ВВЕДЕНИЕ

На сегодняшний день возможность делегирования рутинных обязанностей является неотъемлемой частью повседневной жизни людей, которые ценят своё время. К счастью, в настоящее время создано большое количество сервисов, позволяющих упросить нашу жизнь. Клининг является одним из таких упрощений. Клининг – это комплекс мер, обеспечивающих чистоту офисов, торговых комплексов, отелей, коттеджей, квартир и т.д. Использование автоматизированной системы позволяет ускорить процесс оформления заказов и подойти к этому оформлению более гибко.

Благодаря автоматизации можно достигнуть большего упрощения во взаимодействии между клиентами и персоналом, а также обеспечить получение обратной связи по каждому заказу. Элементы автоматизации подразумевают оформление заказа, отслеживание статуса заказа и систему отзывов.

В настоящее время многие рутинные обязанности уже делегированы или находятся на пути к этому. Например, доставка продуктов или доставка рациона питания. Таким образом можно сэкономить много времени для клиентов.

Цель данной работы – разработка автоматизированной информационной системы работы клининговой компании.

1. Постановка задачи и обоснование метода решения задачи
   1. Описание предметной области и постановка задачи

Предметная область – это часть реального мира, которая подлежит изучению с целью автоматизации организации управления. Любая предметная область может быть разбита на фрагменты. Каждый фрагмент оперирует со своими объектами и с множеством пользователей, которые имеют свои взгляды на предметную область, поэтому выявление предметной области и ее анализ является неотъемлемой частью разработки любой информационной системы [1].

Клининг – это комплекс действий, необходимый для обеспечения чистоты в промышленных, коммерческих и бытовых помещениях. Помимо технического обслуживания и аспектов гигиены, она также включает в себя контроль безопасности и порядка, а также улучшения с эстетической точки зрения [2].

* + 1. Типы уборки

По критерию проводимых мероприятий уборки делится на сухую и влажную. Из названия уже понятно, что во втором случае она проводится с использованием воды.

* + 1. Виды клининга

По частоте проведения уборка делится на четыре вида:

- ежедневная;

- еженедельная;

- генеральная;

- послестроительная.

Каждый из перечисленных видов клининга применяется в зависимости от типа помещения, частоты проведения, тщательности обработки объекта и, конечно, от требований к чистоте у жильцов.

1.1.3 Тип убираемого объекта

Представленная автоматизированная система предоставляет пользователям возможность выбирать тип объекта для уборки, включая участки, частные дома, квартиры или гаражи. Это обеспечивает гибкость и персонализированный подход к выполнению работы.

1.1.4 Процесс заказа клининга

Оформление заказа на уборку обычно состоит из следующих этапов:

- выбор вида клининга: пользователь выбирает, насколько тщательную уборку хотелось бы провести;

- выбор типа объекта: выбирается тип убираемого объекта для понимания клинером деталей работы;

- определение площади уборки: данный шаг уточняет объем уборки, чтобы клинеру было удобнее рассчитать количество моющих средств, время работы;

- определение удобной даты уборки: у пользователя есть возможность гибко управлять временем и вызвать уборку в любой удобный промежуток времени;

- выбор клинера: некоторым пользователям важно выбирать специалиста, с которым у них уже есть опыт сотрудничества или с которым они чувствуют наилучшее взаимопонимание. Этот функционал обеспечивает персонализированный подход к обслуживанию и удовлетворяет предпочтения клиентов, повышая качество обслуживания.

1.1.5 Процесс обратной связи

Обратная связь – в самом широком понимании ответная реакция человека или группы людей на получаемую информацию или совершаемое действие [3]. Обратная связь помогает решать ряд следующих задач:

- вдохновлять, мотивировать и развивать человека;

- устранять ошибки и избегать их в будущем;

- корректировать бизнес-процессы;

- искать новые идеи для развития.

В современных условиях зачастую удобно опираться на объективную оценку о какой-либо сущности. Например, при покупке телевизора можно обратиться к ранее оставленной обратной связи о качестве этого товара (в данном случае – отзывам), чтобы получить впечатления и объективные оценки людей. На основании отзывов можно делать решение о том, действительно необходим именно этот товар или эта услуга.

Для получения обратной связи проводят опросы и интервью, составляют анкеты и квизы, мотивируют написать отзыв – например, в обмен на скидку. Чтобы у пользователя не пропало желание дать обратную связь, нужны удобные каналы связи.

Возможность оперативно дать обратную связь бизнесу позволяет предупреждать конфликтные ситуации и появление негативных отзывов на внешних площадках [4].

В разрабатываемом приложении процесс обратной связи реализован через систему отзывов. Пользователь вправе оставить отзыв к каждому заказу, при этом не только в текстовом формате, но и в виде фотографий и оценки по пятибальной шкале.

1.2 Анализ существующих методов решения и обоснование выбора реализуемого метода

Существует два основных способа заказа клининга: по номеру телефона и через приложение. Каждый из способов имеет свои плюсы и минусы для каждого типа граждан. Рассмотрим каждый из описанных способов подробнее.

1.2.1 Заказ клининга по номеру телефона

Несмотря на век быстроразвивающихся технологий, из жизни людей не уходят более привычные способы взаимодействия с сервисами. Один из этих способов - коммуникация через мобильную связь.

Описываемый способ заказа клининга хоть и устарел, но имеет ряд преимуществ:

- скорость оформления заказа;

- простота использования;

- доступность всем возрастам;

- отсутствие необходимости в интернет подключении.

Из вышеперечисленных положительных аспектов данного метода можно сделать вывод, что он предпочтителен для более старшего поколения.

1.2.2 Заказ клининга через автоматизированную систему

Данный метод появился в результате прогрессивного развития ИТ-технологий и представляет собой отлаженную систему для взаимодействия клининговой компании и конечного клиента. Поскольку описываемый способ заказа клининга опирается на цифровую инфраструктуру, он имеет ряд достойных преимуществ:

- более гибкая и персонализированная настройка параметров заказа;

- возможность изучить все вариации уборки перед оформлением заказа;

- отсутствие необходимости в операторе, принимающем заказы;

- клиент может наглядно отслеживать заказ;

- предоставляется возможность выбора любого клинера, а также оставить ему отзыв после проделанной им работы.

Таким образом, подход с использованием автоматизированной информационной системы имеет подавляющие преимущества как для клиента, так и для бизнеса. Вдобавок к этому, информационную систему можно развивать и интегрировать с другими сервисами автоматизации, что позволит умножить преимущество этого метода

* 1. Обзор имеющихся систем-аналогов

1.3.1 ООО «Первая клининговая служба»

ООО «Первая клининговая служба» - это команда профессионалов с пятнадцатилетним опытом работы в сфере клининговых услуг. Компания занимается профессиональной уборкой квартир и частных домов, офисов и помещений разного назначения, чисткой мебели, ковров и мойкой окон в Москве и Московской области [5].

В качестве преимуществ данного сервиса хочется отметить чрезвычайно прозрачное ценообразование, а также подробный перечень услуг и обязанностей для каждого вида уборки. На рисунке 1 представлен раздел ценообразования данной программы. В совокупности с демократичными ценами пользователь остается крайне заинтересован к заказу клининга именно в этом приложении.

Однако, как у любой другой системы, ООО «Первая клининговая служба» не обошлась и без них. Первоочередный нюанс заключается в неудачном расположении разделов приложения. Начиная просматривать один раздел, случайно можешь перейти на другой, совсем не интересующий. Вдобавок к этому не совсем приемлемо выбраны цветовые решения и спроектирован дизайн приложения, что в совокупности с предыдущим недостатком создает неинтуитивную и непонятную навигацию по сайту.



Рисунок 1 – Ценообразование ООО «Первая клининговая компания»

1.3.2 Qlean

Qlean – это сервис, предоставляющий книнговые услуги по Москве и Московской области. Компания предлагает опытных клинеров, гарантию качества и надёжность уборки, а также честную цену и скидки для постоянных клиентов. Qlean имеет веб-сайт, через который будет удобно заказать клининг, работая за компьютером, и мобильное приложение, через которое можно не только заказывать клининг, но и узнавать об акциях и специальных предложениях. Интерфейс системы продемонстрирован на рисунке 2.

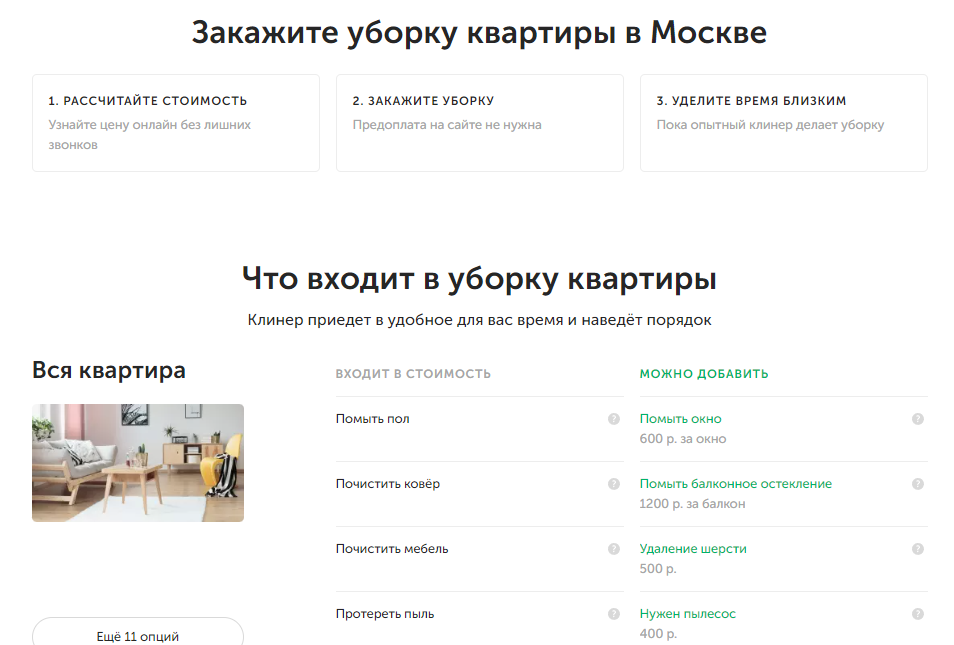


Рисунок 2 – Интерфейс системы Qlean

Из явных преимуществ можно выделить приятный интерфейс одностраничного сайта. Всю необходимую для заказа информацию можно получить порционно, пролистывая сайт. Вдобавок к этому подобраны не броские цвета, что создает приятное ощущение при взаимодействии с интерфейсом системы. Немаловажное преимущество – понятное ценообразование: некоторые услуги уже включены в стоимость уборки, но также существует ряд услуг за дополнительную плату. Рисунок 3 прекрасно описывает ценообразование в данной системе.

Говоря о минусах, можно подметить, что в системе отсутствует возможность выбора определенного клинера. Это немаловажный фактор, который хорошо сказывается как на клиенте, так и на сотруднике. Также у системы отсутствует возможность заказать уборку через телефонный звонок. Пусть эта возможность довольна редкая, однако некоторому слою населения удобно выстраивать коммуникацию по средствам телефонной связи.

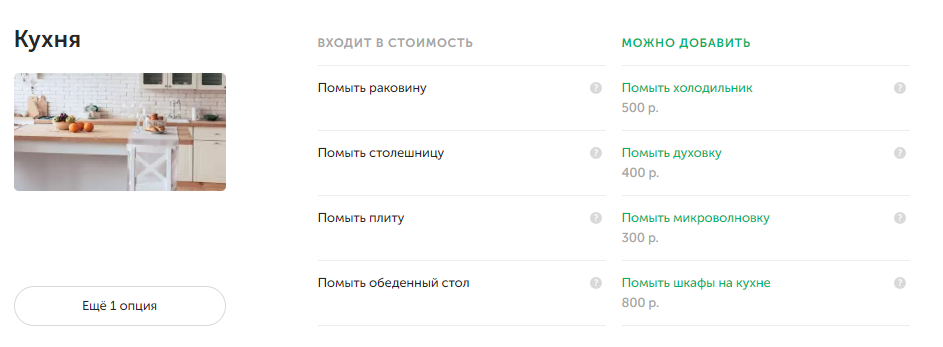


Рисунок 3 – Ценообразование в Qlean

В заключении хочется отметить, что данная система имеет больше преимуществ, чем недостатков, поэтому она может вправе считаться эталонной.

1.3.3 Клининговая компания «Avalon»

Avalon – это Самарская клининговая компания, которая предлагает уборку жилых и коммерческих помещений, химчистку матрасов, мебели и ковровых покрытий.

Сходу хочется отметить недостатки системы в виде перебивающегося интерфейса, неуместных анимаций и слабой адаптации веб-сайта под мобильные устройства. Функционал оформления заказа спрятан глубоко в логику системы, поэтому для оформления изначально требуется перейти по нескольким ссылкам, что может вполне отбить желание пользователей к заказу или требуется разглядеть неприметное меню в конце сайта. Крайне непонятное ценообразование, цены указаны только на основные виды работ. В форме для выбора степени загрязнения можно выбрать вариант «Срочно приехать», что является логической ошибкой. Рисунок 4 показывает это.



Рисунок 4 – Форма для выбора степени загрязнения

Из преимуществ можно отметить наличие номеров телефона для заказа клининга по сотовой связи, наличие физического офиса, куда можно обратиться в случае неопределенной ситуации. К преимуществам можно отнести довольно приемлемые цены, а также быстрый выезд клинера на объект: описана гарантия выезда на объект в течение двух часов.

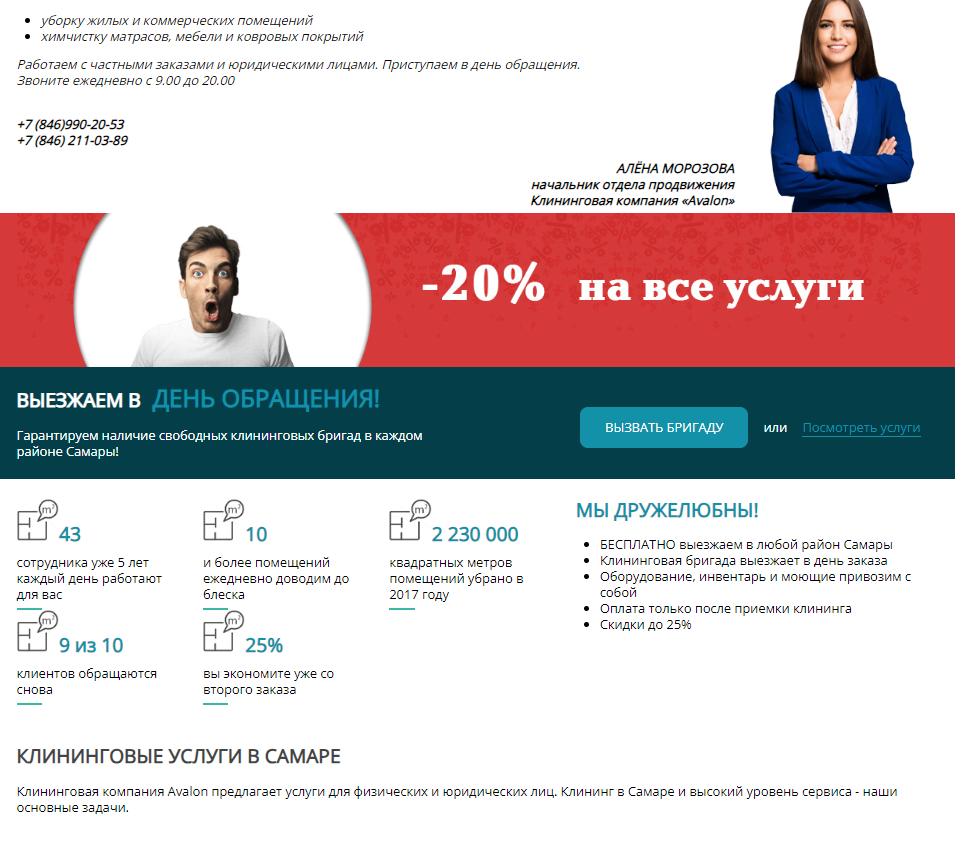


Рисунок 4 – Интерфейс «Avalon»

1. Проектирование и программная реализация метода решения задачи
   1. Обоснование и выбор комплекса программных средств, языка программирования, среды разработки, системы управления базой данных, операционной системы
      1. Выбор языка программирования

В настоящий момент существует большой выбор языков программирования. Они подразделяются на компилируемые и интерпретируемые, высокоуровневые и низкоуровневые, строго и слабо типизируемые. Каждый из вариантов содержит свои плюсы и минусы.

Мой выбор пал на компилируемый, высокоуровневый, строго типизированный язык программирования Java. Java – крайне обильно используемый язык программирования, который заточен под написание интернет-приложений. Однако ключевым преимуществом Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. Затем байт-код выполняется виртуальной машиной JVM (Java Virual Machine). В этом плане Java отличается от стандартных интерпретируемых языков как PHP или Perl, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как С или С++ [7].

Стоит отметить, что в моем случае использовался фреймворк Spring Boot, который упрощает написание приложений, избавляя программиста от написания однотипного кода. Spring Boot – фреймворк для создания приложений на языке Java. Он предоставляет возможность быстрого и удобного создания веб-приложений, используя уже готовый компоненты и шаблоны проектирования. Spring Boot включает все необходимые библиотеки, что позволяет разработчику сразу же начинать работу над проектом, не тратя время на конфигурацию приложения. Он также позволяет создавать микросервисы и интегрировать различные технологии, такие как базы данных, технологии обработки сообщений, без большого количества дополнительного кода [8].

* + 1. Выбор программных средств
       1. Средства для реализации клиентской части

Для реализации клиентской части системы выбор был сделан в сторону фреймворка Angular.

Angular представляет собой фреймворк от компании Google для разработки клиентских приложений. Прежде всего он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений. Одной из ключевых особенностей Angular – использование в качестве языка программирования TypeScript. Однако стоит отметить, что TypeScript носит рекомендательный характер, поскольку Angular-приложение может использовать также Dart или JavaScript.

Angular обладает следующими преимуществами:

- модульность и компонентный подход: Angular-приложение разбивается на отдельные компоненты, которые подлежат многократному использованию. Такой подход значительно упрощает разработку и особенно поддержку;

- поддержка языка программирования TypeScript. Данный язык является строго типизированным, эта особенность значительно облегчает процесс разработки и повышает качество кода;

- поддержка пакетного менеджера npm: NPM – это менеджер пакетов, с помощью которого можно легко внедрять зависимости (библиотеки) в проект.

* + - 1. Средства для отладки серверной части

Перед использованием серверной части в совокупности с клиентской, хорошим тоном является проверка работоспособности всех аспектов разработанного API, инструмент Postman является основным для этой задачи.

Postman – это мощный и интуитивно понятный инструмент, предназначенный специально для тестирования и разработки API [9].

Postman позволяет отправлять HTTP-запросы, выбирая любой тип запроса, тело, заголовки, и просматривать тело и статус ответа. Также это средство позволяет открывать WebSocket-соединения, отправлять и получать сообщения в ws-канал.

* + 1. Выбор среды разработки

В настоящее время большую популярность набирают среды разработки от компании JetBrains, поскольку компания разрабатывает ряд высококачественных продуктов, каждый из которых отлично подходит под определенные языки программирования. В нашем случае для языка программирования Java описанной выше компанией разработана среда разработки IntelliJ IDEA. InelliJ IDEA – это интегрированная среда разработки (IDE) для языков программирования Java, Kotlin, Groovy и других языков. Она имеет множество функций и инструментов, которые облегчают процесс разработки.

Основные функции IntelliJ IDEA:

* автодополнение кода: среда разработки автоматически предлагает подсказки и завершение кода, что повышает производительность и уменьшает шанс ошибиться;
* интеграция с системами контроля версий: среда разработки поддерживает интеграцию с Git: позволяет делать операции commit, push, pull. Также есть возможность сделать patch-файл;
* сохраняет историю изменения кода: в описываемой среде разработки присутствует функционал, который фиксирует измененные файлы при каждом запуске программы. Можно вернуться к каждой зафиксированной точке;
* поддержка сопутствующих программных средств: IDEA предлагает решения для более удобной работы с такими сервисами, как: Maven, Spring, Hibernate, PostgreSQL и других.

На данный момент крайне популярен редактор кода Visual Studio Code (VS Code). Это средство крайне легковесное, поскольку изначально поддерживает лишь несколько языков и служит редактором кода. Однако VS Code имеет раздел расширений, в котором можно установить абсолютно любой плагин для работы с любым языком программирования. VS Code подходит для более примитивных языков, таких как JavaScript, Python, HTML, CSS.

Eclipse – свободная интегрированная среда разработки модульных кроссплатформенных приложений. На данный момент система поддерживается Eclipse Foundation. Eclipse написана на Java, потому является платформо-независимым продуктом [10]. Однако описываемая среда разработки имеет ряд критических недостатков:

- использует много системных ресурсов и процессов;

- использует много памяти;

- имеет медленный запуск;

- функция отладки не идет в ногу со временем по сравнению с другими IDE;

- неполная интеграция с Maven и Gradle.

Из всех вышеперечисленных вариантов наиболее прогрессивной и подходящей средой разработки является IntellJ IDEA. Вдобавок к этому она идеально подходит именно для Java-проектов из-за своих особенностей.

* + 1. Выбор системы управления базой данных

В данной автоматизированной информационной системе была выбрана СУБД PostgreSQL.

СУБД (система управления базами данных) – это комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными. СУБД обеспечивает доступ к данным, позволяет выполнять запросы и обеспечивает целостность и безопасность данных. Она также может обеспечивать масштабируемость и производительность базы данных. СУБД является неотъемлемой частью большинства приложений, которые требуют хранения и управления данными [11].

PostgreSQL – это открытая система управления базами данных, которая обладает высокой степенью надежности и устойчивости к сбоям благодаря использованию транзакций и механизмов восстановления данных. Она обладает высоким уровнем безопасности благодаря использованию механизмов аутентификации, авторизации и шифрования данных. PostgreSQL поддерживает множество стандартов SQL, что делает её совместимой с другими СУБД. Кроме того, PostgreSQL позволяет добавлять новые функции и возможности с помощью расширений, что делает её еще более гибкой и мощной [12].

* + 1. Выбор операционной системы

Поскольку язык программирования, среда разработки и СУБД были выбраны кроссплатформенные, нет рамок в выборе операционной системы и рациональнее использовать ту, которая привычнее.

Исходя из этого, для разрабатываемой информационной системы была выбрана операционная система Windows 10. Windows 10 – это операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства Windows NT.

* 1. Описание информационно-логической модели системы и логической модели базы данных

Проектирование – один из важнейших шагов при разработке сложных систем, который крайне часто игнорируется начинающими разработчиками. Зачастую неопытные программисты пытаются удержать всё у себя в голове. Как результат, с таким подходом не будет ни четкого плана действий, ни удобного варианта, в который можно было бы вносить правки без последствий.

Обычно при проектировании система изображается графически, поскольку человеку легче разобраться в таком представлении. Вследствие этого вместо громоздких текстов строятся различные диаграммы для описания систем [13].

* + 1. Диаграмма вариантов использования

Диаграмма вариантов использования (use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей [13]. Разработку сложной автоматизированной системы принято начинать именно с проектирования диаграммы вариантов использования. Каждый блок из этой диаграммы определяет набор действий, которые могут или должны выполняться проектируемой системой при взаимодействии с соответствующим актором.

На рисунке 5 представлена диаграмма вариантов использования для проектируемой автоматизированной информационной системы.

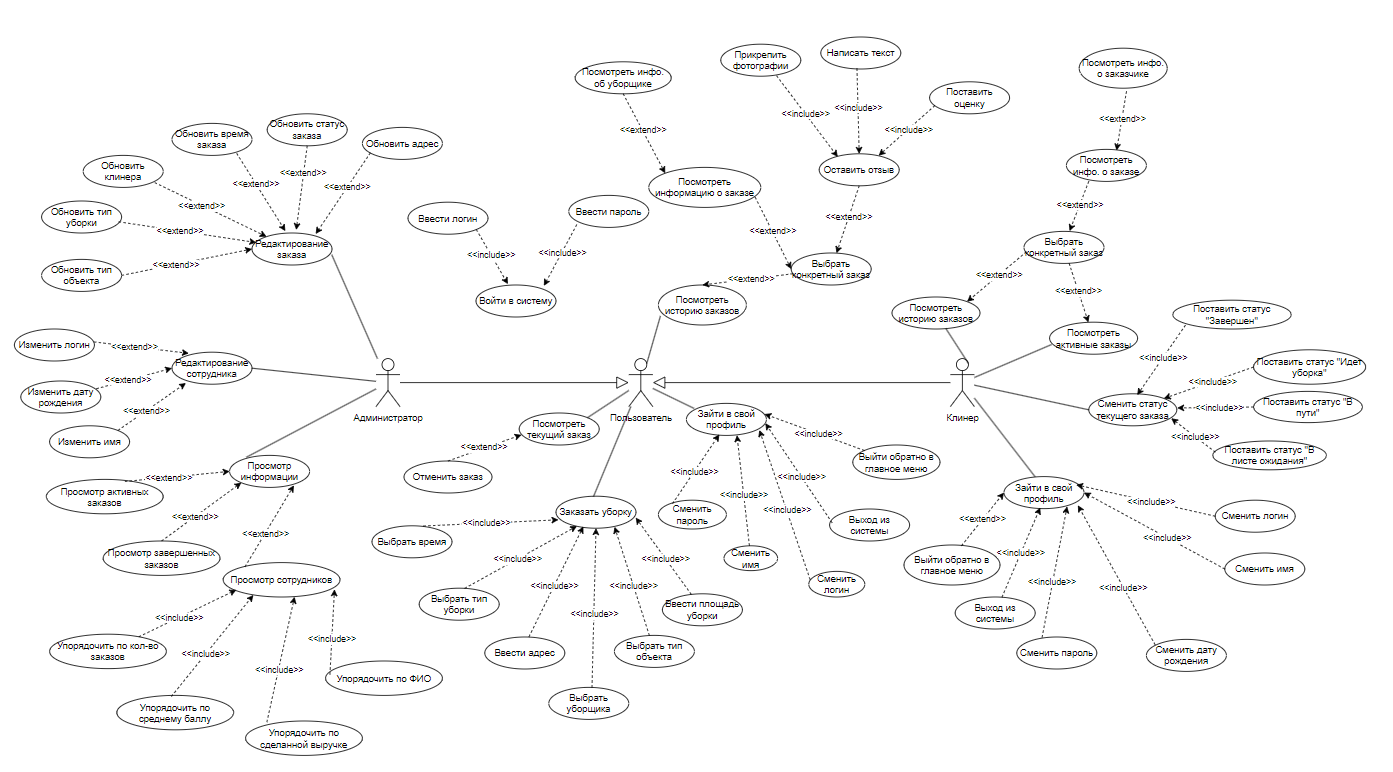


Рисунок 5 – Диаграмма вариантов использования

* + 1. Сценарии использования

Сценарии использования, варианты использования или прецеденты – спецификация последовательностей действий в UML, которые может осуществлять система, подсистема или класс, взаимодействуя с внешними действующими лицами (англ. Actors) [14].

Рассмотрим сценарии использования для нескольких вариантов использования.

Вариант использования: Войти в систему.

Краткое описание: Предоставляет возможность авторизоваться в системе.

Предусловие: Пользователь открыл нужный веб-сайт, на экране форма авторизации.

Актант: Клинер

Основной поток событий:

1. Клинер вводит логин и пароль.
2. Актант нажимает левой кнопкой мыши на кнопку «Войти».

А1: Система выводит сообщение о неправильном формате введённых данных.

А2: Система выводит сообщение о неправильном логине или пароле.

Альтернативы:

А1: Система выводит сообщение об отсутствии введённого логина.

1. Система закрывает форму авторизации.

Постусловие: При успешной авторизации в системе на экране появляется панель управления для сотрудника.

Вариант использования: Оформить заказ на клининг.

Краткое описание: Предоставляет клиенту возможность оформить заказ на уборку помещения.

Предусловие: Клиент открыт нужный веб-сайт, авторизация пройдена от роли клиента и открыта панель управления для пользователей.

Актант: Пользователь.

Основной поток событий:

1. Пользователь нажимает на поле с выбором типа объекта.
2. Система реагирует на нажатие открытием выпадающего списка.

А1: Пользователь нажал на область за пределами списка.

А2: Пользователь выбрал наиболее подходящий вариант.

1. Система закрыла выпадающий список выбора типа объекта.
2. Пользователь нажал на поле для ввода площади уборки.
3. Пользователь вводит площадь уборки.

А3: Система успешно проверила значение площади.

1. Пользователь выбирает дату и время.
2. Пользователь выбирает тип уборки из выпадающего списка

А4: Пользователь нажал на область за пределами списка.

А5: Пользователь выбрал подходящий вариант.

1. Пользователь ввёл адрес уборки.

А6: Система успешно проверила адрес по базе районов и улиц.

1. Пользователь выбирает клинера из выпадающего списка.

А7: Пользователь нажал на область за пределами списка.

А8: Пользователь выбрал подходящий вариант.

1. Пользователь нажал кнопку «Оформить заказ».

А9: Система проверила заказ и занесла в БД.

Альтернативы:

А1: Пользователь нажатием перешёл в другое окно.

А1.1: Система отобразила соответствующее окно.

А6: Система не нашла совпадений в базе районов или улиц.

А6.1: Система выдала сообщение об ошибке.

А9: Система отклонила заказ в силу неправильно введённых данных.

Постусловие: При успешном оформлении заказа в системе пользователь увидит его в списке активных заказов, сам заказ создастся и занесётся в базу данных. Выбранный клинер увидит добавленный заказ.

* + 1. Диаграмма классов системы

Диаграмма классов описывает типы объектов системы и различного рода статические отношения, которые существуют между ними. На диаграммах классов отображаются свойства классов, операции классов и ограничения, которые накладываются на связи между объектами [14].

На рисунке 6 представлена диаграмма классов-сущностей системы.

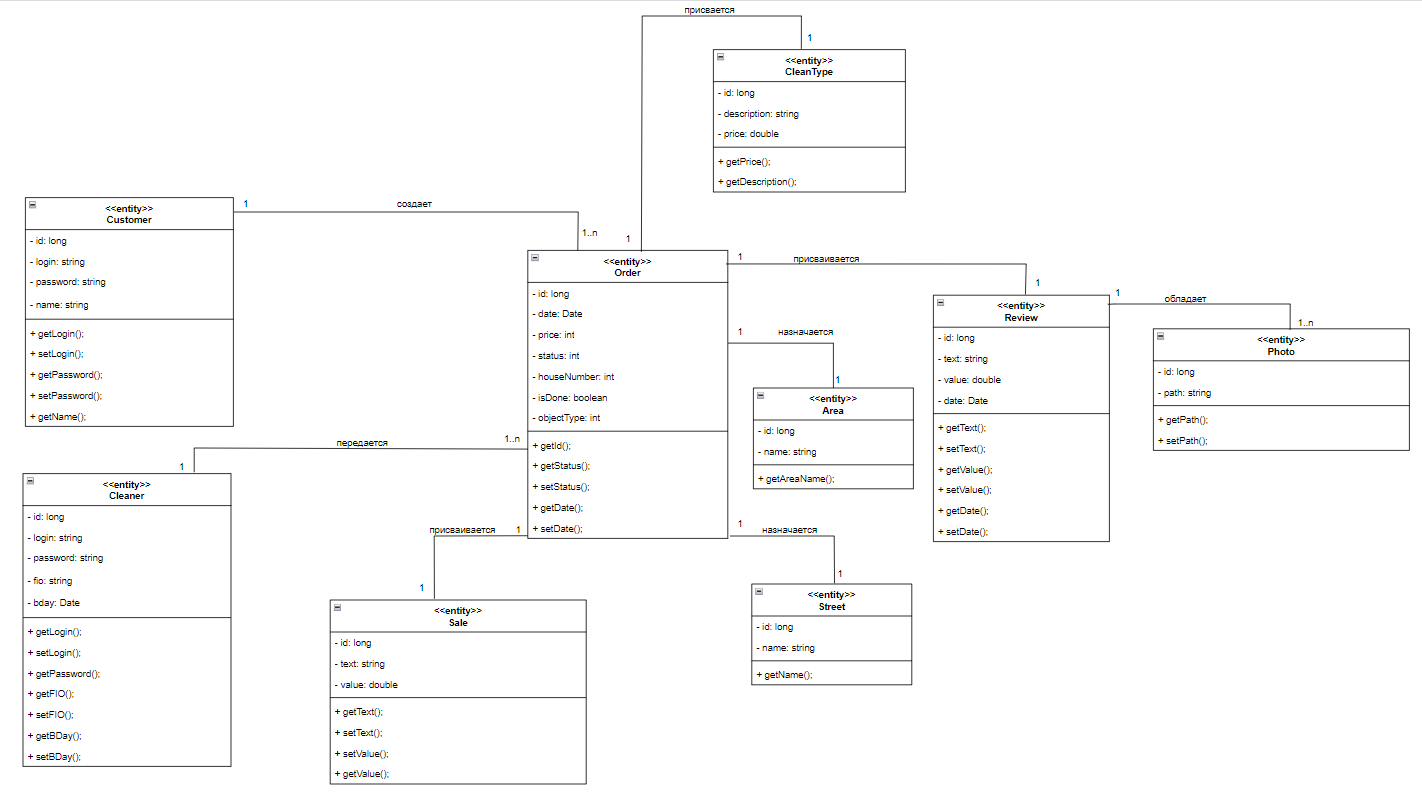


Рисунок 6 – Диаграмма классов-сущностей системы

Класс «Order» отвечает за взаимодействие с таблицей «Order» из БД.

Класс «Customer» отвечает за взаимодействие с таблицей «Customer» из БД.

Класс «Cleaner» отвечает за взаимодействие с таблицей «Cleaner» из БД.

Класс «Sale» отвечает за взаимодействие с таблицей «Sale» из БД.

Класс «Street» отвечает за взаимодействие с таблицей «Street» из БД.

Класс «Area» отвечает за взаимодействие с таблицей «Area» из БД.

Класс «Review» отвечает за взаимодействие с таблицей «Review» из БД.

Класс «Photo» отвечает за взаимодействие с таблицей «Photo» из БД.

Класс «CleanType» отвечает за взаимодействие с таблицей «CleanType» из БД.

На рисунке 7 показана диаграмма классов системы, реализующие всю логику приложения.

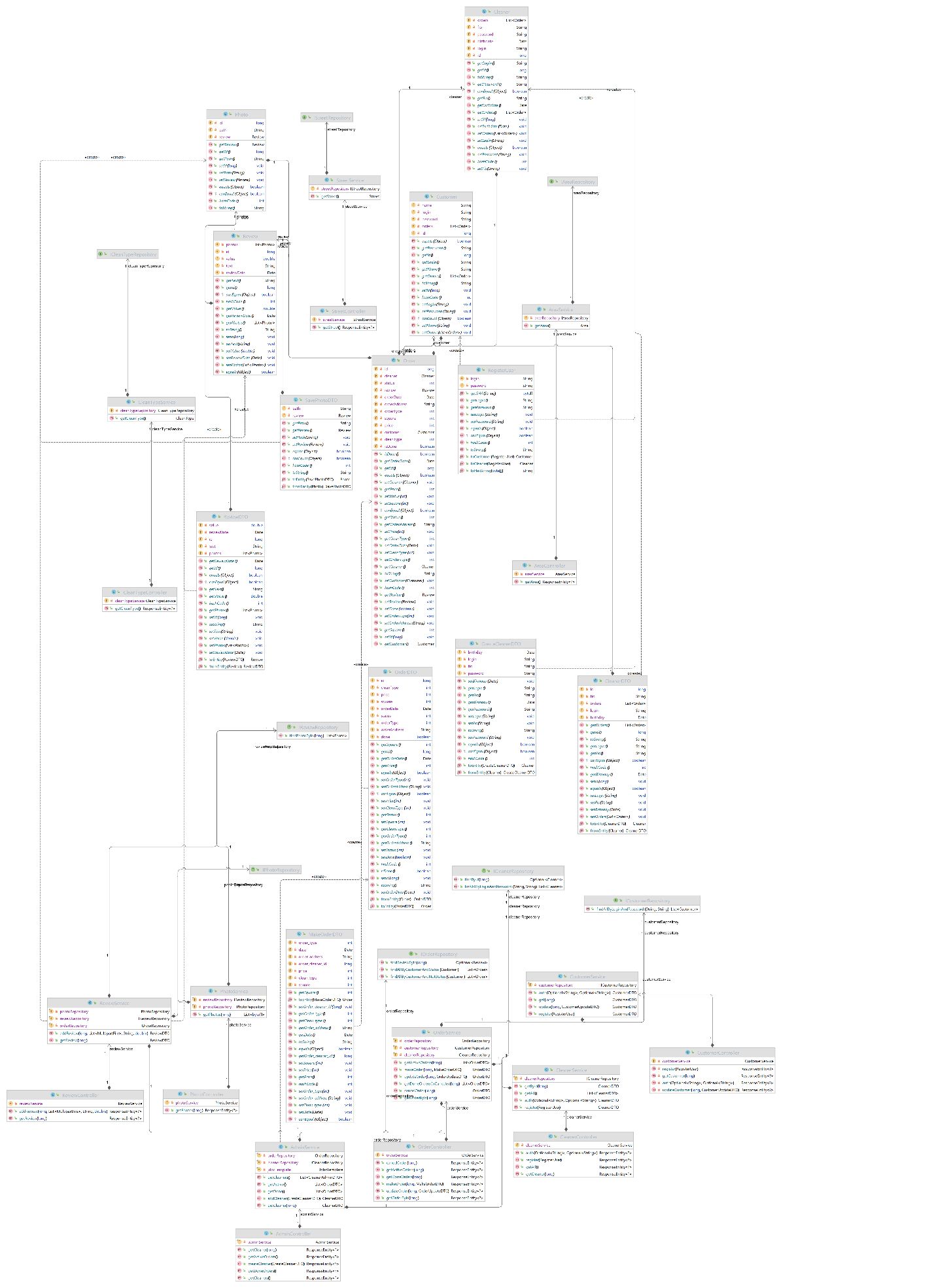


Рисунок 7 – Диаграмма классов системы

Классы «ICustomerRepository», «CustomerService» и «CustomerController» реализуют взаимодействие с сущностью «Клиент» в БД и бизнес-логику, которая затрагивает эту сущность.

Классы «ICleanerRepository», «CleanerService» и «CleanerController» реализуют взаимодействие с сущностью «Клинер» в БД и бизнес-логику, которая затрагивает эту сущность.

Классы «IOrderRepository», «OrderRepository» и «OrderController» реализуют взаимодействие с сущностью «Заказ» в БД и бизнес-логику, которая затрагивает эту сущность.

Классы «IPhotoRepository», «PhotoService» и «PhotoController» реализуют взаимодействие с сущностью «Фотография» в БД и бизнес-логику, которая затрагивает эту сущность.

Классы «ISaleRepository», «SaleService» и «SaleController» реализуют взаимодействие с сущностью «Скидка» в БД и бизнес-логику, которая затрагивает эту сущность.

Классы «ICleanTypeRepository», «CleanTypeService» и «CleanTypeController» реализуют взаимодействие с сущностью «Тип уборки» в БД и бизнес-логику, которая затрагивает эту сущность.

Классы «IAreaRepository», «AreaService» и «AreaController» реализуют взаимодействие с сущностью «Район» в БД и бизнес-логику, которая затрагивает эту сущность.

Классы «IStreetRepository», «StreetService» и «StreetController» реализуют взаимодействие с сущностью «Район» в БД и бизнес-логику, которая затрагивает эту сущность.

Классы «IAdminRepository» «AdminService» и «AdminController» содержат дополнительные методы для работы со всеми сущностями. Данная логика позволяет писать универсальные запросы к базе данных, не ломая архитектуру приложения.

Классы «MakeOrderDTO», «OrderDTO», «CleanerDTO», «ReviewDTO», «SavePhotoDTO», «RegisterUserDTO» представляют собой POJO-сущности, которые сервер отправляет клиенту в виде ответа в случае необходимости.

POJO-объект – это простой Java-объект, не унаследованный от какого-то специфического объекта и не реализующий никаких служебных интерфейсов помимо тех, которые нужны для бизнес-модели [15].

* + 1. Диаграмма состояний

Диаграмма состояний отображает поведение объектов одного класса в динамике, связь состояний объектов с событиями и определяет:

- какие типичные состояние проходит объект;

- какие события ведут к изменению состояния объекта;

- какие действия объект выполняет, когда он получает сообщение об изменении состояния;

- как объекты создаются и уничтожаются (входные и выходные точки диаграммы).

Данная диаграмма полезна при моделировании жизненного цикла объекта. От остальных диаграмм описываемая диаграмма отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса – одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события [16].

Ниже, на рисунке 8, представлена диаграмма состояний для проектируемой системы.

* + 1. Диаграмма деятельности

Диаграмма деятельности – это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов систем. Диаграмма деятельности представляет собой блок-схему, которая показывает, как поток управления переходит от одной деятельности к другой. В отличие от традиционной блок-схемы диаграмма деятельности показывает параллелизм так же хорошо, как и ветвление потока управления.

Моделирование динамических аспектов систем при помощи диаграмм деятельности большей частью подразумевает моделирование последовательных шагов вычислительного процесса [17].

На рисунке 9 представлена диаграмма деятельности для разрабатываемой системы.

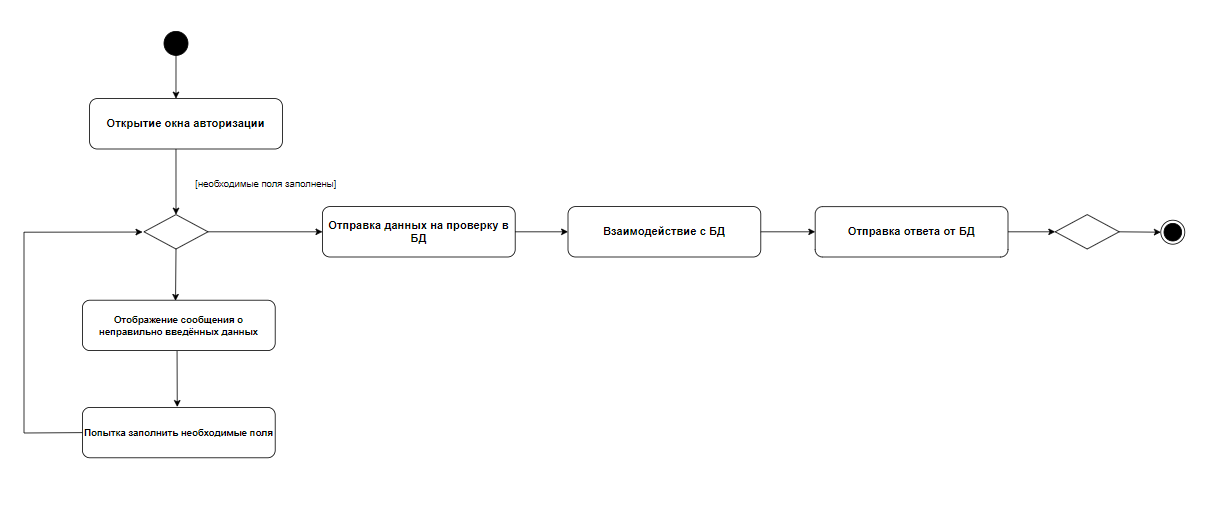


Рисунок 8 – Диаграмма состояний для варианта использования «Авторизация пользователя»

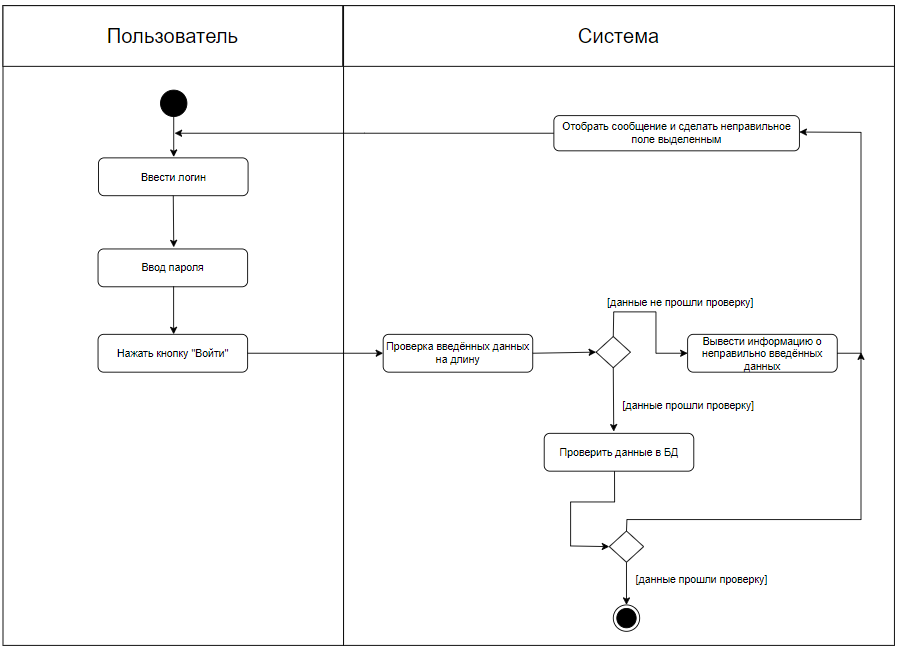


Рисунок 9 – Диаграмма деятельности для варианта использования «Авторизация пользователя»

* + 1. Диаграмма последовательности

Диаграммы последовательностей, используемые разработчиками, моделируют взаимодействия между объектами в едином сценарии использования. Они иллюстрируют, как различные части системы взаимодействуют друг с другом для выполнения функций, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования [18].

Основными элементами диаграммы последовательности являются обозначения объектов, вертикальные «линии жизни», отображающие течение времени, прямоугольники, отображающие деятельность объекта или исполнение им определенной функции, и стрелки, показывающие обмен сигналами или сообщениями между объектами [19].

На рисунке 10 изображена диаграмма последовательности для разрабатываемой системы.

* + 1. Диаграмма кооперации

Диаграмма кооперации предназначена для спецификации структурных аспектов взаимодействия. Главная особенность – графически представить, как последовательность взаимодействия, так и все структурные отношения между объектами, участвующими в этом взаимодействии.

На диаграмме кооперации изображаются только отношения между объектами. Последовательность взаимодействия и параллельных потоков определяется с помощью порядковых номеров.

С помощью диаграммы кооперации описывают совокупность объектов, взаимодействующих между собой для выполнения определенной задачи [20].

На рисунке 11 представлена диаграмма кооперация для варианта использования «Авторизация пользователя» разрабатываемой системы.

* + 1. Диаграмма компонентов

Диаграмма компонентов описывает особенности физического представления системы. Определяет архитектуру разрабатываемой системы. Устанавливает зависимости между программными компонентами: исходных, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу [21].

На рисунке 12 представлена диаграмма компонентов для разрабатываемой системы.

* + 1. Диаграмма развёртывания

Диаграмма развёртывания предназначена для представления общей конфигурации или топологии распределенной программной системы и содержит изображение размещения различных артефактов по отдельным узлам системы.

Данная диаграмма предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения.

На рисунке 13 представлена диаграмма развёртывания для разрабатываемой информационной системы.

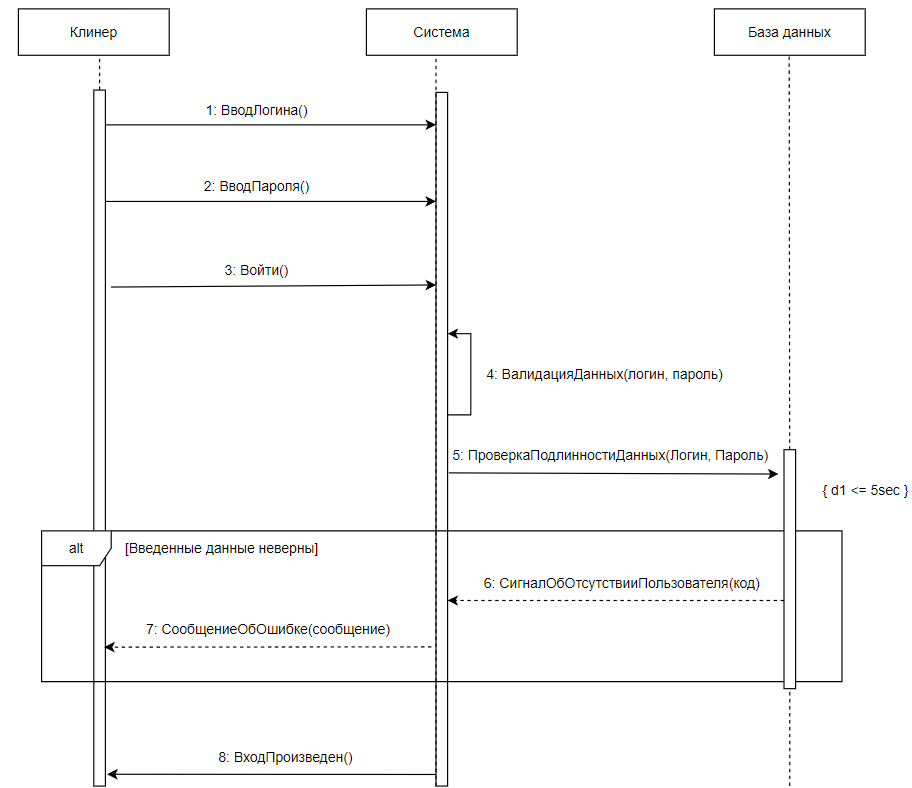


Рисунок 10 – Диаграмма последовательности для варианта использования «Авторизация пользователя»

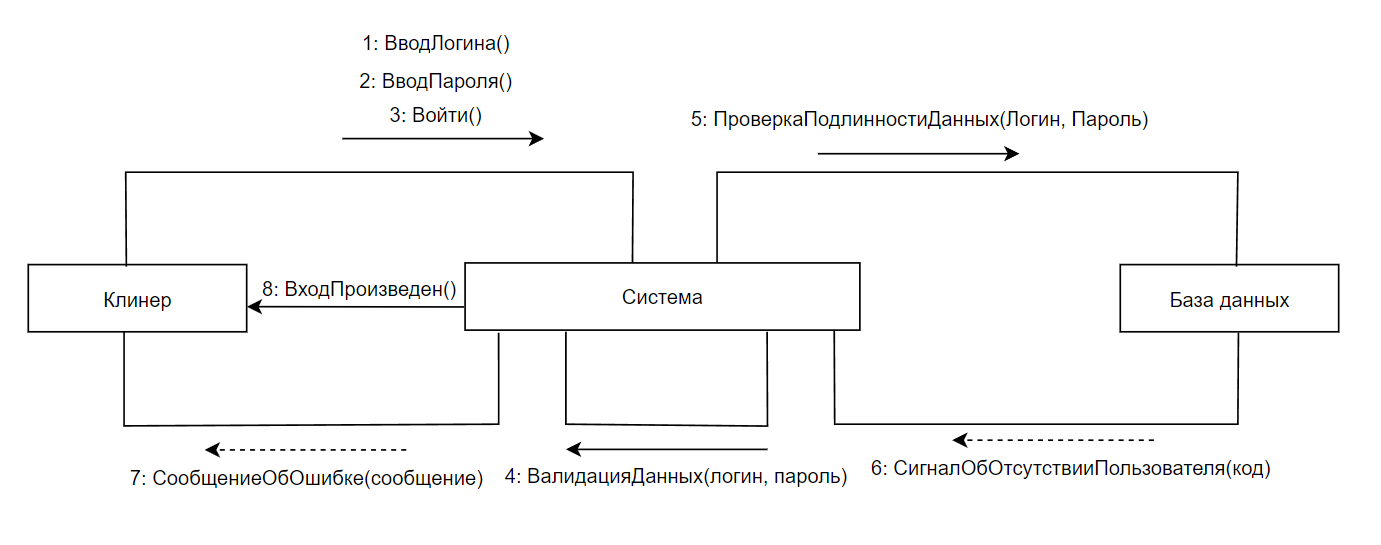


Рисунок 11 – Диаграмма кооперации для варианта использования «Авторизация пользователя»

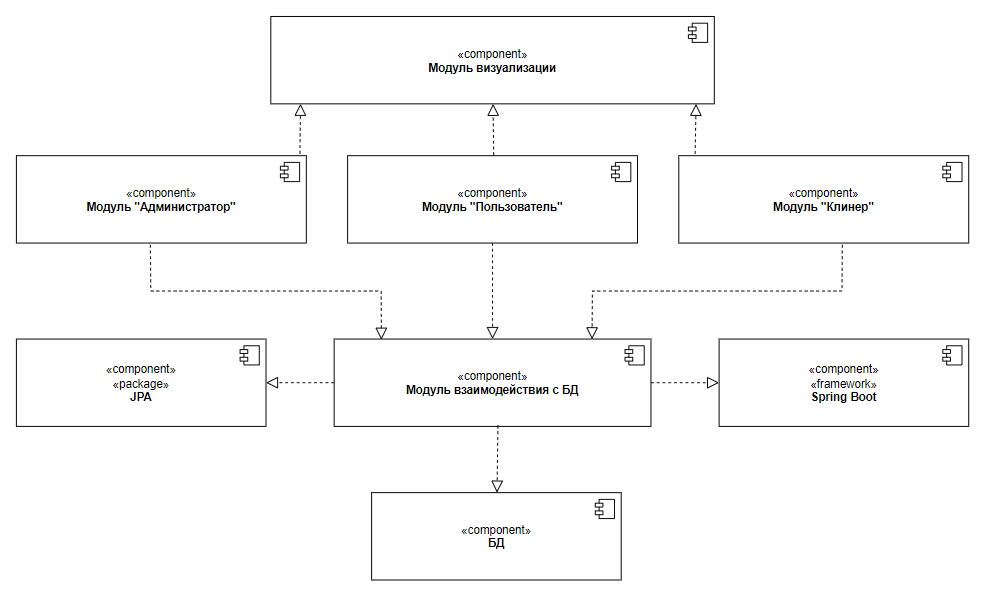


Рисунок 12 – Диаграмма компонентов для разрабатываемой системы

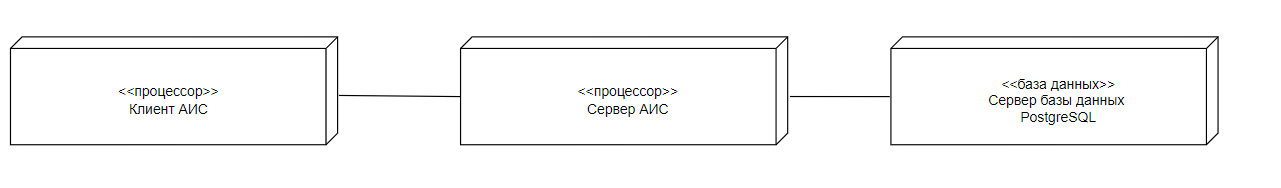


Рисунок 13 – Диаграмма развёртывания для разрабатываемой системы

* 1. Описание логической модели баз данных
  2. Описание физической модели базы данных
  3. Разработка алгоритмов обработки данных

1. Вычислительные эксперименты проверки эффективности программной реализации метода решения задачи
   1. Разработка контрольного примера обработки реальных данных или разработка плана исследования эффективности разработанной программной системы на модельных или реальных данных
   2. Проверка эффективности разработанной программной системы для решения поставленной задачи

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Вики. Предметная область (Object domain) [Электронный ресурс]. URL – <https://wiki.loginom.ru/articles/object-domain.html>
2. Википедия. Уборка [Электронный ресурс]. URL - <https://ru.wikipedia.org/wiki/%D0%A3%D0%B1%D0%BE%D1%80%D0%BA%D0%B0>
3. Википедия. Обратная связь (коммуникации) [Электронный ресурс]. URL - <https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D1%81%D0%B2%D1%8F%D0%B7%D1%8C_(%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8)>
4. Обратная связь [Электронный ресурс]. URL - <https://roistat.com/rublog/obratnaja-svjaz/>
5. ООО «Первая клининговая компания» [Электронный ресурс]. URL - <https://klining-msk.ru/>
6. Что такое Java? [Электронный ресурс]. URL - <https://aws.amazon.com/ru/what-is/java/>
7. Введение в Java [Электронный ресурс]. URL - <https://metanit.com/java/tutorial/1.1.php>
8. Введение в Spring Boot: создание простого REST API на Java [Электронный ресурс]. URL - <https://habr.com/ru/articles/435144>
9. Postman: Основы тестирования API и первые шаги с инструментом [Электронный ресурс]. URL - <https://habr.com/ru/companies/vk/articles/750096/>
10. Eclipse (среда разработки) [Электронный ресурс]. URL - <https://ru.wikipedia.org/wiki/Eclipse_(%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8)>
11. ItGlobal. СУБД (Система управления базами данных) [Электронный ресурс]. URL – <https://itglobal.com/ru-ru/company/glossary/subd-sistema-upravleniyabazami-dannyh/>
12. PostgreSQL — объектно-реляционная система управления базами данных [Электронный ресурс]. URL - <https://web-creator.ru/articles/postgresql>
13. Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс]. URL - <https://habr.com/ru/articles/566218/>
14. Systems Engineering Thinking Wiki. Сценарий использования [Электронный ресурс]. – <http://sewiki.ru/Сценарий_использования>
15. Создание POJO объекта [Электронный ресурс]. URL - <https://devcolibri.com/unit/%D1%83%D1%80%D0%BE%D0%BA-8-%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-pojo-%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%B0-user-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-view-%D0%B8%D0%B7-java-2/>
16. Flex berry. Диаграмма состояний (Statechart diagram) [Электронный ресурс]. – <https://flexberry.github.io/ru/fd_statechart-diagram.html>
17. Унифицированный язык моделирования UML. Понятие диаграммы деятельности [Электронный ресурс]. URL - <https://it.kgsu.ru/UML/uml_0156.html>
18. Учебное пособие по диаграммам последовательности: полное руководство с примерами [Электронный ресурс]. URL - <https://creately.com/blog/ru/%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0/%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5-%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5-%D0%BF%D0%BE-%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9/>
19. Диаграмма последовательности [Электронный ресурс]. URL - <https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8>
20. Диаграмма кооперации [Электронный ресурс]. URL - <https://portal.tpu.ru/SHARED/r/RYBALKA/academic/mct/TabMCTLect/MCTLect4.pdf>
21. Диаграмма компонентов [Электронный ресурс]. URL - <https://portal.tpu.ru/SHARED/r/RYBALKA/academic/mct/TabMCTLect/MCTLect6.pdf>