

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Средства и технологии анализа и разработки информационных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему
Разработка автоматизированной системы учёта выполненных заявок на
ремонт оргтехники

БГУИР КР 1-40 01 02-02 049 ПЗ

Студент гр. 814301
Руководитель

Пристром Р.М.
Федосенко В.А.

СОДЕРЖАНИЕ

<u>Введение</u>	5
<u>1. Анализ и моделирование предметной области программного средства</u>	7
<u>1.1. Описание предметной области</u>	7
<u>1.2. Разработка функциональной модели предметной области</u>	9
<u>1.3. Анализ требований к разрабатываемому программному средству</u>	10
<u>1.4. Спецификация функциональных требований</u>	12
<u>1.5. Разработка информационной модели предметной области</u>	13
<u>1.6. Модель представления программного средства и их описание</u>	15
<u>2. Проектирование и конструирование программного средства</u>	17
<u>2.1. Постановка задачи</u>	17
<u>2.2. Архитектурные решения</u>	17
<u>2.3. Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства</u>	19
<u>2.4. Проектирование пользовательского интерфейса</u>	23
<u>2.5. Обоснование выбора компонентов и технологий для реализации программного средства.</u>	28
<u>3. Тестирование и проверка работоспособности программного средства</u>	32
<u>4. Руководство по развертыванию и использованию программного средства</u>	34
<u>Заключение</u>	40

<u>Список использованных источников</u>	41
<u>ПРИЛОЖЕНИЕ А (обязательное) Функциональная модель системы учёта и регистрации успеваемости студентов (IDEF0)</u>	42
<u>ПРИЛОЖЕНИЕ Б (обязательное) Диаграммы на основе UML</u>	43
<u>ПРИЛОЖЕНИЕ В (обязательное) SQL скрипт базы данных</u>	44

ВВЕДЕНИЕ

Важнейшая задача компьютерных систем – хранение и обработка данных. Для её решения были предприняты усилия, которые привели к появлению в конце 60-х – начале 70-х годов специализированного программного обеспечения – систем управления базами данных (database management systems). СУБД позволяют структурировать, систематизировать и организовать данные для их компьютерного хранения и обработки. Невозможно представить себе деятельность современного предприятия или учреждения без использования профессиональных СУБД. Несомненно, они составляют фундамент информационной деятельности во всех сферах – начиная с производства и заканчивая финансами и телекоммуникациями.

Курсовой проект посвящен изучению теории и практики разработки и проектирования информационных систем с использованием баз данных. В данном курсовом проекте объектом исследования является среда WEB, как платформа приложений информационных систем.

В настоящее время многие процессы, связанные с хранением и обработкой информации, проводятся на бумаге, вместо того, чтобы хранить их в определённой системе. У такого метода учёта есть серьёзный минус: бумажные документы замедляют обработку и передачу информации.

Ремонт (техники) — комплекс мероприятий по восстановлению работоспособного или исправного состояния какого-либо объекта и/или восстановлению его ресурса.

Ремонт производится в случае, если невозможно или нецелесообразно заменить изделия на аналогичные новые. Нередко изделия устаревают морально гораздо раньше, чем вырабатывается их ресурс до ремонта, или

затраты на производство изделий в неремонтируемом исполнении существенно меньше — в этих случаях производители стараются переходить на выпуск изделий в неремонтируемом исполнении. Характерные примеры — однокристальные ЭВМ, устройства в неразборном исполнении и т. д.

Предметом исследования является методы разработки автоматизированных систем. Целью курсового проекта является разработка автоматизированной системы заказа ремонта оргтехники.

Курсовой проект предполагает выполнение следующих задач:

- проанализировать и описать предметную область;
- проанализировать логическую и физическую модель представления данных;
- спроектировать и сконструировать программное средство работы с заказами;
- создать базу данных, содержащей данные о пользователях, заказах, услугах, товара, который будет ремонтироваться, номерах телефона;
- проверить работоспособность полученного приложения;
- описать руководство по развертыванию и использованию программного средства
- реализовать возможности авторизации;
- разделить доступа к управлению программой на пользовательский, рабочий и администраторский с соответствующими программными возможностями;
- разработать удобный и интуитивно понятный пользователю интерфейс;
- сделать вывод о необходимых доработках.

В системе проверки на уникальность «Антиплагиат» работа показала результат 72.25%.

1. АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА

Бизнес-процесс – процесс перевода исходных ресурсов в конечный

продукт под влиянием управленческих ресурсов.

Моделирование бизнес-процессов является основой и начальной точкой

конструирования программного средства. Оно служит повышению качества, эффективности разработки, и описывает взаимосвязи между элементами процесса, модулями.

Моделирование бизнес-процессов позволяет понять и определить модель, которая уже существует в данной сфере, и модель, которая должна быть. Таким образом, опираясь на два понятия «as is» и «to be», можно составить план разработки и создать программное средство, полностью удовлетворяющее запросам пользователя.

1.1. Описание предметной области

Целью курсового проекта является создание продукта, простого и удобного в использовании, позволяющего автоматизировать работу по заказу ремонта оргтехники.

Ремонт (техники) — комплекс мероприятий по восстановлению работоспособного или исправного состояния какого-либо объекта и/или восстановлению его ресурса

Ремонт производится в случае, если невозможно или нецелесообразно заменить изделия на аналогичные новые. Нередко изделия устаревают морально гораздо раньше, чем вырабатывается их ресурс до ремонта, или затраты на производство изделий в неремонтируемом исполнении существенно меньше — в этих случаях производители стараются переходить на выпуск изделий в неремонтируемом исполнении. Характерные примеры — однокристальные ЭВМ, устройства в неразборном исполнении и т. д.

Разновидности ремонта (техники):

- Косметический — восстановление внешнего вида без вмешательства в конструкцию (бытовое название текущего ремонта).
- Средний (Восстановительный) — обычно производится с заменой частей устройства, подвергшихся износу, либо с их модификацией (наплавка, расточка, пайка и т. д.)
- Текущий (Малый) — ремонт с целью восстановления исправности (работоспособности), а также поддержания эксплуатационных показателей.

Малый ремонт был упразднен в 80-х годах и приведен в соответствие к текущему ремонту.

- Капитальный — предполагает разборку и ревизию конструкции с целью выявления скрытых неисправностей и оценки ресурса деталей, замену не только неисправных деталей, но и деталей, выработавших свой ресурс. Такой ремонт предполагает большой объём работ и значительные расходы.

- Плановый (планово-предупредительный) — ремонт в запланированный регламентом промежутков времени. Производится после выработки устройством ресурса, либо в случае, если работоспособность устройства после неисправности частично сохраняется, или частично восстанавливается в результате восстановительного ремонта. Позволяет заранее уведомить пользователей о прекращении функционирования, а также спланировать издержки, связанные с простоем оборудования.

Что относится к оргтехнике. Организационная техника, а сокращенно «оргтехника» — это выражение, которое именует все современные технические средства и аппараты, используемые в офисах малого бизнеса или больших промышленных предприятий. К оргтехнике сейчас зачисляют канцелярские мелочи, столы, шкафы, стулья и другую мебель. Вы можете удивиться, но сюда же причисляют даже предметы быта: чайники, микроволновку, кофеварку и другие приборы. Также легко и смело можно добавлять все, что облегчит выполнение сотрудником своих обязанностей на работе.

Еще несколько десятков лет тому в офисе сложно было встретить так много разнообразной техники. Стандартное оснащение включало в себя печатную машинку, калькулятор или деревянные счета и, скорее всего, на этом перечень заканчивался. Современный вариант рабочего пространства кардинально отличается от того времени.

Чтобы сделать копию текста документа нужно было позаботиться об этом заранее. Использовалась специальная копировальная бумага. Ее подкладывали под оригинал и во время его создания одновременно «производилась» копия за счет перебивания краски с копировальной бумаги из-за нажатия ручки.

ПК. Компьютер — это специальный электронно-цифровой аппарат, который способен хранить и обрабатывать разного рода и сложности информацию и данные. Но также он может быть применен для: ввода

информации, создания текстовых документов, рисунков и картинок, таблиц, презентаций, работы с электронной почтой и видеоматериалами.

ПК — это не только тот персональный компьютер с монитором или ноутбук. Когда Вы пользуетесь банкоматом или терминалом самообслуживания, к примеру, то знайте, что здесь также встроен «комп». Просто, в силу потребности, его не видно.

А также компьютеры могут быть разных форматов и предназначения, но основа у них одна и та же. Рассмотрим каждый из возможных вариантов более детально.

Ноутбуки. Еще их называют «портативными», но это название не прижилось так сильно. Могут работать от сети или от батареи, которую предварительно нужно зарядить и проводить такую процедуру регулярно. Ноутбук ничем не отличается от обычного ПК внутренней «начинкой» и возможностями работы с данными.

Планшеты. Еще один подвид портативных устройств. Их размеры меньше, чем у ноутбука, а физической клавиатуры нет вовсе. Вместо этого используется ее виртуальный аналог. В большинстве случаев внутри установлена мобильная операционная система «Андроид» или «Виндовс Мобайл». Ввод осуществляется с использованием сенсорного экрана.

Серверы. Сервер — это подтип компьютера, но намного мощнее обычного «брата». У него больше частота процессора или количество его ядер, оперативная память, объем винчестера и другие характеристики более высокого плана. Сервер может использоваться для хранения, обработки информации, управления другими рабочими станциями или печатающей техникой.

Принтеры. Чтобы распечатать любой, ранее набранный на «компе» текстовых или графический документ, понадобится принтер. Печатающие устройства бывают нескольких видов. Но по типу используемой печати разделяют лазерные и струйные варианты. Первые используют тонер-порошок, а вторые — красители или чернила. Принцип нанесения изображения на бумагу также отличается кардинально.

1.2. Разработка функциональной модели предметной области

Отобразим функциональную модель предметной области с использованием методологии SADT. SADT (Structured Analysis and Design Technique) представляет собой методологию анализа и проектирования систем.

Данную методологию целесообразно применять на ранних этапах жизненного цикла, для того чтобы можно было рассмотреть систему до ее воплощения. Методология SADT дает возможность сократить дорогостоящие ошибки на ранних этапах разработки системы, улучшить контакт между пользователями и разработчиками, сгладить переход от анализа к проектированию. Методология SADT – нотация IDEF0.

IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

Стандарт IDEF0 представляет организацию как набор модулей, здесь существует правило – наиболее важная функция находится в верхнем левом углу, кроме того, существуют правила сторон:

- стрелка входа всегда приходит в левую кромку активности;
- стрелка управления – в верхнюю кромку;
- стрелка механизма – нижняя кромка;
- стрелка выхода – правая кромка.

Описание выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того, чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

Входными параметрами нашего процесса будут:

- данные пользователя.

К «управлению» будут относиться следующие элементы:

- закон о защите прав потребителей;
- правила использования приложения.

К «механизму» будет относиться база данных, программное обеспечение.

На выходе будет результат выполнения процесса – сохранённый или отменённый заказ.

Разработанная модель представлена в Приложении А (Рисунок А.1).

Декомпозиция модели состоит из трёх процессов: авторизация, заполнение полей и сохранение заказа. Приложение А рисунок А.2.

1.3. Анализ требований к разрабатываемому программному средству

Требования— это первое, на что смотрит команда проекта, это фундамент для проектирования и разработки продукта. Допущенная в документации ошибка или неточность может проявиться в самый неподходящий момент. Тестирование требований направлено на то, чтобы уже на начальных этапах проектирования системы устранить максимально возможное количество ошибок.

Анализ требований – это процесс определения ожиданий пользователей в отношении нового или модифицированного продукта, который жизненно важен для эффективного тестирования программного обеспечения QA. Без тщательного анализа потребностей и четкой дорожной карты даже самая эффективная системная архитектура, хорошо написанный код или всестороннее тестирование окажутся бесполезными. Следовательно, первым шагом к запуску любого проекта является «Анализ требований». И чтобы быть полезными, требования должны быть составлены из сценариев, иметь возможность действовать, измерить, отслеживать и напрямую связывать с бизнес-потребностями или возможностями. Типичный процесс анализа требований должен определять проверяемые требования посредством частого взаимодействия с различными заинтересованными сторонами (клиент, бизнес-аналитик, технические руководители и т. д.). Цель состоит в том, чтобы провести мозговой штурм и прояснить любые неоднозначные требования, прежде чем перейти к следующему этапу, и точно определить потребности автоматизации для проекта.

Цель анализа — достаточно качественно и подробно описать требования, позволяющие менеджерам реалистично оценить все затраты на

проект, а техническому персоналу — начать проектирование, разработку и тестирование.

Аналитик совместно с разработчиками должен определить, насколько возможно реализовать каждое требование при разумных затратах и с приемлемой производительностью в предполагаемой среде. Это позволяет заинтересованным лицам понять риски, связанные с реализацией каждого требования, включая конфликты с другими требованиями, зависимость от внешних факторов и препятствия технического характера. Технически нереализуемые или очень дорогие в реализации требования можно попытаться упростить и в таком виде включить в проект для достижения бизнес-целей.

Полученные общие требования на курсовой проект. Программное средство следует разработать в архитектуре web-приложение с базой данных с использованием объектно-ориентированного языка программирования Java, современных технологий и фреймворков. В рамках работы должны быть представлены: разработка и использование собственной иерархии классов, реализация не менее 2-х паттернов проектирования, сокрытие данных (инкапсуляция), перегрузка методов, переопределение методов, параметризованные классы (шаблоны), абстрактные типы данных (интерфейсы, абстрактные классы), передача параметров по ссылке и по значению, статические методы, обработка исключительных ситуаций.

Уровни архитектуры: приложение должно быть распределено по 2-м отдельным серверам:

Сервер СУБД – СУБД для размещения *базы данных* курсового проекта выбирается студентом самостоятельно.

Сервер Приложений – используется для размещения “серверной” части приложения, представленной *Моделью* на основе ORM технологии (Hibernate/JPA), *Бизнес-логикой* приложения на основе технологий jsp+servlet или иного MVC фреймворка для разработки веб-приложений.

В качестве языка программирования будет использован язык java в связке с SpringFramework.

1.4. Спецификация функциональных требований

Функциональное тестирование (Functional Testing) – тестирование, основанное на сравнительном анализе спецификации и функциональности компонента или системы.

Исходя из критериев для данного типа программ, а также специфики предметной области, выделим основные функциональные требования к системе с точки зрения пользователей:

- возможность создание нового заказа;
- возможность создание новой услуги;
- возможность поиска нужной услуги;
- возможность подробного просмотра заказа;
- возможность переходить по страницам для поиска нужной услуги;
- возможность просмотра статуса ремонта.

Кроме пользователей программное средство должно быть доступно пользователям-работником.

Пользователям-работника доступны:

- возможность просмотра заказа по статусам;
- принять заказ на выполнение;
- подтвердить выполнение работы;
- возможность добавления заказа.

Кроме пользователей программное средство должно быть доступно пользователям-администраторам.

Пользователям-администраторам доступны:

- возможность просмотра заказа по статусам;
- возможность поиска заказа;
- возможность добавления услуги и заказа.

1.5. Разработка информационной модели предметной области

Построение информационной модели предметной области предполагает выделение сущностей, их атрибутов и первичных ключей, идентификацию связей между сущностями. Общепринятым видом графического изображения реляционной модели данных является ER-диаграмма, на которой сущности изображаются прямоугольниками, соединенные между собой связями. Такое графическое представление облегчает восприятие структуры базы данных по сравнению с текстовым описанием.

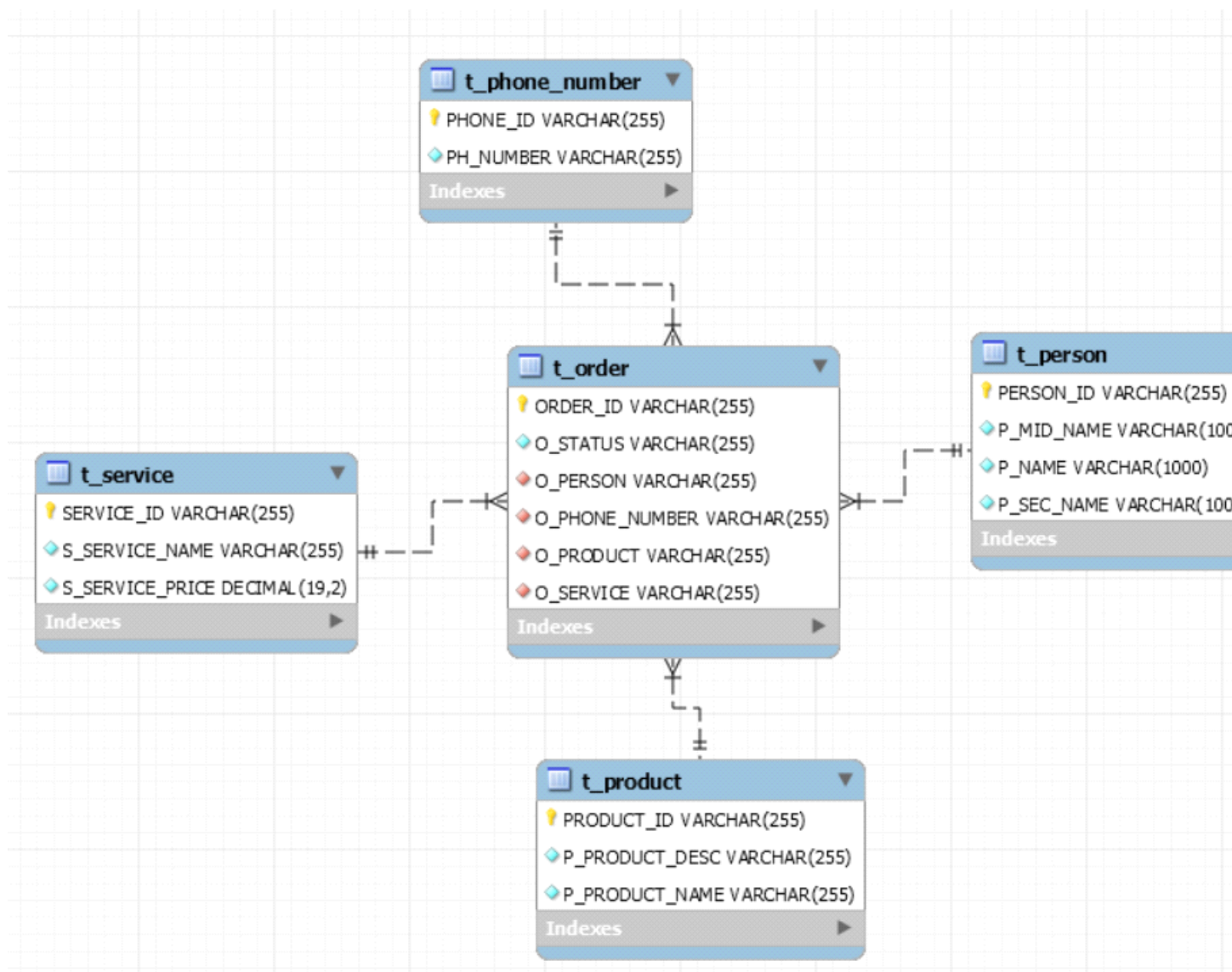


Рисунок 1.1 – Модель базы данных

В базе данных находятся восемь таблиц. Рассмотрим их более подробно.

Таблица **t_product** предназначена для хранения данных о продукте для ремонта. Рассмотрим все поля таблицы **t_product**:

- **PRODUCT_ID** – хранит идентификатор записи, для каждой строки он уникальный;
- **P_PRODUCT_DESC**– хранит поломку техники;

- U_PRODUCT_NAME – хранит наименование техники.

Таблица t_phone_number предназначена для хранения данных о номерах телефонов. Рассмотрим все поля таблицы t_phone_number:

- PHONE_ID – хранит идентификатор записи, для каждой строки он уникальный;
- PH_NUMBER – хранит номер телефона.

Таблица t_order предназначена для хранения данных о заказах. Рассмотрим все поля таблицы t_order:

- ORDER_ID – хранит идентификатор записи;
- O_STATUS – хранит статус заказа;
- O_PERSON (FK) – служит для связи заказчика и заказа. Хранит идентификатор заказчика;
- O_PHONE_NUMBER (FK) – служит для связи номера телефона и заказа. Хранит идентификатор номера телефона;
- O_PRODUCT (FK) – служит для связи продукта для ремонта и заказа. Хранит идентификатор продукта для ремонта;
- O_SERVICE (FK) – служит для связи услуги и заказа. Хранит идентификатор услуги.

Таблица t_service предназначена для хранения данных об услугах. Рассмотрим все поля таблицы t_service:

- SERVICE_ID – хранит идентификатор записи, для каждой строки он уникальный;
- S_SERVICE_NAME – хранит наименование услуги;
- S_SERVICE_PRICE – хранит цену услуги.

Таблица t_person предназначена для хранения данных о заказчиках. Рассмотрим все поля таблицы t_person:

- PERSON_ID – хранит идентификатор записи, для каждой строки он уникальный;
- P_MID_NAME – хранит отчество заказчика;
- P_NAME – хранит имя заказчика;
- P_SEC_NAME – хранит фамилию заказчика.

Идентифицирующая связь – экземпляр сущности-потомка однозначно определяется своей связью (отношением) с сущностью-родителем. В таком случае связь каждый экземпляр подчиненной сущности идентифицируется значениями атрибутов родительской сущности. Это означает, что экземпляр

подчиненной сущности зависит от родительской сущности и не может существовать без экземпляра родительской сущности. В идентифицирующем отношении единственный экземпляр родительской сущности связан с множеством экземпляров подчиненной.

Неидентифицирующая связь – экземпляр сущности-потомка определяется своей связью с сущностью-родителем неоднозначно, т.е. экземпляр дочерней сущности идентифицируется иначе, чем через связь с родительской сущностью, и не зависит от значений атрибутов родительской сущности. Это означает, что экземпляр подчиненной сущности не зависит от родительской сущности и может существовать без экземпляра родительской сущности. В неидентифицирующем отношении единственный экземпляр родительской сущности связан с множеством экземпляров подчиненной.

В данной базе данных используются неидентифицирующие связи типа 1:1 и 1:M. Так, например, между таблицами `t_document` и `t_author` установлена неидентифицирующая связи 1:1, т.е. идентификатор ФИО адреса берётся из таблицы `t_author`, из свойства `A_AUTHOR_ID`, и не может быть равен несуществующей форме.

Докажем, что данная база данных находится в третьей нормальной форме. Данная база данных находится в 1-й нормальной форме, так как в любом допустимом значении отношения каждый кортеж содержит только одно значение для каждого из атрибутов. Также база данных находится и во второй нормальной форме, так как находится в первой нормальной форме, и каждый неключевой атрибут функционально зависит от ее потенциального ключа. Так как база данных находится во второй нормальной форме и у неё отсутствуют функциональные зависимости неключевых атрибутов от ключевых, то она находится в третьей нормальной форме.

1.6. Модели представления программного средства и их описание

Языки визуального моделирования – это формализованные наборы графических символов и правила построения из них визуальных моделей. Сейчас известны и активно используются на практике такие языки *визуального моделирования*, как UML и BPMN.

UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования.

Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

UML предполагает разработку нескольких моделей, совокупность которых описывает разрабатываемую систему. Каждая модель относится к соответствующей фазе и имеет собственное предназначение. При этом каждая из моделей состоит из одной или нескольких UML-диаграмм, которые можно классифицировать следующим образом:

- структурные (structure) диаграммы;
- диаграммы поведения (behaviour);
- диаграммы взаимодействия (interaction).

Стандарт UML версии содержит следующий набор диаграмм:

- диаграммы классов (class diagrams) - для моделирования статической структуры классов системы и связей между ними;
- диаграммы компонентов (component diagrams) - для моделирования иерархии компонентов (подсистем) системы;
- диаграммы размещения (deployment diagrams) - для моделирования физической архитектуры системы
- диаграммы вариантов использования (use case diagrams) - для моделирования функциональных требований к системе (в виде сценариев взаимодействия пользователей с системой);
- диаграммы взаимодействия (interaction diagrams);
- диаграммы последовательности (sequence diagrams) и кооперативные диаграммы (collaboration diagrams) - для моделирования процесса обмена сообщениями между объектами;
- диаграммы состояний (state chart diagrams) - для моделирования поведения объектов системы при переходе из одного состояния в другое;
- диаграммы деятельности (activity diagrams) - для моделирования поведения системы в рамках различных вариантов использования, или потоков управления.

2. ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1. Постановка задачи

Для достижения цели данного курсового проекта, следует выделить следующие задачи, которые необходимо решить:

- выбрать архитектуру будущего приложения;
- создать базу данных по информационной модели предыдущего раздела;
- описать алгоритмы, реализующие бизнес-логику приложения;
- спроектировать пользовательский интерфейс;
- провести тестирование полученного приложения;
- описать руководство по развертыванию и использованию программного средства.

Кроме того, разрабатываемое приложение должно следующим требованиям:

- проектируемая база данных должна отвечать требованиям надёжности, минимальной избыточности, целостности данных;
- содержать не менее 5 сущностей и должна быть приведена к третьей нормальной форме;
- реализовать приложение необходимо при помощи языка программирования Java и Базы данных SQL.

Серверное приложение будет обрабатывать запросы на:

- авторизацию пользователя;
- получение данных о заказах или заказе;

- получение списка услуг;
- получение списка найденных заказов с пагинацией;
- добавление новых заказов;
- обновление данных.

Список функционала приложения будет зависеть от роли вошедшего пользователя: пользователь, работник или администратор.

2.2. Архитектурные решения

Поскольку разработка ведется на основе фреймворка спринг, разумным будет использовать архитектуру Spring MVC.

Фреймворк **Spring MVC** обеспечивает архитектуру паттерна Model – View – Controller (Модель – Отображение (далее – Вид) – Контроллер) при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними [5].

Model (Модель) инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO («Старых добрых Java-объектов», или бинов).

View (Отображение, Вид) отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.

Controller (Контроллер) обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.

Ниже приведена последовательность событий, соответствующая входящему HTTP-запросу:

- После получения HTTP-запроса **DispatcherServlet** обращается к интерфейсу **HandlerMapping**, который определяет, какой Контроллер должен быть вызван, после чего, отправляет запрос в нужный Контроллер.
- Контроллер принимает запрос и вызывает соответствующий служебный метод, основанный на GET или POST. Вызванный метод определяет данные Модели, основанные на определённой бизнес-логике и возвращает в **DispatcherServlet** имя Вида (View).
- При помощи интерфейса **ViewResolver** **DispatcherServlet** определяет, какой Вид нужно использовать на основании полученного имени.

- После того, как Вид (View) создан, DispatcherServlet отправляет данные Модели в виде атрибутов в Вид, который в конечном итоге отображается в браузере.

Все вышеупомянутые компоненты, а именно, **HandlerMapping**, **Controller** и **ViewResolver**, являются частями интерфейса **WebApplicationContext** extends **ApplicationContext**, с некоторыми дополнительными особенностями, необходимыми для создания web-приложений.

DispatcherServlet отправляет запрос контроллерам для выполнения определённых функций. Аннотация `@Controller` указывает, что конкретный класс является контроллером. Аннотация `@RequestMapping` используется для мапинга (связывания) с URL для всего класса или для конкретного метода обработчика.

```
@Controller
public class HomeController {

    @Autowired
    ServiceForServices serviceForServices;

    @GetMapping("/")
    public String home(
        @RequestParam(value = "pageNumber", defaultValue
= "1") String pageNumber,
        Model model) {

        int countDocumentInPage = 5;

        model.addAttribute("serviceList",
serviceForServices.getAllServices(pageNumber,
countDocumentInPage));

        model.addAttribute("pageCount",
serviceForServices.getNumberOfPage(countDocumentInPage));

        return "index";
    }
}
```

Аннотация Controller определяет класс как Контроллер Spring MVC. В первом случае, @GetMapping указывает, что вывод начальной страницы будет производиться методом get в Контроллере и относится к URL-адресу "/".

2.3. Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства

Опишем некоторые алгоритмы работы, использующихся в реализации бизнес-логики.

Взаимодействие с БД.

Для взаимодействия с базой данных, необходимо следовать следующему алгоритму:

- Создать сущность. Сущность представляет собой класс с названием, которому будет соответствовать название таблицы в базе данных и полями, соответствующими полями из базы данных
- Для созданной сущности создать репозиторий. Репозиторий – это интерфейс для спринга, в котором мы описываем свои методы (если это необходимо) для работы с данными в базе данных текущей сущности.
- Используя репозиторий создать класс сервиса сущности. Класс сервиса – это связующий класс, который будет получать данные из базы данных и предоставлять его контроллерам и наоборот.

Пример, как это выглядит, отображен в листинге 1.

Листинг 1. Пример классов для взаимодействия с БД.

```
@Data
@Entity
@Table(name = "T_PRODUCT")
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Product {

    @GeneratedValue(generator = "uuid-generator")
    @GenericGenerator(name = "uuid-generator", strategy =
"uuid")
    @Id
    @Column(name = "PRODUCT_ID")
```

```

private String productId;

@Column(name = "P_PRODUCT_NAME")
private String productName;

@Column(name = "P_PRODUCT_DESC")
private String description;

}

@Repository
public interface DocumentDao extends JpaRepository<Document,
String> {

    List<Document> findAllBy(Pageable pageable);

    Document findByDocumentId(String id);

    @Query("select d from Document d " +
        "join d.content c " +
        "join d.documentName doc " +
        "join d.author a " +
        "join d.personWhoConcludedContract pw " +
        "join d.personWithWhomTheContractWasSigned pwh " +
        "where doc.documentName like ??1% or " +
        "c.content like ??1% or " +
        "a.author like ??1% or " +
        "pw.personWhoConcludedContract like ??1% or " +
        "pwh.personWithWhomTheContractWasSigned like ??1%")
    List<Document> searchDocument(String searchParam, Pageable
pageable);

    @Query("select count(d) from Document d " +
        "join d.content c " +
        "join d.documentName doc " +
        "join d.author a " +
        "join d.personWhoConcludedContract pw " +
        "join d.personWithWhomTheContractWasSigned pwh " +
        "where doc.documentName like ??1% or " +
        "c.content like ??1% or " +
        "a.author like ??1% or " +
        "pw.personWhoConcludedContract like ??1% or " +

```

```

        "pwh.personWithWhomTheContractWasSigned like %?1%")
    int countSearchResults(String searchParam);

    @Query("select d.author from Document d ")
    List<String> getName();
}
@Service
@Transactional
public class ServiceForServices {

    @Autowired
    ServicesDao servicesDao;

    @Autowired
    OrderDao orderDao;

    public List<TypeServices> getAllServices(String pageNumber,
    int countDocumentInPage) {

        Pageable pageable = PageRequest.of(
            Integer.parseInt(pageNumber) - 1,
            countDocumentInPage,
            Sort.by("serviceName")
        );

        return servicesDao.findAllBy(pageable);
    }

    public List<TypeServices> getServices() {

        return servicesDao.findAllBy();
    }

    public void saveService(TypeServices services) {

        servicesDao.save(services);
    }

    public double getNumberOfPage(int countDocumentInPage) {

        long allDocumentCount = servicesDao.count();

```

```

        if (allDocumentCount % countDocumentInPage == 0) {
            return Math.floor(allDocumentCount /
countDocumentInPage);
        } else {
            return Math.floor(allDocumentCount /
countDocumentInPage) + 1;
        }
    }
}
}

```

Взаимодействие пользователя с данными

Для реализации взаимодействия используются контроллеры, со следующим алгоритмом

- Ждем действие пользователя. В данном случае, действием называется переход пользователя на любую страницу сервиса или отправка запроса.
- Получаем адрес, на который пользователь посылает запрос. К примеру, это будет адрес «/» или «/order/add».
- В зависимости от адреса, вызываем метод контроллера.
- Возвращаем результат метода. Результатом любого метода является страница или пересылка на страницу. Также, в теле метода могут быть прикреплены данные для страницы.

Затем алгоритм повторяется для каждого запроса.

Ниже отобразим пример контроллера HomeController.

Листинг 2. Пример метода контроллера

```

@Controller
public class HomeController {

    @Autowired
    ServiceForServices serviceForServices;

    @GetMapping("/")
    public String home(

```



```

        @RequestParam(value = "pageNumber", defaultValue
= "1") String pageNumber,
        Model model) {

        int countDocumentInPage = 5;

        model.addAttribute("serviceList",
serviceForServices.getAllServices(pageNumber,
countDocumentInPage));

        model.addAttribute("pageCount",
serviceForServices.getNumberOfPage(countDocumentInPage));

        return "index";
    }
}

```

2.4. Проектирование пользовательского интерфейса

Проектирование пользовательского интерфейса – это создание тестовой версии приложения. Это начальный этап разработки пользовательского интерфейса, когда распределяются функции приложения по экранам, определяются макеты экранов, содержимое, элементы управления и их поведение.

Интерфейс будет максимально простым, чтобы не отвлекать пользователя от цели использования системы и не тратить его время на дополнительное обучение использования данного ресурса.

При входе пользователь будет видеть начальную страницу с предоставляемыми услугами. (Рисунок 2.2).

S	
Услуга	Цена
Обновление ПО	от 15.00
Ремонт принтера	от 12.00
ывапро	от 1.00
1	

Рисунок 2.2 – Стартовая страница

Все данные будут отображаться в виде таблиц для удобства пользователя. Так, при авторизации будет отображена главная страница пользователя (Рисунок 2.3).

S	Просмотр всех услуг	Добавить заказ	Статус заказов	Logout	Search	П
Услуга			Цена			
Обновление ПО			от 15.00			
Ремонт принтера			от 12.00			
ывапро			от 1.00			
						1

Рисунок 2.3 – Главная страница пользователя

При переходе на главную страницу пользователь может создавать новые заказы, просматривать все заказы, просматривать все услуги и искать необходимую услугу.

Поиск услуги (Рисунок 2.4).

S	Просмотр всех услуг	Добавить заказ	Статус заказов	Logout	Ремонт	П
Услуга			Цена			
Ремонт принтера			от 12.00			
						1

Рисунок 2.4 – Результат поиска

Форма с добавлением. (Рисунок 2.5)

Оформить заказ:

Продукт	что ремонтировать
Проблема	Проблема
Имя	Имя
Фамилия	Фамилия
Отчество	Отчество
Телефон	Телефон
Услуга	Ремонт принтера ▾
<input type="button" value="Submit"/>	

Рисунок 2.5 –Добавление заказа

При заполнении всех полей и нажатии кнопки «Сохранить» пользователь переходит на главную страницу и обновляется таблица с заказами (Рисунок 2.6, 2.7).

Оформить заказ:

Продукт

Принтер

Проблема

Сломан

Имя

Родион

Фамилия

Пристром

Отчество

Юрьевич

Телефон

+375296857421

Услуга

Ремонт принтера



Submit

Рисунок 2.6 –Добавление заказа

S	Просмотр всех услуг	Добавить заказ	Статус заказов
Заказ: 402898f4794c187601794d1e53460004			
Устройство			
Принтер			
Проблема			
Сломан			
ФИО			
Родион			
Пристром			
Юрьевич			
Телефон			
+375296857421			

Рисунок 2.7 – Просмотр сохранённого заказа

Поиск по заказам у работника и администратора производится с помощью поисковой строки в правом верхнем углу (Рисунок 2.8). Введём в поисковую строку слог «Ремонт».

Результат работы представлен на Рисунке 2.9.

Search

Поиск

Рисунок 2.8 – Поисковая строка

S Добавить услугу Просмотр заказов Добавить заказ Просмотр услуг Просмотр выполненных работ					Logout	<input type="text" value="Пристром"/>	<input type="button" value="Поиск"/>
Результат поиска заказа:							
Номер	Услуга	ФИО	Телефон	Статус			
402898f4794c187601794d1e53460004	Ремонт принтера	Пристром Родион Юрьевич	+375296857421	ADOPTED			
					<div>1</div>		

Рисунок 2.9 – Результат поиска

Администратор может добавлять услуги (Рисунок 2.10).

[S](#) [Добавить услугу](#) [Просмотр заказов](#) [Добавить](#)

Добавить услугу:

Наименование	Наименование
Цена	Цена

Submit

Рисунок 2.10 – Страница добавления услуги

Результат добавления услуги «Замена стекла». (Рисунок 2.11)

Услуга	Цена
Замена стекла	от 50.00
Обновление ПО	от 15.00
Ремонт принтера	от 12.00

Рисунок 2.11 – Обновлённая таблица

Работник при нажатии на «Принять» подтверждает, что он взял заказ на выполнение (Рисунок 2.12).

Заказы:

Номер	Услуга	
1	Обновление ПО	Принять
402898f4794c187601794d1e53460004	Ремонт принтера	Принять

Рисунок 2.12 –Принятие заказа на выполнение

Работник при нажатии на «Работа выполнена» подтверждает, что он закончил выполнение заказа (Рисунок 2.13).

Заказы в процессе выполнения:

Номер	Услуга	
2	ывапро	Работа выполнена
402898f4794c187601794c195ecf0000	Ремонт принтера	Работа выполнена

Рисунок 2.13 – Таблица с заказа в процессе выполнения

Также работник и администратор могут просматривать выполненные заказы (Рисунок 2.14).

Выполненные заказы:

Номер	Услуга	ФИО	Телефон
3	Ремонт принтера	Иванов Иван Иванович	+375333125487

Рисунок 2.14 – Таблица выполненных заказов

Администратор может просмотреть все заказы со статусами (Рисунок 2.15)

Результат поиска заказа:

Номер	Услуга	ФИО	Телефон	Статус
1	Обновление ПО	Иванова Ирина Иванова	+375298547548	ADOPTED
2	ывапро	Иванов Иван Иванович	+375298547548	PROGRESS
3	Ремонт принтера	Иванов Иван Иванович	+375333125487	PERFORMED
402898f4794c187601794c195ecf0000	Ремонт принтера	пар авпр пар	345	PROGRESS
402898f4794c187601794d1e53460004	Ремонт принтера	Пристром Родион Юрьевич	+375296857421	ADOPTED

1

Рисунок 2.15 – Таблица со всеми заказами

2.5. Обоснование выбора компонентов и технологий для реализации программного средства.

Архитектура данного проекта – это веб-приложение. Архитектура приложения во многом предопределило и технологии, используемые для построения проекта.

Основные технологии, применяемые при создании приложения:

- серверный язык *JAVA*;
- язык запросов *SQL (MySQL)*;
- фреймворк *Spring*;
- язык разметки *HTML*;
- таблица стилей *CSS*;
- клиентский язык *JavaScript*.

Каждая из выбранных технологий отвечает за разные аспекты работы программы.

Язык Java появился в 1995 году – 90-е годы были вообще урожайными на новые языки и концепции программирования. В таком Эдеме языков важно было не заблудиться, по ошибке приняв за Священный Грааль технологию, которая не пройдет испытания временем. Java прошел испытания, хотя и очень долгие. Очень не рекомендуется путать этот язык с JavaScript – они по виду похожи, но это совсем разные языки.

Вероятно, в Java впервые реализовали концепцию того, что язык должен быть максимально изолирован от платформы разработки, чтобы применять его без изменений везде: в компьютерах, часах, сотовых

телефонах, бытовой технике. С «железной частью» должна была справляться виртуальная машина (JVM), которая, собственно, и создавалась индивидуально под каждое устройство. Сам же язык был неизменен и в качестве результата выдавал байт-код. С самого начала было известно, что код не может исполняться очень быстро, но многие устройства не требовали высокой скорости исполнения. Кроме того, со временем появились оптимизирующие компиляторы, так что, в среднем, программа на Java работает раза в 2-3 медленнее, чем на C++. Постоянное сравнение с C/C++ здесь не случайно: многие современные языки взяли за основу его конструкции и синтаксис, так что, бывает, узнать сходу язык очень трудно. Вместе с тем, Java с тех пор сильно «размножилась», и даже J#, J и прочие аналоги являются не родными братьями, а лишь подобием.

SQL (Structured Query Language, язык структурированных запросов) — это специальный язык, используемый для определения данных, доступа к данным и их обработки. Язык SQL относится к непроцедурным (nonprocedural) языкам — он лишь описывает нужные компоненты (например, таблицы) и желаемые результаты, не указывая, как именно эти результаты должны быть получены. Каждая реализация SQL является надстройкой над процессором базы данных (database engine), который интерпретирует операторы SQL и определяет порядок обращения к структурам БД для корректного и эффективного формирования желаемого результата.

Стандарт SQL определяется ANSI — American National Standards Institute (Американским Национальным Институтом Стандартов) и в настоящее время принят ISO — International Standards Organization (Международной Организацией по Стандартизации).

SQL — непроцедурный язык: серверу базы данных сообщается, что нужно сделать и каким образом. Для обработки запроса сервер базы данных транслирует команды SQL во внутренние процедуры. Благодаря тому, что SQL скрывает детали обработки данных, его легко использовать.

При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует форк для платформы .NET Framework, названный Spring.NET.

Первая версия была написана Родом Джонсоном, который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development» (Wrox Press, октябрь 2002 года).

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первая стабильная версия 1.0 была выпущена в марте 2004. Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 5.2.x.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring предоставляет бóльшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java. Языком JavaScript не владеет какая-либо

компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

3. ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Для проверки работы спроектированной автоматизированной системы, разработаны тесты, с помощью которых можно оценить корректную работу веб-сервиса. В таблицах 3.1-3.2 приведены результаты тестирования.

Таблица 3.1 – Набор тестов для автоматизированной системы

Место проведения теста	Содержание теста	Ожидаемый результат	Отметка о прохождении теста
Главная страница	Загрузка главной страницы	Вывод предоставляемых услуг	Да
Страница добавления заказа	Перейти на страницу. Ввести новый заказ. Нажать кнопку сохранить.	Переход на главную страницу	Да
Страница с поступившими и выполняемыми заказами.	Перейти на страницу работника с поступившими заказами. Нажать кнопку «Принять»	Удаление с этой странице заказами появление его на странице с заказами в обработке	Да
Страница добавления услуг.	Перейти на страницу добавление услуг. Ввести новую услугу. Нажать кнопку сохранить	Переход на страницу с услугами	Да
Страницы с заказами и услугами.	Перейти на страницу. Переходить по страницам	Переход по страницам без повторения услуг и заказов.	Да

Таблица 3.2 – Набор тестов отображения информации

Место проведения теста	Содержание теста	Ожидаемый результат	Отметка о прохождении теста
Все страницы	Перейти на любую из вкладок в произвольном порядке.	Корректное отображение информации, отображение того, чего ожидает пользователь.	Да
Все вкладки	Увеличение масштаба отображения средствами браузера.	Появление скроллов справа и снизу, корректное отображение информации.	Да
Все вкладки	Обновление страницы используя средства браузера.	Состояние веб-сервиса до обновления и после не изменилось.	Да
Меню браузера	Нажатие функциональных кнопок браузера «Назад» и «Вперед» в произвольный момент работы веб-приложения.	Корректное поведение веб-сервиса. Отображение информации.	Да

4. РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

Для установки веб-сервиса необходимым требованием является наличие Java Virtual Machine, Apache Tomcat 9, а также MySQL Server 8.0.1 в качестве целевой СУБД, maven 3.6.

База данных для веб-сервиса генерируется сама после первого запуска приложения, так как используется Hibernate.

Для запуска приложения на клиентском компьютере, необходимо:

- запустить tomcat на компьютере;
- консоли перейти в папку проекта и написать код *mvn clean install*;
- убедиться, что проект задеплоился на tomcat.

Если запуск решено производить посредством war-архива, то его нужно переместить или скопировать в папку, где располагается Apache Tomcat 9. Далее необходимо запустить Apache Tomcat 9.

Для запуска веб-сервиса необходимо открыть браузер на вашем компьютере и написать в адресную строку следующий URL-адрес: <http://localhost:8080/>.

После этого мы попадаем на страницу tomcat (Рисунок 4.1)

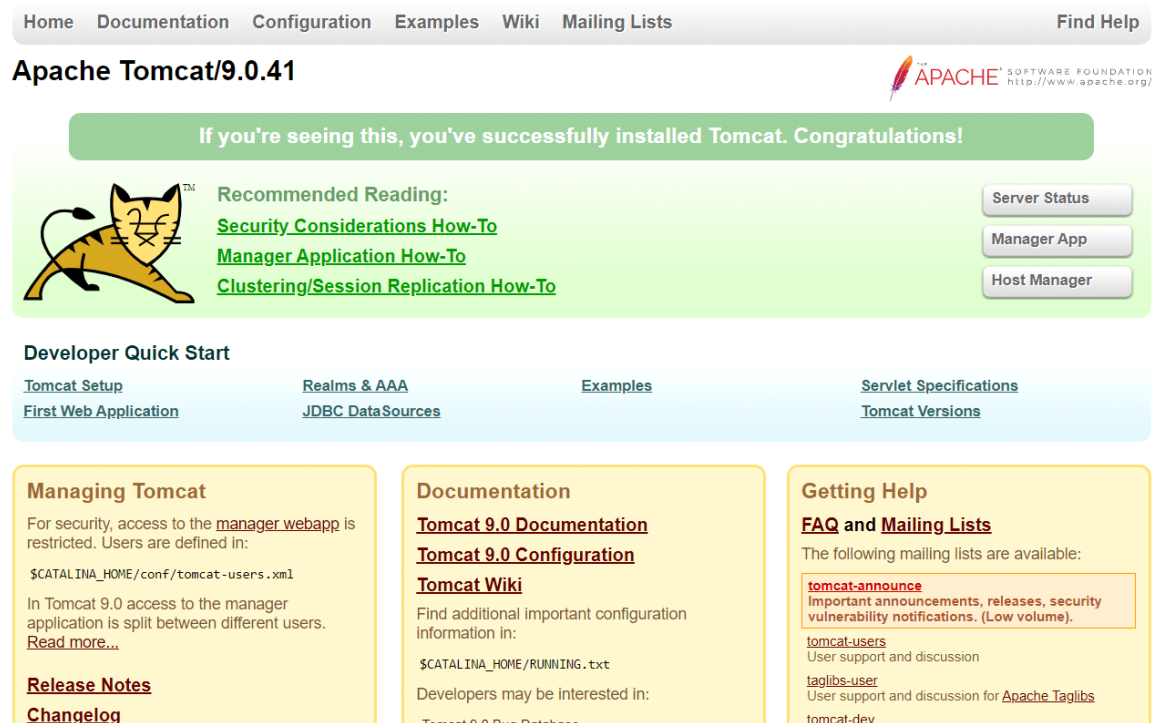


Рисунок 4.1 – Страница tomcat

Далее переходим на вкладку Manager App и в списке находит наш проект, нажимает на ссылку и проект открывается.

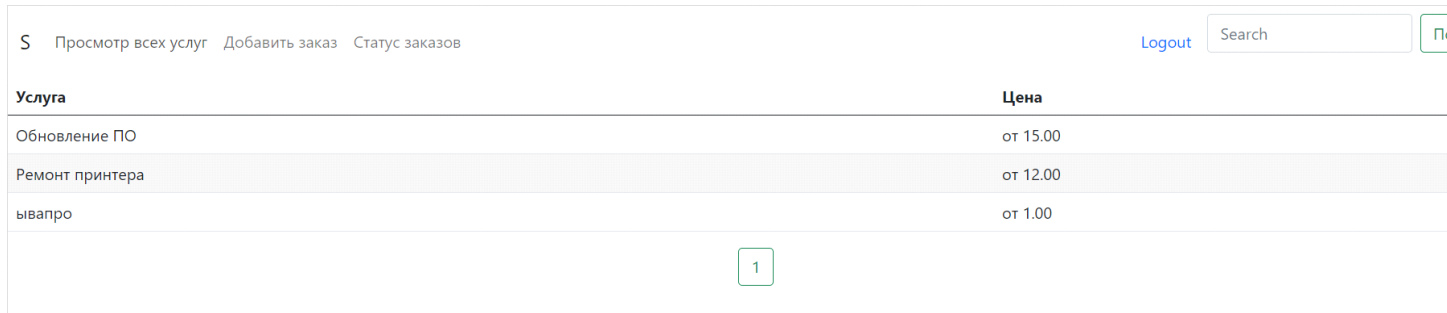
Далее опишем руководство по использованию программного средства.

При входе пользователь будет видеть начальную страницу с предоставляемыми услугами. (Рисунок 4.2).

S	
Услуга	Цена
Обновление ПО	от 15.00
Ремонт принтера	от 12.00
ывапро	от 1.00
1	

Рисунок 4.2 – Стартовая страница

Все данные будут отображаться в виде таблиц для удобства пользователя. Так, при авторизации будет отображена главная страница пользователя (Рисунок 4.3).



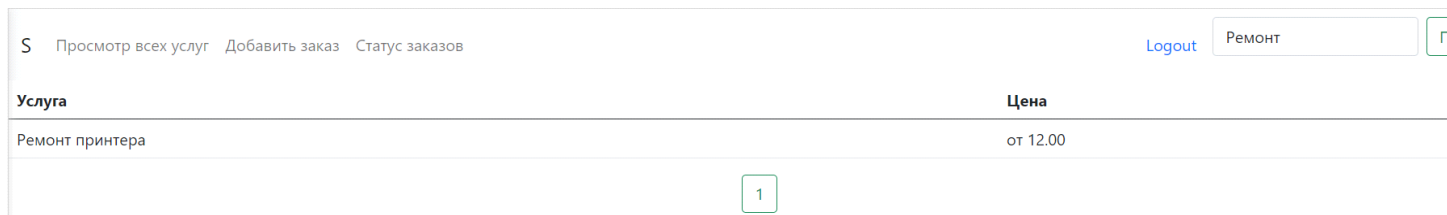
The screenshot shows a web interface for a service provider. At the top, there is a navigation bar with links: 'S', 'Просмотр всех услуг', 'Добавить заказ', and 'Статус заказов'. On the right side of the navigation bar, there is a 'Logout' link and a search input field with the placeholder text 'Search'. Below the navigation bar, there is a table with two columns: 'Услуга' (Service) and 'Цена' (Price). The table contains three rows of data: 'Обновление ПО' (Software update) with a price of 'от 15.00', 'Ремонт принтера' (Printer repair) with a price of 'от 12.00', and 'ывапро' (likely a typo for 'ывапро') with a price of 'от 1.00'. A green box with the number '1' is positioned below the table.

Услуга	Цена
Обновление ПО	от 15.00
Ремонт принтера	от 12.00
ывапро	от 1.00

Рисунок 4.3 – Главная страница пользователя

При переходе на главную страницу пользователь может создавать новые заказы, просматривать все заказы, просматривать все услуги и искать необходимую услугу.

Поиск услуги (Рисунок 4.4).



The screenshot shows the search results page. The navigation bar is identical to the previous screenshot. The search input field now contains the text 'Ремонт'. The table below shows the search results for 'Ремонт'. It has two columns: 'Услуга' (Service) and 'Цена' (Price). The table contains one row of data: 'Ремонт принтера' (Printer repair) with a price of 'от 12.00'. A green box with the number '1' is positioned below the table.

Услуга	Цена
Ремонт принтера	от 12.00

Рисунок 4.4 – Результат поиска

Форма с добавлением. (Рисунок 4.5)

Оформить заказ:

Продукт	что ремонтировать
Проблема	Проблема
Имя	Имя
Фамилия	Фамилия
Отчество	Отчество
Телефон	Телефон
Услуга	Ремонт принтера ▾
<input type="button" value="Submit"/>	

Рисунок 4.5 –Добавление заказа

При заполнении всех полей и нажатии кнопки «Сохранить» пользователь переходит на главную страницу и обновляет таблицу с заказами (Рисунок 4.6, 4.7).

Оформить заказ:

Продукт	Принтер
Проблема	Сломан
Имя	Родион
Фамилия	Пристром
Отчество	Юрьевич
Телефон	+375296857421
Услуга	Ремонт принтера ▾
<button>Submit</button>	

Рисунок 4.6 –Добавление заказа

S	Просмотр всех услуг	Добавить заказ	Статус заказов
Заказ: 402898f4794c187601794d1e53460004			
Устройство			
Принтер			
Проблема			
Сломан			
ФИО			
Родион			
Пристром			
Юрьевич			
Телефон			
+375296857421			

Рисунок 4.7 – Просмотр сохранённого заказа

Поиск по заказам у работника и администратора производится с помощью поисковой строки в правом верхнем углу (Рисунок 4.8). Введём в поисковую строку слог «Ремонт».

Результат работы представлен на Рисунке 4.9.



A search bar with a light blue border and a rounded rectangle. Inside, the word "Search" is written in a light blue font. To the right of the input field is a green button with the word "Поиск" in white text.

Рисунок 4.8 – Поисковая строка

S

Добавить услугу

Просмотр заказов

Добавить заказ

Просмотр услуг

Просмотр выполненных работ

Logout

Пристром

Поиск

Результат поиска заказа:

Номер	Услуга	ФИО	Телефон	Статус
402898f4794c187601794d1e53460004	Ремонт принтера	Пристром Родион Юрьевич	+375296857421	ADOPTED
<div>1</div>				

Рисунок 4.9 – Результат поиска

Администратор может добавлять услуги (Рисунок 4.10).

[S](#) [Добавить услугу](#) [Просмотр заказов](#) [Добавить](#)

Добавить услугу:

Наименование	Наименование
Цена	Цена

Submit

Рисунок 4.10 – Страница добавления услуги

Результат добавления услуги «Замена стекла». (Рисунок 4.11)

Услуга	Цена
Замена стекла	от 50.00
Обновление ПО	от 15.00
Ремонт принтера	от 12.00

Рисунок 4.11 – Обновлённая таблица

Работник при нажатии на «Принять» подтверждает, что он взял заказ на выполнение (Рисунок 4.12).

Заказы:

Номер	Услуга	
1	Обновление ПО	Принять
402898f4794c187601794d1e53460004	Ремонт принтера	Принять

Рисунок 4.12 –Принятие заказа на выполнение

Работник при нажатии на «Работа выполнена» подтверждает, что он закончил выполнение заказа (Рисунок 4.13).

Заказы в процессе выполнения:

Номер	Услуга	
2	ывапро	Работа выполнена
402898f4794c187601794c195ecf0000	Ремонт принтера	Работа выполнена

Рисунок 4.13 – Таблица с заказа в процессе выполнения

Также работник и администратор могут просматривать выполненные заказы (Рисунок 4.14).

Выполненные заказы:

Номер	Услуга	ФИО	Телефон
3	Ремонт принтера	Иванов Иван Иванович	+375333125487

Рисунок 4.14 – Таблица с выполненными заказами

Администратор может просмотреть все заказы со статусами (Рисунок 4.15)

Результат поиска заказа:

Номер	Услуга	ФИО	Телефон	Статус
1	Обновление ПО	Иванова Ирина Иванова	+375298547548	ADOPTED
2	ывапро	Иванов Иван Иванович	+375298547548	PROGRESS
3	Ремонт принтера	Иванов Иван Иванович	+375333125487	PERFORMED
402898f4794c187601794c195ecf0000	Ремонт принтера	пар авпр пар	345	PROGRESS
402898f4794c187601794d1e53460004	Ремонт принтера	Пристром Родион Юрьевич	+375296857421	ADOPTED
<div>1</div>				

Рисунок 2.15 – Таблица со всеми заказами

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была детально изучена предметная область работы с заказами по ремонте оргтехники, построены функциональные и информационные модели с использованием стандартов IDEF.

Итогом написанной курсовой работы является программное приложение, которое предоставляет возможность делать заказы на ремонт оргтехники. Данное приложение является доступным и понятным любому пользователю, располагает удобным и минималистичным интерфейсом, а также соответствует всем требованиям, предъявленным к курсовому проекту.

Данная программа является web-приложением, что говорит о том, что множество клиентов, расположенных на определенном расстоянии друг от друга могут одновременно общаться с сервером.

Вся используемая информация хранится в базе данных, разработанной для данного проекта. Доступ к базе данных был реализован только со стороны веб-сервиса.

Проанализировав все моменты можно сказать, что данная программа соответствует всем требованиям курсового проекта и все задачи, поставленные перед разработкой программы, соблюдены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] [Электронный ресурс]. – Электронные данные. – Режим доступа <https://cyberleninka.ru/article/n/planirovanie-lichnogo-semeynogo-byudzheta/viewer>
- [2] [Электронный ресурс]. – Электронные данные. – Режим доступа <https://cyberleninka.ru/article/n/filosofiya-semeynogo-byudzheta>
- [3] [Электронный ресурс]. – Электронные данные. – Режим доступа <https://cyberleninka.ru/article/n/modelirovanie-semeynogo-byudzheta>
- [4] [Электронный ресурс]. – Электронные данные. – Режим доступа <https://docs.oracle.com/en/java/>
- [5] [Электронный ресурс]. – Серверные языки: Java (обзор). – Режим доступа <https://codomaza.com/article/servernye-jazyki-java-obzor>

ПРИЛОЖЕНИЕ А
(обязательное)
Функциональная модель системы (IDEF0)

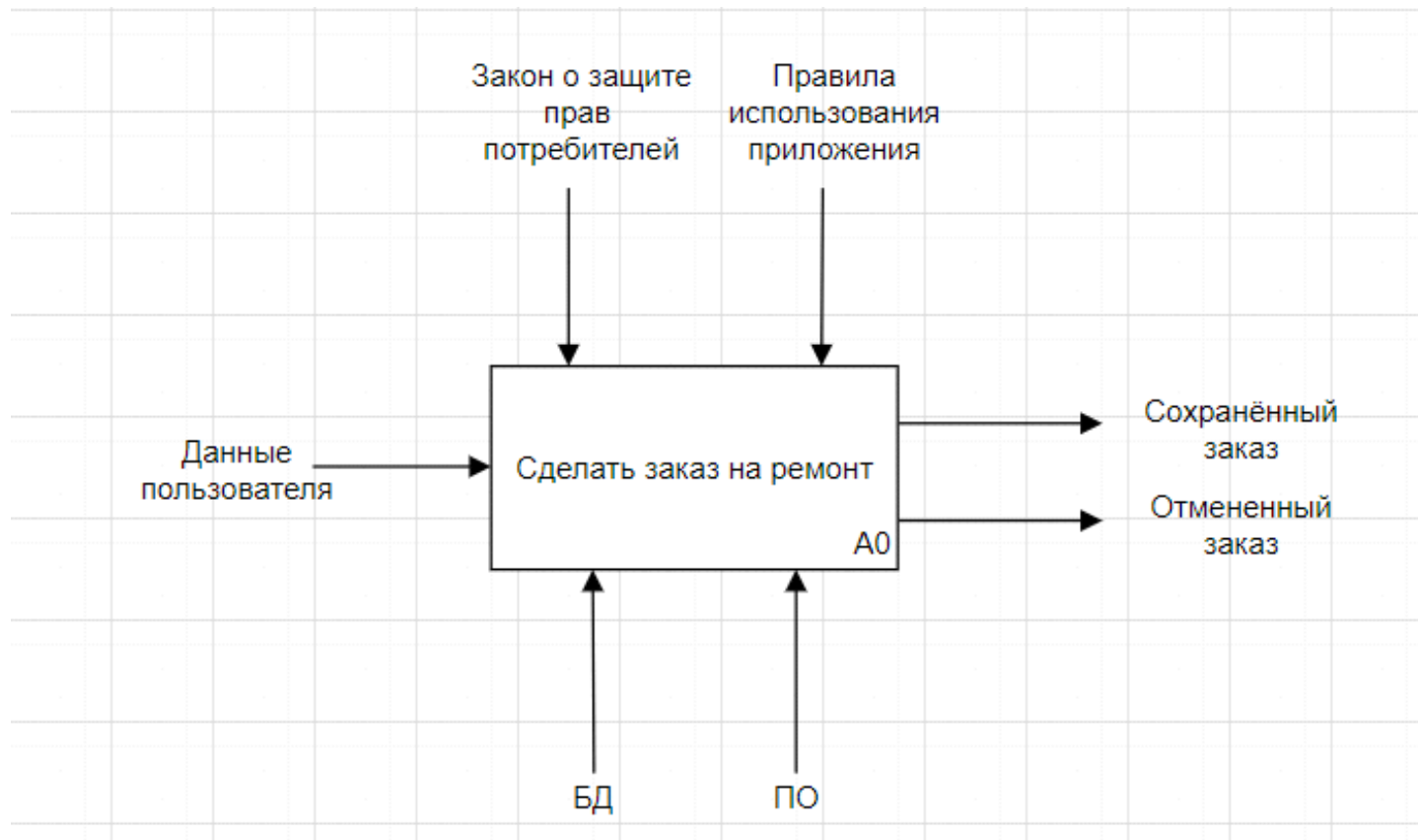


Рисунок А.1 – Функциональная модель предметной области

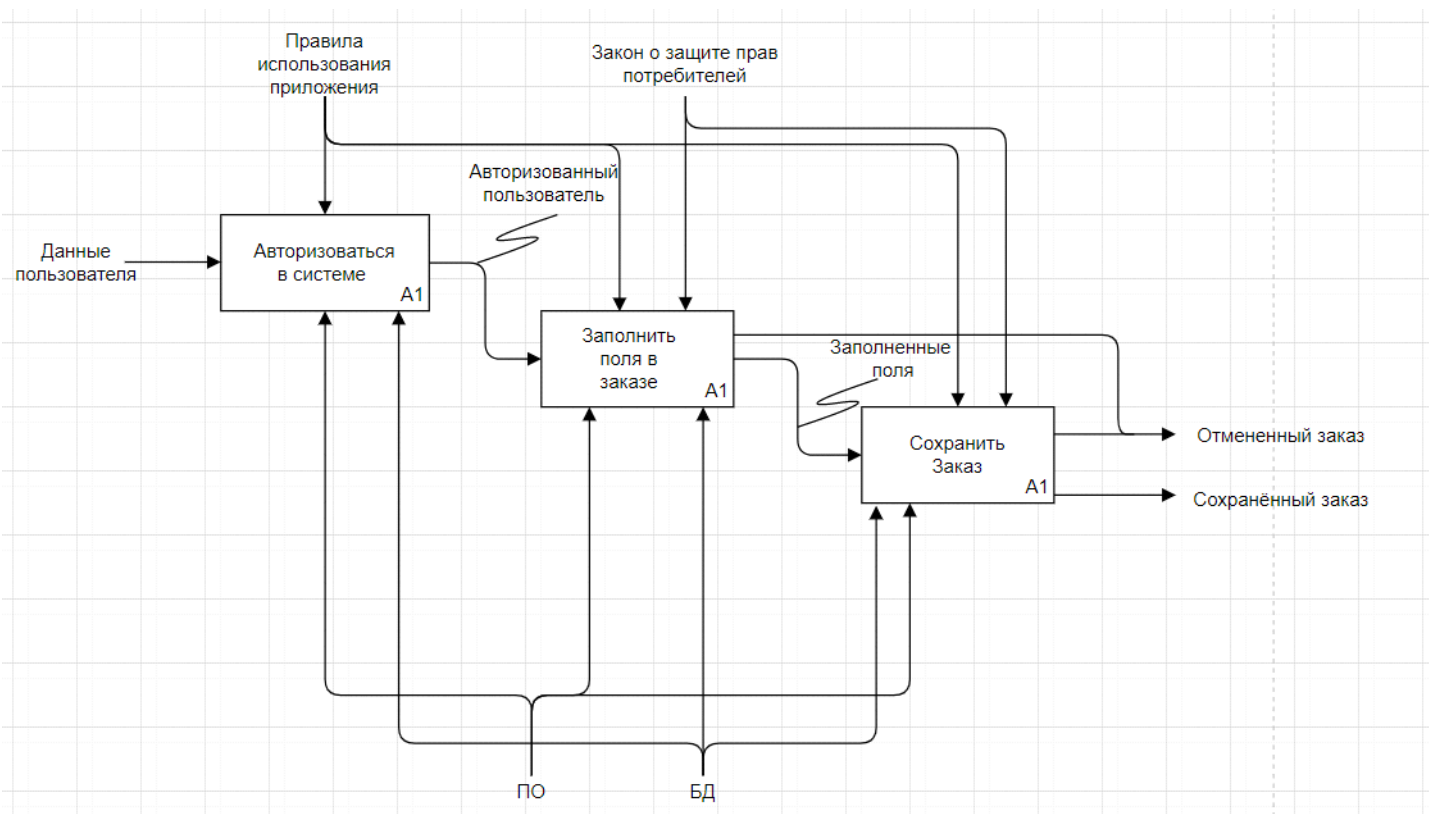


Рисунок А.2 – Декомпозиция функциональной модели

ПРИЛОЖЕНИЕ Б **(обязательное)** **Диаграммы на основе UML**

Рисунок Б.1 – Диаграмма вариантов использования

ПРИЛОЖЕНИЕ В
SQL скрипт базы данных

```
CREATE SCHEMA IF NOT EXISTS `system_pristrom` DEFAULT CHARACTER  
SET utf8mb4 COLLATE utf8mb4_0900_ai_ci ;  
USE `system_pristrom` ;
```

```
-- -----  
-- Table `system_pristrom`.`t_product`
```

```
-----  
CREATE TABLE IF NOT EXISTS `system_pristrom`.`t_product` (  
  `PRODUCT_ID` VARCHAR(255) NOT NULL,  
  `P_PRODUCT_DESC` VARCHAR(255) NULL DEFAULT NULL,  
  `P_PRODUCT_NAME` VARCHAR(255) NULL DEFAULT NULL,  
  PRIMARY KEY (`PRODUCT_ID`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb4  
COLLATE = utf8mb4_0900_ai_ci;
```

```
-----  
-- Table `system_pristrom`.`t_person`  
-----
```

```
CREATE TABLE IF NOT EXISTS `system_pristrom`.`t_person` (  
  `PERSON_ID` VARCHAR(255) NOT NULL,  
  `P_MID_NAME` VARCHAR(1000) NULL DEFAULT NULL,  
  `P_NAME` VARCHAR(1000) NULL DEFAULT NULL,  
  `P_SEC_NAME` VARCHAR(1000) NULL DEFAULT NULL,  
  PRIMARY KEY (`PERSON_ID`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb4  
COLLATE = utf8mb4_0900_ai_ci;
```

```
-----  
-- Table `system_pristrom`.`t_service`  
-----
```

```
CREATE TABLE IF NOT EXISTS `system_pristrom`.`t_service` (  
  `SERVICE_ID` VARCHAR(255) NOT NULL,  
  `S_SERVICE_NAME` VARCHAR(255) NULL DEFAULT NULL,  
  `S_SERVICE_PRICE` DECIMAL(19,2) NULL DEFAULT NULL,  
  PRIMARY KEY (`SERVICE_ID`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb4  
COLLATE = utf8mb4_0900_ai_ci;
```

```
-----  
-- Table `system_pristrom`.`t_phone_number`  
-----
```

```
CREATE TABLE IF NOT EXISTS `system_pristrom`.`t_phone_number` (  
  `PHONE_ID` VARCHAR(255) NOT NULL,
```

```
`PH_NUMBER` VARCHAR(255) NULL DEFAULT NULL,  
PRIMARY KEY (`PHONE_ID`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb4  
COLLATE = utf8mb4_0900_ai_ci;
```

```
-----  
-- Table `system_pristrom`.`t_order`  
-----
```

```
CREATE TABLE IF NOT EXISTS `system_pristrom`.`t_order` (  
  `ORDER_ID` VARCHAR(255) NOT NULL,  
  `O_STATUS` VARCHAR(255) NULL DEFAULT NULL,  
  `O_PERSON` VARCHAR(255) NULL DEFAULT NULL,  
  `O_PHONE_NUMBER` VARCHAR(255) NULL DEFAULT NULL,  
  `O_PRODUCT` VARCHAR(255) NULL DEFAULT NULL,  
  `O_SERVICE` VARCHAR(255) NULL DEFAULT NULL,  
  PRIMARY KEY (`ORDER_ID`),  
  INDEX `FKofpqyohwts7f8rp6kgbcasxy` (`O_PERSON` ASC) VISIBLE,  
  INDEX `FKu61kl309iygk8ge0ef8t8ihe` (`O_PHONE_NUMBER` ASC) VISIBLE,  
  INDEX `FKdu32xayukt5j1e2ck6uw4nkm1` (`O_PRODUCT` ASC) VISIBLE,  
  INDEX `FKrlttmgeec6h331thestns0gm2` (`O_SERVICE` ASC) VISIBLE,  
  CONSTRAINT `FKdu32xayukt5j1e2ck6uw4nkm1`  
    FOREIGN KEY (`O_PRODUCT`)  
      REFERENCES `system_pristrom`.`t_product` (`PRODUCT_ID`),  
  CONSTRAINT `FKofpqyohwts7f8rp6kgbcasxy`  
    FOREIGN KEY (`O_PERSON`)  
      REFERENCES `system_pristrom`.`t_person` (`PERSON_ID`),  
  CONSTRAINT `FKrlttmgeec6h331thestns0gm2`  
    FOREIGN KEY (`O_SERVICE`)  
      REFERENCES `system_pristrom`.`t_service` (`SERVICE_ID`),  
  CONSTRAINT `FKu61kl309iygk8ge0ef8t8ihe`  
    FOREIGN KEY (`O_PHONE_NUMBER`)  
      REFERENCES `system_pristrom`.`t_phone_number` (`PHONE_ID`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb4
```