна легко масштабироваться для поддержки растущего количества пользователей и данных.

3 Проектирование программного продукта

Сложная система обычно может быть разделена на более простые части – модули. Модульность является важным качеством инженерных процессов и продуктов. Большинство промышленных процессов являются модульными и составлены из комплексов работ, которые комбинируются простыми способами (последовательными или перекрывающимися) для достижения требуемого результата. Главное преимущество модульности состоит, по сути, в том, что она позволяет применять принцип разделения задач на двух этапах:

– при работе с элементами каждого модуля отдельно (игнорируя элементы других модулей);

– при работе с общими характеристиками групп модулей и отношениями между ними с целью объединить, их в конкретный, более крупный и сложный компонент.

Программные модули решают относительно небольшие функциональные задачи, и каждый реализуется 10-100 операторами языка программирования. Каждый модуль может использовать на входе около десятка типов переменных. В случае если для решения небольшой функциональной задачи требуется более 100 операторов, то обычно целесообразно проводить декомпозицию задачи на несколько более простых модулей.

Проектирование программных модулей включает в себя разработку локальных функций и подробных описаний алгоритмов обработки данных; межмодульных интерфейсов; внутренних структур данных; структурных схем передач управления; средств управления в исключительных ситуациях. С их помощью определяются функции: порядок следования отдельных шагов обработки, ситуации и типы данных, вызывающие изменения процесса обработки, а также повторно используемые функции программы. Программные модули для их многократного использования должны базироваться на унифицированных правилах структурного построения, оформления спецификаций требований и описаний текстов программ и комментариев. Вместе

с тем, целесообразно для каждого проекта директивно ограничивать размеры модулей по числу строк текста с учетом языка программирования, к примеру, 30-ю или 50-ю операторами.

3.1 Потоки данных в информационной системе

Диаграмма потоков данных – методология графического структурного анализа, описывающая внешние по отношению к системе, источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

Диаграмма потоков данных (data flow diagram, DFD) – один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML. Несмотря на имеющее место в современных условиях смещение акцентов от структурного к объектно-ориентированному подходу к анализу и проектированию систем, «старинные» структурные нотации по-прежнему широко и эффективно используются как в бизнес-анализе, так и в анализе информационных систем.

Для описания диаграмм DFD используются две нотации – Йордана (Yourdon) и Гейна-Сарсон (Gane-Sarson), отличающиеся синтаксисом.

Информационная система принимает извне потоки данных. Для обозначения элементов среды функционирования системы используется понятие внешней сущности. Внутри системы существуют процессы преобразования информации, порождающие новые потоки данных. Потоки данных могут поступать на вход к другим процессам, помещаться (и извлекаться) в накопители данных, передаваться к внешним сущностям [17].

Для понимания работы приложения просмотра оценок составлена диаграмма потоков данных, представленная на рисунке 3.1.

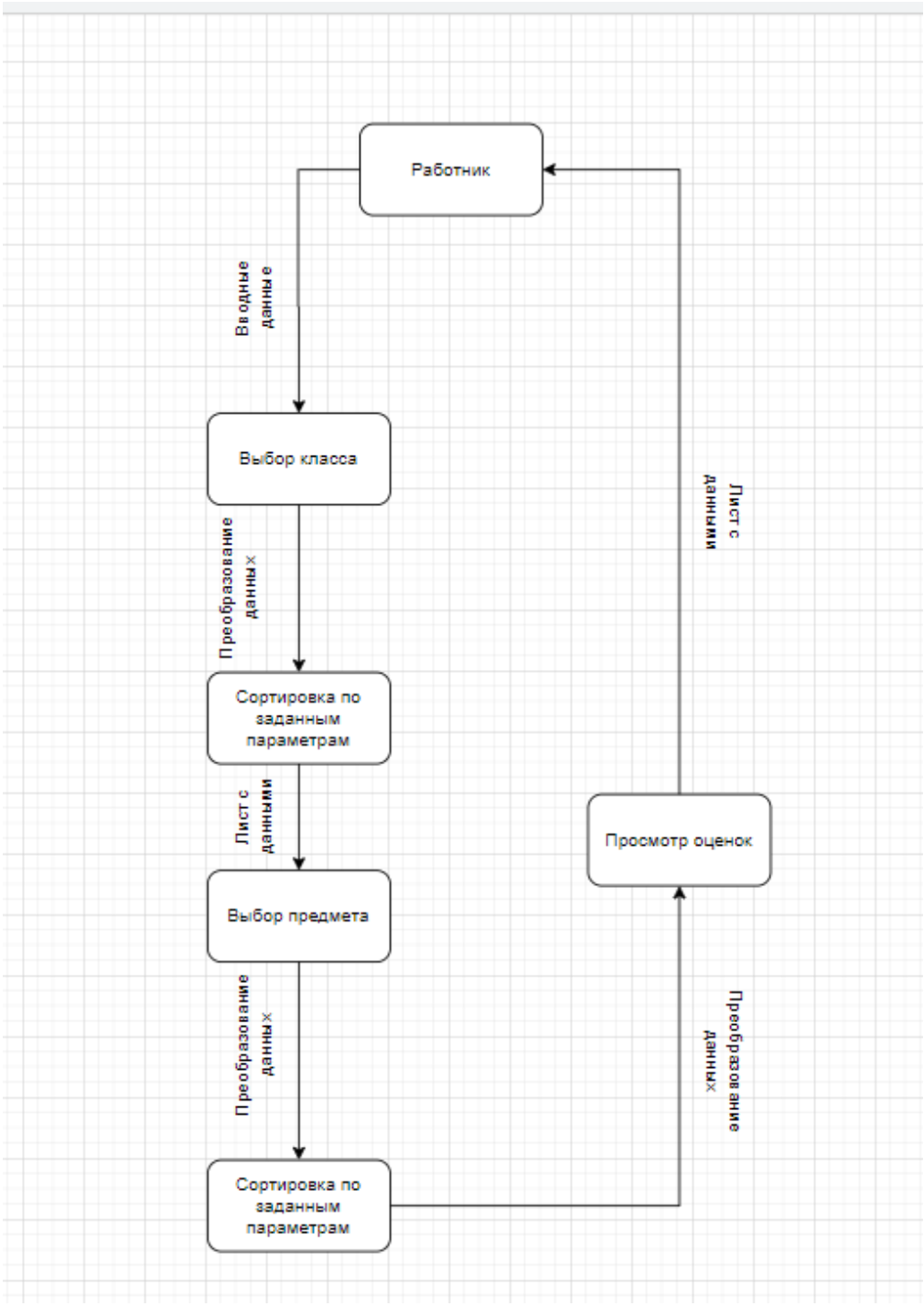

Рисунок 3.1 – Диаграмма потоков данных программного модуля просмотр оценок

3.2 Сценарии использования программного продукта

Диаграмма прецедентов или диаграмма вариантов использования – диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне [18].

Проектирование – один из важных шагов при разработке программы, который очень часто игнорируется начинающими разработчиками. Обычно они пытаются удержать всё в голове или, в лучшем случае, записать некоторые важные сведения на листе бумаги.

Обычно при проектировании разработчики изображают систему графически, поскольку человеку легко разобраться в таком представлении. Именно поэтому вместо написания громоздких текстов про каждую возможность будущей программы разработчики строят различные диаграммы для описания своих систем. Это помогает им не забывать, что нужно реализовать в программе, и быстро вводить в курс дела своих коллег [19].

Разрабатываемый в ходе дипломного проекта программный модуль предназначен для работников учебного заведения «Гимназия «Шанс» , поэтому в составленной диаграмме, представленной на рисунке 3.2, в качестве актёра присутствует только клиент.

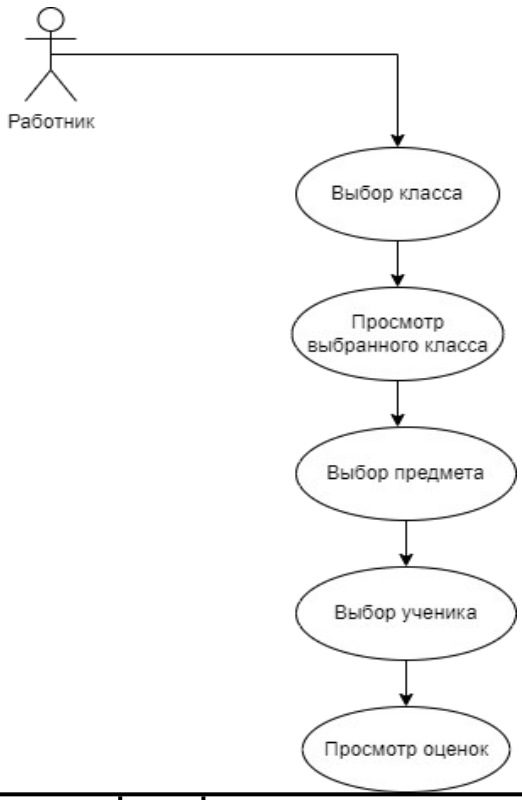

Рисунок 3.2 – Диаграмма прецедентов программного модуля просмотра оценок

3.3 Архитектура программного модуля

Архитектура программного продукта (ПП) представляет собой совокупность и взаимосвязь программных модулей. Модуль – это самостоятельная часть программы, имеющая определенное назначение и обеспечивающая заданные функции обработки автономно от других программных модулей. ПП обладает внутренней структурой, что обеспечивает удобство разработки, программирование, отладку и внесение изменений в ПП. Программные комплексы большой алгоритмической сложности разрабатываются коллективом разработчиков от 2 до 15 человек [21].

Для разрабатываемого программного продукта была выбрана многоуровневая архитектура в качестве наиболее подходящей архитектуры для решаемых задач.

Многоуровневая архитектура – это подход к проектированию программного продукта, который предполагает разделение приложения на несколько слоев, каждый из которых выполняет свою функцию и взаимодействует с другими слоями через определенные интерфейсы. Каждый слой может быть реализован независимо от других, что позволяет упростить разработку, тестирование и сопровождение приложения. Также многоуровневая архитектура обеспечивает гибкость и масштабируемость приложения, что позволяет быстро адаптироваться к изменяющимся требованиям бизнеса. Многоуровневая архитектура данного программного продукта представлена на рисунке 3.3.

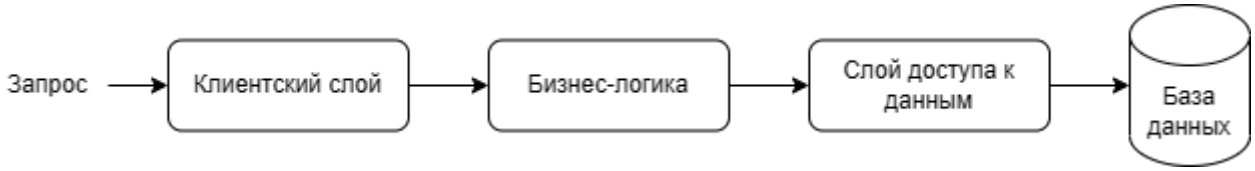

Рисунок 3.3 – Многоуровневая архитектура программного продукта.

Многоуровневая архитектура программного продукта электронного расписания занятий состоит из следующих слоев:

1) Клиентский слой – это пользовательский интерфейс, через который пользователи могут взаимодействовать с приложением. Клиентский слой может включать в себя веб-интерфейс, мобильное приложение или настольное приложение.

2) Бизнес-логика – это слой, который обрабатывает все бизнес-процессы, связанные с составлением расписания. Он может включать в себя модули для управления заказами, инвентаризации, финансовых операций и т.д.

3) Слой доступа к данным – это слой, который обеспечивает доступ к базе данных, где хранятся все данные о продуктах и услугах. Он может включать в себя ORM (Object-Relational Mapping) для упрощения работы с базой данных.

Каждый из этих слоев выполняет определенную функцию и может быть разработан независимо от других слоев. Это позволяет создавать гибкие и масштабируемые приложения, которые могут быть легко адаптированы к изменяющимся потребностям бизнеса.

3.4 Проектирование реляционной базы данных

Инфологическая модель – это потоки информации, сущности и связи данной области. В такой модели указываются связи между сущностями данной предметной области.

Инфологическая модель данных состоит из объектов (entities) и атрибутов (attributes), которые описывают данные, хранимые в системе, и связей (relationships), которые определяют отношения между объектами.

Объект – это сущность или явление, о котором хранятся данные. Например, клиент, продукция, заказ и т.д. Каждый объект имеет свой уникальный идентификатор (primary key), который позволяет однозначно идентифицировать запись в базе данных.

Атрибут – это характеристика объекта, которая описывает ее свойства или параметры. Например, для объекта «пользователь» атрибутами могут быть имя, фамилия, отчество и т.д.

Связь – это отношение между двумя или более объектами. Она определяет, как один объект связан с другим. Например, связь «занятия» может связывать объекты «преподаватели» и «занятия на день».

Инфологическая модель данных используется для проектирования базы данных и определения ее структуры. Она позволяет разработчикам и аналитикам понимать структуру данных и логику их хранения, что упрощает разработку и поддержку программного продукта.

Инфологическая модель базы данных представлена на рисунке 3.4.

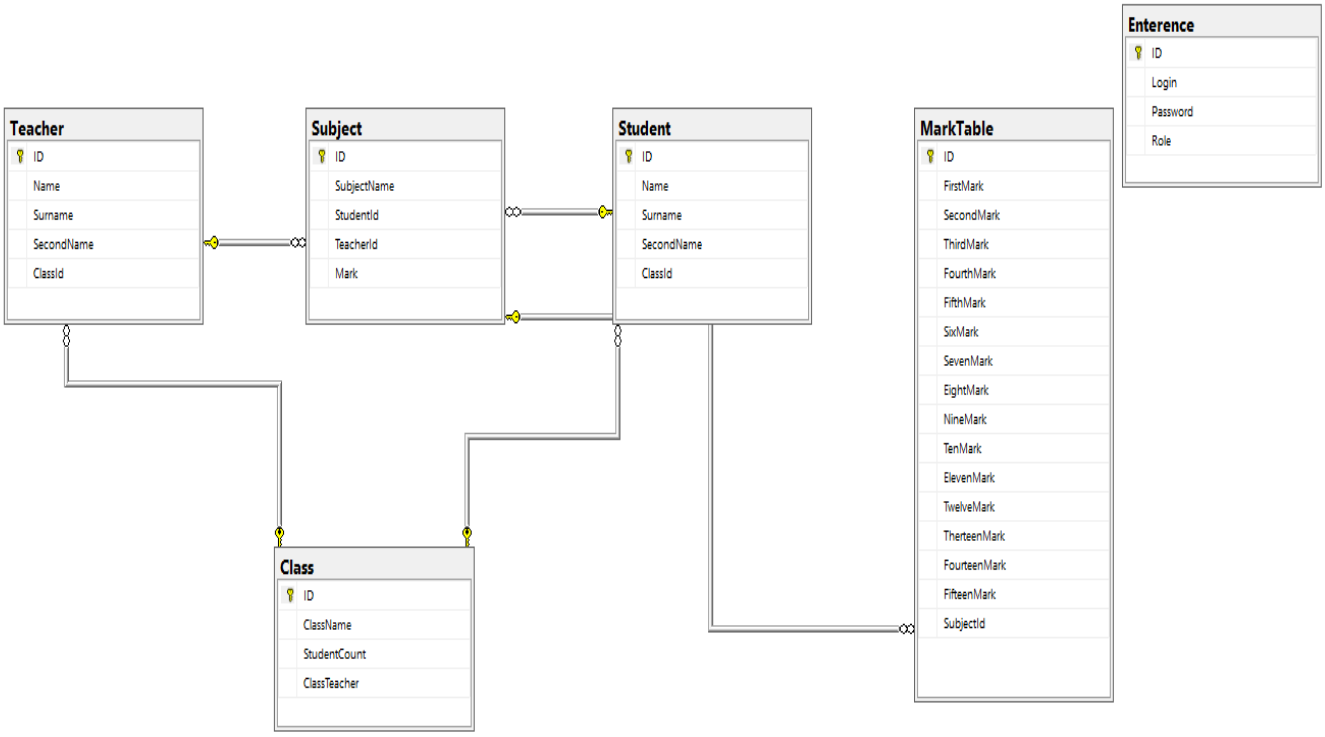

Рисунок 3.4 – Инфологическая модель

Объектами в данной базе данных будут: информация о кабинетах, расписаниях звонков, типах занятий, занятиях на день, группах, занятиях, ролях,

днях расписания, расписаниях, преподавателях и пользователях на таблицах 3.1,3.2,3.3,3.4,3.5,3.6 соответственно.

Таблица 3.1 – Учитель

Объект	Атрибуты
Учитель	Идентификационный код Фамилия Имя Отчество Идентификационный код класса

Таблица 3.2 – Ученик

Объект	Атрибуты
Ученик	Идентификационный код Фамилия Имя Отчество Идентификационный код класса

Таблица 3.3 – Класс

Объект	Атрибуты
Класс	Идентификационный код Название класса Кол-во человек Классный руководитель

Таблица 3.4 – Предмет

Объект	Атрибуты
Предмет	Идентификационный код Название предмета Идентификационный код ученика Идентификационный код учителя Оценка

Таблица 3.5 – Оценки

Объект	Атрибуты
Оценки	Идентификационный код Оценки Идентификационный код предмета

Таблица 3.6 – Авторизация

Объект	Атрибуты
Авторизация	Идентификационный код Логин Пароль Роль

Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая стока таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности [21].

Поэтому для разработки программного электронного расписания занятий была выбрана Microsoft SQL Server. Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка [25].

Схема связей таблиц в базе данных представлена на рисунке 3.7.

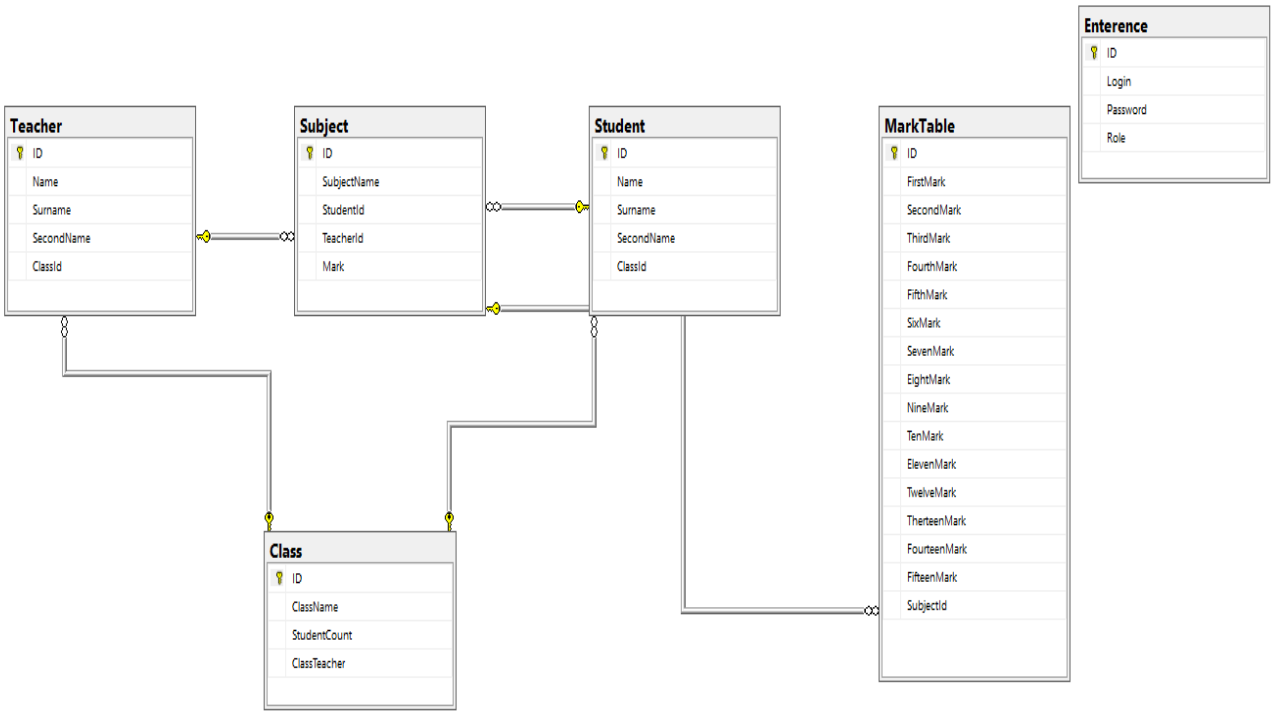

Рисунок 3.7 – Схема связей таблиц в базе данных

Структура таблиц «Учитель», «Ученик», «Предмет», «Оценки», «Класс», «Авторизация» базы данных представлена на рисунках 3.8, 3.9, 3.10,3.11,3.12,3.13 соответственно.

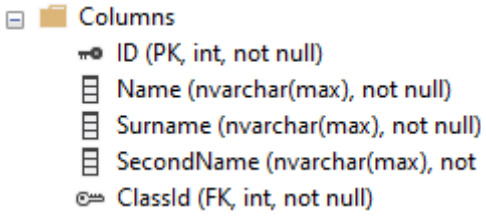

Рисунок 3.8 – Таблица «Учитель»

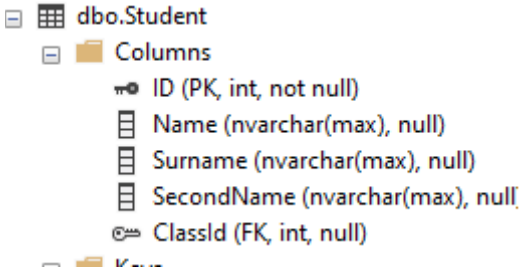

Рисунок 3.9 – Таблица «Ученик»

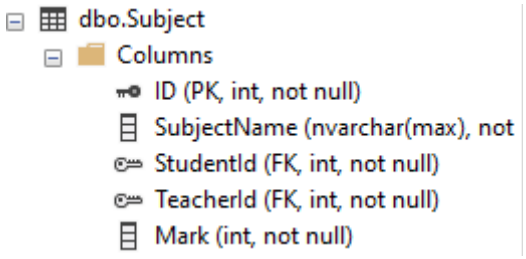

Рисунок 3.10 – Таблица «Предмет»

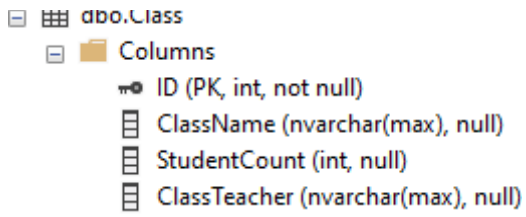

Рисунок 3.11 – Таблица «Класс»

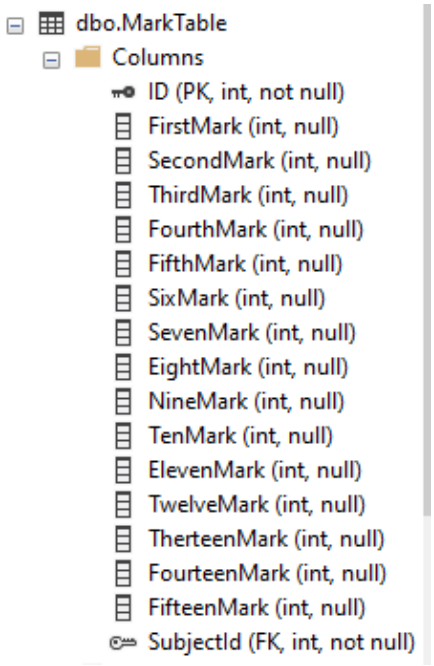

Рисунок 3.12 – Таблица «Оценки»

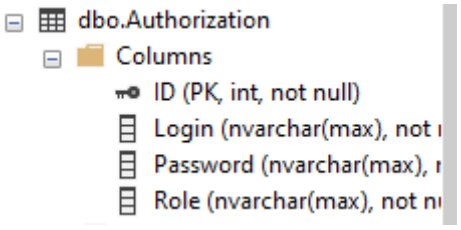

Рисунок 3.13 – Таблица «Авторизация»

4 Разработка программного модуля

Программный модуль – независимая и функционально законченная часть программы, оформленная в виде самостоятельного фрагмента кода, упакованная в отдельный файл или обособленная другим способом. Примером программного модуля может служить DLL-библиотека, которая определенным образом обрабатывает запрос других подсистем и возвращает в ответ полученное значение. Модули формируют структуру программного продукта, позволяют избежать повторяющихся участков кода, делают размер приложения меньше, а его работу – быстрее.

При разработке программного модуля целесообразно придерживаться следующего порядка:

– изучение и проверка спецификации модуля, выбор языка программирования;

– выбор алгоритма и структуры данных;

– программирование (кодирование) модуля;

– шлифовка текста модуля;

– проверка модуля;

– компиляция модуля.

Первый шаг разработки программного модуля в значительной степени представляет собой смежный контроль структуры программы снизу: изучая спецификацию модуля, разработчик должен убедиться, что она ему понятна и достаточна для разработки этого модуля. В завершении этого шага выбирается язык программирования: хотя язык программирования может быть уже предопределен для всего программного продукта, все же в ряде случаев (если система программирования это допускает) может быть выбран другой язык, более подходящий для реализации данного модуля программного продукта (например, язык ассемблера).

На втором шаге разработки программного модуля необходимо выяснить, не известны ли уже какие-либо алгоритмы для решения поставленной или

близкой к ней задачи. И если найдется подходящий алгоритм, то целесообразно им воспользоваться. Выбор подходящих структур данных, которые будут использоваться при выполнении модулем своих функций, в значительной степени предопределяет логику и качественные показатели разрабатываемого модуля, поэтому его следует рассматривать как весьма ответственное решение.

На третьем шаге осуществляется построение текста модуля на выбранном языке программирования. Обилие всевозможных деталей, которые должны быть учтены при реализации функций, указанных в спецификации модуля, легко могут привести к созданию весьма запутанного текста, содержащего массу ошибок и неточностей. Искать ошибки в таком модуле и вносить в него требуемые изменения может оказаться весьма трудоемкой задачей. Поэтому весьма важно для построения текста модуля пользоваться технологически обоснованной и практически проверенной дисциплиной программирования.

Следующий шаг разработки модуля связан с приведением текста модуля к завершенному виду в соответствии со спецификацией качества. При программировании модуля разработчик основное внимание уделяет правильности реализации функций модуля, оставляя недоработанными комментарии и допуская некоторые нарушения требований к стилю программы. При шлифовке текста модуля он должен отредактировать имеющиеся в тексте комментарии и, возможно, включить в него дополнительные комментарии с целью обеспечить требуемые примитивы качества. С этой же целью производится редактирование текста программы для выполнения стилистических требований.

Шаг проверки модуля представляет собой ручную проверку внутренней логики модуля до начала его отладки (использующей выполнение его на компьютере), реализует общий принцип, сформулированный для обсуждаемой технологии программирования, о необходимости контроля принимаемых решений на каждом этапе разработки.

И, наконец, последний шаг разработки модуля означает завершение проверки модуля (с помощью компилятора) и переход к процессу отладки модуля.

4.1 Инструментальные средства разработки

Для разработки данного приложения был выбран язык C# – объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270 [15].

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка [25].

Razor Pages — это предпочтительный способ создания приложений на основе страниц или форм в ASP.NET Core. Из документации: "Razor Pages может сделать программирование сценариев, ориентированных на страницы, проще и эффективнее, чем использование контроллеров и представлений". Если ваше приложение ASP.NET MVC интенсивно использует представления, вы можете рассмотреть возможность перехода с действий и представлений на Razor Pages.

Типичное строго типизированное приложение MVC на основе представления будет использовать контроллер для хранения одного или нескольких действий. Контроллер будет взаимодействовать с доменом или моделью данных и создать экземпляр класса ViewModel. Затем этот класс ViewModel передается в представление, связанное с этим действием. При

использовании этого подхода в сочетании со структурой папок по умолчанию приложений MVC для добавления новой страницы в приложение требуется изменить контроллер в одной папке, представление во вложенной вложенной папке в другой папке и модель представления в еще одной папке.

Razor Pages группировать действие (теперь *обработчик*) и модель представления (называется *PageModel*) в одном классе и связать этот класс с представлением (называется Razor Page). Все Razor Pages переходят в папку *Pages* в корне проекта ASP.NET Core. Razor Pages использует соглашение о маршрутизации на основе их имени и расположения в этой папке. Обработчики ведут себя точно так же, как методы действий, но имеют http-команду, которые они обрабатывают в своем имени (например, `OnGet`). Они также не обязательно должны возвращать, так как по умолчанию предполагается, что они возвращают страницу, с которыми они связаны. Это, как правило, делает Razor Pages и их обработчики более мелкими и более сфокусированными, в то же время упрощая поиск и работу со всеми файлами, необходимыми для добавления или изменения определенной части приложения.

В рамках перехода от ASP.NET MVC к ASP.NET Core командам следует подумать о том, нужно ли переносить контроллеры и представления в ASP.NET Core контроллеры и представления или в Razor Pages. Первый, скорее всего, потребует немного меньше общих усилий, но не позволит команде воспользоваться преимуществами Razor Pages по сравнению с традиционной организацией файлов на основе представлений.

Microsoft Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного

обеспечения (например, клиент Team Explorer для работы с Team Foundation Server) [26].

Главные возможности среды разработки Microsoft Visual Studio:

– включены все «интеллектуальные» возможности по редактированию кода;

– есть возможность визуального просмотра будущего приложения;

– сборка проекта работает быстро;

– удобный конструктор интерфейсов;

– удобное и интуитивно понятное логирование проекта;

– указанные цвета и рисунки, использованные в layout'е отображаются на границе в виде небольших превью, которые легко помогают понять какой конкретно ресурс используется;

– среда разработки является технологиями компании Microsoft;

– при выборе ресурса, его содержимое отображается во всплывающих окнах;

– возможность создания одним кликом новых окон и страниц;

– мониторинг используемой памяти;

– возможность подключать сторонние дополнения(плагины);

– возможность добавления новых инструментов.

Рекомендуемые системные требования к компьютеру на которой будет установлена Visual Studio следующие:

– операционная система – Microsoft® Windows® 7/8/10 (32- или 64-bit);

– 2 ГБ ОЗУ. рекомендуется 8 ГБ ОЗУ (минимум 2,5 ГБ при выполнении на виртуальной машине);

– место на жестком диске: до 210 ГБ (минимум 800 МБ) свободного места в зависимости от установленных компонентов; обычно для установки требуется от 20 до 50 ГБ свободного места;

– видеоадаптер с минимальным разрешением 720p (1280 на 720 пикселей); для оптимальной работы Visual Studio рекомендуется разрешение WXGA (1366 на 768 пикселей) или более высокое.

В шаблоне MVVM есть три основных компонента: модель, представление и модель представления. Каждый из них обслуживает отдельную цель. Кроме понимания обязанностей каждого компонента, важно понимать, как они взаимодействуют друг с другом. На высоком уровне в представлении «известно о модели представления и модели представления» известно о модели, но модель не знает модель представления, а модель представления не знает об этом представлении. Таким образом, модель представления изолирует представление от модели и позволяет модели развиваться независимо от представления.

Ниже приведены преимущества использования шаблона MVVM:

Если реализована существующая реализация модели, которая инкапсулирует существующую бизнес-логику, она может быть сложной или рискованной для ее изменения. В этом сценарии модель представления выступает в качестве адаптера для классов модели и позволяет избежать внесения значительных изменений в код модели;

разработчики могут создавать модульные тесты для модели представления и модели без использования представления. Модульные тесты для модели представления могут выполнять точно те же функциональные возможности, которые используются в представлении;

пользовательский интерфейс приложения можно переконструировать, не затрагивая код, при условии, что представление полностью реализовано в XAML. Поэтому Новая версия представления должна работать с существующей моделью представления;

разработчики и разработчики могут одновременно работать с компонентами в процессе разработки. Дизайнеры могут сосредоточиться на представлении, тогда как разработчики могут работать над моделью представления и компонентами модели [27];

команды являются привязанными объектами, что позволяет разделить логику и пользовательский интерфейс друг от друга.

Если рассматривать команды более подробно, то они представляют из себя следующее:

команды представляют собой объекты, реализующие интерфейс ICommand;

обычно команды связанны с какой-либо функцией;

– элементы пользовательского интерфейса привязываются к командам – кода интерфейс активируется пользователем, то выполняется команда – вызывается соответствующая функция;

– команды знают, включены ли они или нет;

– функции могут отключать команды – автоматическое отключение всех пользовательских элементов, ассоциированных с ней.

На самом деле существует множество различных применений команд. Например, использование команд для создания асинхронных функций, обеспечивающих логику, которая может быть проверена с/без помощи использования пользовательского интерфейса и др.

ASP.NET Core представляет технологию для создания веб-приложений на платформе .NET, развиваемую компанией Microsoft. В качестве языков программирования для разработки приложений на ASP.NET Core используются C# и F#.

История ASP.NET фактически началась с выходом первой версии .NET в начале 2002 года и с тех пор ASP.NET и .NET развивались параллельно: выход новой версии .NET знаменовал выход новой версии ASP.NET, поскольку ASP.NET работает поверх .NET. В то же время изначально ASP.NET была нацелена на работу исключительно в Windows на веб-сервере IIS (хотя благодаря проекту Mono приложения на ASP.NET можно было запускать и на Linux).

Однако 2014 год ознаменовал большие перемены, фактически революцию в развитии платформы: компания Microsoft взяла курс на развитии ASP.NET как кроссплатформенной технологии, которая развивается как opensource-проект.

Данное развитие платформы в дальнейшем получило название ASP.NET Core, собственно, как ее официально именуют Microsoft до сих пор. Первый релиз обновленной платформы увидел свет в июне 2016 года. Теперь она стала работать не только на Windows, но и на MacOS и Linux. Она стала более легковесной, модульной, ее стало проще конфигурировать, в общем, она стала больше отвечать требованиям текущего времени.

Текущая версия ASP.NET Core, которая, собственно, и будет охвачена в текущем руководстве, вышла вместе с релизом .NET 7 в ноябре 2022 года. [24].

4.2 Описание алгоритма программы

Алгоритм для веб приложения публикации вакансий может включать следующие шаги:

– поиск по названию – работник должен ввести название вакансии, которую он будет искать

– фильтрация по городу – программа должна иметь возможность искать вакансии в определённых городах, если город не выбран, то поиск должен осуществляться по Ростовской области

– фильтрация по диапазону времени – программа должна иметь возможность фильтровать вакансии по дате их публикации на сайтах, то есть возможность просматривать вакансии, которые были опубликованы за “последнюю неделю”, ”последний месяц”, ”последние три месяца” и т.п.

приблизительный алгоритм работы программы предоставлен ниже на рисунке 4.1.

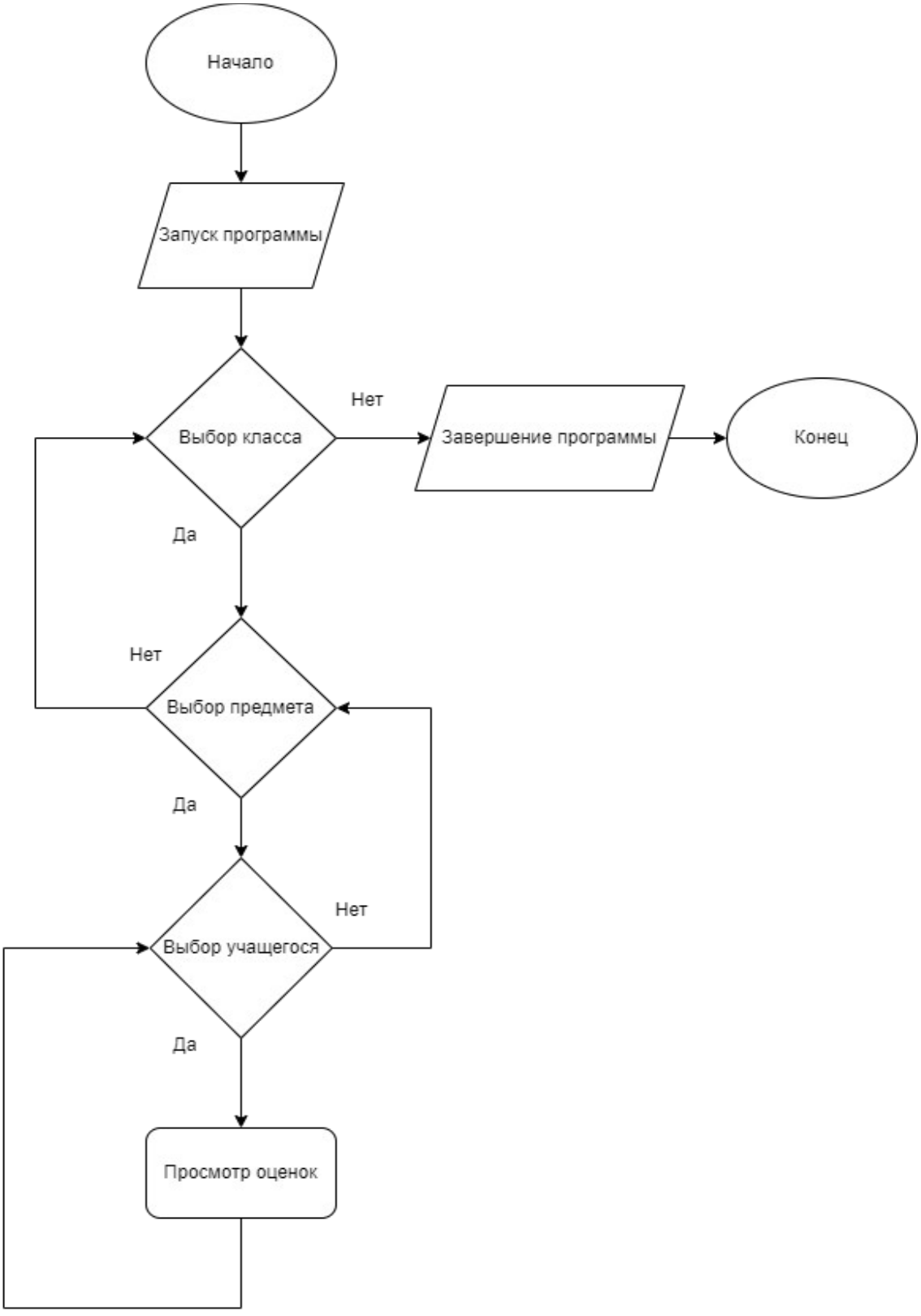

Рисунок 4.1 – Алгоритм работы программы

4.3 Пользовательский интерфейса

Программный модуль просмотра оценок для «Гимназия «Шанс» имеет удобный в использовании интерфейс, реализованный в специально подобранной для предприятия палитре.

В программном модуле представлены все необходимые для функционирования окна.

При открытии программного модуля отображается главное окно, представленная на рисунке 4.2.

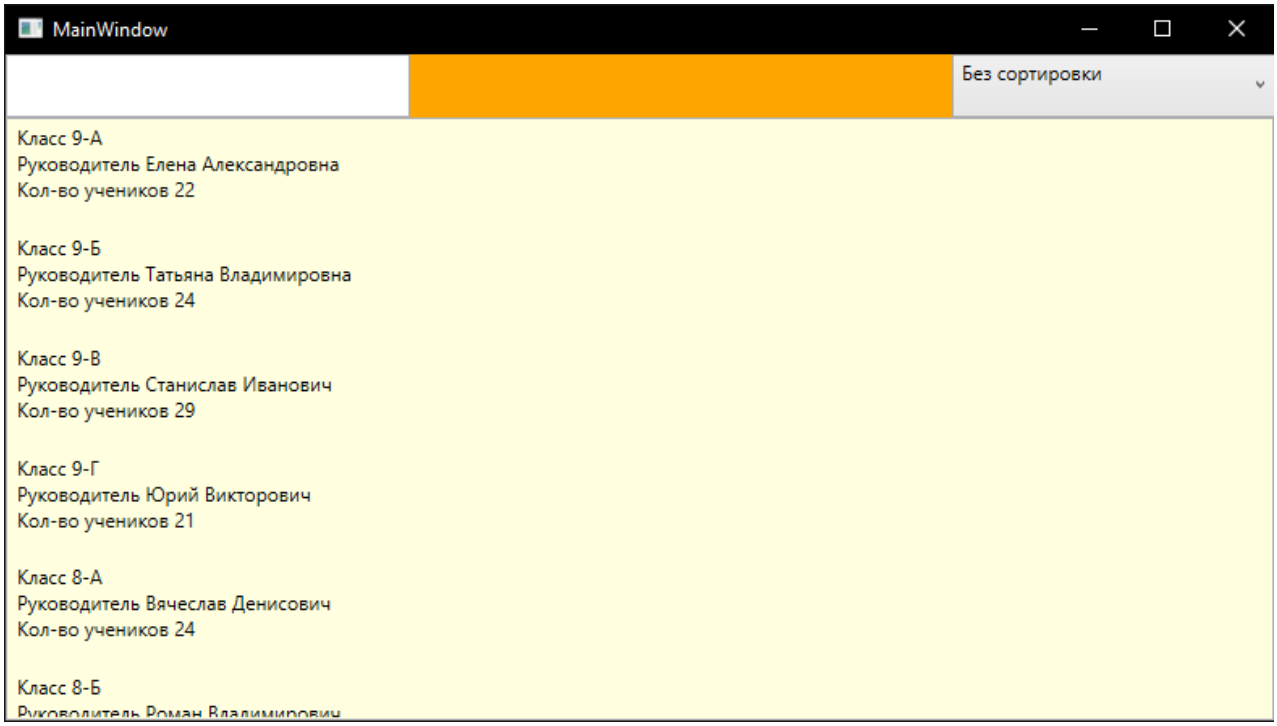

Рисунок 4.2 – Главная страница

По двойному нажатию по листу будет открываться окно выбора предмета, окно предоставлено на рисунке 4.3.

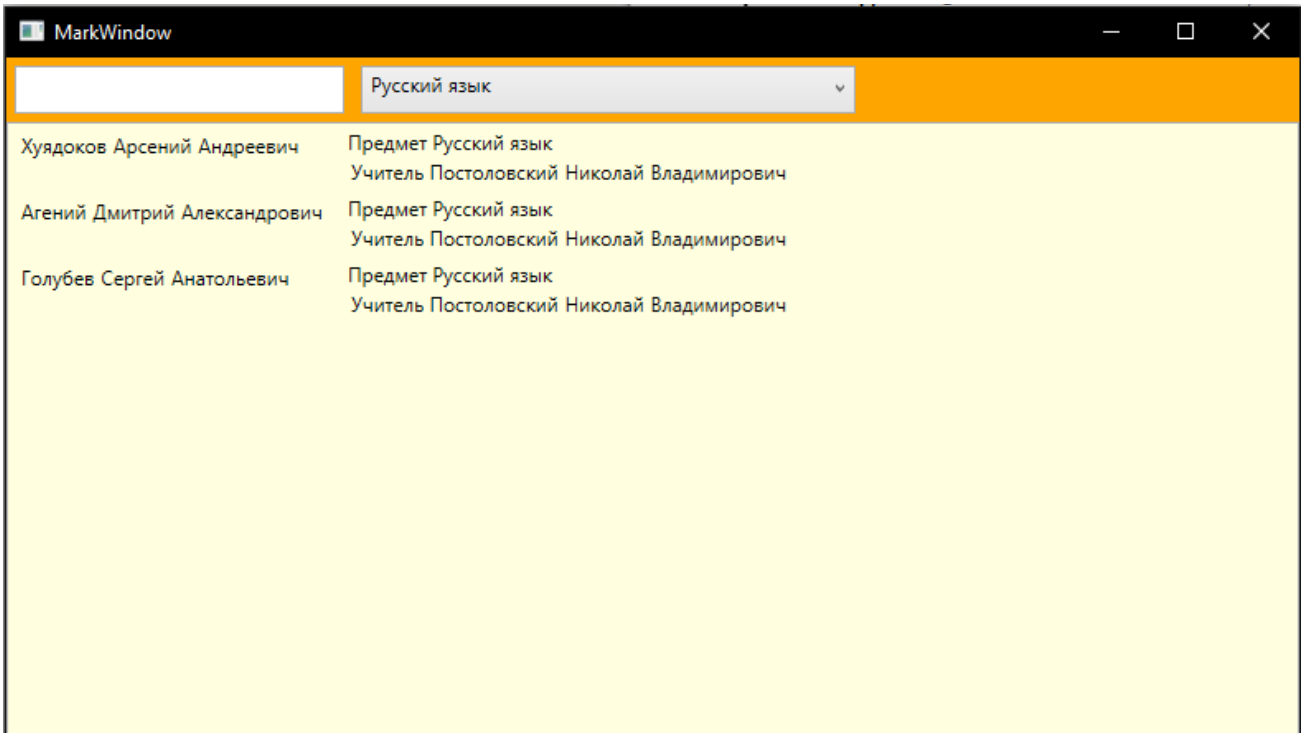

Рисунок 4.3 – Окно выбора предмета

По двойному нажатию по листу будет открываться окно с оценками, окно предоставлено на рисунке 4.4.

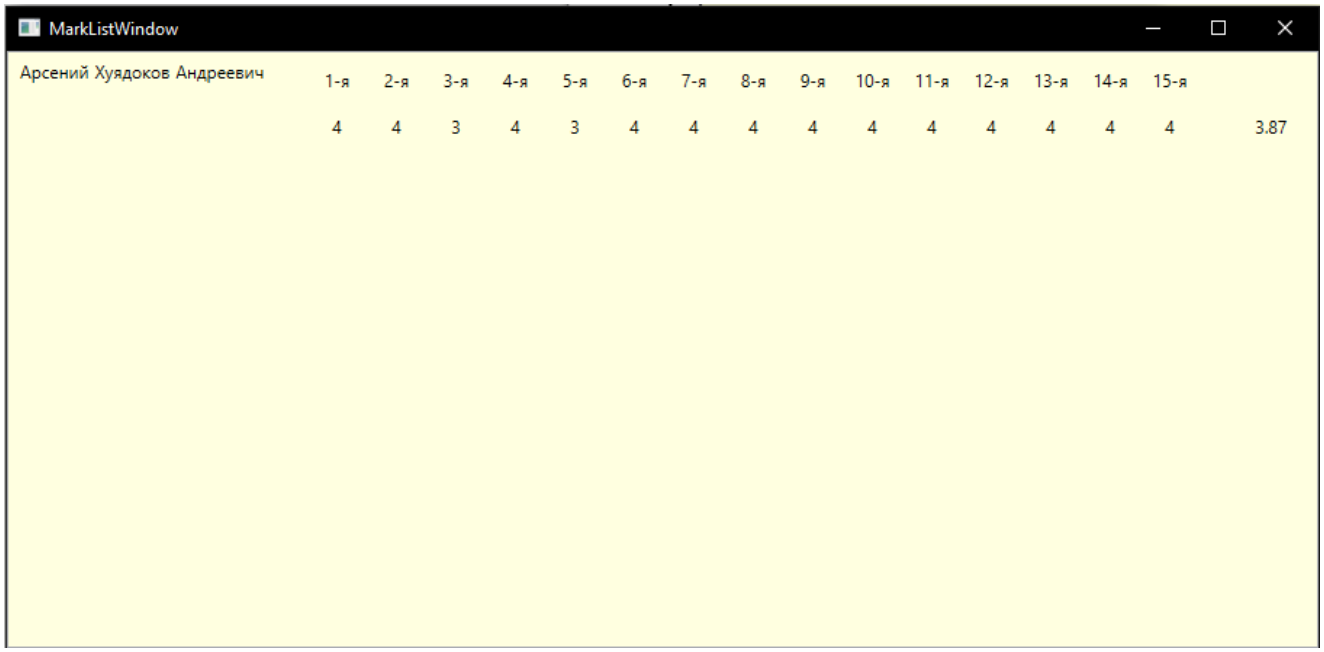

Рисунок 4.4 – Окно Оценок

По двойному нажатию по листу будет открываться окно с редактированием оценок, окно предоставлено на рисунке 4.5.

Рисунок 4.5 – Окно Оценок

4.4 Инструкция по эксплуатации

Для использования программного публикации вакансий не нужно прохождение каких-либо специализированных курсов или обучения.

Функционал программного модуля прост в использовании, все его элементы интуитивно понятны.

Работа с программным модулем начинается с главного окна, в котором пользователь может использовать программу, пользователю нужно выбрать класс для дальнейшего использования приложения на рисунке 4.6.

Рисунок 4.6 – Уведомление об ошибке

Так же при выборе класса пользователь может сортировать классы по номеру и по поиску фамилии классного руководителя 4.7.

Рисунок 4.7 – Сортировка и поиск

В окне выбора предмета пользователь должен выбрать предмет для дальнего пользования 4.8.

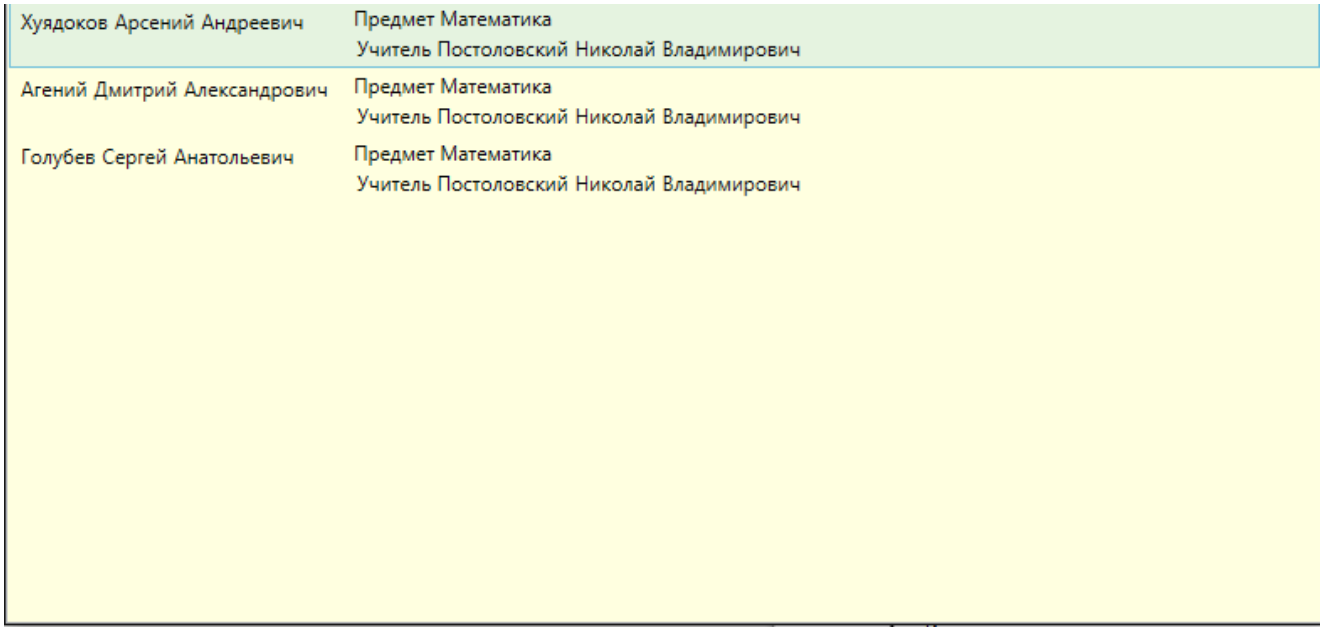

Рисунок 4.8 – Окно выбора предмета

Пользователь так же может сортировать и производить поиск как и в главном окне, представленного на рисунке 4.9.

Рисунок 4.9 – Фильтрация и поиск

В окне просмотра оценок пользователь может смотреть оценки и по двойному нажатию их редактировать. Данный элемент представлен на рисунке 4.10.

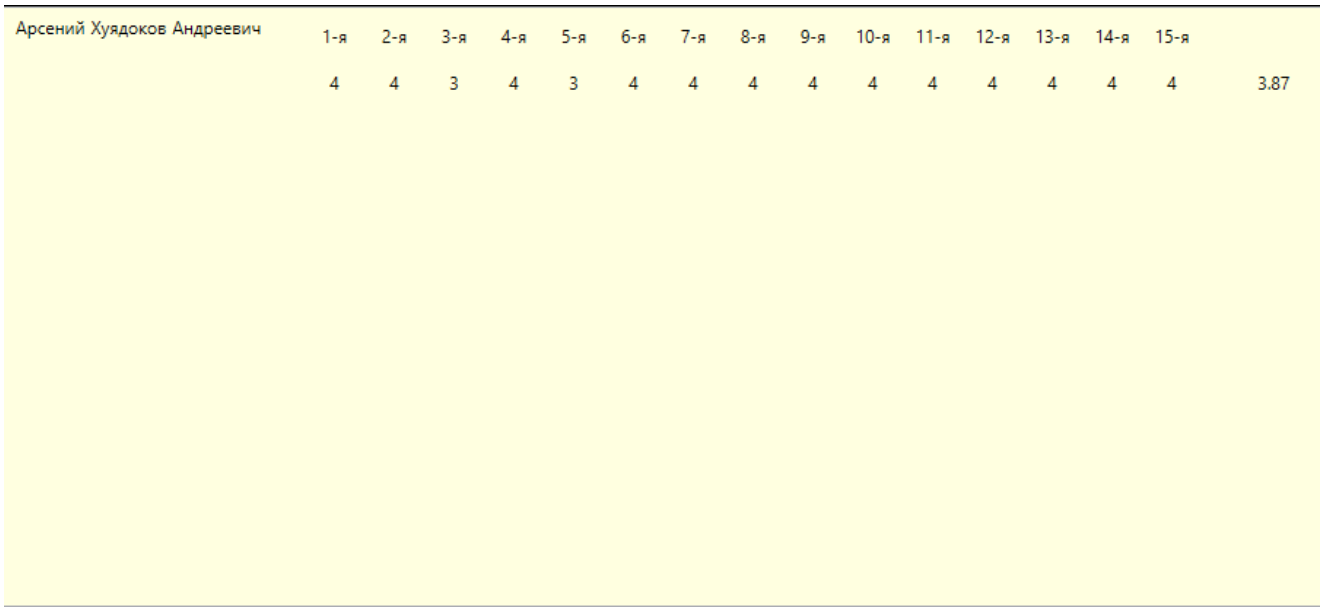

Рисунок 4.10 – Окно просмотра оценок

В окне редактирования оценок пользователь может редактировать любые оценки. Данный функционал предоставлен на рисунке 4.11.

Рисунок 4.11 – Окно редактирования оценок

В окне редактирования оценок пользователь может сохранить измененные оценки нажав на кнопку сохранить. Данный элемент представлен на рисунке 4.12.

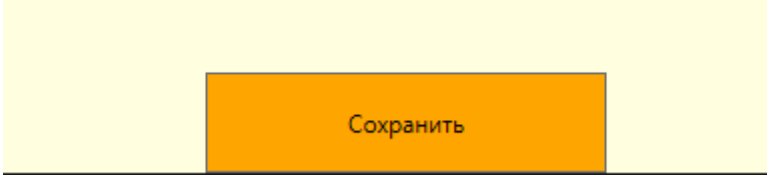

Рисунок 4.12 – Кнопка сохранения

5 Экономическая часть

В расчёте экономической части дипломного проекта используется следующие статьи затрат:

– материальные затраты – материалы, требуемые при выполнении дипломного проекта, включающие свою стоимость в готовую продукцию;

– электроэнергия – затраты, связанные с потреблением электричества оборудованием используемыми при выполнении дипломного проекта;

– заработная плата – плата за фактически отработанное время;

– прочие затраты.

Расчет технологического процесса в написании программного продукта приведен в таблице 5.1.

Таблица 5.1 – Этапы создания программы

Виды работ	Количество времени (ч.)
Полученные задания	2
Подбор материалов	2
Написание программы	92
Отладка программы	24
Тестирование	24
Написание пояснительной записки	16
Итого:	160

Расчет материальных затрат.

В процессе создания программного продукта были осуществлены материальные затраты, список которых приведен в нижеследующей таблице, учитываются как покупные, так и вспомогательные материалы. Расчеты приведены в таблице 5.2.

Таблица 5.2 – Расчеты материальных затрат

Наименование	Единица измерения	Кол-во (шт.)	Цена (руб.)	Сумма (руб.)
Ручка	шт.	2	14	28
Оплата интернета	мес.	1	700	700
Папка	шт.	1	170	170

Бумага	уп.	1	300	300
Итого:				1198

Расчет амортизационных отчислений.

Расчет амортизационных отчислений производятся по годовым нормам амортизации исходя из первоначальной стоимости оборудования по формуле:

A = S × N, где

S – первоначальная стоимость;

N – годовая норма амортизации.

Расчет суммы годовой амортизации выполняется по формуле 1.

A(стол) = 5000 × 10% = 500 руб.;

A(компьютер) = 27294 × 33,3% = 9098 руб.;

A(принтер) = 3399 × 20% = 679,8 руб.;

A(стул) = 700 × 10% = 70 руб.;

Расчет месячной суммы амортизации проводится по формуле:

$$A(\text{мес}) = \frac{A(\text{год})}{12}$$, где

A(мес) – месячная сумма амортизации, руб.;

A(год) – годовая сумма амортизации, руб.;

12 – количество месяцев в году, мес.

Приведем расчет месячной суммы амортизации по формуле 2:

A(мес) = 10347,9 / 12 = 862,31 руб.

Расчет амортизационных отчислений во время фактического создания программного продукта производится по формуле:

A = ((S × N) / Rd) × t, где

S – первоначальная стоимость, руб.;

N – годовая норма амортизации;

Rd – количество рабочего времени, ч.;

t – фактически затраченное время на работу, ч.

Далее, рассчитаем амортизацию на время разработки программного продукта по формуле 3:

А(вр) = 862,31 × 40 / 156 = 221,07 руб.

Расчет амортизации основных фондов показан в таблице 5.3.

Таблица 5.3 – Расчет амортизации основных фондов

Наименование основных фондов	Первоначальная стоимость (руб.)	Норма амортизации	Годовая сумма амортизации (руб.)	Амортизация на выполнение проекта (руб.)
Стол	5000	10%	500	221,07
Компьютер	27294	33,3%	9098	
Принтер	3399	20%	679,8	
Стул	700	10%	70	
Итого:	36393		10347,8	

Расчеты расходов на электроэнергию показаны в таблице 5.4.

Таблица 5.4 – Расходы на электроэнергию

Источники потребления	Потребляемая мощность (кВт/ч)	Время работы (час)	Цена за 1 кВт/ч (руб.)	Сумма (руб.)
Компьютер	0,5	156	4,25	331,5
Эл. лампочка	0,1	40	4,25	17
Принтер	0,6	1	4,25	2,55
Итого:				351,05

Расчет заработной платы производится с учетом фактически использованного времени согласно техническому процессу и средней заработной платы, насчитанной в регионе на момент выполнения дипломного проекта.

Заработная плата за день работы приведена в формуле:

$Z_{Д.} = Z_{МЕС.} / D$,

где $Z_{Д.}$ – зарплата за 1 день,

$Z_{МЕС.}$ – зарплата за 1 месяц,

D – количество рабочих дней в месяце.

Расчет заработной платы приведен в таблице 5.5.

Таблица 5.5 – Расчет заработной платы

Тарифная ставка	Кол-во часов в мес.	Кол-во часов потраченных на написание ПП	Итоговая сумма (руб.)
130	152	160	20800

Расчет коммунальных услуг показан в таблице 5.6.

Сумма затрат на коммунальные услуги рассчитывается за время фактического использования рабочего места в течение написания дипломного проекта и действующих тарифов.

Таблица 5.6 – Расходы на коммунальные услуги

Наименование услуг	Единица измерения	Тариф (руб.)	Количество	Сумма (руб.)
Горячая вода	Метр кубический	68,37	1,5	186,45
Холодная вода	Метр кубический	58,09	3	108,06
Вывоз ТБО	С человека	117,44	1	132,9
Итого:				427,41

Расчет себестоимости (затрат) выполнения дипломного проекта с выполнением указанного задания показан в таблице 5.7.

Таблица 5.7 – Себестоимость по прямым затратам

Наименование статей затрат	Сумма (руб.)
Материальные затраты	1198
Заработная плата	20800
Амортизация основных фондов	221,07
Расходы на электроэнергию	351,05
Прочие затраты	427,41
Итого:	22997,53

Калькуляция рассчитана по прямым затратам.

Цена – это денежное выражение стоимости программного продукта. Для расчета цены принимаем прибыль 30% и рассчитываем по формулам:

$П = (С \times 30)/100$,

$Ц = С+П$, где

П – прибыль;

С – себестоимость;

Ц – цена.

Рассчитаем цену программного продукта по формулам 5 и 6.

$П = (22997,53 \times 15)/100 = 3449,63$ руб.;

$Ц = 22997,53 + 3449,63 = 26447,16$ руб.

Исходя из приведенных расчетов и сведений, что подобная лицензионная программа, написанная на заказ, обходится предприятиям в несколько раз дороже, можно сделать вывод что, созданный продукт экономически выгоднее для предприятия. В этом и состоит эффективность этого программного продукта.

6 Охрана труда и техники безопасности

Работающие с персональным компьютером обязаны:

– знать основные правила и требования безопасности при работе с компьютером;

– соблюдать режим труда и отдыха;

– не допускать к работе лиц, имеющих медицинские ограничения;

– содержать в чистоте рабочее место;

– следить за исправностью оборудования;

– уметь пользоваться первичными средствами пожаротушения;

– немедленно сообщать о неисправности оборудования и других происшествиях.

Требования безопасности перед началом работы:

– подготовить рабочее место. Необходимо убедиться, что на рабочем столе нет лишних предметов, которые могут мешать работе. Также необходимо проверить, что компьютер, монитор и клавиатура находятся в безопасном положении и не имеют повреждений.

– убедиться в достаточности освещенности. Рабочее место должно быть хорошо освещено, чтобы избежать проблем со зрением.

– убедиться в исправности компьютера. Перед началом работы необходимо произвести внешний осмотр компьютера и убедиться, что он не имеет повреждений, а все кабели и провода находятся в исправном состоянии.

– проверить наличие всех необходимых программ и драйверов. Если какие-то программы или драйверы отсутствуют, необходимо их установить.

– убедиться, что все необходимые документы и материалы для работы находятся на своем месте.

– проверить работу вентиляции и кондиционирования воздуха. Если они не работают должным образом, необходимо обратиться к специалистам.

– проверить, что все розетки и выключатели находятся в исправном состоянии и не имеют повреждений.

Требования безопасности в аварийных ситуациях:

– при возникновении аварийной ситуации на рабочем месте, работающий с персональным компьютером обязан работу прекратить, отключить электропитание, сообщить руководителю работ и принять меры к ликвидации создавшейся ситуации.

– в случае возникновения пожара на рабочем месте необходимо немедленно вызвать пожарную службу и принять меры по тушению огня.

– при отравлении газами или другими вредными веществами необходимо немедленно покинуть помещение, открыть окна и двери для проветривания, а также обратиться за медицинской помощью.

– при поражении электрическим током необходимо немедленно отключить электроприборы и оборудование, использовать диэлектрические перчатки и средства защиты, обратиться за помощью к специалистам.

– при возникновении аварийной ситуации необходимо сохранять спокойствие и не паниковать. Действовать по инструкции и рекомендациям руководителя работ.

Требования безопасности по окончанию работы:

– отключить компьютер от электросети, убедившись в том, что все программы были закрыты.

– привести в порядок рабочее место: убрать документы, закрыть все окна и приложения, отключить периферийные устройства.

– проверить отсутствие задымления, огня или других аварийных ситуаций.

– если есть возможность, убрать все потенциально опасные предметы, такие как острые предметы, провода, жидкости и т.д.

– сообщить своему руководителю о завершении работы и готовности к уборке рабочего места.

Заключение

"Просмотр оценок" является актуальной задачей в современном мире. Такое приложение позволяет упростить процесс получения информации об оценках и сократить время на поиск нужных данных. Кроме того, модуль может быть полезен для преподавателей, чтобы они могли легко отслеживать успеваемость своих учеников.

Однако, при разработке программного модуля необходимо учитывать требования пользователей и возможные ошибки, которые могут возникнуть при использовании модуля. Также важно обеспечить безопасность данных, чтобы защитить конфиденциальность пользователей.

В целом, разработка может значительно улучшить качество образования и облегчить жизнь студентов и преподавателей.

Список литературы

1. Mou26 Основные сведения [Электронный ресурс] – Режим доступа: http://mou26shans.ucoz.ru/index/osnovnye_svedenija/0-238
2. Wikipedia свободная энциклопедия [Электронный ресурс]- Режим доступа: https://ru.wikipedia.org/wiki/Заглавная_страница
3. СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работы: Введ. 30-06-03. – Москва: Изд-во стандартов, 2021 - 54 с.
4. СН 512-78 Инструкция по проектированию зданий и помещений для электронно-вычислительных машин: - Введ. 07-01-79. – Москва: Стройиздат, 2019 - 28 с.
5. СП 60.13330.2012 Отопление, вентиляция и кондиционирование воздуха: - Введ. 01-01-2013. - Москва: Минрегион России, 2022 - 75 с.
6. СанПин 2.2.4.548-96 Гигиенические требования к микроклимату производственных помещений: - Введ. 01-10-2019. – Москва: Минздрав России, 2021 - 19 с.
7. ГОСТ Р ЕН 13779-2007 Вентиляция в нежилых зданиях. Технические требования к системам вентиляции и кондиционирования: - Введ. 01-10-2008. – Москва: Стандартинформ, 2018 - 44 с.
8. ГОСТ Р 53320-2009 Светильники. Требования пожарной безопасности. Методы испытаний: - Введ. 18-02-2009. - Москва: Стандартинформ, 2019 - 9 с.
9. ГОСТ Р 54350-2011 Приборы осветительные. Светотехнические требования и методы испытаний: - Введ. 11-07-2011. - Москва: Стандартинформ, 2021 - 37 с.
10. СН 2.2.4/2.1.8.562-96 Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки. Санитарные нормы: - Введ. 31-10-1996. - Москва: Минздрав России, 2020 - 8 с.

11. СН 2.2.4/2.1.8.566-96 Производственная вибрация, вибрация в помещениях жилых и общественных зданий. Санитарные нормы: - Введ. 31-10-1996. - Москва: Минздрав России, 2018.
12. ГОСТ Р 50923-96 Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения: - Введ. 01-07-1997. - Москва: Стандартинформ, 2020 - 9 с.
13. ГОСТ 21889-76 Система «Человек-машина». Кресло человека-оператора. Общие эргономические требования: - Введ. 30-06-1977. - Москва: Издательство стандартов, 2022 - 15 с.
14. ГОСТ 12.1.030-81 Система стандартов безопасности труда. Электробезопасность. Защитное заземление. Зануление: - Введ. 01-07-1982. - Москва: Издательство стандартов, 2021 - 10 с.
15. ПУЭ 7 Правила устройства электроустановок. Издание 7: - Введ. 01-07-2000. - Москва: Издательство НЦ ЭНАС, 2020.
16. ГОСТ 12.1.004-91 Система стандартов безопасности труда. Пожарная безопасность. Общие требования: - Введ. 01-07-1992. - Москва: Стандартинформ, 2019 - 28 с.
17. Васильев А.Н. "Программирование на C# для начинающих. Особенности языка" / А.Н. Васильев. – Москва: Бомбора, 2022.
18. Натан А. "WPF 4. Подробное руководство" / Натан А. – Санкт-Петербург: Символ-Плюс, 2023.

Инетрнет-ресурсы:

19. Диаграммы потоков данных. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/DFD
20. Диаграмма прецедентов. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/use_case

21. Использование диаграммы вариантов использования UML при проектировании программного обеспечения. [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/566218/

22. Прототипирование. [Электронный ресурс] – Режим доступа: https://www.unisender.com/ru/glossary/chto-takoe-prototipirovanie-i-zachem-ono-nuzhno/

23. Реляционные БД. [Электронный ресурс] – Режим доступа: https://aws.amazon.com/ru/relational-database

24. "Трудовой кодекс Российской Федерации" от 30.12.2001 N 197-ФЗ. [Электронный ресурс] – Режим доступа: https://www.consultant.ru/document/cons_doc_LAW_34683/78f36e7afa535cf23e1e865a0f38cd3d230eecf0/

25. Техника безопасности. [Электронный ресурс] – Режим доступа: https://studfile.net/preview/4168836/page:9/

26. aspnet6. [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/aspnet6/1.1.php

27. Microsoft SQL Server. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft_SQL_Server

28. Microsoft Visual Studio. [Электронный ресурс] – Режим доступа: – https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio

29. Model-View-ViewModel. [Электронный ресурс] – Режим доступа: – https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

```
using DimplomBetaTest.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DimplomBetaTest.View.ViewModel
{
    public class MainWindowViewModel : ViewModelBase
    {
        private List<Class> _classes;
        private List<Class> _displayingClasses;
        private Class _selectedClasses;
        private string _searchValue;
        private string _sortValue;


        public Class SelectedClasses
        {
            get => _selectedClasses;
            set => Set(ref _selectedClasses, value, nameof(SelectedClasses));
        }
        public List<Class> DisplayingClasses
        {
            get => _displayingClasses;
                                    set   =>   Set(ref   _displayingClasses,   value,
nameof(DisplayingClasses));
        }

        public string SearchValue
        {
            get => _searchValue;
            set
            {
                Set(ref _searchValue, value, nameof(SearchValue));
                DisplayingClassess();
            }
        }

        public List<string> ValuesToSort => new List<string>
        {
```

```
            "Без сортировки","9-","8-","7-","6-","5-","4-","3-","2-","1-"
        };

        public string SortValue
        {
            get => _sortValue;
            set
            {
                Set(ref _sortValue, value, nameof(SortValue));
                DisplayingClassess();
            }
        }


        public MainWindowViewModel()
        {
            SortValue = ValuesToSort[0];
            using DiplomBetaContext context = new DiplomBetaContext();
            {
                _classes = context.Classes.ToList();
            }
            _displayingClasses = new List<Class>(_classes);
        }

        private List<Class> Search(List<Class> classes)
        {
            if ((SearchValue == null) || (SearchValue == string.Empty))
                return classes;

            var value = SearchValue.ToLower();
                                                    return    classes.Where(p    =>
p.ClassTeacher.ToLower().Contains(value)).ToList();
        }

        private void DisplayingClassess()
        {
            DisplayingClasses = Search(Filter(_classes));
        }

        private List<Class> Filter(List<Class> classes)
        {
            if (SortValue == ValuesToSort[1])
                return classes.Where(p => p.ClassName.StartsWith('9')).ToList();
            else if (SortValue == ValuesToSort[2])
```

```
                    return classes.Where(p => p.ClassName.StartsWith('8')).ToList();
                if (SortValue == ValuesToSort[3])
                    return classes.Where(p => p.ClassName.StartsWith('7')).ToList();
                else if (SortValue == ValuesToSort[4])
                    return classes.Where(p => p.ClassName.StartsWith('6')).ToList();
                if (SortValue == ValuesToSort[5])
                    return classes.Where(p => p.ClassName.StartsWith('5')).ToList();
                else if (SortValue == ValuesToSort[6])
                    return classes.Where(p => p.ClassName.StartsWith('4')).ToList();
                if (SortValue == ValuesToSort[7])
                    return classes.Where(p => p.ClassName.StartsWith('3')).ToList();
                else if (SortValue == ValuesToSort[8])
                    return classes.Where(p => p.ClassName.StartsWith('2')).ToList();
                if (SortValue == ValuesToSort[9])
                    return classes.Where(p => p.ClassName.StartsWith('1')).ToList();
                else return classes;
            }
        }
    }
    using DimplomBetaTest.Models;
    using Microsoft.EntityFrameworkCore;
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;

    namespace DimplomBetaTest.View.ViewModel
    {
        public class MarkWindowListViewModel : ViewModelBase
        {
            private List<MarkTable> _marks;
            private List<MarkTable> _displayingMarks;
            private Subject _Selectedsubject;
            private Student _Selectedstudent;
            private MarkTable _SelectedMarkTable;

            public MarkTable SelectedMarkTable
            {
                get => _SelectedMarkTable;
                set => Set(ref _SelectedMarkTable, value,nameof(SelectedMarkTable));
            }

            public Subject SelectedSubject
```

```
        {
            get => _Selectedsubject;
            set
            {
                Set(ref _Selectedsubject,value,nameof(SelectedSubject));
                DisplayMarks();
            }
        }

        public Student SelectedStudent
        {
            get => _Selectedstudent;
            set
            {
                Set(ref _Selectedstudent, value, nameof(SelectedStudent));
                DisplayMarks();
            }
        }
        public List<MarkTable> DisplayingMarks
        {
            get=> _displayingMarks;
            set=> Set(ref _displayingMarks,value,nameof(DisplayingMarks));
        }

        public MarkWindowListViewModel()
        {
            using DiplomBetaContext context = new DiplomBetaContext();
            {
                 _marks = context.MarkTables.Include(p => p.Subject).Include(p =>
p.Subject.Student).Include(p=>p.Subject.Teacher).Include(p=>p.Subject.Teacher.Class).ToL
ist();
            }
            _displayingMarks = new List<MarkTable>(_marks);
        }

        private List<MarkTable> Filter(List<MarkTable> markTables)
        {
                  return  markTables.Where(markTable  =>  markTable.Subject.Id  ==
SelectedSubject.Id).ToList();
        }

        private void DisplayMarks()
        {
            DisplayingMarks = Filter(_marks);
```

```
        }
    }
}
using DimplomBetaTest.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DimplomBetaTest.View.ViewModel
{
    public class MarkWindowViewModel : ViewModelBase
    {
        private List<Subject> _subjects;
        private List<Subject> _displayingSubjects;
        private Class _selectedCasses;
        private Subject _selectedSubject;
        private Student _selectedStudent;
        private string _searchValue;
        private string _sortValue;

        public Subject SelectedSubject
        {
            get=> _selectedSubject;
            set => Set(ref _selectedSubject, value, nameof(SelectedSubject));
        }

        public Student SelectedStudent
        {
            get=> _selectedStudent;
            set=>Set(ref _selectedStudent,value,nameof(SelectedStudent));
        }

        public string SortValue
        {
            get => _sortValue;
            set
            {
                Set(ref _sortValue, value, nameof(SortValue));
                DisplaySubjects();
            }
        }
```

```
        public List<string> ValuesToSort => new List<string>
        {
             "Без Сортировки","Русский язык","Математика","География","Английский
язык","Биология","История","Обществознание","Физическая культура",
        };

        public string SearchValue
        {
            get => _searchValue;
            set
            {
                Set(ref _searchValue, value, nameof(SearchValue));
                DisplaySubjects();
            }
        }

        public Class SelectedClasses
        {
            get => _selectedCasses;
            set
            {
                Set(ref _selectedCasses, value, nameof(SelectedClasses));
                DisplaySubjects();
            }
        }

        public List<Subject> DisplayingSubjects
        {
            get => _displayingSubjects;
                                    set   =>   Set(ref   _displayingSubjects,   value,
nameof(DisplayingSubjects));
        }

        public void DisplaySubjects()
        {
            DisplayingSubjects = Filter(Search(Sort(_subjects)));
        }

        public MarkWindowViewModel()
        {
            //SortValue = ValuesToSort[0];
            using DiplomBetaContext context = new DiplomBetaContext();
            {
```

```
                        _subjects = context.Subjects.Include(p => p.Student).Include(p
=> p.Student.Class).Include(p=>p.Teacher).ToList();

                    }

                    _displayingSubjects = new List<Subject>(_subjects);
                }
                private List<Subject> Filter(List<Subject> classes)
                {
                                    return  classes.Where(p  =>  p.Student.Class.Id  ==
SelectedClasses.Id).ToList();
                }
                private List<Subject> Search(List<Subject> subjects)
                {
                    if ((SearchValue == null) || (SearchValue == string.Empty))
                        return subjects;

                    var value = SearchValue.ToLower();
                                                        return    subjects.Where(p    =>
p.Student.Surname.ToLower().Contains(value)).ToList();
                }

                private List<Subject> Sort(List<Subject> subjects)
                {
                    if (SortValue == ValuesToSort[1])
                                        return  subjects.Where(p  =>  p.SubjectName  ==
ValuesToSort[1]).ToList();
                    else if (SortValue == ValuesToSort[2])
                                        return  subjects.Where(p  =>  p.SubjectName  ==
ValuesToSort[2]).ToList();
                    if (SortValue == ValuesToSort[3])
                                        return  subjects.Where(p  =>  p.SubjectName  ==
ValuesToSort[3]).ToList();
                    else if (SortValue == ValuesToSort[4])
                                        return  subjects.Where(p  =>  p.SubjectName  ==
ValuesToSort[4]).ToList();
                    if (SortValue == ValuesToSort[5])
                                        return  subjects.Where(p  =>  p.SubjectName  ==
ValuesToSort[5]).ToList();
                    else if (SortValue == ValuesToSort[6])
                                        return  subjects.Where(p  =>  p.SubjectName  ==
ValuesToSort[6]).ToList();
                    if (SortValue == ValuesToSort[7])
```

```
                                        return  subjects.Where(p  =>  p.SubjectName  ==
ValuesToSort[7]).ToList();
                    else if (SortValue == ValuesToSort[8])
                                        return  subjects.Where(p  =>  p.SubjectName  ==
ValuesToSort[8]).ToList();
                    else return subjects;

                }
            }
        }
        using System;
        using System.Collections.Generic;
        using System.ComponentModel;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;

        namespace DimplomBetaTest.View.ViewModel
        {
            public class ViewModelBase : INotifyPropertyChanged
            {
                public event PropertyChangedEventHandler? PropertyChanged;
                protected bool Set<T>(ref T field, T value, string propertyName)
                {
                    if (EqualityComparer<T>.Default.Equals(field, value))
                        return false;
                    field = value;
                    OnPropertyChanged(propertyName);
                    return true;
                }
                protected virtual void OnPropertyChanged(string propertyName)
                {
                                                PropertyChanged?.Invoke(this,    new
PropertyChangedEventArgs(propertyName));
                }
            }
        }
        using DimplomBetaTest.Models;
        using System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;
        using System.Windows;
```

```
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace DimplomBetaTest.View
{
    /// <summary>
    /// Логика взаимодействия для LoginWindow.xaml
    /// </summary>
    public partial class LoginWindow : Window
    {
        public LoginWindow()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            using DiplomBetaContext context = new DiplomBetaContext();
            {
                  Enterence enterence = context.Enterences.Where(p => p.Login ==
LoginBox.Text && p.Password == PasswordBox.Password)
                      .FirstOrDefault();
                MainWindow window = new MainWindow();
                if (enterence != null)
                {
                    window.Show();
                    Close();
                }
                else
                {
                    MessageBox.Show("Неправильный логин или пароль", "Ошибка");
                }
            }
        }
    }
}
using DimplomBetaTest.Models;
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DimplomBetaTest.View.ViewModel
{
    public class MarkChangeWindowViewModel : ViewModelBase
    {
        private MarkTable _selectedMarkTable;
        private Subject _Selectedsubject;
        private List<MarkTable> _markTable;
        private List<MarkTable> _displayingTables;

        public List<MarkTable> DisplayingTables
        {
            get => _displayingTables;
            set => Set(ref _displayingTables, value, nameof(DisplayingTables));
        }
        public MarkTable SelectedMarkTable
        {
            get => _selectedMarkTable;
                                    set   =>   Set(ref   _selectedMarkTable,   value,
nameof(SelectedMarkTable));
        }
        public Subject SelectedSubject
        {
            get => _Selectedsubject;
            set => Set(ref _Selectedsubject, value, nameof(SelectedSubject));



        }

        public MarkChangeWindowViewModel()
        {
            using DiplomBetaContext context = new DiplomBetaContext();
            {
                _markTable = context.MarkTables.ToList();
            }
            DisplayingTables = new List<MarkTable>(_markTable);
        }
```