Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт открытого и дистанционного образования
Кафедра Техники, технологий и строительства

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
к.т.н., доцент
_____К.М. Виноградов
_____2020 г.

Интеграция CRM-модуля в информационную систему предприятия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 09.03.01.2020.014 ПЗ ВКР

Руководитель работы,
к.э.н., доцент
_____А.Г. Калачева
_____2020 г.

Автор работы -
студент группы ДО-525
_____Ю.И. Стручков
_____2020 г.

Нормоконтролер,
преподаватель
_____ О.С. Микерина

_____2020 г.

Челябинск 2020

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Институт открытого и дистанционного образования
Кафедра Техника, технологии и строительство
Направление 09.03.01 «Информатика и вычислительная техника»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ К.М. Виноградов
_____2020 г.

ЗАДАНИЕ

на выпускную квалификационную работу студента
Стручкова Юрия Игоревича
Группа ДО-525

1 Тема работы: Интеграция CRM-модуля в информационную систему предприятия

утверждена приказом по университету от «24» апреля 2020 г. № 627

2 Срок сдачи студентом законченной работы 11.06.2020

3 Исходные данные к работе

3.1 Существующая информационная система предприятия

3.2 Программное и аппаратное обеспечение, используемое для интеграции CRM-модуля в информационную систему предприятия

4 Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

4.1 Анализ требований к CRM-системе

4.1.1 Анализ предметной области разработки

4.1.2 Требования к интерфейсу

4.1.3 Требования к функционалу

4.2 Сравнение отечественных и передовых зарубежных технологий и решений

4.2.1 Обзор современных CRM-систем

4.2.2 Выбор инструментальных средств разработки

4.3 Проектирование CRM-модуля

4.3.1 Проектирование структур данных и алгоритмов

4.3.2 Проектирование пользовательского интерфейса

4.3.3 Разработка базы данных

4.3.4 Разработка программной части системы

4.3.5 Тестирование системы

5 Перечень графического материала

5.1 Пояснительная записка на 50 листах формата А4.

5.2 Презентация на 12 слайдах

Дата выдачи задания 25.03.2020

Руководитель _____А.Г. Калачева
(подпись)

Задание принял к исполнению _____ Ю.И. Стручков
(подпись)

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов выпускной квалификационной работе	Срок выполнения этапов работы	Отметка руководителя о выполнении
Введение		
Анализ требований к CRM-системе		
Анализ предметной области разработки		
Требования к интерфейсу		
Требования к функционалу		
Сравнение отечественных и передовых зарубежных технологий и решений		
Обзор современных CRM-систем		
Выбор инструментальных средств разработки		
Проектирование CRM-модуля		
Проектирование структур данных и алгоритмов		
Проектирование пользовательского интерфейса		
Разработка базы данных		
Разработка программной части системы		
Тестирование системы		
Заключение		

Зав. кафедрой _____ К.М. Виноградов
(подпись)

Руководитель работы _____ А.Г. Калачева
(подпись)

Студент_____ Ю.И. Стручков
(подпись)

АННОТАЦИЯ

Стручков, Ю.И. Интеграция CRM-модуля в информационную систему предприятия. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ИОДО; 2020, 50 с. 16 илл., библиогр. список – 40 наим., 2 прил.

В настоящее время автоматизация деятельности сервисных центров оказывает большое влияние на конкурентоспособность предоставляемых ими услуг и продукции. Целью работы является разработка автоматизированной CRM-системы для учета заказов сервисного центра «iReLab», обеспечивающая оптимизацию учета и управления компанией и охватывающая все ключевые моменты ее функционирования.

В ходе выпускной квалификационной работе будет проведен анализ существующих аналогичных систем. А также проектирование, разработка и тестирование собственной системы.

В результате будет разработан программный продукт, который позволит автоматизировать деятельность сервисного центра.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В жизнь современного человека устойчиво вошли новые технологии, которые кардинальным образом улучшают и процессы производства, и процессы потребления информации. Большинство рутинных операций человека подвергаются автоматизации. В настоящее время автоматизация деятельности предприятий оказывает большое влияние на конкурентоспособность предоставляемых ими услуг и продукции. Основной целью автоматизации является разработка и внедрение единой информационной среды, обеспечивающей оптимизацию учета и управления компанией и охватывающей все ключевые моменты ее функционирования в выбранной рыночной нише.

В нашем случае данной нишей является оказание услуг ремонта и обслуживания цифровой техники в сервисном центре.

Актуальность данной работы заключается в том, что создание информационной системы для автоматизации деятельности сервисного центра, можно рассматривать как неотъемлемый элемент развития данного предприятия, что способствует упрощению работы обслуживающего персонала и более ускоренный процесс обслуживания цифровой техники.

Целью выпускной квалификационной работы является интеграция CRM-модуля в информационную систему предприятия для учета заказов сервисного центра «iReLab».

Задачи выпускной квалификационной работы:

- произвести анализ предметной области;
- выяснить какие основные потоки данных существуют на предприятии;
- установить какие процессы предприятия можно автоматизировать;
- автоматизировать максимально возможное количество бизнес процессов;
- улучшить качество регулирования процессов;
- обеспечить достоверность информации о материальных компонентах, применяемых в производстве;
- хранение информации о ходе технологического процесса и аварийных ситуациях;
- улучшение эргономики труда операторов процесса;
- выбрать способы и средства для разработки программного обеспечения;
- выдвинуть требования к программному продукту;
- разработать ПО согласно выдвинутым требованиям;
- проверить качество программного продукта.

Объектом выпускной квалификационной работы: автоматизация деятельности сервисного центра.

Предметом выпускной квалификационной работы: интеграция CRM-модуля для автоматизации деятельности сервисного центра.

Практическая значимость выпускной квалификационной работы состоит в усовершенствование информационной системы предприятия для учета заказов.

Структура выпускной квалификационной работы состоит из введения, трех разделов, заключения и библиографического списка. Раздел 1 посвящен анализу требований к CRM-системе, предметной области разработки, а также требования к интерфейсу и функционалу. Раздел 2 содержит сравнение отечественных и передовых зарубежных технологий и решений Раздел 3 содержит выбор инструментальных средств разработки, проектирование пользовательского интерфейса, структур данных и алгоритмов, разработку структуры базы данных, слоя взаимодействия с базой данных, слоя бизнес сервисов, пользовательского интерфейса и обоснование методики тестирования, а также результаты тестирования.

Объем выпускной квалификационной работы составляет 50 страниц машинописного текста и содержит 16 иллюстраций, 4 таблицы, библиографический список из 40 наименований и 2 приложение.

1 АНАЛИЗ ТРЕБОВАНИЙ К CRM-СИСТЕМЕ

1.1 Анализ предметной области разработки

CRM-система – это комплекс инструментов для работы с информацией, в том числе с клиентской базой, а также ряд методов, позволяющих систематизировать данные и регулировать порядок работы с ними. Такие системы так же важны, как и рабочий коллектив. Это окажет более значительное влияние на компанию в долгосрочной перспективе, чем любой из сотрудников [30].

Использование систем RM наиболее эффективно на жестких конкурентных рынках. В этой ситуации компаниям приходится бороться практически за всех клиентов. Таким образом, попытки планировать новые меры потерпят неудачу без надлежащего анализа клиентской базы.

CRM – корпоративная информационная система, служит для улучшения обслуживания клиентов за счет использования метода сохранения информации о клиентах и истории отношений с ними, установления и совершенствования текущих бизнес-процессов на основе хранимой информации и последующей оценки их эффективности. Термин CRM можно перевести на русский язык как управление взаимоотношениями с клиентами. Говоря о CRM, стоит выяснить ключевые моменты, которые он влечет за собой:

- Наличие единого хранилища информации kuyatsiorpd, из которого вы можете получить любую информацию о взаимодействии с клиентом в любое время.
- Синхронизация управления несколькими каналами взаимодействия (наличие организационных процедур, устанавливающих правила использования этой системы и информации в каждом отделе компании).
- Постоянный анализ собранной информации о работе с клиентом и принятие соответствующих организационных решений .

Такой подход к делу предполагает, что при каждом взаимодействии с клиентом сотрудники каждого канала организации Вся информация обо всех отношениях с клиентом и решение о дальнейшей работе принимается на их основе [3].

Идеология CRM заключается в том, чтобы обеспечить переход от стратегии массового маркетинга и массовых продаж к индивидуальной продаже или услуге (один на один) в соответствии с индивидуальными требованиями потребителя.

Внедрение автоматизации работы с клиентом с помощью CRM-системы обеспечивает автоматизацию маркетинговых бизнес-процессов, основой которых является личная работа с каждым из клиентов, обслуживание и поддержка клиентов. Такая система обеспечивает быстрый доступ к информации о клиенте и вывод такой информации в различных формах, удобных для разных пользователей.

Концепция CRM позволяет собирать и анализировать информацию о существующем или потенциальном клиенте. Затем система может предоставить информацию о:

- реакция клиента на бизнес-предложение;

- обслуживание-Удовлетворение;
- изменение настроек клиента;
- выполнение обязательств;
- доход клиента.

Система также позволяет отслеживать и анализировать существующие отношения с клиентом. Это важно, потому что стоимость привлечения нового клиента на порядок выше, чем стоимость удержания существующего клиента. программа

CRM предоставляет возможность эффективно обеспечить взаимодействие с клиентами, поставщиками, посредниками и структурными отделами компании.

Управление взаимоотношениями с клиентами [5]:
- прозрачный;
- планируемый;
- управляемый.

Программа позволяет автоматизировать процесс:
- создание отчетов;
- выдержки из документов;
- другие задачи;

Менеджеры могут отслеживать деятельность сотрудников в режиме реального времени. Система также включает в себя возможность запоминать конкретные события или даты [6]. CRM-система имеет следующие функции:
- планирование и анализ активных продаж;
- ведение базы данных клиентов;
- управление базой сотрудников;
- управление графиком работы:
  - спонсор;
  - контролер;
  - мерчендайзер;
- держите базу данных Bucking с программой адресной книги;
- осуществлять контроль взаимного урегулирования;
- создание документов;
- назначение задач;
- контроль выполнения задач;
- управление дебиторской задолженностью;
- анализ финансовой деятельности компании.

Требования к программному обеспечению: описание функций и функциональных возможностей развивающейся системы. Требования – это ожидания пользователей программного продукта. Требования могут быть очевидными или скрытыми с точки зрения клиента, известными или неизвестными,

ожидаемыми или неожиданными. Сбор, анализ и документирование требований клиентов известен как разработка требований.

Соблюдение требований к программному обеспечению является основой всего проекта разработки программного обеспечения. Поэтому они должны быть четко, правильно и четко определены [9].

В связи с этой информацией в настоящее время проводится детальный процесс изучения того, возможно ли разработать нужную систему и ее функциональные возможности. В ходе проведения работ проводились аналитические мероприятия по определению требований информационной системы.

Система должна частично автоматизировать деятельность сервисного центра. Автоматически генерируйте необходимые документы. Отслеживание движения заказов между структурными подразделениями, сотрудниками компании. Соберите статистическую информацию о сотрудниках сервисного центра, о количестве выполненных заказов, их стоимости, а также другую информацию, необходимую для регулирования работы сервисного центра.

Система улучшит как качество услуг, предоставляемых компанией, так и скорость ее предоставления, увеличив контроль над работниками и ресурсами компании. Сократить расходы на бухгалтерский учет за счет структурированного хранения всей необходимой информации о работе СЦ.

При изучении тематической области автоматизации учитываются основные особенности работы сервисного центра, формулируются основные системные требования для учета заказов.

Компания занимается ремонтом и продажей цифровых технологий. Сервисный центр планирует иметь несколько филиалов в Челябинске и Челябинске. В каждом городе будет несколько пунктов приема оборудования и мастерских, где будут проводиться ремонтные работы. Оборудование для ремонта на стойке регистрации доставляется в мастерскую для последующего ремонта. Вернувшись к клиенту, техника может быть выставлена на любой стойке регистрации или мастерской сервисного центра.

Каждый сотрудник сервисного центра несет свою ответственность в зависимости от должности.

Ответственный за работу с заказчиком несет ответственность за прием и выдачу техники, за согласование затрат и времени выполнения ремонтных работ. Вся ответственность за формализацию договоров на проведение работ с заказчиками лежит на нем.

Инженер-ремонтник выполняет все необходимые операции, непосредственно связанные с устранением неполадок в технике. К таким операциям относятся: диагностика неисправностей, поиск необходимых деталей для ремонта,

выполнение работ по устранению неполадок, проверка качества выполнения заказов.

Директор контролирует все ступени сервисного центра, заказывает расходные материалы.

Каждый сотрудник сервисного центра может объединить несколько позиций.

В сервисном центре было выделено несколько бизнес-процессов, основные из которых описаны ниже.

Процесс получения техники в ремонте выглядит следующим образом: клиент приходит на место входа и в разговоре с менеджером по обслуживанию клиентов делает заказ на выполнение определенных услуг. Заказ содержит информацию о заказчике, оборудовании, описание неисправности технического оборудования. После оформления заказа заказчик получает документ, указывающий на прием оборудования в ремонт.

Прием техники может проводиться как на приемных пунктах, так и на семинарах.

Процесс ремонта осуществляется инженером-механиком. Сначала проводится диагностика неисправности, поиск причины неисправности, как ее исправить и необходимых деталей и расходных материалов. Исходя из объема работ и стоимости деталей формируются цена выполнения заказа и сроки выполнения работ. Ремонтные работы проводятся только по согласованию с клиентом, если клиент отказывается выполнять работу, команда возвращается к нему.

Каждый заказ имеет статус, изменяется в зависимости от выполняемых на нем операций. Количество государств не ограничено, администратор сервисного центра имеет право самостоятельно решать, какие государства необходимы для осуществления бизнес-процессов [12]. Однако есть обязательные состояния: «принято», «исправлено», «согласовано», «готово», «оформлено».

1.2 Требования к интерфейсу

Спецификация требований к функциям документирует операции и действия, которые должна выполнять система. Спецификация функционального требования должна быть прочитана широкой аудиторией. Читатели должны понять систему, но для понимания документа не требуется никаких специальных технических знаний.

Основными функциональными требованиями системы являются:

- доступ к базе данных через интернет-браузер;
- автоматическое формирование акта принятия и акта завершенной работы, а также других соответствующих документов;
- обеспечение безопасности информации, хранящейся в системе;
- возможность регистрации, удаления, изменения заказов в системе;
- наличие информационных руководств, заполнение их, изменение, удаление;
- авторизация и аутентификация;

- различать доступ к информации о сотрудниках;
- возможность отслеживать статус заказа клиента в интернете [15].

Выводы по разделу один

В этой главе описываются системные требования CRM. Сначала была проанализирована область развития. При изучении тематической области автоматизации учитываются основные особенности работы сервисного центра, формулируются основные системные требования для учета заказов.

Требования к интерфейсу и функциям также были проанализированы и четко определены. Мы обнаружили, что система должна быть не только оснащена привлекательным, понятным, последовательным и гибким пользовательским интерфейсом, но и нуждаться во всех функциях для работы.

2 СРАВНЕНИЕ ОТЕЧЕСТВЕННЫХ И ПЕРЕДОВЫХ ЗАРУБЕЖНЫХ ТЕХНОЛОГИЙ И РЕШЕНИЙ

2.1 Обзор современных CRM-систем

В ходе выполнения работы был проведен анализ уже существующих CRM-систем с требуемым функционалом. Существует несколько аналогов подобной системы:

WorkPan – новая система для автоматизации бизнес процессов в ремонтной мастерской или сервисного центра. Она отлично подходит для мастерских, где ремонтируется разная техника: ноутбуки, планшеты, телефоны и для прочей техники. Функционал системы включает в себя: уведомления по SMS, учет расходных материалов и ремонта, финансовый учет и единую базу клиентов. Поддерживается возможность управления целой сетью мастерских из нескольких точек. Стоимость сервиса начинается от $19.99/ месяц [6]. Интерфейс представлен на рисунке 2.1.

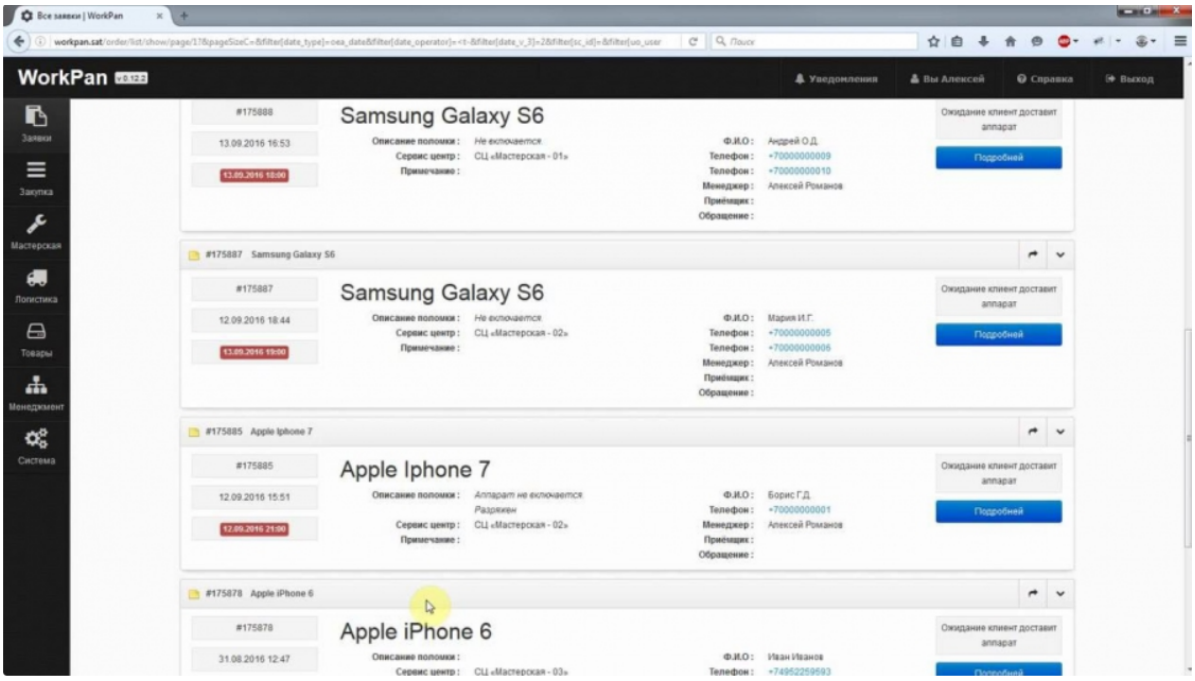

Рисунок 2.1 – Интерфейс web сервиса «WorkPan»

ServiceCenter – программа предназначена для организаций, занимающихся ремонтом в промежутках между цифровыми устройствами. Имеет простой в освоении, интуитивно понятный пользовательский интерфейс программы. Минусами данной программы является то, что она отслеживает заказы только одного агрегата, подходящего для небольшого сервисного центра. Техническая

поддержка сайта практически отсутствует [4]. Новые версии программы выходят достаточно редко. Интерфейс представлен на рисунке 2.2.

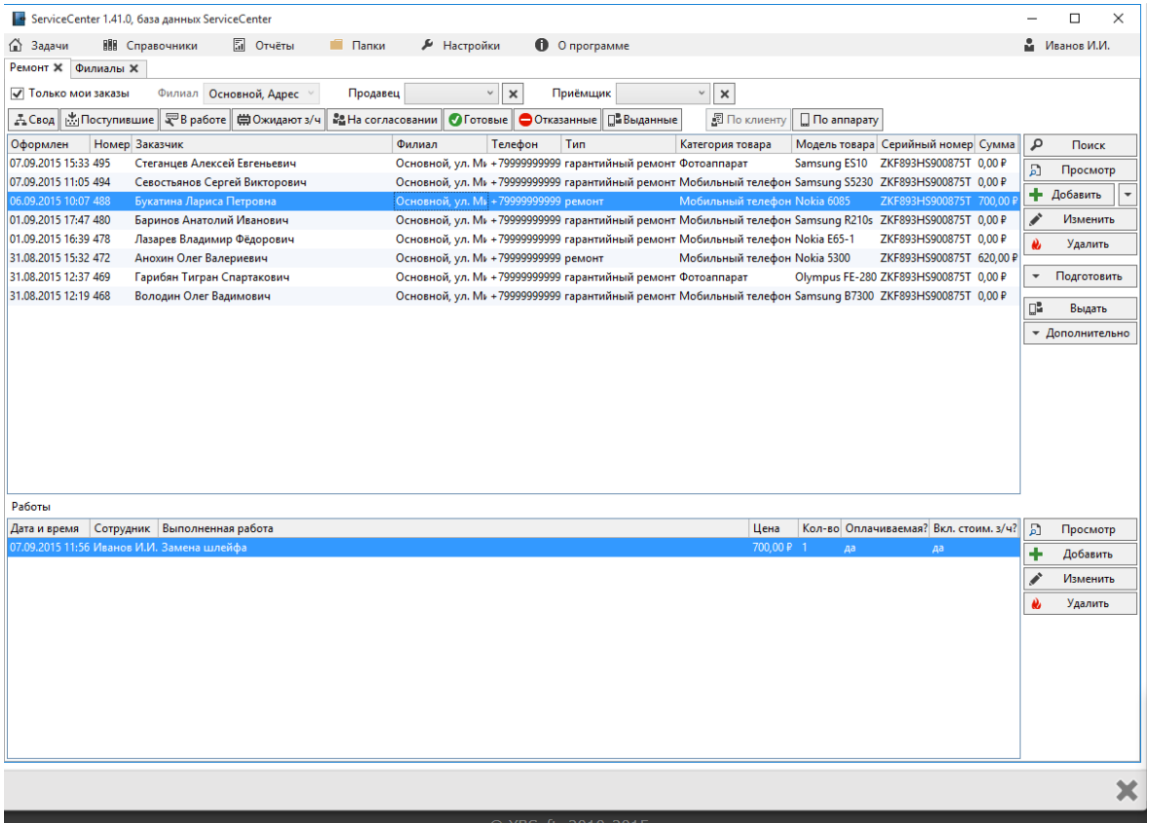

Рисунок 2.2 – Интерфейс программы «ServiceCenter»

«Gincore – web-сервис, имеет похожий функционал, однако стоимость пользования данным ресурсом достаточно велика» [2]. Интерфейс представлен на рисунке 2.3.

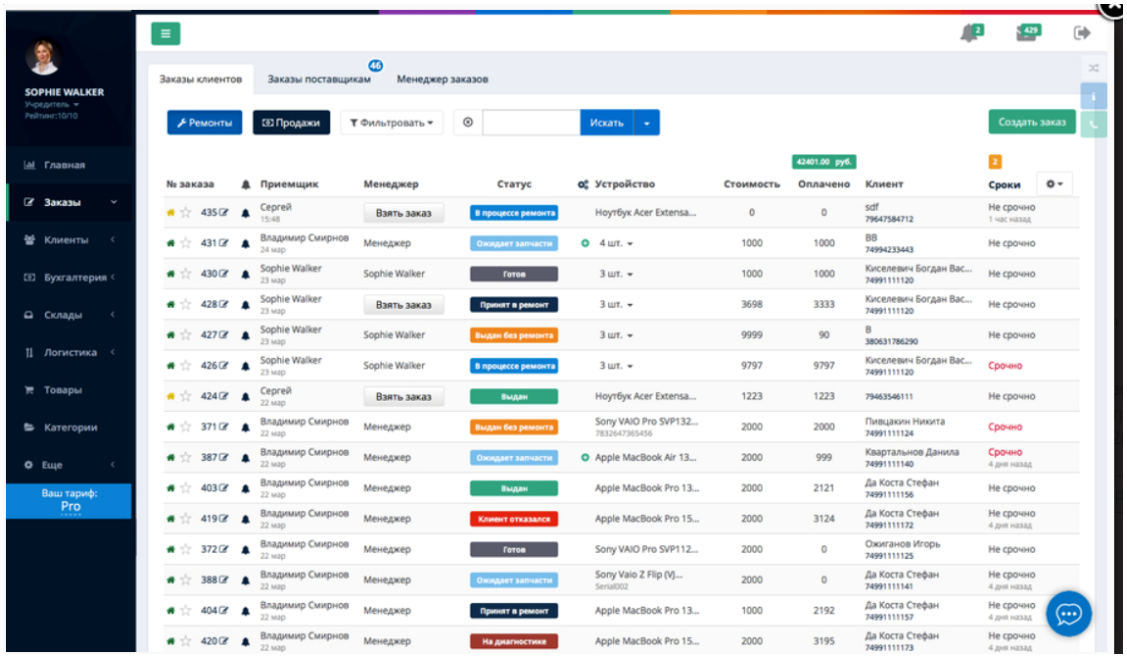

Рисунок 2.3 – Интерфейс web сервиса «Gincore»

«1С – Один из наиболее подходящий вариантов, в плане функционала. Продуманы практически все интересующие заказчика моменты, однако высокая стоимость, заставляет отказаться от данной системы» [1].

Сравним все найденные аналоги и подведем итоги в таблице 2.1.

Таблица 2.1 – Сравнение аналогов по ключевым параметрам

Критерий	WorkPan	ServiceCentr	Gincore	1С:Предприятие 8 Управление сервисным центром
Доступ из веб браузера	+	-	+	-
Бесплатное распространение	-	-	-	-
Возможность отслеживать перемещения заказа между подразделениями	-	-	-	+
Прикрепление файлов к заказу	-	-	-	+
Неограниченное количество мастерских и сотрудников	-	-	+	+

У этих систем есть свои плюсы и минусы. Аналог найден интуитивно понятный интерфейс для пользователей, доступность информации, в соответствии с различными форумами, и обеспечивают высокую скорость пользователя.

Однако во многих системах клиенту не хватает определенной функциональности. В «Gincore» и «WorkPan» владельцам ресурсов доступна вся информация о центре работы, поскольку клиент не готов к работе.

Служба «WorkPan» постоянно приставала к навязчивому обслуживающему персоналу с телефонными звонками и электронной почтой.

2.2 Выбор инструментальных средств разработки

Чтобы решить эту проблему, необходимо определить инструменты и технологии, с помощью которых осуществляется разработка этой системы. Поскольку одним из основных требований является доступ к Системе через Интернет-Браузер, мы вынуждены разрабатывать веб-приложения. Использование CMS для разработки этого приложения невозможно, так как все CMS имеют ограниченную функциональность, а приложения, имеющие сложную бизнес-логику, довольно сложно разрабатывать с помощью таких систем. Поэтому в большинстве случаев такие приложения разрабатываются независимо друг от друга с использованием языков программирования высокого уровня.

Была выбрана трехступенчатая архитектура. Первый уровень – это клиентская часть, она содержит пользовательский интерфейс и всю логику, необходимую для организации взаимодействия пользователя с системой. Второй уровень – сервер, будет реализована основная логика приложения: обработка данных, отчетность, взаимодействие с различными системами. Третий уровень – это база данных, в которой хранятся все данные, необходимые для приложения.

После того, как вы определили архитектуру приложения, вы должны выбрать язык и инструменты разработки.

Поскольку пользователь будет работать с приложением с помощью Интернет-браузера, логично использовать в клиентской части пакета HTML, CSS и JavaScript. HTML – это стандартизированный язык разметки документов, который облегчает разработку пользовательского интерфейса [17].

CSS – это формальный язык для описания внешнего вида документа, написанного на языке разметки. JavaScript – это объектно-ориентированный язык программирования, часто используемый для формирования логики клиентской части приложения. Этот пакет является стандартом в разработке веб-приложений [3].

Для разработки серверной части этой системы был выбран язык программирования Java – типизированный объектно-ориентированный язык программирования, разработанный Sun Microsystems. Этот язык был выбран по нескольким причинам:

- кросс-платформенная переносимость приложений на разных платформах;
- строгая типизация;
- с очистителем мусора, вы не можете тратить время на разработку приложения для обнаружения и исправления утечек памяти;
- много информации о тонкостях языка;
- высокоразвитое интернет-сообщество [20].

Эти и другие преимущества этого языка позволяют быстро разрабатывать различные приложения.

При выборе СУБД необходимо соответствовать нескольким критериям: бесплатному распространению, поддержке производителя, удобству и простоте работы, доступности информации в Интернете, надежности, производительности.

Следующие системы управления базами данных подпадают под этот критерий: Firebird (FirebirdSQL), H2, MySQL, PostgreSQL. Для разработки приложения было решено использовать MySQL, так как это самая известная СУБД для разработчика. Разработка и поддержка MySQL осуществляется корпорацией Oracle. MySQL – это решение для малых и средних приложений. Кроме того, эта СУБД очень популярна. Он имеет полезный инструмент для разработки mysql workbench. Это позволяет легко создать схему базы данных, написать и выполнить все запросы к базе данных.

Для разработки клиентской части приложения самым быстрым способом было решено использовать генераторы HTML-страниц. Преимущества этого подхода включают в себя:

- быстрое развитие пользовательского интерфейса;
- наличие готового визуального компонента;
- гибкий способ интеграции стандартных, сторонних или собственных программных компонентов на страницы [18].

Основным недостатком такого подхода является увеличение нагрузки на сервер. В интернете можно найти следующие технологии: Apache Wicket, Tapestry, OpenLaszlo, ZK. Все представленные фреймворки позволяют создавать пользовательские интерфейсы для веб-приложений с использованием языка Java для решения большинства проблем.

Apache Wicket – это платформа с открытым исходным кодом для создания веб-приложений. Разработано Джонатаном Локком в 2004 году. С июня 2007 года это проект Apache Software Foundation. Использование этой технологии требует знаний HTML и JAVA, можно использовать технологию AJAX без написания кода в JavaScript [22]. Файл шаблона разметки представлен на рисунке 2.4.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:wicket="http://wicket.apache.org/dtds.data/wicket-xhtml1.3-strict.dtd"
xml:lang="en" lang="en">
<body>
<span wicket:id="message" id="message">Message goes here</span>
</body>
</html>
```

Рисунок 2.4 – Файл шаблона разметки на HTML фреймворка Apache Wicket

OpenLaszlo – это платформа с открытым исходным кодом, предназначенная для разработки и доставки богатых интернет-приложений. Приложение OpenLaszlo – это документ .lzx с XML – подобным синтаксисом [22]. На рисунке 2.5 представлен пример синтаксиса OpenLaszlo.

```
<canvas height="350" width="1050" bgcolor="#FFBE7D" >
<view width="500" height="250" align="center" valign="middle"
bgcolor="#FFFF66">
<text align="center" valign="middle">
<font size="25">Welcome to Hello World!</font>
</text>
</view>
</canvas>
```

Рисунок 2.5 – Пример синтаксиса OpenLaszlo

Tapestry – это структура для создания веб-приложений, которые реализуют шаблон Model View Controller (MVC). Гобелен был создан Говардом Льюисом Шипом и продолжает активно развиваться. Структура позволяет легко использовать технологию AJAX, проверку данных, обеспечивает возможность поиска веб-приложений [22]. Пример синтаксиса см. на рисунке 2.6.

```
<t:eventlink event="updateTime" async="true">update</t:eventlink>
...
<t:zone t:id="timeArea" id="timeArea">
The current time is ${currentTime}
</t:zone>
```

Рисунок 2.6 – Пример Синтаксиса Tapestry

ZK – фреймворк для разработки веб-приложений доступна как в бесплатной, так и в платной версии, что облегчает разработку пользовательского интерфейса программы. Он имеет синтаксис, похожий на XML.

ZK надежен и построен с солидным послужным списком в качестве базы выбора во всех отраслях промышленности. Десятки тысяч разработчиков используют ZK для создания своих критических систем, в том числе миллионные системы для миллионов пользователей и десятки тысяч параллельных сеансов на международном уровне [29].

Файлы для презентации пользователю имеют расширение *.zul [22]. В таблице 2.2 приведены результаты сравнения данных кадра.

Таблица 2.2 – Сравнение генераторов HTML страниц

Критерий	Apache Wicket	OpenLaszlo	Tapestry	ZK
Наличие множества готовых компонент	-	-	-	+
Возможность управлять содержимым страницы с помощью Java кода	+	+	+	+
Отсутствие необходимости знать HTML	-	+	+	+
Взаимодействие по шаблону MVVM	-	-	-	+
Наличие большого количества примеров использования	+	+	-	+

Среди представленных фреймворков была выбрана технология ZK для решения проблемы взаимодействия с пользователем. Поэтому мы представим вам более подробное описание.

Zkoss преобразует файлы в формат *.zul в html документе по внутренней логике фреймворка. Синтаксис языка отличается от html, но поддерживает все его конструкции. Поддержка JavaScript и скрипта. Существует поддержка CSS3 и CSS. Сам Zkoss генерирует запрос на сервер, а также обрабатывает данные для отображения пользователю.

Это позволяет работать над двумя шаблонами проектирования MVC и MVVM.

Приложение, которое вы разрабатываете, использует второй метод, поскольку он более продуктивен и прост в развертывании. На рисунке 2.7 показан принцип применения в соответствии с шаблоном MVVM.

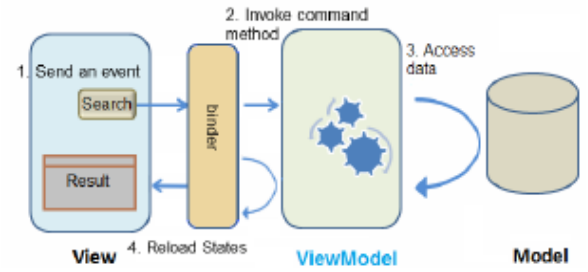

Рисунок 2.7 – Принцип работы приложения по шаблону MVVM

Также Zkoss поддерживает другой, пожалуй, самый популярный фреймворк в языке java – Spring. Во всех ViewModel используемых Zkoss необходимо использовать вместо аннотации @Autowired или @Inject аннотацию @WireVariable или @Wire.

Spring Framework – это универсальная платформа с открытым исходным кодом для платформы Java. Spring предлагает решения многих проблем, с которыми сталкиваются разработчики и организации Java, которые хотят создать информационную систему на основе платформы Java.Spring не полностью привязан к платформе Java Enterprise, несмотря на ее массовую интеграцию, что является важной причиной ее популярности [35].

Авторизация необходима для отключения прав пользователей и обеспечения безопасности. Каждый пользователь имеет уникальное имя пользователя и пароль. Каждому пользователю присваиваются определенные права доступа к данным [5]. Пользовательские данные и роли хранятся в базе данных. Spring Security был выбран в качестве инструмента безопасности. Эта структура предоставляет комплексные службы безопасности для корпоративных программных приложений на основе Java EE. Особое внимание уделяется поддержке проектов, созданных с помощью Spring Framework.

Для организации взаимодействия базы данных и приложений на языке Java существует специальная спецификация-JPA (Java Persistence API) [38].

В этой спецификации описывается система управления для удобного хранения объектов Java в таблицах реляционной базы данных. Сам язык не имеет запланированной реализации этой спецификации, но есть много реализаций этой спецификации от разных компаний. Этот метод хранения объектов Java в базе данных является самым популярным из того, что используется в этом проекте.

Реализация спецификации JPA использует Hibernate, библиотеку языка программирования Java для решения проблем реляционного сопоставления объектов (Orm). Это самый популярный фреймворк [5] для работы с базами данных.

Чтобы выполнять автоматизированные рутинные операции, такие как создание, компиляция, выполнение тестов и добавление библиотек в проект, вам нужно использовать специальную структуру сборки. Наиболее популярными компиляторами являются: Apache Maven, Gradle, Apache Ant. Сравниваем данные кадров, результаты сравнения приведены в таблице 2.3.

Поскольку Apache Maven является самым популярным из всех средств для строительства проекта, он используется в этом проекте.

Веб-приложения, разработанные на Java, работают с Java Servlet API. Сервлет – это Java-интерфейс, реализация которого расширяет функциональность сервера. Сервлет взаимодействует с клиентами по принципу запроса и ответа [37].

Таблица 2.3 – Сравнение средств автоматической сборки проекта

Критерий	Apache Maven	Gradle	Apache Ant
Независимость от OS	+	+	+
Управление зависимостями	+	+	-
Возможна сборка из командной строки	+	+	+
Интеграция со средами разработки	+	+	-

В общем случае связь между клиентом и сервлетом заключается в следующем: после того, как пользователь отправляет запрос серверу, сервер отправляет запрос контейнеру сервлета. Контейнер сервлета выполняет запрос и отправляет ответ в виде html на странице на веб-сервер.

Приложение требует наличия контейнера сервлетов и сервера. Apache Tomcat можно использовать одновременно как контейнер сервлетов и как сервер.

Использование Apache Tomcat является стандартом при разработке веб-приложений с использованием Java Servlet API. По сравнению с пристанью бежит быстрее.

Чтобы убедиться, что код проекта не потерян, вам придется использовать систему контроля версий. Система контроля версий позволяет вести историю изменений, следить за командой разработчиков, быстро общаться с удаленным хранилищем данных.

Среди популярных систем контроля версий можно выделить следующие: Git, SVN (Subversion), Ace [8]. Все системы имеют свои преимущества и недостатки. Этот проект использует GIT, потому что это распределенная система по сравнению с любой другой системой контроля версий с изменениями, а не файлами. Работа с Git намного проще, чем с SVN и Mercurial, и git отлично работает с такими редакторами, как Intellij Idea и Eclipse [40].

Архитектура программного обеспечения обычно относится к более крупной структуре программной системы, и речь идет о том, сколько программных процессов коммуникации для достижения своей задачи. Архитектура системы описывает основные компоненты, их взаимосвязи (структуры) и то, как они взаимодействуют друг с другом. Архитектура программного обеспечения включает в себя различные факторы, такие как бизнес-стратегия, атрибуты качества, человеческая динамика, дизайн и ИТ-среда.

Архитектура служит основой для системы. Он обеспечивает абстракцию для управления сложностью системы и создания механизма для связи и координации между компонентами. Он определяет структурированное решение, отвечающее всем техническим требованиям и требованиям при одновременной оптимизации общих атрибутов качества, таких как производительность и безопасность.

Кроме того, он содержит ряд важных решений организации, связанных с разработкой программного обеспечения, и каждое из этих решений может оказать существенное влияние на качество, расширяемость, производительность и успех конечного продукта.

Наиболее распространенным архитектурным паттерном является многоуровневый архитектурный паттерн, также известный как n-вспышка архитектурного паттерна. Эта модель является стандартом де-факто для большинства приложений Java EE и поэтому хорошо известна большинству архитекторов, дизайнеров и разработчиков. Структура многоуровневой архитектуры тесно связана с традиционными ИТ-коммуникационными и организационными структурами, встречающимися в большинстве фирм, что делает ее естественным выбором для большинства усилий по разработке бизнес-приложений. Части в структуре многослойной архитектуры организованы в горизонтальные слои, каждый из которых выполняет определенную роль внутри приложения (например, логика представления или бизнес-логика) [25]. Хотя шаблон многоуровневой архитектуры не определяет количество и типы слоев,

которые должны присутствовать в шаблоне, большинство из них являются архитектурой слоев.

Состоит из четырех стандартных уровней: уровень презентации, бизнес-уровень, уровень связи с базой данных и база данных. В некоторых случаях бизнес-уровень и уровень взаимодействия с базой данных объединяются в один бизнес-уровень, особенно если логика взаимодействия с базой данных (например, система или HSQL) находится в части бизнес-уровня.

Таким образом, небольшие программы могут иметь только три уровня, но более крупные и сложные бизнес-приложения могут включать пять или более уровней.

Каждый уровень многоуровневой архитектуры шаблона играет свою роль и несет ответственность внутри приложения. Например, уровень представления отвечает за обработку всего пользовательского интерфейса и логики взаимодействия браузера, но бизнес-уровень отвечает за реализацию конкретных бизнес-правил для запросов. Каждый слой в архитектуре формирует абстракцию работы, которую необходимо выполнить для выполнения определенного бизнес-процесса [39].

Одной из мощных особенностей многоуровневой архитектуры является разделение задач между компонентами. Часть внутри уверенности относится только к логике, которая обращается к уровню [25].

В результате выбора инструментальных средств разработки была сформирована следующая архитектура приложения, изображенная на рисунке 2.8.

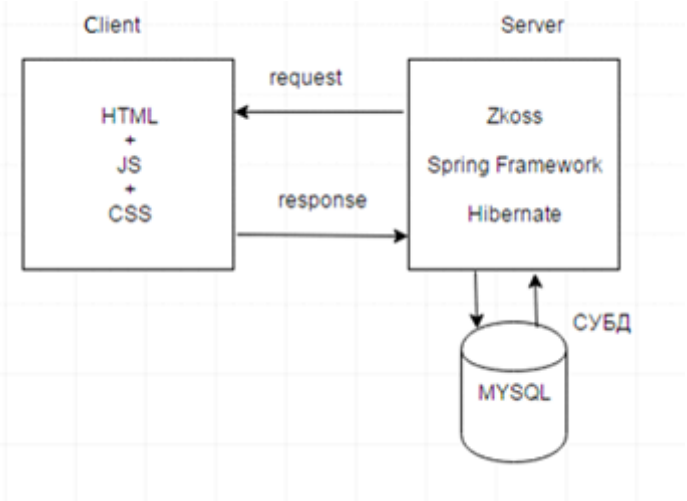

Рисунок 2.8 – Архитектура приложения

Серверная часть приложения разделена на слои, ответственные за определенный тип задачи. Всего 4 уровня: слои, отвечающие за пользовательский интерфейс – User Interface(UI), слой сервисов отвечающих за логику работы приложения – Business Service(BS), слой для работы с базой данных – Data Access(DA) и слой хранения данных [28] в качестве данного слоя выступает выбранная нами СУБД MySQL.

Выводы по разделу два

Несмотря на все преимущества представленной системы, палата отказывается использовать ее по разным причинам. Во-первых, по экономическим причинам. Чтобы купить лицензию, вы платите ежегодную плату за использование продукта и нанимаете серверы для хранения информации, которая стоит организации больших денег. Во-вторых, есть планы по продаже франшиз сервисного центра в будущем. Соответственно, было решено разработать новую систему.

Чтобы решить эту проблему, мы определили инструменты и методы, используемые для разработки этой системы. Была выбрана трехуровневая архитектура: клиент, сервер и база данных. На клиентской части мы используем связку HTML, CSS и JavaScript. Для разработки серверной части данной системы был выбран язык Java. Данный язык был выбран по ряду причин:

- кросс-платформенный обеспечивает переносимость приложений на различных платформах;
- много нажмите;
- наличие сборщика мусора позволяет разрабатывать приложение, не тратя время на анализ и исправление утечек памяти;
- большое количество информации о тонкостях языка;
- высокоразвитое сообщество [20].

MySQL был выбран в качестве СУБД, поскольку это самая известная СУБД для разработчиков. Кроме того, эта СУБД очень популярна. У него есть инструмент для разработки mysql workbench. Это позволяет легко создавать, писать схему базы данных и выполнять все запросы в базу данных.

В рамках для решения проблемы общения с пользователем была выбрана технология ZK. Для автоматизированного выполнения рутинных операций таких как компиляция, сборка, выполнение тестов и добавление библиотек к проекту будем использовать специализированный фреймворк-сборщик Apache Maven.

Apache Tomcat используется одновременно в качестве контейнера сервлета и сервера. В соответствии с популярной системой управления версиями git был выбран по многим причинам. Одним из них является высокое взаимодействие с редактором в качестве Intellij Idea, которая также используется.

3 ПРОЕКТИРОВАНИЕ CRM-МОДУЛЯ

3.1 Проектирование структур данных и алгоритмов

Алгоритм – это пошаговая процедура, которая определяет набор команд, которые должны выполняться в определенном порядке, чтобы получить желаемый результат.

Структура данных – это структура данных в памяти компьютера или даже на одном диске. Пример несколько распространенных структур данных массивы, связанные списки, являются очереди, стеки, двоичные деревья и хэш-таблицы ожидания. С другой стороны алгоритмы используются, чтобы управлять в этих данных, структур данных, таких как поиск и сортировка. Многие алгоритмы применяются непосредственно на определенные структуры данных. Если вы работаете с определенными структурами данных [10].

Для начала разработки сайта необходимо установить следующее программное обеспечение: GIT, Apache Maven, Intellij Idea, Tomcat 7.0. JDK 1.8. Если Intellij Idea выполняется, укажите расположение JDK и Apache Maven и создать пустой maven-проекта. В файле pom.xml указываем необходимые зависимости: ZK, Spring, Spring security, JPA, Hibernate.

Apache Maven автоматически загружает необходимые библиотеки из удаленного репозитория перед запуском приложения. Нам лишь необходимо указать версию необходимого нам фреймворка, groupId – наименование организации или подразделения, artifactId – название проекта. Также если есть необходимость указывается Так как один фреймворк может иметь большое количество библиотек необходимых для разработки, а версии библиотек должны совпадать, для избежания всевозможных конфликтов, более удобного управления версиями библиотек и избежания дублирования информации версии библиотек прописываются в переменные [10].

На рисунке 3.1 приведен пример конфигурации зависимости в файле pom.xml для фреймворка ZK.

```
<properties>
<zk.version>8.0.3.1</zk.version>
</properties>
<repositories>
<repository>
<id>zk repository</id>
<url>http://mavensync.zkoss.org/maven2</url>
</repository>
</repositories>
<dependency>
<groupId>org.zkoss.zk</groupId>
<artifactId>zkplus</artifactId>
<version>${zk.version}</version>
</dependency>
<dependency>
<groupId>org.zkoss.zk</groupId>
<artifactId>zk</artifactId>
<version>${zk.version}</version>
</dependency>
```

Рисунок 3.1 – Пример конфигурации зависимостей в maven

В Intellij Idea есть панель управления сборщиком maven в которой представлены основные команды для работы с данным инструментом [34]. После нажатия кнопки «eimport All Maven projects» произойдет автоматическое скачивание необходимых библиотек и удаление ненужных. По аналогии прописываем все остальные зависимости.

3.2 Проектирование пользовательского интерфейса

Дизайн пользовательского интерфейса (UI) или разработка пользовательского интерфейса-это разработка пользовательских интерфейсов для машин и программного обеспечения, таких как компьютеры, бытовая техника, мобильные устройства и другие электронные устройства, с акцентом на максимизацию юзабилити.

Цель дизайна пользовательского интерфейса –сделать взаимодействие с пользователем максимально простым и эффективным для достижения целей пользователя (user-oriented design) [23].

Хороший дизайн пользовательского интерфейса облегчает задачу под рукой, не привлекая особого внимания. Графический дизайн и типография поддерживают юзабилити, влияют на то, как пользователь выполняет определенные взаимодействия, и повышают эстетическую привлекательность дизайна.

Дизайн может улучшить или уменьшить способность пользователей использовать функции интерфейса. В процессе проектирования необходимо привести техническую функциональность и визуальные элементы (например, ментальные модели) в соответствии друг с другом, чтобы создать системы, способные жить только, но и по цене к изменившимся потребностям пользователей является настраиваемым. Дизайн интерфейса связан с различными проектами, от компьютерных систем до автомобилей до коммерческих самолетов.Все эти проекты включают в себя одно и то же базовое человеческое взаимодействие, но также требуют уникальных навыков и знаний. В результате разработчики, как правило, специализируются на определенных типах проектов и имеют навыки, ориентированные на опыт, будь то разработка программного обеспечения, исследования пользователей, веб-дизайн или промышленный дизайн [23].

На рисунке 3.2 изображен пример макета пользовательского интерфейса.

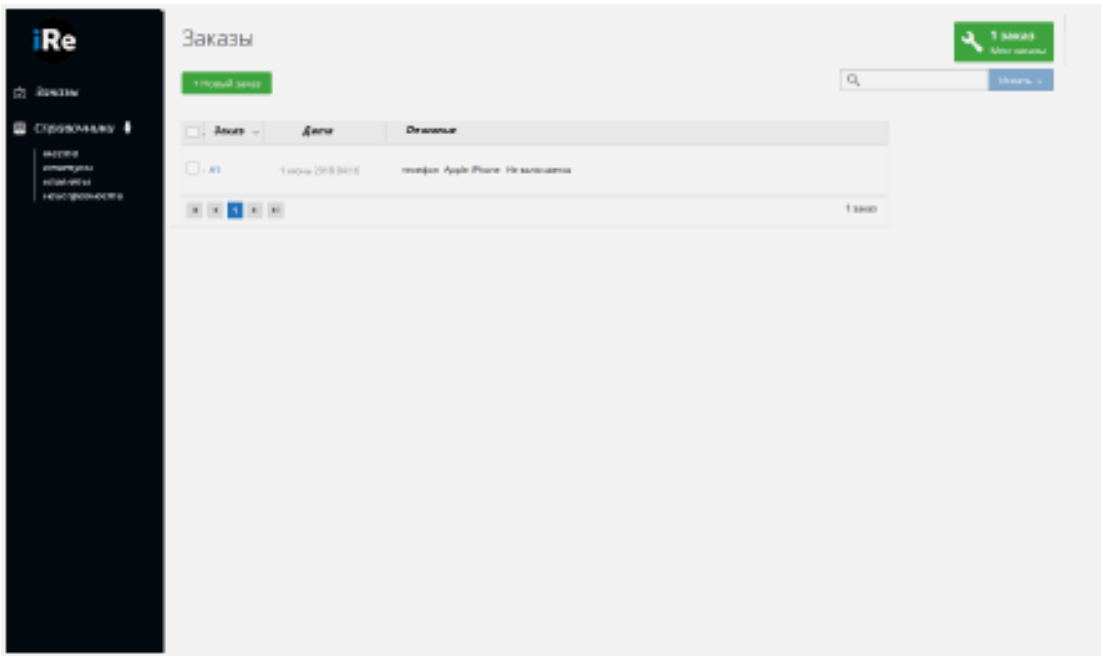

Рисунок 3.2 – Пример макета пользовательского интерфейса

3.3 Разработка базы данных

Как уже упоминалось, решение об использовании MYSQL было принято в качестве СУБД. В базе данных мы будем хранить всю необходимую информацию для обеспечения работы сервисного центра.

Таблица USERS хранит информацию о пользователях системы, содержит следующие поля:

- ID INT (11) – идентификационный номер;
- NAME VARCHAR (45) – имя пользователя;
- SECOND_NAME VARCHAR (45) – отчество;
- FAMILY_NAME VARCHAR (45) – фамилия;

- EMAIL VARCHAR VARCHAR (45) – электронная почта;
- PASSWORD – пароль для доступа к системе хранится в зашифрованном виде;
- PLACE_ID INT (11) – идентификация местоположения, к которому привязан пользователь системы.

Каждый пользователь имеет определенный набор прав, хранящихся в системе, полный список прав в таблице прав:

- ID INT (11) – идентификационный номер;
- TYPE VARCHAR (45) – тип ресурса в системе;
- таблицы USERS и RIGHTS имеют отношение многих ко многим.

Каждый филиал имеет определенный адрес, который затем учитывается при найме сотрудников, размещении заказов и других бизнес-процессах. Таблица мест имеет четыре поля:

- ID INT (11) – идентификационный номер;
- ADDRESS VARCHAR (255) – полный адрес сервисного центра.

Следующие два поля используются для визуального отображения отсека на карте:

- LAT DOUBLE – координата широты;
- LNG DOUBLE – координата долготы.

Таблица запросов используется для хранения рабочих данных и имеет четыре поля:

- ID INT (11) – идентификационный номер;
- DATE DATE – дата заказа;
- Описание VARCHAR (45) – описание заказа;
- CLIENT_ID INT (11) – ссылка на клиента, который разместил заказ [11].

Каждый заказ меняет статус в зависимости от фазы акции, в которой вы находитесь. Все состояния хранятся в специальной таблице:

- STATUS:
- ID INT (11) – идентификационный номер;
- STATUS VARCHAR (255) – имя состояния.

Таблицы request и STATUS имеют отношение многих ко многим.

Таблица подключения – это таблица STATUS_HISTORY. Эта таблица сохраняет:

- Дата дата-дата добавить статус в заказ;
- STATUS_ID INT (11) – ссылка на Status;
- REQUEST _id INT (11) – ссылка на заказ.

В таблице DEFECT хранятся ошибки техники, есть только два поля:

- ID INT (11) – идентификационный номер;
- NAME VARCHAR (45) – имя ошибки.

Поскольку каждое задание может иметь несколько ошибок с одинаковой ошибкой в разных заданиях, таблица DEFECT связана со многими ко многим отношениями с таблицей REQUEST.

Информацию о клиенте можно найти в таблице CLIENT:

- ID INT (11) – идентификационный номер;
- NAME VARCHAR (45) – имя пользователя;
- SECOND_NAME VARCHAR (45) – отчество;
- FAMILY_NAME VARCHAR (45) – фамилия;
- ADDRESS VARCHAR (45) – адрес размещения (используется при доставке заказа на дом);
- PHONE VARCHAR (45) – номер телефона для связи.

Во время ремонта заказ перемещается от одного сотрудника к другому, и необходимо хранить информацию о том, какой сотрудник в настоящее время имеет заказ. Для этой цели таблица запросов и USERS связаны таблицей REQUEST_USERS со многими отношениями ко многим:

- ID INT (11) – идентификационный номер;
- DATE DATE – дата;
- REQUEST _id INT (11) – ссылка на заказ;
- User_id INT (11) – ссылка на пользователя системы [7].

Листинг создания базы данных (Приложение А).

3.4 Разработка программной части системы

Эта фаза также известна как фаза программирования. Реализация дизайна программного обеспечения начинается с написания программного кода на соответствующем языке программирования и эффективного использования исполняемых программ без ошибок. Цель этапа реализации-перевести дизайн системы в код на определенном языке программирования. Для этого проекта целью этого этапа является наилучшая реализация дизайна. Фаза кодирования влияет как на тестирование, так и на техническое обслуживание. Хорошо написанный код снижает затраты на тестирование и техническое обслуживание. Поскольку затраты на тестирование и обслуживание программного обеспечения намного выше, чем затраты на кодирование, цель кодирования должна заключаться в сокращении затрат на тестирование и техническое обслуживание. Поэтому во время кодирования основное внимание должно быть уделено разработке программ, которые легко писать. На этапе кодирования надо стремиться к простоте и ясности [13]. Поскольку этот проект имеет слоистую архитектуру, следующее объясняет больше о разработке каждого уровня программного продукта.

Поскольку все данные приложения хранятся в базе данных, необходимо настроить взаимодействие СУБД и систему разработки. Язык Java является объектно-ориентированным языком программирования, поэтому все данные в нем должны быть представлены в виде объектов. Структура Hibernate позволяет отображать реляционные данные в объектной форме. В настоящее время в базе данных находятся двенадцать сущностей. Поэтому необходимо создать объект для каждого объекта, отображающего структуру объектов в базе данных. В структуре объектов вам просто нужно указать атрибуты базы данных, необходимые для

работы приложения, что уменьшает объем оперативной памяти, необходимый приложению.

Чтобы использовать выбранную реализацию JPA фреймворка Hibernate, вам нужно записать зависимость в файл POM.xml и повторный импорт всех библиотек. Maven автоматически загружает все указанные библиотеки.

Для работы с этой платформой необходимо указать адрес, по которому находится база данных, имя схемы, имя пользователя и пароль в базе данных [2].

Через созданный класс объектов необходимо указать две аннотации @ entity, а @Table в атрибутах последнего должен указать имя сущности в базе данных. Вместо аннотаций также приемлемо использовать конфигурацию XML. Каждое поле класса имеет аннотацию @ Column, атрибуты которой указывают имя атрибута объекта в базе данных, а также дополнительные параметры. Также может быть подмножество других сущностей в качестве атрибута Java класса. Подмножества функций в классе Java представляют собой набор данных.

Использование этого метода хранения связано с тем, что в этом хранилище данных нет повторяющихся объектов. В классе Java аннотация @ OneToMany записывается в поле этого подмножества, атрибуты этой аннотации указывают, как извлекаются данные.

Есть две формы нетерпеливого и ленивого. Использование второго метода предпочтительнее, так как по сравнению с EAGER данные извлекаются из базы данных только во время вызова get к методу этого атрибута. Согласно спецификации класса Java, которая отражает сущность в базе данных, необходимо создать неприкосновенный конструктор без аргументов [33].

Для операций с объектами используется специальный интерфейс JPA EntityManager. Этот интерфейс позволяет выполнять основные функции, такие как добавление, изменение, поиск, удаление и блокирование изменений из других потоков [31]. EntityManager устанавливает только один сеанс взаимодействия с базой данных. Сам по себе EntityManager содержит интерфейс EntityTransaction, который позволяет создавать транзакции, применять изменения в базе данных и откатывать эти изменения назад. Использование транзакций позволяет безопасно вносить изменения в базу данных [2]. Реализация, специфичная для EntityManager, определяется классом EntityManagerFactory: это фабрика соединений, поддерживает подключение к базе данных, всю конфигурацию и кэш для работы с базой данных

Процесс добавления объекта в базу данных заключается в создании определенного экземпляра класса, заполнении его атрибутов данными и правильной проверке данных для заполнения. После этого используйте интерфейс EntityManager и его методы persist и merge.

На рисунке 3.3 представлена общая схема взаимодействия DataAccess слоя с базой данных с использованием связки JPA и Hibernate.

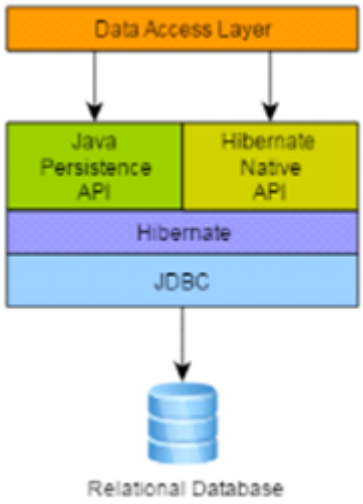

Рисунок 3.3 – Общая схема взаимодействия DataAccess слоя с базой данных

В этом проекте было создано 12 классов, которые отражают объекты в базе данных. Для каждого объекта на уровне базы данных был создан специальный класс, который отвечает за операции вставки, удаления, обновления и поиска.

Singleton используется в качестве шаблона для использования определенной службы DataAccess. Это позволяет использовать один экземпляр класса во всем приложении. Это достигается за счет использования Spring Framework и его способности управлять зависимостями инъекции зависимостей (DI). В Spring Framework существует несколько способов управления реализацией зависимостей: конфигурация XML, аннотации, использование специальных интерфейсов Spring Framework для настройки зависимостей с помощью Java Code. Первые два метода используются в этом проекте. Конфигурация XML определяет конкретную реализацию интерфейсов EntityManagerFactory и JpaTransactionManager. Службы Layer DataAccess, которые мы разрабатываем, настроены только с аннотациями. Annotation @Repository указывает, что этот класс действует как Repository, что требует прозрачной передачи исключений. Независимо от технологии, используемой на уровне доступа к данным, уровень обслуживания обрабатывает общую иерархию исключений Spring (DataAccessException). Spring открывает аннотацию @Transactional для классов, которые работают непосредственно с базой данных перед запуском этого метода, и после запуска метода транзакция фиксируется, и когда RuntimeException запускается, транзакция сбрасывается.

В этом проекте есть большая бизнес-логика для того, чтобы отделить бизнес-процессы от технических услуг системы, вся логика, связанная с бизнес-процессами, помещается в отдельный слой, называемый Business Service(BS) [36].

При разработке этого уровня активно используются Spring Framework и его способность управлять реализацией зависимостей [14].

В этом проекте, как описано выше, для каждого объекта, присутствующего в базе данных, есть объект, который отображает его в коде Java. По аналогии с

объектами, называемыми моделями, были созданы службы, которые позволяют работать с базой данных.

В настоящее время в рамках проекта создано 6 сервисов. ClientService отвечает за все операции, которые выполняются с клиентами: добавление нового клиента, поиск клиента по фамилии, изменение данных клиента. Каждая услуга включает в себя операции, выполняемые с моделями. Такие действия, как создание заказов, изменение заказов, удаление.

Пример: когда вы создаете новое задание, некоторые поля, которые вы хотите вставить в базу данных, будут автоматически заполнены в бизнесе службы уровня. Статус заказа, местоположение его регистрации и фактическое местоположение, а также ответственный сотрудник автоматически определяются путем повторного использования услуг. При таком подходе вы можете повторно использовать отдельные бизнес-процессы.

Объекты приложения имеют зависимости друг от друга. Хотя платформа Java предлагает множество функций разработки приложений, отсутствуют средства для организации основных строительных блоков в единое целое, что оставляет архитекторам и разработчикам эту задачу. Чтобы решить эту проблему, вы можете использовать шаблоны проектирования, такие как Factory, Abstract Factory, Builder, Decorator и Service Locator, для создания различных классов и экземпляров объектов, составляющих приложение.Эти шаблоны включают в себя: лучшие практики с именем, с описанием того, что делает шаблон, где он применяется, проблемы, с которыми он имеет дело, и т. д. шаблоны-это формализованные рекомендации, которые вы можете реализовать в своем приложении. Компонент Spring Framework Inversion of Controll (IOC) решает эту проблему, предоставляя формализованные инструменты проектирования для различных компонентов в полностью функционирующем приложении, которое готово к использованию [32]. Spring Framework кодирует шаблоны проектирования, формализованные как первоклассные объекты, которые могут быть интегрированы в ваши собственные приложения.

Управление зависимостями и реализация зависимостей-это разные вещи. Чтобы получить эти хорошие функции Spring в нашем приложении, вам нужно собрать все необходимые библиотеки (файлы jar) и поместить их в путь к классу во время выполнения и, возможно, во время компиляции.

Эти зависимости не являются виртуальными компонентами, импортированными в физические ресурсы файловой системы. Процесс управления зависимостями включает размещение, сохранение и добавление этих ресурсов в пути к классам. Зависимости могут быть прямыми (например, мое приложение зависит от среды выполнения Spring) или косвенными (например, мое приложение зависит от commons-dbcp , который зависит от общего пула). Косвенные зависимости также известны как «переходные», и эти зависимости сложнее идентифицировать и управлять.

Каждая корпоративная служба использует структуру Spring для реализации зависимости от службы, ответственной за взаимодействие с базой данных. С этой зависимостью Business Services работает с данными, находящимися в базе данных [5].

При необходимости веб-сайт должен иметь две страницы: страницу авторизации и домашнюю страницу, на которой отображается панель навигации и все элементы, необходимые для работы с веб-сайтом. Такая структура сайта обеспечивает наиболее удобное использование сервиса пользователем.

Благодаря структуре ZK пользовательский интерфейс развивается относительно быстро.

Поскольку приложение доступно через Интернет, вам нужно подумать о безопасном использовании приложения пользователем. Доступ пользователей к системе осуществляется через авторизацию и аутентификацию. «Аутентификация» – это процесс настройки объекта безопасности, что означает, что пользователь, устройство или другая система могут выполнять действие в приложении. «Авторизация» относится к процессу принятия решения о том, может ли участник выполнять действия [12] в приложении. В разделе Дизайн было решено использовать Spring Security для этих целей.

Spring Security предоставляет комплексное решение для обеспечения безопасности корпоративного программного обеспечения на основе приложений Java в США. Особое внимание уделяется проектам поддержки, созданным с помощью Spring Framework, ведущего решения Java EE для разработки корпоративного программного обеспечения.

Большинство причин включения Spring Security в проект – это функции безопасности спецификации Java EE Servlet или спецификации EJB, поскольку глубины, необходимой для типичных сценариев корпоративных приложений, недостаточно. Несмотря на упоминание этих стандартов, важно признать, что они не являются устойчивыми на уровне WAR или EAR. Поэтому, когда приложение переключается на разные серверные среды, обычно требуется много работы, чтобы превратить безопасность вашего приложения в новую целевую среду. Использование Spring Security позволяет вам преодолеть эти проблемы и дает вам десятки других полезных настраиваемых функций безопасности.

Spring Security имеет архитектуру, которая отделяет аутентификацию от авторизации и имеет стратегии и точки расширения для обои [5]. По левому краю страницы располагается навигационная панель содержащая главный пункт меню «Заказы» когда пользователь заходит на сайт фокус на данной странице установлен по умолчанию. Далее следует пункт меню «Справочники» с подпунктами «места», «Статусы», «Клиенты», «Неисправности». На рисунке 3.4 изображено меню сайта.

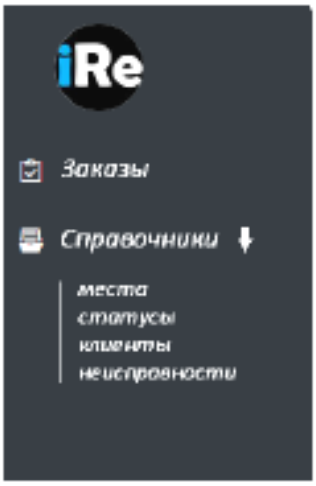

Рисунок 3.4 – Меню сайта

На вкладке задания в системе есть список новых заданий. Перейдите на вкладку «Справочники» дополнительно список категорий руководства.

Вкладка «Места» содержит список существующих мастерских и точек приема в этом сервисном центре.

В пункте меню «Статусы» в списке отображаются состояния, назначенные заказам во время выполнения бизнес-процессов.

Пункт меню «Клиенты» содержит список клиентов из сервисного центра, вы можете просмотреть список заказов для каждого клиента.

Каталог популярных ошибок техники находится в пункте меню «ошибка».

Чтобы облегчить разработку меню сайта, было решено использовать два шаблона взаимодействия клиент-сервер MVVM и MVC одновременно. Шаблон MVVM используется для заполнения компонента данными, а шаблон MVC-для управления поведением компонентов меню.

Нажатие на нижнюю кнопку меню сайта ломается влево. Механизм складывания выполняется путем изменения стиля CSS в Java-коде.

Чтобы создать этот компонент сайта, Вам нужно создать файл с расширением .zul, в нем, с помощью специального языка, чтобы описать, как должен выглядеть компонент. Кроме того, чтобы отформатировать отображение компонента на странице, необходимо указать стили компонентов CSS [26].На рисунке 3.5 представлен исходный код файла sidebar.zul.

```
<div id="sidebar" sclass="sidebar"
apply="uiController.SidebarComposer"> //Устанавливаем контроллер для всех элементов находящихся внутри тега div
<navbar id="navbar" sclass="nav-list" orient="vertical"> // Объявляем компонент navbar
<navitem label="Заказы" iconSclass="z-icon-home" selected="true" //Пункт меню
onClick="@command('showRequests')"// Команда которую выполнит ZK на событие клик мышью
<nav label="Справочники" iconSclass="z-icon-list"> // Раздел
<navitem label="Места" iconSclass="z-icon-angle-double-right" //Подпункты
onClick="@command('showPlaces')"/>
<navitem label="Статусы" iconSclass="z-icon-angle-double-right" onClick="@command('showStatus')" />
<navitem label="Клиенты" iconSclass="z-icon-angle-double-right" onClick="@command('showClients')" />
<navitem label="Неисправности" iconSclass="z-icon-angle-double-right"
onClick="@command('showDefects')" />
</nav>
</navbar>
<div sclass="sidebar-collapse"> //
<a id="toggler" iconSclass="z-icon-angle-double-left"/>
</div>
</div>
```

Рисунок 3.5 – Исходный код файла sidebar.zul

Чтобы создать меню, ZK Framework имеет специальный тег «navbar», с помощью этого тега вы можете легко и быстро создать меню для веб-сайта . Для этого компонента вы можете указать, как он должен отображаться, указав атрибуте «orien» значение «vertical» или «horizontal». Любой компонент, предоставляемый zk Framework, может указать стиль отображения CSS.

День «navitem» объявляет пункт меню, вы должны указать имя элемента, который отображается пользователю в атрибуте «label». Существует два основных способа установить значение для строковых констант на веб-сайте. Первый метод-поместить все значения в специальный файл с расширением.свойства, константы и их значение объявляются в этом файле. Значение атрибута label задается следующим образом: ${labels.* }, где указывается константа, значение которой необходимо для отображения пользователя вместо звездочки.

Этот подход используется для создания веб-сайтов, которые могут изменять язык пользовательского интерфейса, и этот подход позволяет использовать константу в разных частях пользовательского интерфейса. Второй метод-указать значение атрибута тега напрямую и заключить его в кавычки, которые отображаются пользователю. Поскольку веб-сайт не предусматривает изменения языка пользовательского интерфейса, второй метод используется во время разработки.

Для пункта меню заказы задается значение атрибута selected=«true», поскольку это главная страница веб-сайта, и пользователь должен отображать эту страницу при открытии страницы веб-сайта. Для обработки событий в форме требуется

механизм, который соединяет пользовательские команды и обрабатывает эти события на сервере [19].

Команда – это действие для управления свойством ViewModel.

Каждая команда указывает метод, который View может выполнять в ViewModel. Эти действия также позволяют пользователям взаимодействовать с представлением. Например, «ViewModel» предлагает две команды: «сохранить» и «отменить». Это означает, что пользователи могут выполнять только эти 2 действия в представлении с помощью ViewModel.

Поскольку ViewModel действует как Controller, можно связать событие компонента пользовательского интерфейса с командой, указав аналогичное имя команды при регистрации в событии Listener. Несколько событий могут быть связаны с одной и той же командой. Когда пользователь взаимодействует с компонентом (например, при нажатии кнопки), компонент запускает событие, модуль привязки данных инициирует выполнение команды. Команда реализована как метод ViewModel. Поскольку ViewModel является обычным объектом Java, метод должен быть аннотирован с помощью Annotation @Command с помощью ZK, чтобы механизм привязки данных распознал, какой метод является командой. Эти методы управляют свойством ViewModel, таким как удаление элемента.

Команда – это действие для управления свойством ViewModel. Каждая команда предоставляет действие, которое может выполнять представление в ViewModel. Необязательный элемент аннотации – это имя строки для имени команды, и это имя ссылается на PERM, который связан с событиями. Если не указано, имя метода устанавливается как имя команды по умолчанию.

Команда ViewModel похожа на обработчик событий, каждое событие связано с командой на сервере. Связь между событиями и командами называется «привязка команд событий». Прежде чем делать эту фиксацию, мы должны объявить команду с ее именем в ViewModel.

Имена команд в ViewModel не должны дублироваться, иначе исключение запускается во время выполнения [19].

Команды пунктов меню содержат информацию обо всех заказах на ремонт в системе. Пользователь этой страницы может просматривать, добавлять и редактировать задания. Для отображения данных в виде списка ZK Framework предоставляет готовый компонент «listbox», который поддерживает представление страницы по данным пользователю, который разбивает данные на столбцы и строки. Навигация по списку может быть выполнена с помощью клавиш управления [21].Это можно наглядно увидеть на рисунке 3.6.

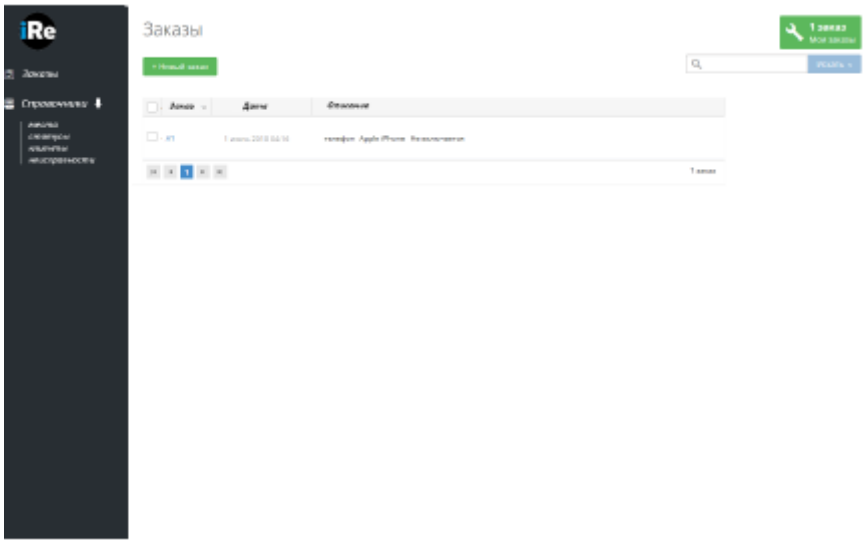

Рисунок 3.6 – Интерфейс пункта меню заказы

Чтобы создать этот компонент, вам нужно создать файл с расширением, похожим на компонент меню.Зул и прописывают необходимые этикетки и их атрибуты.

Связывание данных с серверным и клиентским браузерами похоже на буфер. Промежуточный объект создается автоматически. Перед сохранением в ViewModel все записи сохраняются в промежуточном объекте. Таким образом, мы можем хранить данные из сохраненного ViewModel для подтверждения пользователя. Когда пользователь завершает форму в веб-приложении, входные данные пользователя хранятся непосредственно в свойствах ViewModel, целевого объекта. Пользователь может отменить действие заполнения перед отправкой формы, чтобы данные, хранящиеся в модели представления, устарели. Это может привести к проблемам, если мы продолжим обрабатывать устаревшие данные, поэтому входные данные сначала хранятся в промежуточном объекте, прежде чем они переключатся на фактическую цель после подтверждения пользователя. Связывание формы гарантирует, что промежуточный объект хранит информацию, которая не подтверждена пользователем.

Привязка формы позволяет сохранять целевой объект в ViewModel без изменений до тех пор, пока не будет выполнена команда фиксации. Перед сохранением свойств ViewModel (целевого объекта) с помощью команды мы можем сохранить запись в промежуточном объекте. Если команда выполняется (например, кнопку нажали), хранится на самом деле запись в ViewModel свойства. Структура позволяет легко достичь этого эффекта, просто записывая выражение zk bind, поскольку оно освобождает разработчика от ручного удаления неправильных данных или реализации самого буфера [27]. Ниже, на рисунке 3.7 изображен поток данных между ZUL, объектом посредником и целевым объектом:

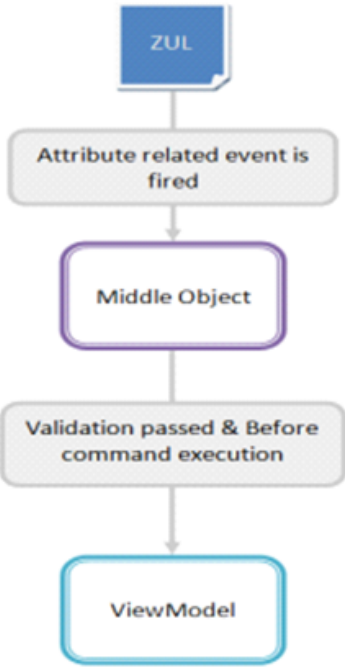

Рисунок 3.7– Поток данных между ZUL, объектом посредником и целевым объектом

На рисунке 3.8 представлен пример связывания данных в компоненте «listbox» для отображения списка заказов пользователю.

```
<listbox model="@bind(vm.placeListModel)" selectedItem="@bind(vm.selectedPlace)" style="margin-top:10px"
mold="paging" pageSize="10" onDoubleClick="@command('viewPlace')">
<listhead>
<listheader width="10%" label="№"/>
<listheader width="80%" label="Адрес"/>
</listhead>
<template name="model">
<listitem>
<listcell label="@bind(each.id)"></listcell>
<listcell label="@bind(each.address)"></listcell>
</listitem>
</template>
</listbox>
```

Рисунок 3.8 – Листинг файла place-list.zul

В первой строке на рисунке 3.8 объявляется компонент listbox, в атрибутах указывается из какого объекта в ViewModel необходимо брать данные для

заполнения списка и в какой объект, находящийся во ViewModel необходимо устанавливать выбранный пользователем объект из списка.

Следующие теги головка списка головка списка и появится в заголовках столбцов.

Тег Listcell показывает, какие поля, специфичные для объекта, будут отображаться пользователю. В верхней части каждой страницы является компонентом, который позволяет создавать, удалять, искать данные, а также кнопку, которая позволяет обновлять данные на странице.

При нажатии кнопки Добавить данные на странице заказа появляется форма создания заказа. Чтобы создать задание, вам нужно выбрать клиента или создать новый, добавить описание задания, добавить ошибки. Только тогда вы можете создать новую работу. Если вы нажмете кнопку выбора клиента, откроется новое окно выбора клиента, в котором вы можете создать нового клиента. После нажатия кнопки Создать все данные будут сохранены в базе данных.

Подтверждение ввода пользователем является незаменимой функцией веб-приложения. Валидатор ZK помогает выполнить эту задачу. Validator – это многоразовый элемент, который выполняет проверки и проверки сообщений в списке хранилищ сообщений проверки.

При применении он вызывается до того, как данные будут сохранены в целевой привязке (объект ViewModel или Staging). Если проверка не удалась, свойства ViewModel (или промежуточного объекта) не будут изменены.

Databinding предоставляет стандартный механизм для хранения и отображения сообщений проверки. После завершения проверки валидатор может сохранить сообщение проверки в списке сообщений фиксации [24].

3.5 Тестирование системы

Тестирование программного обеспечения – это исследование, проводимое с целью предоставления заинтересованным сторонам информации о качестве продукта или тестируемой услуги. Тестирование программного обеспечения также может иметь независимое, объективное представление программного обеспечения, что позволяет компании оценить и понять риски развертывания программного обеспечения.

Методы тестирования включают запуск программы или приложения с целью поиска программных ошибок (ошибок или других недостатков) и проверки того, подходит ли программный продукт для использования.

Поскольку количество тестов, возможных даже для простых программных компонентов, практически бесконечно, все тесты программного обеспечения используют стратегию выбора тестов, доступных для времени и ресурсов. В результате software testing обычно пытается (но не исключительно) запустить программу или приложение с целью поиска ошибок программного обеспечения (ошибок или других недостатков). Тестовое задание – это итеративный процесс,

который исправляет ошибку, освещает другие более глубокие ошибки или даже создает новые. Тестирование программного обеспечения может предоставить объективную и независимую информацию о качестве программного обеспечения и риске отказа пользователей или спонсоров [25].

Тестирование программного обеспечения может быть сделано, как только исполняемое программное обеспечение присутствует (даже если он частично завершен). Общий подход к разработке программного обеспечения часто определяет, когда и как проводятся тесты.

Например: в пошаговом процессе большинство тестов происходят после того, как системные требования определены, а затем реализованы в тестируемых программах. Напротив, при гибком подходе требования, программирование и тестирование часто выполняются одновременно.

Методы включают в себя составление плана, реализацию идеи, а техника-это метод или метод выполнения задачи. Таким образом, методы тестирования создают набор входов для конкретного программного обеспечения, которые дают ряд ожидаемых результатов. Идея заключается в том, что система работает достаточно хорошо и может быть запущена с минимальными проблемами для конечного пользователя.

Статические методы тестирования обеспечивают отличный способ улучшить качество и производительность разработки программного обеспечения. Он содержит обзоры и предоставляет обзор того, как они выполняются.

Основная цель статического тестирования-улучшить качество программных продуктов, позволяя инженерам обнаруживать и исправлять свои собственные ошибки на ранней стадии процесса разработки программного обеспечения.

При тестировании этого программного продукта применяются системные тесты. Системный тест – это метод тестирования черного ящика, который выполняется для оценки всей системы, отвечающей указанным требованиям. При тестировании системы функциональность системы проверяется с сквозной точки зрения. Системные тесты обычно проводятся командой, которая не зависит от команды разработчиков, чтобы беспристрастно измерить качество системы. Она включает в себя функциональные и нефункциональные тесты [25].

Тестирование разработка приложения осуществляется путем тестирования функции сайта, такие как добавление клиентов, добавление заказа, просмотр заказа, изменение заказа.

В тестах были проверены все функциональные элементы сайта. В результате проверки все функции сайта выполняются в соответствии с требованиями.В качестве требований к функциональному тестированию взяты данные диаграммы вариантов использования. Оценка соответствия функциональности программы к заявленным требованиям представлена в таблице 3.1.

Таблица 3.1 – Параметры оценки соответствия функциональности программы

Функция программы	Реализация	Вывод сообщения при ошибке
Регистрация/ авторизация	Реализовано при входе пользователя на сайт.	В случае ввода ошибочных данных выводится сообщение «Неверный ввод логина и/или пароля».
Просмотр списков данных	Выбор в пункте меню соответствующего списка данных.	В случае отсутствия записей в списке выводится сообщение «Список пуст».
Не все поля заполнены в формах создания	Создание нового пользователя, заказа, клиента или любой другой записи.	В случае заполнения не всех обязательных полей выводится ошибка над каждым полем.
При не возможности подключения к БД	Отправкой запроса к базе Данных.	В случае невозможности подключения к БД ошибка «Невозможно подключиться к БД».

Выводы по разделу три

Также было подробно рассказано о разработке каждого слоя программного продукта. При тестировании были проверены все функциональные элементы сайта. В результате тестирования все функции сайта выполняются согласно предъявленным требованиям.

ЗАКЛЮЧЕНИЕ

Были проанализированы требования к CRM-системе. Во-первых, анализируется область развития. При рассмотрении вопроса автоматизации, рассматриваются основные особенности работы сервисного центра, были сформулированы основные требования к системе учета заказов.

Также были проанализированы и четко определены требования к интерфейсу и функциональности. Мы обнаружили, что система должна быть оснащена не только привлекательным, понятным, последовательным и гибким пользовательским интерфейсом, но и всеми необходимыми функциями.

Обзор существующих CRM-систем с требуемой функциональностью.

Мы выбрали инструменты и технологии, с помощью которых осуществлялась разработка этой системы. Была выбрана трехступенчатая архитектура: клиент, сервер и база данных. В клиентской части он использует пакеты HTML, CSS и JavaScript. Для разработки бэкэнд-части данной системы был выбран язык Java. MySQL был выбран в качестве СУБД.

Среди фреймворков была выбрана технология ZK для решения проблемы взаимодействия с пользователем. Для автоматического выполнения рутинных операций, таких как компиляция, создание, выполнение тестов и добавление библиотек в проект, мы используем специальную среду сборки Apache Maven.

В то время система разработала базу данных, которая отражает основную логику хранения данных. Связь между приложением и базой данных организована. Были разработаны и протестированы основные запросы данных. Кроме того, осуществляется HTTP-связь между клиентом и сервером. Ведется разработка бизнес-логики и клиентской части приложения.

Функциональность была разработана, что позволяет реальным пользователям использовать эту информационную систему.

В будущем функциональность этой системы будет увеличена за счет появления новых потребностей. Эти требования включают в себя::

- Автоматический расчет стоимости заказа.
- Склад запасных частей.
- Интеграция разработанной системы с основным сайтом сервисного центра .
- Увеличение образца для подражания взаимодействия с сайтом .
- Возможность печати штрих-кодов для заказов .
- Возможность подключения сканера штрих-кода.
- Прикрепите фотографии, сделанные с помощью веб-камеры, к заказу.
- Панель администратора сайта.

Все вышеперечисленные требования будут реализованы в более поздних версиях программы.

На данный момент разработка первой версии программы завершена.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 1С сервисный центр 2.0 [Электронный ресурс]: офиц. сайт. – Режим доступа: http://solutions.1c.ru/catalog/service-center/features

2 Gincore [Электронный ресурс]: офиц. сайт. – Режим доступа: https://gincore.net/

3 Joshua, B. Effective Java / B. Joshua. – Москва: КНОРУС, 2015. – 369 с.

4 ServiceCenter [Электронный ресурс]: офиц. сайт. – Режим доступа: http://www.ybsoft.ru/ServiceCenter.aspx

5 Spring Framework [Электронный ресурс]: офиц. сайт. – Режим доступа: http://spring-projects.ru/projects/spring-framework/

6 WorkPan [Электронный ресурс]: офиц. сайт. – Режим доступа: https://workpan.ru/

7 Голицына, О.Л. Базы данных: Учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. – М.: Форум, 2012. – 400 с.

8 Гончарик, Н.Г. Цифровые мультимедийные технологии – смысловые средства передачи информационного содержания // Проблемы создания информационных технологий: сб. науч. тр. – 2012. – Вып. 21. – С. 74–76.

9 Грекул, В.И Проектирование информационных систем / В.И Гренкул. – М.: ИНТУИТ.ру, 2009. – 135 с.

10 Гринберг, А.С. Информационные технологии управления: Учебное пособие для вузов / А.С. Гринберг, Н.Н. Горбачев, А.С. Бондаренко. – М.: ЮНИТИ, 2010. – 479 с.

11 Диго, С.М. Базы данных: проектирование и использование: Учебое пособие для вузов / С.М. Диго. – М.: Финансы и статистика, 2010. – 591 с.

12 Заботина, Н.Н. Проектирование информационных систем: учебное пособие / Н.Н. Заботина. – М.: ИНФРА-М, 2011. – 331 с.

13 Исаев, Г.Н. Информационные технологии: Учебное пособие / Г.Н. Исаев. – М.: Омега - Л, 2013. – 464 с.

14 Карпова, И.П. Базы данных: Учебное пособие / И.П. Карпова. – СПб.: Питер, 2013. – 240 с.

15 Коноплева, И.А. Информационные технологии: учебное пособие / И.А. Коноплева, О.А. Хохлова, А.В. Денисов . – М.: Проспект, 2010. – 294 с.

16 Максимов, Н.В. Современные информационные технологии: Учебное пособие / Н.В. Максимов, Т.Л. Партыка, И.И. Попов. – М.: Форум, 2013. – 512 с.

17 Максимюк, К.С. Новый интернет для бизнеса / К.С. Максимюк. – М.: Эксмо, 2011. – 458 с.

18 Малыхина, М.П. Базы данных: основы, проектирование, использование / М.П. Малыхина. – СПб.: БХВ Петербург, 2009. – 512 с.

19 Марков, А.С. Базы данных: Введение. в теорию и методологию: Учебник. / А.С. Марков, К.Ю. Лисовский. – М.: Финансы и статистика, 2009. – 511 с.

20 Машнин, Т.С. Технология Web-сервисов платформы Java / Т.С. Машнин. – БХВ-Петербург, 2012. – 560 с.

21 Мишенин, А.И. Теория экономических информационных систем / А.И. Мишенин. – М.: Финансы и статистика, 2010. – 240 с.

22 Официальная документация по Фреймворку ZK [Электронный ресурс]: инф.-справ. Система. – Режим доступа: http://books.zkoss.org/zk-mvvm-book/8.0/

23 Поспелов, Д.А. Логико-лингвистические модели в системах управления / Д.А. Поспелов. – М.: Энергоиздат, 1981. – 231 с.

24 Сборник задач по теории автоматического регулирования и управления / под редакцией В.А. Бесекерского. – М.: Наука, 1978. – 512 с.

25 Смирнова, Г.Н. Проектирование экономических систем: Учебник / Г.Н. Смирнова, А.А. Сорокин, Ю.Ф. Тельнов. – М. : Финансы и статистика, 2010. – 240 с.

26 Стяблина, А.В. Электронные технологии в формировании информационной среды // Вестн. Тамбов. ун-та. Сер.: Гуманитар. науки. – 2011. – Т. 103, № 11. – С. 207–211.

27 Сундукова, Т.О. О содержании понятия «информационная система» // Современные проблемы математики, механики, информатики: материалы конф. – Тула, 2006. – С. 263–268.

28 Тарасов, В.Л. Работа с базами данных в Access 2010. ЧАСТЬ 1: Учебно-методическое пособие / В.Л. Тарасов. – Нижний Новгород: Нижегородский государственный университет, 2014.

29 Тител, Э. HTML5 и CSS3 для чайников / Э. Тител, К. Минник. – Москва: Вильямс, 2017. – 400 с.

30 Трофимов, В.В. Информационные технологии: учебник для вузов / В.В. Трофимов. – М: Издательство Юрайт, 2012.

31 Фатхутдинов, Р.А. Организация производства / Р.А. Фатхутдинов. – Москва: Экономика, 2001. – 237 с.

32 Федорова, Г.Н Информационные системы Учебник 3-е издание, стереотипное / Г.Н Федорова. – М.: Издательский центр «Академия», 2013.

33 Филиппов, В.А. Многомерные СУБД при создании корпоративных информационных систем / В.А. Филиппов. – М.: Едиториал УРСС, 2014.

34 Хорстманн, К. Java SE 8. Вводный курс / К. Хорстманн. – Москва: Вильямс, 2014. – 208 с.

35 Хорстманн, К. Java библиотека профессионала / К. Хорстманн. – Москва: Вильямс, 2017. – 864 с.

36 Шелухин, О.И. Моделирование информационных систем / О.И. Шелухин, А.М. Тенякшев, А.В. Осин. – М.: Радиотехника, 2011.

37 Шилдт, Г. Java 8. Полное руководство / Г. Шилдт. – Москва: Вильямс, 2017. – 1376 с.

38 Шилдт, Г. Java 8. Руководство для начинающих / Г. Шилдт. – Москва: Вильямс, 2017. – 720 с.

39 Щеглов, С. Программное обеспечение и его классификация / С. Щеглов. – http://scheglov–sergey.narod.ru/prog_obes_pk_i_clas.htm

40 Эккель, Б. Философия Java / Б. Эккель. – Санкт-Петербург: Питер, 2017. – 1168 с.

ПРИЛОЖЕНИЕ А
Программный код

```
Класс State
Unit
@ Table (name = " client")
public class client implements Serializable {
@ID
@ GeneratedValue (Strategy = GenerationType.IDENTITY)
private whole id;
@ Column (name = "name", nullable = false)
private String name;
// TODO: єiiiiiiiiiiiiiii
@ Column (name = "MIDLE_NAME", which accepts null values = false)
Midline private string;
@ Column (name = "name", which accepts null = false)
private String last name;
@ Column (name = "address", nullable = false)
Private chain address;
@ Column (name = "phone", unique = true, nullable = false)
private phone String;
@Onetomany (fetch = FetchType.Looking forward, mappedBy = " client")
Private set < request > requests = New HashSet < request>();
Public integer getId() {
Return ID;
}
Integer identifier) {
Esto.id = id;
}
Public string get() {
return name;
}
public void setName) {
this.name = number;
}
Public string getmid() {
Back mid;
}
Sets the middle name of the string.) {
this.Data encoding method:;
}
public string get() {
return last name;
}
public void setSurname) {
this.Last Name = Last Name;
}
Getaddress public string() {
Address of sender;
```

```
}
Insert the public address (chain address) {

this.Address = Address;
}
GetPhone public string() {
back phone;
}
The pay-phone (phone string) {
this.Phone = Phone;
}
Gets a series of requests.() {
Claims for reimbursement;
}
Sets a series of requests.) {
this.Applications = Applications;
}
}

Stat StatusDao
@Repository
ClientDao public class {
@ PersistenceContext
Private EntityManager entityManager;
@Transaction
Public list < client> getAll() {
Query query = entityManager.createQuery ("select client C as c");
List < Client> Result = Query.getResultList();
Result;
}
@ Transactional (propagation = propagation . REQUIRES_NEW)
contracting authorities insertClient (client-client) {
entityManager.(the client);
entityManager.go (customers);
Return customers;
}
@ Transactional (readOnly = true)
The public client findClientByName (string client name) starts
UsernameNotFoundException {
Customer Customer;
try {
Query query = entityManager.createQuery ("select client C as C, where
How to do that?")
.You can use the client name.);
client = (client) query.getSingleResult();
```

```
yes (customer != zero) {
Sleep.initialize (client);
}
} Catch (exception e) {
return null;
}
return customer;
}
}
```

Интерфейс Customer Service

```
public interface client service {
GetAllClient < client > list();
SaveClient client (client-client);
FindByClientName client (string name);
}
```

Kimpl Service

```
Wireless network name (SSID):")
serviceimpl public class client implements client service {
@ Autowired
ClientDao clientDao;

@Overwrite
Public list < client> getAllClient() {
return client.getAll();
}
@Overwrite
contracting authorities saveClient (client-client) {
returns findByClientName (client.How can I do that?
clientDao.insertClient (client): null;
}
@Overwrite
Public client findByClientName (string name) {
return client.findClientByName (name);
}
}
```

Kvm client list

```
@VariableResolver (DelegatingVariableResolver.Class)
ClientListVM public class {
private window;
Wireless network name (SSID):
ClientService clientService;
private List Template < Client> Client List Template;
A private client, the selected client;
private string keyword;
Final public static string CREATE_CLIENT = " sc / create-client.permissible";
```

```
Public static end-string CLIENT = " client";
How can I do that?)
For more information, see the following example:component) window
Window) {
this.Window = Window;
Update();
}
@Command

public void addClient() {
Window wind = (window)
Execution.createComponents (CREATE_CLIENT, window, null);
Wind.doModal();
}
About us
This is the phone settings page.")
public update cancelled() {
Further information can be found under the following link: getAllClient());
((ListModelList<Client>) clientListModel).setMultiple (true);
}
@Command
public void close() {
Data coding: further information can be found in our Privacy Policy and terms of use,
Collection.< String, Object>singletonMap ("value", selectedClient));
Window.onClose();
}
Further information can be found under the following link:() {
client list model return;
}
Specifies the client list model.) {
this.You can use the customer list model to create a customer list.;
}
GetKeyWord public string() {
Return keyword;
}
Sets the keyword for the string.) {
this.Keyword = Keyword;
}
Get client public client() {
return selectedclient;
}
Sets the selected client.) {
this.selectedClient = selectedClient;
}
}
```

Код файла client-list.zul

```
<window
onClose="@command('close')"
apply="org.zkoss.bind.BindComposer"
viewModel="@id('vm') @init('ServiceCentre.ui.ClientListVM')">
<hbox width="100%" align="center">
Адрес:
<textbox value="@bind(vm.keyWord)"/>
<button iconSclass="z-icon-search" onClick="@command('search')"/>
<button iconSclass="z-icon-plus" onClick="@command('addClient')
@global-command('refresh')"/>
<button iconSclass="z-icon-refresh" onClick="@global-
command('refresh')"/>
</hbox>
<listbox height="100%" width="100%"
 model="@bind(vm.clientListModel)" emptyMessage="Клиентов не найдено"
selectedItem="@bind(vm.selectedClient)"
mold="paging" pageSize="10">
<listhead>
<listheader width="15%" label="№"/>
<listheader width="15%" label="Имя"/>
<listheader width="15%" label="Фамилия"/>
<listheader width="15%" label="Отчество"/>
<listheader width="15%" label="Адрес"/>
<listheader width="15%" label="Телефон"/>
</listhead>
<template name="model">
<listitem>
<listcell label="@bind(each.id)"></listcell>
<listcell label="@bind(each.name)"></listcell>
<listcell label="@bind(each.midleName)"></listcell>
<listcell label="@bind(each.surname)"></listcell>
<listcell label="@bind(each.address)"></listcell>
<listcell label="@bind(each.phone)"></listcell>
</listitem>
</template>
</listbox>
<button label="Сохранить" iconSclass="z-icon-save"
onClick="@command('close')" visible=" not @load('isMenuMod')"/>
</window>
```

ПРИЛОЖЕНИЕ Б

Диаграмма классов для функции сайта просмотр списка клиентов

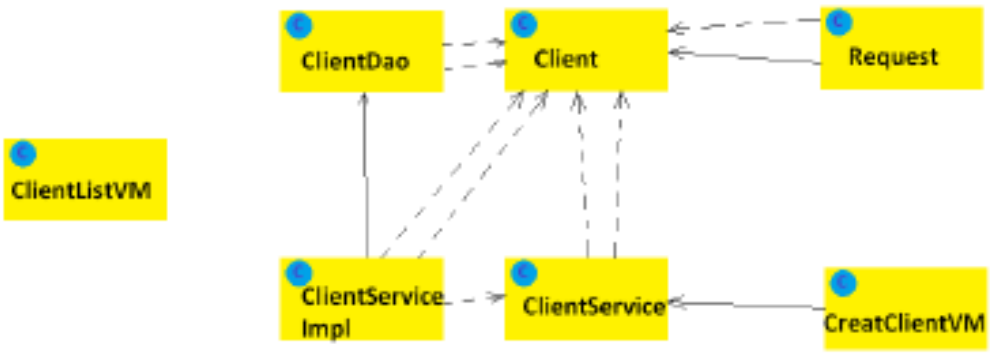

Рисунок Б.1 – Диаграмма классов для функции сайта просмотр списка клиентов