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: «Сервисный центр по ремонту смартфонов»

Выполнил: Кутузов В.А.
студент гр.СО251КОБ
Проверил: Анисимов В.В.

Хабаровск
2020 г.

**Содержание**

1. Описание предметной области ..... 3

2. Модель вариантов использования ..... 3

2.1 Диаграммы вариантов использования ..... 4

2.2 Диаграммы автоматов ..... 9

3. Модель анализа ..... 13

3.1 Диаграмма классов анализа ..... 13

3.2 Диаграммы последовательности ..... 17

3.3 Диаграммы коммуникации ..... 20

4. Модель проектирования ..... 24

4.1 Диаграммы классов ..... 24

4.2 Диаграммы деятельности ..... 31

5. Модель реализации ..... 35

5.1 Диаграммы компонентов ..... 36

5.2 Диаграмма развертывания ..... 39

6. Сгенерированный программный код ..... 40

7. Руководство пользователя ..... 42

Заключение ..... 47

Список используемых источников ..... 48

## 1. Описание предметной области

Целью данного курсового проекта является разработка информационной системы «Сервисный центр по ремонту смартфонов».

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 три актера:

1. Пользователь — учащийся высшего учебного заведения, роль которая является одной из ключевых, так как именно для них и создается данный продукт.
2. Сотрудник сервиса – работник, который оказывает услуги, по диагностике и ремонту смартфонов.
3. 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 для пользователя (далее Регистрация);
2. Регистрация сотрудника сервиса;
3. Авторизация пользователей;
4. Оформление заявки на консультацию;
5. Оформление заказа на оказание услуг;
6. Ознакомление с перечнем услуг;
7. Просмотр информации о заказе;
8. Изменение заказа;
9. Принятие заказа;
10. Управление контентом сайта;
11. Просмотр истории ошибок сайта;

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

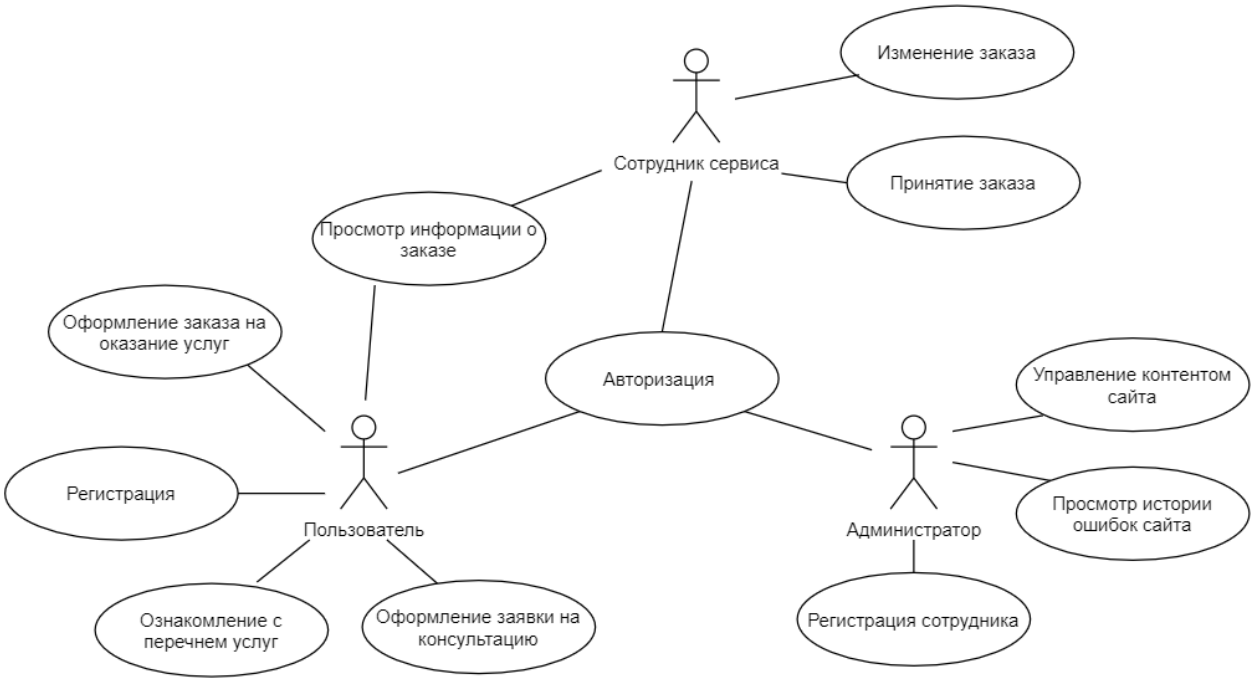


Рисунок 2.1.1.Контекстная диаграмма вариантов использования

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 «Изменение статуса заказа» (рис. 2.1.2), на которой в качестве «центрального» актера выступает «Сотрудник сервиса». В данном разделе пользователь может выполнять следующие действия

1. Изменение заказа
    a. Добавление комментария
    b. Изменение сроков окончания работ
2. Изменение статуса заказа
    a. Выбор статуса из списка
    b. Поиск по статусам

Для выполнения данных действий сотруднику сервиса необходимо авторизоваться.

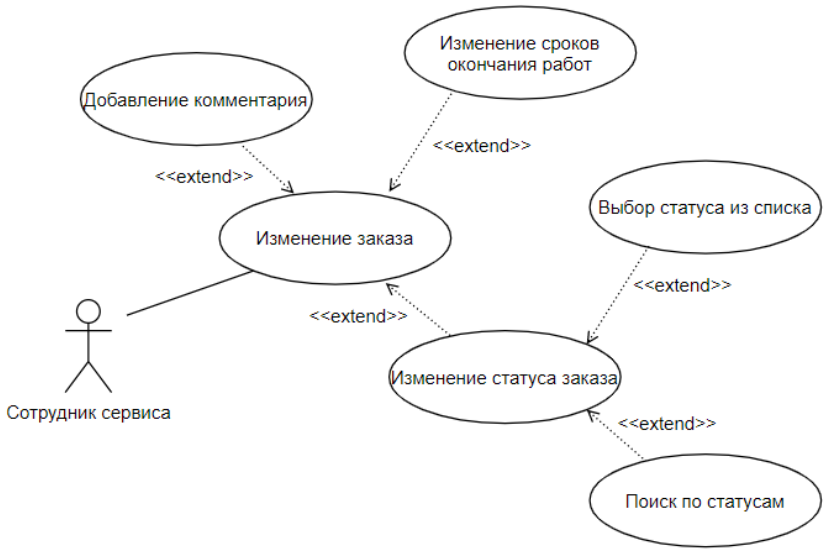


Рисунок 2.1.2.Диаграмма декомпозиции варианта использования «Изменение заказа»

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

1. Ввод ФИО
2. Ввод логина
3. Ввод электронной почты
4. Ввод номера телефона
5. Ввод пароля
    a. Подтверждение пароля

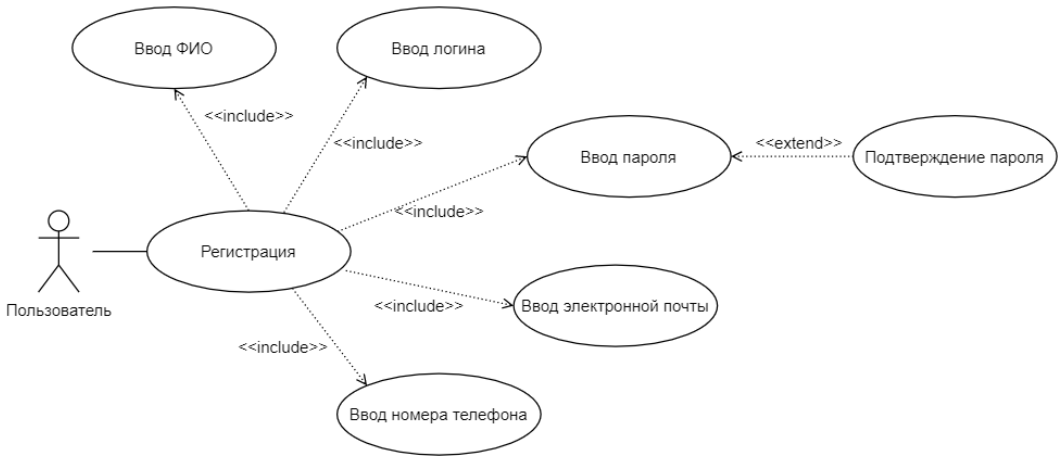


Рисунок 2.1.3. Диаграмма декомпозиции варианта использования «Регистрация»

На диаграмме декомпозиции варианта использования «Оформление заказа на оказание услуг» (рис.2.1.4) показаны основные возможности пользователя в данном разделе:

1. Выбор устройства
    a. Выбор производителя устройства
        i. Выбор производителя устройства из списка
    b. Выбор модели устройства
        i. Выбор модели устройства из списка
2. Выбор типа работ
    a. Выбор типа работ из списка
3. Просмотр стоимости заказа
4. Просмотр сроков выполнения заказов
5. Добавление комментария

Для выполнения действий в данном разделе, пользователю необходимо авторизоваться.

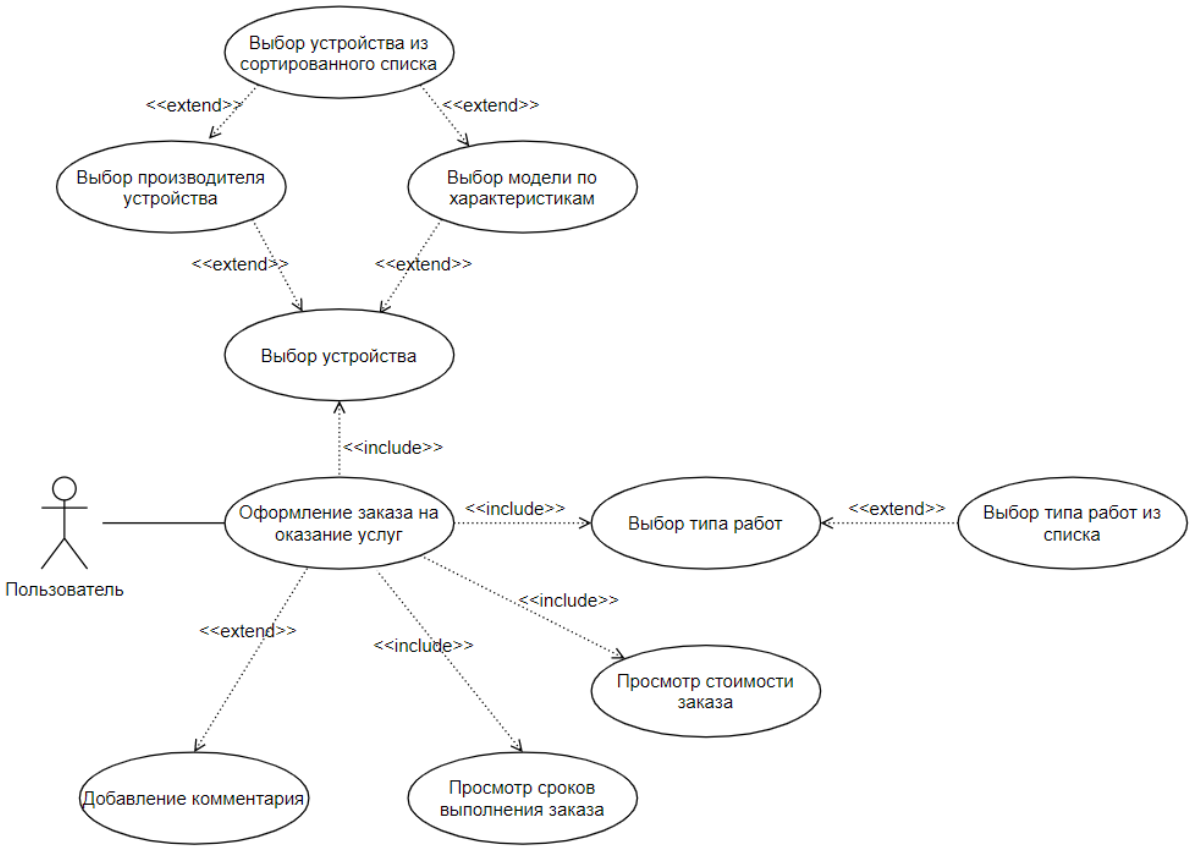


Рисунок 2.1.4. Диаграмма декомпозиции варианта использования «Оформление заказа на оказание услуг»

## 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 совершенно разные пользователи («Пользователи», сотрудники сервиса, администраторы), 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.

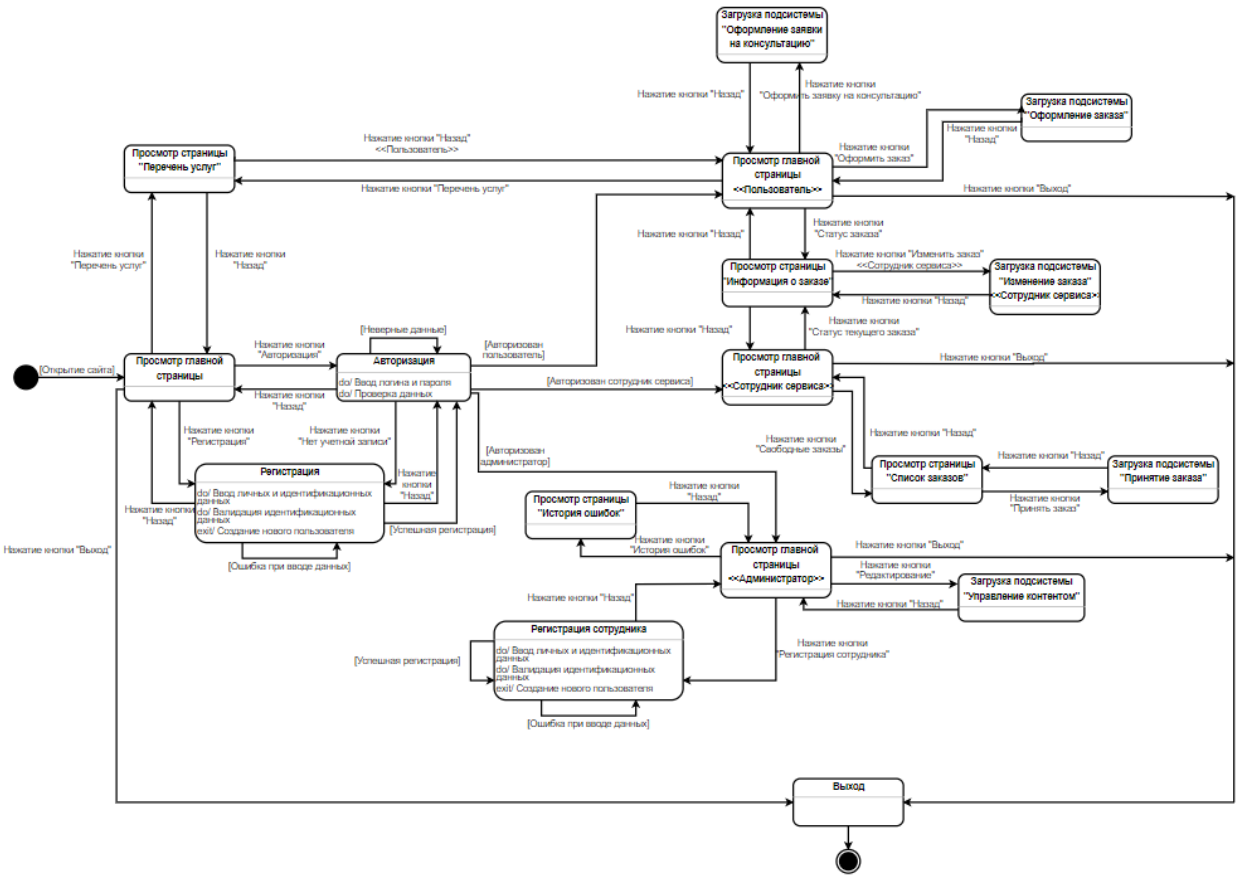


Рисунок 2.2.5. Контекстная диаграмма автоматов.

На диаграмме автоматов подсистемы «Регистрация» (рис. 2.2.2) отображены состояния системы при управлении системой студентом.

Диаграмма более подробно детализирует вариант использования «Регистрация». Сценарий «Регистрация» следующий: пользователь переходит на страницу «Регистрация» либо со страницы «Главная страница», либо со страницы «Авторизация», где при ее загрузке, происходит инициализация формы для ввода регистрационных данных. Далее пользователь должен заполнить все поля формы. Стоит отметить, что на определенных полях, которые имеют особую важность, происходит валидация данных, в случае ошибки при валидации, необходимо скорректировать информацию в этих полях. Также при валидации, например, поля «Логин» происходит асинхронный запрос на сервер, для проверки уникальности логина, путем сравнения его, с логинами уже зарегистрированных пользователей. Также интересной особенностью, является наложение маски на поле «Номер телефона», что облегчает пользователю ввод своего номера телефона.

После того, как данные были введены, происходит асинхронный запрос на сервер для, создание новой записи в БД и после успешного ответа процесс регистрации завершается.

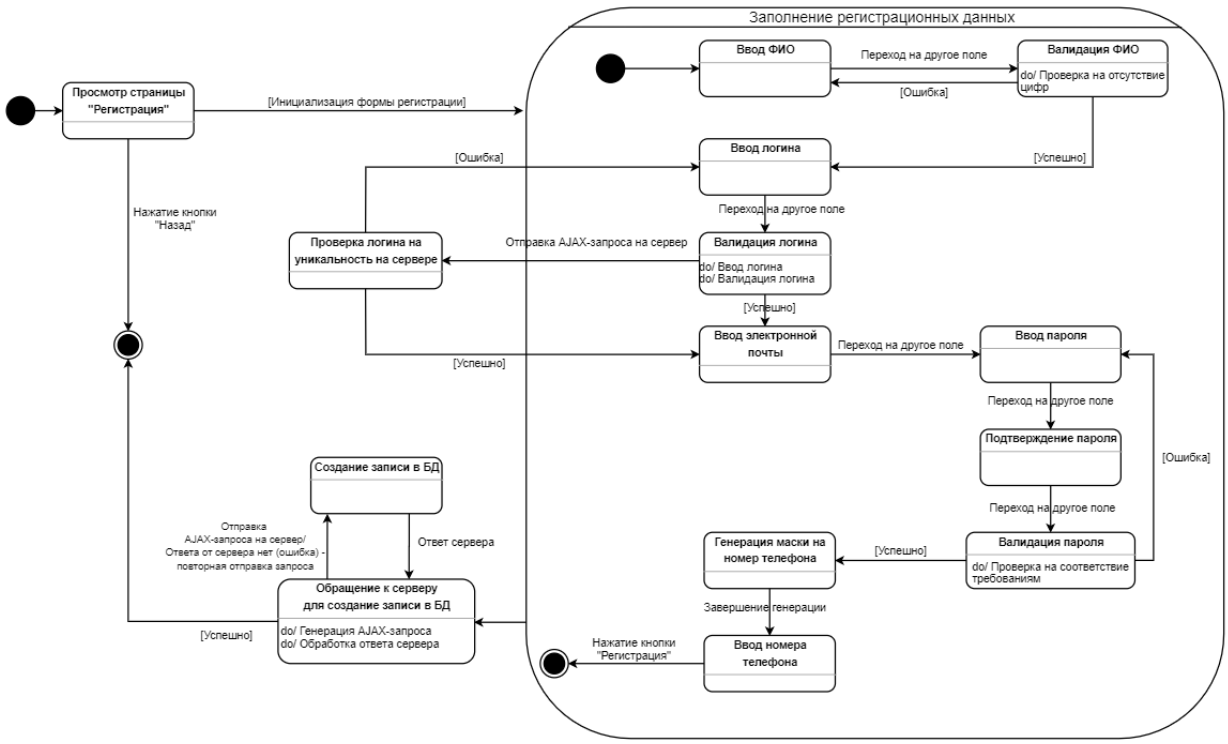


Рисунок 6.2.2. Диаграмма автоматов «Регистрация»

На диаграмме автоматов подсистемы «Оформление заказа на оказание услуг» (рис. 2.2.3) отображены состояния системы при управлении системой студентом. Диаграмма более подробно детализирует вариант использования «Оформление заказа на оказание услуг» (далее «Оформление заказа»). Сценарий «Оформление заказа» следующий: пользователь, прошедший процесс авторизации, с главной страницы может перейти на страницу «Оформление заказа». После чего происходит выгрузка данных из БД, которые необходимы для оформления заявки. Загрузка марок и моделей телефона, а также списка услуг, происходит потому, что рассматриваемый сервис обслуживает ограниченный спектр смартфонов и оказывается ограниченный перечень услуг. После успешной загрузки данных из БД, система ожидает действия пользователя. Пользователь должен заполнить все поля, после того, как система получит достаточно информации, произойдет автоматический расчет стоимости оказания услуг и сроков проведения работ. Также пользователь может дополнительно оставить комментарий в заявке.

После того, как пользователь нажмет кнопку «Создать заявку» система сформирует заявку и отправит асинхронный запрос на сервер, для создания новой записи в БД. После успешного выполнения запроса процесс оформления заявки завершиться.

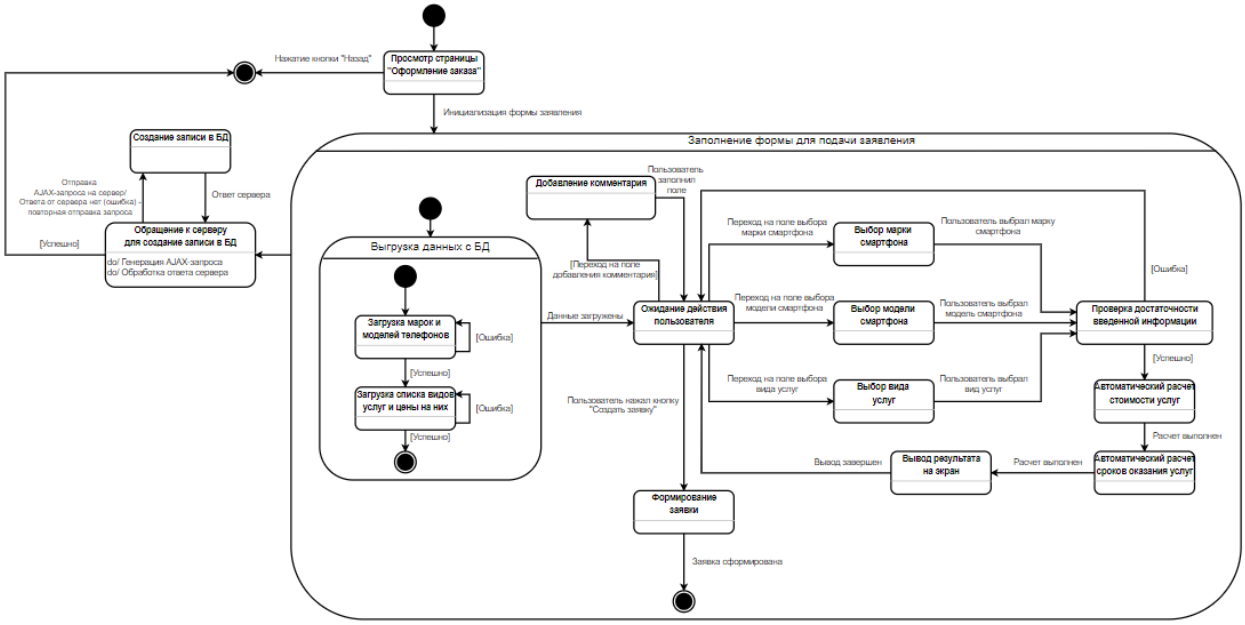


Рисунок 2.2.7. Диаграмма автоматов "Оформление заказа на оказание услуг"

## 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 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.

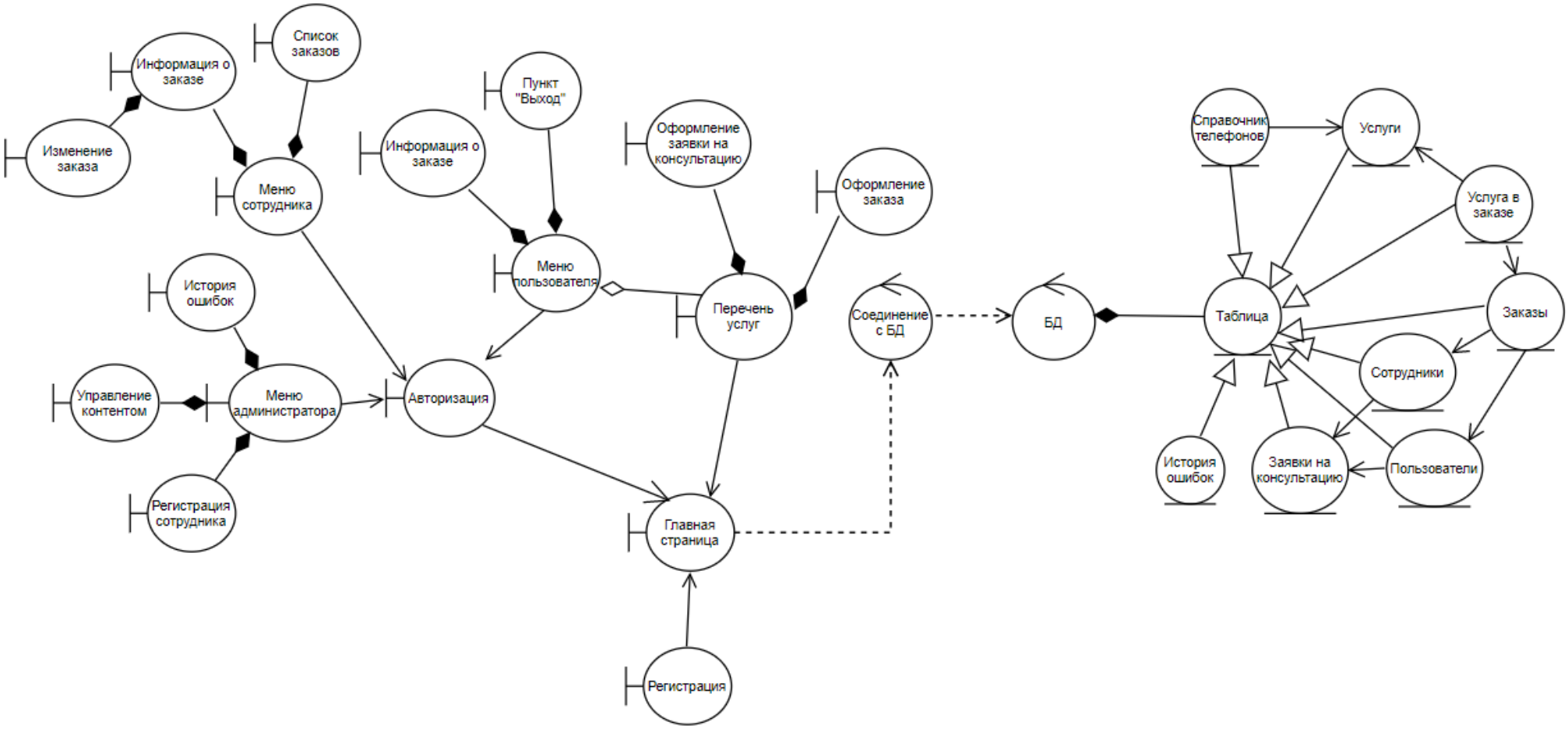


*Рисунок 3.1.8. Диаграмма классов анализа*

### 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 «Оформление заявки». Для того чтобы попасть на страницу «Оформление заявки» пользователю необходимо выбрать советующий пункт меню в шапке сайта. После открытия будет произведен запрос у сервера на получение перечня смартфонов и услуг. Сам сервер обратится к базе данных через провайдера и получит записи. Произведет формирования данных в формат JSON и после этого отправит ответ клиенту. После чего данные будут

отображены на странице. В случае ошибки загрузки, пользователь увидит соответствующее уведомление.

Далее пользователь сможет: выбрать модель и марку смартфона, а также вид необходимых услуг. После любого из этих действий, при наличии достаточной информации будет произведен автоматический расчет стоимости услуг и сроков оказания услуг.

После заполнения всех полей, пользователь может нажать кнопку «Оформить заявку», которая в свою очередь запустит процесс, добавления заявки в БД. Для этого также будет произведена проверка достаточности информации, в случае ошибки пользователь увидит соответствующее уведомление. В случае успеха, будет отправлен запрос на сервер для добавления заявки в БД, а сам сервер в свою очередь через провайдера обратиться к базе данных и создастся новая запись.

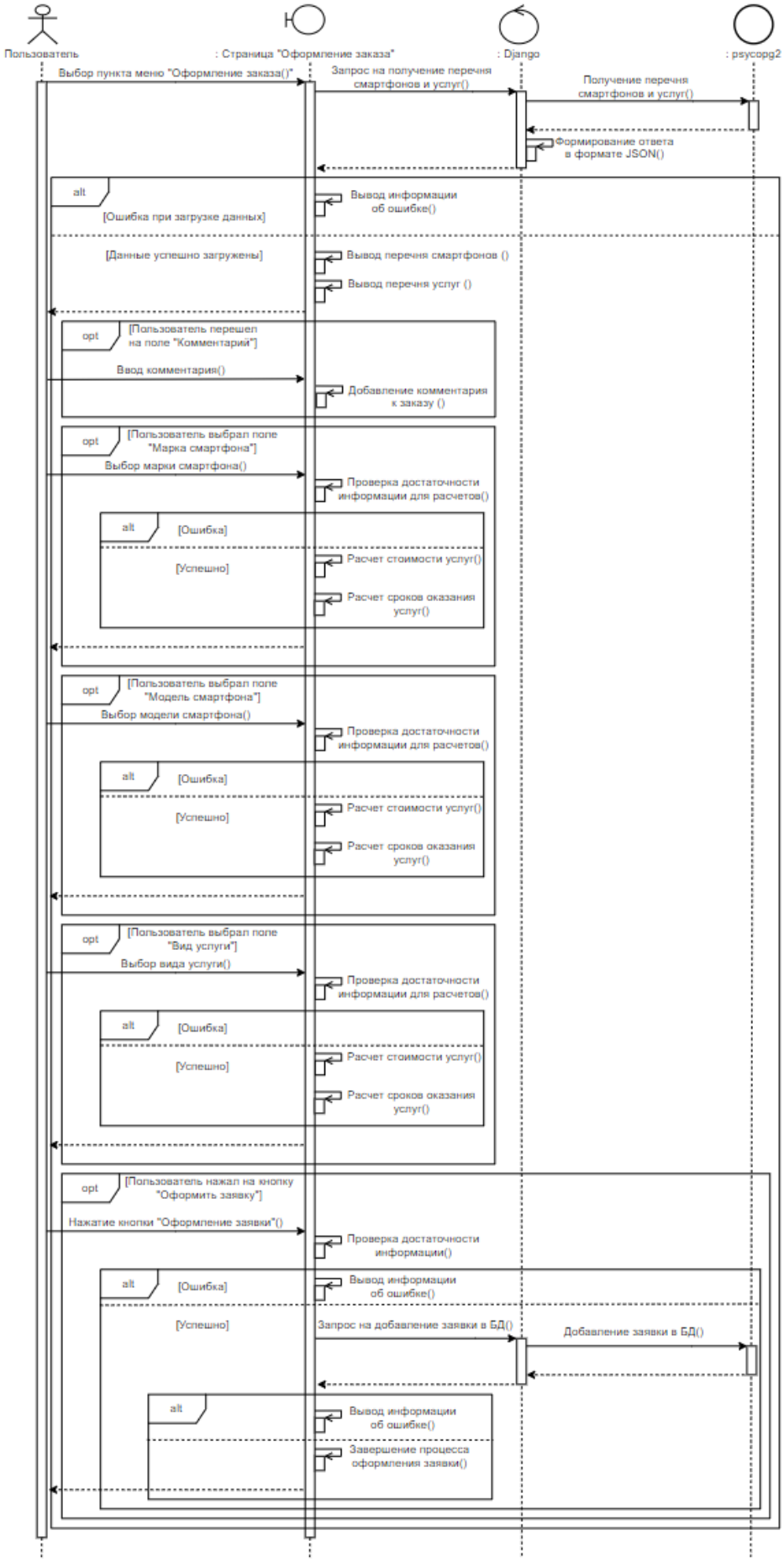


Рисунок 3.2.9. Диаграмма последовательности "Оформление заявки"

На рисунке 3.2.2 представлена диаграмма последовательности «Регистрация»

Чтобы пользователь прошел процесс регистрации ему необходимо перейти на страницу «Регистрация», после чего пользователь вводит регистрационные данные, происходит валидация этих данных, в случае ошибки, пользователю необходимо исправить ошибки, далее необходимо нажать кнопку «Регистрация» страница 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.

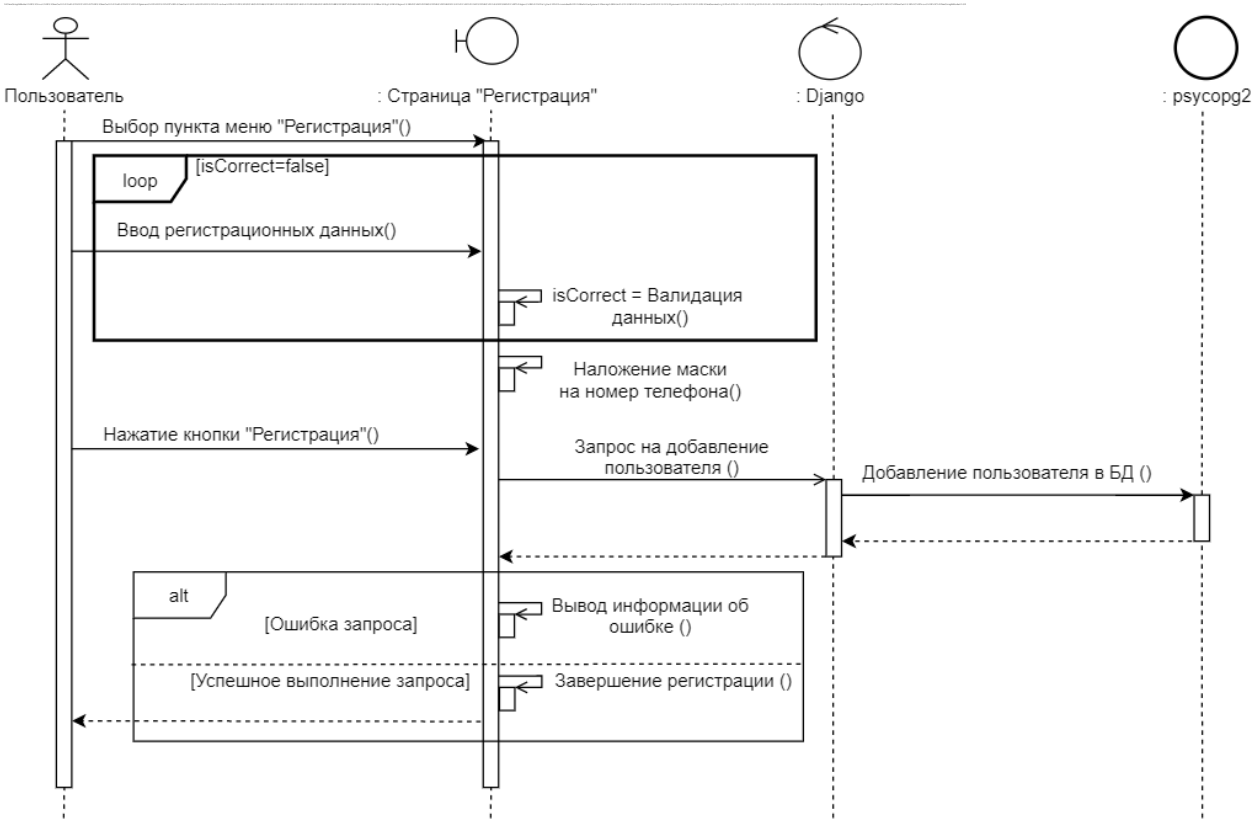


Рисунок 3.2.10. Диаграмма последовательности "Регистрация"

### 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 «Оформление заявки». Данная диаграмма сгенерирована автоматически на основе диаграммы последовательности, изображенной на рисунке 3.2.1.

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

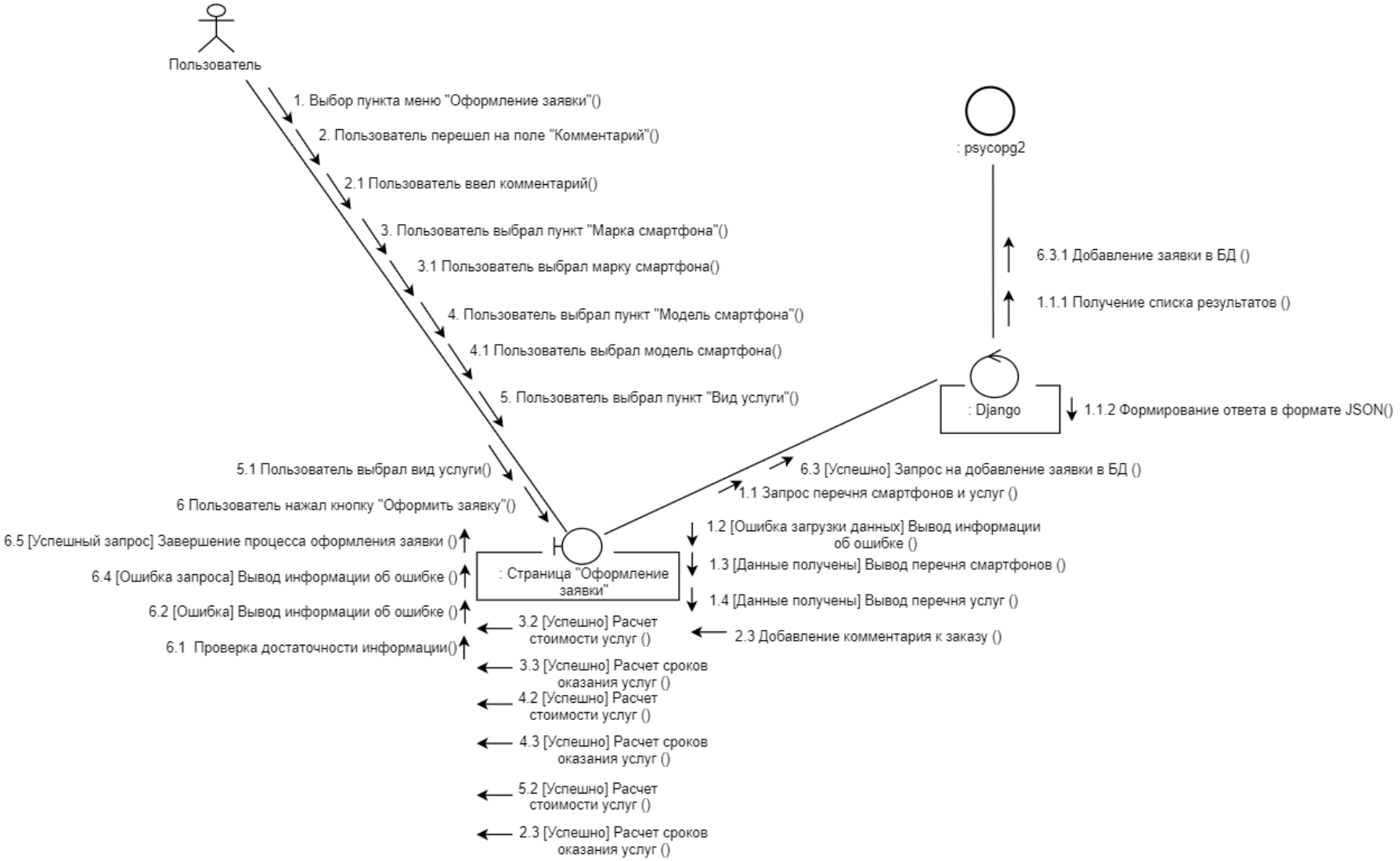


Рисунок 3.3.11. Диаграмма коммуникации "Оформление заявки"

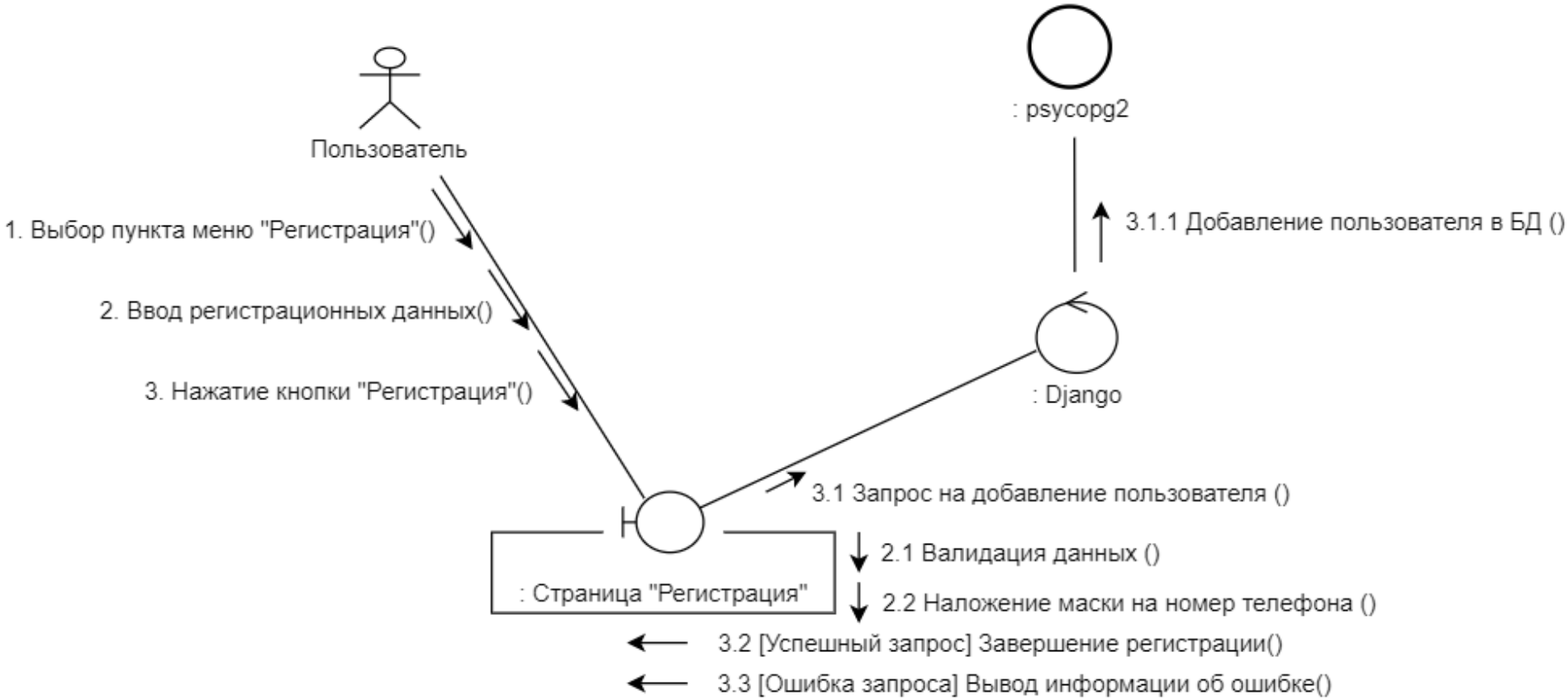


Рисунок 3.3.12. Диаграмма коммуникации "Регистрация"

## 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:

- концептуальная точка зрения – диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;
- точка зрения спецификации – диаграмма классов применяется при проектировании информационных систем;
- 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 |
| Логин | nvarchar | Логин пользователя |
| Пароль | nvarchar | Пароль пользователя |
| Номер телефона | nvarchar | Номер телефона пользователя |
| E-mail | nvarchar | E-mail пользователя |

Таблица 2. Сотрудники (Staff)

| Название поля | Тип данных | Описание |
|---|---|---|
| ID сотрудника | int | Уникальный идентификатор сотрудника |
| Фамилия | nvarchar | Фамилия пользователя |
| Имя | nvarchar | Имя пользователя |
| Отчество | nvarchar | Отчество пользователя |
| Логин | nvarchar | Логин пользователя |
| Пароль | nvarchar | Пароль пользователя |
| Должность | nvarchar | Должность сотрудника |
| Администратор | Boolean | Является ли пользователь администратором |
| Сотрудник | Boolean | Является ли пользователь сотрудником |

Таблица 3. Услуги (Services)

| Название поля | Тип данных | Описание |
|---|---|---|
| ID услуги | int | Уникальный идентификатор услуги |
| ID телефона | int | Ссылка на телефон |
| Цена | int | Стоимость услуги |
| Название | nvarchar | Название услуги |
| Примечание | nvarchar | Примечание к услуге |

Таблица 4. Заказы (Orders)

| Название поля | Тип данных | Описание |
|---|---|---|
| ID заказа | int | Уникальный идентификатор заказы |
| ID пользователя | int | Ссылка на пользователя |
| ID сотрудника | int | Ссылка на сотрудника |
| Статус | nvarchar | Статус заказа |
| Дата создания | Data Time | Дата принятия заказа |
| Дата окончания | Data Time | Дата закрытия заказа |

Таблица 5. Услуга в заказе (Services and Orders)

| Название поля | Тип данных | Описание |
|---|---|---|
| ID услуги в заказе | int | Уникальный услуги |
| ID услуги | int | Ссылка на услугу |
| ID заказа | int | Ссылка на заказ |

Таблица 6. Заявки на консультацию (Consultation orders)

| Название поля | Тип данных | Описание |
|---|---|---|
| ID заявки | int | Уникальный идентификатор заявки |
| ID сотрудника | int | Ссылка на сотрудника |
| ID пользователя | int | Ссылка на пользователя |
| Номер телефона | nvarchar | Номер телефона клиента |
| e-mail | nvarchar | E-mail клиента |
| Статус | nvarchar | Статус заявки |

Таблица 7. История ошибок (Error history)

| Название поля | Тип данных | Описание |
|---|---|---|
| ID ошибки | int | Уникальный идентификатор ошибки |
| Название | nvarchar | Название ошибки |
| Лог | nvarchar | Лог ошибки |
| Дата и время | Data Time | Дата и время ошибки |

Таблица 8. Справочник телефонов (Phone)

| Название поля | Тип данных | Описание |
|---|---|---|
| ID телефона | int | Уникальный идентификатор модели |
| Название | nvarchar | Название телефона |
| Модель | nvarchar | Модель телефона |
| Производитель | nvarchar | Производитель телефона |

Далее приведены диаграммы классов для БД (рис. 4.1.3 и рис. 4.1.4) и диаграммы классов для приложения (рис. 4.1.1 и рисунок 4.1.2). 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).

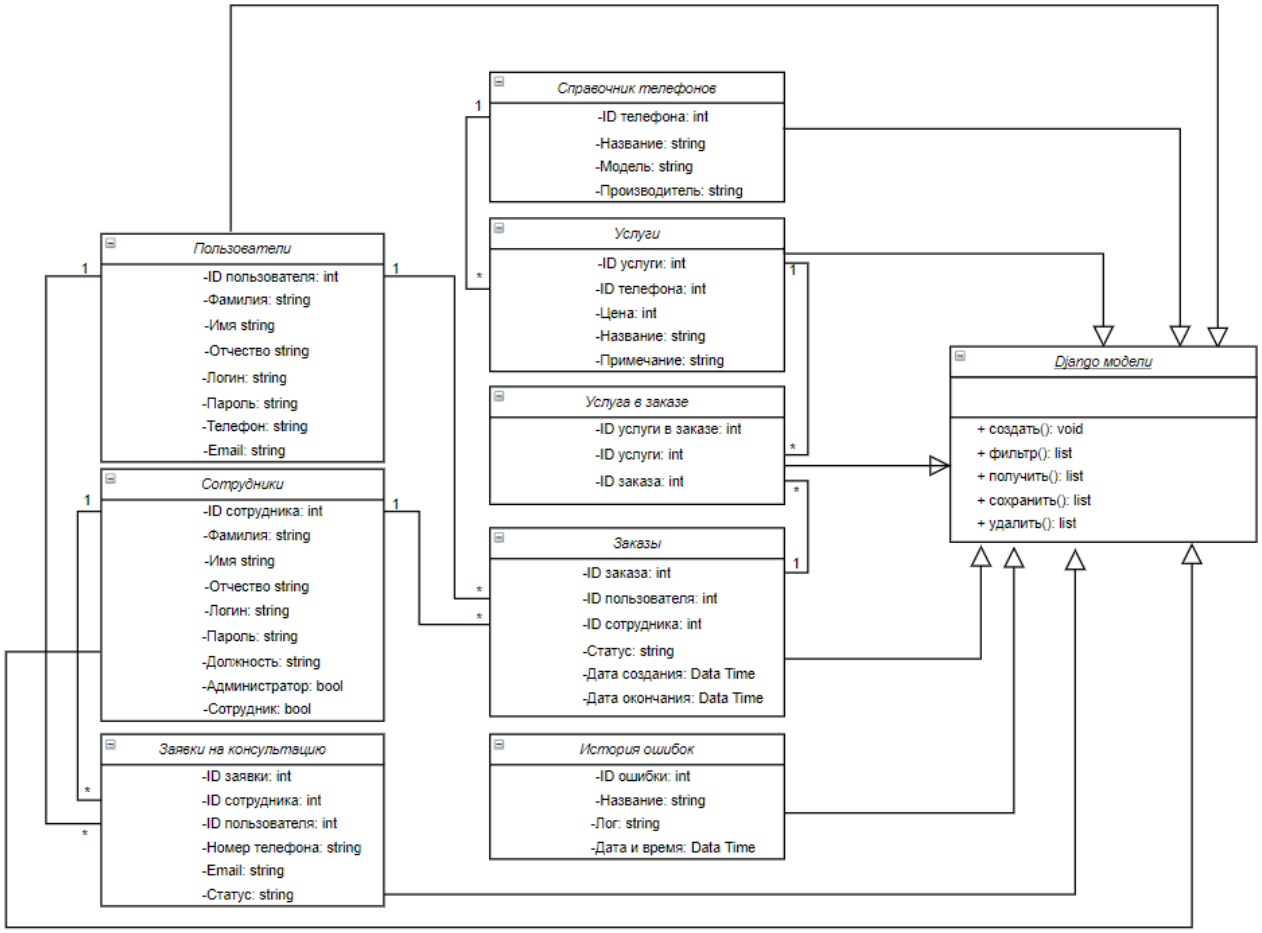


Рисунок 4.1.1. Логическая диаграмма классов приложения

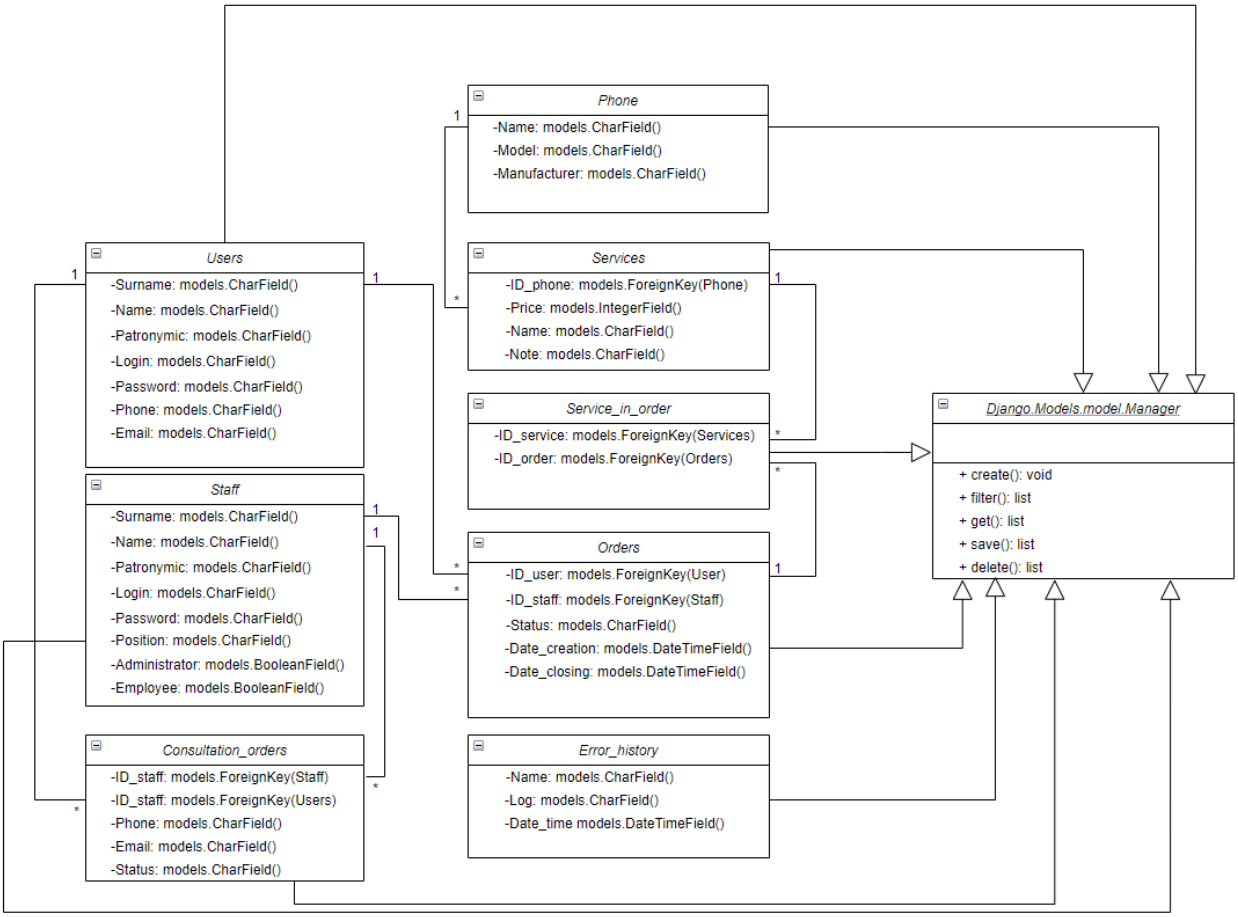


Рисунок 4.1.2. Физическая диаграмма классов приложения

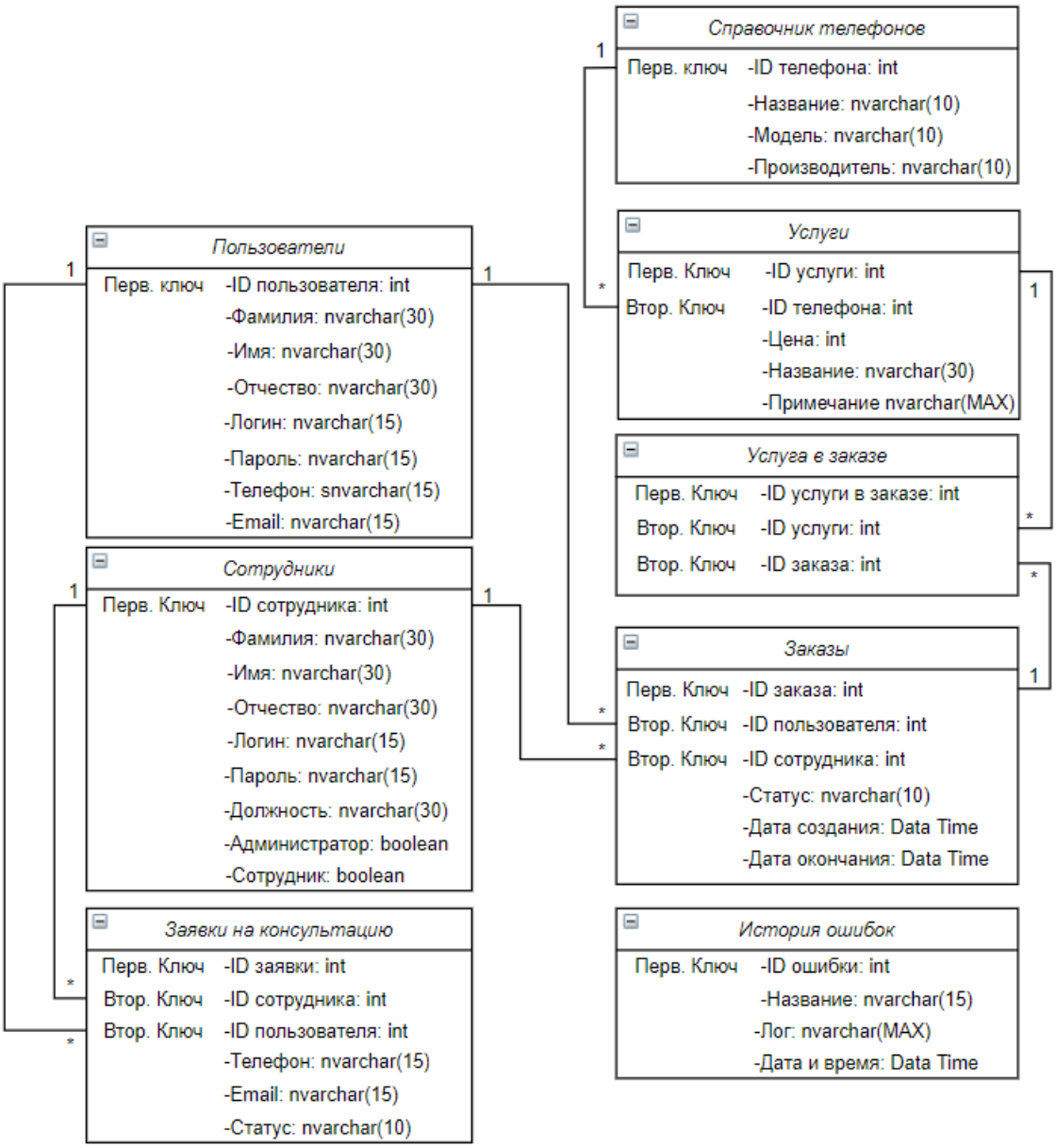


Рисунок 13.1.3. Логическая диаграмма классов базы данных

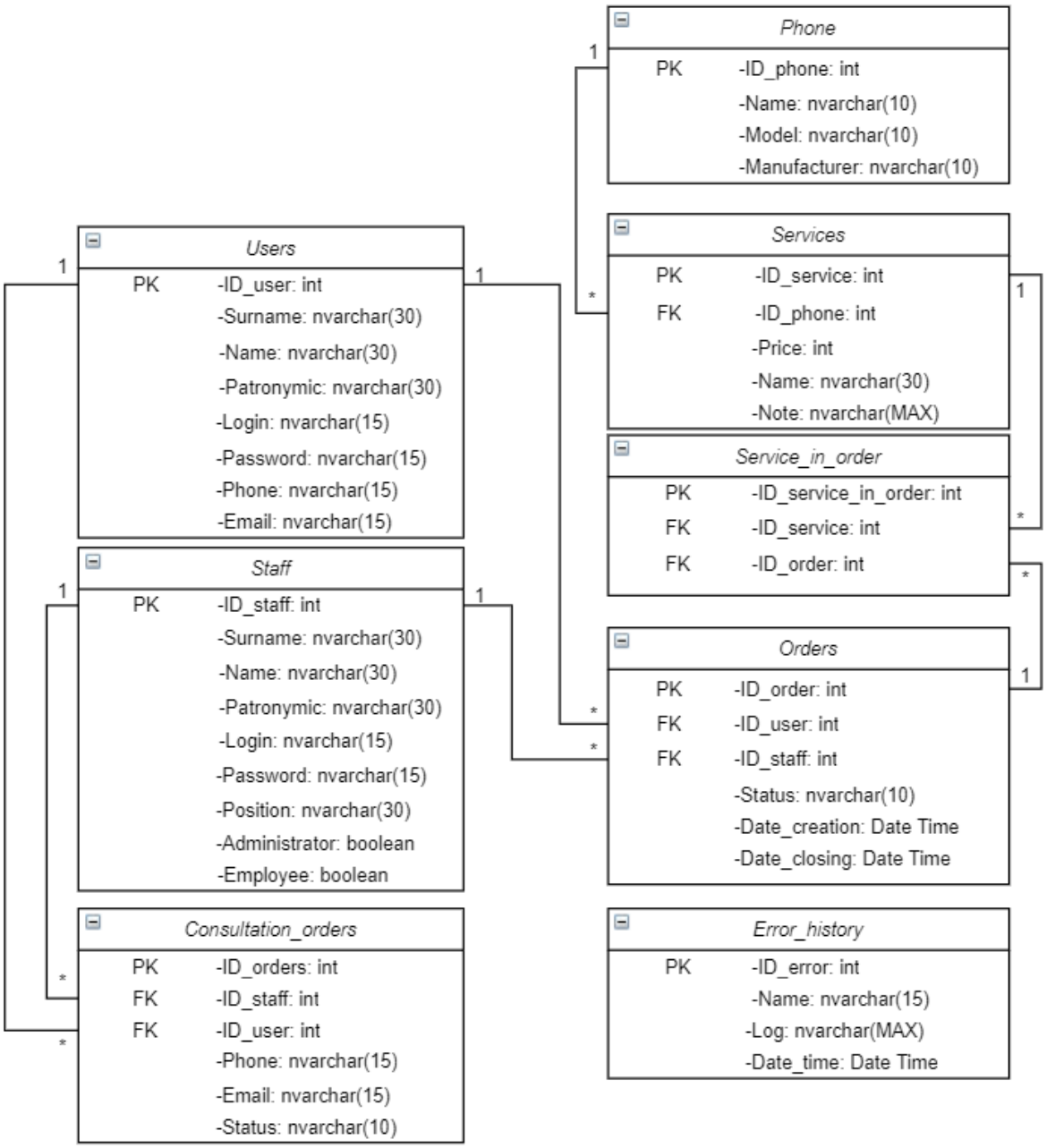


Рисунок 4.1.4. Физическая диаграмма классов базы данных

## 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 валидации пароля. На ней наглядно видно взаимодействие между разделами деятельности.

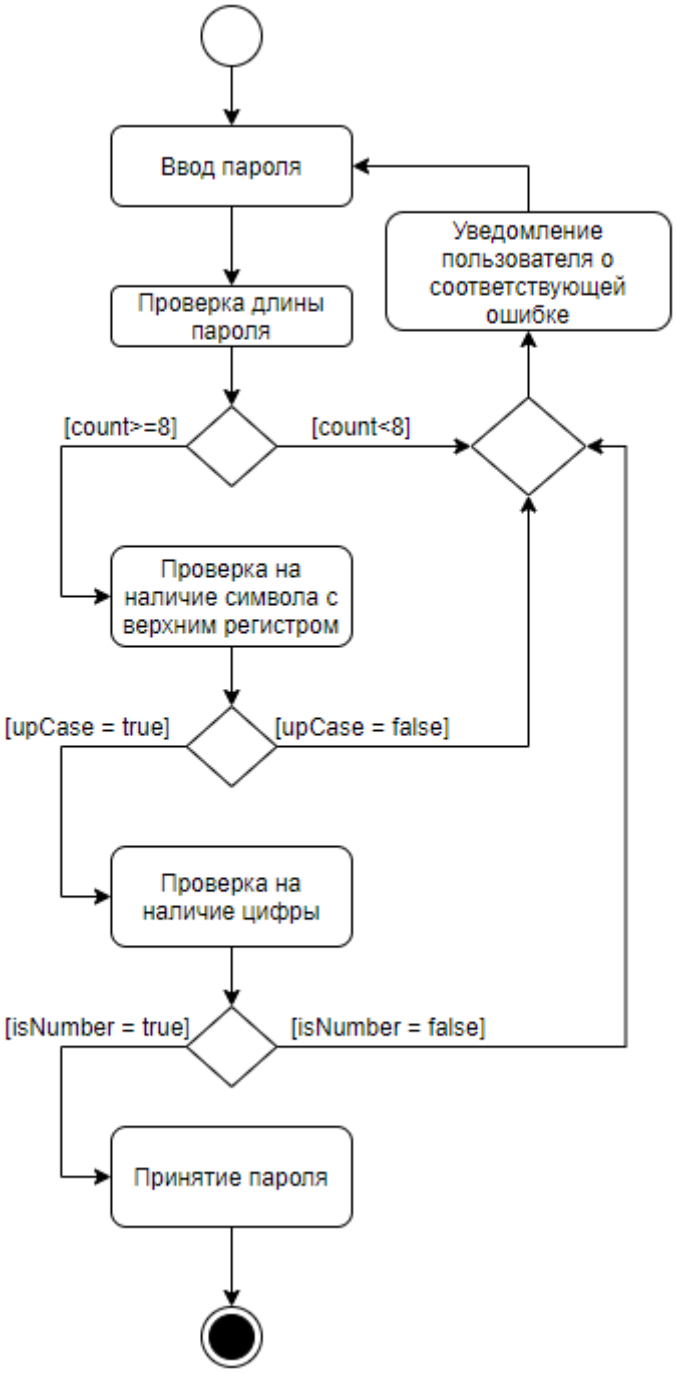


Рисунок 4.2.1. Диаграмма деятельности, описывающая процесс валидации пароля

На рисунке 4.2.2 изображена диаграмма деятельности, описывающая процесс регистрации пользователя. На ней наглядно видно взаимодействие между разделами деятельности.

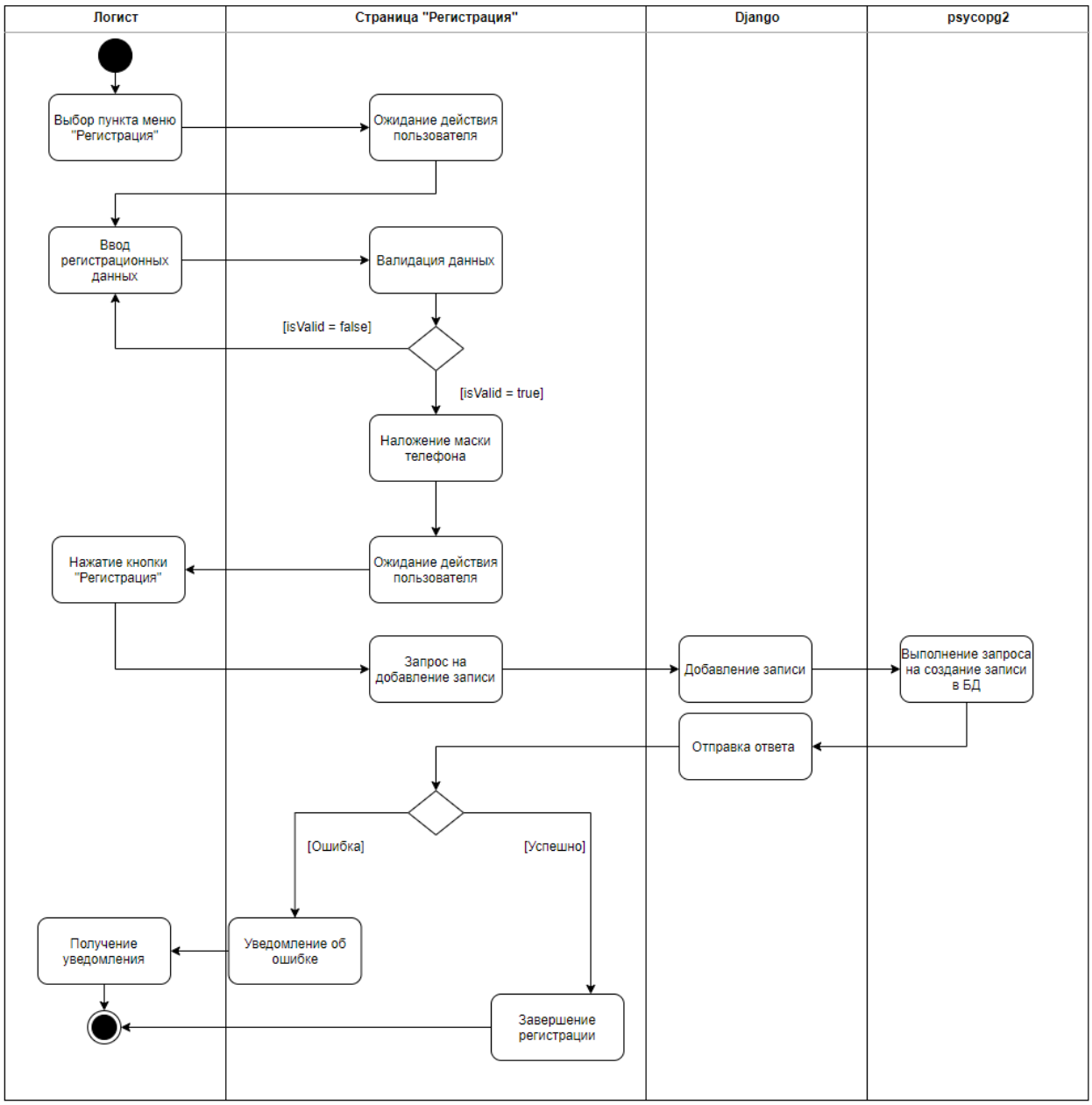


Рисунок 4.2.1. Диаграмма деятельности, описывающая процесс регистрации пароля

## 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 серверного приложения.

На рисунке 5.1.2 показана диаграмма компонентов серверного приложения.

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.

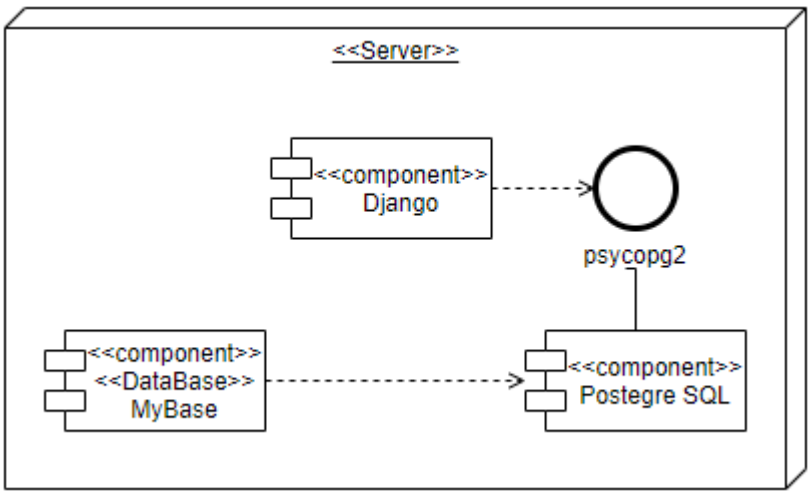


Рисунок 5.1.1. Диаграмма компонентов серверного приложения

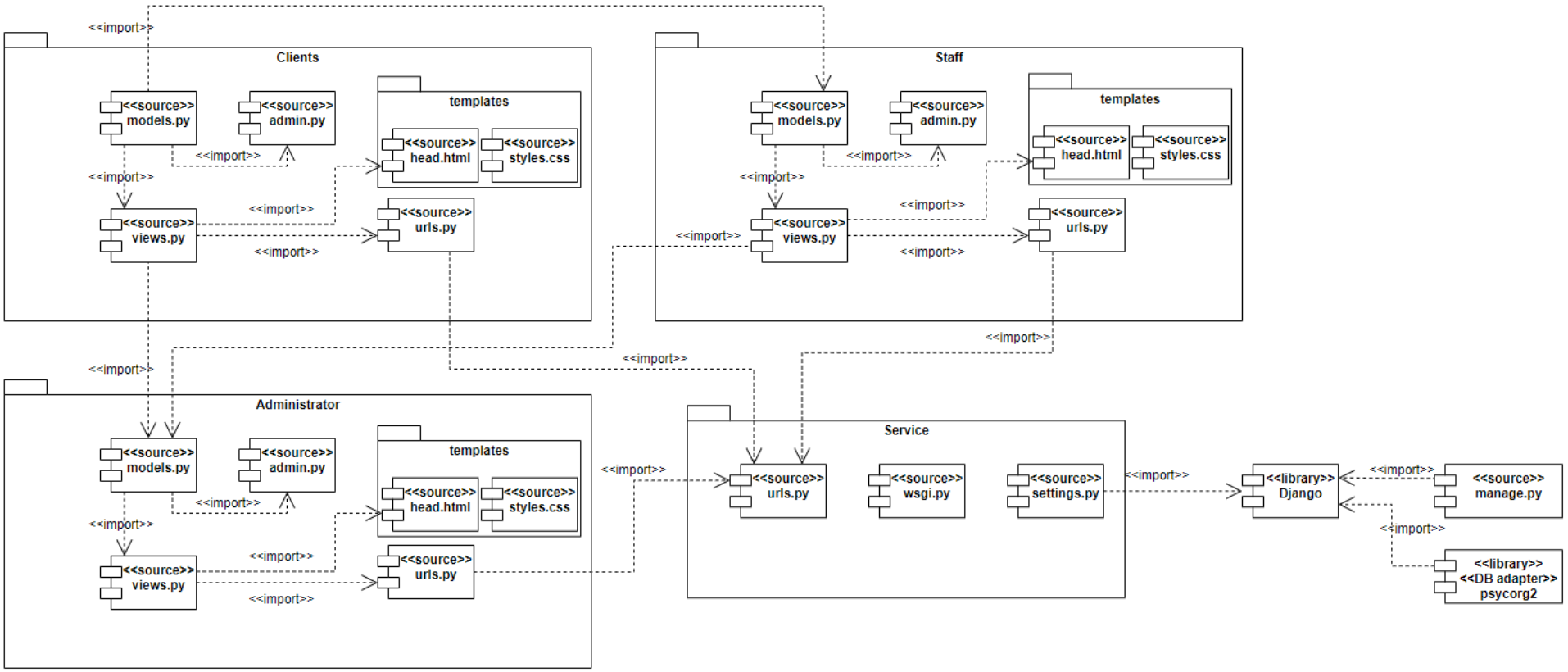


Рисунок 5.1.2. Диаграмма компонентов клиентского приложения

### 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 и база данных устанавливаются на сервер.

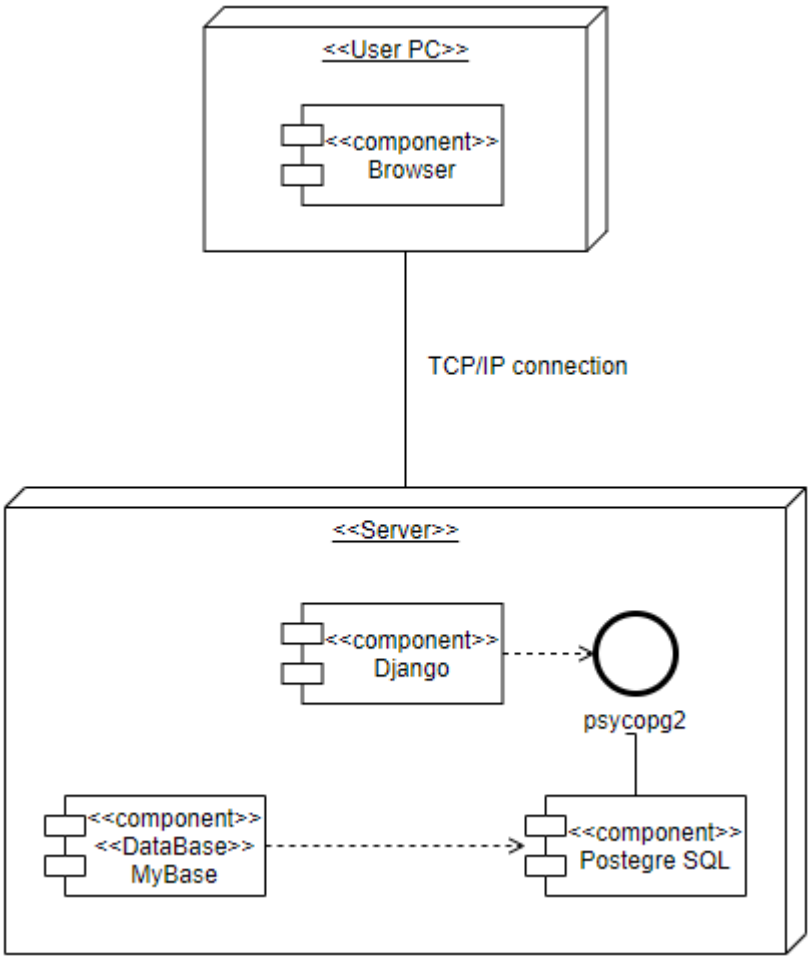


Рисунок 5.2.1 Диаграмма развертывания ИС

## 6. Сгенерированный программный код

```
class Users(models.Model):
    Surname = models.CharField()
    Name = models.CharField()
    Patronymic = models.CharField()
    Login = models.CharField()
    Password = models.CharField()
    Phone = models.CharField()
    Email = models.BooleanField()
```

```
class Staff(models.Model):
    Surname = models.CharField()
    Name = models.CharField()
    Patronymic = models.CharField()
    Login = models.CharField()
    Password = models.CharField()
    Position = models.CharField()
    Administrator = models.BooleanField()
    Employee = models.BooleanField()

class Consultation_orderls(models.Model):
    ID_staff = models.ForeignKey(Staff)
    ID_staff = models.ForeignKey(Users)
    Phone = models.CharField()
    Email = models.BooleanField()
    Status = models.CharField()


class Phone(models.Model):
    Name = models.CharField()
    Model = models.CharField()
    Manufacturer = models.CharField()


class Services(models.Model):
    ID_phone = models.ForeignKey(Phone)
    Price = models.IntegerField()
    Name = models.CharField()
    Note = models.CharField()


class Service_in_order(models.Model):
    ID_service = models.ForeignKey(Services)
    ID_order = models.ForeignKey(Orders)

class Orders(models.Model):
    ID_user = models.ForeignKey(User)
    ID_staff = models.ForeignKey(Staff)
    Status = models.CharField()
    Date_creation = models.DateTimeField()
    Date_closing = models.DateTimeField()
```

```
class Error_history(models.Model):
    Name = models.CharField()
    Log = models.CharField()
    Date_time = models.DateTimeField()
```

## 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.

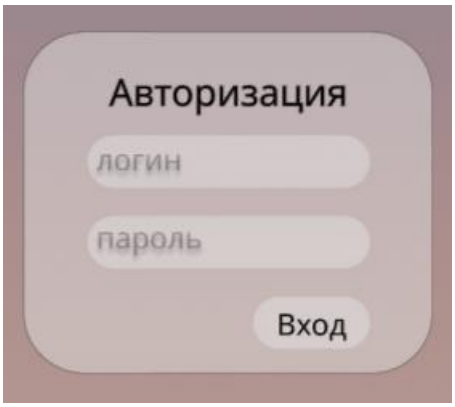


Рисунок 7.1.1. Форма авторизации

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

### 7.2 Главная страница

Функционал программы зависит от того, кто был авторизирован. Так, например, если процесс авторизации прошел пользователь, он увидит главную страницу в следующем формате (рис 7.2.1). Тут мы видим функционал, который доступен пользователю.

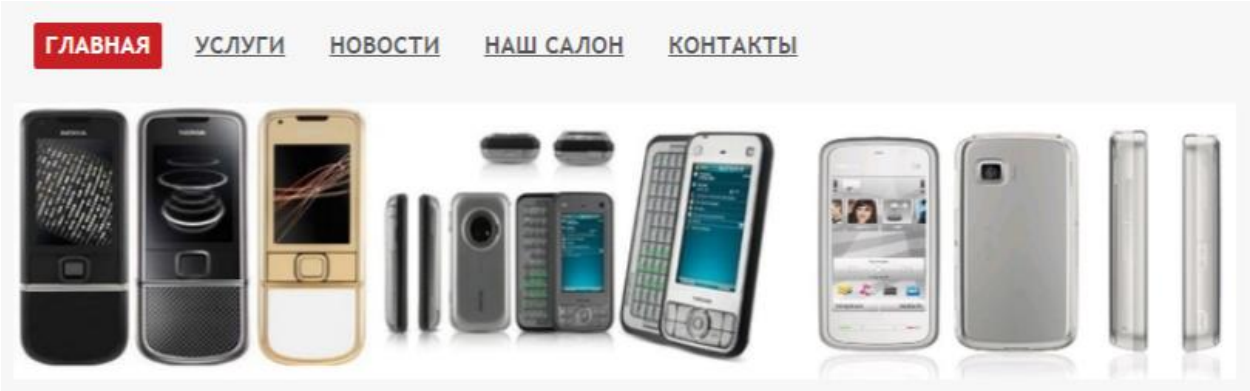


Рисунок 7.2.1 Главная страница пользователя

Действия пользователя могут быть различными:

1. Просмотр услуг;
2. Оформление заказа;
3. Оформление заявки на обратную связь;
4. Просмотр заказа;
5. Выйти из системы, нажав кнопку «Выход».

На рисунке 7.2.2 изображена главная страница, для пользователя с ролью администратор. Из него видно, что пользователь при авторизации автоматически попадает на страницу просмотра списка пользователей.

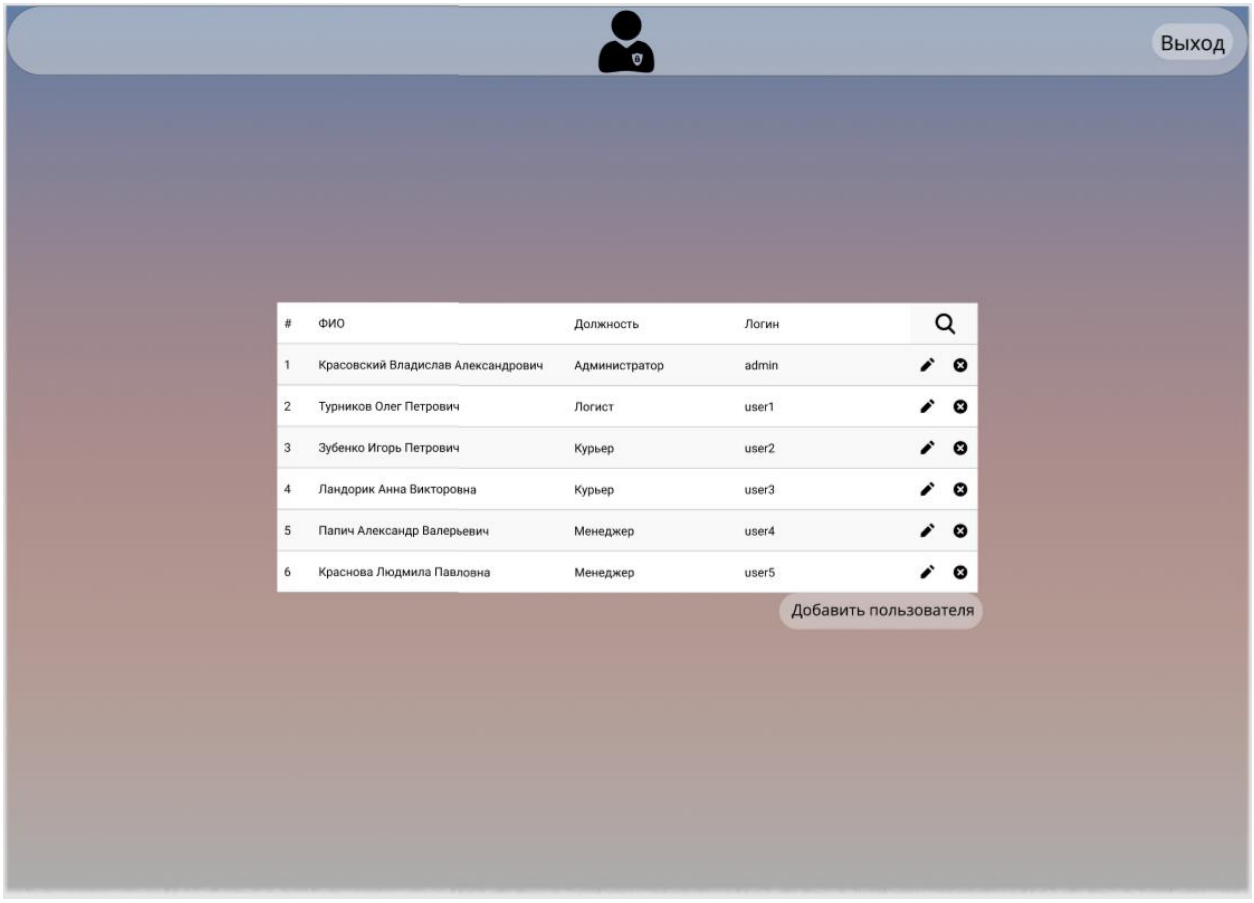


Рисунок 7.2.2. Главная страница администратора

Действия администратора могут быть следующими:

1. Просмотр конкретной учётной записи, нажав кнопку карандаша на нужной учётной записи в таблице;
2. Сортировка таблицы, нажав на необходимый столбец таблицы;
3. Поиск записи в таблице, нажав на кнопку поиска;
4. Добавление новой учётной записи, нажав на кнопку «Добавить пользователя»;
5. Выйти из системы, нажав кнопку «Выход».

### 7.3 Процесс работы пользователя

Процесс работы пользователя сводится к заполнению форм на страницах: «Создание заказа» (Рисунок 7.3.1), «Обратная связь» (Рисунок 7.3.2)

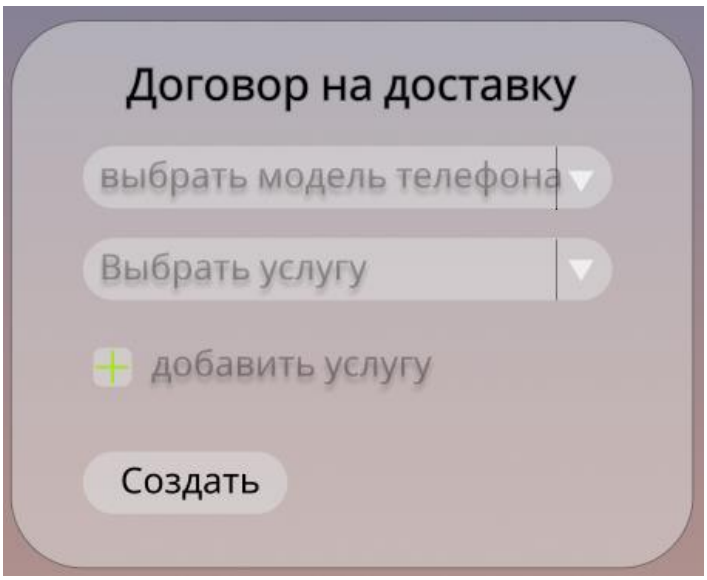


Рисунок 7.3.1. Форма создание заказа

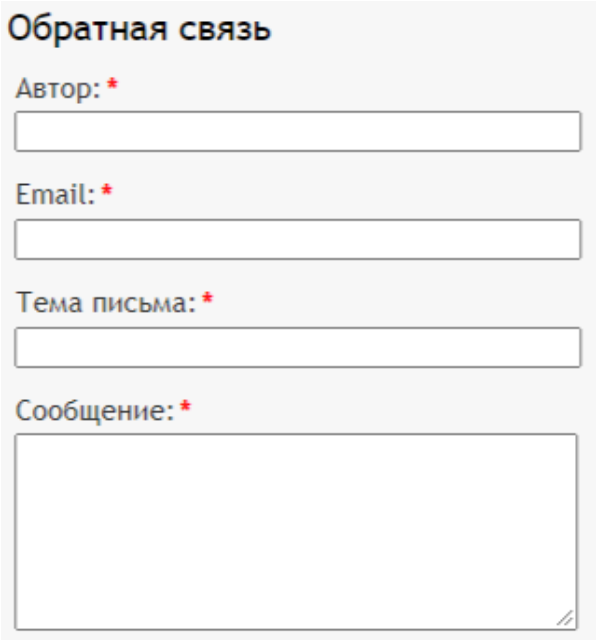


Рисунок 7.3.2. Форма обратной связи

Во всех случаях пользователю необходимо заполнить все поля на форме и нажать кнопку «Создать». При успешном создании, появиться соответствующее уведомление.

Для просмотра своего обращения и созданной заявки необходимо выбрать его в списке в своём профиле.

### 7.4 Процесс работы администратора

Процесс работы администратора сводится к заполнению формы на странице «Создание пользователя» (Рисунок 7.4.1))

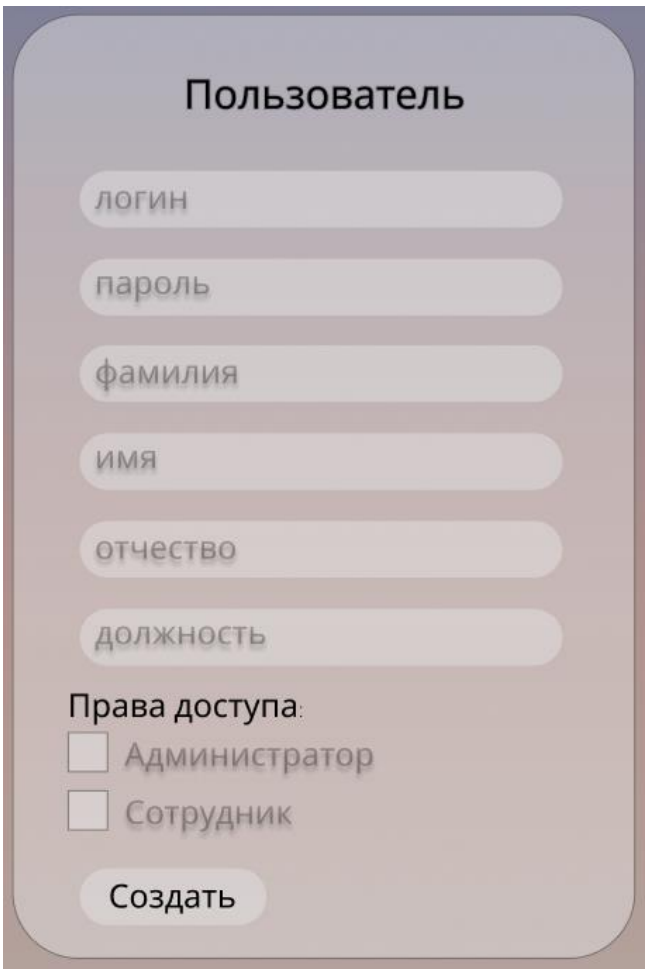


Рисунок 7.4.1. Форма создания пользователя

Администратору необходимо заполнить все поля на форме и нажать кнопку «Создать». При успешном создании, появиться соответствующее уведомление.

Если необходимо изменить существующую запись, необходимо выбрать её в общей таблице. Далее откроется форма, упомянутая выше, с уже заполненными значениями.

**Заключение**

Целью данного курсового проект являлась разработка проекта информационной системы «Сервисный центр по ремонту смартфонов»

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 реальном примере [Текст] / Г. А. Гурвиц. Санкт-Петербург : БХВ-Петербург, 2010. - 493 с.