МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ЧАСТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

««МЕЖДУНАРОДНЫЙ ИНСТИТУТ ДИЗАЙНА И СЕРВИСА»

(ЧОУВО МИДИС)

Кафедра математики и информатики

**КУРСОВАЯ РАБОТА**

По дисциплине «Объектно-ориентированный анализ

и программирование на языке С#»

на тему: «Проектирование и разработка автоматизированной системы управления для приёмной комиссии ВУЗа»

Направление прикладная информатика

Научный руководитель

Доцент КМиИ

Чеботарёв Сергей Сергеевич

Автор работы

Студент группы ПИ-225

Шакиров Артур Ренатович

Челябинск – 2022

**СОДЕРЖАНИЕ**

Введение.............................................................................................................................................3

Глава 1. Анализ предметной области...............................................................................................4

* 1. Актуальность проблемы.............................................................................................................4
  2. Выделение словаря предметной области.................................................................................4

1.3 Анализ существующих продуктов на рынке........……...……………………....................….5

1.4 Составление технического задания………………………..…………………....................…9

1.5 Выводы........................................................................................................................................10

Глава 2 Проектирование..................................................................................................................11

2.1 Бизнес-процессы........................................................................................................................11

2.2 Моделирование базы данных....................................................................................................11

2.3 Дизайн интерфейса пользователя.............................................................................................16

2.4 Выводы........................................................................................................................................24

Глава 3 Разработка...........................................................................................................................25

3.1 Инструменты разработки..........................................................................................................25

3.2 Архитектура приложения..........................................................................................................26

3.3 Выводы........................................................................................................................................40

Список литературы..........................................................................................................................41

**ВВЕДЕНИЕ**

Мы живём в век информационных технологий. Любая организация, желающая преуспеть на рынке, обязана применять современное программное обеспечение. И даже вузы не исключение. Например, необходимо организовать приёмную комиссию. Сотни абитуриентов, группы, потоки, кафедры, в голове такое держать невозможно. Необходимо создать программу способную хранить и изменять данные, а также создавать необходимые документы. Программа должна быть простой в использовании и легко настраиваемой.

Какие технологии необходимы для написания такой программы? Какие знания необходимы? С какими проблемами придется столкнуться? На эти вопросы и ответит наше исследование «Создание программы управления базой данных для организации работы Приёмной комиссии ВУЗа».

Главные задачи данной работы:

1. Рассмотреть актуальность проблемы создания автоматизированной системы управления для Приёмной комиссии ВУЗа.
2. Изучить существующие продукты на рынке.
3. Рассмотреть существующие инструменты разработки.
4. Сформировать техническое задание и разработать программное обеспечение.

**Методы исследования.** В курсовой работе были использованы такие методы как анализ существующих программных продуктов на рынке. Анализ литературы по теме исследования.

**Практическая значимость работы.** Результат курсовой работы может быть использован ВУЗами для организации работы Приёмной комиссии.

**Глава 1. Анализ предметной области**

* 1. **Актуальность проблемы**

Высшее образование всегда будет необходимо обществу, а вместе с ним и учебные заведения, которые это образование предоставляют. Но используя устаревшие методы работы нельзя достичь высоких результатов, более того можно разориться на издержках. Бумага — это прошлый век, сегодня век цифровой. Все храниться в цифровом виде. Это проще, дешевле, компактнее и безопасней. Используя программное обеспеченье можно ускорить работу, следовательно, сэкономить деньги. И ВУЗы не исключение.

Приёмная комиссия может принять сотни, а то и тысячи абитуриентов. Страшно подумать сколько бумаги понадобится для обслуживания такой задачи. Но с помощью системы управления можно свести траты к минимуму.

**1.2 Выделение словаря предметной области**

Проанализировав «ПОЛОЖЕНИЕ О ПРИЁМНОЙ КОМИССИИ» ЧОУВО «МИДиС» мы сформировали сущности необходимые для работы АСУ: Университет; Пользователь; Факультет; Кафедра; Поток; Группа; Абитуриент; Экзаменационный лист; Предмет; Экзамен; Консультация; Экзаменационная ведомость.

Также мы сформулировали процессы, которыми будет оперировать программа:

Добавление экземпляров сущности;

Удаление экземпляра сущности;

Редактирование экземпляра сущности;

Формирование экзаменационного листа в форме word документа;

Формирование экзаменационной ведомости в форме word документа;

**1.3 Анализ существующих продуктов на рынке**

Проанализировав рынок, а также официальные сайты некоторых ВУЗов РФ, мы обнаружили несколько существующих АСУ с функционалом необходимым для выполнения задач, поставленных в нашем исследовании.

# Автоматизированная программа «Абитуриент» в Белгородском государственном университете.

В целях оптимизации работы с персональными данными работников, студентов и абитуриентов Белгородского государственного университета (далее − БелГУ) управление кадров совместно с управлением информатизации разработало «Единую программу ведения данных по персональному учету». Эта программа включает в себя ряд специализированных систем, установленных в различных структурных подразделениях университета.

Комплекс программ «Абитуриент» разработан отделом автоматизированных систем БелГУ и ориентирован на делопроизводство университета. АИС «Абитуриент» интегрирована в online и offline режиме.

На сентябрь 2008 г. база имела объем информации равный примерно 50000 записей.

Таким образом, проследив историю создания, исследовав процесс внедрения и совершенствования «Единой программы ведения данных по персональному учету в БелГУ», мы пришли к выводу, что комплекс программ, разработанный непосредственно управлением информатизации БелГУ направлен на совершенствование и оптимизацию делопроизводственных процедур, связанных со спецификой приема граждан в ГОУ ВПО «Белгородский государственный университет». На наш взгляд, главной положительной чертой программного продукта, разработанного и поддерживаемого вузом, является экономия бюджетных средств, поскольку подразделение, создавшее и совершенствующее систему, включено в штат университета.

**АПК АСУ "Абитуриент-ЮФУ"**

C 20 июня разработанный в ЮФУ в 2011 г. и модернизированный в 2012 г. комплекс автоматизации деятельности приёмной комиссии ЮФУ работает в режиме постоянной эксплуатации во всех подразделениях университета.

В приёмную кампанию 2011 г. АПК АСУ "Абитуриент-ЮФУ" проходил пилотную эксплуатацию в структурных подразделениях университета в городах Ростов-на-Дону, Таганрог и Железноводск.

АПК АСУ "Абитуриент-ЮФУ" обеспечивает централизацию данных всех отделений приёмной комиссии ЮФУ и формирование единой базы данных приёмной кампании 2012 г.

Главные возможности АПК АСУ "Абитуриент-ЮФУ":

1. "Сквозная" подача заявлений (например: абитуриент, пришедший в отделение ЦПК в ИАрхИ, может подать заявление в любое подразделение ЮФУ).
2. Возможность подачи заявления on-line (абитуриент может подать заявление в ЮФУ не выходя из дома посредством электронного сервиса на официальном сайте ЮФУ).
3. Позволяет автоматически формировать общеуниверситетские сводки, отчеты, ранжированные списки, осуществлять централизованную выгрузку данных на официальный сайт ЮФУ по всем подразделениям.
4. Обеспечивает возможность централизованного обмена данными с федеральными системами.
5. Обеспечивает унификацию процессов в отделениях приёмной комиссии, единообразие в оформлении личных дел, договоров, приказов на зачисление и др. во всех подразделениях ЮФУ.
6. Обеспечивает автоматический перенос данных поступивших абитуриентов в единое информационное пространство ЮФУ, являясь его частью.

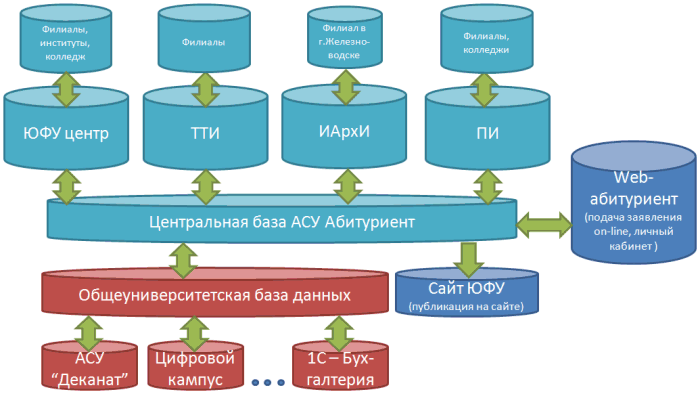


Рисунок 1 - структурная схема обмена информацией между узлами приемной комиссии и информационными системами ЮФУ.

Проанализировав создание, структурную схему и возможности АПК АСУ "Абитуриент-ЮФУ" мы пришли к выводу, что данная система выполняет задачи по автоматизации и повышению качества и скорости работы приёмной комиссии ВУЗа. Так же, как и у предыдущего продукта, главной положительной чертой АСУ является экономия денежных средств, в связи с тем, что продукт разработан и поддерживается сотрудниками университета.

# БИТ: Приемная комиссия

Решение «БИТ: Приемная комиссия» предусмотрено для автоматизация всего цикла задач приёмной комиссии вуза от регистрации данных абитуриентов до предоставления отчетности по итогам работы. Обеспечивает контроль хода приёма в режиме реального времени, автоматическое формирование итоговой отчетности по приёмной комиссии, единая база для хранения всех сведений об абитуриентах, повышение эффективности работы операторов приёмной комиссии. Разработчиком продукта является 1С: Первый БИТ (ранее 1С: Бухучет и Торговля)

Преимущества:

Автоматическое формирование ранжированного списка к зачислению;

Учёт льгот при поступлении;

Автоматизация проверки результатов ЕГЭ;

Программа может быть использована при создании информационной системы персональных данных любого класса, дополнительная сертификация решения не требуется.

Программа решает задачи:

Регистрация анкетных данных абитуриентов - по данным абитуриента реализовано формирование необходимого пакета документов (расписки, описи, анкета и т.д.);

Регистрация заявлений абитуриентов, при подаче заявления могут учитываться результаты единого государственного экзамена (ЕГЭ), целевые направления и письма направляющих организаций;

Автоматическая проверка результатов ЕГЭ по данным сайта ФБС (www.fbsege.ru);

Утверждение списка дисциплин, по которым сдаются вступительные экзамены, для каждой конкурсной группы;

Формирование списков рекомендованных к зачислению по результатам испытаний с учётом наличия у абитуриента льгот при поступлении в вуз, документов с отличием и результатов олимпиад;

Формирование отчетности для анализа данных об абитуриентах, поданных заявлениях и результатах вступительных испытаний.

Для анализа эффективности и успешности внедрения и использования продукта БИТ: Приемная комиссия мы проследили за одним из проектов компании [«Первый БИТ»](https://www.tadviser.ru/index.php/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F:1%D0%A1:%D0%9F%D0%B5%D1%80%D0%B2%D1%8B%D0%B9_%D0%91%D0%98%D0%A2_(%D1%80%D0%B0%D0%BD%D0%B5%D0%B5_1%D0%A1:%D0%91%D1%83%D1%85%D1%83%D1%87%D0%B5%D1%82_%D0%B8_%D0%A2%D0%BE%D1%80%D0%B3%D0%BE%D0%B2%D0%BB%D1%8F))

[«Первый БИТ»](https://www.tadviser.ru/index.php/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F:1%D0%A1:%D0%9F%D0%B5%D1%80%D0%B2%D1%8B%D0%B9_%D0%91%D0%98%D0%A2_(%D1%80%D0%B0%D0%BD%D0%B5%D0%B5_1%D0%A1:%D0%91%D1%83%D1%85%D1%83%D1%87%D0%B5%D1%82_%D0%B8_%D0%A2%D0%BE%D1%80%D0%B3%D0%BE%D0%B2%D0%BB%D1%8F)) и [Грозненский государственный нефтяной технический университет имени академика М.Д. Миллионщикова](https://www.tadviser.ru/index.php/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F:%D0%93%D1%80%D0%BE%D0%B7%D0%BD%D0%B5%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B5%D1%84%D1%82%D1%8F%D0%BD%D0%BE%D0%B9_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D0%B8%D0%BC._%D0%9C._%D0%94._%D0%9C%D0%B8%D0%BB%D0%BB%D0%B8%D0%BE%D0%BD%D1%89%D0%B8%D0%BA%D0%BE%D0%B2%D0%B0) завершили проект автоматизации работы приёмной комиссии учебного заведения на платформе программного продукта [«БИТ.ВУЗ.Приёмная комиссия»](https://www.tadviser.ru/index.php/%D0%9F%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82:%D0%91%D0%98%D0%A2:%D0%9F%D1%80%D0%B8%D0%B5%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BE%D0%BC%D0%B8%D1%81%D1%81%D0%B8%D1%8F). В связи с удалённостью университета и необходимостью скорейшей автоматизации, все работы по установке и настройке ПО, доработке исходной системы в соответствии с требованиями заказчика, обучение специалистов и запуск в эксплуатацию, компания [«Первый БИТ»](https://www.tadviser.ru/index.php/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F:1%D0%A1:%D0%9F%D0%B5%D1%80%D0%B2%D1%8B%D0%B9_%D0%91%D0%98%D0%A2_(%D1%80%D0%B0%D0%BD%D0%B5%D0%B5_1%D0%A1:%D0%91%D1%83%D1%85%D1%83%D1%87%D0%B5%D1%82_%D0%B8_%D0%A2%D0%BE%D1%80%D0%B3%D0%BE%D0%B2%D0%BB%D1%8F)) выполнила удаленно, в течение двух недель. Пользователями системы стали 16 сотрудников учебного заведения.

Проанализировав функционал и преимущества продукта, мы выяснили что продукт полностью выполняет задачи по автоматизации приемной комиссии. Тот факт, что разработчиком является сторонняя кампания, а не ВУЗ, как в случае с предыдущими продуктами, позволяет предположить, что у команды разработчиков будет преимущество в виде опыта разработки и внедрения продукта в разные ВУЗы. АСУ, созданные ВУЗами чаще всего, существуют в единственном экземпляре и поэтому их тестирование и развитие происходит существенно медленнее.

**1.4 Составление технического задания**

Требуется создать программную систему, предназначенную для работников приемной комиссии высшего учебного заведения. Она должна обеспечивать хранение, просмотр и изменение сведений об абитуриентах, а также о расписании экзаменов и консультаций. Результатом работы приемной комиссии должен быть список абитуриентов, зачисленных в институт.

Секретарь приемной комиссии регистрирует абитуриентов. Для каждого абитуриента в базу данных заносятся следующие сведения: фамилия, имя, отчество, паспортные данные, какое учебное заведение, где и когда окончил, наличие золотой или серебряной медали, название кафедры и факультета, на которые поступает абитуриент. При регистрации абитуриенту выдают экзаменационный лист, имеющий уникальный номер, и сообщают номер группы и потока.

Группы формируются на период вступительных экзаменов и объединяются в потоки по 3-4 группы. Для каждой группы по каждому предмету в базу данных заносится экзаменационная ведомость. Оценка, полученная абитуриентом, может быть изменена на апелляции.

Абитуриент может не только подать, но и забрать документы, а также перевести их на другую кафедру. Для каждого потока формируется расписание консультаций и экзаменов по предметам. Медалисты сдают только один экзамен. Известно количество мест на каждый факультет.

Приемная комиссия по результатам экзаменов должна сформировать списки абитуриентов, зачисленных в институт.

Секретарю приемной комиссии могут потребоваться следующие сведения: Список абитуриентов на заданный факультет? Оценки, полученные указанным абитуриентом? Когда и в какой аудитории будет консультация и экзамен у заданного абитуриента по указанному предмету? Где, когда и по каким предметам будут проходить экзамены у заданной группы? Конкурс на каждый факультет? Средний балл по каждому предмету на каждом факультете?

**1.5 Выводы**

Формирование грамотного технического задания необходимо для создания качественного программного продукта. Мы обнаружили, что только технических знаний недостаточно для формирования технического задания. Необходимо разбираться в предметной области. В нашем случае, в том, как работает приёмная комиссия ВУЗа.

**Глава 2. Проектирование**

**2.1 Бизнес-процессы**

Любой продукт создаётся для удовлетворения каких-либо потребностей. В нашем случае, продуктом является программное обеспеченье. А потребности — это задачи, которые выполняет программа, или по-другому — бизнес-процессы.

Задачи, выполняемые нашим приложением, включают в себя: хранение и изменение данных об абитуриентах, кафедрах, группах, потоках, факультетах, учебных предметах, консультациях и экзаменах.

Приложение должно уметь взаимодействовать с документами. Создавать экзаменационный лист для выдачи абитуриенту. Должно уметь создать экзаменационную ведомость в форме docx документа.

**2.2 Моделирование базы данных**

Создание реляционной базы данных начинается с формирования структуры таблиц. При этом определяется состав полей, их имена, тип данных каждого поля, размер поля, ключи, индексы таблицы и другие свойства полей. После определения структуры таблиц создается схема данных, в которой устанавливаются связи между таблицами.

Название БД — AdmissionsCommitteeBD (с англ.приёмная комиссия**)**

Сущности базы данных и их атрибуты:

1. **Учебное заведение** (Название, Адрес)
2. **Факультет** (Номер факультета, Название, Кол-во мест)
3. **Кафедра** (Номер кафедры, Номер факультета, Название)
4. **Абитуриент** (Номер абитуриента, Фамилия, Имя, Отчество, Паспортные данные, Наименование учебного заведения, Адрес учебного заведения, Дата окончания учебного заведения, Золотая медаль, Серебряная медаль, Номер экзаменационного листа)
5. **Экзаменационный лист** (Номер экзаменационного листа, Номер группы, Номер потока)
6. **Поток** (Номер кафедры, Номер потока, Название потока)
7. **Группа** (Номер потока, Номер группы, Название группы)
8. **Предмет** (Номер предмета, Название предмета, Количество баллов для получения оценки 3, Количество баллов для получения оценки 4, Количество баллов для получения оценки 5)
9. **Экзаменационная ведомость** (Номер экзаменационной ведомости, Номер расписания экзаменов, Номер экзаменационного листа, Баллы, Оценка)
10. **Расписание экзаменов** (Номер расписания экзаменов, Номер потока, Номер предмета, Аудитория, Дата экзамена)
11. **Расписание консультаций** (Номер расписания консультаций, Номер потока, Номер предмета, Название предмета, Аудитория, Дата консультации)
12. **Пользователь** (Номер пользователя, Логин, Пароль)

После описания сущностей перейдём, непосредственно, к созданию базы данных.

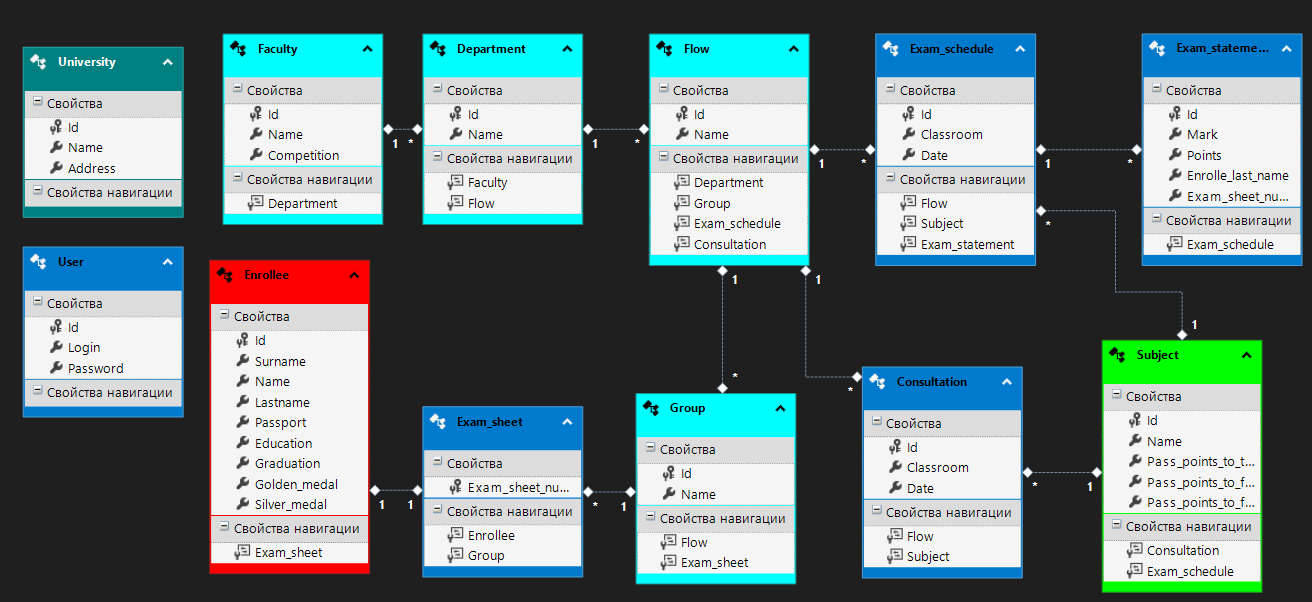


Рисунок 2 - представление базы данных в виде таблиц и связей

База данных создана. Опишем каждую таблицу.

**University**. Таблица предназначена для хранения информации об учебном заведении. Имеет три атрибута: id, Name, Address. Связи с другими таблицами отсутствуют.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Name нужен для хранения названия учебного заведения. Имеет тип string.

Атрибут Address нужен для хранения адреса учебного заведения. Имеет тип string.

**User.** Таблица предназначена для хранения информации о пользователях. Имеет атрибуты: id, Login, Password. Связи с другими таблицами отсутствуют.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Login предназначен для хранения логина пользователя. Имеет тип string.

Атрибут Password предназначен для хранения хеш версии пароля пользователя. Имеет тип string.

**Faculty.** Таблица предназначена для хранения информации о факультетах учебного заведения. Хранит в себе ссылку на таблицу **Department**(кафедра), потому что каждая кафедра относится к определённому факультету.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Name нужен для хранения названия факультета. Имеет тип string.

Атрибут Competition предназначен для хранения количества мест на факультет. Имеет тип int32.

**Department.** Таблица предназначена для хранения информации о кафедрах учебного заведения. Хранит в себе ссылку на таблицу **Flow**(поток), потому что каждый поток относится к определённой кафедре.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Name нужен для хранения названия кафедры. Имеет тип string.

**Flow.** Таблица предназначена для хранения информации о потоках учебного заведения. Хранит в себе ссылку на таблицу **Group**(группа), потому что каждая группа относится к определённому потоку. Также имеет ссылки на таблицы **Exam\_schedule** (расписание экзамена) и **Consultation**(консультация), так как для каждого потока формируется расписание консультаций и экзаменов по предметам.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Name нужен для хранения названия потока. Имеет тип string.

**Group.** Таблица предназначена для хранения информации о группах учебного заведения. Хранит в себе ссылку на таблицу **Exam\_sheet** (экзаменационный лист), потому что в каждой группе есть абитуриенты. Ссылка на экзаменационный лист, а не напрямую на абитуриента не является ошибкой, так как у каждого абитуриента свой уникальный экзаменационный лист. Зная лист, мы знаем абитуриента.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Name нужен для хранения названия потока. Имеет тип string.

**Exam sheet.** Таблица предназначена для хранения информации об экзаменационных листах. Имеет ссылку на **Enrollee (**абитуриент), так как каждый абитуриент имеет свой уникальный лист.

Имеет единственный атрибут Exam\_sheet\_number. Атрибут необходим для хранения уникального номера экзаменационного листа. Он же является первичным ключом. Имеет тип int32.

**Subject.** Таблица предназначена для хранения информации об учебных предметах.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Name нужен для хранения названия предмета. Имеет тип string.

Атрибут Pass\_points\_to\_three нужен для хранения информации о баллах необходимых для получения оценки три. Имеет тип int16.

Атрибут Pass\_points\_to\_four нужен для хранения информации о баллах необходимых для получения оценки четыре. Имеет тип int16.

Атрибут Pass\_points\_to\_five нужен для хранения информации о баллах необходимых для получения оценки пять. Имеет тип int16.

**Enrollee.** Таблица предназначена для хранения информации об абитуриентах. Имеет ссылку на таблицу **Exam\_sheet,** потому что каждый абитуриент имеет уникальный экзаменационный лист.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Surname нужен для хранения фамилии абитуриента. Имеет тип string.

Атрибут Name нужен для хранения имени абитуриента. Имеет тип string.

Атрибут Lastname нужен для хранения отчества абитуриента. Имеет тип string.

Атрибут Passport нужен для хранения паспортных данных абитуриента. Имеет тип string.

Атрибут Education нужен для хранения названия образовательного учреждения, которое закончил абитуриент. Имеет тип string.

Атрибут Graduation нужен для хранения даты окончания образовательного учреждения абитуриентом. Имеет тип DateTime.

Атрибут Golden\_medal нужен для хранения информации о наличии золотой медали у абитуриента. Имеет тип Boolean.

Атрибут Silver\_medal нужен для хранения информации о наличии серебряной медали у абитуриента. Имеет тип Boolean.

**Consultation.** Таблица предназначена для хранения информации об консультациях. Имеет ссылку на таблицу **Subject.** Потому что консультация проводится по определенному предмету.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Classroom нужен для хранения информации о кабинете, в котором будет проходить консультация. Имеет тип string.

Атрибут Date нужен для хранения даты консультация. Имеет тип DateTime.

**Exam\_schedule.** Таблица предназначена для хранения информации об экзаменах. Имеет ссылку на таблицу **Subject.** Потому что экзамен проводится по определенному предмету.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Classroom нужен для хранения информации о кабинете, в котором будет проходить экзамен. Имеет тип string.

Атрибут Date нужен для хранения даты экзамена. Имеет тип DateTime.

**Exam\_statement.** Таблица предназначена для хранения информации об экзаменационной ведомости. Имеет ссылку на таблицу **Exam\_schedule.** Потому что экзаменационная ведомость формируется по определенному экзамену.

Атрибут id является первичным ключом для записи и имеет тип int32.

Атрибут Enrollee\_last\_name нужен для хранения фамилии абитуриента. Имеет тип string.

Атрибут Exam\_sheet\_number нужен для хранения номера экзаменационного листа абитуриента. Имеет тип int32.

Атрибут Mark нужен для хранения оценки, полученной абитуриентом. Имеет тип Byte.

Атрибут Points нужен для хранения баллов, полученных абитуриентом. Имеет тип int16.

**2.3 Дизайн интерфейса пользователя**

Пользовательский интерфейс, или UI (User Interface) — это внешний вид продукта, способ общения между пользователем и программой. Программа не имеет смысла если ей невозможно пользоваться. Наше приложение предназначено для секретаря приёмной комиссии. Этот человек не программист, а значит ему нужен простой и понятный пользовательский интерфейс.

Используя такой графический редактор как paint.net, мы создали шаблоны страниц и окон пользовательского интерфейса.

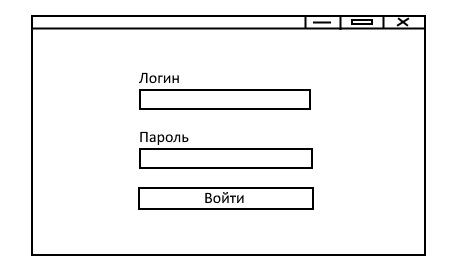


Рисунок 3 — страница авторизации.

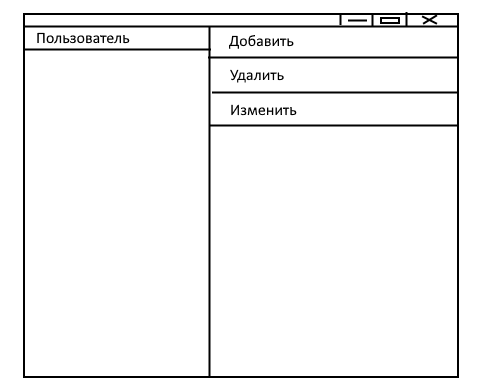


Рисунок 4 — страница администратора.

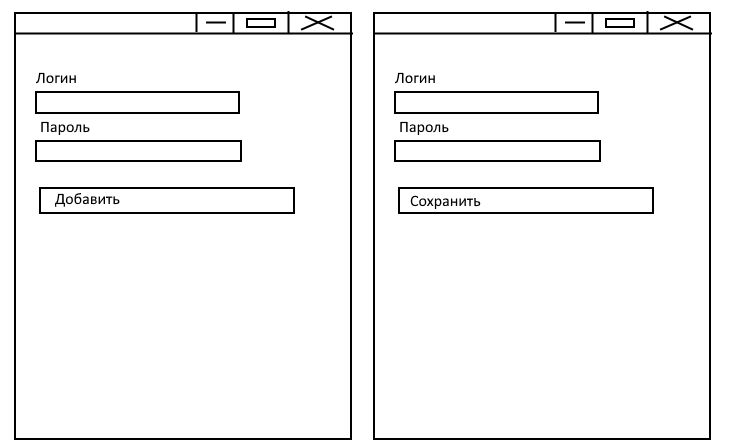


Рисунок 5 — окна добавления и редактирования пользователей.

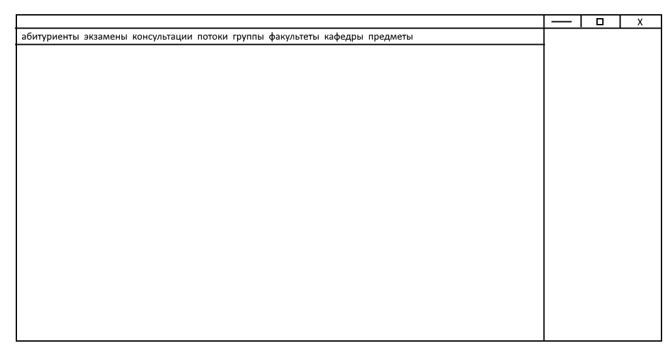


Рисунок 6 —главная страница приложения.

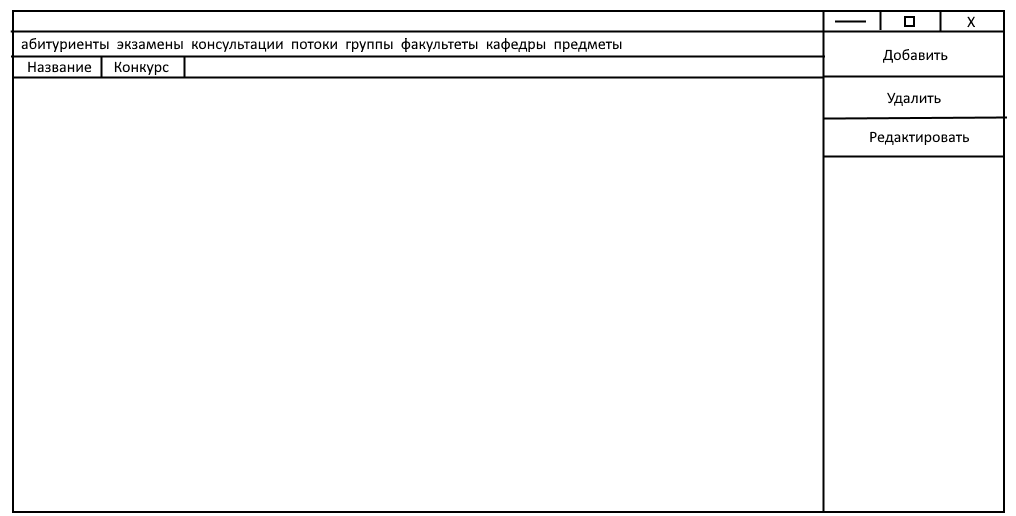


Рисунок 7 — страница факультетов.

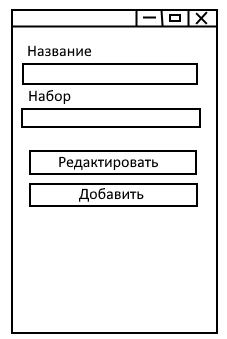


Рисунок 8 — окно редактирования и добавления факультетов.

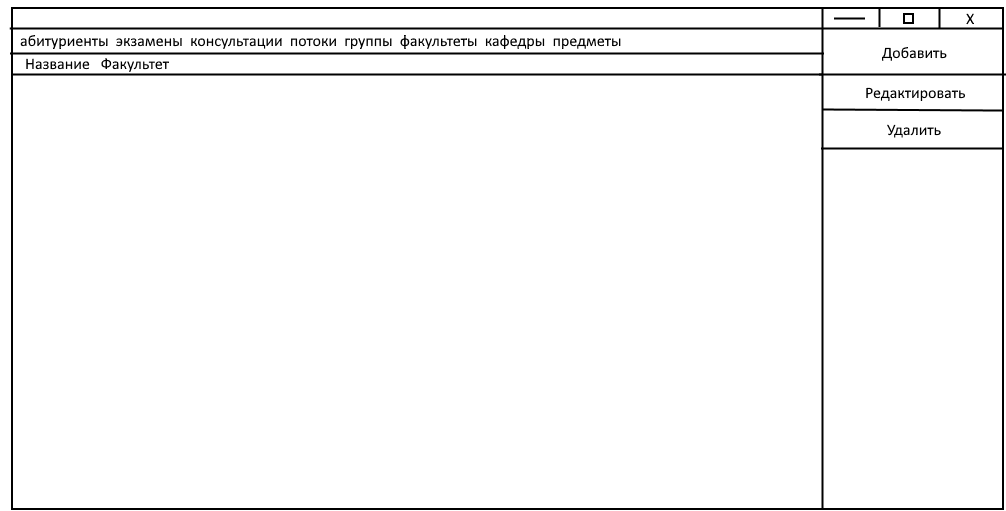


Рисунок 9 — страница кафедр

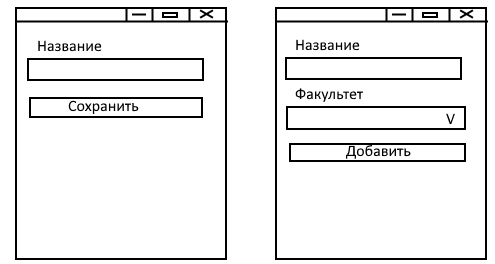


Рисунок 10 — окна добавления и редактирования кафедр.

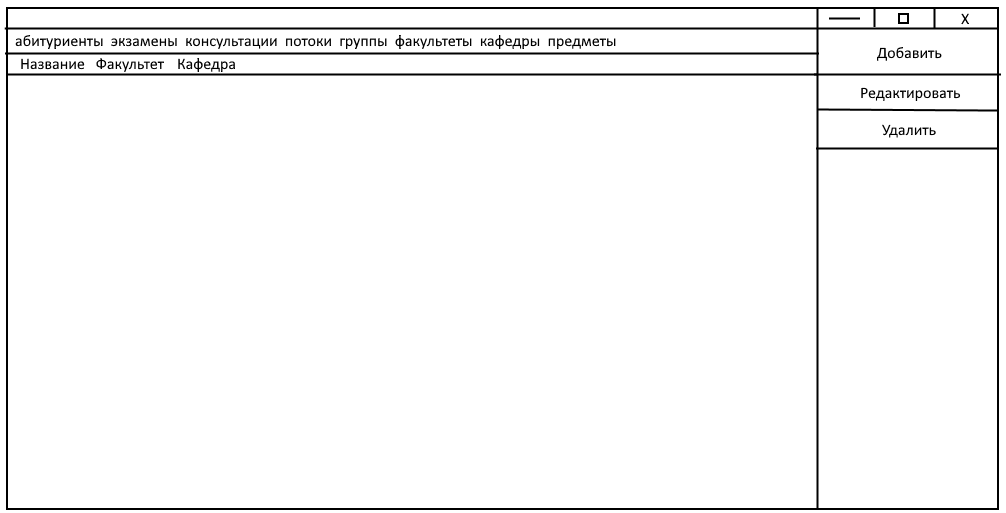


Рисунок 11— страница потоков.

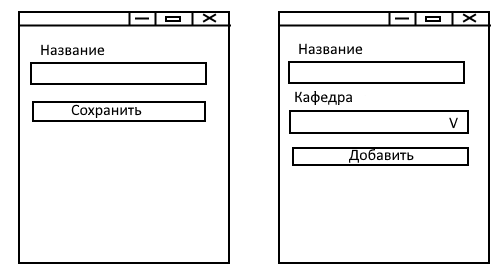


Рисунок 12 — окна добавления и редактирования потоков.

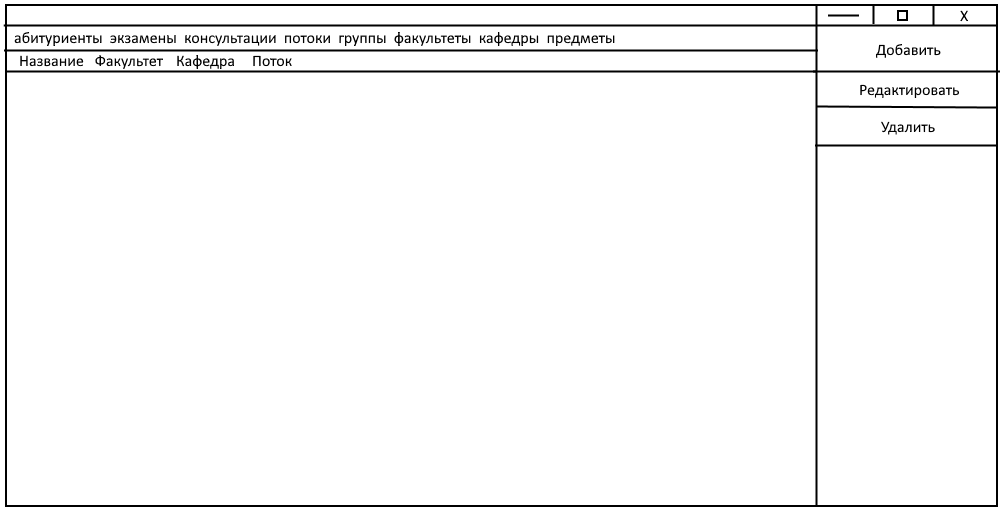


Рисунок 13 — страница групп.

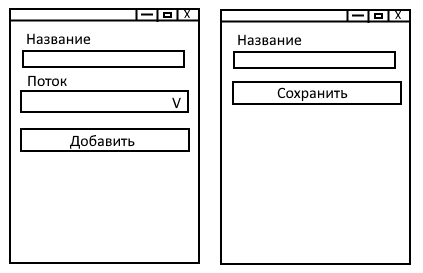


Рисунок 14 — окна добавления и редактирования групп.

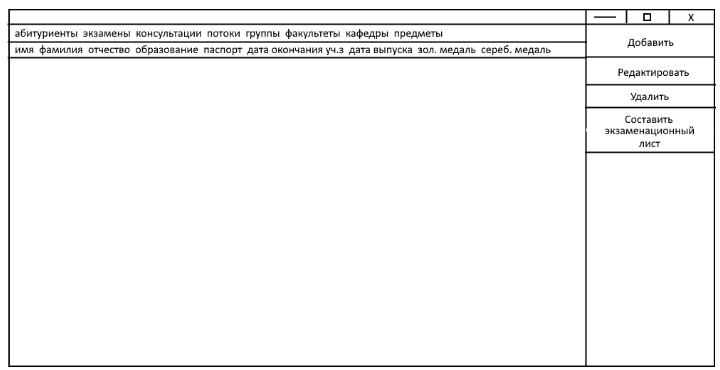


Рисунок 15 — страница абитуриентов.

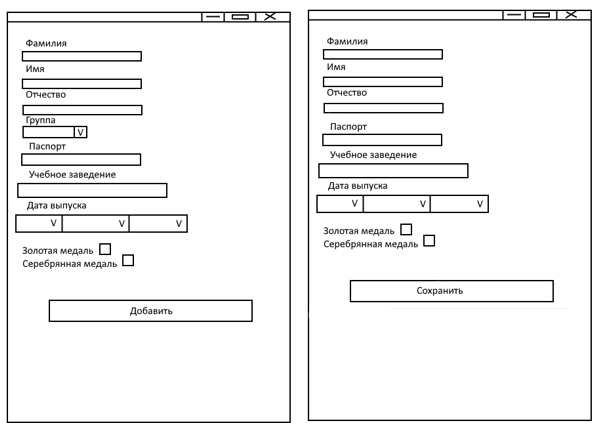


Рисунок 16 — окна добавления и редактирования абитуриентов.

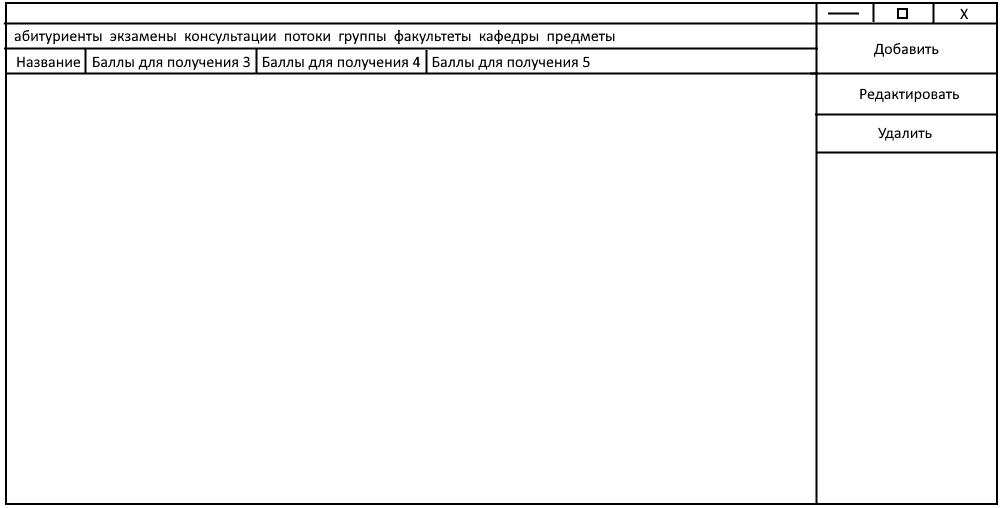


Рисунок 17 — страница предметов.

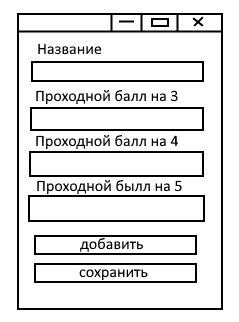


Рисунок 18 — окно добавления и редактировании предметов.

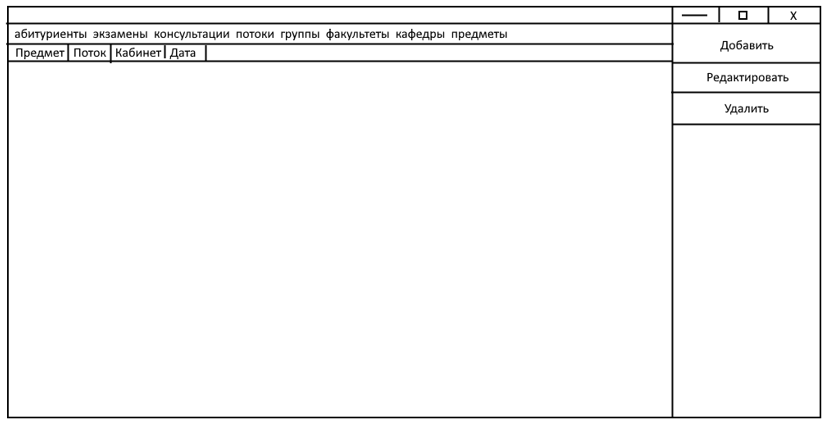


Рисунок 19 — страница консультаций.

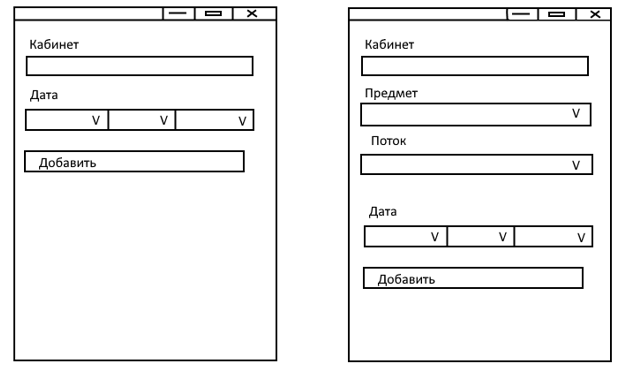


Рисунок 20 — окна добавления и редактирования консультаций.

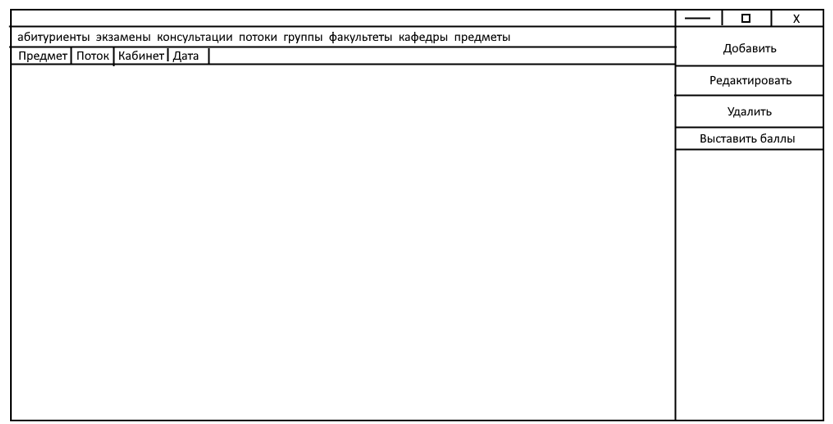


Рисунок 21 — страница экзаменов.

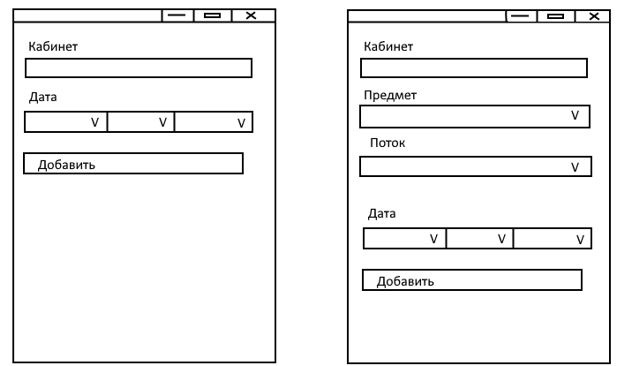


Рисунок 22 — окна добавления и редактирования экзаменов.

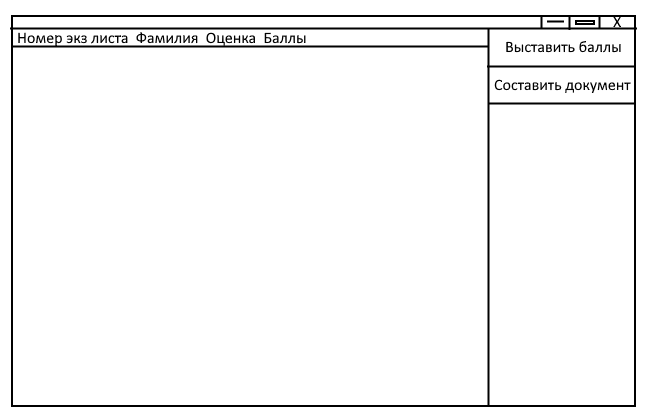


Рисунок 23 — окно составления экзаменационный ведомости.

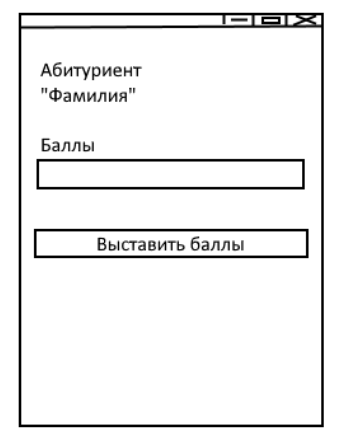


Рисунок 24 — Окно выставления баллов абитуриенту

**2.4 Выводы**

На первый взгляд, может показаться что пользовательский интерфейс не является большой и важной частью программы. Бизнес логика и базы данных куда важнее, ведь они выполняют все задачи. Но это не совсем так.

Нельзя забывать, что пользователь не видит и не хочет видеть код, базу данных и другие «внутренности». Все что он видит — это пользовательский интерфейс. Поэтому, даже если программа работает идеально, то это не будет иметь значения, если интерфейс будет неудобным и непонятным. Какая разница как быстро выполняется запрос к базе данных, если пользователь не увидит кнопку для выполнения этого запроса?

Вывод, не стоит экономить на UI. Ведь это не просто красивая обертка, но единственный способ общения между пользователем и приложением.

**Глава 3. Разработка**

**3.1 Инструменты разработки**

Для написания приложения необходимы инструменты разработки. Язык программирования, среда разработки, система контроля версий, графический редактор и многое другое.

Для работы мы выбрали среду разработки Visual Studio. На самом деле этот продукт включает в себя не только среду разработки, но и другие инструменты, например, дизайнер классов, который мы будем использовать для создания диаграммы классов. Мы выбрали этот инструмент, так как язык программировании, который мы выбрали это С#. Более того, у нас уже есть опыт использования Visual Studio.

Система контроля версий git позволяет нам более комфортно работать и не бояться серьёзных изменений программы. Более того сервис GitHub позволяет работать над приложением на любом компьютере, главное, чтобы инструменты были. Мы выбрали git, так как у нас уже есть опыт работы с ним.

Графический редактор paint.net. Мы использовали этот графический редактор так как он бесплатный и простой в использовании. Его возможностей достаточно для создания графического интерфейса, который подойдёт нашему проекту.

Приложение создано на платформе WPF с применением Entity Framework. Эти инструменты идеально подходят для создания настольного приложения с базой данных.

Для взаимодействия с документами word была использована библиотека Microsoft.Office.Interop.Word.

Для хеширования паролей было использовано пространство имён System.Security.Cryptography.

**3.2 Архитектура приложения**

Архитектура программного обеспечения (англ. software architecture)— 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.

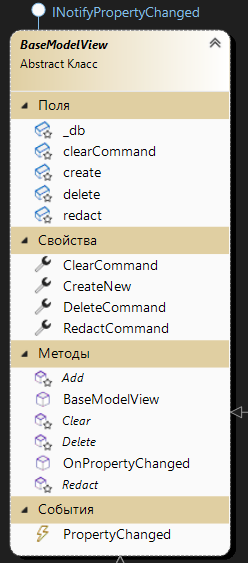


Рисунок 25 — Класс BaseModelView

Класс BaseModelView это абстрактный класс, предназначенный для наследования. Он имеет все необходимые свойства, поля, методы и события для реализации паттерна MVVM. Все методы, за исключением OnPropertyChanged абстрактные и не имеют тела. Класс реализует интерфейс INotifyPropertyChanged, необходимый для работы события PropertyChanged. Данное событие необходимо для работы паттерна MVVM.

Данный класс будет использоваться для создания классов, отвечающих за реализацию бизнес-логики с использованием паттерна MVVM. Методы Redact, Add, Delete помогут в этом.

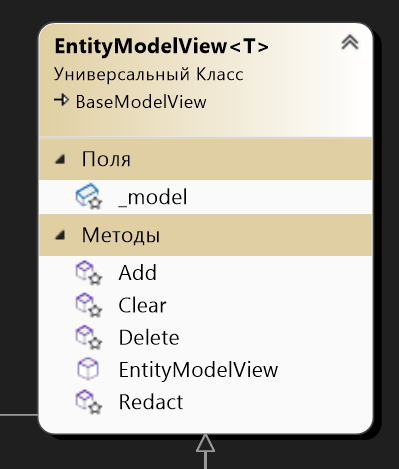


Рисунок 26 — Класс EntityModelView

Класс EntityModelView наследуется от класса BaseModleView. В отличие от BaseModleView данный класс предназначен для создания классов сущностей. Главное поле — \_model представляет собой сущность определенного объекта, абитуриента, предмета и т.д.

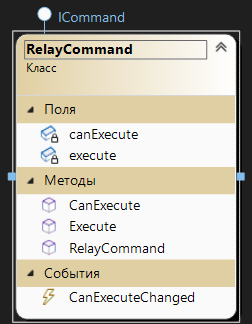


Рисунок 27 — Класс RelayCommand

Данный класс необходим для реализации патерна MVVM. Он позволит другим классам использовать паттерн Команда. Класс реализует интерфейс ICommand.

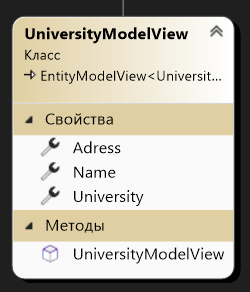


Рисунок 28 — Класс UniversityModelView

Класс UniversityModelView является моделью сущности Университет. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

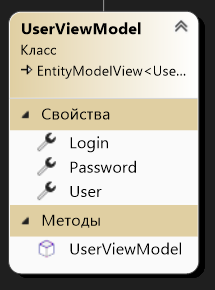


Рисунок 29 — Класс UserModelView

Класс UserModelView является моделью сущности Пользователь. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

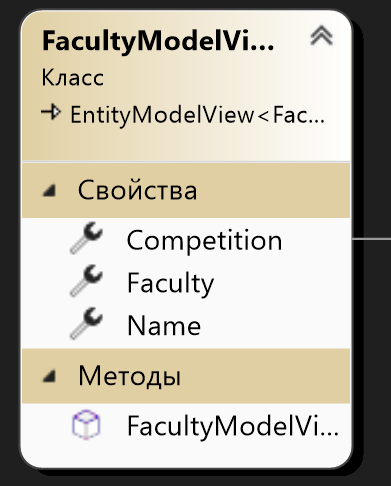


Рисунок 30 — Класс FacultyModelView

Класс FacultyModelView является моделью сущности Факультет. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

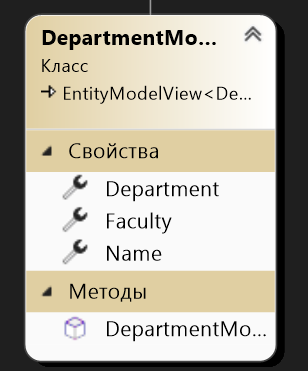


Рисунок 31 — Класс DepartmentModelView

Класс DepartmentModelView является моделью сущности Кафедра. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

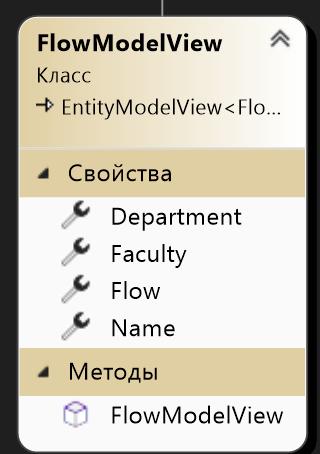


Рисунок 32 — Класс FlowModelView

Класс FlowModelView является моделью сущности Поток. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

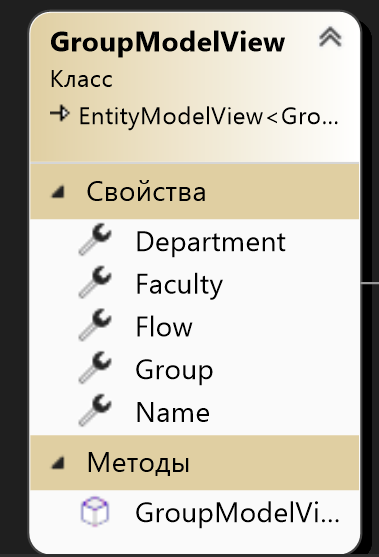


Рисунок 33 — Класс GroupModelView

Класс GroupModelView является моделью сущности Группа. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

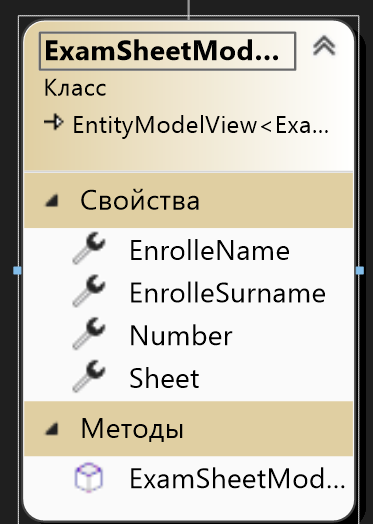


Рисунок 34 — Класс ExamSheetModleView

Класс ExamSheetModleView является моделью сущности Экзаменационный лист. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

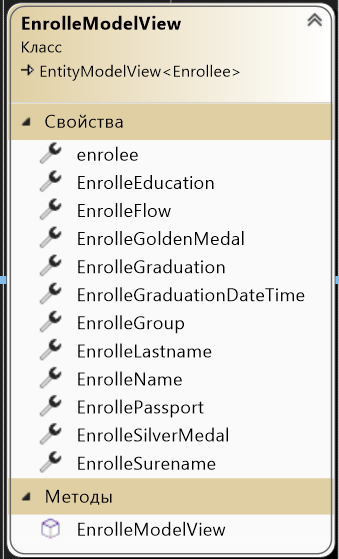


Рисунок 35 — Класс EnroleeModelView

Класс EnroleeModelView является моделью сущности Абитуриент. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

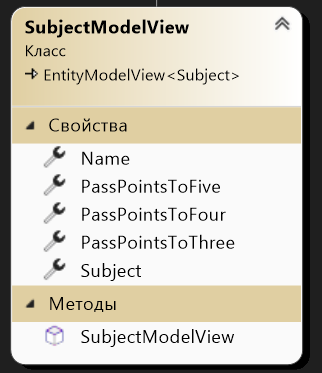


Рисунок 36 — Класс SubjectModelView

Класс SubjectModelView является моделью сущности Предмет. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

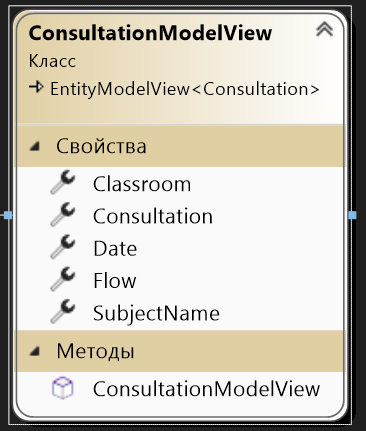


Рисунок 37 — Класс ConsultationModelView

Класс ConsultationModelView является моделью сущности Консультации. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

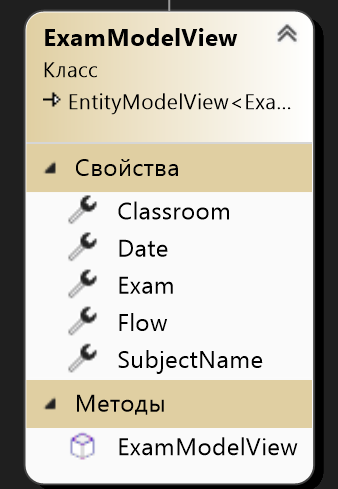


Рисунок 38 — Класс ExamModelView

Класс ExamModelView является моделью сущности Экзамены. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

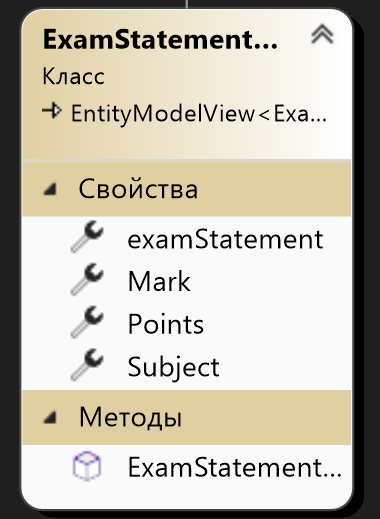


Рисунок 39 — Класс ExamStatementViewModel

Класс ExamStatementViewModel является моделью сущности Экзаменационная ведомость. Его свойства отображают атрибуты сущности из базы данных. Но только те, которые необходимы для реализации бизнес-логики.

Описанные классы начиная с UniversityModelView наследуют класс EntityModelView и являются моделями сущностей базы данных. В каждом их свойстве в разделе set вызывается метод OnPropertyChanged. Это позволяет наладить связь между интерфейсом пользователя и бизнес-логикой, не создавая сильной связи.

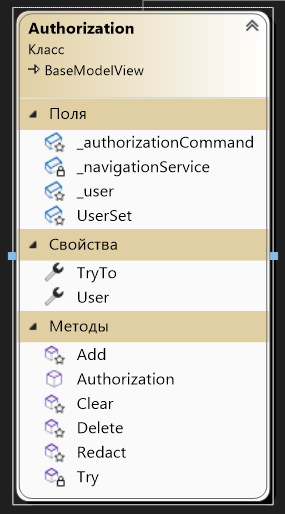


Рисунок 40 — Класс Authorization

Данный класс предназначен для авторизации пользователя. Он использует паттерн Команда. С помощью свойства User, мы общаемся с интерфейсом пользователя. Свойства поля \_user позволят нам узнать логин и пароль, введенные пользователем. Команда TryTo вызывает метод Try, который проверяет наличие логина, и если такой есть, правильность пароля. Если проверка оказывается успешной с помощью поля \_navigationService перенаправляем пользователя на главную страницу. В противном случает выводим окно с текстом ошибки.

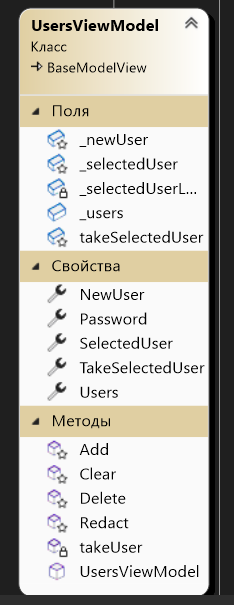


Рисунок 41 — Класс UsersViewModel

Используя паттерн Команда и свойства SelectedUser и NewUser, данный класс выполняет бизнес-логику, а именно функции по добавлению, удалению и редактированию пользователей.

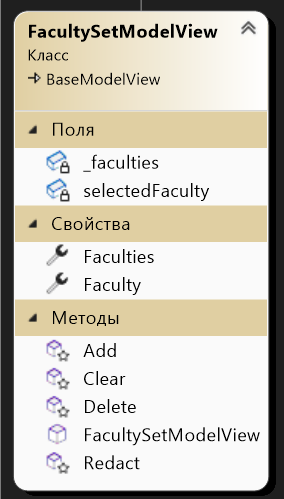


Рисунок 42 — Класс FacultySetModelView

Используя паттерн Команда и свойство Faculty, данный класс выполняет бизнес-логику, а именно функции по добавлению, удалению и редактированию факультетов.

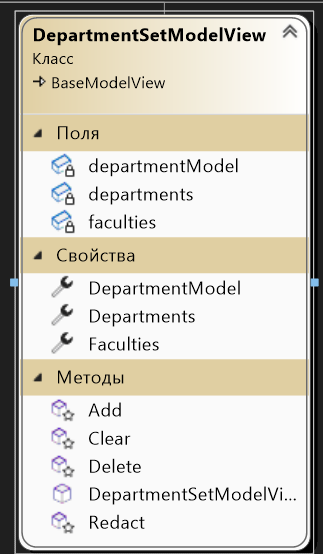


Рисунок 43 — Класс DepartmentSetModelView

Используя паттерн Команда и свойство Department, данный класс выполняет бизнес-логику, а именно функции по добавлению, удалению и редактированию кафедр.

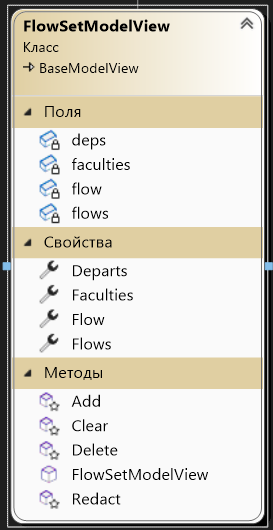


Рисунок 44 — Класс FlowSetModelView

Используя паттерн Команда и свойство Flow, данный класс выполняет бизнес-логику, а именно функции по добавлению, удалению и редактированию потоков.

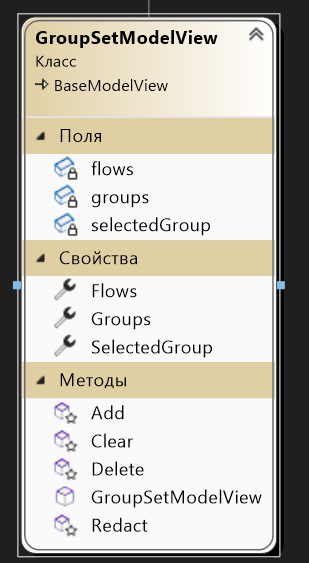


Рисунок 45 — Класс GroupSetModelView

Используя паттерн Команда и свойство Group, данный класс выполняет бизнес-логику, а именно функции по добавлению, удалению и редактированию групп.

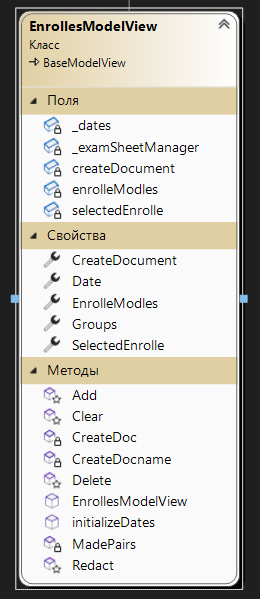


Рисунок 46 — Класс EnrollesModelView

Используя паттерн Команда и свойство SelectedEnrolle, данный класс выполняет бизнес-логику, а именно функции по добавлению, удалению и редактированию абитуриентов. Также данный класс позволяет формировать экзаменационный лист, в виде документа word, для выбранного абитуриента.

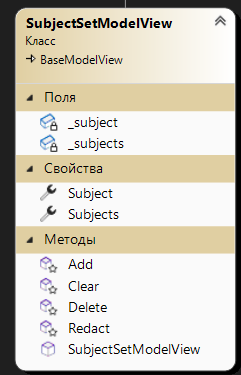


Рисунок 47 — Класс SubjectSetModelView

Используя паттерн Команда и свойство Subject, данный класс выполняет бизнес-логику, а именно функции по добавлению, удалению и редактированию предметов.

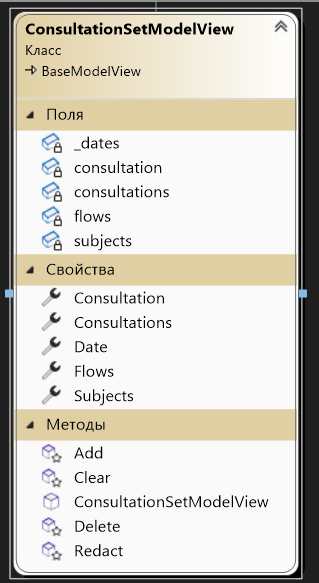


Рисунок 48 — Класс ConsultationSetModelView

Используя паттерн Команда и свойство Consultation, данный класс выполняет бизнес-логику, а именно функции по добавлению, удалению и редактированию консультаций.

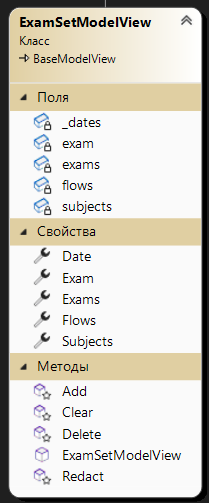


Рисунок 49 — Класс ExamSetModelView

Используя паттерн Команда и свойство Exam, данный класс выполняет бизнес-логику, а именно функции по добавлению, удалению и редактированию экзаменов.

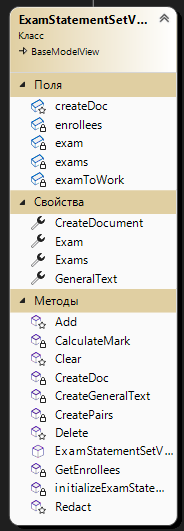


Рисунок 50 — Класс ExamStatementSetViewModel

Используя паттерн Команда и свойство Exams, данный класс позволяет создать и заполнить экзаменационную ведомость.

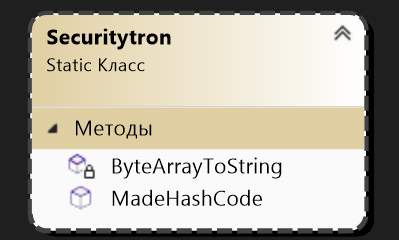


Рисунок 51 — Класс Securitytron

Этот класс предназначен для хеширования паролей пользователей. Данный класс является статическим и имеет всего один публичный метод MadeHashCode. Этот метод получает на вход строку преобразует её в массив байтов, затем для удобства переводит массив в вид шестнадцатеричной строки. Все пароли в базе данных хранятся в виде этой строки.

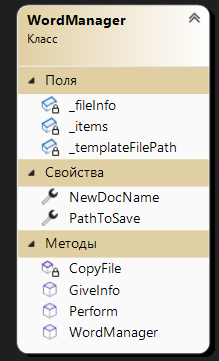


Рисунок 52 — Класс WordManager

Данный класс создан для работы с документами Word. Он получает путь к файлу, копирует его в папку указанную в свойстве PathToSave. Затем выполняет метод Perform. Данный метода обрабатывает файл.

Поле \_items является словарём и хранит в себе ключи и значения типа string. Класс получает данный словарь с помощью метода GiveInfo. Имея словарь, можно обработать копию шаблона документа заменяя ключевые слова значениями из словаря. Например в шаблоне может быть слово <University>. Метод заменит его на соответствующее значение из \_items. Таким образом можно обрабатывать шаблоны документов, заполняя их нужной информацией.

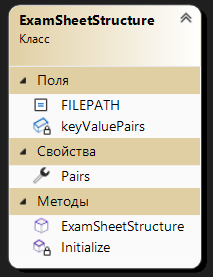


Рисунок 53 — Класс ExamSheetStructure

Данный класс хранит в себе информацию для работы с Экзаменационными листами. Константа FILEPATH хранит в себе адрес шаблона документа. Словарь keyValuePairs — набор из ключей, необходимых для заполнения документа.

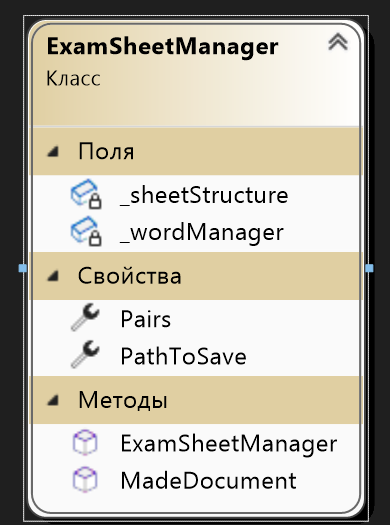


Рисунок 54 — Класс ExamSheetManager

Класс использует экземпляры классов WordManager и ExamSheetStructure. Это позволяет создавать документ Экзаменационный лист.

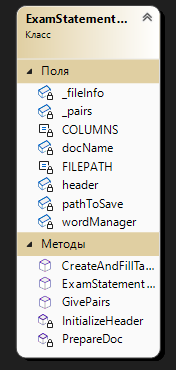


Рисунок 55 — Класс ExamStatementManager

Класс, аналогично классу ExamSheetManager, позволяет создавать документ Экзаменационную ведомость.

**3.3 Выводы**

Итак, приложение создано и выполняет поставленные задачи. Выполнение задачи можно считать успешным. Но, нет предела совершенству! Есть ещё многоие вещи, которые можно сделать.

Например, наша программа умеет создавать всего два документа: экзаменационный лист и экзаменационную ведомость. Очевидно, что есть ещё много документов, создание которых можно было бы автоматизировать.

Также, было бы неплохо отрефакторить код. Разбить некоторые классы на большее количество классов. Увеличить количество методов, для упрощения их понимания. Добавить больше комментариев. Данные действия не повлияют на работу программы, но позволят упростить модификацию программы и исправление ошибок.

Стоит поработать над дизайном интерфейса, например, использовать библиотеку Fluent.Ribbon чтобы программа выглядела более представительно.

**Список литературы**

1. [Kruchten, Philippe](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D1%80%D1%8E%D1%88%D1%82%D0%B5%D0%BD,_%D0%A4%D0%B8%D0%BB%D0%B8%D0%BF&action=edit&redlink=1). The Rational Unified Process-An Introduction, Addison-Wesley, 1998.
2. [Rumbaugh, J.](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%BC%D0%B1%D0%BE,_%D0%94%D0%B6%D0%B5%D0%B9%D0%BC%D1%81), [Jacobsen, I.](https://ru.wikipedia.org/wiki/%D0%AF%D0%BA%D0%BE%D0%B1%D1%81%D0%BE%D0%BD,_%D0%98%D0%B2%D0%B0%D1%80) and [Booch, G.](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D1%87,_%D0%93%D1%80%D0%B0%D0%B4%D0%B8) The Unified Modelling Language Reference Manual. Reading, Mass.: Addison-Wesley, 1999
3. Автоматизация докуметов word: [Электронный ресурс]. // Официальный сайт Microsoft. URL: <https://docs.microsoft.com/ru-ru/dotnet/api/microsoft.office.interop.word?view=word-pia> (Дата обращения: 01.05.2022).
4. Автоматизированная программа «Абитуриент» в Белгородском государственном университете: [Электронный ресурс]. // Журнал Управление персоналом URL: https://www.top-personal.ru/officeworkissue.html?75 (Дата обращения: 01.05.2022).
5. Бизнес аналитика: [Электронный ресурс]. URL: https://www.tadviser.ru/index.php. (Дата обращения: 01.05.2022).
6. Вычисление и сравнение хэш-значений с помощью C#: [Электронный ресурс]. // Официальный сайт Microsoft. URL: <https://docs.microsoft.com/ru-ru/troubleshoot/developer/visualstudio/csharp/language-compilers/compute-hash-values> (Дата обращения: 20.04.2022).

# Документация по Entity Framework: [Электронный ресурс]. // Официальный сайт Microsoft. URL: <https://docs.microsoft.com/ru-ru/ef/> (Дата обращения: 10.04.2022).

1. Документы: [Электронный ресурс] // Международный Институт Дизайна и Сервиса. URL: https://midis.ru/sveden/document/ (Дата обращения: 20.05.2022).
2. Использование элементов управления WPF в Office решениях: [Электронный ресурс]. // Официальный сайт Microsoft. URL: <https://docs.microsoft.com/ru-ru/visualstudio/vsto/using-wpf-controls-in-office-solutions?view=vs-2022> (Дата обращения: 20.04.2022).
3. Паттерн MVVM: [Электронный ресурс]. // Официальный сайт Microsoft. URL: https://docs.microsoft.com/en-us/windows/communitytoolkit/mvvm/introduction (Дата обращения: 20.04.2022).
4. ПОЛОЖЕНИЕ О ПРИЁМНОЙ КОМИССИИ ЧОУВО «МИДиС». — 2019.
5. Структура Белгородского государственного национального исследовательского университета: [Электронный ресурс]. // URL: https://bsuedu.ru/bsu/info/structure/ (Дата обращения: 01.05.2022).