Министерство транспорта Российской Федерации  
Федеральное агентство железнодорожного транспорта  
ФГБОУ ВО «Дальневосточный государственный университет путей   
сообщения»

Кафедра «Информационные технологии  
и системы»

Курсовой проект

по дисциплине «Методы проектирования информационных систем»  
Тема: «Курьерская служба»

Выполнил: Красовский В. А.  
студент гр.СО251КОБ

Проверил: Анисимов В.В.

Хабаровск

2020 г.

Оглавление

[1. Описание предметной области 3](#_Toc51612163)

[2. Модель вариантов использования 4](#_Toc51612164)

[2.1 Диаграммы вариантов использования 5](#_Toc51612165)

[2.2 Диаграммы автоматов 10](#_Toc51612166)

[3. Модель анализа 14](#_Toc51612167)

[3.1 Диаграмма классов анализа 14](#_Toc51612168)

[3.2 Диаграммы последовательности 18](#_Toc51612169)

[3.3 Диаграммы коммуникации 21](#_Toc51612170)

[4. Модель проектирования 24](#_Toc51612171)

[4.1 Диаграммы классов 24](#_Toc51612172)

[4.2 Диаграммы деятельности 30](#_Toc51612173)

[5. Модель реализации 35](#_Toc51612174)

[5.1 Диаграммы компонентов 35](#_Toc51612175)

[5.2 Диаграмма развертывания 39](#_Toc51612176)

[6. Сгенерированный программный код 40](#_Toc51612177)

[7. Руководство пользователя 41](#_Toc51612178)

[7.1 Авторизация 41](#_Toc51612179)

[7.2 Регистрация 42](#_Toc51612180)

[7.3 Главная страница 43](#_Toc51612181)

[7.4 Справочные материалы 45](#_Toc51612182)

[7.5 Построение схем 45](#_Toc51612183)

[7.6 Просмотр статистики 47](#_Toc51612184)

[7.7 Администрирование 47](#_Toc51612185)

[Заключение 49](#_Toc51612186)

[Список используемых источников 50](#_Toc51612187)

# 1. Описание предметной области

Целью данного курсового проекта является разработка информационной системы «Курьерская служба».

В свою очередь основной целью создания системы является: автоматизация человеческой деятельности и электронный документооборот компании.

Требования заказчика к информационной системе:

* Пользователи должны иметь доступ к приложению средствами браузера (Google Chrome, Mozilla Firefox, Microsoft Edge и другие.)
* Кроссплатформенность (macOS, Windows, Linux);
* Простота в развертывании и администрировании системы;
* Электронный документооборот;
* Отдельные учётные записи и права доступа для каждого пользователя, исходя из его должности;

На основе требований заказчика платформой для информационной системы было выбрано веб-приложение с использованием архитектуры MVC. Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Технологии задействованы в работе информационной системы:

* Django;
* PostgreSQL.

# 2. Модель вариантов использования

Модель вариантов использования — это модель, описывающая взаимодействие пользователей и системы между собой, для решения поставленных задач. Данная модель описывает цели пользователей, поведение системы, а также взаимодействие пользователей с системой или между собой.

Главной целью при разработке этой модели является достижение максимального взаимопонимания между разработчиками и заказчиками по вопросам назначения, возможностям и технологиям использования ИС.

Достижение этой цели, в первую очередь, достигается за счет разработки диаграмм UML, которые являются основными факторами технологического процесса «Формирование требований».

Унифицированный язык моделирования (UML) в настоящий момент является стандартом де-факто при описании (документирования) результатов проектирования и разработки объектно-ориентированных систем.

Диаграммы, которые включает в себя модель вариантов использования:

* диаграмма вариантов использования;
* диаграмма автоматов.

## 2.1 Диаграммы вариантов использования

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Диаграмма вариантов использования включает в себя три основных элемента: актеров, варианты использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Актером — это любой объект, субъект или система, взаимодействующая с моделируемой системой извне.

Вариант использования — это спецификация сервисов (функций), которые система предоставляет актеру.

В процессе анализа проектируемой информационной системы было определено четыре актера:

1. Менеджер — работник компании, который оформляет договора с клиентами, регистрирует посылки и создаёт заявки на доставку.
2. Логист – работник компании, который просматривает заявки на доставку и, на их основе, составляет оптимальный маршрут для курьера.
3. Курьер – работник компании, который по маршрутному листу доставляет посылки получателям и оформляет отчёты о доставке.
4. Администратор — сотрудник отдела программного и технического обеспечения, который следит за работоспособностью системы, управлением техническими настройками, занимается непосредственной работой с информационной системой.

Так же определены следующие варианты использования:

1. Регистрация пользователей;
2. Редактирование пользователя;
3. Авторизация;
4. Составление договора о доставке;
5. Изменение договора о доставке;
6. Регистрация посылки;
7. Изменение посылки;
8. Создание заявки на доставку;
9. Изменение заявки на доставку;
10. Просмотр списка заявок;
11. Создание маршрутного листа;
12. Редактирование маршрутного листа;
13. Просмотр маршрутного листа;
14. Создание отчёта о доставке.

На основе перечисленных данных была построена контекстная диаграмма, описывающая общую схему взаимодействия актеров в пределах ИС (рис. 2.1.1).



Рисунок 2.1.1. Контекстная диаграмма вариантов использования

Основываясь на контекстную диаграмму, были построены три диаграммы декомпозиций.

Обычно в центре диаграммы декомпозиции располагается декомпозируемый вариант использования, а вокруг – входящие в него обязательные (««include»») или расширяющие (««extend»») составные части.

Рассмотрим декомпозицию варианта использования «Регистрация посылки» (рис. 2.1.2), на которой в качестве «центрального» актера выступает «Менеджер». В данном разделе пользователь может выполнять следующие действия:

1. Ввод характеристик посылки;
2. Ввод данных получателя;
3. Ввод данных отправителя;
4. Выбор адреса доставки:
   1. Получение списка улиц.

Для выполнения каких-либо манипуляций в рамках данной диаграммы пользователю необходимо успешно авторизоваться в информационной системе.

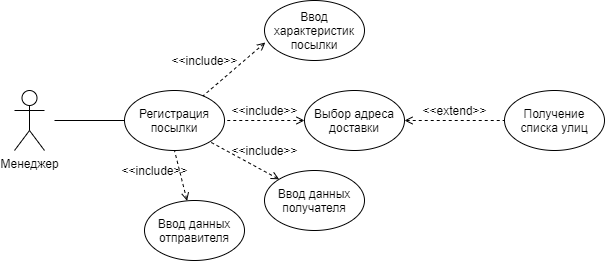


Рисунок 2.1.2. Диаграмма декомпозиции "Регистрация посылки"

На диаграмме декомпозиции варианта использования «Создание маршрутного листа» (рис. 2.1.3) показаны основные возможности пользователя в данном разделе:

1. Добавление заявки на доставку в маршрутный лист
   1. Выбор заявки из списка;
   2. Получение информации о заявке;
2. Удаление заявки из маршрутного листа;
3. Выбор курьера:
   1. Получение информации о курьере;
4. Изменение статуса маршрутного листа;

Для выполнения каких-либо манипуляций в рамках данной диаграммы пользователю необходимо успешно авторизоваться в информационной системе.

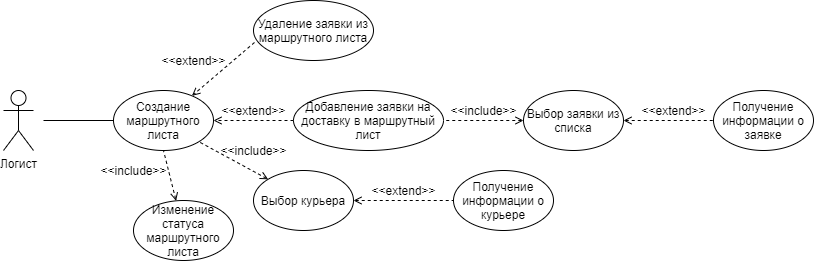


Рисунок 2.1.3. Диаграмма декомпозиции "Создание маршрутного листа"

На диаграмме декомпозиции варианта использования «Регистрация пользователя» (рис. 2.1.4) показаны основные возможности пользователя в данном разделе:

1. Ввод должности пользователя;
2. Ввод логина пользователя:
   1. Проверка уникальности логина;
3. Ввод пароля пользователя;
4. Добавление прав доступа:
   1. Просмотр прав доступа
   2. Получения данных о правах доступа
   3. Ввод ФИО пользователя

Для выполнения каких-либо манипуляций в рамках данной диаграммы пользователю необходимо успешно авторизоваться в информационной системе.

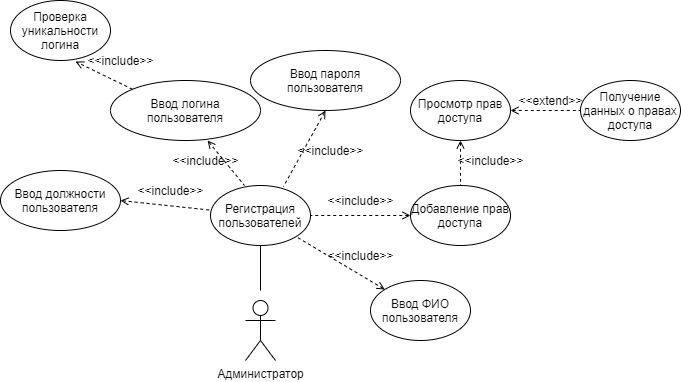


Рисунок 2.1.4. Диаграмма декомпозиции "Просмотр статистики"

## 2.2 Диаграммы автоматов

Диаграммы автоматов (англ. state machine) используются для описания поведения, реализуемого в рамках варианта использования, или поведения экземпляра сущности (класса, объекта, компонента, узла или системы в целом). Поведение моделируется через описание возможных состояний экземпляра сущности и переходов между ними на протяжении его жизненного цикла, начиная от создания и заканчивая уничтожением. Диаграмма автоматов представляет собой связный ориентированный граф, вершинами которого являются состояния, а дуги служат для обозначения переходов из состояния в состояние.

Под состоянием (англ. state) понимается ситуация в ходе жизни экземпляра сущности, когда эта ситуация удовлетворяет некоторому условию, экземпляр выполняет некоторые операции или ждет наступления некоторого события.

Дуги графа служат для обозначения переходов из состояния в состояние. Диаграммы автоматов могут быть вло­жены друг в друга, образуя вложенные диаграммы более детального пред­ставления отдельных элементов модели.

В связи с тем, что к данной системе будут иметь доступ разные пользователи (администраторы, менеджеры, логисты и курьеры), которые соответственно обладают разными правами, то у нас появляется необходимость для каждого типа пользователей организовать индивидуальный интерфейс с определёнными правами именно для данного типа. Эта ситуация отражена на контекстной диаграмме автоматов (рис. 2.2.1).

В данном случае, мы видим состояния системы, при ее использовании пользователями, имеющими разные роли, и как следствие, типы доступа к компонентам системы.

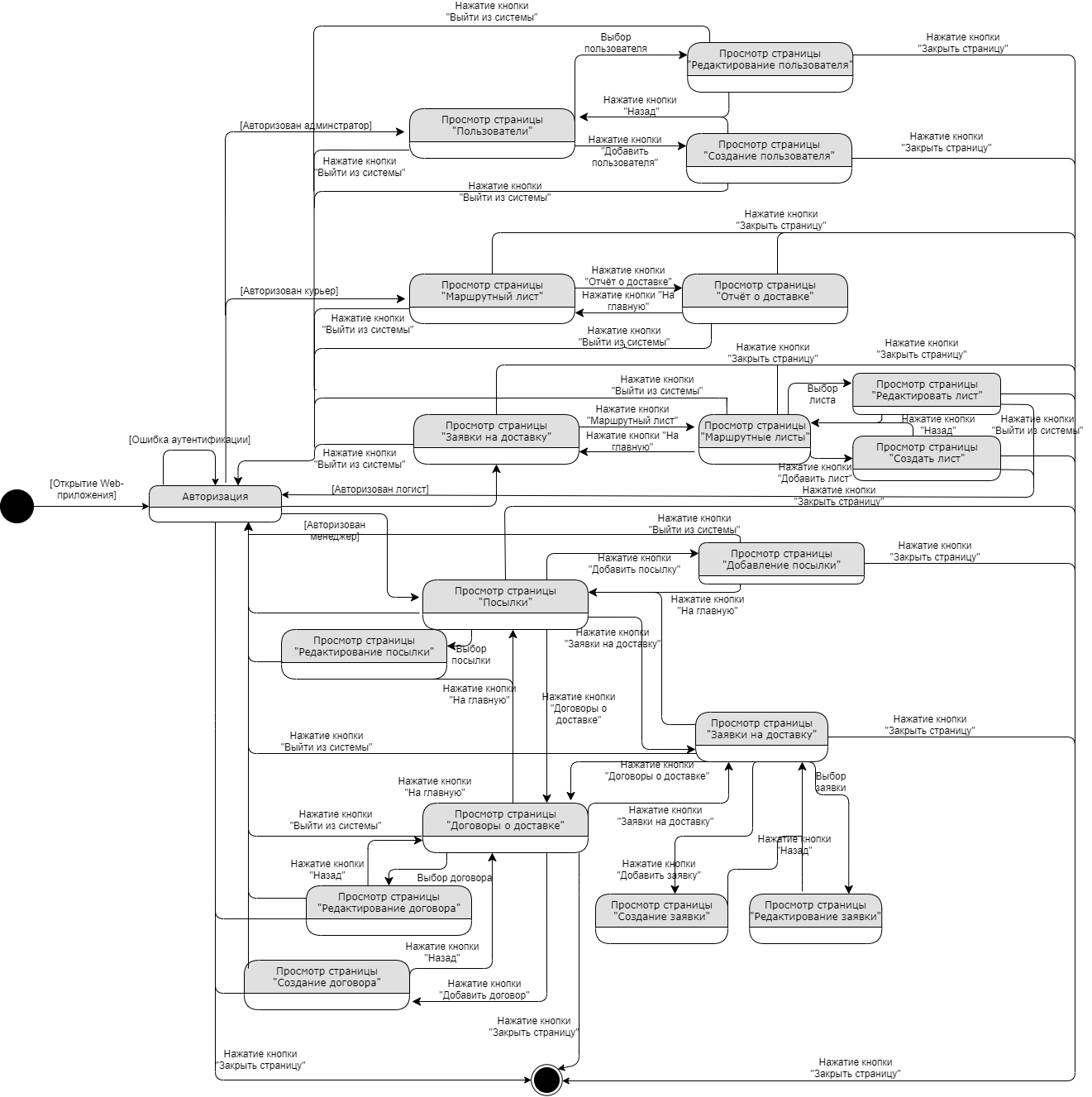


Рисунок 2.2.1. Контекстная диаграмма автоматов.

На диаграмме автоматов подсистемы «Создание маршрутного листа» (рис. 2.2.2) отображены состояния системы при управлении системой логистом. Диаграмма более подробно детализирует вариант использования «Создание листа». Сценарий «Создание листа» следующий: логист переходит на страницу «Создание листа», где при ее загрузке, происходит загрузка заявок на доставку и доступных курьеров. В случае неудачи, процесс загрузки повторяется, в то время как пользователь получает уведомление об ошибке. Далее, после успешной загрузки данных, пользователь вправе добавить заявку на доставку из списка, при этом происходит обновление списка заявок (т.к. один элемент может быть добавлен единожды, и при добавлении его на поля из списка он должен удалиться). В ходе дальнейшей работы пользователь может удалять и добавлять заявки. После этого логисту необходимо выбрать курьера для доставки и перенести маршрутный лист в состояние «Активен». Далее необходимо нажать кнопку «Создать» для добавления листа в БД.

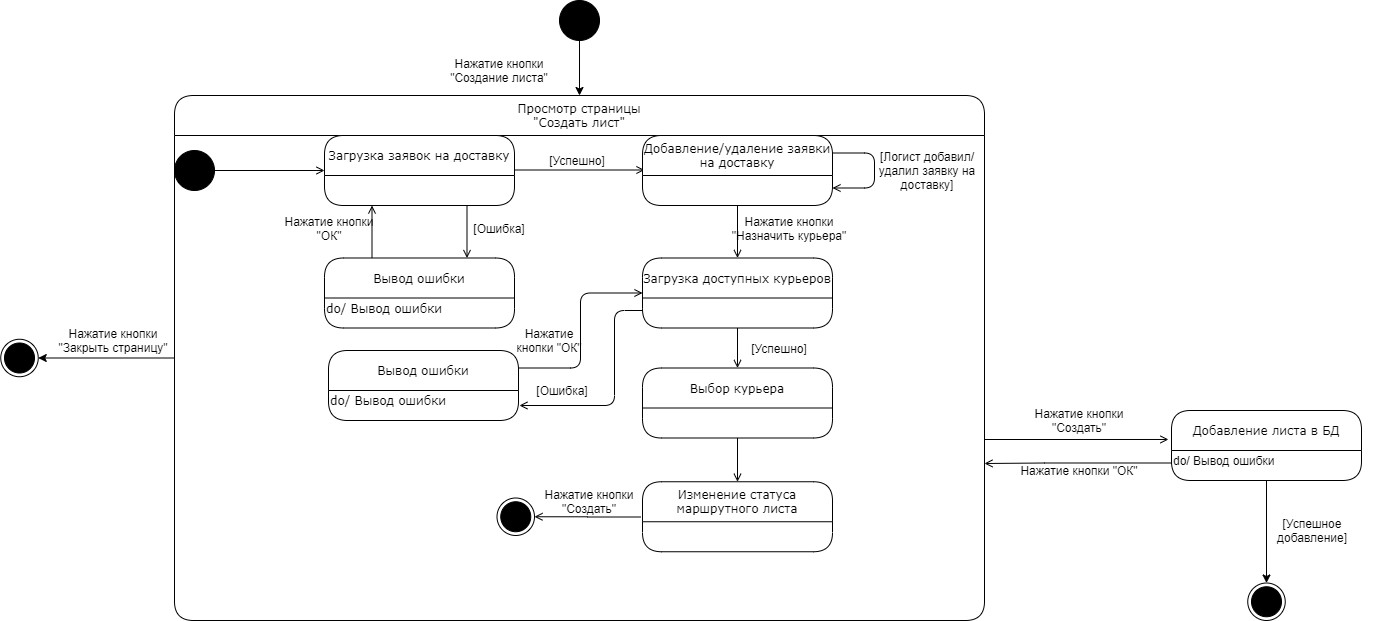


Рисунок 2.2.2 Диаграмма автоматов «Создание маршрутного листа»

На диаграмме автоматов подсистемы «Создание пользователя» (рис. 2.2.3) отображены состояния системы при управлении системой администратором. Диаграмма более подробно детализирует вариант использования «Создание пользователя». Сценарий «Создание пользователя» следующий: администратор должен создать учётную запись и выдать соответствующие права доступа новому сотруднику. При посещении страницы «Создание пользователя», происходит инициализация формы для заполнения данных, далее администратор вводит ФИО работника и должность, после чего логин, который проходит валидацию (выполняется запрос на сервер с целью того, чтобы не допустить дублирование логинов). Если все прошло успешно, то пользователь вводит пароль, далее администратор выбирает необходимые пара доступа для работника. Если на этом этапе возникли ошибки, то система уведомит об этом пользователя и ему необходимо будет заполнить поле новым паролем. Если же все прошло успешно, то при нажатии кнопки «Создать», система отправит запрос на сервер для создания записи в БД, после успешного ответа, процесс регистрации заканчивается.

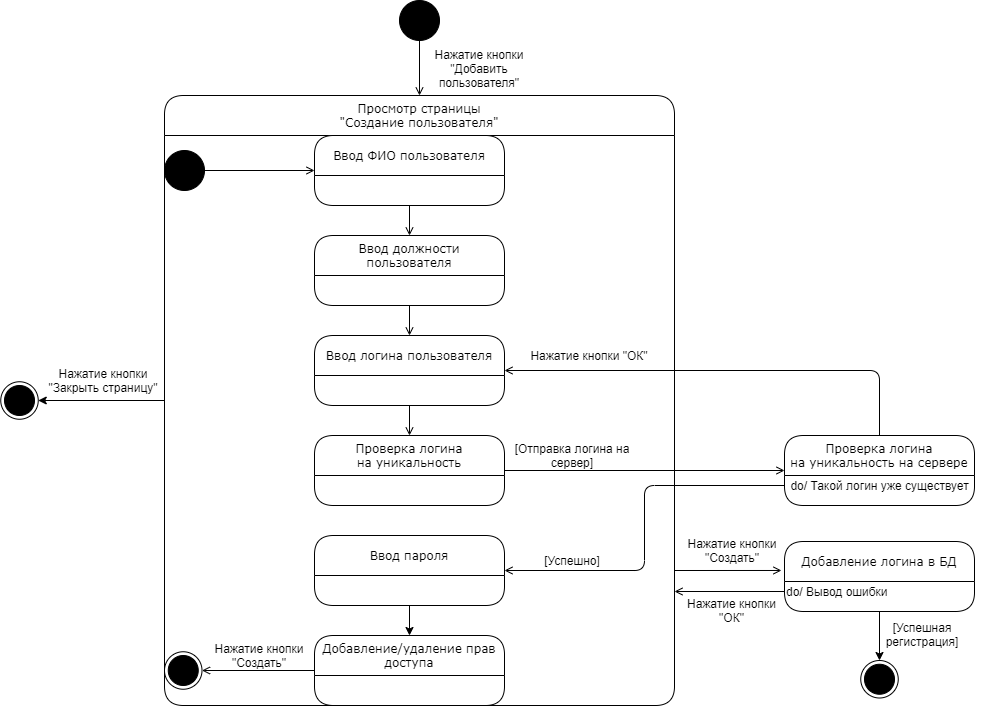


Рисунок 2.2.3 Диаграмма автоматов "Создание пользователя"

# 3. Модель анализа

Главная цель построения модели анализа заключается в уточнении вариантов использования с учетом внутренней архитектуры проектируемой системы.

Построение этой модели необходимо:

* для того чтобы выявить внутренние архитектуры, то есть определить основные классы и подсистемы;
* для поиска альтернативных вариантов реализации системы (подсистем) и выбора основного;
* для уточнения всех требований (функциональных и нефункциональных).

При разработке модели анализа строятся следующие диаграммы:

* классов анализа;
* последовательности;
* коммуникации.

## 3.1 Диаграмма классов анализа

Класс анализа – это укрупненная абстракция, которая на концептуальном уровне (без точного определения атрибутов и операций) описывает некоторый фрагмент системы.

Существует три вида классов анализа:

* граничный;
* управляющий;
* сущности.

Диаграмма классов анализа является прообразом классической диаграммы классов. Элементами, отображаемыми на диаграмме, являются классы и отношения между ними.

Назначение классов анализа:

* граничный класс – используется для моделирования взаимодействия между системой и актерами (пользователями, внешними системами или устройствами);
* управляющий класс – отвечает за координацию, взаимодействие и управление другими объектами, выполняет сложные вычисления, управляет безопасностью, транзакциями и т. п.;
* класс сущности – используется для моделирования долгоживущей, нередко сохраняемой информации. Классы сущности являются абстракциями основных понятий предметной области – людей, объектов, документов и т. д., как правило, хранимых в табличном или ином виде.

Связи между классами анализа отображаются с использованием отношений пяти видов:

* ассоциаций – показывает, что объекты одного класса содержат информацию о существовании (наличии в памяти) объектов другого класса и между ними имеется некоторая логическая или семантическая связь;
* агрегаций – указывает на отношение «часть–целое» и отображается сплошной линией с не закрашенным ромбиком со стороны «целого»;
* композиций – аналогично агрегации, в которой «части» не могут существовать отдельно от «целого»;
* обобщения – является обычным таксонометрическим отношением между более общим (абстрактным) классом (родителем или предком) и его частным случаем (дочерним классом или потомком);
* зависимостей – означает, что в спецификации или теле методов объектов одного класса (зависимого) выполняется обращение к атрибутам, методам или непосредственно к объектам другого класса.

Диаграмма классов анализа изображена на рисунке 3.1.1.

На диаграмме в виде классов отображено клиент-серверное взаимодействие между клиентским приложением и сервером. Соединение обеспечивает некоторый управляющий класс, который позволяет всем граничным классам приложения получать информацию, которая содержится на сервере, и наоборот, отправлять информацию на сервер для дальнейшей ее обработки. Все граничные классы представляют собой View-компоненты, которые отображаются на странице. В свою очередь все сущности представляют собой таблицы в базе данных.

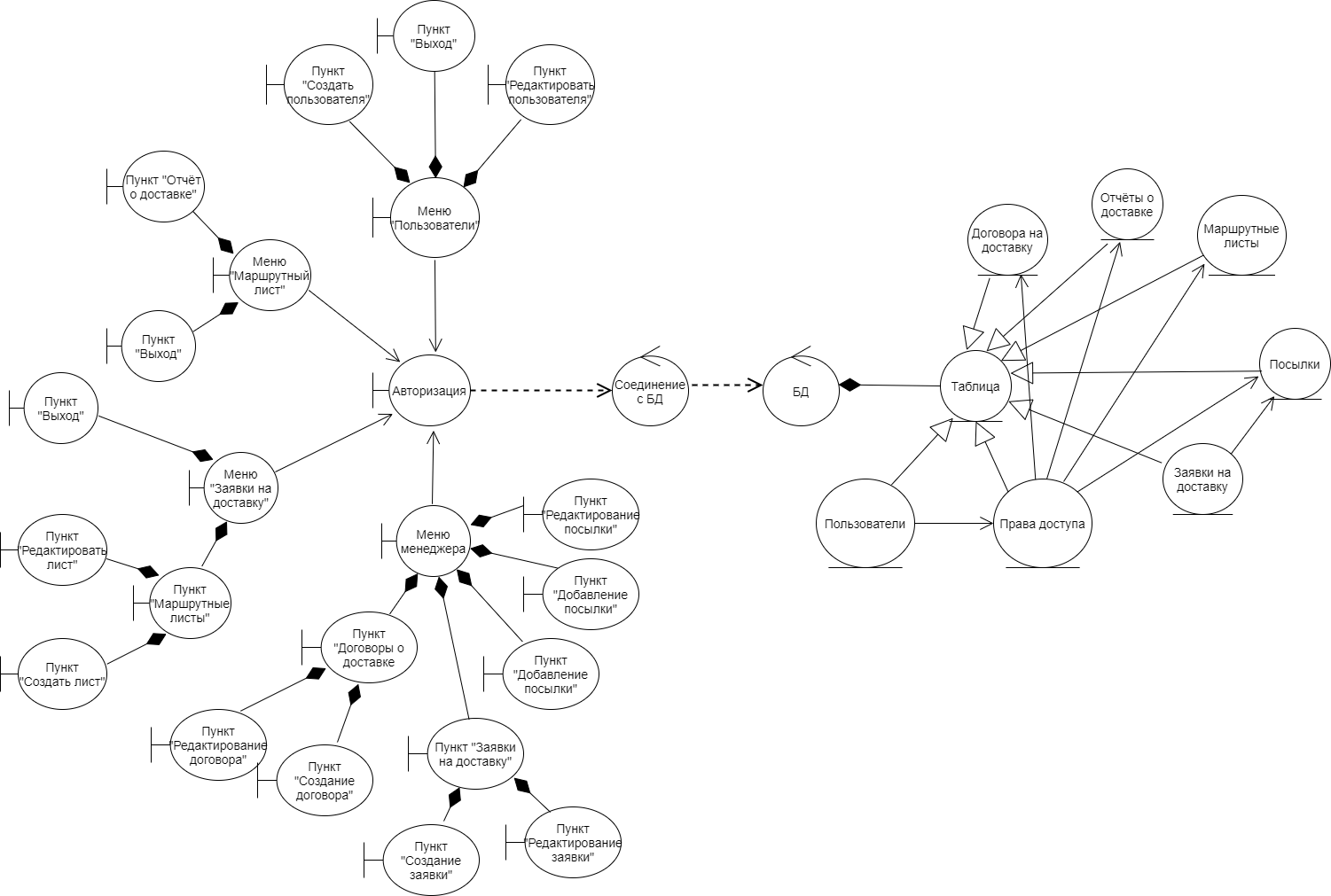


Рисунок 3.1.1 Диаграмма классов анализа

## 3.2 Диаграммы последовательности

Диаграмма последовательности – это одна из разновидностей диаграмм взаимодействия, её назначение заключается в моделирование взаимодействий различных объектов системы во времени, а также в обмене сообщениями между этими объектами.

На диаграмме последовательности изображаются объекты, которые в основном представляют экземпляры класса или сущности, обладающие поведением. Объектами на диаграмме последовательности могут быть пользователи, классы, программные компоненты, а иногда и системы в целом.

Диаграмма последовательности наглядно отображает временной аспект взаимодействия. Она имеет два измерения. Одно измерение (слева-направо) указывает на порядок вовлечения экземпляров сущностей во взаимодействие. Крайним слева на диаграмме отображается экземпляр актера или объект, который является инициатором взаимодействия. Правее отображается другой экземпляр сущности, который непосредственно взаимодействует с первым и т.д. Второе измерение (сверху-вниз) указывает на порядок обмена сообщениями. Начальному моменту времени соответствует самая верхняя часть диаграммы. Масштаб на оси времени не указывается, поскольку диаграмма отображает лишь временную упорядоченность взаимодействия типа «раньше-позже».

Для проектируемой информационной системы были построены две диаграммы последовательностей.

На рисунке 3.2.1 представлена диаграмма последовательности «Просмотр статистики». Для того чтобы попасть на страницу «Статистика» преподавателю (далее пользователь) необходимо выбрать советующий пункт меню в шапке сайта. После чего произойдет вмонтирование компонента «Просмотр статистики» на страницу. После вмонтирования компонент запросит у сервера список результатов. Сам сервер обратится к базе данных через провайдера и получит записи. Произведет сериализацию данных в формат JSON. После чего отправит ответ клиенту. После чего компонент осуществит вывод полученных диалогов. В случае ошибки загрузки, пользователь увидит соответствующее уведомление.

Далее пользователь может сортировать данные, для этого необходимо нажать кнопку «Сортировать» и ввести критерии сортировки. После чего компонента произведет сортировку, сформирует новое представление результатов и отобразит на странице результат сортировки.

Также пользователь может произвести поиск по записям, для этого необходимо будет ввести критерии поиска, после чего компонента проверит корректность введенных критериев, после чего, в случае успеха, сформирует результат поиска и отобразит его на странице, в противном случае, пользователя уведомят о некорректности введенных критериев.

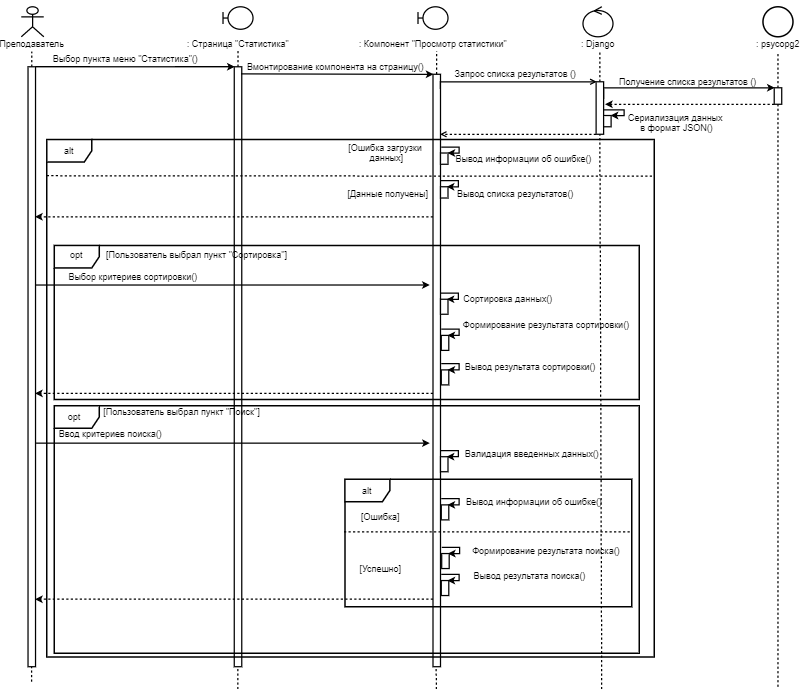


Рисунок 3.2.1. Диаграмма последовательности «Просмотр статистики»

На рисунке 3.2.2 представлена диаграмма последовательности «Регистрация студента». Чтобы студент прошел процесс регистрации ему необходимо перейти на страницу «Регистрация», после чего произойдет вмонтирование компонента «Регистрация» на страницу. После вмонтирования компонент запросит у сервера список существующих групп. Сам сервер обратится к базе данных через провайдера и получит записи. Произведет сериализацию данных в формат JSON. После чего отправит ответ клиенту. После чего компонент осуществит заполнение списка для поля «Группа». В случае ошибки загрузки, пользователь увидит соответствующее уведомление.

Далее студент вводит регистрационные данные, происходит валидация этих данных, в случае ошибки, пользователю необходимо исправить ошибки, на которые указывает компонента, после чего нажать кнопку «Регистрация» компонента запросит у сервера создать нового пользователя, сам сервер также обратиться к базе через провайдера, для создания новой записи в базе данных. В случае ошибки пользователь увидит соответствующее уведомление, в случае успеха, процесс регистрации завершиться.

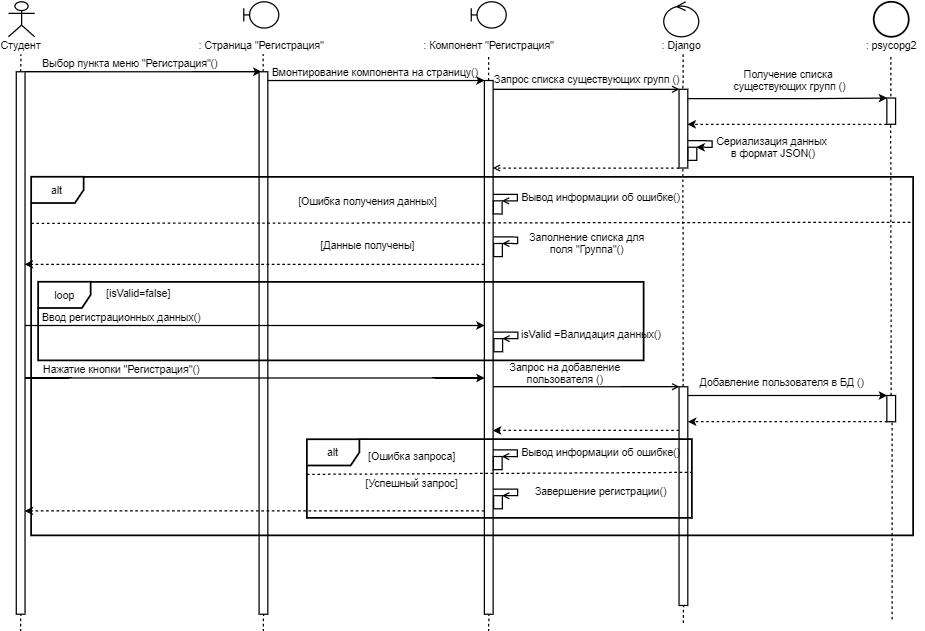


Рисунок 3.2.2. Диаграмма последовательности "Регистрация студента"

## 3.3 Диаграммы коммуникации

В отличие от диаграммы последовательности на диаграмме коммуникации основное внимание уделяется структуре взаимодействия. Помимо общих элементов (экземпляров актеров, объектов и сообщений) между участниками взаимодействия отображаются ненаправленные ассоциации, над которыми указываются передаваемые ими сообщениями. Другой отличительной особенностью является использование в спецификации сообщений нумерации, отражающей порядок их выполнения.

Проектировщикам диаграмма коммуникации может дать богатый материал о распределении обязанностей между объектами. Так, например, если диаграмма напоминает форму звезды, то можно сделать вывод, что система сильно зависит от центрального объекта. В этом случае стоит подумать о более равномерном распределении обязанностей между участниками взаимодействия. Или, наоборот, если в системе хранится и обрабатывается конфиденциальная информация, то большинство сообщений должно проходить через ядро безопасности – классы, отвечающие за идентификацию, аутентификацию и, возможно, шифрование / расшифрование данных.

Таким образом, цель самой коммуникации состоит в том, чтобы специфицировать особенности реализации отдельных наиболее значимых операций в системе. Коммуникация определяет структуру поведения системы.

На рисунке 3.3.1 показана диаграмма коммуникации «Просмотр статистики». Данная диаграмма сгенерирована автоматически на основе диаграммы последовательности, изображенной на рисунке 3.2.1.

На рисунке 3.3.2 показана диаграмма коммуникации «Регистрация студента». Данная диаграмма сгенерирована автоматически на основе диаграммы последовательности, изображенной на рисунке 3.2.2.

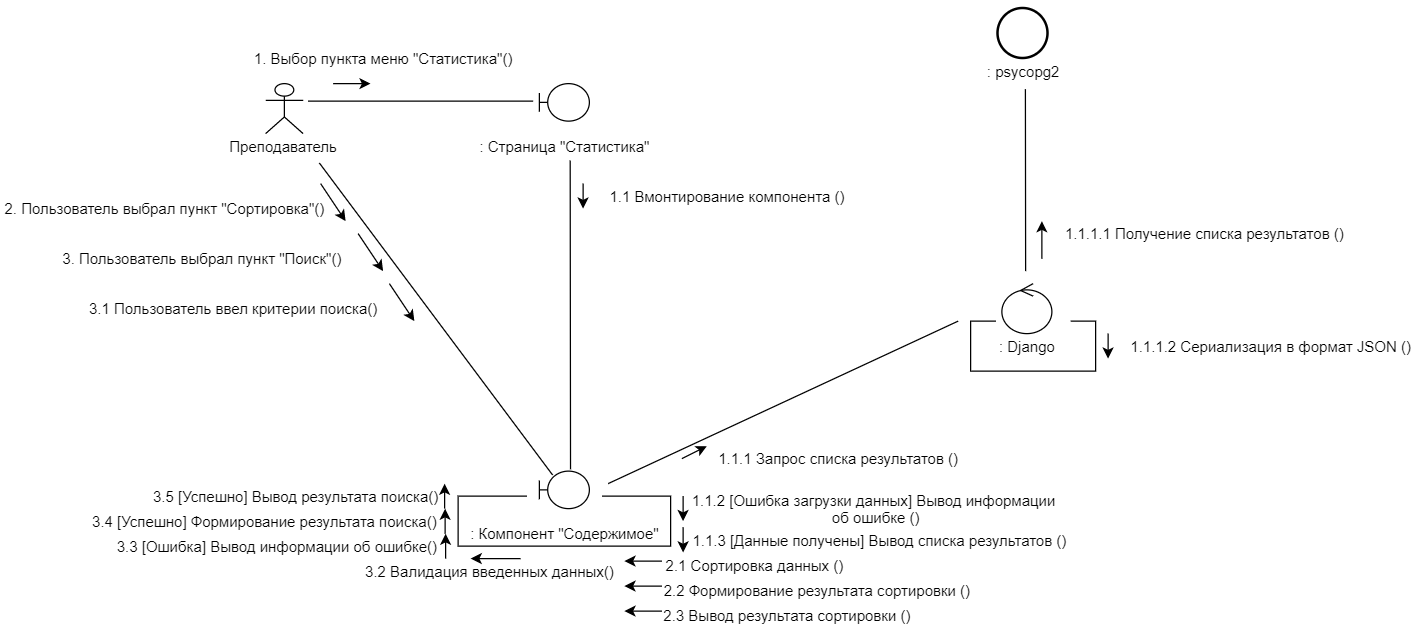


Рисунок 3.3.1. Диаграмма коммуникации «Просмотр статистики»

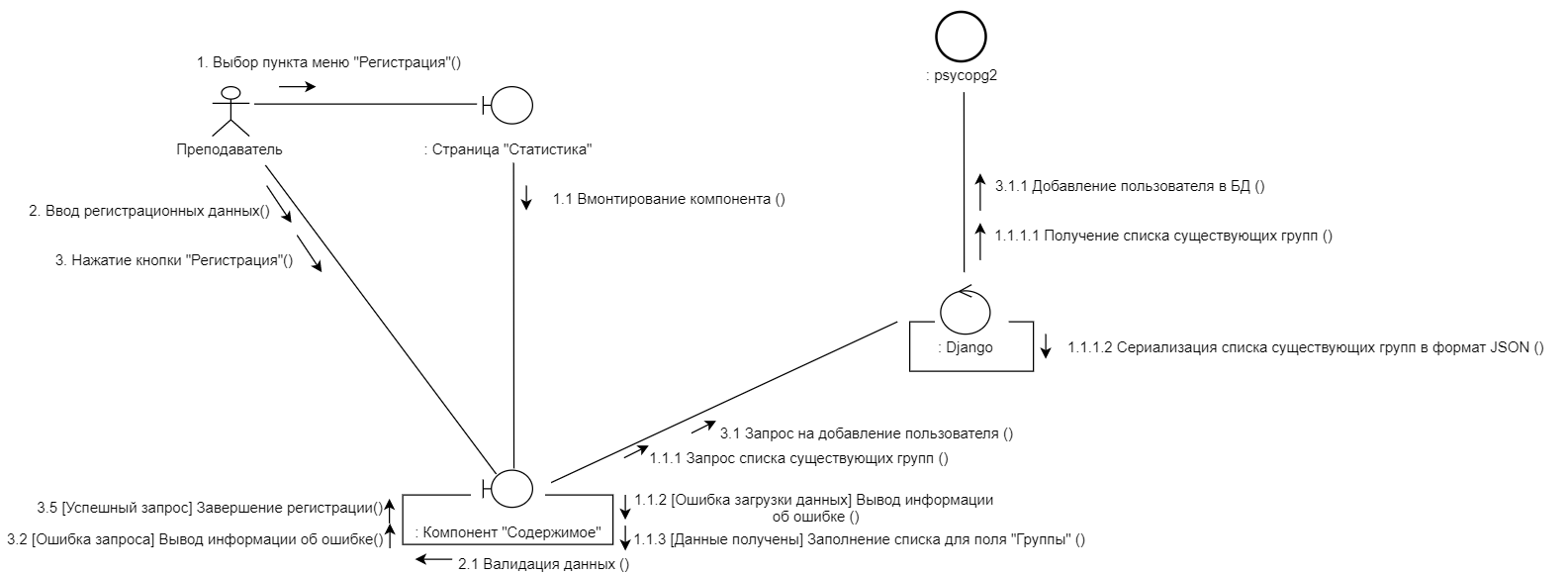


Рисунок 3.3.1. Диаграмма коммуникации «Регистрация студента»

# 4. Модель проектирования

В процессе проектирования создается архитектура системы, которая позволит реализовать и затем поддерживать все функции информационной системы.

Назначение модели проектирования заключается в создании полного детализированного описания внутренней архитектуры и алгоритмов работы системы.

Рекомендуется разрабатывать данную модель без привязки к конкретным языкам программирования, с помощью которых будет создаваться программный продукт, т. е. разрабатывать логическую модель.

Стоит оговориться, что создать модель без оглядки на используемые языки программирования невозможно, но, по крайней мере, необходимо стремиться к этому.

Построение модели проектирования необходимо:

* для уточнения внутренней архитектуры и вариантов использования системы;
* уточнения требований;
* определения детализированных алгоритмов работы системы в целом и ее отдельных элементов.

Модель проектирования представляется диаграммами классов и диаграммами деятельности.

## 4.1 Диаграммы классов

Диаграммы классов используются при моделировании информационных систем наиболее часто. Они являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграмма классов не отображает динамическое поведение объектов, изображенных на ней классов.

На диаграммах классов показываются классы, интерфейсы и отношения между ними.

Диаграмма классов представляет собой граф, вершинами которого являются элементы типа «классификатор», связанные различными типами структурных отношений.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

1. концептуальная точка зрения – диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;
2. точка зрения спецификации – диаграмма классов применяется при проектировании информационных систем;
3. точка зрения реализации – диаграмма классов содержит классы, используемые непосредственно в программном коде.

Классы могут иметь логическую и физическую реализации. Логические диаграммы классов в отличие от физических, строятся без привязки к языкам программирования.

В ходе выполнения курсового проекта было разработано два типа диаграмм классов: для клиентского приложения и для серверного приложения.

Прежде чем перейти к разработке этих двух моделей, опишем структуру таблиц базы данных:

Таблица 1. Пользователи (Users)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| ID пользователя | int | Уникальный идентификатор пользователя |
| Фамилия | nvarchar | Фамилия пользователя |
| Имя | nvarchar | Имя пользователя |
| Отчество | nvarchar | Отчество пользователя |
| Менеджер | boolean | Является ли пользователь менеджером |
| Логист | boolean | Является ли пользователь логистом |
| Администратор | boolean | Является ли пользователь администратором |
| Курьер | boolean | Является ли пользователь курьером |
| Логин | nvarchar | Логин пользователя |
| Пароль | nvarchar | Пароль пользователя |
| Должность | nvarchar | Должность пользователя |

Таблица 2. Отчёты о доставке (Delivery reports)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| ID отчёта | int | Уникальный идентификатор отчёта |
| ID заявки | int | Ссылка на заявку |
| Дата доставки | Data Time | Дата доставки |

Таблица 3. Договора на доставку (Delivery contract)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| ID contract | int | Уникальный идентификатор контракта |
| Фамилия отпр | nvarchar | Фамилия отправителя |
| Имя отпр | nvarchar | Имя отправителя |
| Отчество отпр | nvarchar | Отчество отправителя |
| Телефон отпр | nvarchar | Телефон отправителя |
| Адрес доставки | nvarchar | Адрес доставки |

Таблица 4. Маршрутные листы (Route maps)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| ID маршрутного листа | int | Уникальный идентификатор листа |
| ID пользователя | int | Ссылка на пользователя |
| Статус | Boolean | Обработана ли заявка |

Таблица 5. Заявки на доставку (Delivery orders)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| ID заявки | int | Уникальный идентификатор заявки |
| ID маршрутного листа | int | Ссылка на маршрутный лист |
| Дата создания | Data Time | Дата создания заявки |
| Срочность | boolean | Нужно ли выполнять заявку в первой |

Таблица 6. Посылки(Parcels)

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| ID послыки | int | Уникальный идентификатор посылки |
| ID заявки | nvarchar | Ссылка на заявку |
| ID договора | nvarchar | Ссылка на договор |
| Вес | int | Вес посылки |
| Высота | int | Высота посылки |
| Ширины | int | Ширина посылки |
| Длина | int | Длина посылки |
| Хрупкая | nvarchar | Хрупкая ли посылка |
| Фамилия получателя | nvarchar | Фамилия получателя |
| Имя получателя | nvarchar | Имя получателя |
| Отчество получателя | nvarchar | Отчество получателя |
| Телефон получателя | nvarchar | Телефон получателя |

Далее приведены диаграммы классов для БД (рис. 4.1.1 и рис. 4.1.2) и диаграммы классов для приложения (рис. 4.1.3 и рисунок 4.1.4). Каждый вид диаграмм представлен в двух экземплярах: логическом (на русском языке) и физическом (с учетом языка программирования).

Целевой СУБД для описания диаграмм классов БД, является PostgreSQL.

Целевым языком программирования для описания диаграмм классов приложения является Python (Django).

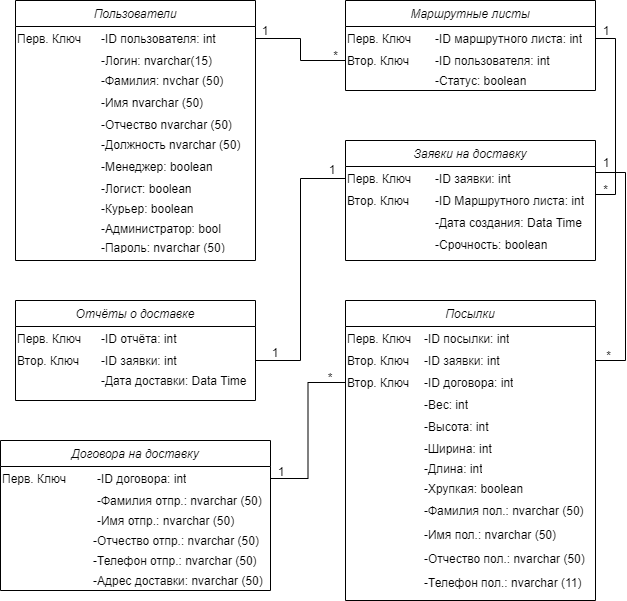


Рисунок 4.1.1. Логическая диаграмма классов базы данных

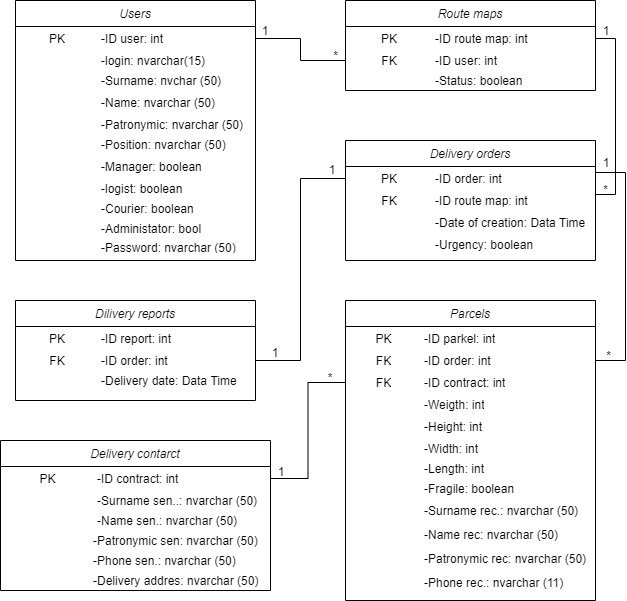


Рисунок 4.1.2. Физическая диаграмма классов базы данных

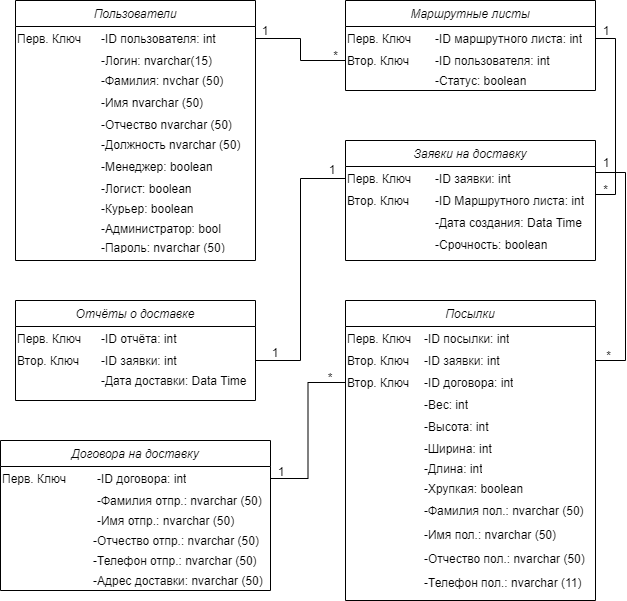


Рисунок 4.1.3. Логическая диаграмма классов приложения

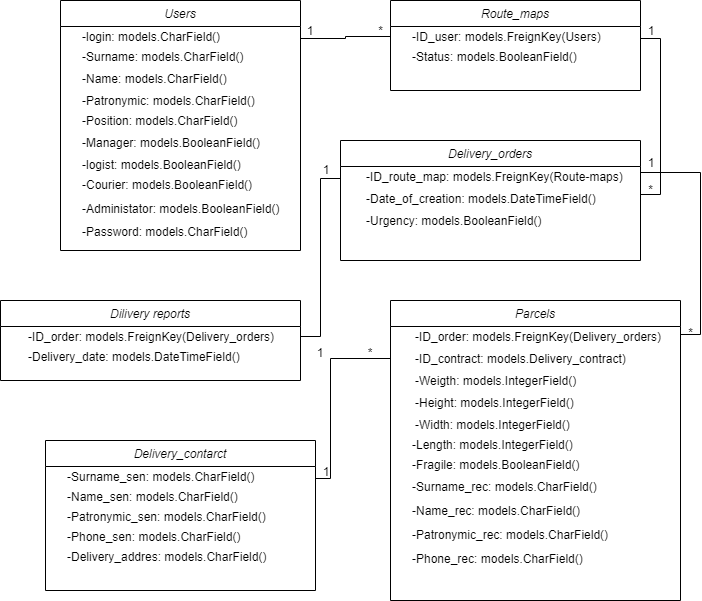


Рисунок 4.1.4. Физическая диаграмма классов приложения

## 4.2 Диаграммы деятельности

При моделировании поведения системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций.

Для описания поведения системы и ее отдельных элементов (поведенческих моделей) в UML предусмотрено четыре вида диаграмм:

* диаграммы автоматов;
* диаграммы последовательности;
* диаграммы коммуникации;
* диаграммы деятельности.

Несмотря на то, что первые три вида диаграмм, так или иначе, отображают динамические аспекты системы, они недостаточно формальны для детального описания алгоритмов работы. В структурном подходе для этого применяются блок-схемы, диаграммы EPC и BPMN. В UML аналогом блок-схем являются диаграммы деятельности (активности), схожие с ними по своей семантике и выразительным средствам (набору элементов).

Каждая диаграмма деятельности акцентирует внимание на последовательности выполнения определенных действий, которые в совокупности приводят к получению желаемого результата. Они могут быть построены для отдельного варианта использования, кооперации, метода и т. д. Диаграммы деятельности являются разновидностью диаграмм автоматов, но если на второй основное внимание уделяется статическим состояниям, то на первой – действиям.

Графически диаграмма деятельности, как и диаграмма автоматов, представляется в виде ориентированного графа, вершинами которого являются действия или деятельности, а дугами – переходы между ними. При этом в UML действие – это атомарная операция, выполнение которой не может быть прервано, а деятельность – составная операция, с возможностью ее прерывания. Переход к следующему действию или деятельности срабатывает сразу по их завершении.

Основными элементами диаграммы являются:

* исполняемые узлы – к исполняемым узлам (англ. executable nodes) относятся действия (англ. action) и деятельности (англ. activity);
* объекты – к объектам относятся непосредственно объекты (англ. object) в традиционном понимании UML, отправка сигнала (англ. send signal), прием сигнала (англ. accept signal) и событие времени (англ. time event);
* переходы – переход (англ. transition или activity edge), как и на диаграмме автоматов, отображается ассоциацией. На диаграммах деятельности различают следующие виды переходов:
  + поток управление;
  + объектный поток;
  + поток прерывания;
  + поток исключения.
* управляющие узлы – управляющим узлам (англ. control nodes) на диаграмме деятельности соответствуют псевдосостояния на диаграмме автоматов;
* коннекторы – коннекторы (англ. connectors) выступают в качестве соединителей, применяемых на блок-схемах;
* группирующие элементы – к группирующим элементам (англ. activity groups) относятся разделы деятельности (англ. activity partitions) и прерываемые регионы (англ. interruptible activity regions).

На рисунке 4.2.1 изображена диаграмма деятельности, описывающая процесс регистрации студента. На ней наглядно видно взаимодействие между разделами деятельности.

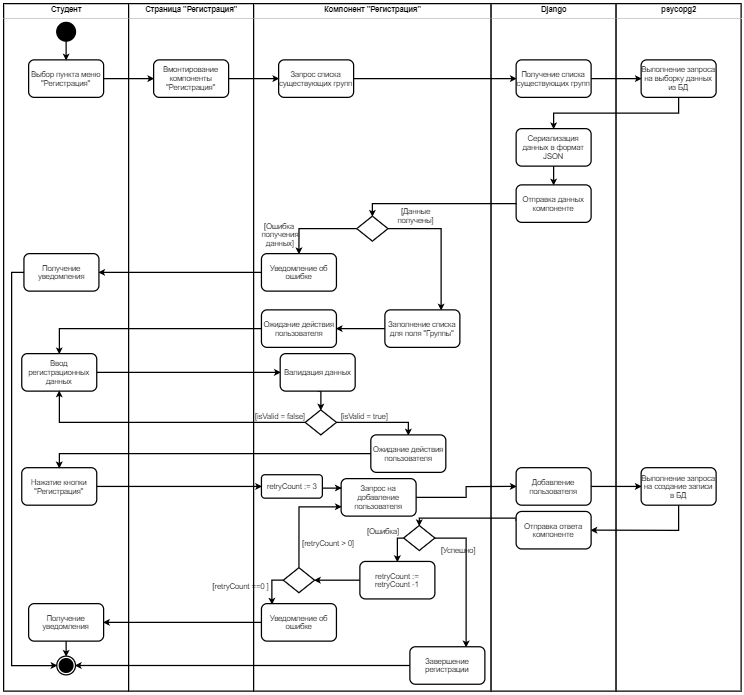


Рисунок 4.2.1. Диаграмма деятельности, описывающая процесс регистрации студента

На рисунке 4.2.2 изображена диаграмма деятельности, описывающая процесс валидации пароля, при регистрации. На ней отображен процесс валидации пароля с точки зрения его реализации в информационной системе.

Сценарий изменения пароля:

1. Пользователь вводит пароль;
2. Проверяется длинна пароля (не менее 8 символов)
   1. В случае ошибки дальнейшая проверка прерывается, и пользователя уведомляет о том, что пароль имеет недостаточную длину
3. Пароль проверяется на наличие символа с верхним регистром
   1. В случае ошибки дальнейшая проверка прерывается, и пользователя уведомляет о том, что пароль не содержит символ с верхним регистром.
4. Пароль проверяется на наличие цифр
   1. В случае ошибки дальнейшая проверка прерывается, и пользователя уведомляет о том, что пароль не содержит хотя бы одну цифру.
5. Принятие пароля.

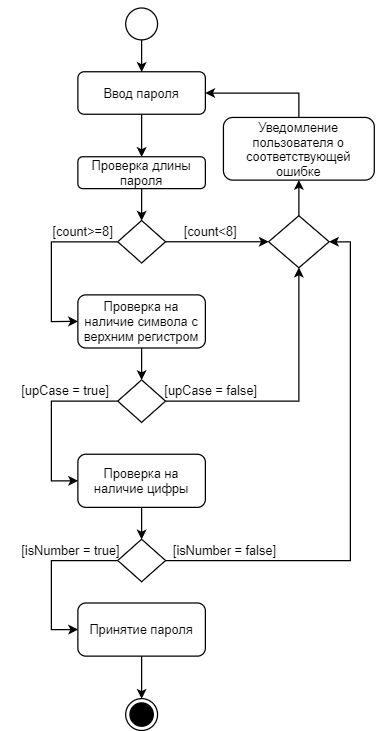


Рисунок 4.2.2. Диаграмма деятельности, описывающая процесс валидации пароля, при регистрации

# 5. Модель реализации

Модель реализации представляет физическую структуру реализации в терминах подсистем реализации и элементов реализации (каталогов и фалов, включая файлы исходного кода, файлы данных и исполняемые файлы и т.д).

Модель реализации идентифицирует физические компоненты реализации с целью облегчения их восприятия и управления ними.

Модель реализации определяет основные блоки интеграции, вокруг которых организованы рабочие группы, а также блоки, которым можно по отдельности присваивать версии, отдельно развертывать и заменять.

Модель реализации представляется диаграммами компонентов и развертывания.

Таким образом, при разработке модели преследуются цели:

* определение окончательного состава, структуры и кода классов;
* распределение классов по компонентам и подсистемам;
* определение топологии распределенной системы и распределение подсистем по узлам сети;
* планирование итераций (версий) сборки системы;
* сборка версий системы.

При разработке модели реализации рекомендуется построить диаграммы:

* компонентов;
* развертывания.

## 5.1 Диаграммы компонентов

Диаграмма компонентов описывает особенности физического представления системы.

Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код.

Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма компонентов разрабатывается для следующих целей:

* визуализации общей структуры исходного кода программной системы;
* спецификации исполняемого варианта программной системы;
* обеспечения многократного использования отдельных фрагментов программного кода;
* представления концептуальной и физической схем баз данных.

В языке UML для компонентов определены следующие стереотипы:

* «file» – любой файл, кроме таблицы;
* «executable» – программа (исполняемый файл);
* «library» – статическая или динамическая библиотека;
* «source» – файл с исходным текстом программы;
* «document» – остальные файлы (например, файл справки);
* «table» – таблица базы данных.

На рисунке 5.1.1 показана диаграмма компонентов

На данной диаграмме отражены основные элементы разрабатываемой информационной системы, а также основной исполняемый файл, файлы компонентов клиентского приложения, служебные файлы с дополнительной информацией, подключаемые библиотеки.

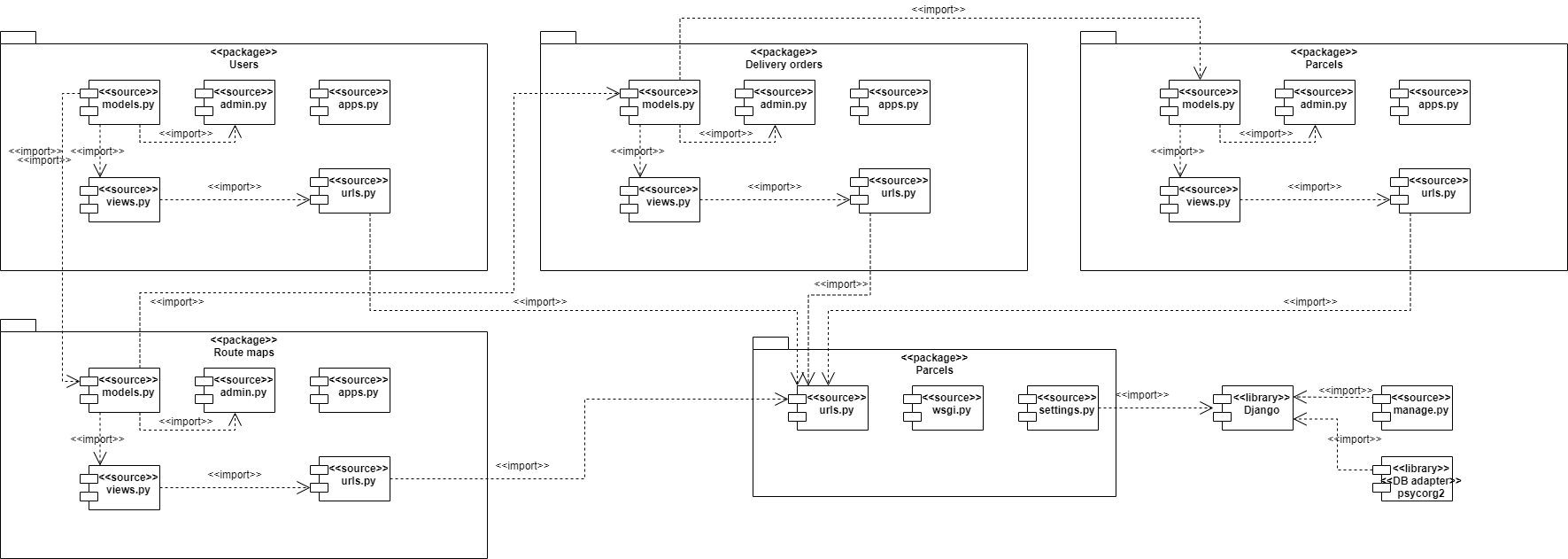


Рисунок 5.1.1. Диаграмма компонентов

## 5.2 Диаграмма развертывания

Целью диаграммы развертывания является представление физического расположения системы, взаимодействия физического оборудования, на котором запускается та или иная составляющая программного обеспечения.

Основные цели, преследуемые при разработке диаграммы развертывания:

* распределение компонентов системы по ее физическим узлам;
* отображение физических связей между узлами системы на этапе исполнения;
* выявление узких мест системы и реконфигурация ее топологии для достижения требуемой производительности.

Элементами диаграммы реализации являются узлы, компоненты и связи между ними.

На рисунке 5.2.1 представлена диаграмма развертывания, которая отражает физические компоненты информационной системы.

Для работы с информационной системой, на машинах пользователей должен быть установлен браузер. Само приложение (причем как фронтенд, так и бэкенд) и база данных устанавливаются на сервер.

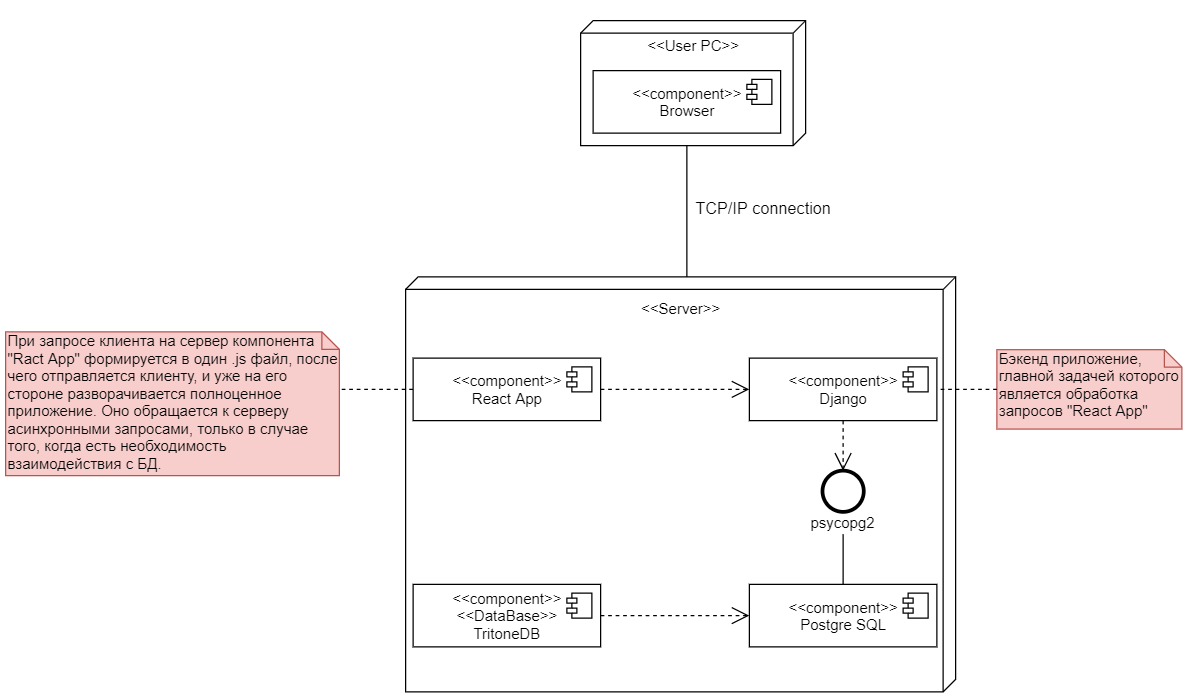
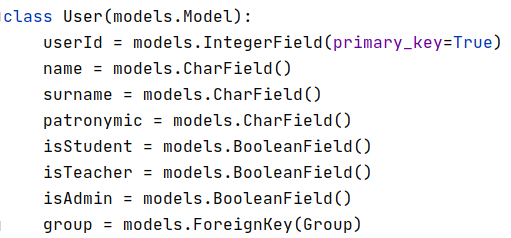


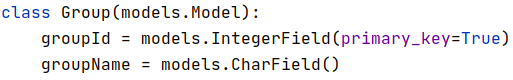
Рисунок 5.2.1 Диаграмма развертывания ИС

# 6. Сгенерированный программный код

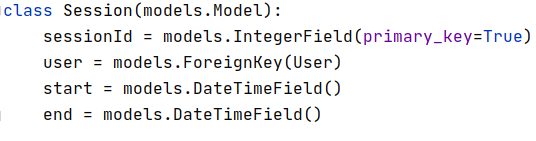
Средствами программного продукта АSTAH-PROFESSIONAL (Python Django Patch) для сформированной статической модели классов был автоматически сгенерирован программный код.



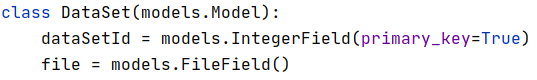
Листинг 6.1 Класс User (Пользователь)



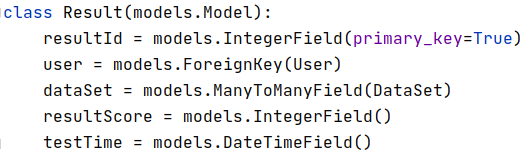
Листинг 6.2 Класс Group (Группа)



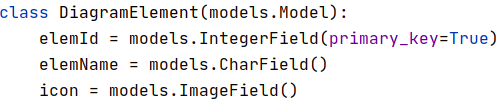
Листинг 6.3 Класс Session (Сессия)



Листинг 6.4 Класс DataSet (Данные)



Листинг 6.5 Класс Result (Результат)



Листинг 6.6 Класс DiagramElement (Элемент диаграммы)

# 7. Руководство пользователя

## 7.1 Авторизация

После открытия сайта в браузере, сперва пользователь попадает на страницу Авторизации (рис 7.1.1), если пользователь не был авторизован до этого в системе. Пользователю необходимо указать логин и пароль.

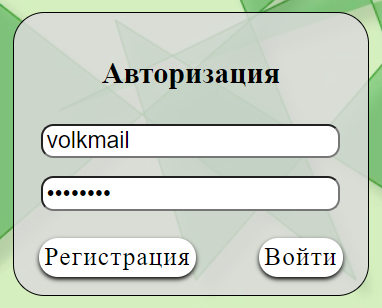


Рисунок 7.1.1. Форма авторизации

После ввода пароля, пользователю необходимо нажать кнопку «Войти». В случае если логин и пароль верный, пользователь увидит главную страницу (рис. 7.3.1), иначе пользователя уведомят о неправильности введенных данных.

Также еще не зарегистрированный пользователь сможет пройти процесс регистрации, нажав кнопку «Регистрация» (такой вариант, позволит зарегистрироваться только студенту).

## 7.2 Регистрация

Процесс регистрации имеет несколько вариантов:

1. Студент регистрирует сам себя(рис 7.2.1). Для этого ему необходимо заполнить соответсвующие поля. После регистрации сайт перенаправит студента на форму авторизации (рис 7.1.1).
2. Преподаватель регистрирует студента, в данном варианте форма регистрации имеет вид как и в 1 случае (рис 7.2.1).
3. Администратор регистрирует преподавателя, в данном случае форма регистрации имеет вид (рис 7.2.2)

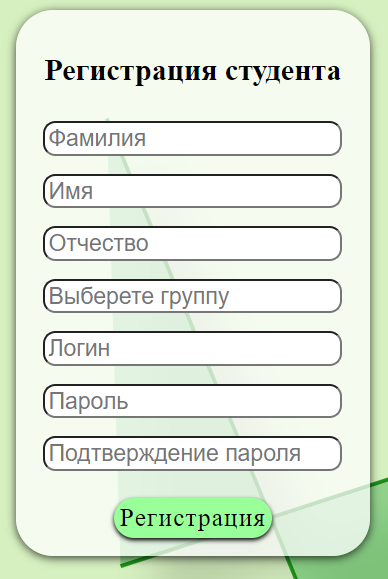


Рисунок 7.2.1. Форма регистрации студента

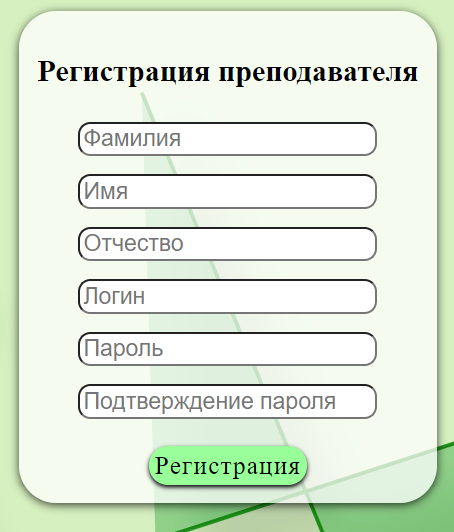


Рисунок 7.2.2. Форма регистрации преподавателя

## 7.3 Главная страница

Функционал программы зависит от того, кто был авторизирован. Так, например, если процесс авторизации прошел студент, он увидит главную страницу в следующем формате (рис 7.3.1). Тут мы видим функционал, который доступен студенту.

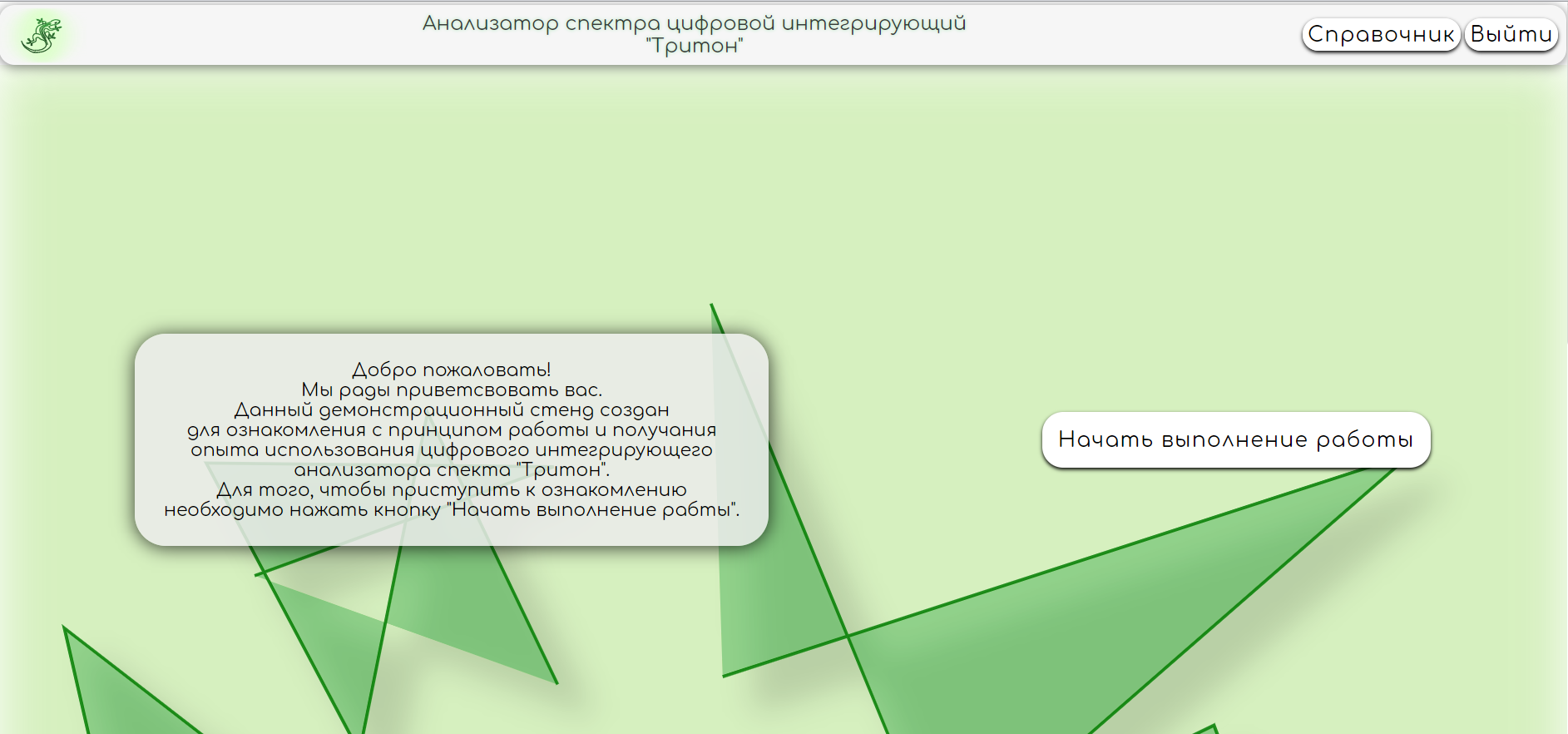


Рисунок 7.3.1 Главная страница "Студента"

Действия студента могут быть различными:

1. Начать прохождение тестирования, путем нажатия «Начать выполнение работы»;
2. Ознакомиться со справочными материалами, нажав кнопку «Справочник» (справочнику посвящен раздел 7.4).
3. Выйти из системы, нажав кнопку «Выйти».

На рисунке 7.3.2 изображена главная страница, для пользователя с ролью преподавателя. Из него видно, что пользователь при авторизации автоматически попадает на страницу просмотра статистики. Более подробно об этой странице раскрыто в разделе 7.7.

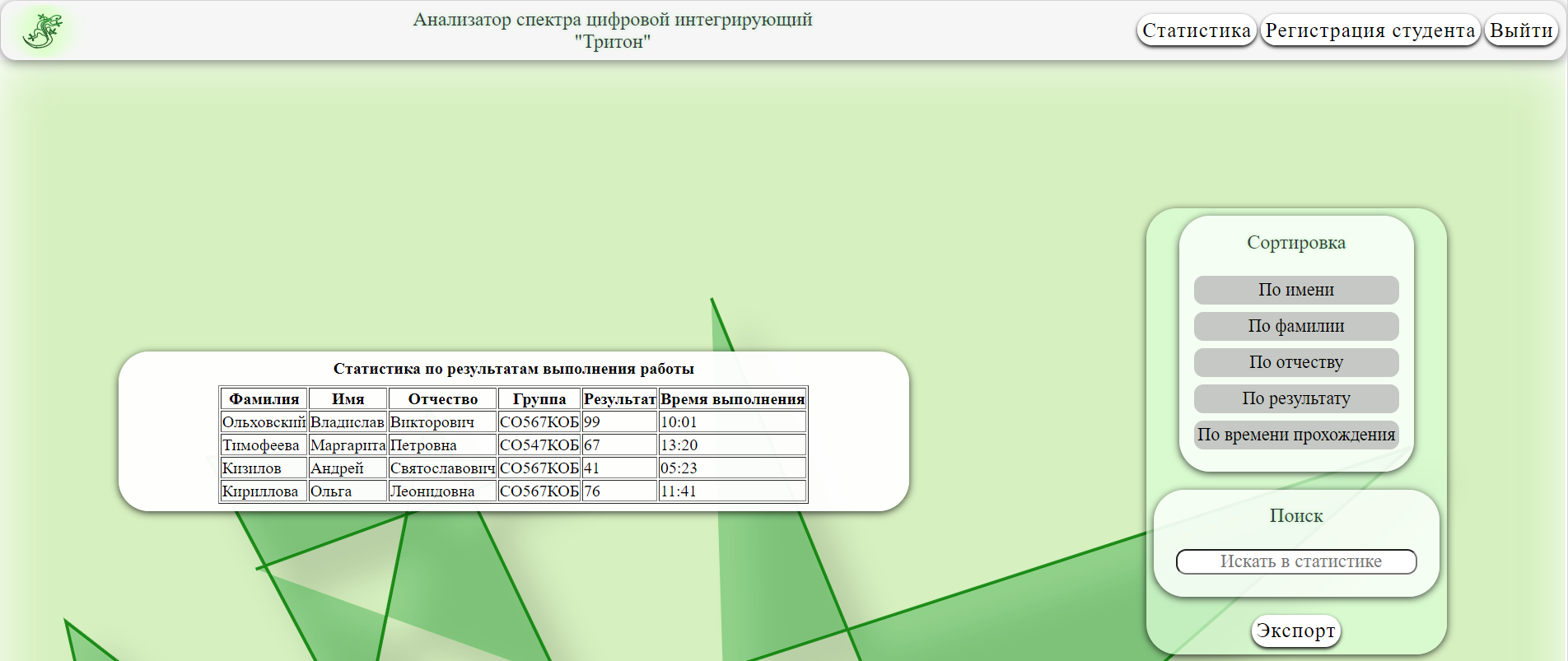


Рисунок 7.3.2. Главная страница преподавателя

На рисунке 7.3.3, продемонстрирована главная страница, пользователя, который является администратором. Более подробно ее функционал расписан в разделе 7.7.

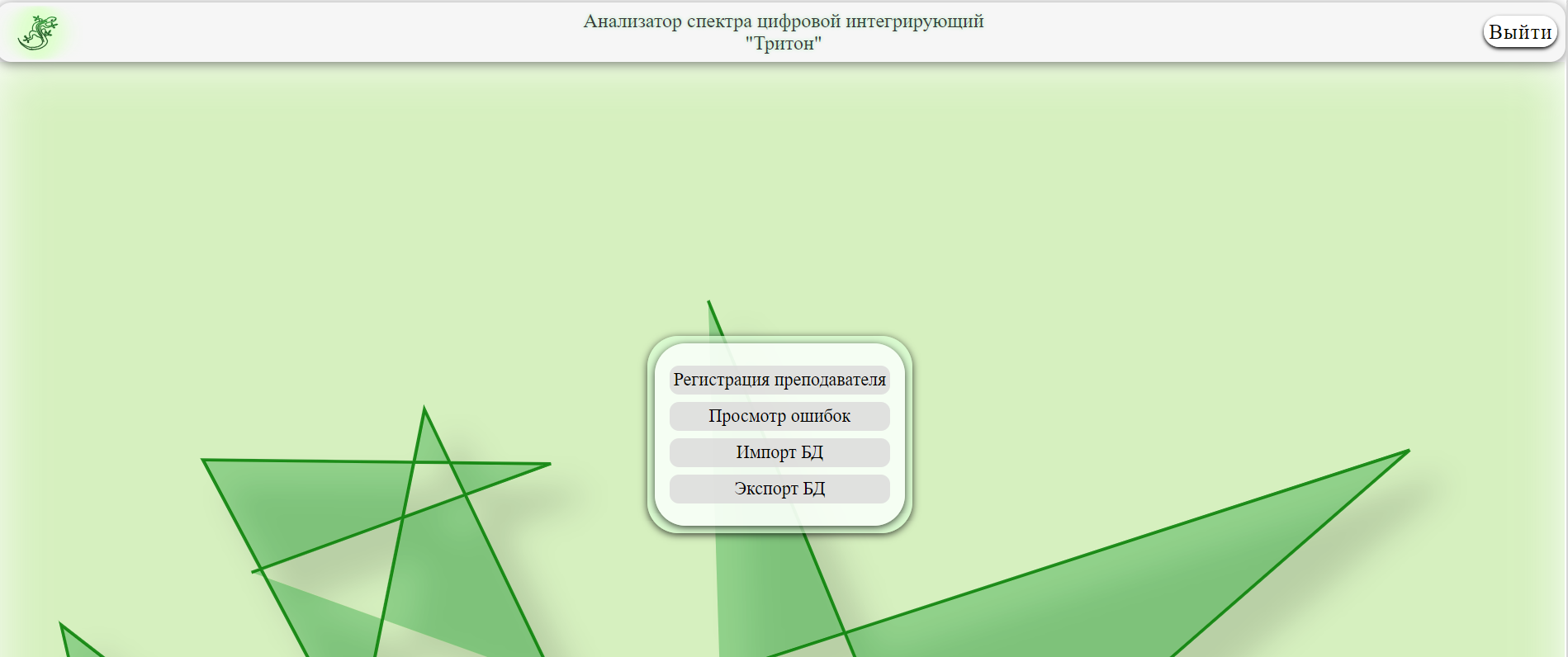


Рисунок 7.3.3. Главная страница администратора

## 7.4 Справочные материалы

Авторизованный студент, может ознакомиться со справочными материалами (рис 7.4.1), для этого ему будет необходимо, находясь на главной странице, нажать кнопку «Справочник».

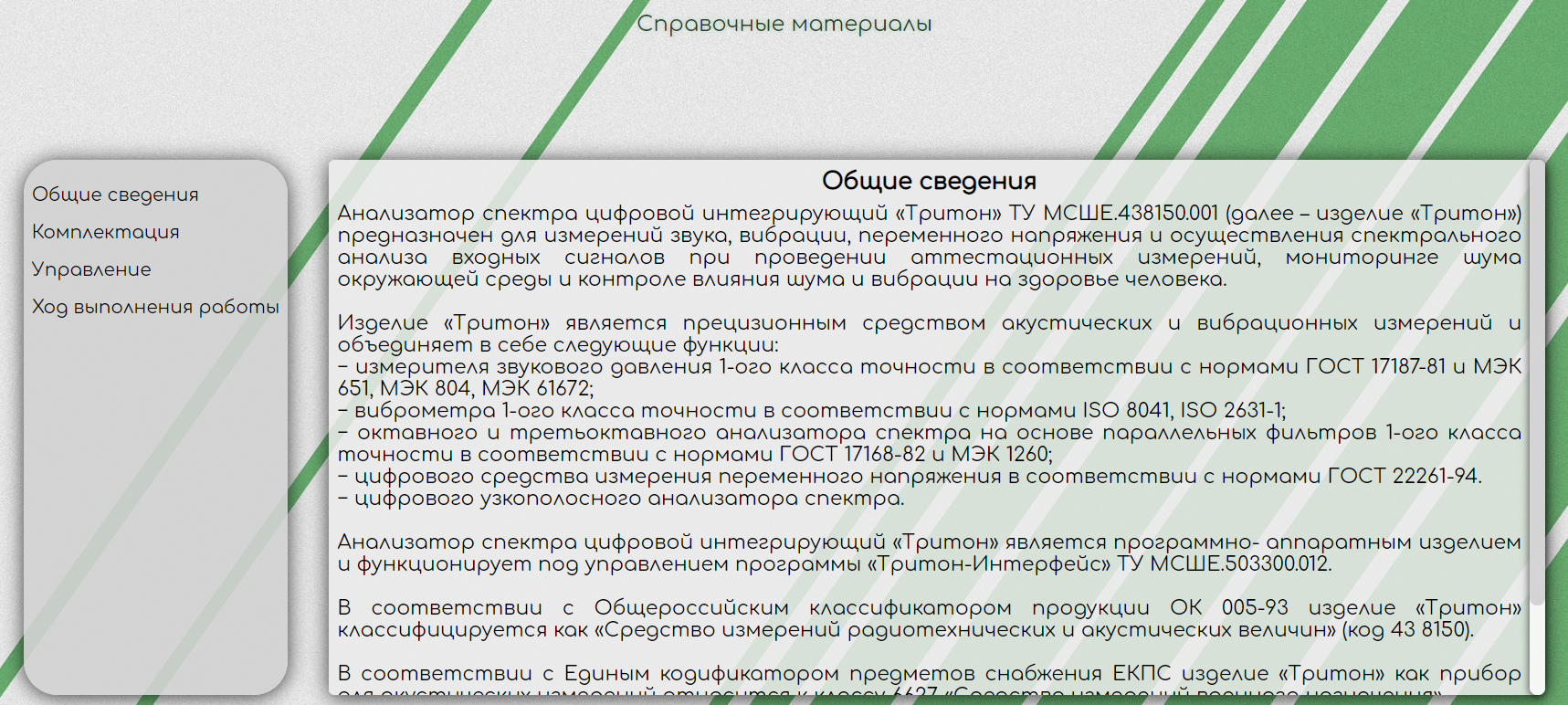


Рисунок 7.4.1. Страница справочных материалов

Справочник содержит различную информации относительно самого анализатора спектра «Тритон», управления при построении схем, и о ходе выполнения работы.

## 7.5 Построение схем

Первым этапом выполнения работы, является процесс построения схем. Опять же для этого необходимо авторизоваться, с ролью студента. Для того, чтобы приступить к построению схемы, необходимо нажать кнопку «Начать выполнение работы», находясь на главной странице. После чего произойдет перенаправление на страницу «Построение схемы устройств» (рис 7.5.1).

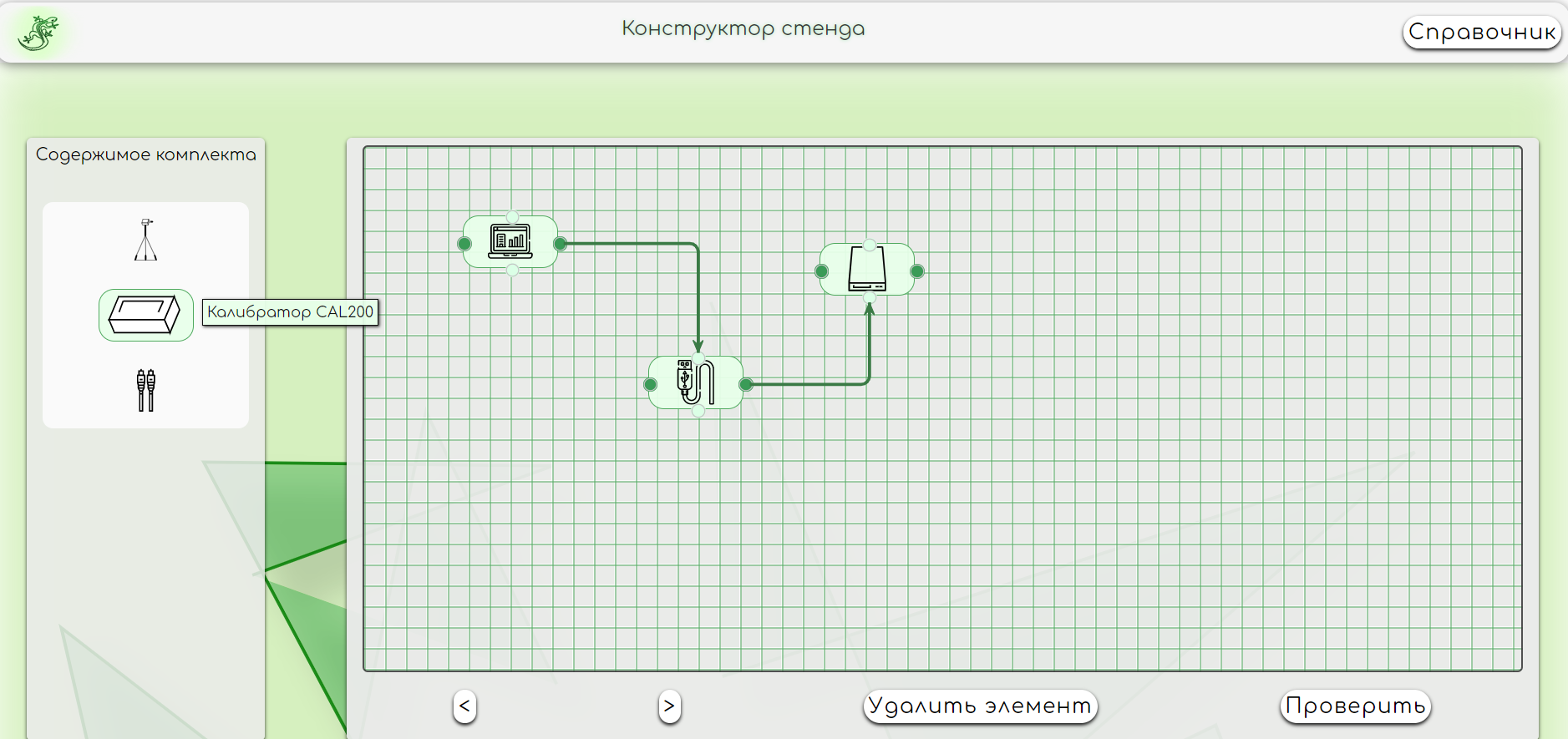


Рисунок 7.5.1. Страница "Построение схемы устройств"

Находясь на этой странице, пользователь, будучи авторизованным может собрать схему устройств.

Для этого ему необходимо выбрать из списка справа элемент, который он бы хотел добавить на поле, после чего он удалиться из списка и добавиться на поле, его можно будет перемещать и соединять с другими элементами, находящимися на поле.

Также студент все еще имеет доступ к справочнику, для этого ему необходимо нажать кнопку «Справочник».

Под полем, находятся кнопки управления:

1. «<» и «>» служат для возврата поля в предыдущее состояние или возвращения в текущее.
2. Кнопка «Удалить элемент» необходима, для того, чтобы удалить элемент с поля. При удалении, элемент опять вернется в список элементов.
3. Кнопка «Проверить» необходима для того, чтобы студент мог проверить правильность построенной схемы. Если проверка пройдет успешно, его перенаправить на страницу «Просмотр результатов».

## 7.6 Просмотр статистики

На странице «Просмотр статистики» (рис. 7.6.1), преподаватель, который прошел процесс авторизации, может просматривать статистику выполнения задания студентами. Также ему доступен следующий функционал:

1. Сортировка по определенным критериям
   1. По имени;
   2. По фамилии;
   3. По отчеству;
   4. По результату;
   5. По времени прохождения
2. Поиск по записям.

Также отдельно стоит выделить, что при необходимости регистрации студента, преподаватель должен нажать кнопку «Регистрация студента» и ему откроется форма, изображенная на рисунке 7.2.1.

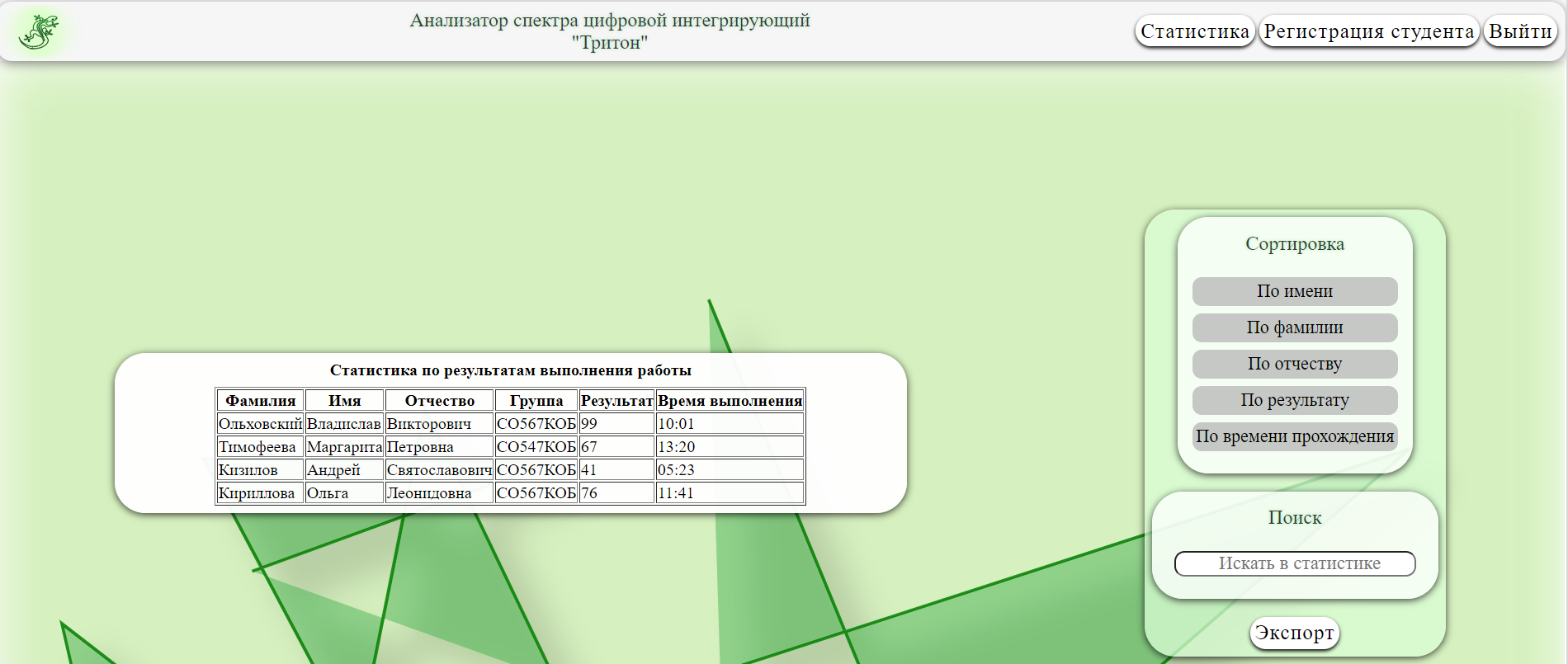


Рисунок 7.6.1. Страница "Просмотр статистики"

## 7.7 Администрирование

На рисунке 7.7.1 изображена главная страница Администратора. Для того, чтобы попасть на нее, соответственно необходимо авторизироваться под администратором.

Она несет в себе следующий функционал:

1. Кнопка «Регистрация преподавателя» перенаправит на соответствующую форму (рис 7.2.2), для того, чтобы зарегистрировать в системе нового преподавателя;
2. Кнопка «Просмотр ошибок» вмонтирует компонент «Просмотр ошибок», с помощью которого можно будет отследить системные ошибки;
3. Кнопка «Импорт БД» вмонтирует компонент «Импорт БД», с помощью которой можно произвести импорт текущей базы данных;
4. Кнопка «Экспорт БД» вмонтирует компонент «Экспорт БД», с помощью которого можно произвести экспорт стороней БД (например, бэкап БД);
5. Конопка «Выйти» позволит пользователю выйти из системы.

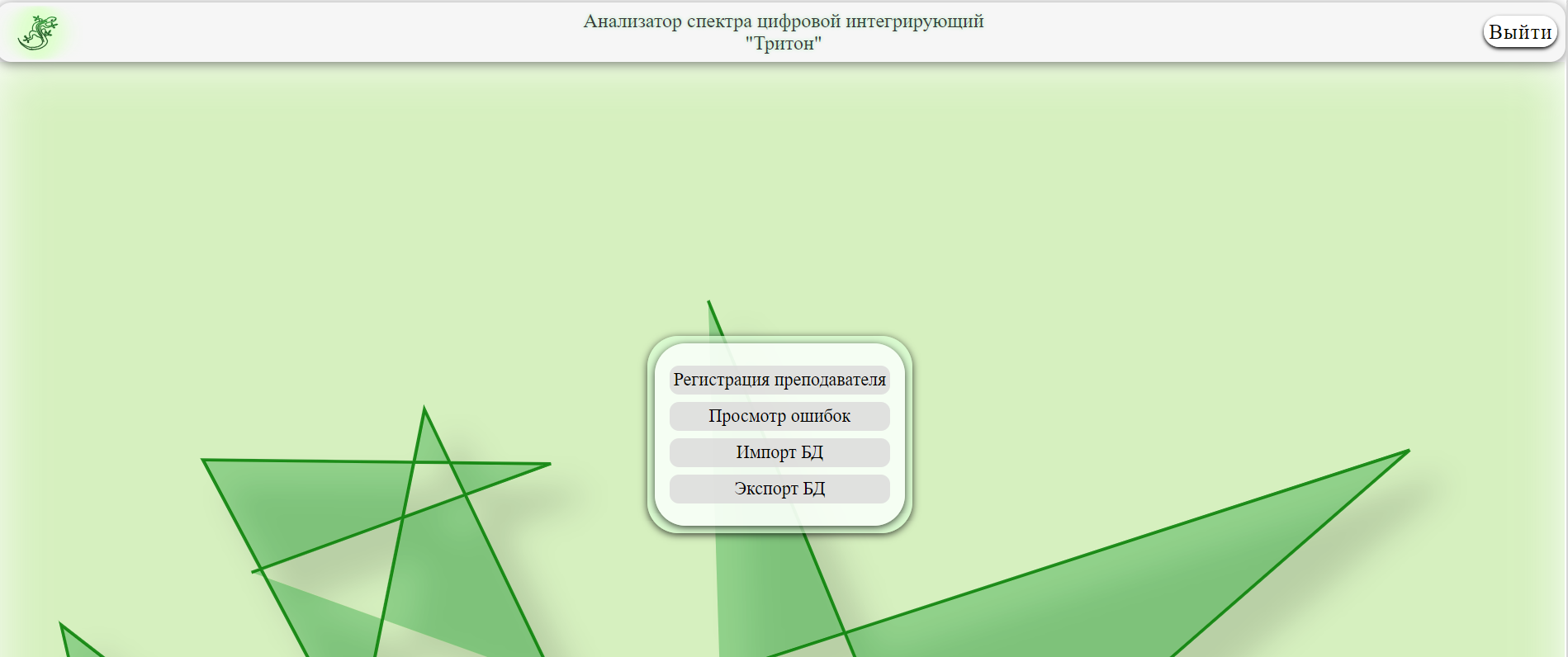


Рисунок 7.7.1. Главная страница Администратора

# Заключение

Целью данного курсового проект являлась разработка проекта информационной системы «Курьерская служба».

В ходе курсового проекта было проведено исследование предметной области, беседы с заказчиком и в полной мере раскрыта тема поставленной цели.

Разработаны и построены различные модели для проекта информационной системы. С учетом построенных моделей была разработана информационная система. Для разработанной информационной системы было написано руководство пользователя.

Разработанное приложение построено с использованием актуальных технологий веб-разработки на сегодняшний день.

# Список используемых источников

1. Анисимов, В. В. Проектирование информационных систем: курс лекций [Текст] : в 2 ч. / В.В. Анисимов. – Хабаровск: Изд-во ДВГУПС, 2006. – Ч. 1; Структурный подход. – 2006. – 112 с.

2. Анисимов, В. В. Проектирование информационных систем: курс лекций [Текст] : в 2 ч. / В. В. Анисимов. – Хабаровск: Изд-во ДВГУПС, 2007. – Ч. 2; Обектно-ориентированный подход. – 2007. – 100 с.

3. Гурвиц, Г. А. Microsoft Access 2010. Разработка приложений на реаль-ном примере [Текст] / Г. А. Гурвиц. Санкт-Петербург : БХВ-Петербург, 2010. - 493 с.