Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**Иркутский национальный исследовательский**

**технический университет**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

Допускаю к защите

|  |  |
| --- | --- |
| Руководитель |  |
|  | подпись |
|  | З.А. Бахвалова |
|  | И.О. Фамилия |

|  |
| --- |
|  |
| Разработка прикладного программного обеспечения |
| наименование темы |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

|  |
| --- |
| Технологии разработки программных комплексов |

|  |
| --- |
| 1.006.00.00 ПЗ |
| обозначение документа |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ИСТб-19-1 |  |  |  | О.Р. Путинцев |
|  |  | шифр группы |  | подпись |  | И.О. Фамилия |
| Нормоконтроль |  |  |  |  |  | З.А. Бахвалова |
|  |  |  |  | подпись |  | И.О. Фамилия |
| Курсовая работа защищена с оценкой | | | |  | | |

Иркутск 2022 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

ЗАДАНИЕ

на курсовую работу

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| По курсу | Технологии разработки программных комплексов | | | | | | | | | | | | | | | | | |
| Студенту | Путинцеву О.Р. | | | | | | | | | | | | | | | | | |
|  | (фамилия, инициалы) | | | | | | | | | | | | | | | | | |
| Тема работы | | Разработка прикладного программного обеспечения | | | | | | | | | | | | | | | | |
| Исходные данные | | | Разработать веб-приложения для управления гостиницей | | | | | | | | | | | | | | | |
|  | | |  | | | | | | | | | | | | | | | |
| Рекомендуемая литература | | | | | | | |  | | | | | | | | | | |
| 1. Курсовая работа «Разработка прикладного программного обеспечения» Руководство и методические указания для студентов направления: 09.03.02 «Информационные системы и технологии». Составитель Бахвалова З.А. – Иркутск, Изд-во ИРНИТУ, 2018, 61 с. | | | | | | | | | | | | | | | | | | |
| Графическая часть на | | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ листах. | | | | | | | | | | | | | | |
| Дата выдачи задания | | | | « | 18 | » | февраля | | | | | 2022 г. | | | | |
|  | | | | | | | | | | | | | | | | | | |
| Задание получил | | | | | | | | |  | | | |  | | О.Р. Путинцев | | | |
|  | | | | | | | | | подпись | | | |  | | И.О. Фамилия | | | |
|  | | | | | | | | | |  |  | | |  | |  | |  |
| Дата представления работы руководителю | | | | | | | | | | « |  | | | » | |  | | 2022г. |
|  | | | | | | | | | | | | | | | | | | |
| Руководитель курсовой работы | | | | | | | | |  | | | |  | | З.А. Бахвалова | | | |
|  | | | | | | | | | подпись | | | |  | | И.О. Фамилия | | | |

# Введение

Целью курсового проекта является закрепление теоретических знаний, полученных в процессе изучения курса «Технология разработка программных комплексов» и развитие практических навыков проведения анализа и проектирования программного продукта в ходе решения конкретной практической задачи – разработки программного продукта на основе каскадной модели создания программного обеспечения, а также развитие способностей к обоснованному принятию самостоятельных решений в ходе проведения анализа и проектирования и получение базового опыта разработки программного обеспечения.

# В рамках курсового проектирования решается создания веб-приложения на основе ASP.N «Студенты» - многопользовательского приложения для учета и ведения списков студентов.

# 1 Выработка системных требований

## 1.1 Описание предметной области

При поступлении студента в университет его необходимо добавить в какую-либо группу. Сотрудники университета вручную проверял количество студентов в группах и специальность, которая будет получена в результате обучения. Далее он вручную заносил студента в подходящую группу.

Чаще всего сотрудник вносил студента в какой-то локальный документ на компьютере или физический документ. Существует вероятность, что документ будет утерян или на компьютере, на котором хранится документ, произойдет поломка.

Так же есть риск человеческого фактора, что студент может быть зачислен не в ту группу по ошибке сотрудника.

Учитывая все эти факторы было создано приложение, ускоряющее и облегчающее работу сотрудника университета.

## 1.2 Формулировка цели работы

Основной целью проекта является облегчение и ускорение добавления и редактирования списков групп студентов.

## 1.3 Постановка задачи

**Учетная запись сотрудника**

Выдается университетом. Сотрудник может войти в свою учетную запись или выйти из нее.

**Учетная запись студента**

Студент может зарегистрировать свою учетную запись, войти в нее и редактировать ее.

**Добавление студента**

После того как ВУЗ одобрил поступление студента на указанную специальность, сотрудник университета может добавить студента в одну из групп, удовлетворяющих условиям. Так же студент может сам при регистрации указать в какую группу он хочет попасть, и если это не противоречит никаким условиям, то студент будет привязан к этой группе. В группе может быть до 30 человек.

**Просмотр списков**

Каждый пользователь в состоянии просматривать списки всех студентов, групп, факультетов и специальностей.

**Фильтрация**

Пользователи могут отфильтровать просматриваемые списки по нужному критерию, например группы можно отсортировать по году поступления.

## 1.4 Функциональность ПО

**Учетная запись сотрудника**

Сотрудник может войти в приложение, используя логин и пароль, которые ему были выданы заранее, после этого он может войти в свой аккаунт. Из учетной записи можно выйти, чтобы войти в другую. Также можно поменять имя пользователя.

**Учетная запись студента**

Для входа в приложение студенту необходимо зарегистрироваться в системе. Студент должен указать пароль, ФИО, выбрать группу из списка предложенных, куда он может быть зачислен. В качестве логина студента будет использован номер зачетки, который студент получил заранее от университета.

**Добавление студента**

Сотрудник может вручную добавить какого-либо студента в список группы. Сделать это можно только если в случае, если количество человек не превышает 30. Студент сам в праве выбрать подходящую ему группу вручную, указав номер группы при регистрации.

**Просмотр списков**

Каждый пользователь в состоянии просматривать списки всех студентов, групп, факультетов и специальностей. Так же каждый сотрудник может вывести выходной документ с со списком группы, в котором будут указаны данные, которые не будут видны обычному пользователю.

**Фильтрация**

Пользователи могут отфильтровать просматриваемые списки по нужному критерию, например группы можно отсортировать по году поступления.

## 1.5 Словари

Ниже представлен словарь терминов предметной области.  
Таблица 1.1 –Словарь предметной области

|  |  |  |
| --- | --- | --- |
| № п/п | Термин | Назначение |
| 1 | Студент | авторизированный пользователь, обладает всеми возможностями обычного пользователя. |
| 2 | Группа | определенное число лиц с примерно одинаковым уровнем подготовки, изучающих одно и то же в одно и то же время под руководством одних и тех же преподавателей на протяжении одинакового для всех периода |
| 3 | Факультет | подразделение вуза, ведущее подготовку студентов по нескольким родственным специальностям. |
| 4 | Специальность | Направление подготовки в университете. |
| 5 | Средняя зарплата | Показывает среднюю зарплату на члена семьи студента |
| 6 | Номер зачетки | Уникальный номер каждого студента |
| 7 | Сотрудник университета | суперпользователь, может редактировать информацию на ресурсе. |
| 8 | Авторизация | Процесс входа в учетную запись по своим данным. Позволяет пользователю получить доступ к изменению настроек своей учетной записи, просмотр списков, редактирование данных и т. |
| 9 | Регистрация | Создание личной учетной записи. |
| 10 | ФИО декана | Полное имя заведу |
| 11 | Шифр специальности | Кодовое обозначение специальности. |
| 12 | Название факультета | Название факультета |
| 13 | Название специальности | Название специальности |
| 14 | Шифр группы | Кодовое обозначение группы |
| 15 | Год поступления | Год создания группы |
| 16 | ФИО студента | Полное имя студента |
| 17 | Зарплата отца | Размер зарплаты отца студента за месяц |
| 18 | Зарплата матери | Размер зарплаты матери студента за месяц |
| 19 | Количество членов семьи | Количество членов семьи студента |

Таблица 1.2 – Объектно-ориентированный словарь

|  |  |  |
| --- | --- | --- |
| **Класс (сущность, актер)** | **Свойство (состояние)** | **Метод (функция)** |
| Студент | ФИО, номер зачетки, зарплата отца, зарплата матери, количество членов семьи, логин и пароль | Вводить, просматривать, изменять, хранить, авторизоваться, зарегистрироваться |
| Группа | Год создания и шифр | Вводить, просматривать, изменить, хранить |
| Специальность | Название, шифр и квалификация | Вводить, просматривать, изменить, хранить |
| Факультет | Название и ФИО декана | Вводить, просматривать, изменить, хранить |
| Сотрудник университета | Логин и пароль | Авторизовать |

Таблица 1.3- Словарь предметной области по методу Аббота

|  |  |
| --- | --- |
| **Существительное** | **Глагол** |
| Система «Университет» | - |
| Студент | добавить, изменить, удалить, найти |
| группа | найти, изменить, удалить, добавить |
| Специальность | найти, изменить, удалить, добавить |
| Факультет | найти, изменить, удалить, добавить |
| Средняя зарплата студента | Посчитать |
| Номер зачетки | Сформировать, редактировать, найти |
| ФИО студента | Редактировать, найти, указать |
| Номер группы | Редактировать, указать, найти |
| Зарплата отца | Редактировать, указать, найти |
| Зарплата матери | Редактировать, указать, найти |
| Количество членов семьи | Указать, найти |
| Год поступления | Найти, указать |
| Название специальности | Указать, найти, изменить |
| Квалификация | Указать, найти, изменить |
| Название факультета | Указать, найти |
| ФИО Декана | Указать, изменить, найти |

# 2 Анализ

## Модель предметной области (IDEF0)

На вход процесса поступают: данные о группах, данные о студентах, данные для аутентификации. Процесс использует базу данных и два типа пользователя: сотрудник университета и студент. Процесс протекает по определенным правилам. На выходе: Сформированный список, выходной документ, занесенный студент.

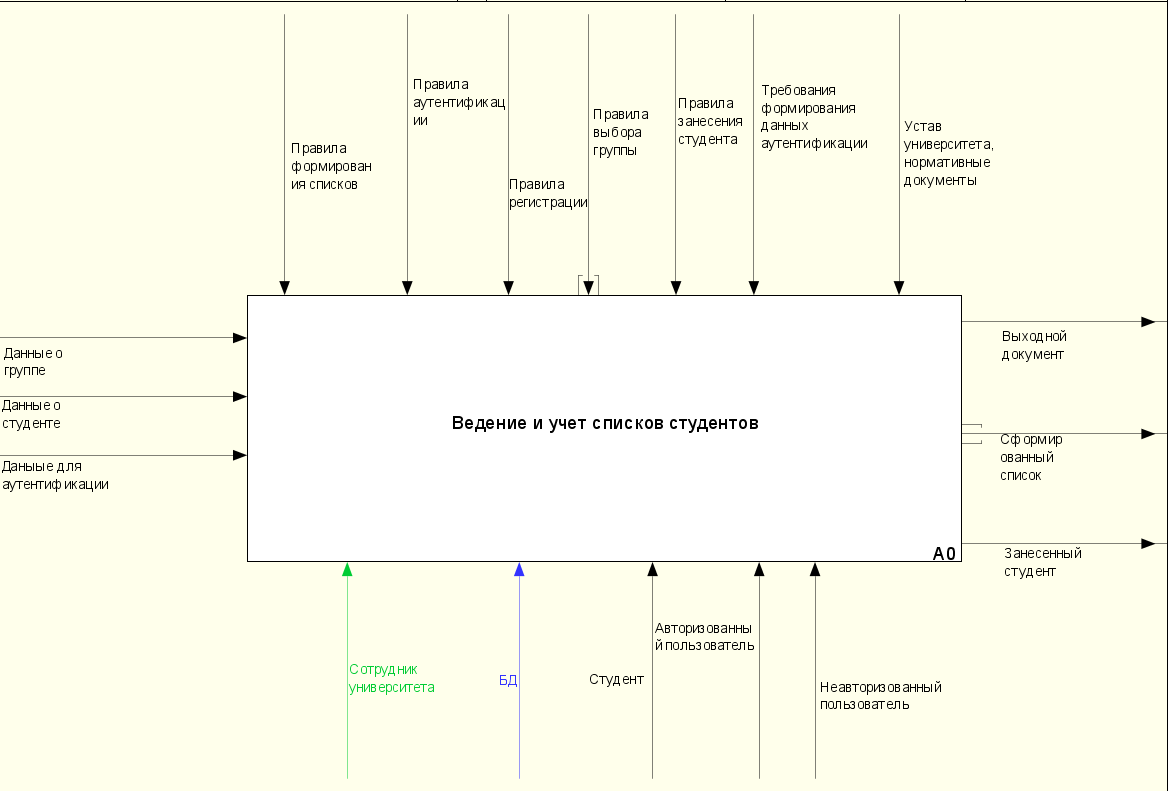


Рисунок 1 – Границы моделируемого процесса.

Для начала пользователь должен авторизоваться. После авторизации пользователь выбрать группу и посмотреть состоящих в ней студентов. Сотрудник университета может так же создать документ со списком студентов и распечатать его и добавить нового студента в список группы.

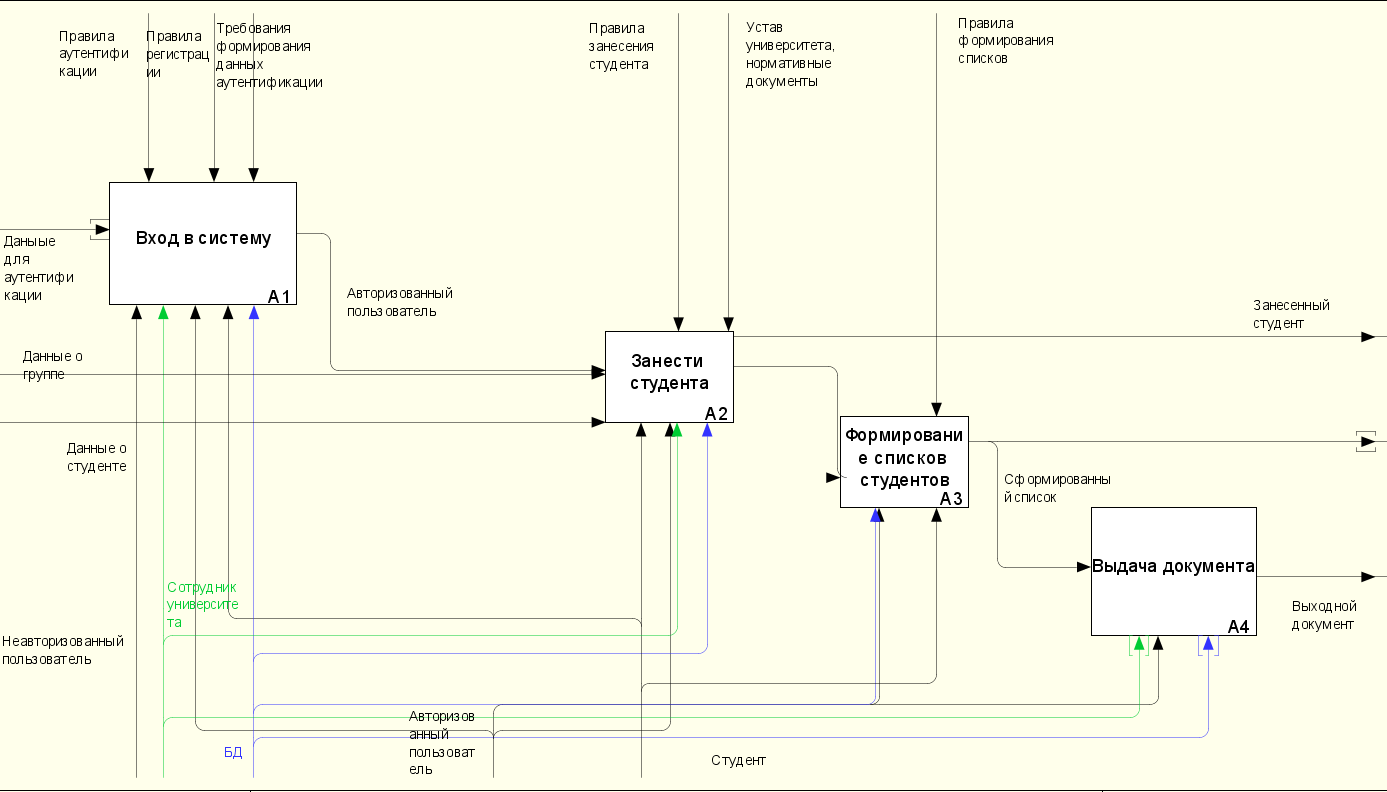


Рисунок 2 – Декомпозиция процесса «Ведение и учет списков студентов»

Если студент не зарегистрирован, то для начала ему надо будет пройти регистрацию, после этого система его автоматически авторизует и добавит в список группы, которую он указал при регистрации.

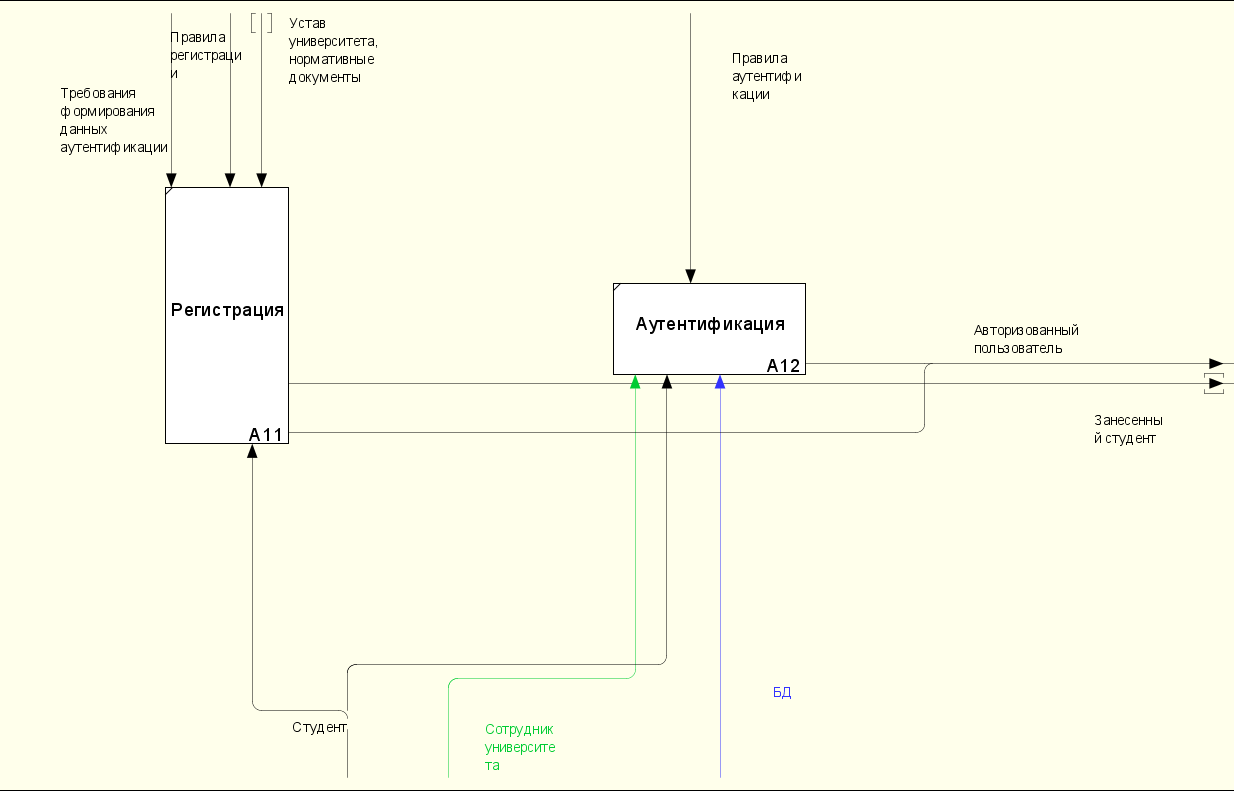


Рисунок 3 – Декомпозиция процесса «Вход в систему»

Для выбора группы пользователю нужно выбрать из списка существующих групп одну.

Для добавления студента в список группы сотрудник должен выбрать группу, в которую будет добавлять студента и ввести данные о студенте.

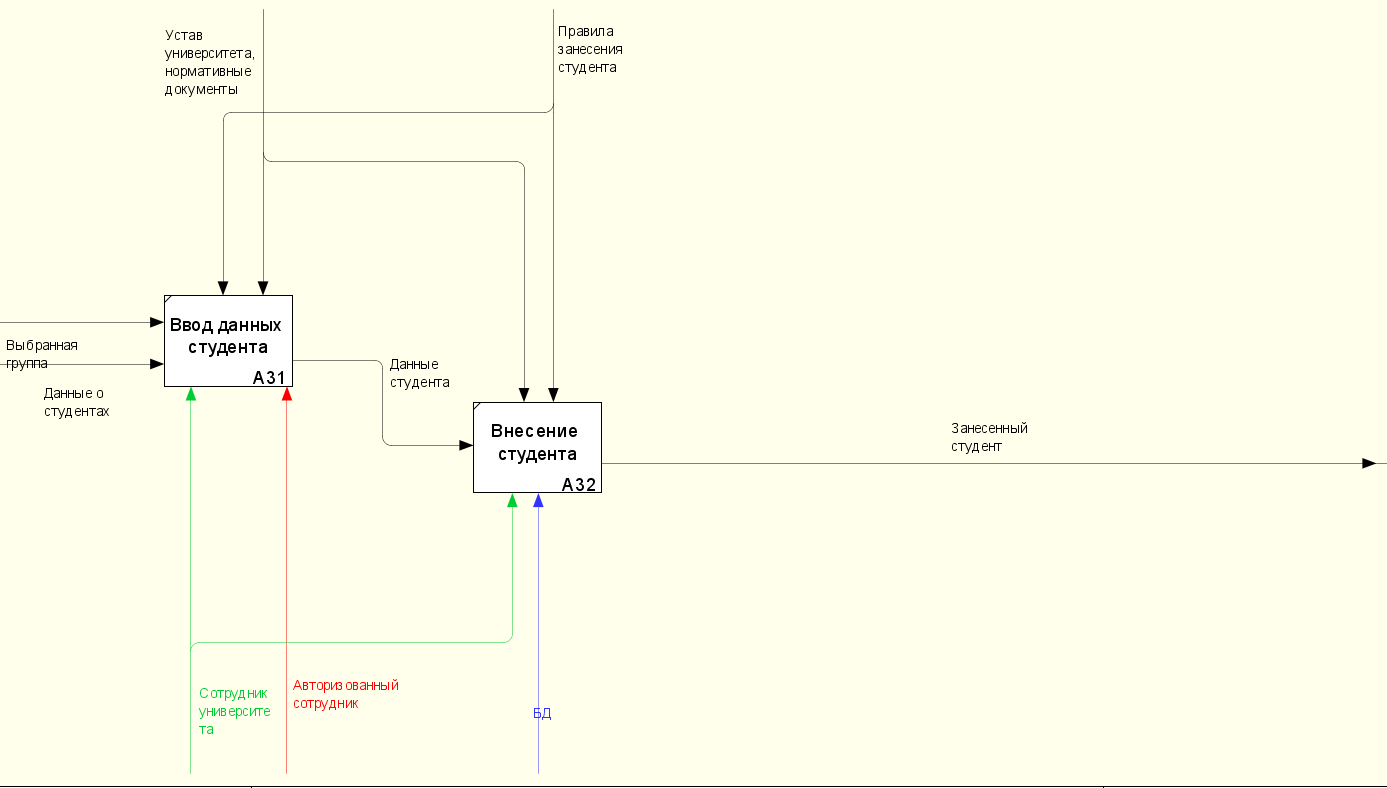


Рисунок 5 – Декомпозиция процесса «Внесение студента»

Для начала система загружает всех студентов из группы в список и далее весь список начинает сортировку по параметру, который выбрал пользователь.

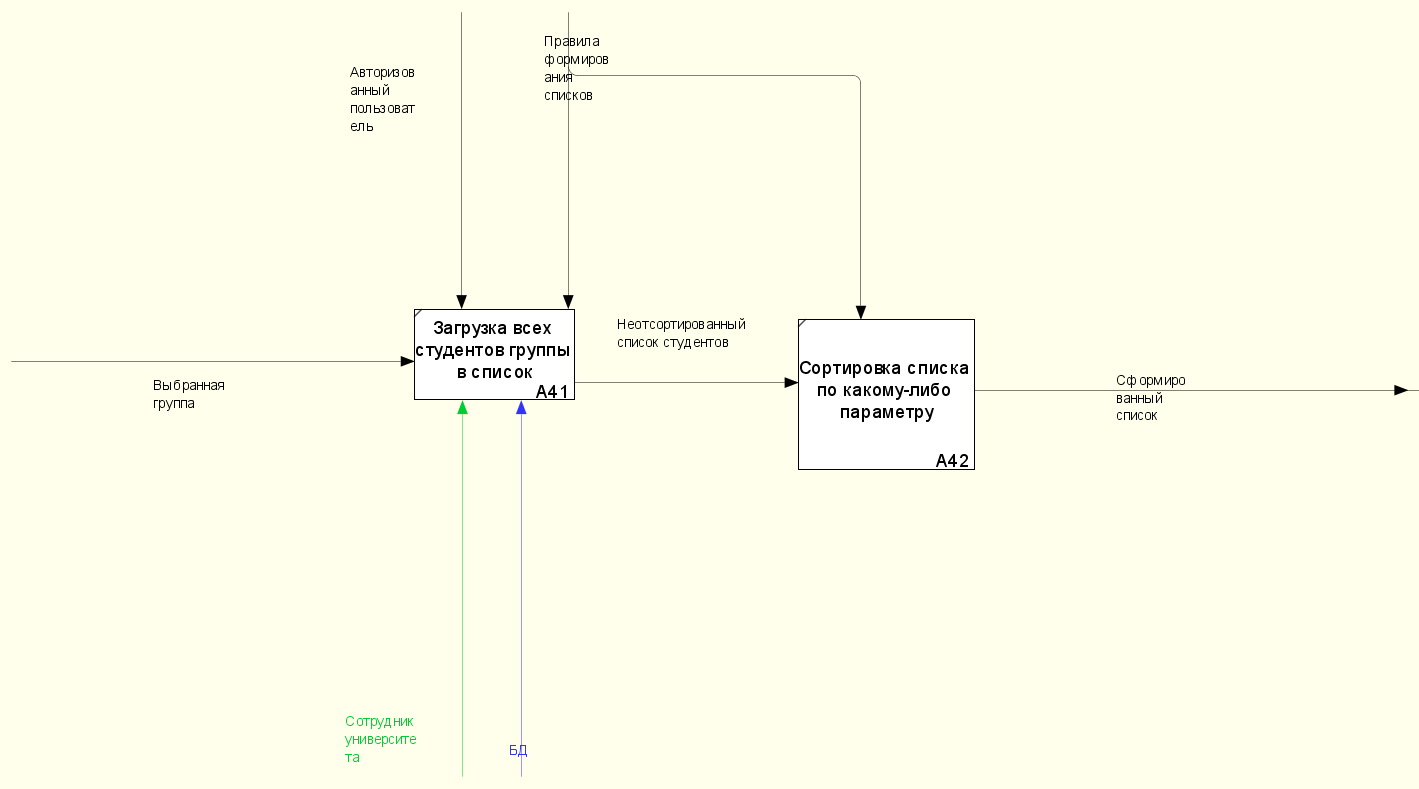


Рисунок 6 – Декомпозиция процесса «Формирование списков студентов»

Для выдачи документа пользователю нужно выбрать группу, которую он хочет просмотреть, далее отсортированный список переделывается в выходной документ, который можно распечатать.

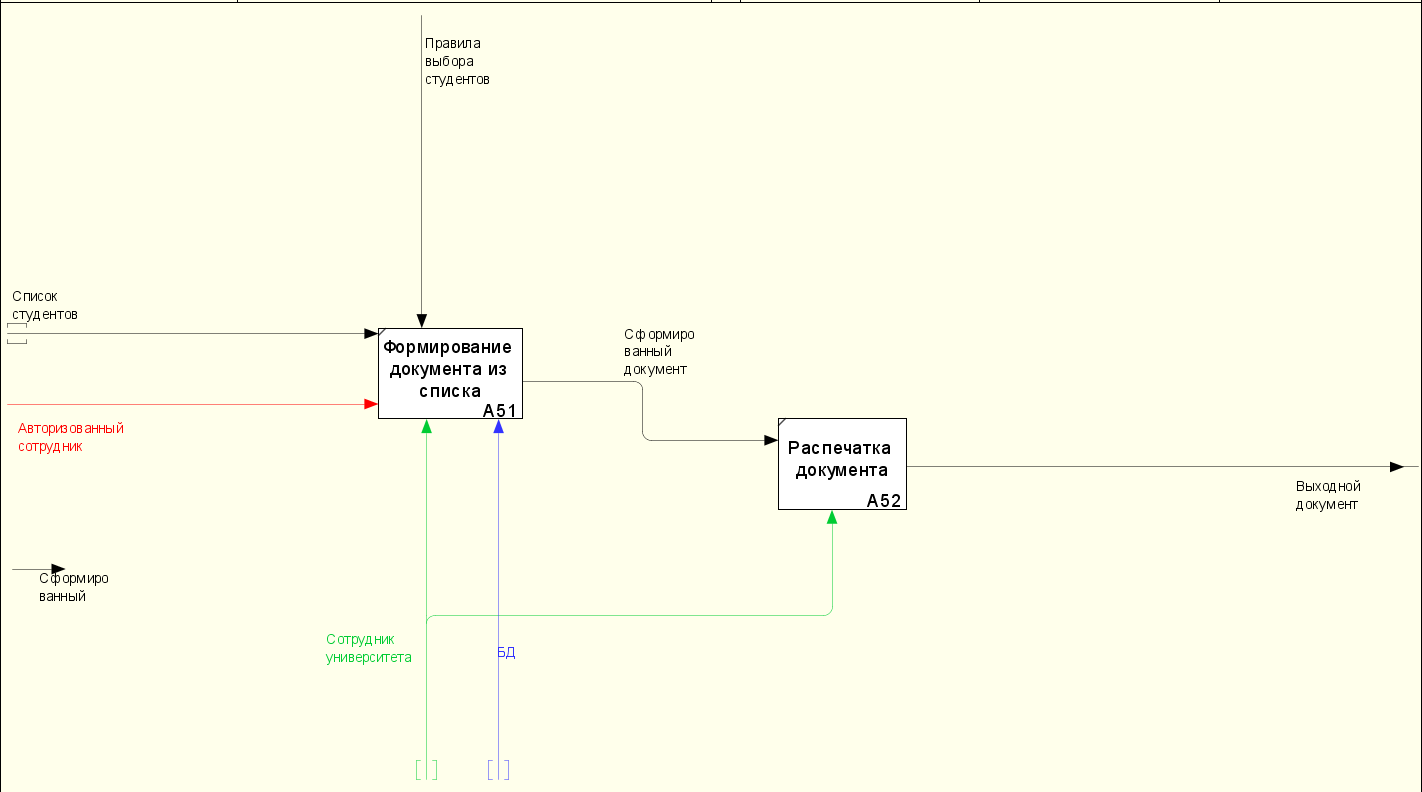


Рисунок 7 – Декомпозиция процесса «Выдача документа»

## 2.2 Диаграмма вариантов использования UML Use Case

Пользователи разделяются на два типа:

* Студент – может авторизоваться, сформировать список групп,.
* Сотрудник университета – может авторизоваться, редактировать и просмотреть информацию о студентах, редактировать списки студентов.

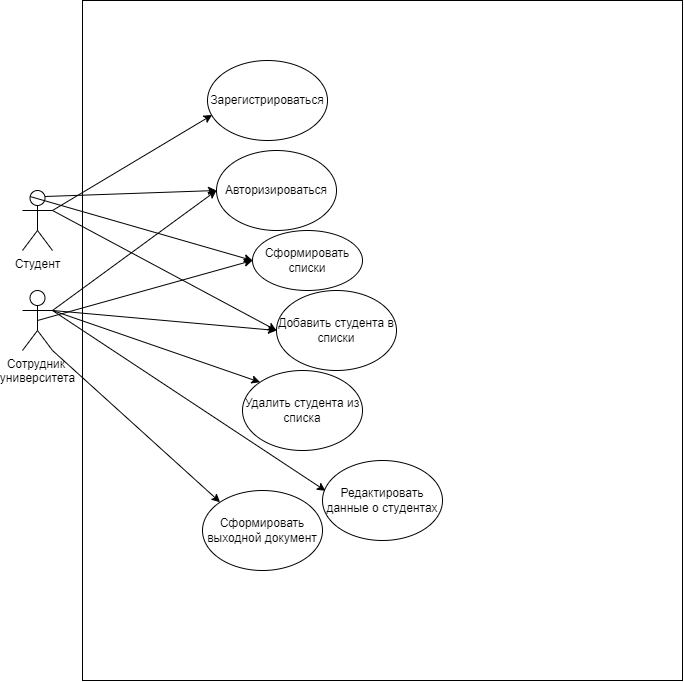


Рисунок 7 – Диаграмма функций пользователей

## 2.3 Концептуальная модель хранилища данных

Факультет характеризуется названием и именем декана.

Специальность характеризуется названием, шифром и названием факультета.

Группа характеризуется шифром, годом поступления и названием специальности

Студент характеризуется ФИО, шифром группы, номером зачетки, зарплатой отца, зарплатой матери, количеством членов семьи.

Пользователь характеризуется типом (Студент или Сотрудник), логином и паролем

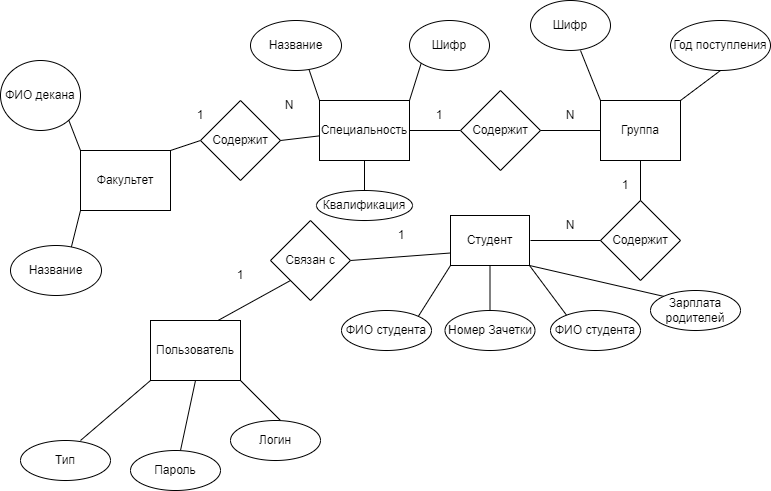


Рисунок 8 – Концептуальная модель хранилища данных

**2.4 DFD - диаграмма потоков данных**

Имеется одна сущность – пользователь. Он может передать свои данные авторизации, данные для добавления и редактирования, запрос на просмотр списков.

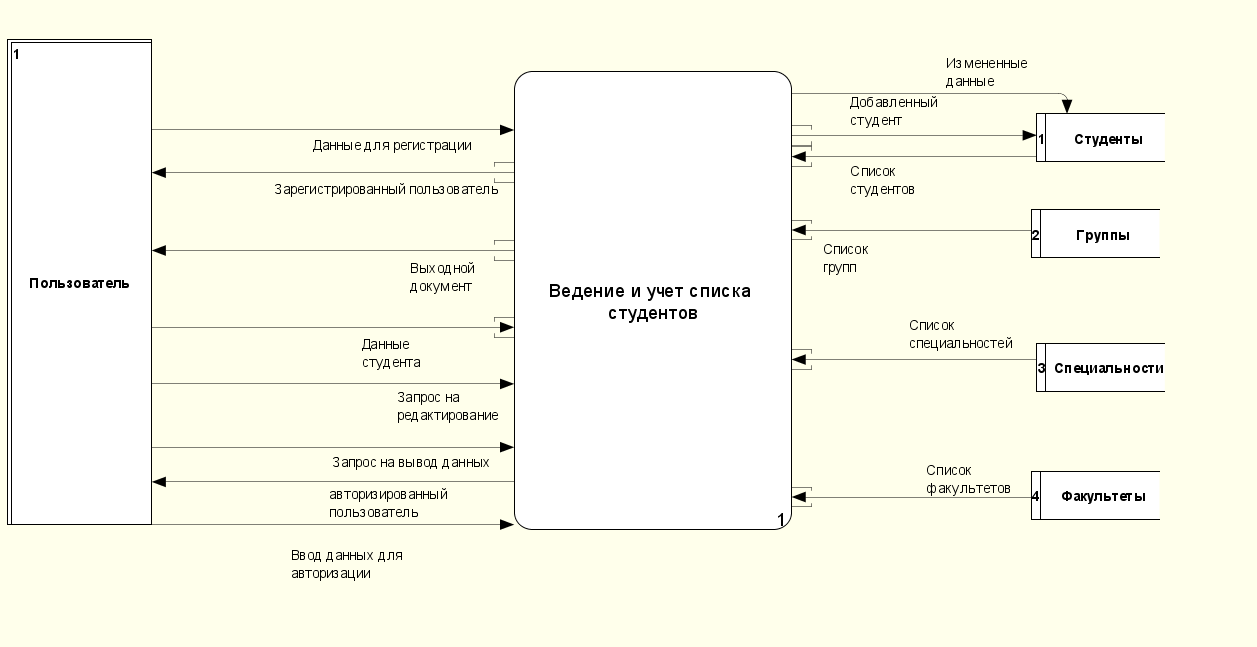


Рисунок 9 - Диаграмма потоков данных

От пользователя могут пойти данные авторизации в процесс «Авторизация», где будут браться данные из «Пользователи» для проверки. Потом уже данные авторизованного пользователя возвращаются обратно к пользователю.

Для создания студентов от пользователя поступают «Данные для добавления», после чего дело сохраняется в «Студенты». Данные из «Студенты» отправляются в процессы «Вывод данных о студенте» и «Вывод списков». На выходе из процесса «Вывод данных о студенте» будет «Подробная информация о студенте», который интересует пользователя. В процесс «Вывод списков» поступает название группы, которую выберет пользователь и выводятся все студенты, состоящие в ней.

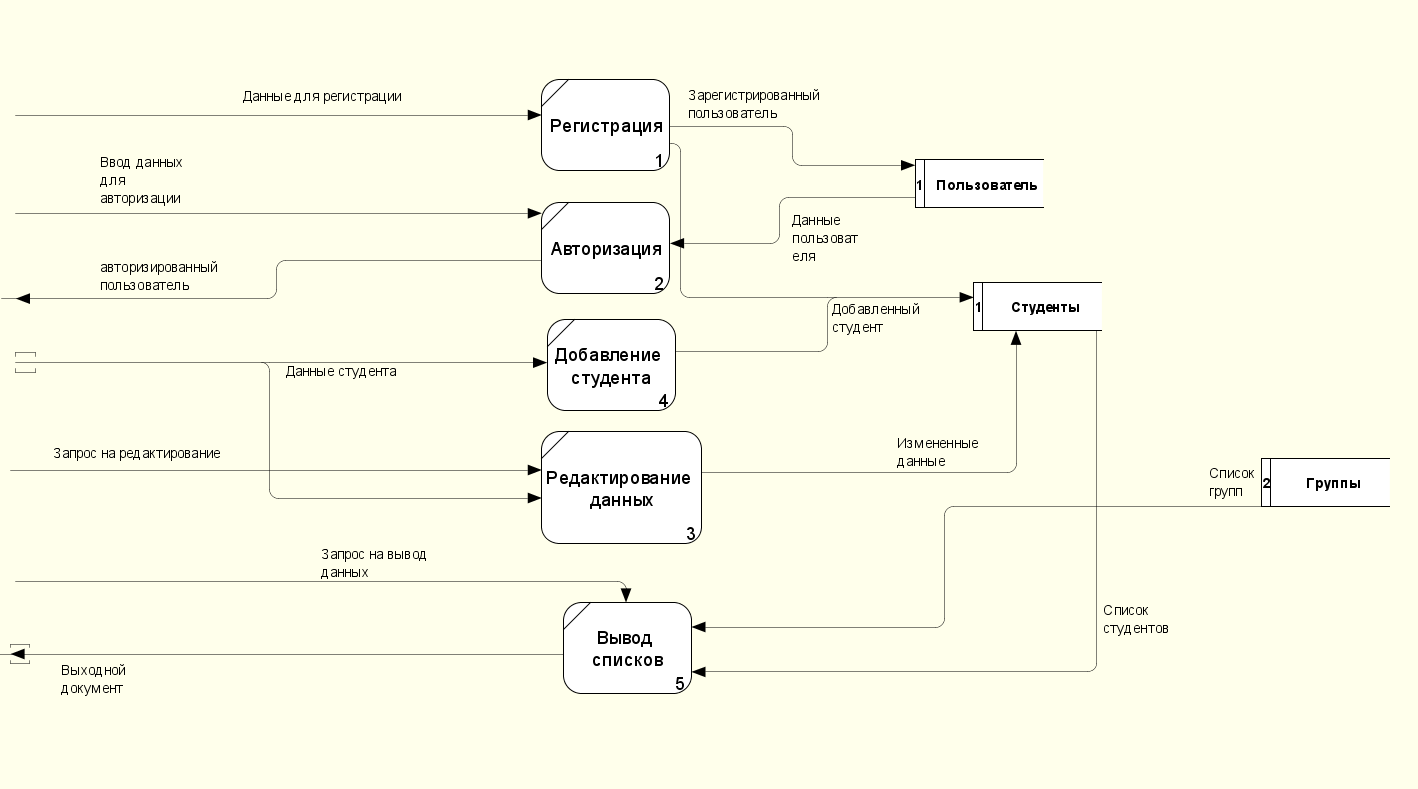


Рисунок 10 – Декомпозиция процесса «Ведение и учет списка студентов»

## 2.5 Описание вариантов использования

### **2.5.1. Спецификация варианта использования «Создать дело».**

**Цель:** добавить информацию о больнице.

**Активные субъекты:** Главврач

**Краткое описание:** Главврач может добавить информацию о больнице.

**Основной поток событий:**

1. Пользователь инициирует добавление информации
2. Пользователь заполняет группу, ФИО студента, зарплату отца, зарплату матери, количество членов семьи.
3. Пользователь подтверждает добавление
4. Система формирует запрос
5. Система показывает результат
6. Система завершает работу варианта использования.

**Сценарий обработки ошибок:**

1. Не удалось добавить информацию - пользователю показывается уведомление об ошибке: “Информация не добавлена”

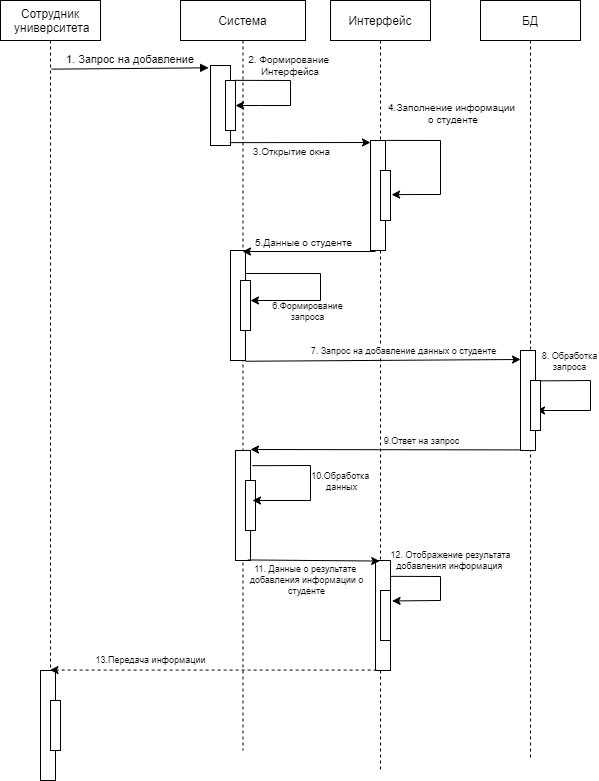


Рисунок 14 – Диаграмма процесса «Добавить информацию о студенте»

## 2.6 Описание интерфейса

Шапка сайта позволяет перейти на три вкладки: страница пользователя, лекарства в больнице и лекарства в аптеке.

После авторизации пользователь попадает на страницу пользователя. Откуда может изменить свои данные или удалить аккаунт. Из личного кабинета пользователь может просмотреть информацию о лекарствах в своем учреждении.

На страницу просмотра лекарств может попасть пользователь из личного кабинета. На странице находятся поля для названия, фирмы лекарства, а также для указания количество и их стоимости. Также есть возможность удалить или изменить данные о определенном лекарстве.

Страница подачи заявления позволяет подать заявку на получение лекарств, попасть на нее можно только в том случае, если при авторизации был указан тип пользователя «Главврач». На странице необходимо указать адрес больницы, выбрать необходимые лекарства и указать их количество.

На страницу проверки заявления можно попасть только если тип пользователя — это «Фармацевт». На странице указан адрес больницы, а также список необходимых ей лекарств.

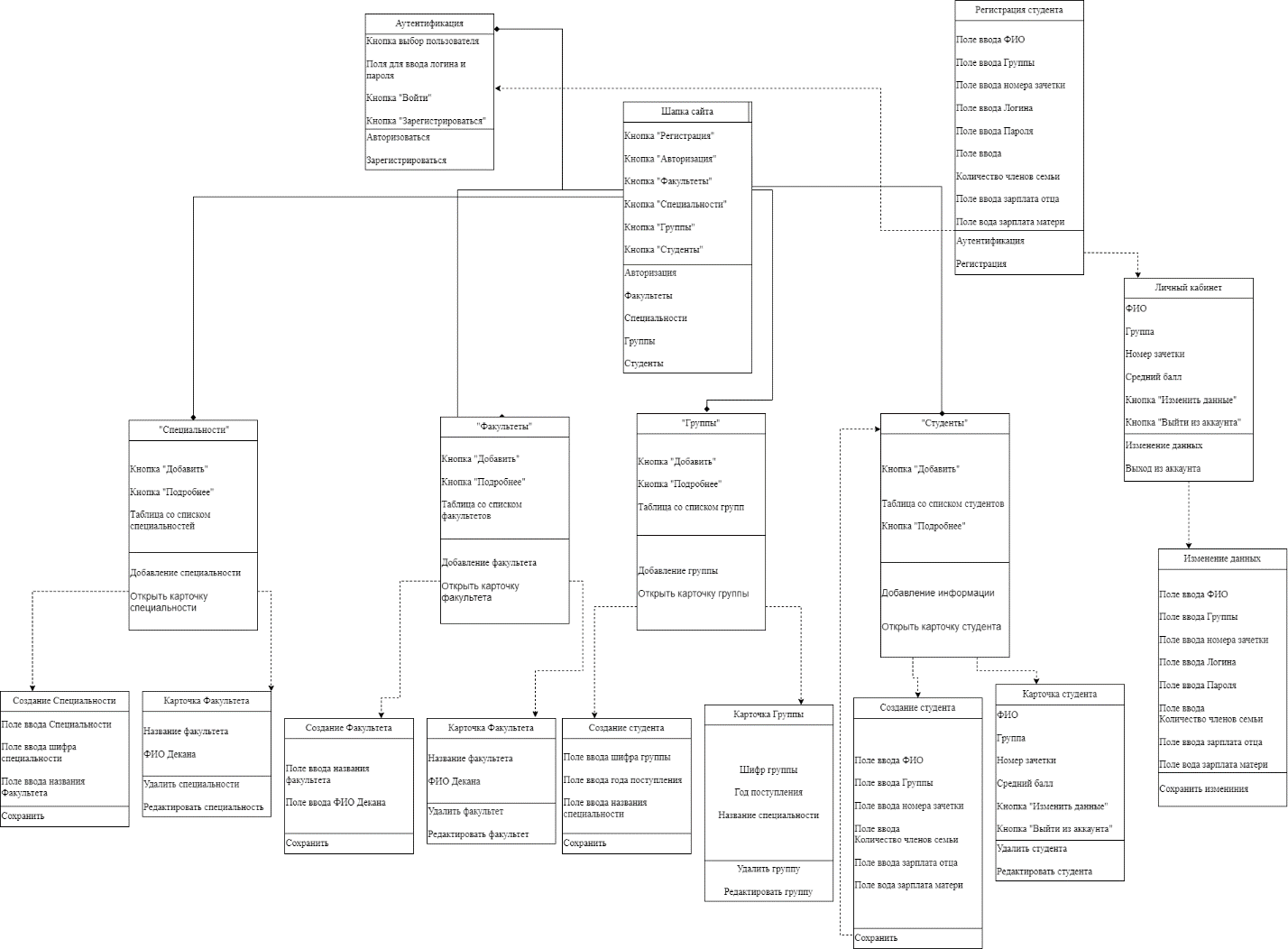


Рисунок 15 – Диаграмма интерфейсных классов

Таблица 3 – Описание элементов интерфейсов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Страница | Функция | Элемент интерфейса | Описание элемента | Ссылка на другую страницу |
| Шапка сайта | Осуществелять переходы между основными страницами | Кнопка «Авторизоваться» | Открытие страницы «Авторизация» | Страница авторизации |
| Кнопка «Факультеты» | Открытие страницы «Факультеты» | Факультеты |
| Кнопка «Специальности» | Открытие страницы «Специальности» | Специальности |
| Кнопка «Группы» | Открытие страницы «Группы» | Группы |
| Кнопка «Студенты» | Открытие страницы «Студенты» | Студенты |
|  |  | Кнопка «Авторизация» | Открытие страницы для авторизации | Авторизация |
|  |  | Кнопка «Регистрация» | Открытие страницы для регистрации | Регистрация |
| Аутентификация | Авторизация пользователя | Кнопка выбора типа пользователя | Выбор типа пользователя | - |
| Кнопка «Войти» | Инициирование входа | Страница пользователя |
| Поле ввода для пароля | Ввод данных | - |
| Поле ввода для логина | Ввод данных | - |
| Регистрация | Регистрация пользователя | Поля для ввода данных | Ввод данных |  |
| Кнопка «Регистрация» | Инициирование регистрации и переход в личный кабинет | Личный кабинет |

# Продолжение таблицы 3 – Описание элементов интерфейсов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Личный кабинет | Просмотр информации и возможность ее изменения | Кнопка «Изменить» | Кнопка для перехода на страницу редактирования | Изменение информации |
| Поля с информацией о пользователе | Поля, отображающие основную информацию о пользователе | - |
| Изменение информации | Изменение информации | Редактируемые текстовые поля с информацией | Поля для изменения информации о пользователе | - |
| Кнопка «Изменить» | Инициирование изменения информации |  |
| Факультеты | Добавление факультета | Кнопка «Добавить» | Открывает окно для добавления факультета | Добавление факультета |
| Открыть карточку факультета | Кнопка «Подробнее» | Открывает окно с информацией о факультете | Карточка факультета |
| Добавление факультета | Добавление факультета | Кнопка «Добавить» | Инициализирует добавление | - |
| Поля для заполнения | Данные для добавления |
| Карточка факультета | Просмотр данных | Поля с информацией о факультете | Данные о факультете | - |
| Изменение | Кнопка «Изменить» | Инициализирует изменение | - |
| Удаление | Кнопка «Удалить» | Удаляет факультет | - |
| Специальности | Добавление Специальности | Кнопка «Добавить» | Открывает окно для добавления Специальности | Добавление Специальности |
| Открыть карточку Специальности | Кнопка «Подробнее» | Открывает окно с информацией о Специальности | Карточка Специальности |

Продолжение таблицы 3 – Описание элементов интерфейса

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Добавление Специальности | Добавление Специальности | Кнопка «Добавить» | Инициализирует добавление | - |
| Поля для заполнения | Данные для добавления |
| Карточка Специальности | Просмотр данных | Поля с информацией о Специальности | Данные о Специальности | - |
| Изменение | Кнопка «Изменить» | Инициализирует изменение | - |
| Удаление | Кнопка «Удалить» | Удаляет Специальность | - |
| Группы | Добавление Группы | Кнопка «Добавить» | Открывает окно для добавления Группы | Добавление Группы |
| Открыть карточку Группы | Кнопка «Подробнее» | Открывает окно с информацией о Группы | Карточка Группы |
| Добавление Группы | Добавление Группы | Кнопка «Добавить» | Инициализирует добавление | - |
| Поля для заполнения | Данные для добавления |
| Карточка Группы | Просмотр данных | Поля с информацией о Группы | Данные о Группы | - |
| Изменение | Кнопка «Изменить» | Инициализирует изменение | - |
| Удаление | Кнопка «Удалить» | Удаляет Группу | - |
| Студенты | Добавление Студенты | Кнопка «Добавить» | Открывает окно для добавления Студента | Добавление Студента |
| Открыть карточку Студенты | Кнопка «Подробнее» | Открывает окно с информацией о Студента | Карточка Студента |
| Добавление Студента | Добавление Студента | Кнопка «Добавить» | Инициализирует добавление | - |
| Поля для заполнения | Данные для добавления |

Продолжение таблицы 3 – Описание элементов интерфейса

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Карточка Студента | Просмотр данных | Поля с информацией о Группы | Данные о Студенте | - |
| Изменение | Кнопка «Изменить» | Инициализирует изменение | - |
| Удаление | Кнопка «Удалить» | Удаляет Студента | - |

# 3 Проектирование программного продукта

## 3.1 Выбор и обоснование программных инструментов

Для построения диаграмм этапа анализа были использованы такие ПО, как Ramus, draw.io и DBDesigner. Ramus использовалась для построение диаграмм процессов в нотации IDEF0 и DFD. Draw.io, в свою очередь, использовалась как для построения вариантов использования, так и для описания хранилища данных в нотации Чена. А DBDesigner – для описания физической модели данных. Выбранные программы полностью покрывают соответствующие нужды и выполняют свою работу на отлично.

Для разработки экранов приложения был выбран онлайн-сервис Figma, обладающий большим функционалом для данной задачи.

В качестве среды разработки приложения была выбрана Visual Studio, Интегрированная среда разработки Visual Studio — является официальной средой для разработки на С# от компании Microsoft.

Для разработки приложения был выбран язык С#, который был разработан компанией Microsoft. Причина выбора заключается в том, что язык прост в использовании, а также, из-за большого разнообразия синтаксических конструкций C# позволяет быстрее, чем любой другой язык, разрабатывать программные решения.

Для работы с базой данных был выбран Entity Framework, так как он позволяет взаимодействовать с СУБД с помощью сущностей (entity), а не таблиц, также код с использованием EF пишется гораздо быстрее.

## 3.2 Проектирование хранилища данных

Как и в нотации Чена, в логической и физической моделях данных были представлены все, описанные на этапе постановки задачи, сущности.

Рисунок 16 – Логическая модель хранилища данных

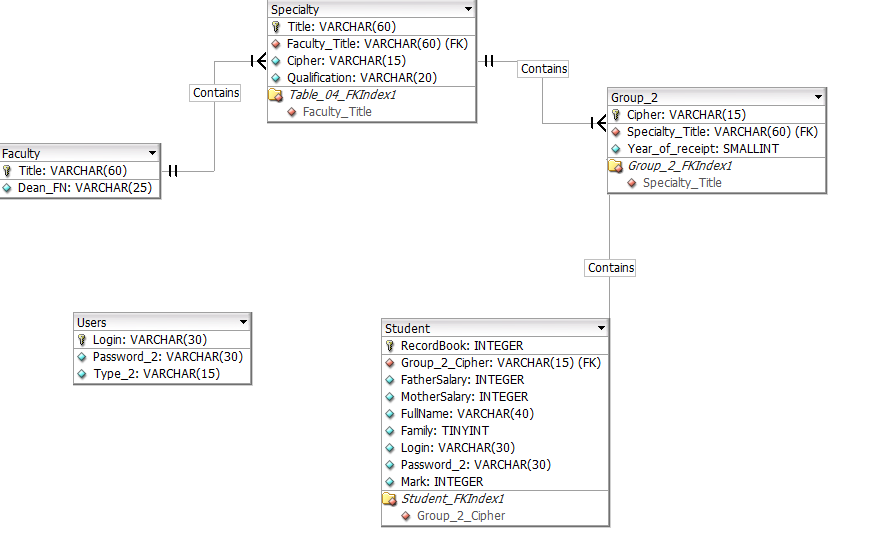


Рисунок 17 – Физическая модель хранилища данных

## 3.3 Проектирование классов разрабатываемой системы

Для реализации, поставленной на этапе анализа, задачи, требуется разработать веб приложение. Для реализации был выбран фреймворк ASP.NET MVC является многофункциональной платформой для создания веб-приложений и API-интерфейсов с помощью структуры проектирования Model-View-Controller.

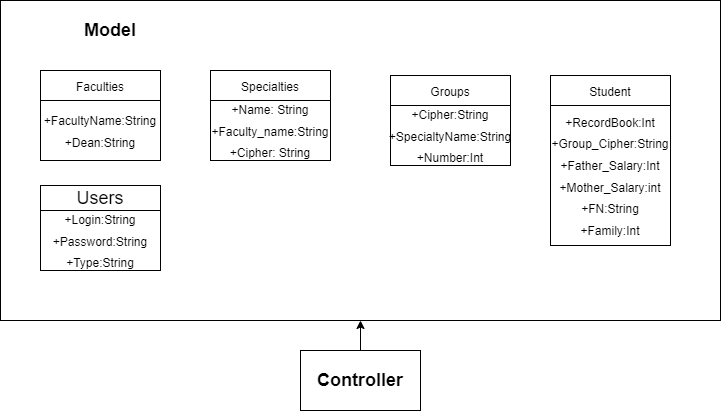


Рисунок 18 – Диаграмма классов пакета Model

Пакет Model реализует работу с базой данных, то есть хранение и доступ к данным.

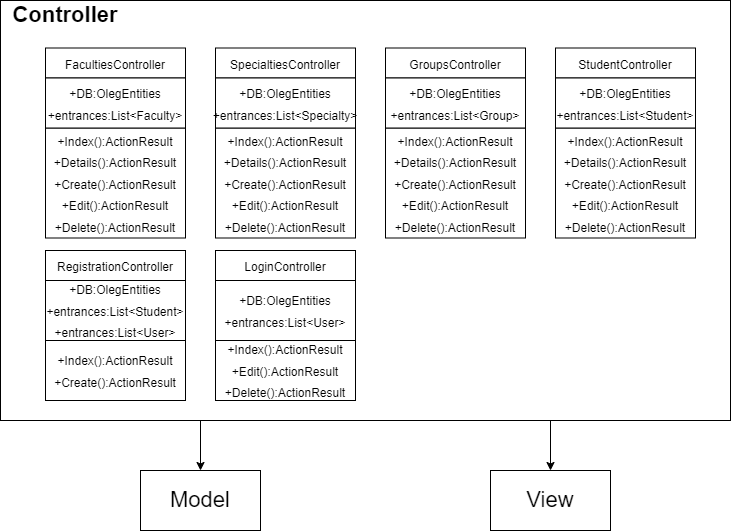
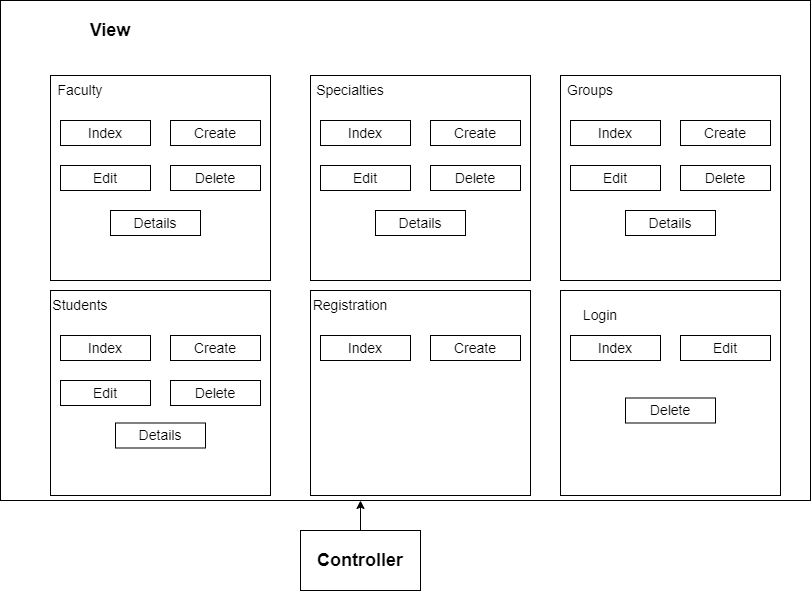


Рисунок 18 – Диаграмма классов пакета Controller

Пакет Controller реализует взаимодействие пакетов Model и View. Controller обрабатывает действия пользователя и затем обновляет Model или View. Если пользователь взаимодействует с приложением, контроллер получает уведомление об этих действиях и решает, что с ними делать.



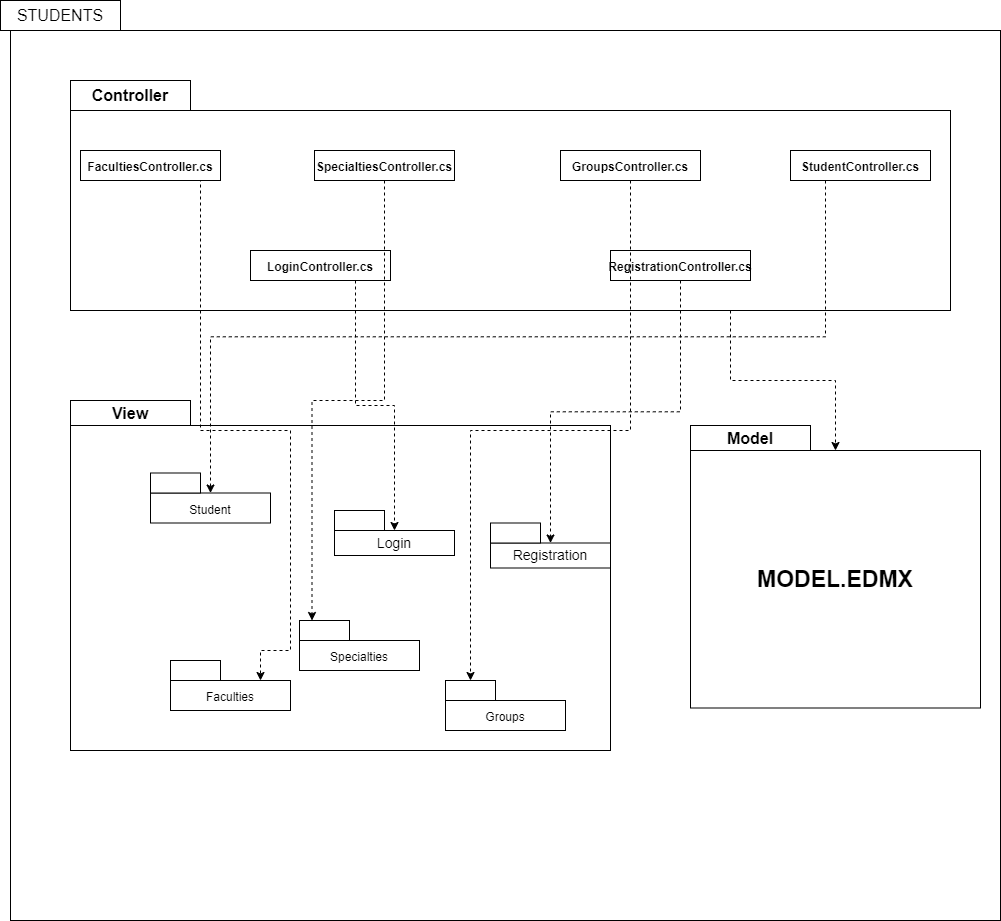
Пакет View отвечает за отображение данных пользователю. Все, что видит пользователь, генерируется этим пакетом.

## 3.5 Фактическая организация модулей системы

На рисунке 22 показана вложенность пакетов и их связью

В проекте находятся три основных пакета: Controller, Models и View.

Controller перехватывает событие извне и в соответствии с заложенной в него логикой, реагирует на это событие изменяя Model, посредством вызова соответствующего метода. После изменения Model использует событие о том что она изменилась, и все подписанные на это события View, получив его, обращаются к Model за обновленными данными, после чего их и отображают.

  
Рисунок 20 – Модель развертывания продукта

## 3.6 Модель развертывания продукта

Так как это веб-приложение, то для его работы понадобится клиент с браузером и сервер. Клиент для запуска приложения в браузере, а сервер для хранения базы данных.

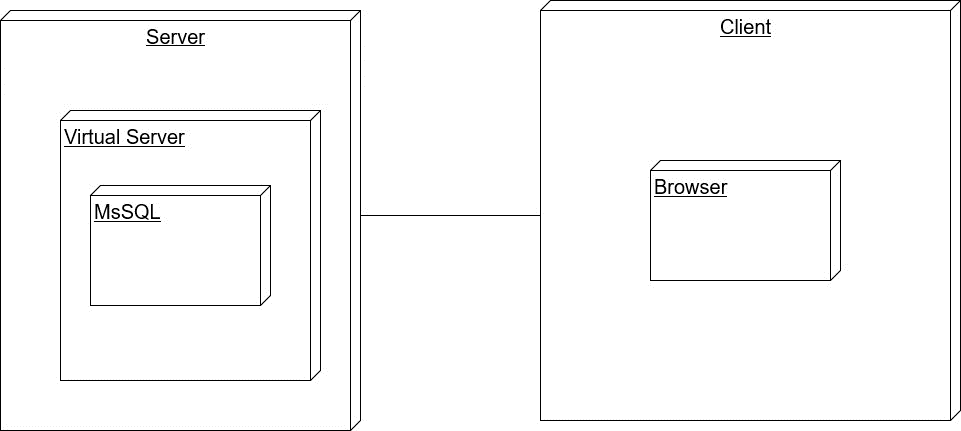


Рисунок 21 – Модель развертывания продукта

## 3.7 Макеты пользовательских интерфейсов

Первым окном пользователь видит окно авторизации.

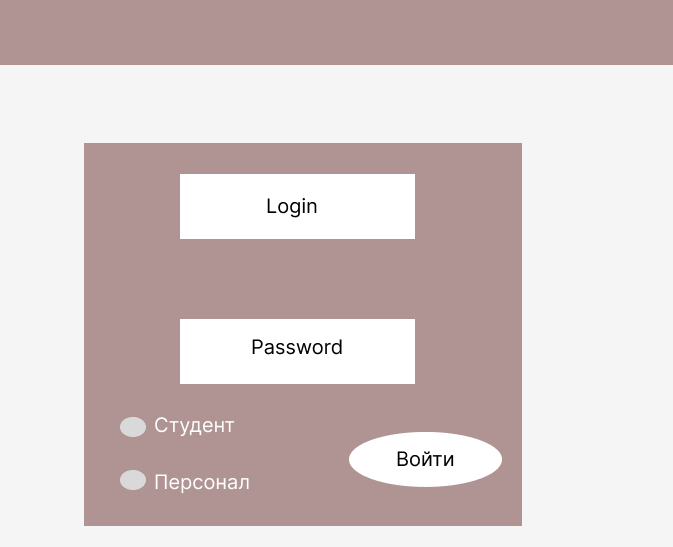
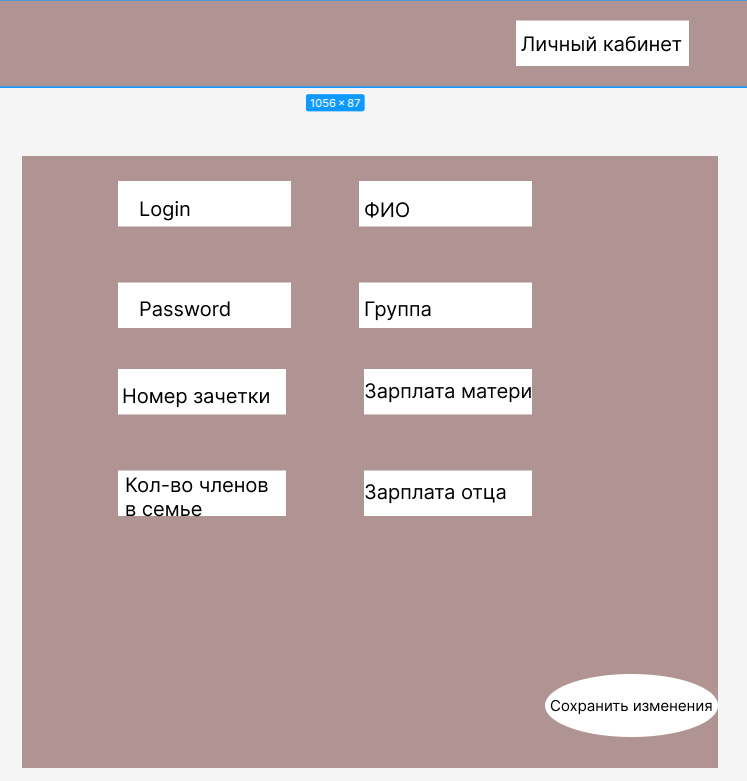


Рисунок 24 – Макет страницы «Авторизация»



Рисунок 25 – Макет страницы «Страница регистрации»

После авторизации или регистрации пользователь попадает на свою личную страницу, которая зависит от типа пользователя. На данных страницах размещена основная информация о пользователе, также есть возможность изменить аккаунт или удалить его.

  
Рисунок 26 – Макет страницы «Страница изменения аккаунта»

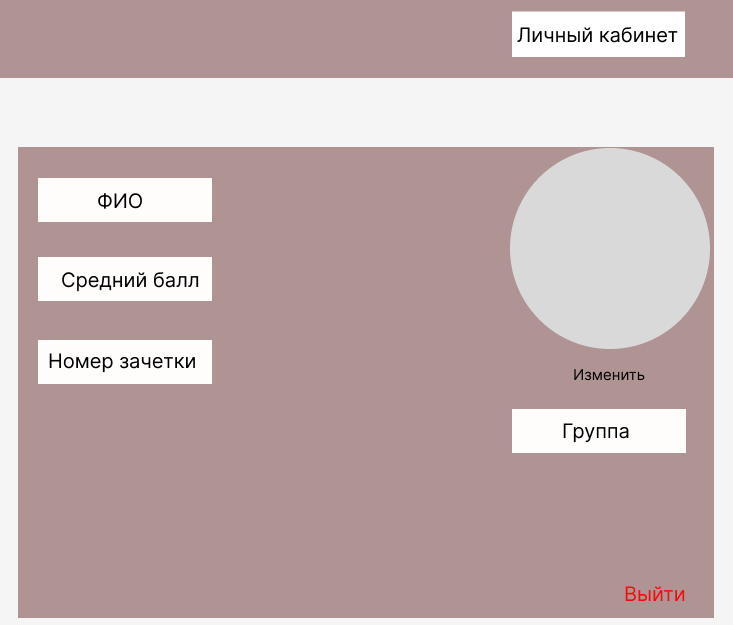


Рисунок 27 – Макеты страниц «Личный кабинет»

Страницы «Карточка факультета», «Факультеты» и «Добавление факультета» по своему оформлению и строению идентичны страницам с такими же функциями, поэтому описывать отдельно каждую не стоит.

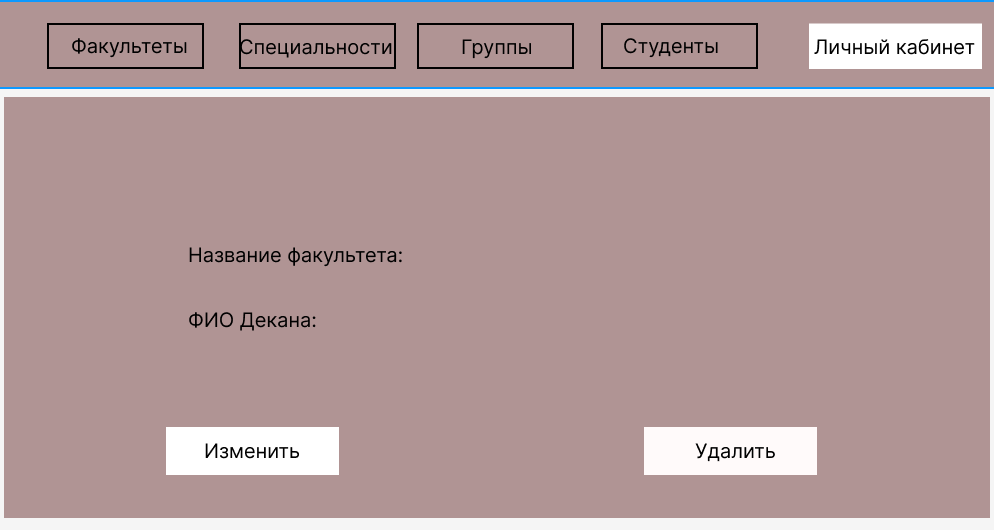


Рисунок 28 – Макет страницы «Карточка факультета»



Рисунок 29 – Макеты страницы «Факультеты»

На экранах добавления размещены выпадающие списки, а также текстовые поля для ввода необходимой информации.

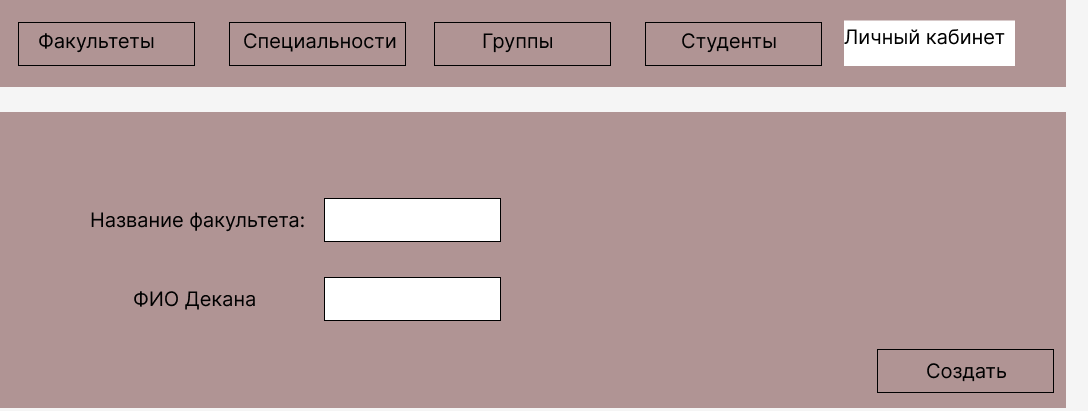


Рисунок 32 – Макеты страниц добавления

# 4 Программная реализация

## Описание классов данных

Для выполнения задач, связанных с базой данных были реализованы классы и контроллеры для каждой таблицы.

Был создан класс Faculty

public partial class Faculty

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Faculty()

{

this.Specialties = new HashSet<Specialty>();

}

[Required(ErrorMessage = "Ожидается")]

public string Title { get; set; }

[Required(ErrorMessage = "Ожидается")]

public string Dean\_FN { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Specialty> Specialties { get; set; }

}

}

И контроллер для этого класса: FacultiesController

public class FacultiesController : Controller

{

private OlegEntities db = new OlegEntities();

// GET: Faculties

public ActionResult Index(int pg=1)

{

List<Faculty> entrances = db.Faculties.ToList();

const int pageSize = 40;

if (pg < 1)

pg = 1;

int rescCount = entrances.Count();

var pager = new Pager(rescCount, pg, pageSize);

int recSkip = (pg - 1) \* pageSize;

var data = entrances.Skip(recSkip).Take(pager.PageSize).ToList();

this.ViewBag.Pager = pager;

return View(data);

}

// GET: Faculties/Details/5

public ActionResult Details(string id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Faculty faculty = db.Faculties.Find(id);

if (faculty == null)

{

return HttpNotFound();

}

return View(faculty);

}

// GET: Faculties/Create

public ActionResult Create()

{

return View();

}

// POST: Faculties/Create

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Title,Dean\_FN")] Faculty faculty)

{

try

{

if (ModelState.IsValid)

{

//db.Faculties.Add(faculty);

db.addFaculty(faculty.Title, faculty.Dean\_FN);

db.SaveChanges();

return RedirectToAction("Index");

}

}

catch (Exception ex)

{

if (ex.InnerException == null)

{

ViewBag.error = ex.Message;

}

else

{

ViewBag.error = ex.InnerException.Message;

}

ModelState.AddModelError("", "Ошибка! Такой факультет уже существует");

}

return View(faculty);

}

// GET: Faculties/Edit/5

public ActionResult Edit(string id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Faculty faculty = db.Faculties.Find(id);

if (faculty == null)

{

return HttpNotFound();

}

return View(faculty);

}

// POST: Faculties/Edit/5

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в разделе https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Title,Dean\_FN")] Faculty faculty)

{

if (ModelState.IsValid)

{

db.Entry(faculty).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(faculty);

}

// GET: Faculties/Delete/5

public ActionResult Delete(string id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Faculty faculty = db.Faculties.Find(id);

if (faculty == null)

{

return HttpNotFound();

}

return View(faculty);

}

// POST: Faculties/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(string id)

{

Faculty faculty = db.Faculties.Find(id);

try

{

db.Faculties.Remove(faculty);

db.SaveChanges();

return RedirectToAction("Index");

}

catch (Exception ex)

{

ModelState.AddModelError("", "Ошибка! Объект уже используется в другой таблице!");

}

return View(faculty);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

Эти данные были произведены для всех сущностей таблицы, за исключением пользователя. Весь исходный код выложен в [репозитории](file:///C:\Users\oleg.pytin\Downloads\KursovayaBolnitsyAVTOR%20(2).docx#_Список_литературы) онлайн-сервиса Github[4].

# Заключение

В результате курсовой работы было разработано приложение, содержащее одну часть системы – ведение и учет студентов в университете. Для его разработки были решены следующие задачи:

1. описана постановка задачи предметной области;
2. проведен анализ задуманной информационной системы;
3. проведено проектирование программного продукта

В процессе разработки программного обеспечения были получены следующие навыки:

1. реализация веб клиента с помощью фреймворка ASP.NET MVC;
2. для работы с базой данных был использован Entity Framework;
3. были реализованы, основные операции CRUD;
4. установка экземпляра сервера в SQL Server Management Studio.

# Список литературы

* 1. ASP.NET Core MVC. Введение в MVC [сайт], URL: https://metanit.com/sharp/aspnet5/3.1.php (дата обращения: 08.05.2022)
  2. Отношения классов — от UML к коду [Электронный ресурс] // Хабрахабр: [сайт]. [2006]. URL: http://habrahabr.ru/post/150041/ (дата обращения: 10.05.2022)
  3. Библиотека Entity Framework documentation [сайт]. URL: <https://docs.microsoft.com/en-us/ef/> (Дата обращения: 12.05.2022)
  4. Репозиторий с исходных кодом проекта [Электронный ресурс] // GitHub: [сайт]. URL: https://github.com/kagenetzu/Hospitals
  5. ASP.NET | Open-source web framework for .NET. [официальный сайт] URL: https://dotnet.microsoft.com/en-us/apps/aspnet/ (дата обращения: 10.05.2022)