МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

Направление специальности 1-40 01 01 Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема «Картотека»

Исполнитель

студент 2 курса группы 4 Столяров Денис Сергеевич

(Ф.И.О.)

Руководитель работы ассистент Рауба А.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Рауба А.А.

(подпись)

Минск 2020

Содержание

[ВВЕДЕНИЕ 4](#_Toc41249569)

[1. Постановка задачи и аналитический обзор прототипов 5](#_Toc41249570)

[1.1 Используемые технологии 5](#_Toc41249571)

[1.2 Обзор прототипов 6](#_Toc41249572)

[2. Разработка функциональной модели приложения 8](#_Toc41249573)

[3. Разработка архитектуры проекта 10](#_Toc41249574)

[3.1 Структура проекта 10](#_Toc41249575)

[3.2 Проектирование базы данных 11](#_Toc41249576)

[3.3 Проектирование доступа к базе данных 12](#_Toc41249577)

[3.4 Проектирование архитектуры приложения 12](#_Toc41249578)

[3.5 Проектирование уровня представления 13](#_Toc41249579)

[4 Реализация программного средства 15](#_Toc41249580)

[4.1 Реализация сущностей 15](#_Toc41249581)

[4.2 Реализация уровня доступа к данным 15](#_Toc41249582)

[4.3 Реализация архитектуры MVVM 16](#_Toc41249583)

[4.4 Реализация представления 17](#_Toc41249584)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 19](#_Toc41249585)

[ЗАКЛЮЧЕНИЕ 23](#_Toc41249586)

[Список используемых источников 24](#_Toc41249587)

[ПРИЛОЖЕНИЕ А 25](#_Toc41249588)

[ПРИЛОЖЕНИЕ Б 26](#_Toc41249589)

[ПРИЛОЖЕНИЕ В 27](#_Toc41249590)

[ПРИЛОЖЕНИЕ Г 28](#_Toc41249591)

[ПРИЛОЖЕНИЕ Д 29](#_Toc41249592)

[ПРИЛОЖЕНИЕ Е 30](#_Toc41249593)

[ПРИЛОЖЕНИЕ 6 31](#_Toc41249594)

# ВВЕДЕНИЕ

В повседневной жизни большинство студентов сталкиваются с проблемой незнания: какие предметы и преподаватели ждут их на определенном курсе, а также о составе их группы (в частности первокурсники). Достичь наиболее эффективной работы можно с помощью автоматизации. Она позволит хранить, структурировать и систематизировать данные студента.

Данная тема курсового проекта была выбрана для того, чтобы при помощи разработанной программы решить проблемы, описанные выше. Данное программного средства поможет отобрать только те данные, которые взаимосвязаны со студентом.

Для разработки графической составляющей данного продукта используется технология Windows Presentation Foundation (WPF), которая позволяет обеспечить приложению насыщенный дизайн и интерактивность.

# 1. Постановка задачи и аналитический обзор прототипов

Программное средство должно выполнять следующие функции:

Функции администратора:

* + Выполнять авторизацию,
  + Регистрировать новых пользователей,
  + Обновлять и редактировать записи,
  + Выполнять поисковые запросы,
  + Выполнять импорт и экспорт записей,
  + Выполнять удаление записей,

Функции студента:

* Выполнять авторизацию,
* Просматривать соответствующую информацию;

## Используемые технологии

Для реализации ПС были выбраны следующие технологии:

* + ***C#*** — объектно-ориентированный язык программирования. Разработан в 1998 — 2001 под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework.C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем.
  + **Windows Presentation Foundation (WPF)** – это современная графическая система отображения для Windows. Это радикальное изменение от технологий, которые были до него, с инновационными функциями, такими как встроенные аппаратное ускорение и независимость от разрешения
    - **Microsoft SQL Server** — [система управления реляционными базами данных (РСУБД)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Основной используемый язык запросов — [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL), создан совместно Microsoft и [Sybase](https://ru.wikipedia.org/wiki/Sybase). Transact-SQL является реализацией стандарта [ANSI](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2)/[ISO](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8) по структурированному языку запросов ([SQL](https://ru.wikipedia.org/wiki/SQL)) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

## Обзор прототипов

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов.

На сегодняшний день можно встретить большое количество программных решений, разработанных для автоматизации учета записей.

Среди программных решений популярными являются:

* «Exiland Assistent»,
* «File Cabinet Pro»,
* «Электронная картотека»;

«Exiland Assistent» ⎯ это прикладное программное обеспечение для организаций, которое позволяет хранить множество контактной и личной информации. Это могут быть Ф.И.О., телефоны (рабочий, домашний, мобильный), email адреса, адреса проживания и прописки, адрес web-сайта,  Skype,  ссылки на Web-ресурсы и файлы, дата рождения, учетные записи в социальных сетях и мессенджерах, паспортные данные,  фото, увлечения, события.

Интерфейс «Exiland Assistent» представлен на рисунке 1.1.

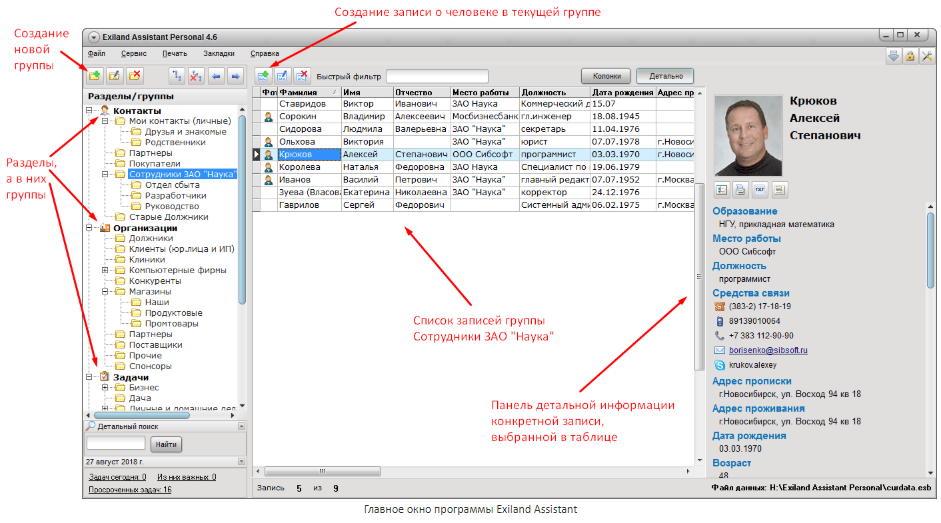


Рисунок 1.1 − Интерфейс «Exiland Assistent»

Проанализировав систему «Exiland Assistent», можно выделить её основные плюсы и минусы.

Из плюсов особенно выделяются:

* широкий функционал;
* простой и удобный интерфейс с гибкой настройкой;
* гибкие настройки разных этапов работы и отчетов;
* планировщик задач и личных дел с возможностью напоминания

Основные минусы:

* сложность настройки и доработки системы;
* сложная и запутанная документация;
* сложная API с большим числом ограничений;
* качество работы поддержки на неудовлетворительном уровне.

# 2. Разработка функциональной модели приложения

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

* простота и лёгкость интерфейса;
* использование принципов объектно-ориентированного программирования;
* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы. Пользователь со статусом администратора данного программного решения должен иметь возможность:

* входить в приложение, после ввода данных, необходимых для аутентификации,
* регистрировать пользователей в системе,
* редактировать данные,
* просматривать данные о студентах, предметах, преподавателях, факультетах и регистрационных данных,
* вносить новую информацию о студентах, предметах, преподавателях, факультетах и регистрационных данных,
* удалять информацию о студентах, предметах, преподавателях, факультетах и регистрационных данных,
* экспортировать и импортировать информацию в формате JSON;
* просматривать информацию с использованием поиска;
* прикреплять изображения к информации о студентах и преподавателях;

Пользователь со статусом пользователя данного программного решения должен иметь возможность:

* входить в приложение, после ввода данных, необходимых для аутентификации,
* просматривать данные о студентах, которые находятся в одной группе с пользователем,
* просматривать данные о преподавателях, которые будут у студента на протяжении курса,
* просматривать личные данные;

Данные требования обобщены в виде диаграммы вариантов использования программного средства, которая приведена в приложении А. Она отражает функциональность программного средства с точки зрения получения значимого результата для пользователя.

К последнему уровню требований относятся функциональные требования. Функциональные требования определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований. Данные требования могут быть описаны в виде утверждений, способов взаимодействия и методов реализаций. После проведения анализа были выявлены следующие функциональные требования:

* архитектура приложения должна соответствовать шаблонам проектирования, таким как MVVM, Command, Repository, Unit of Work,
* вся информация должна храниться в базе данных,
* в приложении должна быть многоуровневая архитектура,
* приложение должно производить проверку на корректность вводимых пользователем данных,
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке,
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем,
* приложение должно предоставлять возможность добавлять объекты с помощью отдельной панели,
* приложение должно предоставлять возможность поиска записей;

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

# 3. Разработка архитектуры проекта

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

## 3.1 Структура проекта

В данном курсовом проекте используется многоуровневая архитектура.

Классическая реализация схема многоуровневой архитектуры представлена на рисунке 3.1.

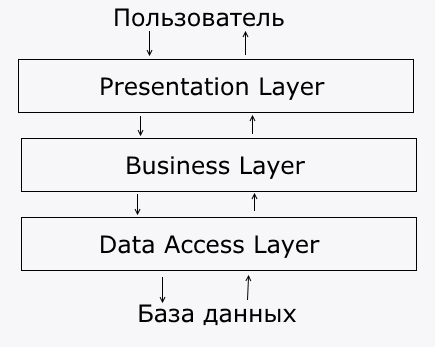


Рисунок 3.1 − Схема многоуровневой архитектуры

**Presentation layer** (уровень представления): это тот уровень, с которым непосредственно взаимодействует пользователь. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя.

**Business layer** (уровень бизнес-логики): содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

**Data Access layer** (уровень доступа к данным): хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, например, класс контекста данных Entity Framework. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных.

Структура проекта представлена в приложении Б.

## 3.2 Проектирование базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное (инфологическое) и логическое (даталогическое) проектирование.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. В результате этого этапа создаётся ER-модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности − это любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана ER-модель, которая включает 9 сущностей:

* Группа,
* Студент,
* Факультет,
* Предмет,
* Преподаватель,
* Специальность
* Авторизация,
* Кафедра,
* Связывающая сущность ПредметПреподаватель;

Также в ER-модели были определены необходимые связи. Например, между сущностями преподаватель и предмет была установлена связь многие-ко-многим. Для каждой сущности были выделены атрибуты.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД. Даталогическая модель базы данных представлена в приложении В.

## 3.3 Проектирование доступа к базе данных

Для инкапсулирования логики работы с источниками данных в программе используется паттерн Репозиторий. Класс, реализующий данный паттерн, не содержит бизнес-логику, не управляет бизнес-процессами, он только содержит операции над данными. Как правило, репозиторий реализует CRUD-интерфейс, то есть представляет операции по извлечению, добавлению, редактированию и удалению данных.

Для организации доступа к одному подключению для всех репозиториев приложения используется другой паттерн − Unit Of Work.

## 3.4 Проектирование архитектуры приложения

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбранных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.2 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.2 – Структура шаблона MVVM

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Представление 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 элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Model или Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения. В данном решении в роли модели будет выступать Репозиторий, который организовывает доступ к хранилищу данных − базе данных.

## 3.5 Проектирование уровня представления

Пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, поэтому разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей.

Общие требования к интерфейсу программы:

* любое окно может быть закрыто;
* все ошибки валидации должны быть отображены и видны пользователю.

После запуска программы должно быть открыто окно, с помощью которого пользователь может осуществить вход в систему. Стартовое окно должно быть интуитивно понятным и простым. После аутентификации пользователя в случае успеха должно быть открыто главное окно, при этом стартовое окно должно закрыться. В случае неуспеха должно быть показано окно с информацией, описывающей ошибку, которая привела к этой ситуации.

Главное окно должно быть реализовано в виде контейнера для страниц. Оно должно содержать удобное меню для навигации по страницам и панель управления. Для каждого пункта меню будет вызываться соответствующая страница, при этом такие элементы, как меню и панель управления будут оставаться неизменными.

Каждая страница должна соответствовать следующим требованиям:

* страница должна содержать таблицу, отображающую данные об одной конкретной сущности;
* на странице должна быть панель для добавления нового объекта сущности;
* на странице должна быть реализована возможность удаления и редактирования данных.

4 Реализация программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

## 4.1 Реализация сущностей

В соответствии с требованиями в качестве хранилища данных программного средства должна быть база данных, поэтому первым шагом в реализации программы является выбор технологии, позволяющей это осуществить. Выбор остановился на ORM технологии Entity Framework. Она предоставляет три подхода по проектированию базы данных. В данном программном решении был использован подход Database-First. При данном подходе используется модель EDMX. Создание базы данных происходит вручную и из созданной базы данных формируются модели объектов C#. Созданные модели объектов совпадают с сущностями, которые были сформированы раннее в разделе 3.1.

Диаграмма классов UML для сущностных классов представлена в приложении Г.

Подробный код класса, контекста сущностей, представлен в приложении Д.

## 4.2 Реализация уровня доступа к данным

В программном решении доступ к данным должен быть организован с помощью паттернов Repository и Unit of Work.

Паттерн Repository позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. В соответствии с паттерном Repository был создан обобщенный интерфейс IRepository<Entity, Id> и соответствующие классы для каждой сущности, которые реализуют этот интерфейс. Код интерфейса IRepository <Entity, Id> представлен в приложении E.

Паттерн Unit of Work представлен классом UnitOfWork, в котором находятся реализации методов создания репозиториев. Также этот класс использует интерфейс IDbContextFactory<T>. Интерфейс IDbContextFactory<T> объявляет один единственный метод CreateDbContext, в котором при реализации интерфейса в классе должно происходить создание контекста данных. Код интерфейсов IUnitOfWork и IDbContextFactory<T> представлен в приложении Е.

Полная структура элементов, а именно двух классов и одного интерфейса, данного уровня представлена на рисунке 4.1.

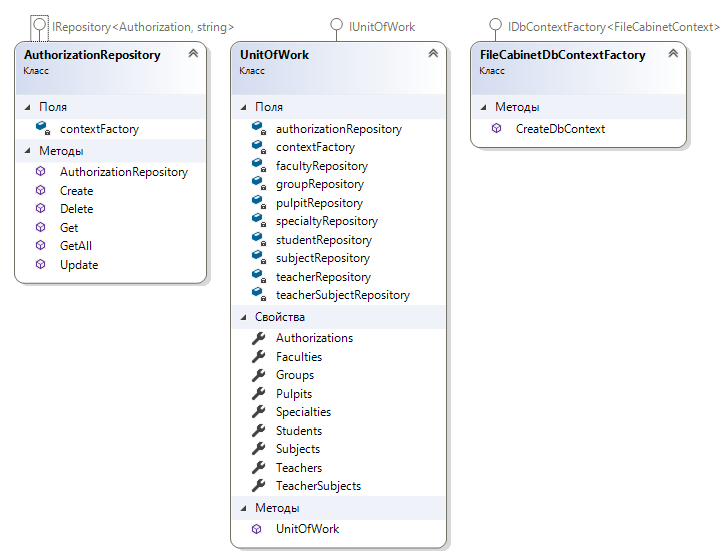


Рисунок 4.1 – Repository и Unit of Work

## 4.3 Реализация архитектуры MVVM

Для ускорения создания и разработки приложений с архитектурой MVVM существует ряд различных библиотек. В качестве такой библиотеки в данном решении был выбран инструментарий PropertyChanged.Fody который помогает упростить привязку данных и сократить написание кода.

Для реализации паттерна MVVM файлы программы были распределены по соответствующим пространствам имён и реализовали соответствующие функции. Разделение проекта на логические модули представлено на рисунке 4.2.

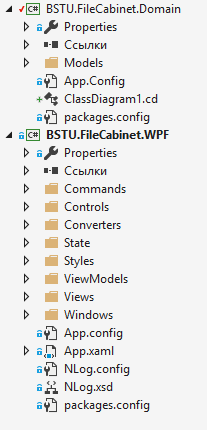


Рисунок 4.2 – Логические модули проекта

В папке Models содержатся сущностные классы, которые используются для создания БД. Их описание представлено в разделе 4.1.

В папке ViewModels находятся все классы ViewModel для каждого представления. Программа спроектирована таким образом, что в ней предполагается одна и та же работа над разными сущностями, поэтому для удобства был выделен абстрактный класс BaseViewModel. Также в папку State вынесены классы для работы с навигацией по страницам.

В папке Views хранятся все используемые представления.

## 4.4 Реализация представления

Для разработки графической части приложения была выбрана технология WPF.

Windows Presentation Foundation (WPF) — это библиотека для создания пользовательских интерфейсов для интеллектуальных клиентских приложений. В основе WPF лежит мощная инфраструктура, основанная на DirectX — API-интерфейсе графики с аппаратным ускорением, который обычно используется в современных компьютерных играх. Это означает, что применение развитых графических эффектов не приведёт к снижению производительности.

Одной из важных особенностей WPF является использование языка декларативной разметки интерфейса XAML, основанного на XML. Разработка с использованием XAML позволяет отделить графический интерфейс от логики приложения, а также создавать насыщенный интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#.

В конечном итоге в приложении «Картотека» было реализовано три окна:

* стартовое окно (вход в систему);
* окно администратора (содержит различные страницы);
* окно пользователя;

А также девять страниц на стороне администратора для редактирования под каждую сущность и три окна на стороне пользователя для просмотра информации.

В результате выполнения данного этапа было создано функционирующее программное средство.

5 Тестирование, проверка работоспособности и анализ полученных результатов

Прежде всего были проведены тесты работоспособности класса-репозитория: проверка на взаимодействие с базой данных и выполнения основных операций. Пример теста приведен на рисунке 5.1.

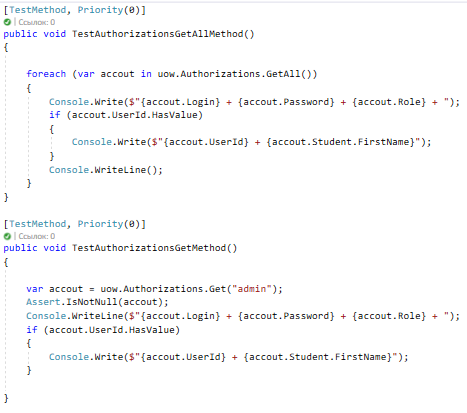


Рисунок 5.1 – Тест работоспособности репозитория

Проверка на корректность введенных данных осуществляется с помощью сервиса IAuthorizationService, который проверяет валидность данных и возвращает нам тип запрашиваемого окна, на оснавании которого и происходит генерация главного окна. Процесс авторизации представлен в приложении Ж. В противном случае будет сгенерировано исключение. Пример теста приведен на рисунке 5.2.

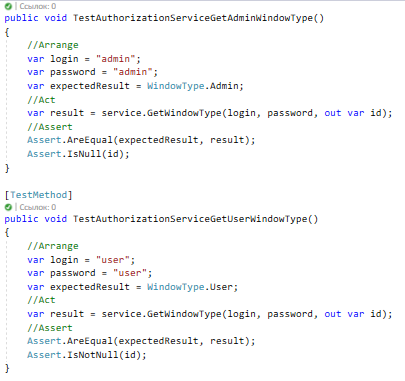
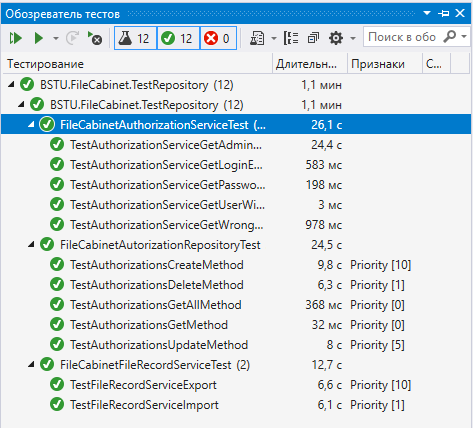


Рисунок 5.2 – Тест работоспособности сервиса

Также еще производится тестирование CommonFileRecordService который отвечает за импорт и экспорт записей в нашем приложении. Результаты прохождения всех тестов представлены на рисунке снизу.

Проверки работоспособности валидации со стороны пользователя представлены на рисунках 5.3− 5.4

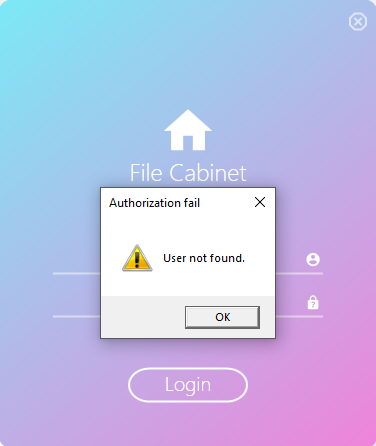


Рисунок 5.3 – Валидация формы входа

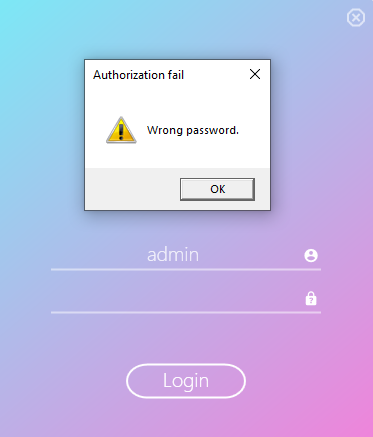


Рисунок 5.4 – Вход с неверным паролем

Также были проведены тесты на работоспособность внутри приложения. Наличие проверки является обязательным в связи с тем, что в приложении постоянно ведётся работа с базой данных.

При нарушении корректности ввода данных будет выведена ошибка приведеная на рисунке 5.5.

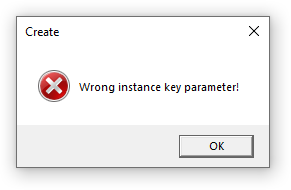


Рисунок 5.5 – Ошибка ввода

На данном этапе были выполнены тесты на проверку работоспособности приложения, а именно на проверку данных.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта был изучен теоретический материал по теме проекта, рассмотрены аналоги, выявлены их преимущества и недостатки.

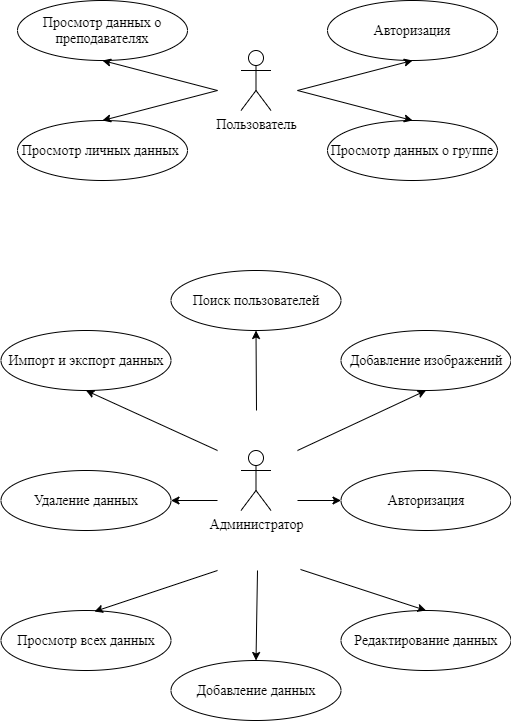
Для реализации программного средства были изучены и применены технологии такие, как WPF, Entity Framework, Microsoft SQL Server. Само приложение было реализовано на языке C# в среде Visual Studio. Для структуры проекта применён архитектурный паттерн MVVM, для работы с базой данной – Repository и UnitOfWork.

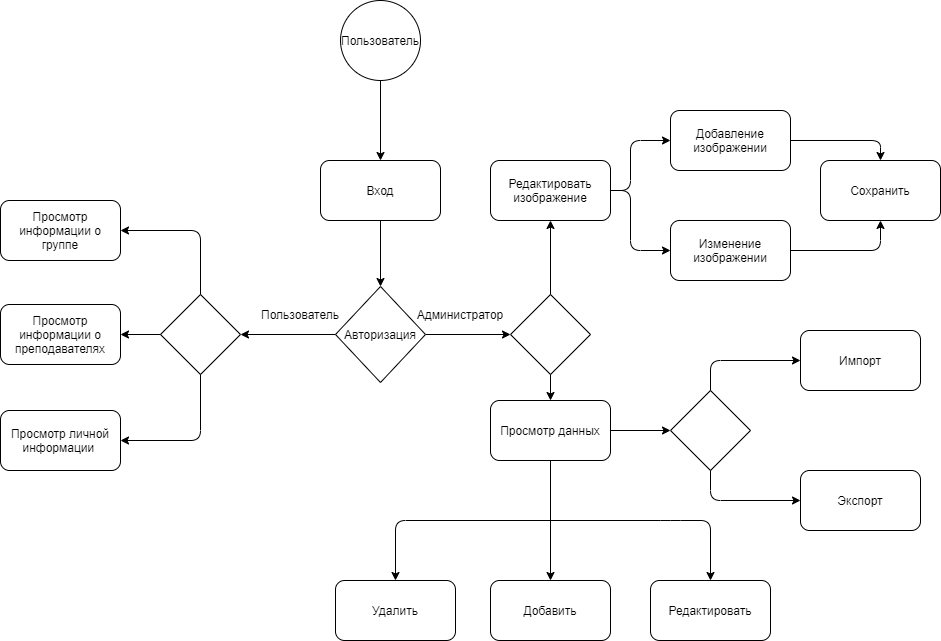
В данном курсовом проекте было реализовано приложение “Картотека”. Оно позволяет удобно вести электронные записи, а также облегчает поиск информации о предстоящем семестре для студента; помогает найти сведения о своих одногруппниках и преподавателях, которые будут вести предметы.

# Список используемых источников

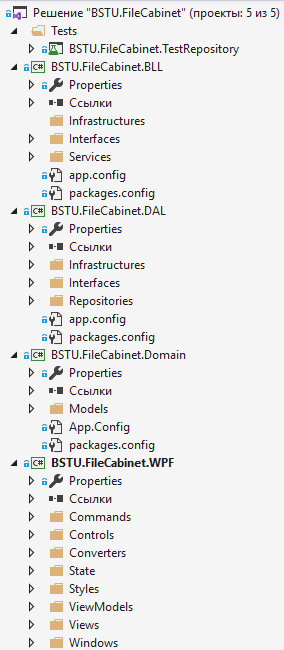
1. Портал вопросов и ответов для программистов[Электронный ресурс]/stackoverflow.com– Режим доступа http://stackoverflow.com
2. Руководство по Visual Studio[Электронный ресурс]/msdn.microsoft.com– Режим доступа http://msdn.microsoft.com
3. Сайт о программировании Metanit: Введение в MS SQL Server [Электронный ресурс] / metanit.com – Режим доступа: <https://metanit.com/sql/sqlserver/1.1.php>
4. Пацей, Н.В. Курс лекций по языку программирования C# / Н.В. Пацей. – Минск: БГТУ, 2016.

# ПРИЛОЖЕНИЕ А

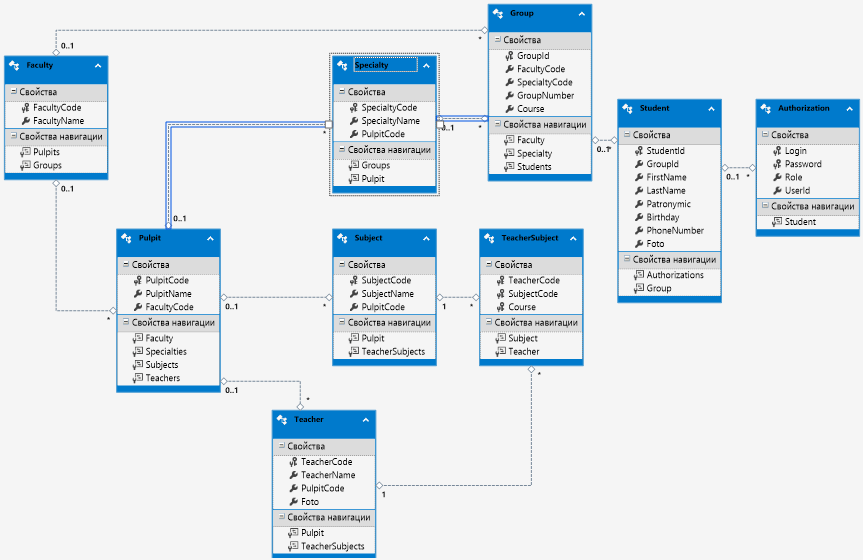




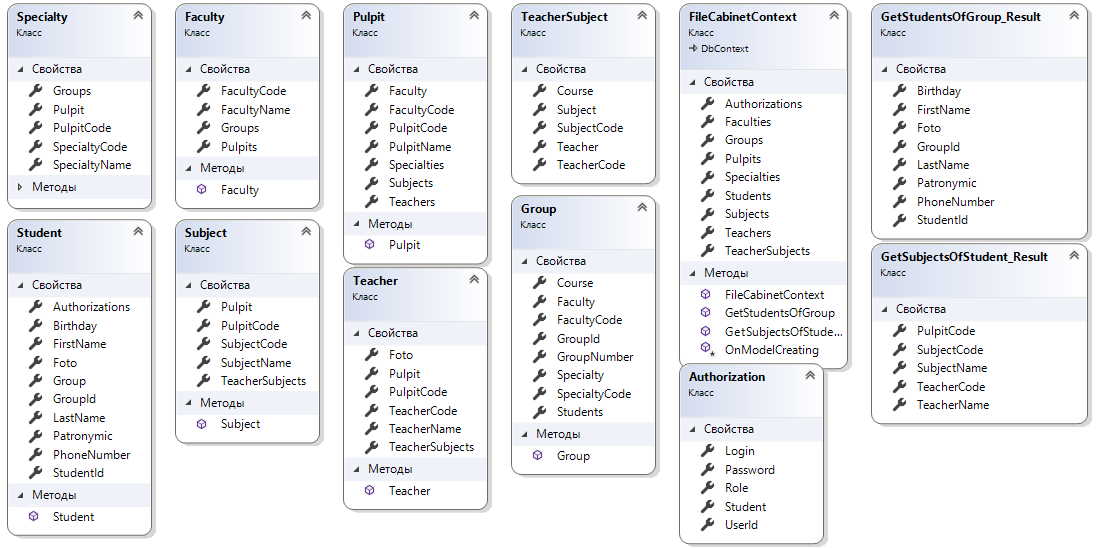
# ПРИЛОЖЕНИЕ Б



ПРИЛОЖЕНИЕ В

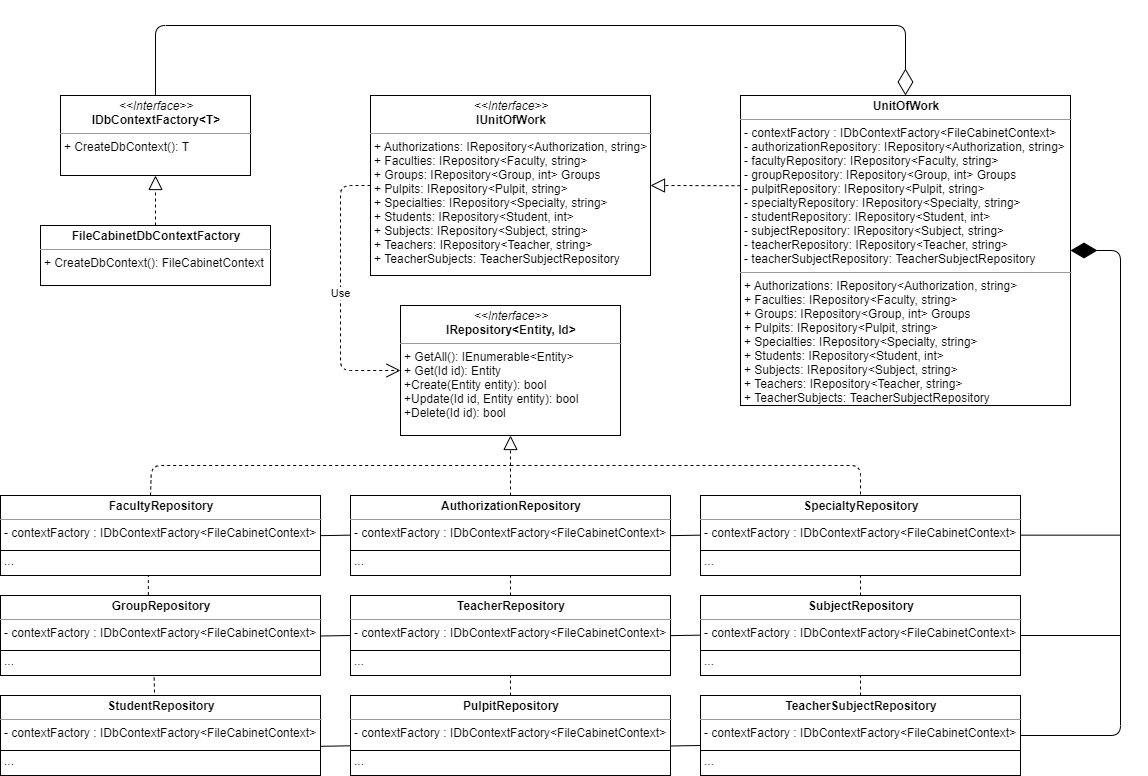


# ПРИЛОЖЕНИЕ Г

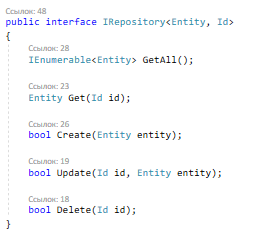
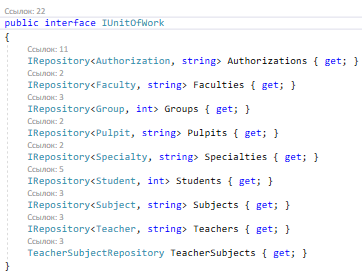
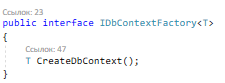


# ПРИЛОЖЕНИЕ Д





# ПРИЛОЖЕНИЕ Е



# ПРИЛОЖЕНИЕ Ж

