**Оглавление**

[Введение 2](#_Toc101811549)

[1. Описание предметной области 4](#_Toc101811550)

[2. Диаграмма прецендентов 9](#_Toc101811551)

[3. Обзор аналогов 16](#_Toc101811552)

[4. Архитектура разрабатываемого приложения. 22](#_Toc101811553)

[5. Проектирование пользовательского интерфейса 25](#_Toc101811554)

[6. Конструирование макетов графического интерфейса 44](#_Toc101811555)

[7. Построение логической и физической модели базы данных 53](#_Toc101811556)

[8. Разработка алгоритмов функционирования 59](#_Toc101811557)

[9. Выбор инструментальных средств разработки 88](#_Toc101811558)

[10. Описание реализации 94](#_Toc101811559)

[11. Тестирование приложения 105](#_Toc101811560)

[Заключение 112](#_Toc101811561)

[Список использованной литературы 113](#_Toc101811562)

[Приложение А 114](#_Toc101811563)

# **Введение**

Основной задачей системы высшего образования является удовлетворение потребностей государства в специалистах нужного профиля. При этом выпускаемые специалисты могут иметь различное качество подготовки, которое будет зависеть от того, насколько полно выпускник вуза сможет соответствовать предъявляемым к нему требованиям.

Развивающиеся наука и техника предъявляют новые требования к содержанию высшего образования. Специалист каждого нового выпуска того или иного учебного заведения всегда должен иметь более высокий уровень подготовки, чем специалист предыдущего выпуска.

Качество подготовки специалиста во многом определяется программой его обучения, и, в частности, главным документом этой программы - учебным планом вуза (далее УП).

Разработка УП это трудоемкий, объемный процесс, который требует держать в голове огромное количество соотношений. Поэтому необходим инструмент, который позволит формировать ядро (набор дисциплин/модулей, их объем в зачетных единицах, а также указание вида промежуточной аттестации) УП для его дальнейшей детализации.

Объектом выпускной квалификационной работы является процесс формирования учебного плана.

Предметом выпускной квалификационной работы является приложение для формирования ядра учебного плана.

Целью выпускной квалификационной работы является разработка приложения для формирования ядра учебного плана.

Для достижения поставленной цели, необходимо решить следующие задачи:

* + 1. Анализ предметной области и формирование требований к приложению.
    2. Построение и описание диаграммы прецендентов.
    3. Обзор и анализ аналогов.
    4. Архитектура разрабатываемого приложения.
    5. Проектирование пользовательского интерфейса.
    6. Конструирование макетов графического интерфейса.
    7. Разработка алгоритмов функционирования.
    8. Построение логической и физической модели базы данных.
    9. Выбор инструментальных средств разработки.
    10. Описание реализации.
    11. Тестирование приложения.

# **1. Описание предметной области**

В рамках выпускной квалификационной работы необходимо разработать приложение, которое бы позволяло в графическом интерфейсе производить манипуляции над дисциплинами, а также соответствовало общим положениям документа федеральный государственный образовательный стандарт высшего образования (далее - ФГОС ВО).

Настоящий ФГОС ВО представляет собой совокупность требований, обязательных при реализации основных профессиональных образовательных программ высшего образования. Подробнее рассмотрим программы бакалавриата в очной форме обучения по направлению подготовки 09.03.03 Прикладная информатика (далее соответственно - программа бакалавриата, направление подготовки).

Получение образования по программе бакалавриата допускается только в образовательной организации высшего образования (далее - Организация).

Обучение по программе бакалавриата в организациях осуществляется в очной, очно-заочной и заочной формах обучения.

Содержание высшего образования по направлению подготовки определяется программой бакалавриата, разрабатываемой и утверждаемой Организацией самостоятельно. При разработке программы бакалавриата Организация формирует требования к результатам ее освоения в виде универсальных, общепрофессиональных и профессиональных компетенций выпускников (далее вместе - компетенции).

Организация разрабатывает программу бакалавриата в соответствии с ФГОС ВО, с учетом соответствующей примерной основной образовательной программы, включенной в реестр примерных основных образовательных программ (далее - ПООП), если таковая имеется.

Срок получения образования по программе бакалавриата в очной форме обучения, включая каникулы, предоставляемые после прохождения государственной итоговой аттестации, составляет 4 года.

Объем программы бакалавриата составляет 240 зачетных единиц (далее - з.е.) вне зависимости от применяемых образовательных технологий, реализации программы бакалавриата с использованием сетевой формы, реализации программы бакалавриата по индивидуальному учебному плану.

Объем программы бакалавриата, реализуемый за один учебный год, составляет не более 70 з.е. вне зависимости от формы обучения, применяемых образовательных технологий, реализации программы бакалавриата с использованием сетевой формы, реализации программы бакалавриата по индивидуальному учебному плану (за исключением ускоренного обучения), а при ускоренном обучении - не более 80 з.е.

Требования к структуре программы бакалавриата.

Структура программы бакалавриата включает следующие блоки:

Блок 1 «Дисциплины (модули)», объемом не менее 160 з.е.;

Блок 2 «Практика», объемом не менее 20 з.е.;

Блок 3 «Государственная итоговая аттестация», объемом не менее 9 з.е.

Содержание блоков структуры программ бакалавриата.

Блок 1 «Дисциплины (модули)».

1. Обязательная часть:
   1. обеспечивает формирование универсальных, общепрофессиональных обязательных профессиональных компетенций;
   2. предписывает изучение модулей по философии, истории (истории России, всеобщей истории), иностранному языку, безопасности жизнедеятельности;
   3. объем контактной работы не менее 40% общего объема программы бакалавриата.
2. Часть, формируемая участниками образовательных отношений.

Блок 2 «Практика».

1. Учебная практика:
   1. ознакомительная практика;
   2. проектно-технологическая практика;
   3. эксплуатационная практика;
   4. научно-исследовательская работа (получение первичных навыков научно-исследовательской работы).
2. Производственная практика типов:
   1. технологическая (проектно-технологическая) практика;
   2. эксплуатационная практика;
   3. научно-исследовательская работа.

Типы практик определяются из перечня, могут быть установлены организацией самостоятельно.

Блок 3 «Государственная итоговая аттестация».

Включает:

* подготовку к сдаче и сдачу государственного экзамена;
* выполнение и защита выпускной квалификационной работы.

Редактирование УП в графическом интерфейсе предполагает наличие трех основных окон:

1. Главное окно – здесь будут происходить основные манипуляции дисциплин между семестрами, банком и изменением з.е. для каждой дисциплины.
2. Окно редактирования дисциплин – работает как справочник дисциплин. Отсюда происходит пополнение банка дисциплин главного окна.
3. Окно редактирования категорий – отвечает за наличие категории у каждой дисциплины.

Исходя из общих положений, можно выделить функциональные требования к разрабатываемой информационной системе.

1. Требования к главному окну:

* открыть проект существующий проект;
* сохранить проект;
* экспорт в Excel;
* перейти к предыдущему действию;
* перейти к следующему действию;
* изменить значение з.е. теоретического обучения для дисциплины;
* отметить наличие промежуточной аттестации для дисциплины;
* отметить наличие курсовой работы/курсового проекта для дисциплины;
* удалить дисциплину из «Банка дисциплин»;
* переместить дисциплину в «Банк дисциплин»;
* удалить дисциплину из семестра;
* переместить дисциплину в семестр;
* открыть редактор дисциплин;
* открыть справку о программе.

1. Требования к окну редактирования дисциплин:

* ввести название дисциплины;
* изменить название дисциплины;
* ввести сокращенное название дисциплины;
* изменить сокращенное название дисциплины;
* выбрать блок;
* изменить количество з.е. теоретического обучения;
* отметить наличие промежуточной аттестации;
* отметить наличие курсовой работы/проект;
* добавить дисциплину;
* удалить дисциплину;
* выбрать дисциплину;
* фильтровать дисциплину;
* поиск дисциплин.

1. Требования к окну редактирования категориями:

* ввести название категории;
* изменить название категории;
* выбрать цвет категории;
* изменить цвет категории;
* добавить категорию;
* удалить категорию;
* выбрать категорию.

Также можно выделить нефункциональные требования:

* + 1. работа с УП в графическом интерфейсе;
    2. представление дисциплины в таблице в виде плитки с количеством з.е. теоретической, промежуточной аттестации и курсовой работы/проекта;
    3. подсказки при работе с приложением;
    4. предупреждения в случае превышения з.е. за учебный курс;
    5. стабильность работы;
    6. быстродействие системы;
    7. инструкция для работы с приложением.

# **2. Диаграмма прецендентов**

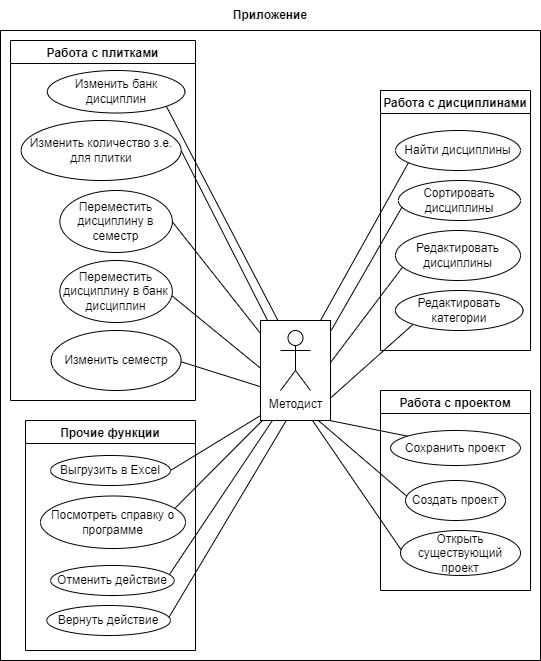
Диаграмма прецендентов является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Разработка данной диаграммы преследует следующие цели:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Разрабатываемое приложение необходимо тому, кто будет непосредственно составлять учебный план. Следовательно, в данной модели будет один актер – «Методист», который будет непосредственно взаимодействовать с приложением (см. рис .1). Диаграмма прецендентов приложения для формирования ядра учебного плана состоит из следующих пакетов:

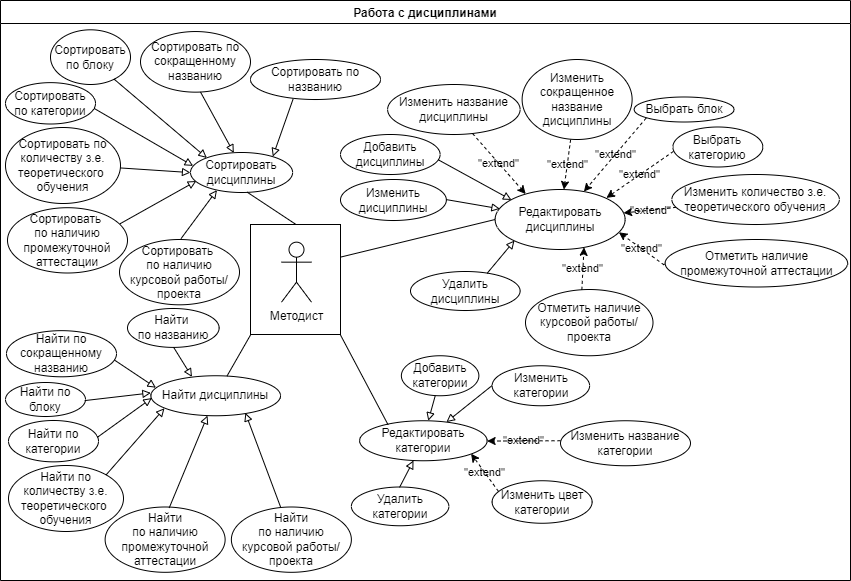
1. работа с дисциплинами;
2. работа с проектом;
3. работа с плитками;
4. прочие функции.



**Рисунок 1 – Диаграмма прецендентов**

Пакет «Работа с дисциплинами» (см. рис. 2) состоит из следующих основных прецендентов:

1. редактировать дисциплины;
2. редактировать категории;
3. найти дисциплины;
4. сортировать дисциплины.



**Рисунок 2 – Пакет «Работа с дисциплинами» диаграммы прецендентов**

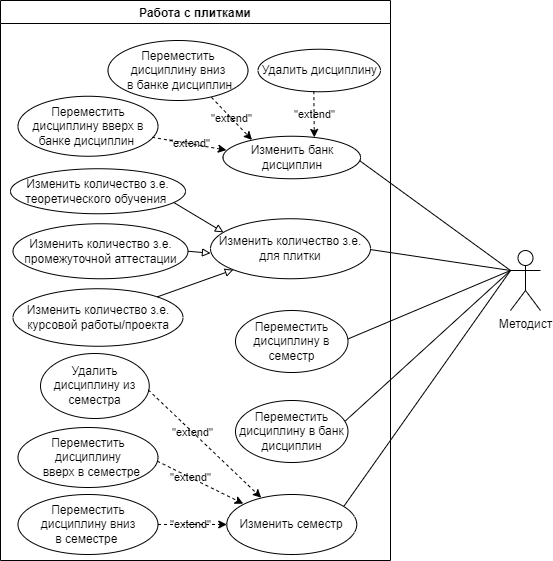
Прецедент «Редактировать дисциплины» позволяет выполнять операции над дисциплинами. В тоже время преценденты «Изменить название дисциплины», «Изменить сокращенное название дисциплины», «Выбрать блок», «Выбрать категорию», «Изменит количество з.е. теоретического обучения», «Отметить наличие промежуточной аттестации» и «Отметить наличие курсовых работ/проектов» и расширяют базовую последовательность действий прецендента «Редактировать дисциплины». Также, преценденты «Добавить дисциплины», «Изменить дисциплины» и «Удалить дисциплины» конкретизируют прецендент «Редактировать дисциплины» по базовым операциям. Следовательно, они связаны между собой с помощью отношения обобщения.

Прецедент «Редактировать категории» позволяет выполнять операции над категориями. Преценденты «Изменить название категории» и «Изменить цвет категории» расширяют базовую последовательность действий прецендента «Редактировать категории». Дочерними прецендентами являются «Добавить категории», «Изменить категории» и «Удалить категории». Перечисленные преценденты конкретизируют прецендент «Редактировать категории» по базовым операциям, поэтому они связаны между собой отношениями обобщения.

За возможность поиска и сортировки отвечают преценденты «Найти дисциплины» и «Сортировать дисциплины».

Пакет «Работа с плитками» (см. рис. 3) состоит из следующих основных прецендентов:

1. изменить банк дисциплин;
2. изменить семестр;
3. изменить количество з.е. для плитки;
4. переместить дисциплину в семестр;
5. переместить дисциплину в банк дисциплин.



**Рисунок 3 – Пакет «Работа с плитками» диаграммы прецендентов**

Прецедент «Изменить банк дисциплин» позволяет совершать манипуляции над плитками в пределах банка дисциплин. В тоже время преценденты «Удалить дисциплину», «Переместить дисциплину вниз в банке дисциплин» и «Переместить дисциплину вверх в банке дисциплин» расширяют базовую последовательность действий прецендента «Изменить банк дисциплин».

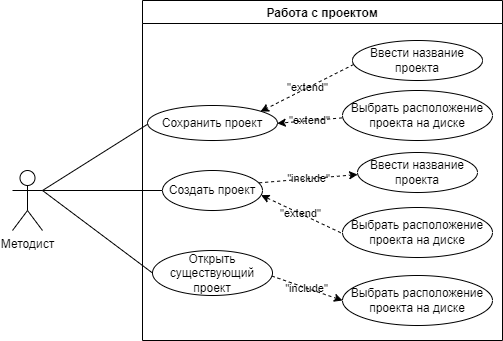
Прецедент использования «Изменить количество з.е. для плитки» дает возможность изменять зачетные единицы индивидуально для каждой плитки. Дочерние прецеденты «Изменить значение з.е. теоретического обучения», «Изменить значение з.е. промежуточной аттестации» и «Изменить значение з.е. курсовой работы/проекта» конкретизируют прецедент «Изменить количество з.е. для плитки» по видам зачетных единиц и поэтому они связаны между собой отношениями обобщения.

Прецедент «Изменить семестр» позволяет совершать манипуляции над плитками в пределах семестра. В тоже время преценденты «Удалить дисциплину из семестра», «Переместить дисциплину вниз в семестре» и «Переместить дисциплину вверх в семестре» расширяют базовую последовательность действий прецендента «Изменить семестр».

Перемещение дисциплин из банка дисциплин в семестр и из семестра в банк дисциплин происходит при выполнении прецендентов «Переместить дисциплину в семестр» и «Переместить дисциплину в банк дисциплин»

Пакет «Работа с проектом» (см. рис. 4) состоит из следующих основных прецендентов:

1. создать проект;
2. сохранить проект;
3. открыть существующий проект.



**Рисунок 4 – Пакет «Работа с проектом» диаграммы прецендентов**

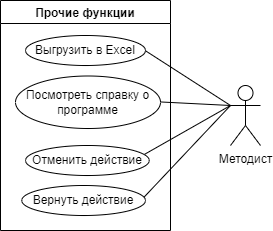
Выполнение прецендента «Создать проект» позволяет создать на физическом носителе экземпляр проекта, при этом необходимо ввести название проекта. Это можно сделать, выполнив прецендент «Ввести название проекта». За возможность выбора расположения созданного проекта на физическом носителе отвечает прецендент «Выбрать расположение проекта на диске», который расширяет расширяют базовую последовательность действий прецендента «Создать проект».

Методист, выполняя прецендент «Открыть существующий проект» должен выбрать расположение уже существующего проекта. Этого можно добиться, выполнив прецендент «Выбрать расположение проекта на диске»

Прецедент «Сохранить проект» позволяет сохранить уже существующий или только что созданный проект на физическом носителе, а преценденты «Ввести название проекта» и «Выбрать расположение проекта на диске» расширяют базовую последовательность действий данного прецендента.

Пакет «Прочие функции» (см. рис. 5) состоит из следующих прецендентов:

1. выгрузить в Excel;
2. посмотреть справку о программе;
3. отменить действие;
4. вернуть действие.



**Рисунок 5 – Пакет «Работа с проектом» диаграммы прецендентов**

При выполнении прецендента «Выгрузить в Excel» на физическом носителе появится файл, в котором проект будет представлен в формате учебного плана.

Прецендент «Посмотреть справку о программе» позволяет нам получить информацию и о том, как работать с данной программой.

Преценденты «Отменить действие» и «Вернуть действие» позволяют выполнить откат к выполненным действиям.

# **3. Обзор аналогов**

Рассмотрим несколько программных продуктов для формирования УП.

1. АС «Учебные планы». Данная система позволяет создать в рамках высшего учебного заведения единую систему автоматизированного планирования учебного процесса. УП, создаваемые в АС «Учебные Планы» полностью совместимы с форматом, используемым в процедуре государственной аккредитации.

Система также включает в себя комплект формализованных ФГОС, которые можно использовать для создания на их основе учебных планов и проверки качества УП.

Интерфейс программы представлен на рисунке 6.



**Рисунок 6 – Интерфейс приложения АС «Учебные планы»**

Функциональные возможности АС «Учебные планы», следующие:

- возможность хранения в одном документе несколько планов по одному направлению с различными профилями (направленностями) и настройку отличий профилей;

- реализацию заочной формы обучения не только по сессиям, но и по семестрам;

- загрузку планов универсального макета и Excel-макетов (планов ФГОС-3, ФГОС-3+ и ФГОС-3++);

- доступность нескольких вложений в модулях (заголовках разделов);

- организацию модулей, содержащих составные части различных блоков (частей блоков) (например, модуль дисциплин по физической культуре и спорту);

- добавление целых модулей по выбору;

- добавление контроля по модулю (заголовку) вне зависимости от наличия контроля по его составляющим;

- взаимодействие с обновляемой базой образовательных стандартов с видами деятельности, включая загрузку компетенций для актуальных видов деятельности, упорядоченность профессиональных компетенций по видам деятельности;

- возможность у дисциплин по выбору иметь в одном блоке различные показатели (например, виды аудиторных занятий);

- отображение только актуальных видов занятий в учебном плане;

- добавление новых видов аудиторных занятий с определением их типа;

- возможность планирования в астрономических часах;

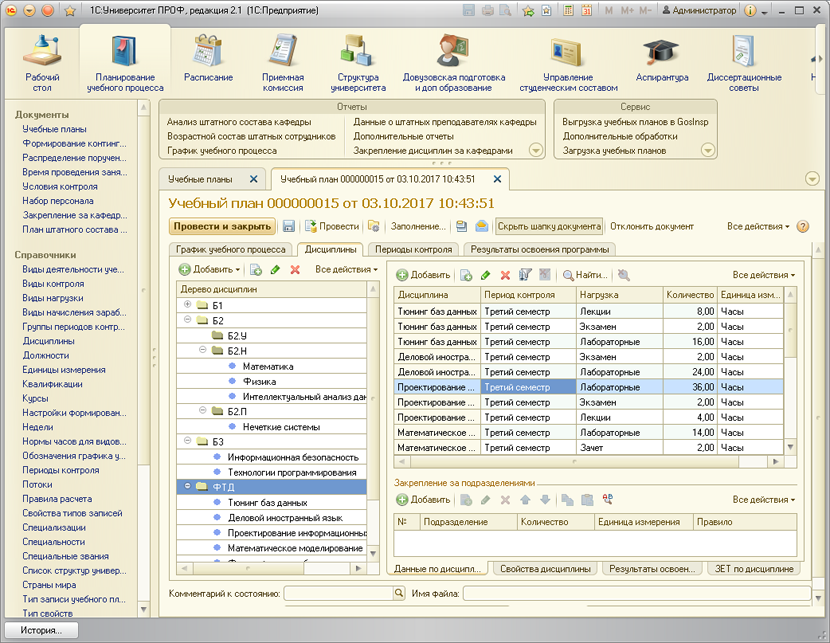
- формирование индивидуального плана студента;

- возможность создания планов дополнительного профессионального образования.

2. Программный продукт «1С: Университет ПРОФ». Представляет собой решение для автоматизации управленческой деятельности в учреждениях высшего профессионального образования и разработан на технологической платформе «1С: Предприятие 8.3».

Формирование УП в «1С: Университет ПРОФ» осуществляется путем ввода данных на форме с требуемыми атрибутами.

Интерфейс программы представлен на рисунке 7.



**Рисунок 7 – Интерфейс программного продукта «1С: Университет ПРОФ»**

Функционал программного продукта, следующий:

1. формирование и учет учебных и рабочих учебных планов в соответствии с ГОС, ФГОС ВПО и ФГОС ВО;
2. поддержка уровневой системы (бакалавр, магистр и специалист);
3. создание, хранение и обработка графиков учебного процесса;
4. закрепление дисциплин учебного плана за кафедрами и подразделениями вуза;
5. Интеграция с GosInsp (загрузка и выгрузка данных в формате .xml):

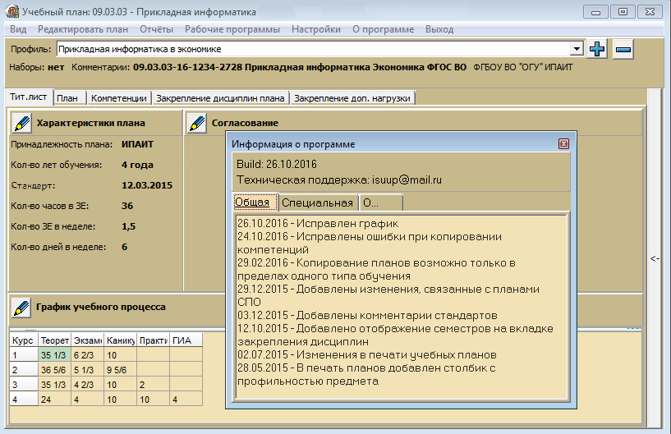
* возможность единичной и массовой загрузки учебных планов;
* возможность автоматического поиска планов-соответствий в «1С: Университет» на основании реквизитов планов формата .xml при массовой загрузке;
* возможность создания недостающих учебных планов при загрузке;
* настройка правил проведения загружаемых учебных планов;
* возможность автоматического создания комплекта рабочих учебных планов на основании загружаемого базового;

1. учет специализаций в документе «Учебный план»;
2. в документе «Учебный план» могут быть указаны основной и дополнительный сроки обучения;
3. возможность пометки учебного плана как учебного плана по сокращенной программе;
4. возможность проверки учебного плана согласно заданным критериям;
5. возможность копирования данных из одного учебного плана в другой (как одиночное, так и массовое копирование);
6. возможность автоматической замены реквизитов документа «Учебный план» в соответствии с заданными условиями;
7. возможность выгрузки учебных планов в файлы формата .xml (возможна как одиночная, так и массовая выгрузка);
8. вывод на печать:

* учебных планов;
* графиков учебного процесса;
* форм согласования закрепления дисциплин;
* матрицы компетенций.

3. «Информационная система управления учебным процессом вуза» (далее - ИСУУП). Предназначена для автоматизации работы структурных подразделений вуза, участвующих в управлении учебным процессом, и ставит своей целью повышение качества обучения студентов.

ИСУУП состоит из подсистем, соответствующих основным процессам подготовки и сопровождения учебной деятельности вуза (см. рис. 8). Каждая из подсистем может эксплуатироваться как в одном структурном подразделении, так и в нескольких.



**Рисунок 8 – Интерфейс информационной системы управления учебным процессом вуза**

Недостатками всех этих трех систем управления УП являются:

- отсутствие возможности работать с ядром УП на верхнем уровне, т.е. предназначен только для ввода и контроля основных показателей УП;

- отсутствие наглядности представления всего УП образовательной программы и механизмов интерактивной корректировки.

# **4. Архитектура разрабатываемого приложения.**

Архитектура приложения описывает шаблоны и методы, используемые для проектирования и создания приложения, и дает дорожную карту и рекомендации, которым необходимо следовать при создании приложения, чтобы в итоге получить хорошо структурированное приложение.

Шаблоны проектирования программного обеспечения могут помочь создать приложение, а также описывает повторяющееся решение проблемы.

Шаблоны могут быть связаны друг с другом для создания более универсальных архитектур приложений. Вместо того, чтобы полностью создавать архитектуру самостоятельно, можно использовать существующие шаблоны проектирования, которые также гарантируют, что все будет работать так, как должно.

Как часть архитектуры приложения, будут существовать как интерфейсные, так и серверные сервисы. Интерфейсная разработка связана с пользовательским интерфейсом приложения, в то время как внутренняя разработка сосредоточена на предоставлении доступа к данным, службам и другим существующим системам, которые обеспечивают работу приложения.

За основу архитектуры приложения будет взят шаблон проектирования Model-View-ViewModel (далее MVVM).

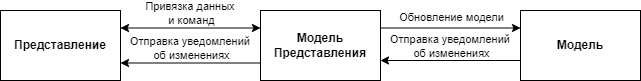
MVVM — это шаблон проектирования программного обеспечения, который структурирован для разделения программной логики и дисциплин управления пользовательского интерфейса. MVVM также известен как model-view-binder и был создан архитекторами Microsoft Кеном Купером и Джоном Госсманом.

Как и многие другие шаблоны проектирования, MVVM помогает упорядочивать код и разбивать программы на модули, чтобы упростить и ускорить разработку, обновление и повторное использование кода. Шаблон часто используется в программном обеспечении для представления Windows.

Шаблон MVVM используется в Windows Presentation Foundation (WPF), который работает в Microsoft .NET. Silverlight, мультимедийный плагин Microsoft WPF, эквивалентный Интернету, также использует MVVM.

Шаблон MVVM делится на Модель, Представление и Модель Представления (см. рис. 9).

* + 1. Представление — графический интерфейс (окна, списки, кнопки и т. п.). Выступает подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновлённое значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо дисциплину интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.
    2. Модель Представления с одной стороны, абстракция Представления, а с другой — обёртка данных из Модели, подлежащих связыванию. То есть, она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.
    3. Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.



**Рисунок 9 – Шаблон MVVM**

Преимущества MVVM:

1. облегчает параллельную разработку пользовательского интерфейса и строительных блоков, которые его поддерживают;
2. абстрагирует представление и, таким образом, уменьшает количество бизнес-логики (или клея), требуемой в коде, стоящем за ним;
3. модель представления может быть протестирована без проблем с автоматизацией пользовательского интерфейса и взаимодействием.

Теперь перейдем к построению архитектуры разрабатываемого приложения. В ходе проектирования были выделены следующие представления:

1. редактор дисциплин;
2. редактор категорий.

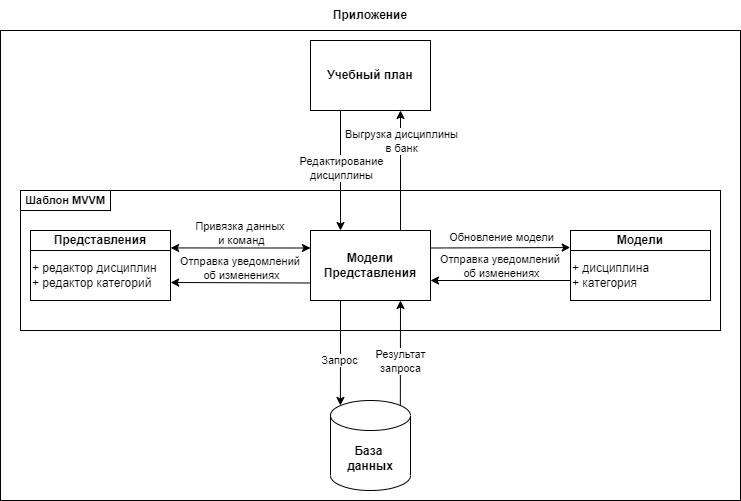
Список моделей выглядит следующим образом:

1. дисциплина;
2. категория.

Для хранения данных будет использоваться база данных, с которой будет взаимодействовать описанные модели.

Добавление дисциплин непосредственно в учебный план происходит путем редактирования дисциплин, и их последующей выгрузки.

Разработанная схема взаимодействия объектов представлена на рисунке 10.



**Рисунок 10 – Архитектура приложения**

С MVVM каждый фрагмент кода является более детализированным и внешние и внутренние зависимости находятся в отдельных фрагментах кода от частей с базовой логикой.

# **5. Проектирование пользовательского интерфейса**

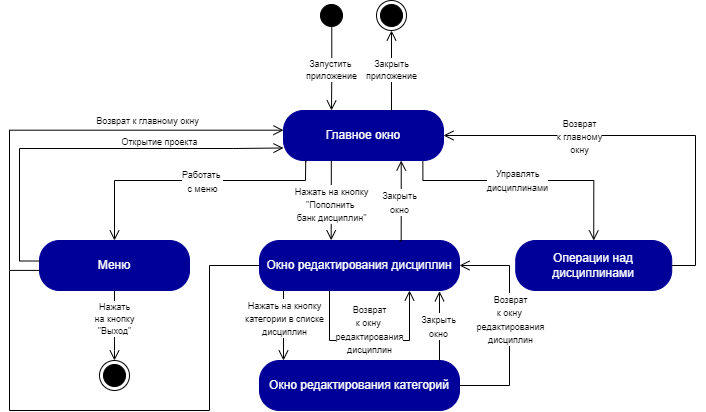
Для моделирования поведения приложения будет использоваться диаграмма состояний.

Диаграмма состоянийпо существу является графом специального вида, который представляет некоторый автомат.

Главное предназначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение дисциплины модели в течение его жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий.

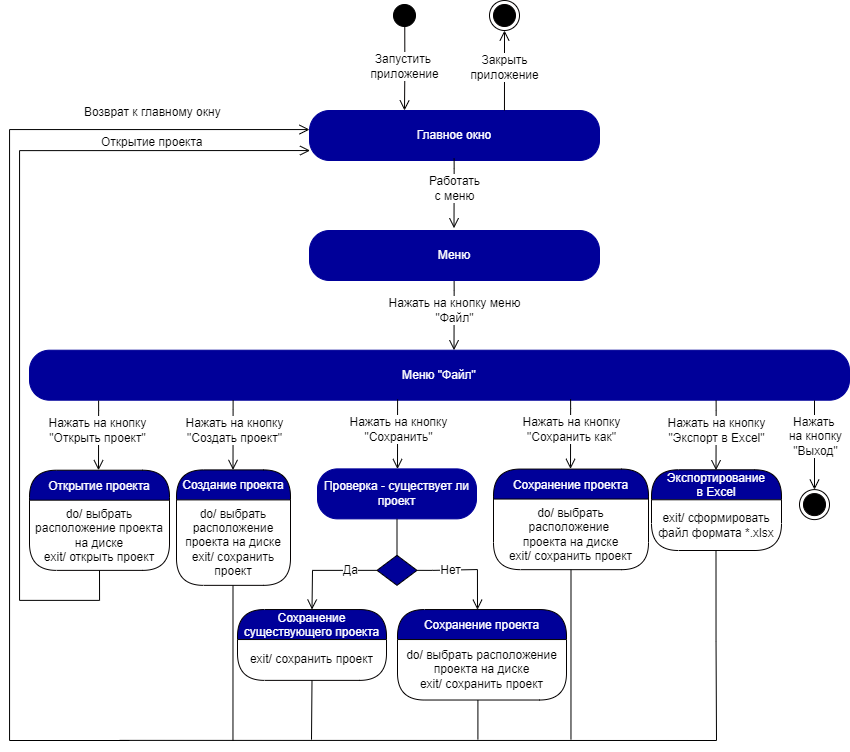
Общий вид диаграммы, представленный на рисунке 11, состоит из 5 состояний:

1. главное окно;
2. меню;
3. окно редактирования дисциплин;
4. окно редактирования категорий;
5. операции над дисциплинами.

**Рисунок 11 – Общий вид диаграммы состояний приложения**

Дальше будет рассмотрено и описано каждое состояние.

1. Состояние «Меню» декомпозируется на две детализированные диаграммы. Первая представлена на рисунке 12, вторая на рисунке 13.



**Рисунок 12 – Детализированная диаграмма состояния «Меню»**

После перехода из начального состояния «Запустить приложение» происходит переход в состояние «Главное» окно. Далее, приложение попадает в состояние «Меню», откуда можно по нажатию на кнопку меню «Файл» перейти в состояние «Меню «Файл».

По нажатию на кнопку «Открыть проект» произойдет переход в состояние «Открытие проекта». Текущее состояние содержит метку выполняющегося действия «Выбрать расположение проекта на диске» и метку выходного действия «Открыть проект». Выполнив перечисленные действия, откроется проект, и произойдет переход в состояние «Главное окно».

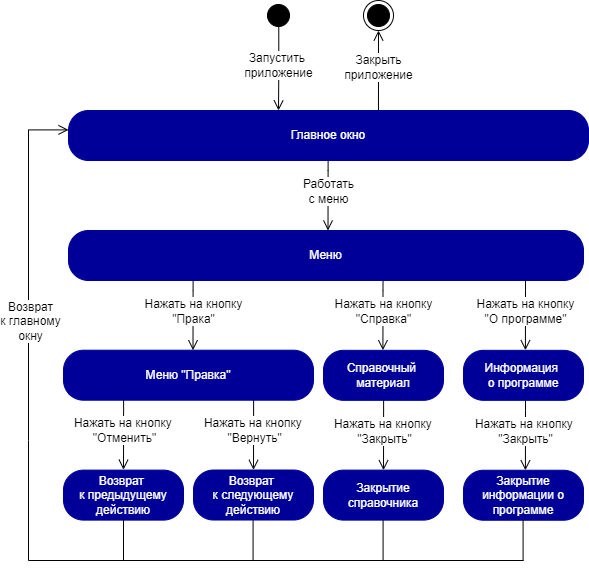
По нажатию на кнопку «Создать проект» произойдет переход в состояние «Создание проекта». Текущее состояние содержит метку выполняющегося действия «Выбрать расположение проекта на диске» и метку выходного действия «Сохранить проект». Выполнить перечисленные действия, произойдет возврат в состояние «Главное окно».

По нажатию на кнопку «Сохранить» произойдет переход в состояние «Проверка – существует ли проект». После, произойдет переход в состояние «Сохранение текущего проекта», если проект существует, и в ином случае в состояние «Сохранение проекта», если нет. Далее произойдет возврат в состояние «Главное окно».

По нажатию на кнопку «Сохранить как» произойдет переход в состояние «Сохранение проекта». Текущее состояние содержит метку выполняющегося действия «Выбрать расположение проекта на диске» и метку выходного действия «Сохранить проект». Выполнив перечисленные действия, произойдет возврат в состояние «Главное окно».

По нажать на кнопку «Экспорт в Excel» произойдет переход в состояние «Экспортирование в Excel». Текущее состояние содержит метку выходного действия «Сформировать файл формата \*.xlsx». После произойдет возврат в состояние «Главное окно».

По нажатию на кнопку «Выход» или закрытию приложения произойдет переход в конечное состояние.



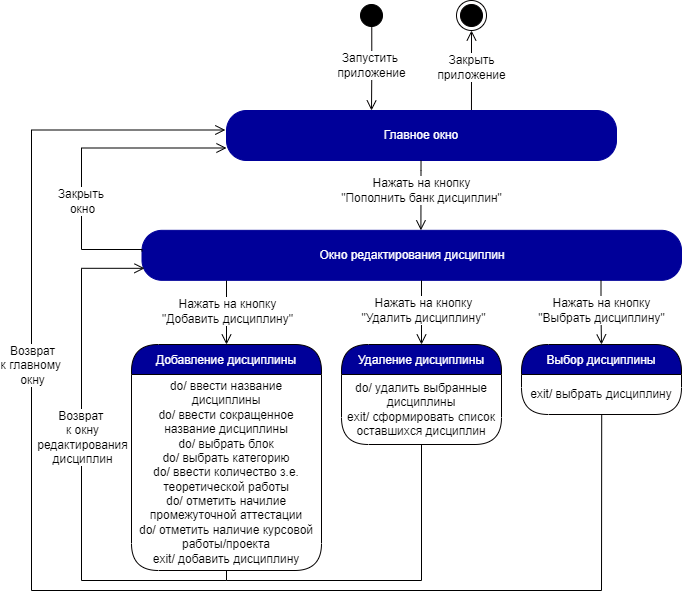
**Рисунок 13 – Детализированная диаграмма состояния «Меню»**

По нажатию на кнопку «Правка» произойдет переход в состояние «Меню «Правка». Из текущего состояния по нажатию на кнопки «Отменить» и «Вернуть» произойдет переход в состояния «Возврат к предыдущему действию» и «Возврат к следующему действию» соответственно. После, выполнится возврат в состояние «Главное окно»

По нажатию на кнопку «Справка» произойдет переход в состояние «Справочный материал». Из текущего состояния по нажатию на кнопку «Закрыть» произойдет переход в состояние «Закрытие справочника» и возврат в состояние «Главное окно».

По нажатию на кнопку «Информация о программе» произойдет переход в состояние «Информация о программе». Из текущего состояния по нажатию на кнопку «Закрыть» произойдет переход в состояние «Закрытие информации о программе» и возврат в состояние «Главное окно».

2. Состояние «Окно редактирования дисциплин» декомпозируется на шесть дочерних диаграмм, представленных на рисунках 14-19.



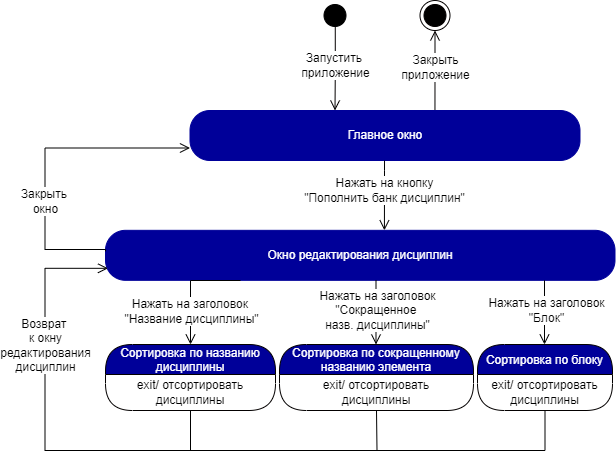
**Рисунок 14 – Детализированная диаграмма состояния «Окно редактирования дисциплин»**

По нажатию на кнопку «Пополнить банк дисциплин» произойдет переход в состояние «Окно редактирования дисциплин». Из текущего состояния по нажатию на кнопки возможен переход в три состояния.

По нажатию на кнопку «Добавить дисциплину» произойдет переход в состояние «Добавление дисциплины». Текущее состояние содержит следующие метки выполняющихся действий: «Ввести название дисциплины», «Ввести сокращенное название дисциплины», «Выбрать блок», «Выбрать категорию», «Ввести количество з.е. теоретической работы», «Отметить наличие промежуточной аттестации» и «Отметить наличие курсовой работы/проект». Также есть метка выходного действия «Добавить дисциплину». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

По нажатию на кнопку «Удалить дисциплину» произойдет переход в состояние «Удаление дисциплины». Текущее состояние содержит метку выполняющегося действия «Удалить выбранные дисциплины» и метку выходного действия «Сформировать список оставшихся дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

По нажатию на кнопку «Выбрать дисциплину» произойдет переход в состояние «Выбор дисциплины». Текущее состояние содержит метку выходного действия «Выбрать дисциплину». Далее происходит переход в состояние «Главное окно».



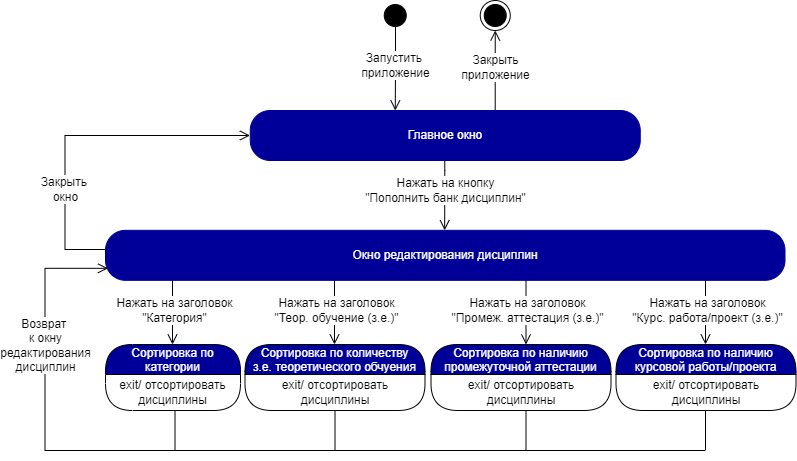
**Рисунок 15 – Детализированная диаграмма состояния «Окно редактирования дисциплин»**

По нажатию на заголовок с названием «Название дисциплины» произойдет переход в состояние «Сортировка по названию дисциплины». Текущее состояние содержит метку выходного действия «Отсортировать дисциплины». Далее происходит возврат в состояние «Окно редактирования дисциплин».

По нажатию на заголовок с названием «Сокращенное назв. дисциплины» произойдет переход в состояние «Сортировка по сокращенному названию дисциплины». Текущее состояние содержит метку выходного действия «Отсортировать дисциплины». Далее происходит возврат в состояние «Окно редактирования дисциплин».

По нажатию на заголовок с названием «Блок» произойдет переход в состояние «Сортировка по блоку». Текущее состояние содержит метку выходного действия «Отсортировать дисциплины». Далее происходит возврат в состояние «Окно редактирования дисциплин».

Также, можно напрямую выйти из состояния «Окно редактирования дисциплин» в состояние «Главное окно».



**Рисунок 16 – Детализированная диаграмма состояния «Окно редактирования дисциплин»**

По нажатию на заголовок с названием «Категория» произойдет переход в состояние «Сортировка по категории». Текущее состояние содержит метку выходного действия «Отсортировать дисциплины». Далее происходит возврат в состояние «Окно редактирования дисциплин».

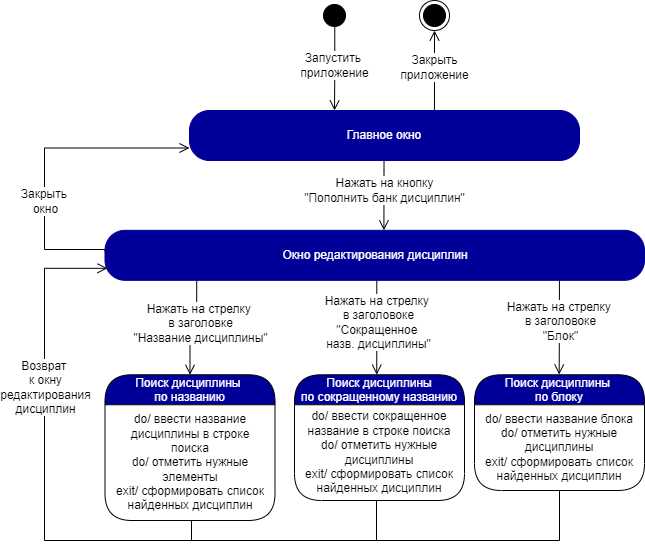
По нажатию на заголовок с названием «Теор. Обучение (з.е.)» произойдет переход в состояние «Сортировка по количеству з.е. теоретического обучения». Текущее состояние содержит метку выходного действия «Отсортировать дисциплины». Далее происходит возврат в состояние «Окно редактирования дисциплин».

По нажатию на заголовок с названием «Теор. Обучение (з.е.)» произойдет переход в состояние «Сортировка по количеству з.е. теоретического обучения». Текущее состояние содержит метку выходного действия «Отсортировать дисциплины». Далее происходит возврат в состояние «Окно редактирования дисциплин».

По нажатию на заголовок с названием «Промеж. аттестация (з.е.)» произойдет переход в состояние «Сортировка по наличию промежуточной аттестации». Текущее состояние содержит метку выходного действия «Отсортировать дисциплины». Далее происходит возврат в состояние «Окно редактирования дисциплин».

По нажатию на заголовок с названием «Курс. работа/проект (з.е.)» произойдет переход в состояние «Сортировка по наличию курсовой работы/проекта». Текущее состояние содержит метку выходного действия «Отсортировать дисциплины». Далее происходит возврат в состояние «Окно редактирования дисциплин».

Также, можно напрямую выйти из состояния «Окно редактирования дисциплин» в состояние «Главное окно».



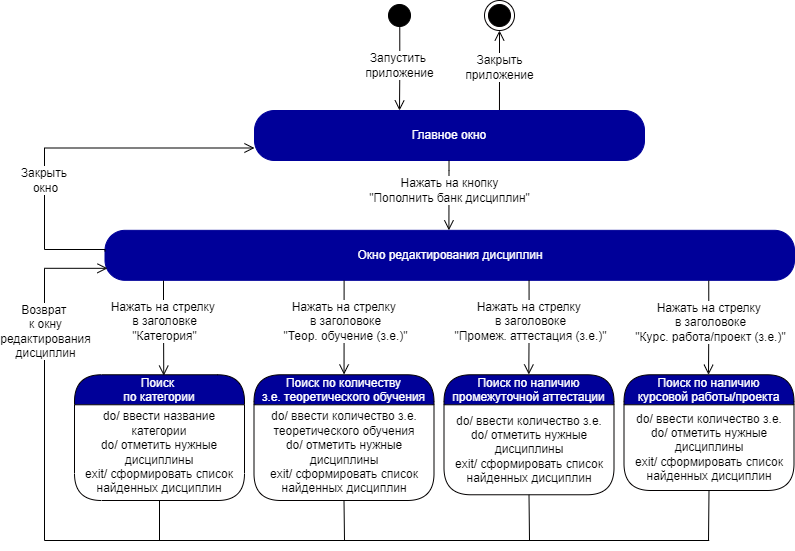
**Рисунок 17 – Детализированная диаграмма состояния «Окно редактирования дисциплин»**

По нажатию на стрелку в заголовке с названием «Название дисциплины» произойдет переход в состояние «Поиск по названию». Текущее состояние содержит две метки выполняющегося действия «Ввести название дисциплины в строке поиска» и «Отметить нужные дисциплины», а также метку выходного действия «Сформировать список найденных дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

По нажатию на стрелку в заголовке с названием «Сокращенное назв. дисциплины» произойдет переход в состояние «Поиск по сокращенному названию». Текущее состояние содержит две метки выполняющегося действия «Ввести сокращенное название дисциплины в строке поиска» и «Отметить нужные дисциплины», а также метку выходного действия «Сформировать список найденных дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

По нажатию на стрелку в заголовке с названием «Блок» произойдет переход в состояние «Поиск дисциплины по блоку». Текущее состояние содержит две метки выполняющегося действия «Ввести название блока» и «Отметить нужные дисциплины», а также метку выходного действия «Сформировать список найденных дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

Также, можно напрямую выйти из состояния «Окно редактирования дисциплин» в состояние «Главное окно».



**Рисунок 18 – Детализированная диаграмма состояния «Окно редактирования дисциплин»**

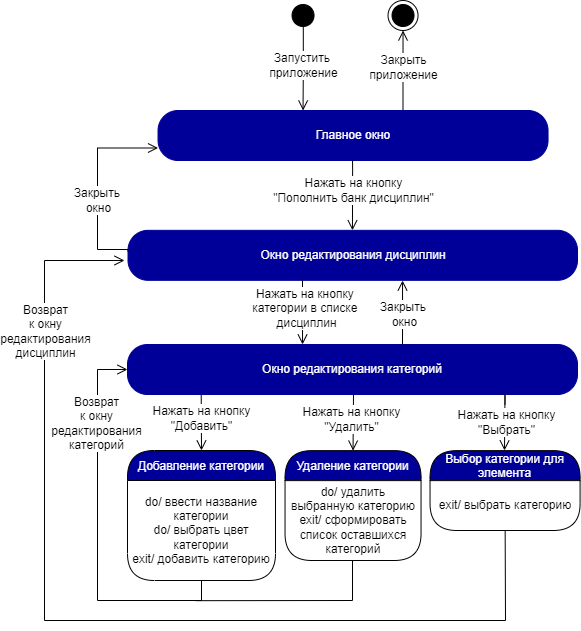
По нажатию на стрелку в заголовке с названием «Категория» произойдет переход в состояние «Поиск по категории». Текущее состояние содержит две метки выполняющегося действия «Ввести название категории» и «Отметить нужные дисциплины», а также метку выходного действия «Сформировать список найденных дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

По нажатию на стрелку в заголовке с названием «Теор. обучение (з.е.)» произойдет переход в состояние «Поиск по количеству з.е. теоретического обучения». Текущее состояние содержит две метки выполняющегося действия «Ввести количество з.е. теоретического обучения» и «Отметить нужные дисциплины», а также метку выходного действия «Сформировать список найденных дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

По нажатию на стрелку в заголовке с названием «Промеж. аттестация (з.е.)» произойдет переход в состояние «Поиск по наличию промежуточной аттестации». Текущее состояние содержит две метки выполняющегося действия «Ввести количество з.е.» и «Отметить нужные дисциплины», а также метку выходного действия «Сформировать список найденных дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

По нажатию на стрелку в заголовке с названием «Курс. работа/проект (з.е.)» произойдет переход в состояние «Поиск по наличию курсовой работы/проекта». Текущее состояние содержит две метки выполняющегося действия «Ввести количество з.е.» и «Отметить нужные дисциплины», а также метку выходного действия «Сформировать список найденных дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

Также, можно напрямую выйти из состояния «Окно редактирования дисциплин» в состояние «Главное окно».



**Рисунок 19 – Детализированная диаграмма состояния «Окно редактирования дисциплин»**

По нажатию на кнопку категории в списке дисциплин произойдет переход в состояние «Окно редактирования категорий». Из текущего состояния по нажатию на кнопки возможен переход в три состояния.

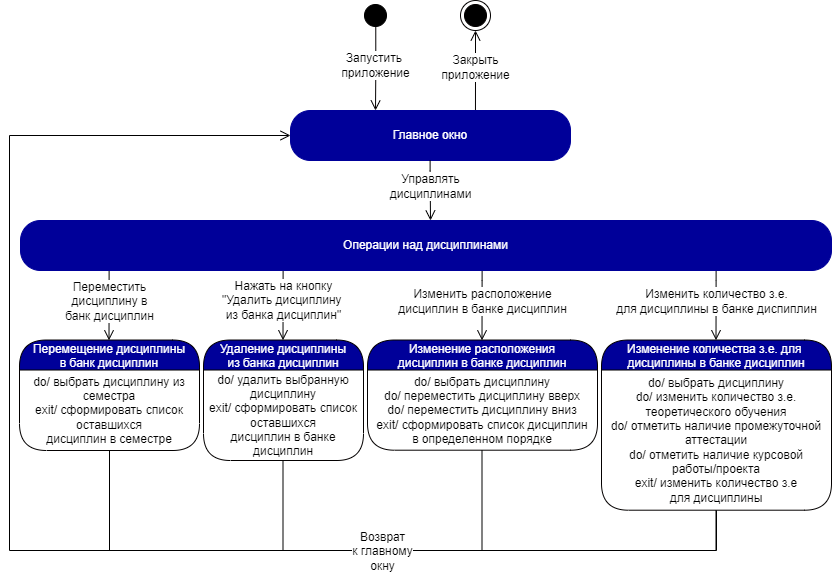
По нажатию на кнопку «Добавить» произойдет переход в состояние «Добавление категории». Текущее состояние содержит две метки выполняющегося действия «Ввести название категории» и «Выбрать цвет категории», а также метку выходного действия «Добавить категорию». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования категорий».

По нажатию на кнопку «Удалить» произойдет переход в состояние «Удаление категории». Текущее состояние содержит метку выполняющегося действия «Удалить выбранную категорию» и метку выходного действия «Сформировать список оставшихся категорий». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования категорий».

По нажатию на кнопку «Выбрать» произойдет переход в состояние «Выбор категории для дисциплины». Текущее состояние содержит метку выходного действия «Выбрать категорию». Выполнив перечисленные действия, произойдет переход в состояние «Окно редактирования дисциплин».

Также, можно напрямую выйти из состояния «Окно редактирования категорий» в состояние «Окно редактирования дисциплин».

3. Состояние «Операции над дисциплинами» декомпозируется на две дочерних диаграммы, представленных на 20 и 21.



**Рисунок 20 – Детализированная диаграмма состояния «Операции над дисциплинами»**

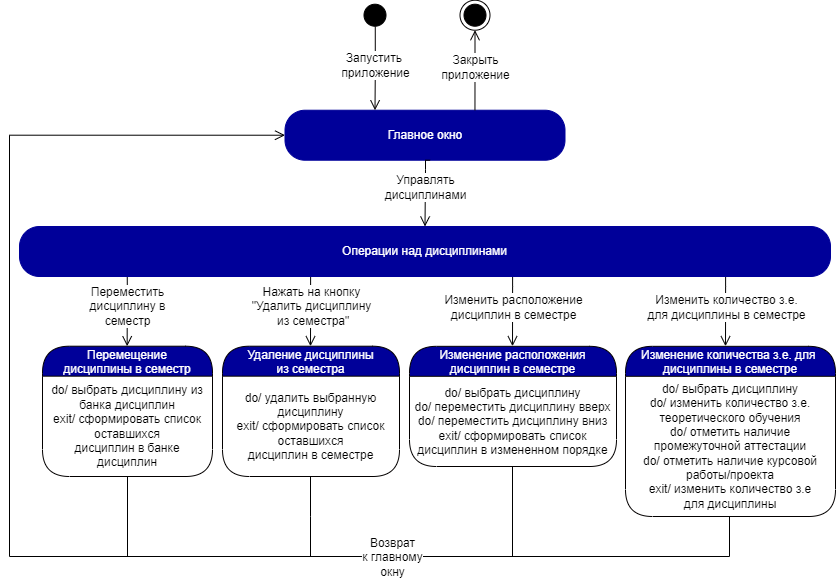
Переход из состояния «Главное окно» в состояние «Операции над дисциплинами» происходит при управлении дисциплинами.

После перемещения дисциплины в банк дисциплин происходит переход в состояние «Перемещение дисциплины в банк дисциплин». Текущее состояние содержит метку выполняющегося действия «Выбрать дисциплину из семестра» и метку выходного действия «Сформировать список оставшихся дисциплин в семестре». Выполнив перечисленные действия, произойдет переход в состояние «Главное окно».

После нажатия на кнопку «Удалить дисциплину из банка дисциплин» происходит переход в состояние «Удаление дисциплины из банка дисциплин». Текущее состояние содержит метку выполняющегося действия «Удалить выбранную дисциплину» и метку выходного действия «Сформировать список оставшихся дисциплин в банке дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Главное окно».

После изменения расположения дисциплин в банке дисциплин происходит переход в состояние «Изменения расположения в банке дисциплин». Текущее состояние содержит три метки выполняющегося действия «Выбрать дисциплину», «Переместить дисциплину вверх» и «Переместить дисциплину вниз», а также метку выходного действия «Сформировать список дисциплин в определенном порядке». Выполнив перечисленные действия, произойдет переход в состояние «Главное окно».

После изменения количества з.е. для дисциплины в банке дисциплин происходит переход в состояние «Изменение количества з.е. для дисциплины в банке дисциплин». Текущее состояние содержит несколько меток выполняющегося действия: «Выбрать дисциплину», «Изменить количество з.е. теоретического обучения», «Отметить наличие промежуточной аттестации» и «Отметить наличие курсовой работы/проекта», а также метку выходного действия «изменить количество з.е. для дисциплины». Выполнив перечисленные действия, произойдет переход в состояние «Главное окно».



**Рисунок 21 – Детализированная диаграмма состояния «Операции над дисциплинами»**

После перемещения дисциплины в семестр происходит переход в состояние «Перемещение дисциплины в семестр». Текущее состояние содержит метку выполняющегося действия «Выбрать дисциплину из банка дисциплин» и метку выходного действия «Сформировать список оставшихся дисциплин в банке дисциплин». Выполнив перечисленные действия, произойдет переход в состояние «Главное окно».

После нажатия на кнопку «Удалить дисциплину из семестра» происходит переход в состояние «Удаление дисциплины из семестра». Текущее состояние содержит метку выполняющегося действия «Удалить выбранную дисциплину» и метку выходного действия «Сформировать список оставшихся дисциплин в семестре». Выполнив перечисленные действия, произойдет переход в состояние «Главное окно».

После изменения расположения дисциплин в семестре происходит переход в состояние «Изменения расположения в семестре». Текущее состояние содержит три метки выполняющегося действия «Выбрать дисциплину», «Переместить дисциплину вверх» и «Переместить дисциплину вниз», а также метку выходного действия «Сформировать список дисциплин в определенном порядке». Выполнив перечисленные действия, произойдет переход в состояние «Главное окно».

После изменения количества з.е. для дисциплины в семестре происходит переход в состояние «Изменение количества з.е. для дисциплины в семестре». Текущее состояние содержит несколько меток выполняющегося действия: «Выбрать дисциплину», «Изменить количество з.е. теоретического обучения», «Отметить наличие промежуточной аттестации» и «Отметить наличие курсовой работы/проекта», а также метку выходного действия «изменить количество з.е. для дисциплины». Выполнив перечисленные действия, произойдет переход в состояние «Главное окно».

Исходя из разработанной диаграммы состояний, спроектируем макеты графического интерфейса программы.

# **6. Конструирование макетов графического интерфейса**

Конструирование макетов графического интерфейса будет осуществляться с помощью условно-бесплатного редактора Figma.

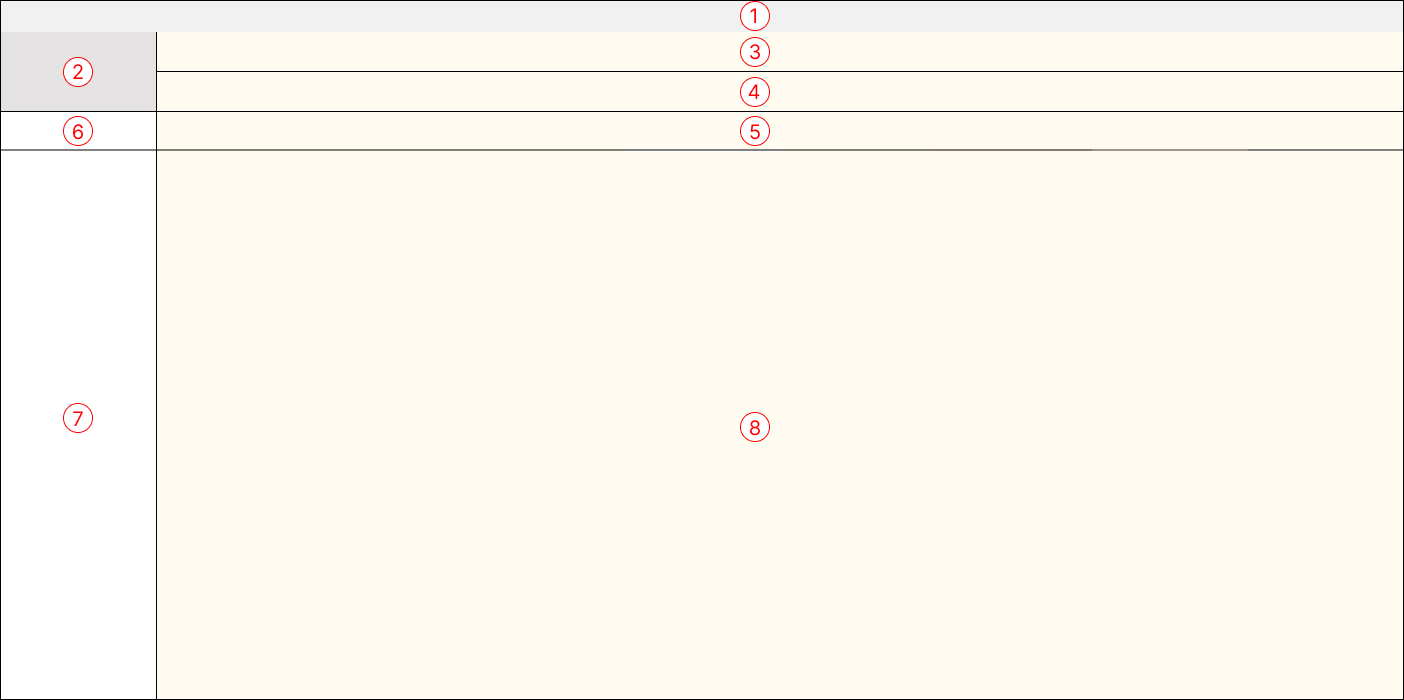
Figma — кроссплатформенный графический онлайн-редактор для совместной работы. Программа позволяет создавать графические интерфейсы, прототипы, презентации и с лёгкостью передавать материалы в разработку.

Часть графических интерфейсов были упомянуты в первой главе. Опишем их поподробнее.

Главное окно состоит из восьми основных областей:

1. блок меню:
   * + кнопка меню «Файл». По нажатию должна отображать следующие кнопки: «Создать проект», «Открыть проект», «Сохранить», «Сохранить как», «Экспорт в Excel» и выход;
     + кнопка меню «Правка». По нажатию должна отображать кнопки «Отменить» и «Вернуть»;
     + кнопка меню «Справка». По нажатию должен открыться файл с инструкцией к приложению;
     + кнопка меню «О программе». По нажатию откроется окошко с информацией о разработчике;
2. кнопка «Пополнить банк дисциплин». По нажатию должно открыться окно «Редактирования дисциплин»;
3. заголовок «Учебный план» и общее количество з.е. в УП;
4. заголовок «Номер курса» и сумма з.е. для каждого курса;
5. заголовок «Номер семестра» и сумма з.е. для каждого семестра, с декомпозированным вычислением теоретического обучения, промежуточная аттестация и курсовая работа/проект;
6. заголовок «Банк дисциплин»;
7. колонка банка дисциплин. Сюда выгружаются дисциплины, выбранные в окне «Редактор дисциплин»;
8. колонки семестров. Перемещение плитки происходит между одним из семестров и банком дисциплин;

Области макета главного окна представлены на рисунке 22.

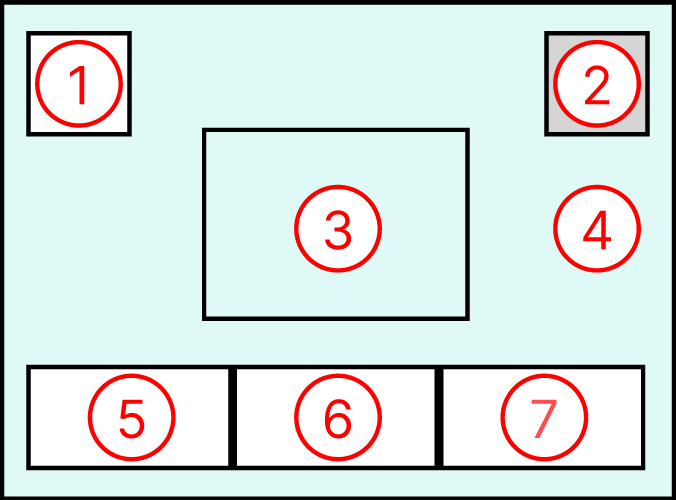


**Рисунок 22 – Области макета главного окна**

Выполнение различных операций над дисциплинами предполагает наличие отдельного графического элемента, который будет являться частью главного окна. Таким элементом будет интерфейс, который по внешнему виду напоминает плитку. Она состоит из следующих областей:

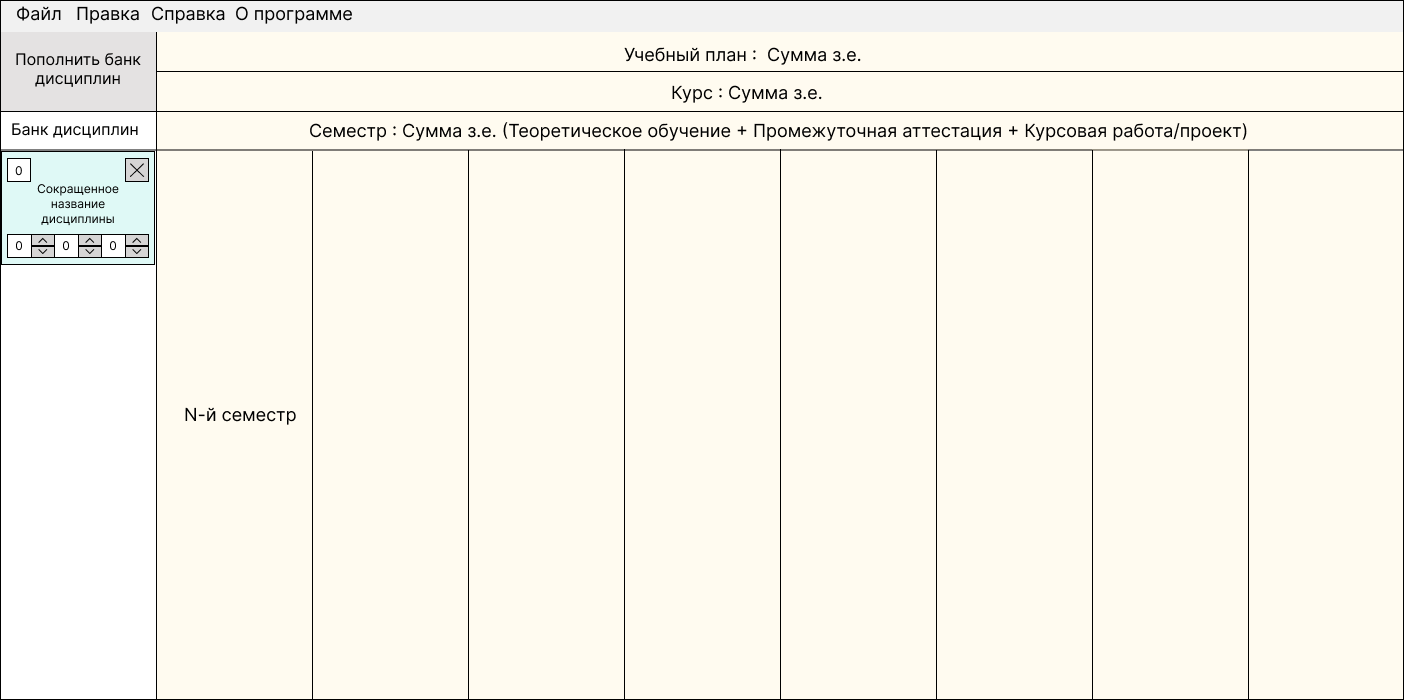
1. общее количество зачетных единиц для плитки;
2. кнопка «Удалить плитку». Если плитка находится в семестре, то по нажатию на эту кнопку, она вернется в банк дисциплин. А нажатие в банке дисциплин, просто удалит плитку из главного окна;
3. сокращенное название дисциплины;
4. цвет категории для дисциплины;
5. элемент управления значением з.е. для теоретического обучения, который выглядит как сочетание текстового поля и пары кнопок со стрелками для корректировки значения пользователем. Он выводит и задает отдельное числовое значение из списка вариантов. Пользователь может увеличить и уменьшить число путем нажатия кнопок со стрелками вверх и вниз. Нажатие клавиши со стрелкой вверх увеличивает значение до максимума; нажатие клавиши со стрелкой вниз уменьшает значение до минимума. Данный элемент управления также используется в следующих двух областях;
6. элемент управления значением з.е. промежуточной аттестации;
7. элемент управления значением з.е. курсовой работы/проекта.

Области макета плитки представлены на рисунке 23.



**Рисунок 23 – Области макета плитки**

Общий вид макета главного окна с графическим элементом плитка представлен на рисунке 24.

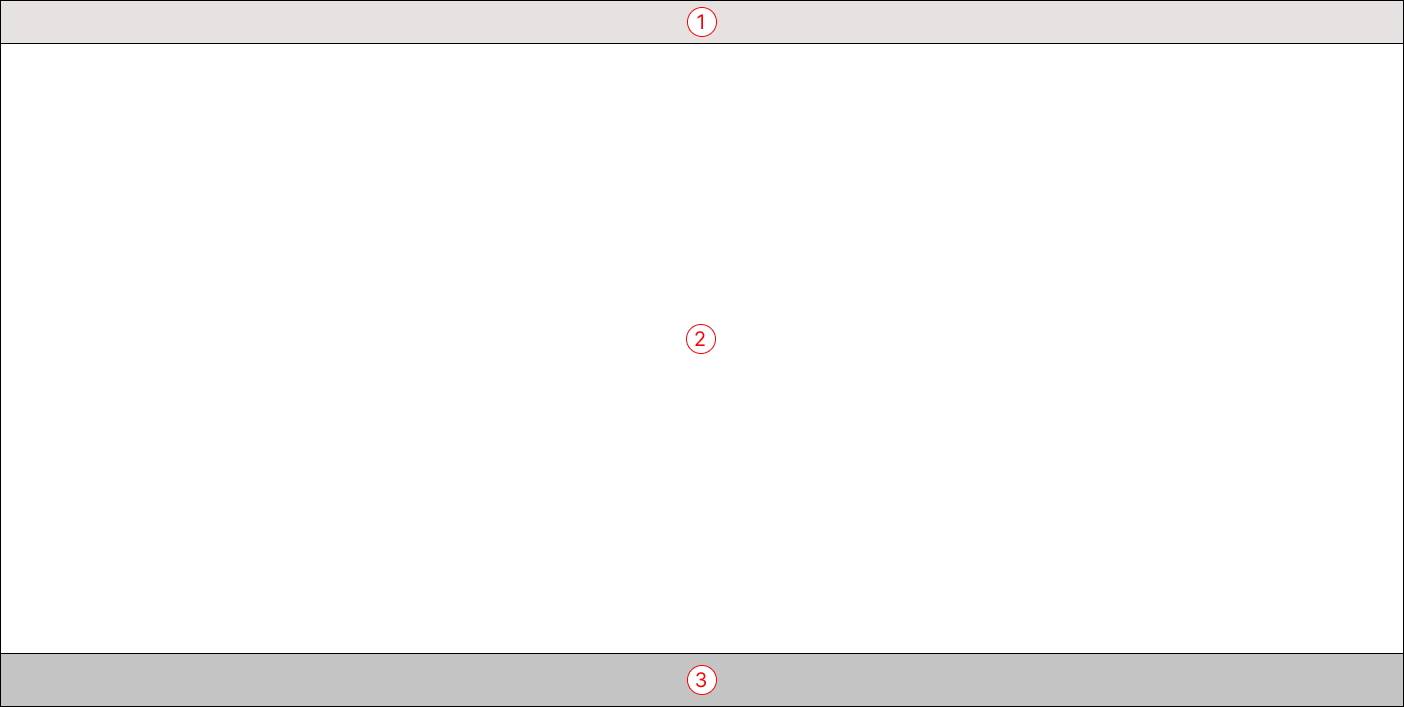


**Рисунок 24 – Макета главного окна**

Пополнения банка дисциплин будет происходить при помощи окна редактирования дисциплин, состоящих из следующих областей:

1. заголовки колонок, по нажатию на которые должна будет происходить сортировка, а также кнопок в форме стрелки, позволяющих выполнить поиск элементов;
2. строки, расположенные под каждой из колонок, необходимые для внесения данных о дисциплинах;
3. кнопки удаления, добавления и выбора дисциплин.

Области макета окна редактирования дисциплин представлены на рисунке 25.

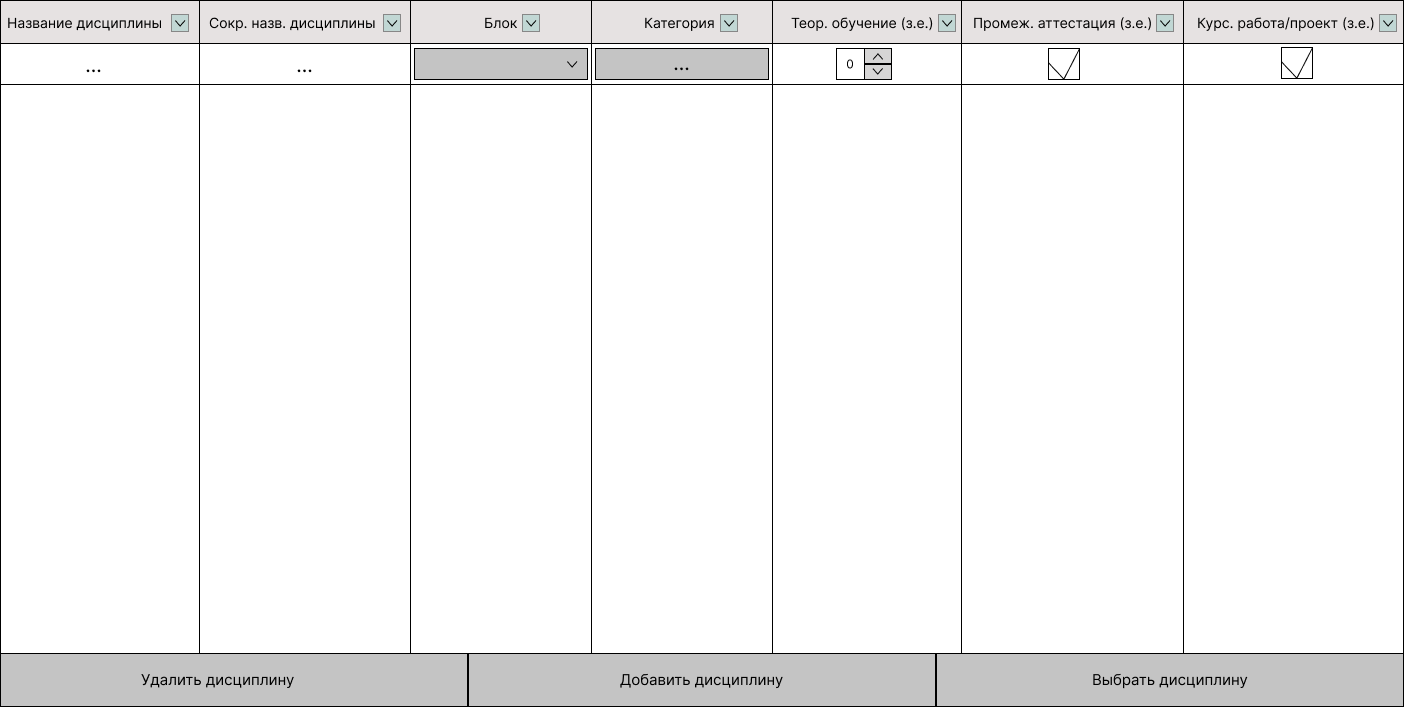


**Рисунок 25 – Области макета окна редактирования дисциплин**

Каждая из колонок представляет собой уникальное поле. Рассмотрим их поподробнее:

1. название дисциплины – изменяемое текстовое поле;
2. сокращенное название дисциплины – то же самое, что и в названии дисциплины;
3. блок – выпадающий список, к котором представлены блоки, описанные в документе ФГОС ВО впервой главе;
4. категория – кнопка, по нажатию на которую открывается окно редактирования категорий, которое будет описано ниже;
5. теоретическое обучение - элемент управления значением з.е.;
6. промежуточная аттестация – ящик с флажком. Означает наличие промежуточной аттестации, если галочка стоит и отсутствие, если не стоит;
7. курсовая работа/проект – ящик с флажком.

Общий вид макета окна редактирования дисциплин представлен на рисунке 26.

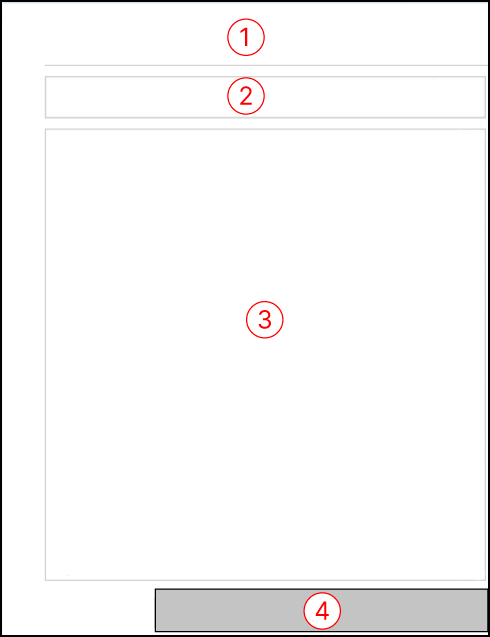


**Рисунок 26 – Макет окна редактирования дисциплин**

За поиск и выбор необходимых элементов отвечает макет поиска дисциплин. Он состоит из следующих областей:

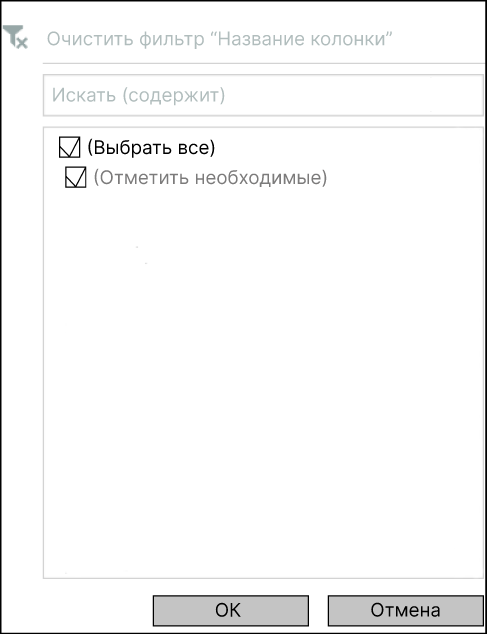
1. заголовок с названием колонки, которая подвергается поиску и фильтрации, по нажатию на который можно очистить фильтр;
2. строка поиска
3. ящик с флажком и название для выбора элемента;
4. кнопки «ОК» и «Отмена».

Области макета поиска дисциплин представлены на рисунке 27.



**Рисунок 27 – Области макета поиска дисциплин**

Общий вид макета поиска представлен на рисунке 28.

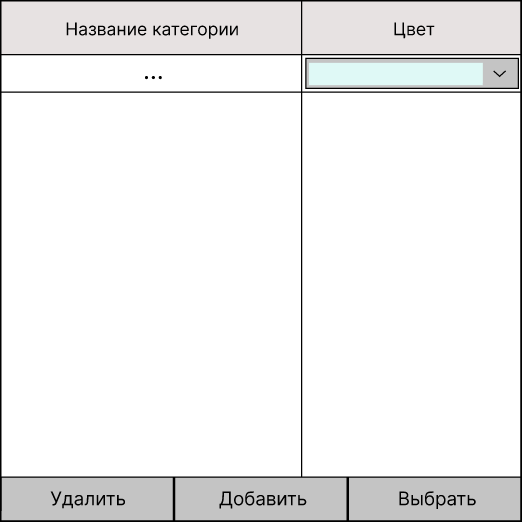


**Рисунок 28 – Макет поиска дисциплин**

Макет окна редактирования дисциплин будет вызываться по нажатию на одну из кнопок колонки категория в списке дисциплин. Внешний вид областей окна редактирования категорий идентичен внешнему виду областей окна редактирования дисциплин и состоит из следующих областей:

1. заголовки колонок;
2. строки, расположенные под каждой из колонок, необходимые для внесения данных о категориях;
3. кнопки удаления, добавления и выбора категорий.

Общий вид макета окна редактирования категорий представлен на рисунке 29.



**Рисунок 29 – Макет окна редактирования категорий**

Колонки «Название категории» и «Цвет» также, как и в окне редактирования дисциплин являются уникальными полями. Рассмотрим их поподробнее:

1. название категории – изменяемое текстовое поле;
2. блок – выпадающий список с цветами. Также можно настроить необходимый цвет воспользовавшись расширенным набором.

# **7. Построение логической и физической модели базы данных**

На основании предметной области, можно выделить следующие объекты: дисциплина и категория.

На основании выделенных объектов, можно выделить следующие сущности: дисциплина и категория.

Рассмотрим связи между данными сущностями.

Дисциплина может относиться только к одной категории, но к одной конкретной категории могут относиться множество различных дисциплин. Поэтому между ними будет идентифицирующая связь «один ко многим».

Рассмотрим сами сущности и их атрибуты.

Сущность «Дисциплина» содержит следующие атрибуты:

* код дисциплины;
* название дисциплины;
* сокращенное название дисциплины;
* блок;
* объем теоретической работы;
* промежуточная аттестация;
* курсовая работа/проект.

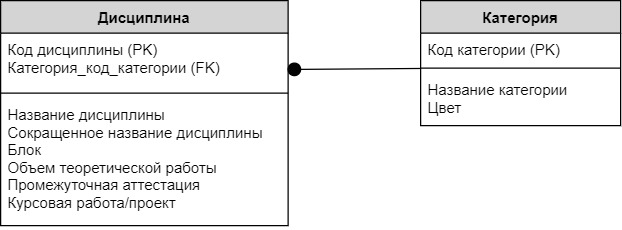
В качестве первичного ключа будет служить поле «Код дисциплины».

Сущность «Категория» содержит в себе атрибуты:

* код категории;
* название категории;
* цвет.

В качестве первичного ключа будет служить поле «Код категории».

На рисунке 30 представлена логическая модель базы данных.



**Рисунок 30 – Логическая модель базы данных**

Однако, полученная логическая схема базы данных не соответствует первым трем нормальным формам. Исходя из этого, мы не можем построить физическую схему и перейти к реализации базы данных. Рассмотрим процесс нормализации базы данных, в конце которого мы должны получить нормализованную логическую схему базы данных, которая соответствует третьей нормальной форме.

Приведем нашу схему к первой, второй и третьей нормальным формам.

Первая нормальная форма (1НФ) говорит, что каждый атрибут отношения должен хранить атомарное значение, каждое отношение (строка в таблице) должно содержать одинаковое количество атрибутов (столбцов), т.е.

* запрещает повторяющиеся столбцы (содержащие одинаковую по смыслу информацию);
* запрещает множественные столбцы (содержащие значения типа списка и т.п.);
* требует определить первичный ключ для таблицы, то есть тот столбец или комбинацию столбцов, которые однозначно определяют каждую строку.

Вторая нормальная форма (2НФ) говорит, что отношение находится во второй нормальной форме, если оно находится в 1НФ, и при этом все не ключевые атрибуты зависят только от первичного ключа, т.е.:

* вторая нормальная форма требует, чтобы не ключевые столбцы таблиц зависели от первичного ключа в целом, но не от его части;
* если таблица находится в первой нормальной форме и первичный ключ у нее состоит из одного столбца, то она автоматически находится и во второй нормальной форме.

Отношение находится в третьей нормальной форме (3НФ), если оно находится во второй нормальной форме и каждый не ключевой атрибут зависит только от первичного ключа и не зависят друг от друга.

Обозначим связи между сущностями:

«Категория» - «Дисциплина»: «один ко многим», поскольку одна категория может относиться к нескольким дисциплинам, но одна дисциплина может принадлежать только одной категории; идентифицирующая, поскольку без сущности «Категория» нельзя однозначно определить сущность «Дисциплина».

Проанализировав полученную логическую схему, можно сделать вывод, что она находится во второй нормальной форме (так как она находится в первой нормальной форме и каждый не ключевой атрибут неприводимо зависит от первичного ключа).

Продолжив изучать полученную схему, можно заметить, что она также находится и в третьей нормальной форме, поскольку она находится во второй нормальной форме и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа.

Опираясь на совокупность всех вышеупомянутых и перечисленных фактов, построим нормализованную логическую схему, которая одновременно является и физической моделью базы данных.

Для построения физической модели БД, необходимо сделать выбор и обоснование целевой СУБД. Одной из самых популярных систем управления базами данных на сегодняшний день является SQLite, распространяемая свободно. Эта серверная система способна эффективно функционировать во взаимодействии с интернет-сайтами и веб-приложениями. При этом она проста в освоении, что лишь увеличивает ее популярность.

Помимо универсальности и распространенности СУБД SQLite обладает целым комплексом важных преимуществ перед другими системами. В частности, следует отметить такие качества как:

1) Легковесность. SQLite - это очень легкая база данных, поэтому ее легко использовать в качестве встроенного программного обеспечения с такими устройствами, как телевизоры, мобильные телефоны, камеры, домашние электронные устройства и т.д.

2) Более высокая производительность. Операции чтения и записи выполняются очень быстро для базы данных SQLite. Это почти на 35% быстрее, чем файловая система. Она загружает только те данные, которые необходимы вместо того, чтобы считывать весь файл и сохранять его в памяти. Если редактируется небольшая часть скрипта, то перезаписываются только те части, которые были изменены.

3) Отсутствие необходимости настройки. SQLite очень прост в освоении. Не нужно его устанавливать и настраивать. Можно просто загрузить библиотеки SQLite на свой компьютер, и он готов к созданию базы данных.

4) Надежность. SQLite постоянно обновляет данные, поэтому в случае сбоя питания или сбоя работа практически не теряется. SQLite менее подвержен ошибкам, чем специально написанные коды ввода-вывода файлов. Запросы SQLite меньше, чем эквивалентные процедурные коды, поэтому вероятность ошибок минимальна.

5) Портативность. SQLite переносим на все 32-разрядные и 64-разрядные операционные системы, а также на большие и малые архитектуры. Несколько процессов могут быть подключены к одному и тому же файлу приложения и могут читать и записывать, не мешая друг другу. Его можно использовать со всеми языками программирования без каких-либо проблем с совместимостью.

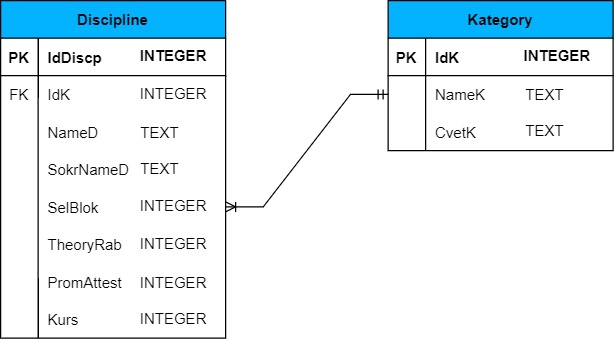
6) Доступность. База данных SQLite доступна с помощью широкого спектра сторонних инструментов. Содержимое базы данных SQLite с большей вероятностью можно восстановить, если оно было утеряно. Данные живут дольше, чем код.

7) Масштабируемость. SQLite можно легко расширить в будущих версиях, просто добавив новые таблицы и/или столбцы. Это также сохраняет обратную совместимость.

Данные в СУБД должны быть типизированными. Для этого, необходимо указать конкретные типы данных для атрибутов в таблицах. SQLite обладает всеми необходимыми для этого типами, за исключением логического. Вместо этого логические значения будут храниться в виде целых чисел 0 (false) и 1 (true). Типы данных, использующиеся в физической модели, следующие:

* INTEGER - длинное целое число. Используется для первичных ключей каждой таблицы, блока, теоретической работы, промежуточной аттестации и курсовой работы.
* TEXT - представляет собой текстовую строку, сохраненную с использованием кодировки базы данных (UTF-8, UTF-16BE или UTF-16LE).

Разработанная физическая модель БД представлена на рисунке 31.



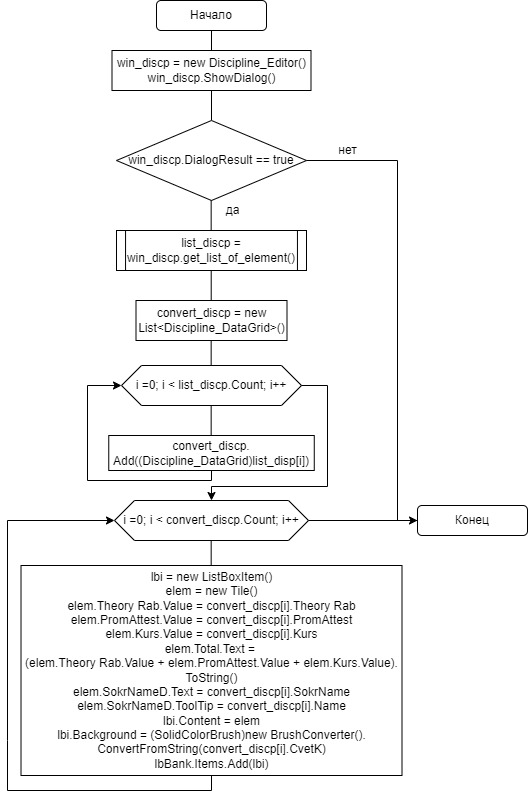
**Рисунок 31 – Физическая модель базы данных**

# **8. Разработка алгоритмов** **функционирования**

Создание алгоритмов, как и проектирование макетов можно разделить на три основных части: главное окно, окно редактирования дисциплин и окно редактирования категорий.

1. Главное окно.
   1. Пополнение банка дисциплин. Шаги:
2. Создадим экземпляр окна редактирования дисциплин (далее ОРД) и запишем его в переменную win\_discp.
3. Вызовем функцию, которая отобразит ОРД как модальное окно.
4. Если результат диалога для ОРД равен истине, то в переменную list\_discp запишем результат внешней функции get\_list\_of\_element(), которая возвращает список дисциплин в ОРД. Иначе, завершаем работу алгоритма.
5. Далее, создадим список типа таблицы ОРД и запишем его в переменную convert\_discp. Этот список необходим для преобразования данных ОРД и доступа к каждому полю таблицы.
6. В цикле добавим преобразованные данные в список.
7. Второй цикл необходим для преобразования конвертированных данных в плитку и добавления плиток в банк дисциплин.

Блок-схема алгоритма представлена на рисунке 32.

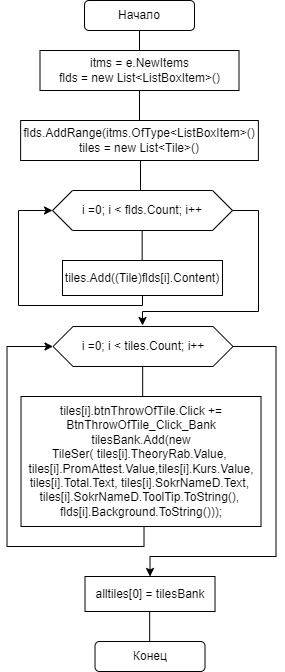


**Рисунок 32 – Блок-схема метода пополнения банка дисциплин**

* 1. Добавление плиток в коллекцию элементов банка дисциплин. Шаги:

1. Переменной itms присваиваем обновленный список элементов, связанных с изменением коллекции элементов банка дисциплин.
2. Создадим список типа ListBoxItem и запишем его в переменную flds. Данный тип является экземпляром коллекции элементов нашего банка дисциплин. Он необходим нам как обертка для нашей плитки.
3. Преобразуем элементы в тип ListBoxItem, связанные с изменением коллекции элементов.
4. В переменную tiles присвоим созданный список типа плитки.
5. В цикле обратимся к свойству Content каждого преобразованного элемента и приведем его к типу плитки.
6. Далее, для каждой кнопки удаления дисциплины, в цикле, присвоим событие удаления дисциплины из банка дисциплин и добавим все элементы в список объектов для сериализации данных: за это отвечает переменная tilesBank. Пояснение, что такое сериализация будет приведено после рассмотрения всех разработанных алгоритмов.
7. Первому элементу, являющемуся списком списка списков, переменная alltiles, отмечающему за сериализацию плиток банка данных и семестров, присвоим значение tilesBank.

Блок-схема алгоритма представлена на рисунке 33.

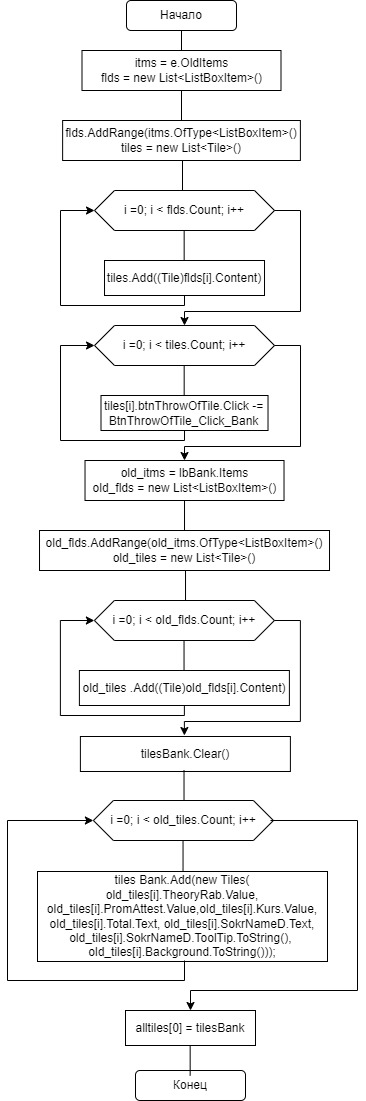


**Рисунок 33 – Блок-схема метода добавления плиток в коллекцию элементов банка дисциплин**

* 1. Удаление плиток из коллекции элементов банка дисциплин. Шаги:

1. Переменной itms присваиваем список удаленных элементов, связанных с изменением коллекции элементов банка дисциплин.
2. Создадим список типа ListBoxItem и запишем его в переменную flds.
3. Преобразуем элементы в тип ListBoxItem, связанные с изменением коллекции элементов.
4. В переменную tiles присвоим созданный список типа плитки.
5. В цикле обратимся к свойству Content каждого преобразованного элемента и приведем его к типу плитки.
6. Далее, для каждой кнопки удаления дисциплины, в цикле, присвоим событие удаления дисциплины из банка дисциплин.
7. Потом выполняются шаги 1-5 для оставшихся дисциплин.
8. Очистим список для сериализации.
9. Добавим все элементы в список объектов для сериализации данных: за это отвечает переменная tilesBank.
10. Первому элементу, являющемуся списком списка списков, переменная alltiles, отмечающему за сериализацию плиток банка данных и семестров, присвоим значение tilesBank.

Блок-схема алгоритма представлена на рисунке 34.

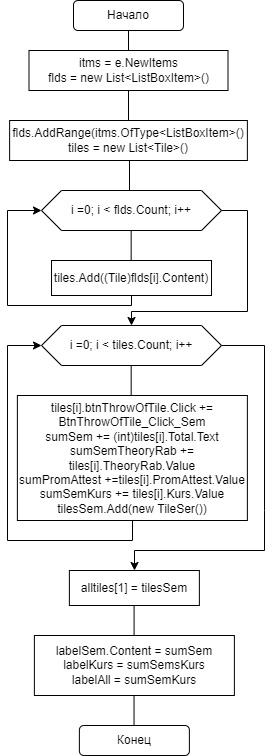


**Рисунок 34 – Блок-схема метода удаления плитки из семестра в банке дисциплин**

* 1. Добавление плиток в коллекцию элементов семестра. Шаги:

1. Переменной itms присваиваем обновленный список элементов, связанных с изменением коллекции элементов семестра.
2. Создадим список типа ListBoxItem и запишем его в переменную flds.
3. Преобразуем элементы в тип ListBoxItem, связанные с изменением коллекции элементов.
4. В переменную tiles присвоим созданный список типа плитки.
5. В цикле обратимся к свойству Content каждого преобразованного элемента и приведем его к типу плитки.
6. Далее, для каждой кнопки удаления дисциплины, в цикле, присвоим событие удаления дисциплины из семестра.
7. Увеличим значение переменных сумм семестра, которая состоит из теоретической работы, промежуточной аттестации и курсовой работы/проекта. Также добавим все элементы в список объектов для сериализации данных: за это отвечает переменная tilesSem.
8. Второму элементу, являющемуся списком списка списков, переменная alltiles, отмечающему за сериализацию плиток банка данных и семестров, присвоим значение tilesSem.
9. Текстовые метки, отвечающие за вывод з.е. с третьей по пятую области макета главного окна присвоим значения сумм.

Блок-схема алгоритма представлена на рисунке 35.

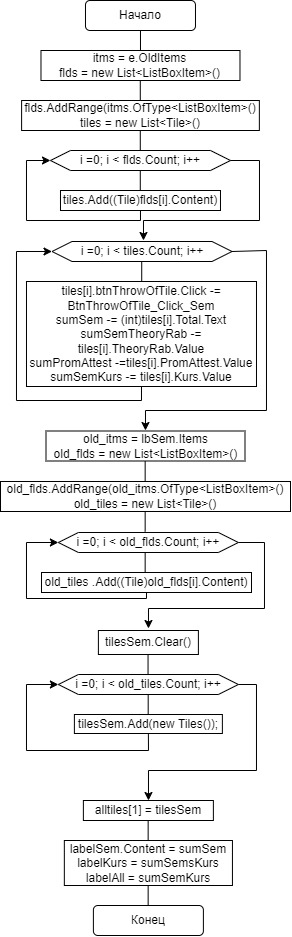


**Рисунок 35 – Блок-схема метода добавления плиток в коллекцию элементов семестра**

* 1. Удаление плиток из коллекции элементов семестра. Шаги:

1. Переменной itms присваиваем список удаленных элементов, связанных с изменением коллекции элементов семестра.
2. Создадим список типа ListBoxItem и запишем его в переменную flds.
3. Преобразуем элементы в тип ListBoxItem, связанные с изменением коллекции элементов.
4. В переменную tiles присвоим созданный список типа плитки.
5. В цикле обратимся к свойству Content каждого преобразованного элемента и приведем его к типу плитки.
6. Далее, для каждой кнопки удаления дисциплины, в цикле, присвоим событие удаления дисциплины из семестра.
7. Уменьшим значение переменных сумм семестра на количество з.е. удаленных плиток.
8. Выполним шаги 1-5 для оставшихся дисциплин.
9. Очистим список для сериализации плиток семестра.
10. Добавим все элементы в список объектов для сериализации данных: за это отвечает переменная tilesSem.
11. Второму элементу, являющемуся списком списка списков, переменная alltiles, отмечающему за сериализацию плиток банка данных и семестров, присвоим значение tilesSem.
12. Обновим значение текстовых меток, отвечающих за вывод з.е.

Блок-схема алгоритма представлена на рисунке 36.

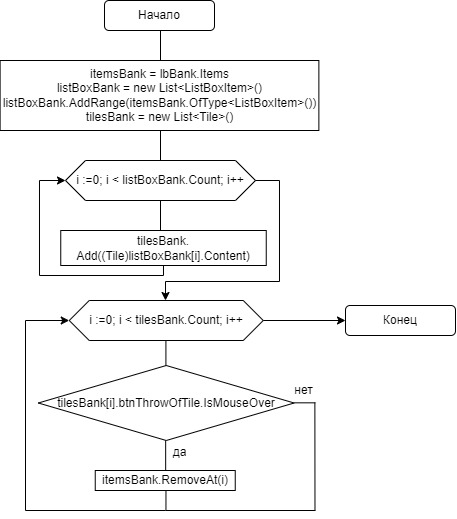


**Рисунок 36 – Блок-схема метода удаления плиток из коллекции элементов семестра**

* 1. Удаление дисциплины из банка дисциплин по нажатию кнопки на плитке. Шаги:

1. В переменную itemsBank записать список коллекции элементов банка дисциплин.
2. Создадим список типа ListBoxItem и запишем его в переменную listBoxBank.
3. Преобразуем элементы в тип ListBoxItem, связанные с изменением коллекции элементов.
4. В переменную tilesBank присвоим созданный список типа плитки.
5. В цикле обратимся к свойству Content каждого преобразованного элемента и приведем его к типу плитки.
6. Далее, для каждой кнопки удаления дисциплины, в цикле, проверим, находился ли курсор мыши над ней в момент нажатия. Если да, то удалить элемент из коллекции элементов банка дисциплин.

Блок-схема алгоритма представлена на рисунке 37.

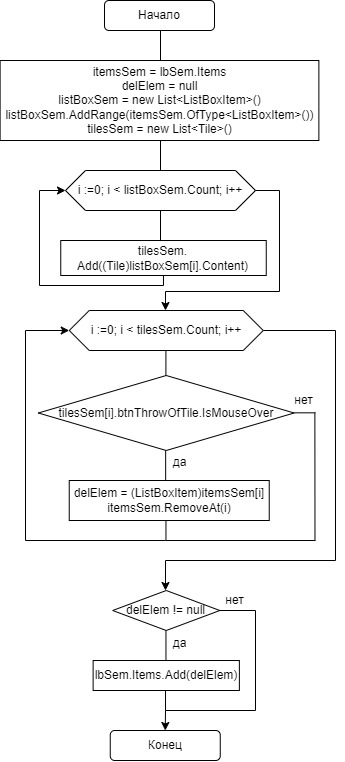


**Рисунок 37 – Блок-схема метода удаления дисциплины из банка дисциплин по нажатию кнопки на плитке**

* 1. Удаление дисциплины из семестра по нажатию кнопки на плитке. Шаги:

1. В переменную itemsBank записать список коллекции элементов банка дисциплин.
2. Объявим переменную delElem, отвечающую за добавление дисциплины в банк дисциплин после удаления из семестра и присвоим ей значение null.
3. Создадим список типа ListBoxItem и запишем его в переменную listBoxSem.
4. Преобразуем элементы в тип ListBoxItem, связанные с изменением коллекции элементов.
5. В переменную tilesSem присвоим созданный список типа плитки.
6. В цикле обратимся к свойству Content каждого преобразованного элемента и приведем его к типу плитки.
7. Далее, для каждой кнопки удаления дисциплины, в цикле, проверим, находился ли курсор мыши над ней в момент нажатия. Если да, то присвоить переменной delElem удаленную плитку и удалить элемент из коллекции элементов семестра.
8. Если переменная delElem не пуста, то добавим удаленную дисциплину в банк дисциплин.

Блок-схема алгоритма представлена на рисунке 38.



**Рисунок 38 – Блок-схема метода удаления дисциплины из семестра по нажатию кнопки на плитке**

* 1. Сохранение данных в файл. Шаги:

1. Создадим экземпляр диалогового окна сохранения файла и запишем его в переменную fl.
2. Переменной result присвоим результат отображения диалогового окна.
3. Если пользователь согласился сохранить данные, то вызываем процедуру сохранения данных в файл. В качестве параметра передается название файла.

Блок-схема алгоритма представлена на рисунке 39.

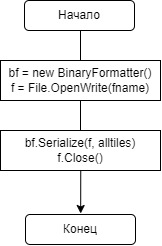


**Рисунок 39 – Блок-схема метода сохранения данных в файл**

* 1. Функция сохранения данных в файл. Принимает в качестве параметра название файла - fname. Шаги:

1. Выделяем память для объекта сериализации/десериализации данных данных.
2. Открываем файл для записи.
3. Сериализуем данные. В качестве параметров передается файл и список списков alltiles, в который входят коллекции банка дисциплин и семестров.
4. Закрываем файл для записи.

Блок-схема алгоритма представлена на рисунке 40.

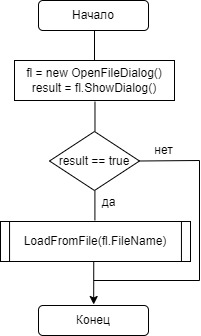


**Рисунок 40 – Блок-схема внешней функции сохранения данных в файл**

* 1. Загрузка данных из файла. Шаги:

1. Создадим экземпляр диалогового окна открытия файла и запишем его в переменную fl.
2. Переменной result присвоим результат отображения диалогового окна.
3. Если пользователь согласился открыть данные, то вызываем процедуру открытия файла. В качестве параметра передается название файла.

Блок-схема алгоритма представлена на рисунке 41.

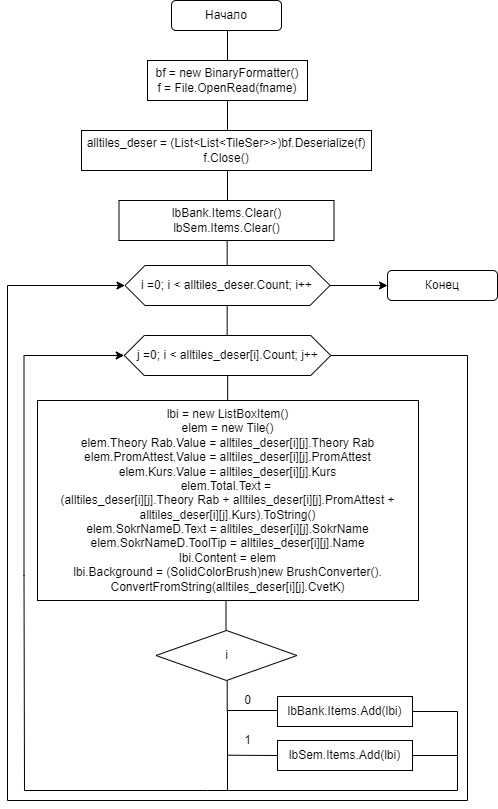


**Рисунок 41 – Блок-схема метода загрузки данных из файла**

* 1. Функция загрузки данных из файла. Шаги:

1. Выделяем память для объекта сериализации/десериализации данных данных.
2. Открываем файл.
3. Десериализуем преобразование данные в список списков alltiles\_deser.
4. Закрываем открытый файл.
5. Очищаем коллекции элементов банка дисциплин и семестров.
6. С помощью первого цикла перебираем списки alltiles\_deser
7. Далее, перебираем элементы каждого списка списков.
8. Преобразуем каждый элемент в плитку.
9. Если элемент из первого списка alltiles\_deser, тогда добавляем его в коллекцию элементов банка дисциплин.
10. Если элемент из оставшихся списков alltiles\_deser, тогда добавляем его в коллекцию элементов оставшихся семестров.

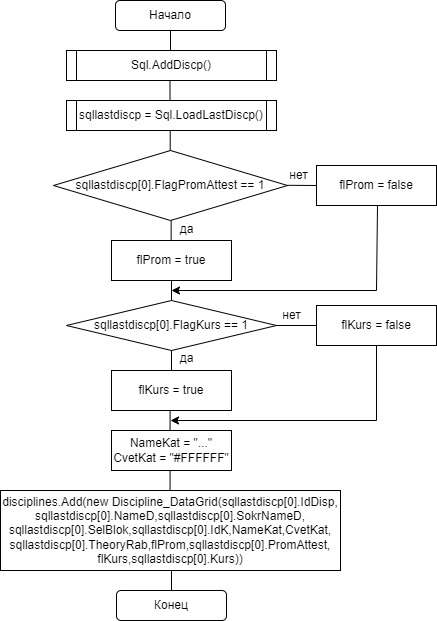
Блок-схема алгоритма представлена на рисунке 42.



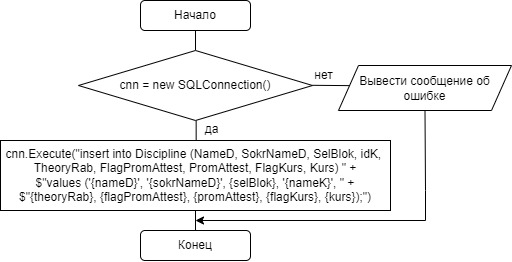
**Рисунок 42 – Блок-схема внешней функции загрузки данных из файла**

1. Редактор дисциплин.
   1. Добавление дисциплины. Шаги:
2. Вызваем функцию добавления дисциплины в базу данных таблица «Discipline». Блок-схема и сам запрос к базе данных представлен на рисунке 44.
3. Переменной sqllastdiscp присваиваем результат выполнения функции с запросом на последний элемент в таблице дисциплин (см. рис. 45).
4. Далее, двумя ветвлениями преобразуем значения промежуточной аттестации и курсовой работы/проекта в логический тип данных.
5. Присвоим значения по умолчанию переменным, отвечающим за название и цвет категории.
6. Добавим дисциплину в список дисциплин.

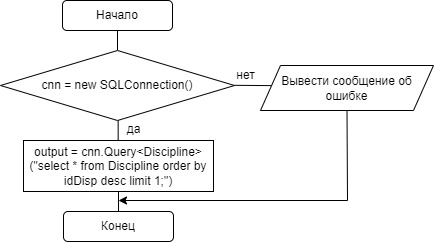
Блок-схема алгоритма представлена на рисунке 43.



**Рисунок 43 – Блок-схема метода добавления дисциплины в список дисциплин**



**Рисунок 44 – Блок-схема внешней функции ввода данных в таблицу «Discipline»**

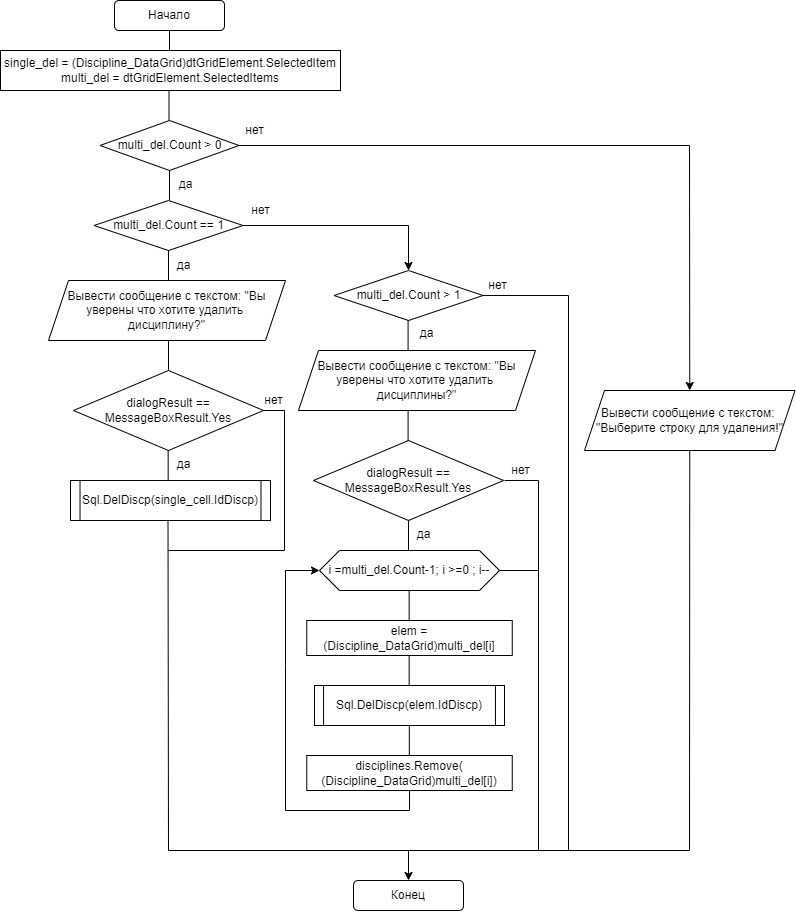


**Рисунок 45 – Блок-схема внешней функции для получения последнего элемента в таблице «Discipline»**

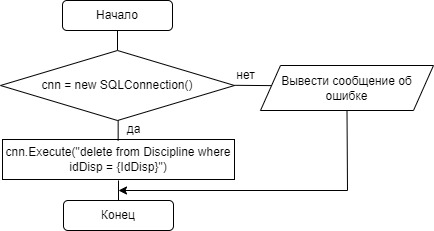
* 1. Удаление дисциплины. Шаги:

1. В переменную single\_del запишем один выделенный элемент в списке дисциплин, а переменной multi\_del присвоим несколько выделенных элементов.
2. Если количество выделенных дисциплин больше нуля перейти к следующему ветвлению, иначе, выводим сообщение с текстом: «Выберите строку для удаления!».
3. Если равно единице, тогда выводим предупредительное сообщение с текстом: «Вы уверены, что хотите удалить дисциплину?» и переходим к следующему ветвлению.
4. Если пользователь нажимает на кнопку «Да», вызываем внешнюю функцию, которая выполняет удаление дисциплины из базы данных (см. рис. 47).
5. Иначе, если количество выделенных сообщений больше одного, выводим предупреждающее сообщение: «Вы действительно ходите удалить выбранные дисциплины?» и переходим к следующему ветвлению.
6. Если пользователь нажимает кнопку «Да», перебираем дисциплины и для каждой выполняем запрос на удаление из базы данных. Также, выполняем удаление из списка дисциплин.

Блок-схема алгоритма представлена на рисунке 46.



**Рисунок 46 – Блок-схема метода удаления дисциплины из списка дисциплин**



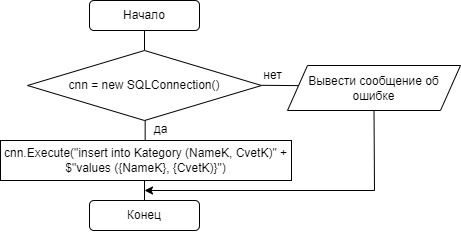
**Рисунок 47 – Блок-схема внешняя функция удаления данных из таблицы «Discipline»**

1. Редактор категорий.
   1. Добавление категории. Шаги:
2. Вызываем функцию добавления категории в базу данных таблица «Kategory». Блок-схема и сам запрос к базе данных представлен на рисунке 49.
3. Переменной sqllastkat присваиваем результат выполнения функции с запросом на последний элемент в таблице категорий (см. рис. 50).
4. Далее, двумя ветвлениями преобразуем значения промежуточной аттестации и курсовой работы/проекта в логический тип данных.
5. Добавим категорию в список категорий.

Блок-схема алгоритма представлена на рисунке 48.



**Рисунок 48 – Блок-схема метода добавления категории в список категорий**



**Рисунок 49 – Блок-схема внешней функции ввода данных в таблицу «Kategory»**

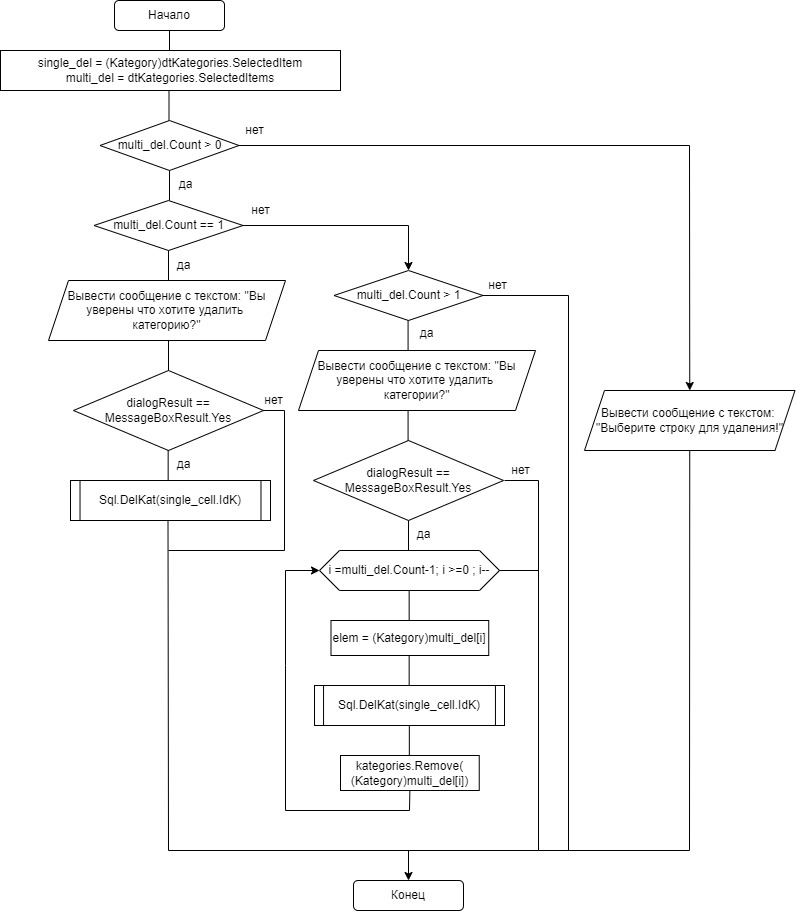


**Рисунок 50 – Блок-схема внешней функции для получения последнего элемента в таблице «Kategory»**

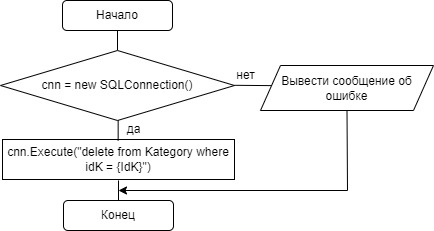
* 1. Удаление категории. Шаги:

1. В переменную single\_del запишем один выделенный элемент в списке категорий, а переменной multi\_del присвоим несколько выделенных элементов.
2. Если количество выделенных категорий больше нуля переходим к следующему ветвлению, иначе, выводим сообщение с текстом: «Выберите строку для удаления!».
3. Если равно единице, тогда выводим предупредительное сообщение с текстом: «Вы уверены, что хотите удалить категорию?» и переходим к следующему ветвлению.
4. Если пользователь нажимает на кнопку «Да», вызываем внешнюю функцию, которая выполняет удаление категории из базы данных (см. рис. 52).
5. Иначе, если количество выделенных сообщений больше одного, выводим предупреждающее сообщение: «Вы действительно ходите удалить выбранные категории?» и переходим к следующему ветвлению.
6. Если пользователь нажимает кнопку «Да», перебрать категории и для каждой выполняем запрос на удаление из базы данных. Также, выполняем удаление из списка категорий.

Блок-схема алгоритма представлена на рисунке 51.



**Рисунок 51 – Блок-схема метода удаления категории из списка категорий**



**Рисунок 52 – Блок-схема внешняя функция удаления данных из таблицы «Kategory»**

Пояснение, что такое сериализация и как будет осуществляться работа с базой данных.

Сериализация представляет собой процесс сохранения состояния объекта в среду хранения. Во время этого процесса открытые и закрытые поля объекта и имя класса, включая сборку с классом, преобразуются в поток байтов, который затем записывается в поток данных.

В разрабатываемом проектом будет использоваться бинарная сериализация.

Бинарная сериализация — это процесс преобразования какого-либо объекта в поток байтов. После преобразования можно этот поток байтов или записать на диск или сохранить его временно в памяти. А при необходимости можно выполнить обратный процесс — десериализацию, то есть получить из потока байтов ранее сохраненный объект.

Подключение к базе данных будет осуществляться с помощью сторонней библиотеки, или же бесплатного расширения.

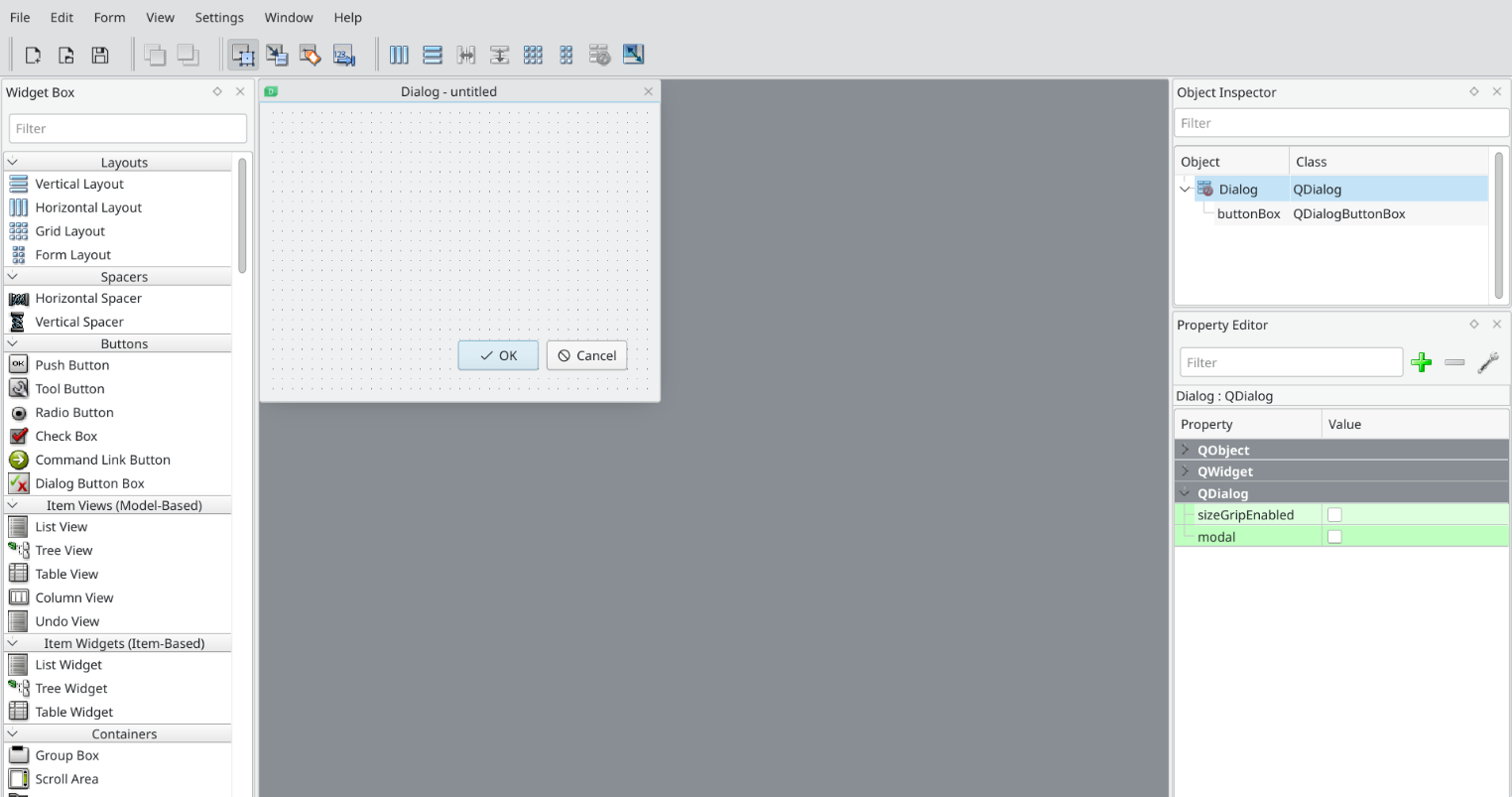
# **9. Выбор инструментальных средств разработки**

Основными рассматриваемыми инструментальными средствами разработки приложения с графическим интерфейсом будут технологии компаний Qt и Microsoft, а именно Qt Creator+Qt Desigret+QML и Windows Presentation Foundation. Рассмотрим данные продукты поподробнее.

* + - 1. Qt - это набор инструментов виджетов для создания графических пользовательских интерфейсов, а также кроссплатформенных приложений, которые работают на различных программных и аппаратных платформах, таких как Linux, Windows, macOS, Android или встроенные системы, практически без изменений в базовой кодовой базе, оставаясь при этом нативным приложением с собственными возможностями и скоростью.

Основным языком программирования является С++. Разметка графического интерфейса осуществляется с помощью декларативного языка Qt Modeling Language (далее QML) программирования, в основании которого лежит среда JavaScript. QML используется для разработки приложений, делающих основной упор на пользовательский интерфейс и, в целом, на дизайн графической части.

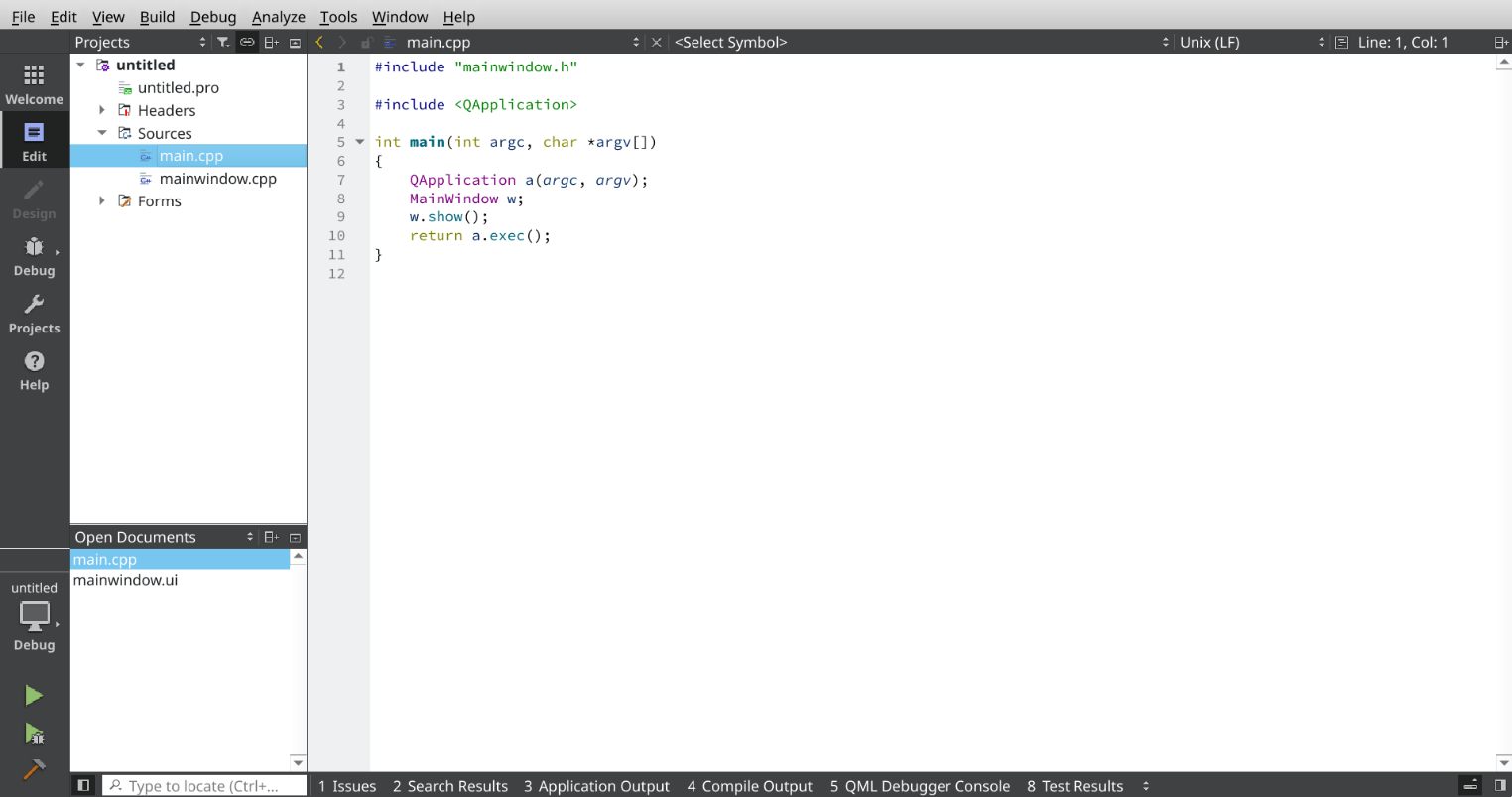
Инструмент, с помощью происходит проектирование пользовательского интерфейса называется Qt Designer (см. рис. 53).



**Рисунок 53 – Среда разработки графического интерфейса Qt Designer**

Основными элементами в Qt Designer являются виджеты и формы. Виджеты и формы, созданные легко интегрируются с запрограммированным кодом, так что вы можно легко назначать поведение графическим элементам. Все свойства, заданные в Qt Designer, могут быть динамически изменены в коде. Кроме того, такие функции, как продвижение виджетов и пользовательские плагины, позволяют вам использовать свои собственные компоненты с Qt Designer.

Описание обработчиков событий виджетов и форм осуществляется в среде разработки Qt Creator (см. рис. 54).



**Рисунок 54 – Среда разработки Qt Creator**

Qt Creator - это кроссплатформенная интегрированная среда разработки, созданная для максимального удобства разработчиков. Qt Creator работает в настольных операционных системах Windows, Linux и Mac OS и позволяет разработчикам создавать программное обеспечение на настольных, мобильных и встраиваемых платформах.

C моей точки зрения преимуществами Qt является:

* возможность создавать приложения с графическим интерфейсом на разных операционных системах;
* использование языка C++, который славится своей скоростью работы и высокой производительностью;
* наличие мощных сред разработки.

Недостатками будут отсутствие документации на русском языке и сложность в освоении языка С++.

* + - 1. Windows Presentation Foundation (далее WPF) — это фреймворк, работающий на платформе .NET, с помощью которого можно создать настольные клиентские приложения. Платформа разработки WPF поддерживает широкий спектр элементов графического интерфейса, таких как метки, текстовые поля, таблицы, кнопки и другие хорошо известные элементы.

Основными языками при разработке на платформе WPF программирования являются C# и Extensible Application Markup Language (далее XAML).

XAML — это декларативный язык, который может инициализировать объекты и задавать свойства объектов, используя языковую структуру, которая показывает иерархические отношения между несколькими объектами, и соглашение о типах поддержки, поддерживающее расширение типов. Можно создавать видимые элементы пользовательского интерфейса в декларативной разметке XAML, а также связывать отдельный кодовый файл для каждого файла XAML, который может реагировать на события и управлять объектами, которые первоначально были объявлены в XAML.

Язык XAML поддерживает обмен исходными данными между различными инструментами и ролями в процессе разработки, например, обмен исходными данными XAML между инструментами проектирования и интерактивной средой разработки или между основными разработчиками и разработчиками локализации. Используя XAML в качестве формата обмена, роли дизайнера и разработчика могут быть разделены или объединены, а дизайнеры и разработчики могут выполнять итерации во время создания приложения.

По своему синтаксису XAML очень напоминает теги стандартизированного языка разметки документа HyperText Markup Language (далее HTML), который сам по себе очень легок в освоении. Создатели данного языка как раз таки вдохновлялись технологией HTML, чтобы сделать что-то подобное, но только для разметки интерфейса приложений рабочего стола. У них это отлично получилось.

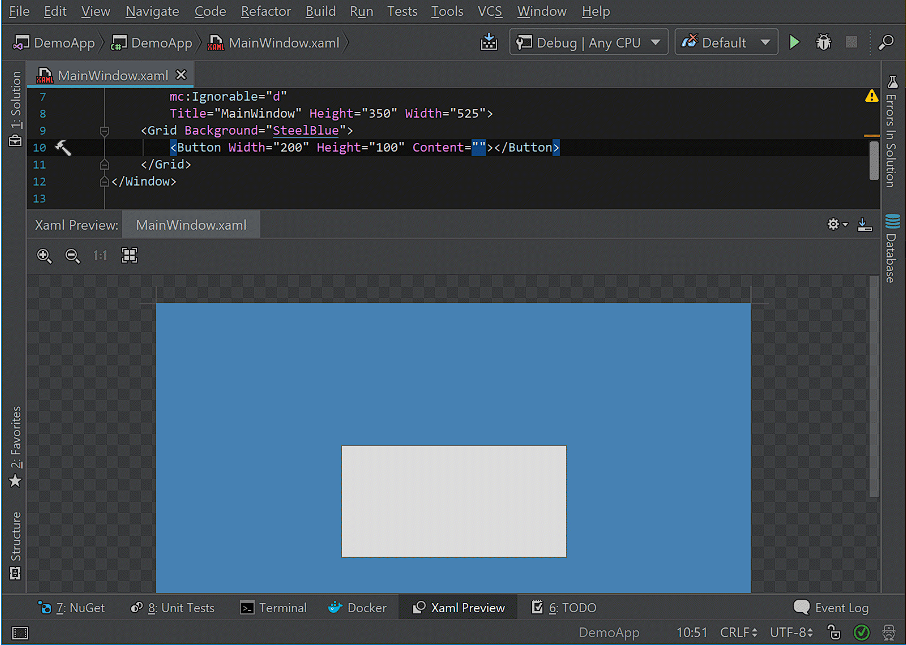
За поведение приложения и обработку событий отвечает C#.

C# - это современный объектно-ориентированный язык программирования общего назначения, разработанный корпорацией Microsoft и одобренный Европейской ассоциацией производителей компьютеров и Международной организацией по стандартизации.

C# предназначен для инфраструктуры общего языка, которая состоит из исполняемого кода и среды выполнения, что позволяет использовать различные языки высокого уровня на разных компьютерных платформах и архитектурах.

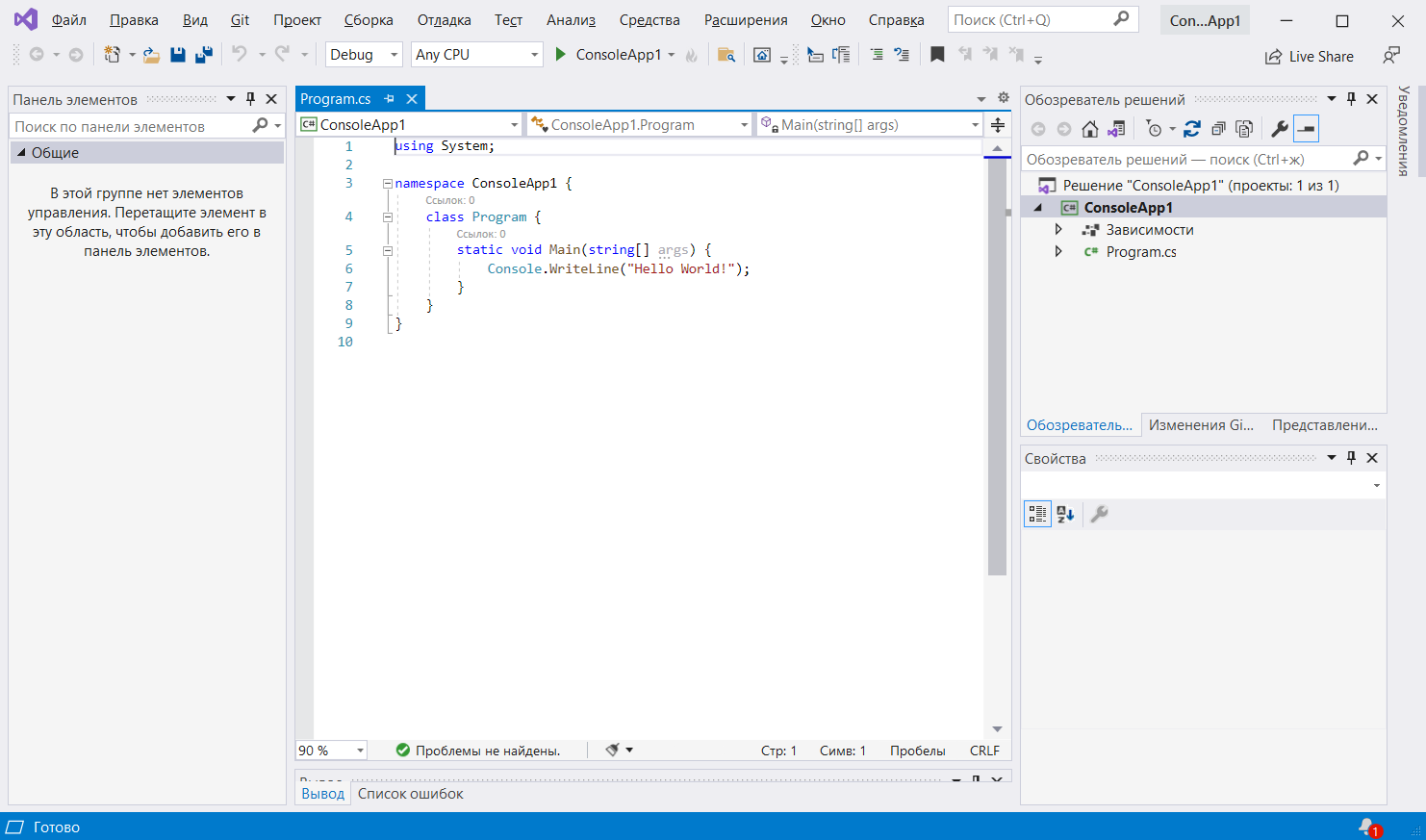
Основными средами, которых возможна разработка приложений на платформе WPF являются Rider от компании JetBrains и Microsoft Visual Studio.

Rider — это среда .NET-разработки, которой можно пользоваться на Windows, Mac и Linux. Rider подходит для многих видов .NET-приложений. Интерфейс программы представлен на рисунке 55.



**Рисунок 55 – Среда разработки Rider**

Visual Studio – это среда разработки, созданная корпорацией Microsoft и используемая для различных типов разработки программного обеспечения, таких как компьютерные программы, веб-сайты, веб-приложения, веб-службы и мобильные приложения (см. рис. 56). Она содержит инструменты завершения, компиляторы и другие функции, облегчающие процесс разработки программного обеспечения.



**Рисунок 56 – Среда разработки Microsoft Visual Studio**

По моему мнению, преимуществами WPF являются:

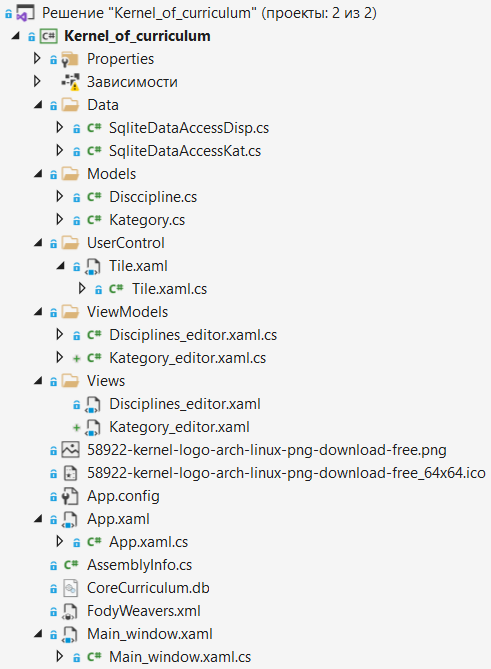
* синтаксис языка C# меньше нагружен разными конструкциями, что делает порог разработки на нем ниже, по сравнению с C++;
* декларативный язык XAML, потому что синтаксис похож на HTML, с которым уже был опыт разработки;
* наличие пакетного менеджера NuGet, с которым поиск и подключение различных библиотек гораздо проще, чем в продуктах Qt.

Недостатком является отсутствие кроссплатформенности.

В качестве основной платформы будет использоваться WPF со средой разработки Microsoft Visual Studio. Фреймворк WPF является технологией Microsoft и поддержка самого разного функционала доступна сразу, без установки дополнительных расширений, как это необходимо делать в Rider.

# **10. Описание реализации**

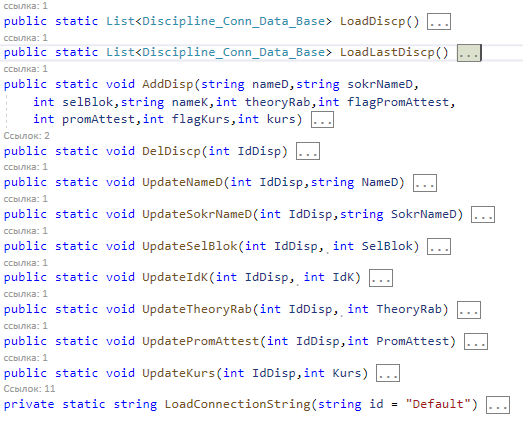
Отталкиваясь от спроектированной архитектуры, создадим файловую структуру проекта (см. рис. 57). Ее описание будет представлено ниже.



**Рисунок 57 – Структура проекта**

В папку Data добавим два файла, отвечающих за работу с базой данных: SqliteDataAccessDisp.cs и SqliteDataAccessKat.cs.

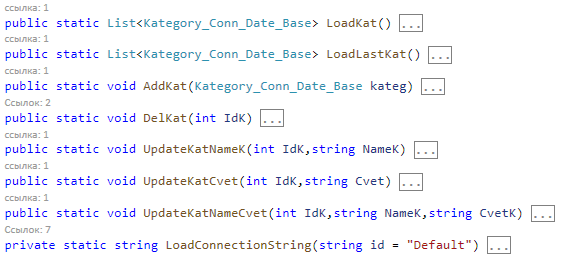
SqliteDataAccessDisp.cs содержит 12 методов, которые выполняют запросы к таблице «Discipline» (см. рис 58). Рассмотрим их поподробнее.



**Рисунок 58 – Методы файла SqliteDataAccessDisp.cs**

1. LoadDiscp необходим для загрузки дисциплин из таблицы «Discipline» в список дисциплин ОРД.
2. LoadLastDiscp – добавление последней добавленной дисциплины в список ОРД.
3. DelDiscp – удаление выбранной дисциплины из базы данных.
4. Шесть методов, начинающихся со слова Update, выполняют запрос на обновление данных в таблице.
5. LoadConnectionString – с помощью него выполняется подключение к базе данных.

SqliteDataAccessKat.cs содержит 8 методов, которые выполняют запросы к таблице «Kategory» (см. рис. 59). Методы очень схожи с методами в файле SqliteDataAccessDisp.cs.



**Рисунок 59 – Методы файла SqliteDataAccessKat.cs**

1) LoadKat необходим для загрузки категорий из таблицы «Kategory» в список категорий ОРК.

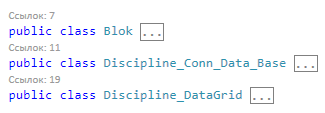
2) LoadLastKat – добавление последней добавленной категории в список ОРК.

3) DelKat – удаление выбранной категории из базы данных.

4) Три метода, начинающихся со слова Update, выполняют запрос на обновление данных в таблице.

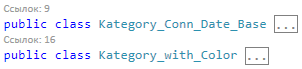
5) LoadConnectionString. Также как и в SqliteDataAccessDisp.cs отвечает за подключение к базе данных.

Файлы, находящиеся в папке Models, будут отвечать за логику работы с данными. Для ОРД это файл Discipline.cs. Он содержит три класса: Discipline\_Conn\_Data\_Base, отвечающий за загрузку данных из базы данных и Discipline\_DataGrid, который необходим для привязки данных к представлению ОРД (см. рис. 60) и класс блока дисциплины. Discipline\_DataGrid наследует интерфейс INotifyPropertyChanged, уведомляющий об изменении значения свойств графических элементов списка ОРД.



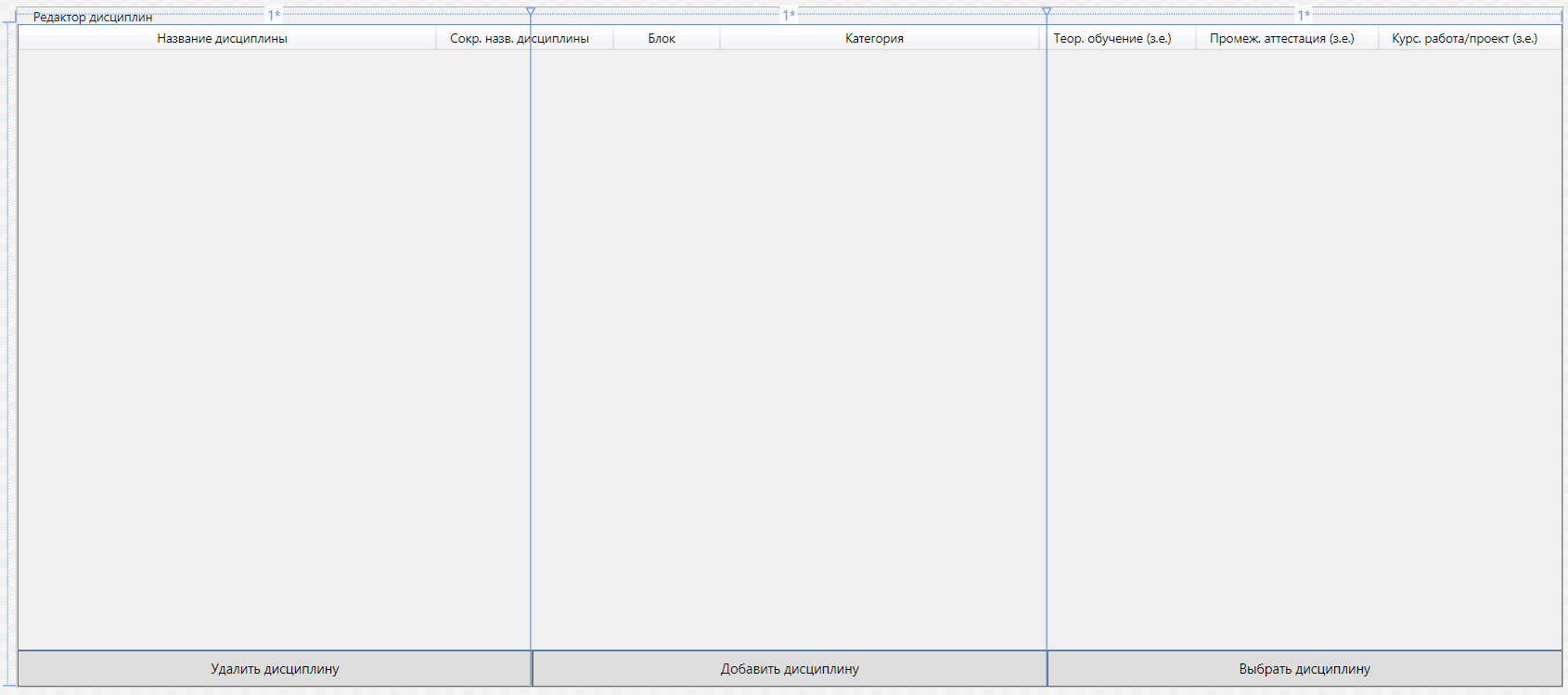
**Рисунок 60 – Классы файла Discipline.cs**

Подобная структура файла реализуется и для ОРК в файле Kategory (см. рис. 61).

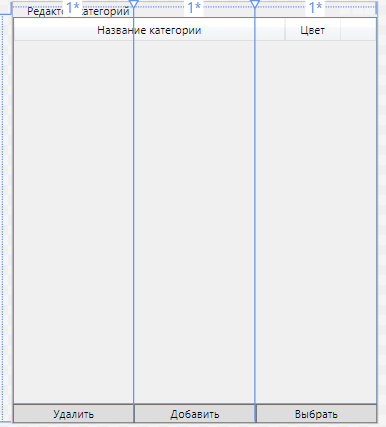


**Рисунок 61 – Классы файла Kategory.cs**

Графические представления ОРД и ОРК находится в папке Views. С помощью декларативного языка XAML и панели элементов управления WPF создадим разметку графических интерфейсов ОРД и ОРК. Спроектированные окна представлены на рисунках 62 и 63.

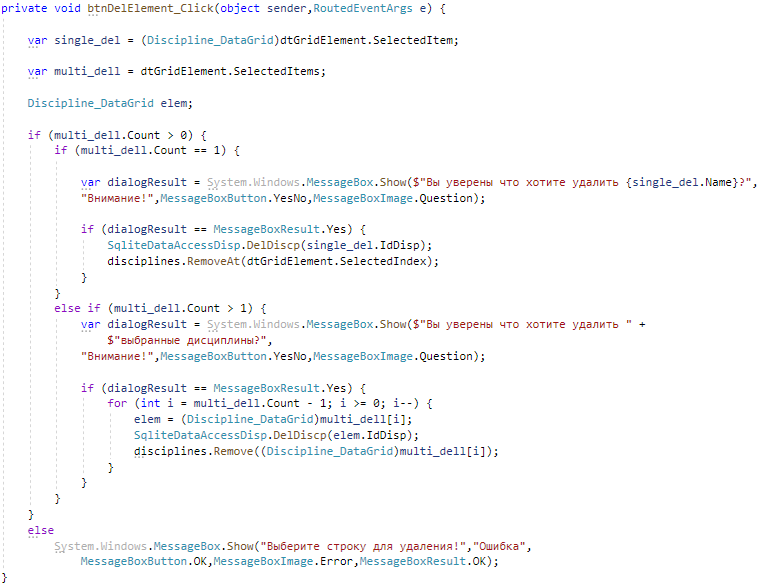


**Рисунок 62 – Окно редактирования дисциплин**



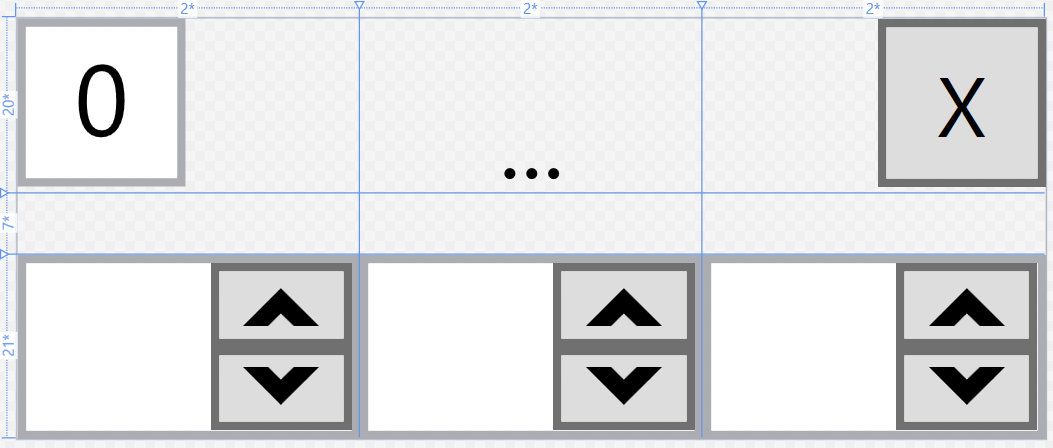
**Рисунок 63 – Окно редактирования категорий**

Реализация основных алгоритмов, поведение приложения, взаимодействие представлений с моделями и привязка данных к представлению находится в папке ViewModels. Пример реализации одного из алгоритмов, а именно удаление дисциплины из списка дисциплин и базы данных представлен на рисунке 64.



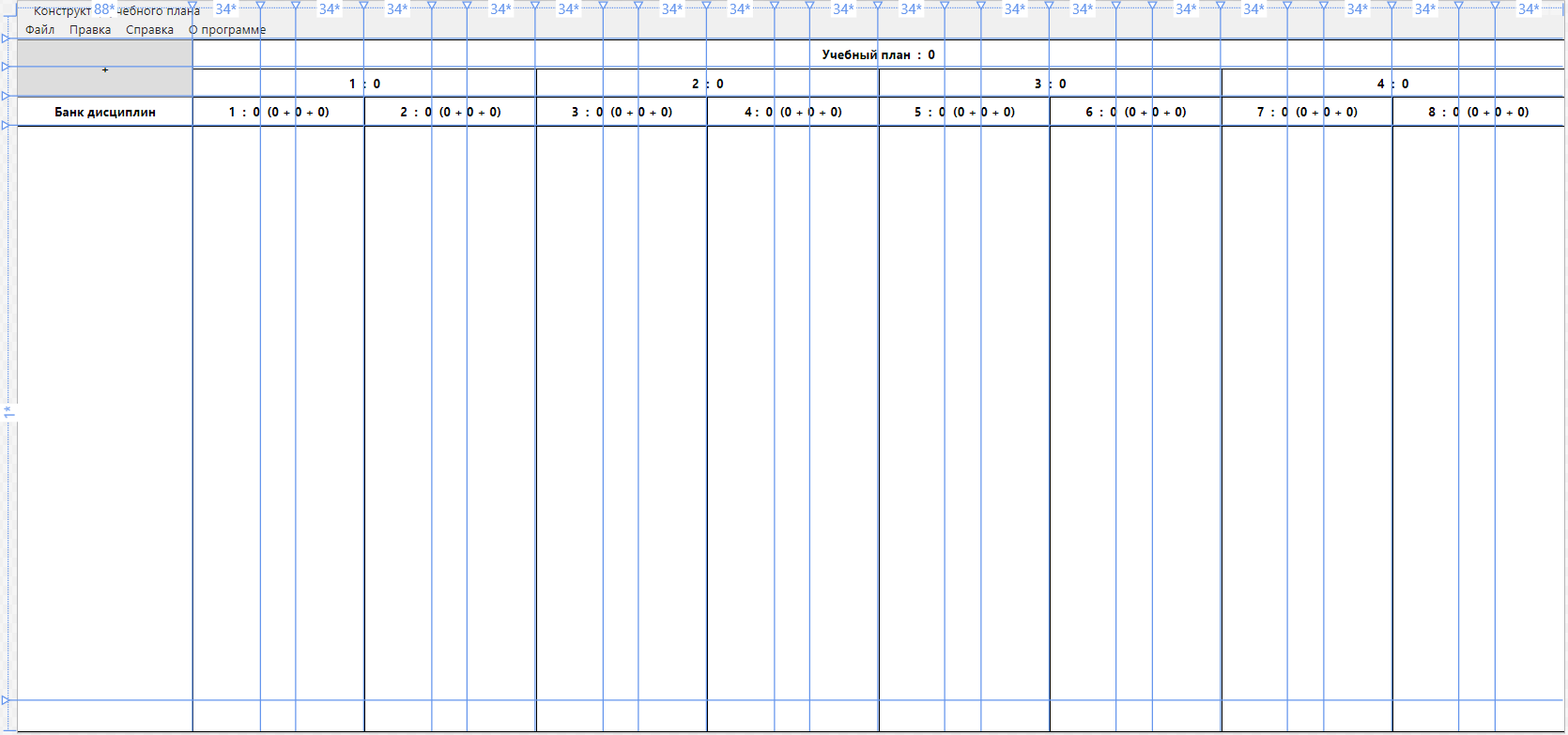
**Рисунок 64 – Реализация алгоритма удаления дисциплины из списка дисциплин и базы данных**

Папка UserControl будет содержать представление (см. рис. 65) и модель элемента управления плитка, который будет использоваться в главном окне.



**Рисунок 65 – Графическое представление плитки как элемента управления**

Главное окно представлено двумя файлами: Main\_window.xaml – внешний вид УП (см. рис. 66) и Main\_window.xaml.cs – реализация основных алгоритмов взаимодействия между банком дисциплин и семестрами, а также пополнение банка дисциплин (см. рис. 67). Переменные сумм, плитки для сериализации, кнопки закрыть и операции изменения коллекций семестра и банка заключены в region, который позволяет группировать отдельные участки кода.

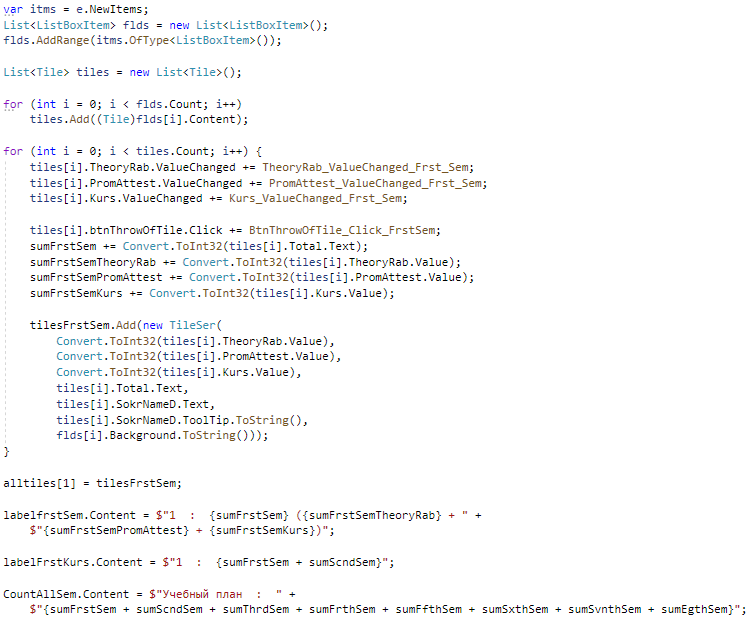


**Рисунок 66 – Графическое представление главного окна**



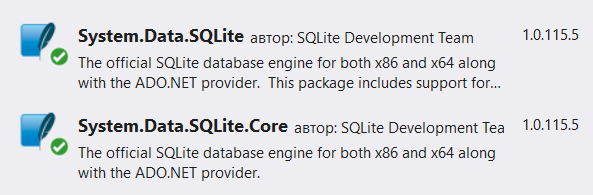
**Рисунок 67 – Методы и классы главного окна**

Пример реализации добавления дисциплины в первый семестр представлен на рисунке 68.



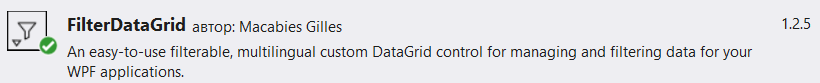
**Рисунок 68 – Методы и классы главного окна**

Подключение к базе данных SQLite будет осуществляться с помощью специальных библиотек System.Data.SQLite и System.Data.SQLite.Core, которые установим с помощью встроенного пакетного менеджера NuGet (см. рис. 69). Саму базу данных будем хранить в файле CoreCurriculum.db.



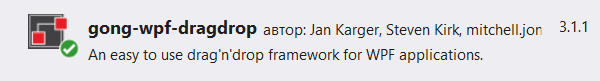
**Рисунок 69 – Библиотеки для работы с базой данных SQLite**

За окно поиска и сортировки данных в ОРД будет отвечать библиотека FilterDataGrid, в которой содержится специальный редактор списка (см. рис. 70).



**Рисунок 70 – Библиотека редактор списка дисциплин**

Возможность перемещения плиток между семестром и банком дисциплин в главном окне осуществляется с помощью библиотеки gong-wpf-dragdrop.

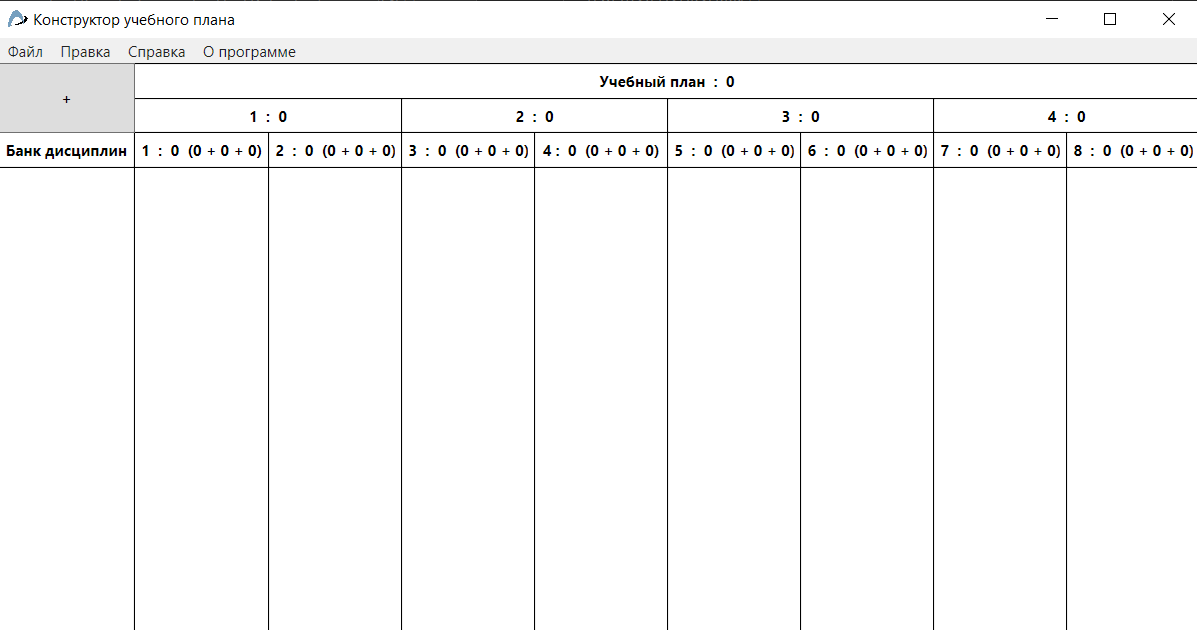


**Рисунок 70 – Библиотека перемещения плиток**

Подробнее с кодом приложения можно будет ознакомиться в приложении А.

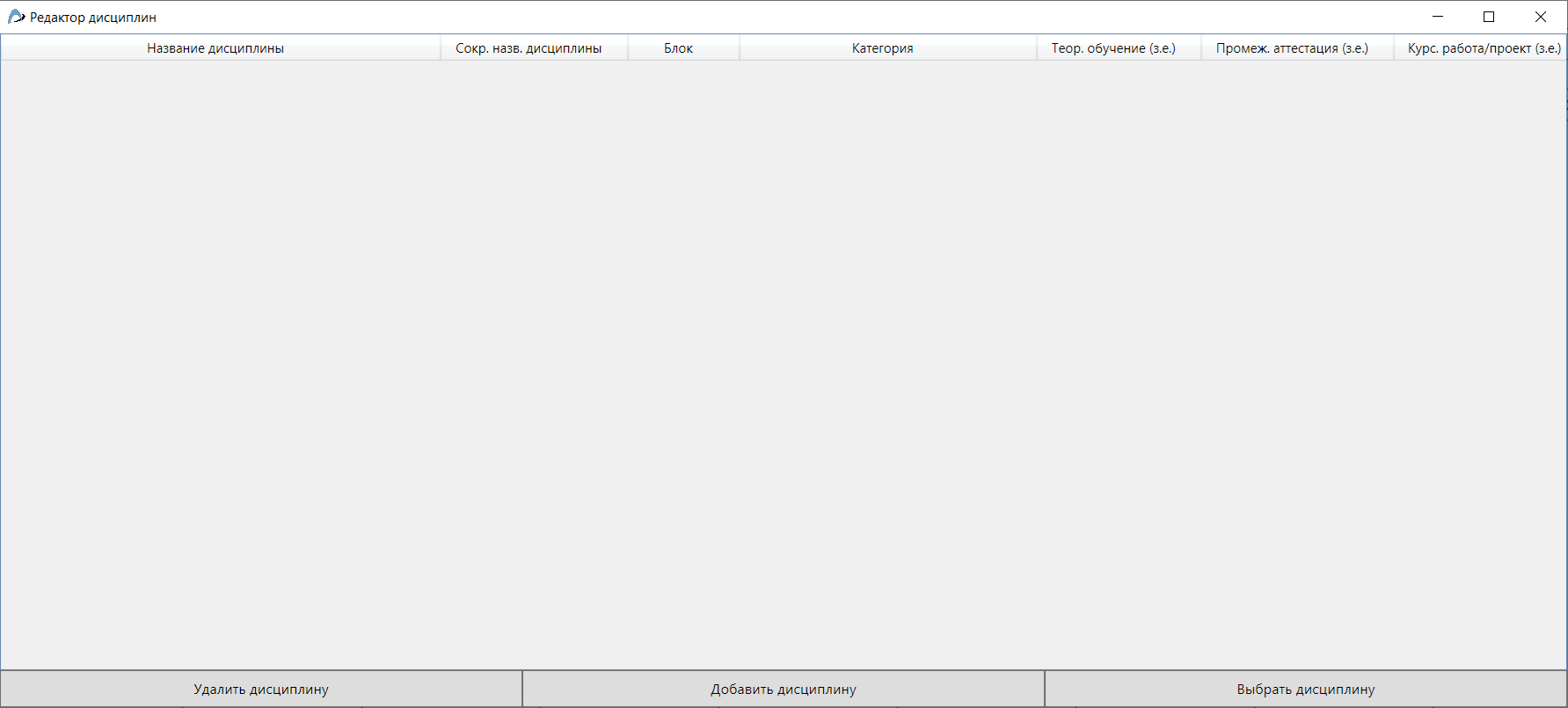
# **11. Тестирование приложения**

При открытии приложения открывается главное окно (см. рис. 71).



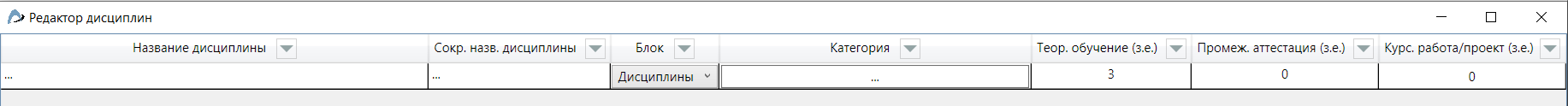
**Рисунок 71 – Главное окно**

Нажав на кнопку «Пополнить банк дисциплин», она представлена в виде плюса, происходит открытие окна редактирования дисциплин (см. рис. 72)



**Рисунок 72 – Окно редактирования дисциплин**

Для добавления дисциплины в список дисциплин с параметрами по умолчанию необходимо нажать кнопку «Добавить дисциплину» (см. рис. 73)



**Рисунок 73 – Добавление дисциплины**

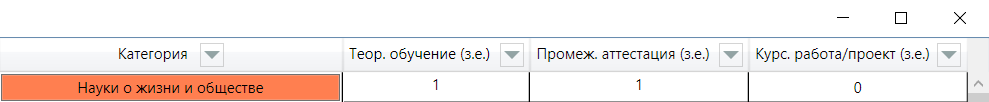
После можно выполнить редактирование полей дисциплины. Заменим поля следующими значениями:

1. Название дисциплины: Физическая культура и спорт
2. Сокращенное название дисциплины: ФКС
3. Блок: Дисциплины
4. Категория: откроем окно редактирования категорий, добавим категорию Науки о жизни и обществе, установим для нее цвет и выберем для дисциплины.
5. Теоретическое обучение: установим 1 з.е.
6. Отметим наличие промежуточной аттестации.

Результат редактирования представлен на рисунках 74 и 75.



**Рисунок 74 – Редактирование дисциплины**



**Рисунок 75 – Редактирование дисциплины**

Окно редактирования категорий с необходимыми категориями представлено на рисунке 76.



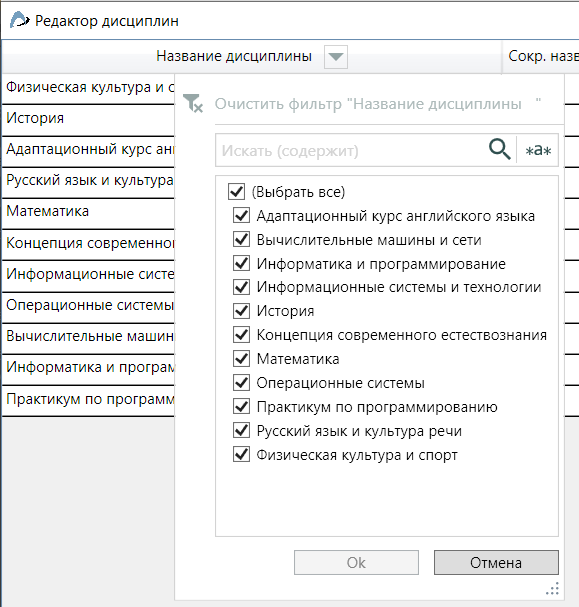
**Рисунок 76 – Редактирование дисциплины**

Добавим еще несколько дисциплин, и проверим, как выполняется поиск и фильтрация (см. рис. 77).



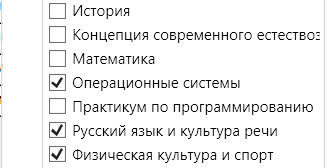
**Рисунок 77 – Добавление дисциплин**

По нажатию на одну из стрелок в заголовке списка, откроется окошко поиска и фильтрации дисциплин, в котором можно как ввести название дисциплины для поиска, так и отметить необходимые в исписке (см. рис. 78).



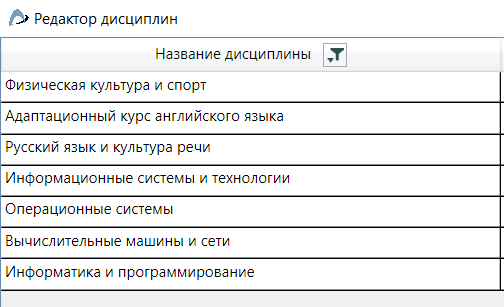
**Рисунок 78 – Окошко поиска и фильтрации**

Отметим галочкой ненужные дисциплины (см. рис. 79).



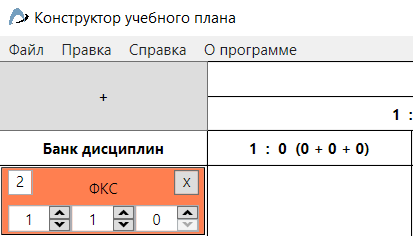
**Рисунок 79 – Окошко поиска и фильтрации**

Результат фильтрации по названию дисциплины представлен на рисунке 80.



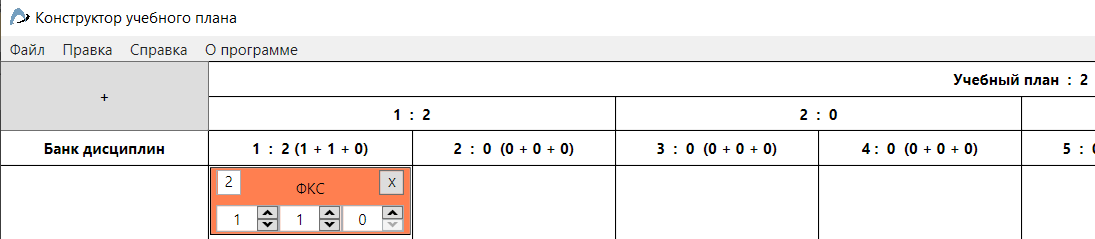
**Рисунок 80 – Отфильтрованный список дисциплин**

Далее выберем одну дисциплину, и она автоматически в виде плитки добавится в банк редактирования дисциплин (см. рис. 81).



**Рисунок 81 – Результат пополнения банка дисциплин**

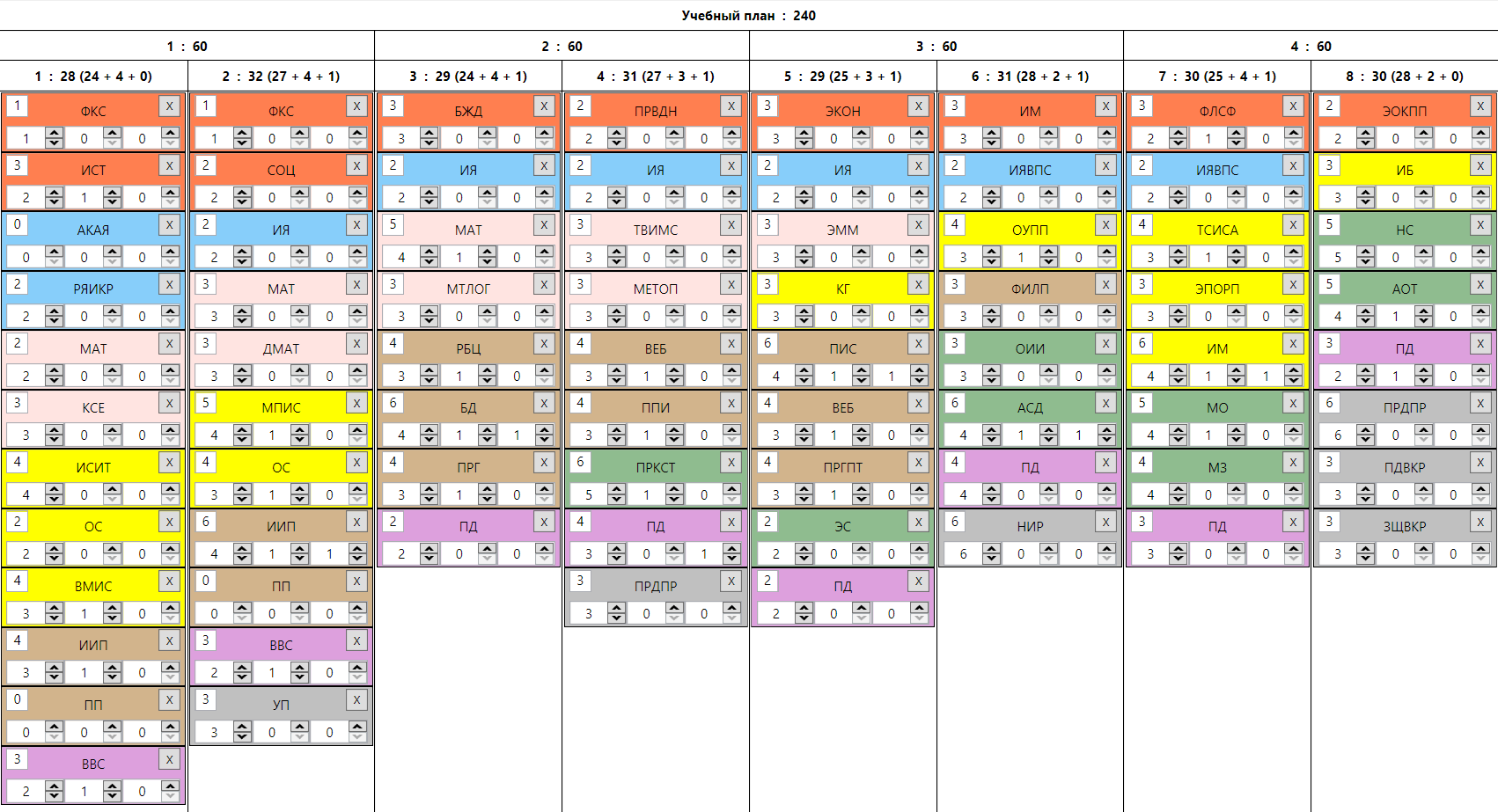
При перемещении дисциплины из банка дисциплин в один из семестров, происходит изменение значений сумм з.е. семестров, курсов и общего количества (см. рис. 82).



**Рисунок 82 – Изменение значений з.е. для сумм**

Кнопка в виде крестика - кнопка «Удалить плитку». По нажатию на нее, дисциплина сначала вернется в банк дисциплин. А при нажатии в банке дисциплин, вовсе исчезнет из главного окна.

Добавим в окно редактора дисциплин реальные дисциплины, которые входят в программу обучения в ОГУ им. И.С. Тургенева. Выберем все и распределим по семестрам. Результат представлен на рисунке 83.



**Рисунок 83 – Реальный учебный план в виде плиток**

Также, если необходимо, можно выгрузить проект в файл Excel, или сохранить его на диске. Справку и информацию о программе можно посмотреть нажав соответствующие кнопки на меню в главном окне.

# **Заключение**

В ходе выполнения данной работы были решены следующие задачи:

* + - 1. Проанализирована предметная область и сформированы требования к приложению.
      2. Построены и описаны диаграммы прецендентов.
      3. Проведен обзор и анализ аналогов.
      4. Спроектирована архитектура разрабатываемого приложения.
      5. Спроектирован пользовательский интерфейс.
      6. Сконструированы макеты графического интерфейса.
      7. Разработаны алгоритмы функционирования.
      8. Построены логическая и физическая модели базы данных.
      9. Выбраны инструментальные средства разработки.
      10. Описана реализация.
      11. Проведено тестирование приложения.

Таким образом, поставленные задачи были выполнены, а цель работы, разработка приложения для формирования ядра учебного плана, была достигнута.

# **Список использованной литературы**

# **Приложение А**