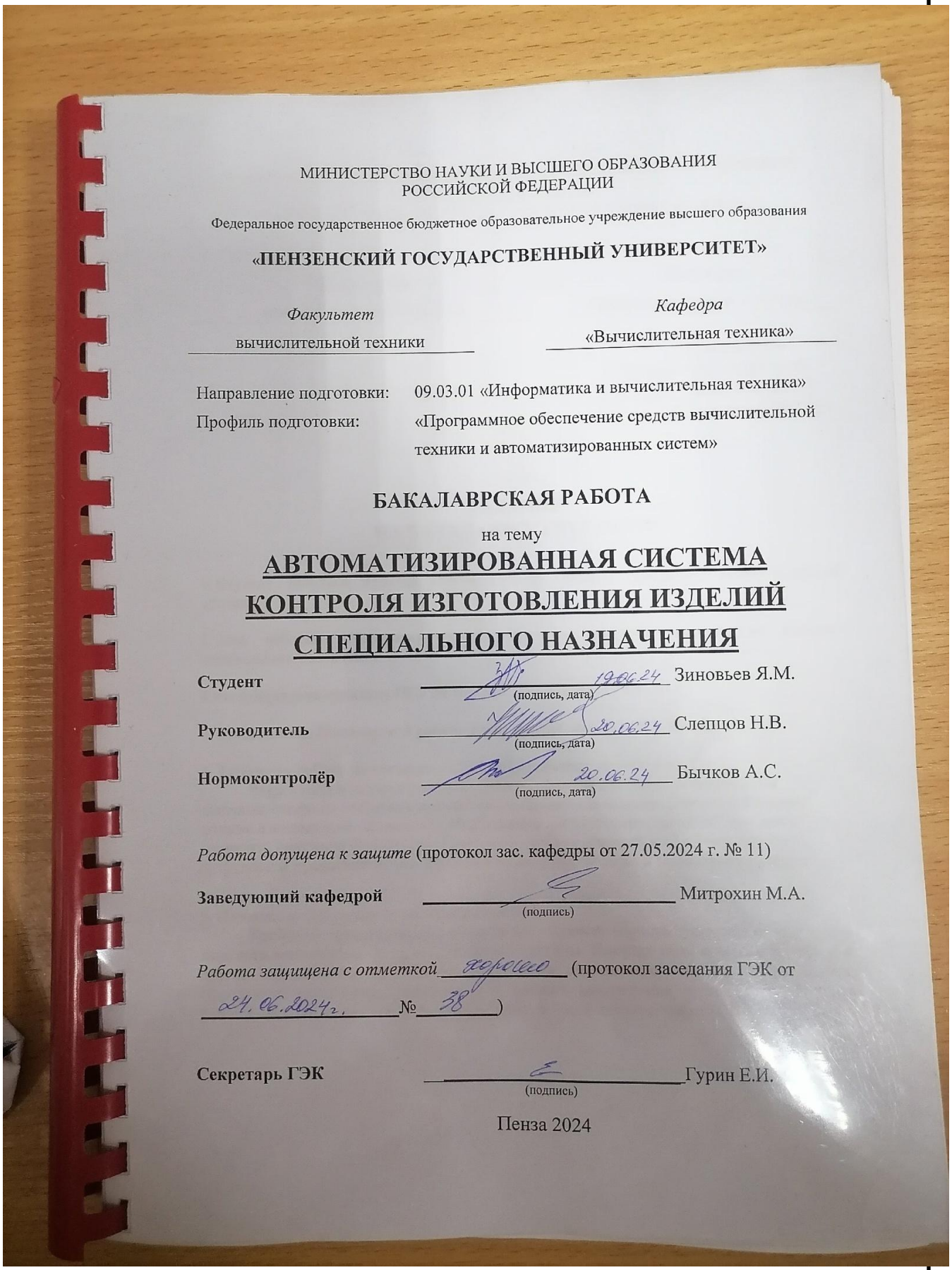


МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

АВТОМАТИЗИРОВАННАЯ СИСТЕМА КОНТРОЛЯ ИЗГОТОВЛЕНИЯ ИЗДЕЛИЙ СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ

Студент _____ (подпись, дата) 19.06.24 Зиновьев Я.М.

Руководитель _____ (подпись, дата) 20.06.24 Слепцов Н.В.

Нормоконтролёр _____ (подпись, дата) 20.06.24 Бычков А.С.

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой _____ (подпись) Митрохин М.А.

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 24.06.2024г. № 38)

Секретарь ГЭК _____ (подпись) Гурин Е.И.

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«19» апреля 2024 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Зиновьев Ярослав Михайлович гр. 20ВВП1 факультета Вычислительной техники.

2.Тема работы: «Автоматизированная система контроля изготовления изделий специального назначения»

Тема утверждена приказом ПГУ № 405/о от "19" апреля 2024 г.

3. Руководитель бакалаврской работы к.т.н., доцент, каф. ВТ ПГУ Слепцов Н.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать и реализовать программное обеспечение предназначенное для контроля этапов изготовления изделий специального назначения. Данное приложение должно в автоматическом режиме обрабатывать поступающие данные об изделии и сохранять полученный результат. Программа должна ускорить и облегчить работу сотрудников предприятия, что обеспечит более качественную и эффективную работу.

5. Перечень подлежащих разработке вопросов
Требуется провести анализ предметных областей: изучить существующие аналоги, выделить достоинства и недостатки данной системы. Выбрать язык программирования и другие составляющие разработки, подходящие для оптимальной реализации задачи. Определить архитектуру приложения и описать внутреннюю организацию. Изучить функционал библиотек для работы с Qt Creator и базой данных PostgreSQL.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	20	26.04.2024 – 27.04.2024	
2	Анализ требований	32	28.04.2024 – 30.04.2024	
3	Разработка основной части	82	31.04.2024 – 14.05.2024	
4	Разработка логической модели	16	15.05.2024 – 16.05.2024	
5	Разработка физической модели данных системы	40	16.05.2024 – 19.05.2024	
6	Отладка и тестирование	50	19.05.2024 – 23.05.2024	
7	Оформление пояснительной записки	54	24.05.2024 – 31.05.2024	

Дата выдачи задания "19" апреля 2024 г.

Руководитель бакалаврской работы _____ Слепцов Н.В.
(подпись, дата)

Задание к исполнению принял студент _____ Зиновьев Я.М.
(подпись, дата)

Работу к защите допустить
Декан факультета 29.06.2024 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 124 листа, 22 рисунка, 11 таблиц, 6 источников, 4 приложения

LINUX, QTCREATOR, БАЗА ДАННЫХ, АВТОМАТИЗИРОВАННАЯ СИСТЕМА, C++, POSTGRESQL

Объектом исследования является процесс контроля изготовления изделий в научно-техническом центре.

Цель работы – автоматизация процесса контроля изготовления изделий специального назначения.

В процессе работы проведён анализ предметной области, проведено концептуальное проектирование (модели системы), разработаны логическая и физическая модели системы, разработана структура приложения и проведён анализ целесообразности использования заданных средств разработки.

В результате проделанной работы разработана автоматизированная система контроля изготовления изделий специального назначения.

Автоматизированная система контроля изготовления изделий специального назначения может использоваться с незначительными доработками в подразделениях научно-технического центра.

Эффективность разработки заключается в значительном снижении временных затрат на составление оперативных документов и повышении качества изготовления изделий.

Среда разработки программного обеспечения QtCreator.

Содержание

Перечень сокращений и обозначений 9

Введение 10

1 Автоматизированная система контроля изготовления изделий специального назначения 11

1.1 Анализ предметной области 11

1.1.1 Актуальность разработки 11

1.1.2 Обоснование актуальности разработки программы 13

1.1.3 Постановка задачи 14

1.1.4 Стадии и этапы разработки 17

1.2 Выбор технологий и средств разработки приложения 18

1.2.1 Выбор языка программирования 18

1.2.2 Выбор СУБД 19

1.3 Проектирование и разработка автоматизированной системы контроля изготовления изделий специализированного назначения 21

1.3.1 Архитектура приложения 21

1.3.2 Разработка структуры приложения 25

1.4 Разработка базы данных для автоматизированной системы контроля изготовления изделий специализированного назначения 30

1.4.1 Разработка логической модели данных системы 30

1.4.2 Разработка структуры связей 35

1.4.3 Нормализация отношений 36

1.4.4 Разработка физической модели данных системы 45

1.5 Описание программы 50

1.5.1 Общие сведения 50

1.5.2 Описание логической структуры 50

1.5.3 Входные данные 63

1.5.4 Выходные данные 64

1.6 Тестирование 64

1.6.1 Объект испытаний 64

1.6.2 Цель испытаний 64

1.6.3 Требования к программе 64
1.6.4 Методы испытаний 64
1.6.5 Результаты испытаний системы 68
1.7 Описание применения 68
1.7.1 Назначение применения 68
1.7.2 Описание задачи 68
1.7.3 Входные и выходные данные 74
2 Охрана труда 75
2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения 75
2.1.1 Оборудование рабочего места 75
2.1.2 Освещенность рабочего места 77
2.2 Электробезопасность 78
2.2.1 Пожарная безопасность 79
2.2.2 Режим труда и отдыха 79
Заключение 81
Список использованных источников 82
Приложение А UML-диаграммы 83
Приложение А.1 – UML-диаграмма вариантов использования 84
Приложение А.2 – UML-диаграмма последовательностей 85
Приложение А.3 – UML-диаграмма классов 86
Приложение А.4 – UML-диаграмма развёртывания 87
Приложение Б Листинг программы 88
Приложение Г Презентация 117

Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения

БД – база данных

ГОЗ – государственные оборонные заказы

ГОСТ – единый государственный стандарт, который чётко устанавливает требования к качеству определённого изделения или определённой деятельности

ИВС – информационно вычислительные системы

НПП – научно-производственное предприятие

НТЦ – научно-технический центр

НФ – нормальная форма

ООП – объектно-ориентированное программирование

ОС – операционная система

ПРИ – предъявительские испытания

ПК – персональный компьютер

ПО – программное обеспечение

ПСИ – приемо-сдаточные испытания

СУБД – система управления базами данных

Фреймворк – программная платформа, которая упрощает разработку программного продукта, предварительно определяет структуру проекта

IDE (Integrated Development Environment) – комплекс программных средств для разработки программного обеспечения

UML (Unified Modeling Language) – язык графического описания для объектного моделирования, системного проектирования и отображения организационных структур

Введение

В настоящее время трудно представить нашу жизнь без информационных технологий. Вряд ли можно найти организацию, которая в той или иной степени не была бы оснащена средствами приёма, хранения, передачи, обработки информации.

Информация является одним из основных ресурсов роста производительности. Одним из видов программного обеспечения являются автоматизированные системы. Автоматизированная система состоит из компонентов, предназначенных для выполнения набора задач, которые были предварительно настроены. Функциональные и утомительные задачи становятся менее обременительными и делают рабочий процесс проще и легче.

В настоящее время основной сложностью ведения контроля производства в отделе является отсутствие автоматизированной обработки данных по изготовлению изделий и представления их в установленной форме. В частности, учет выполняемых этапов, работ ведется путем регистрации изменений в файле в текстовом редакторе офисного пакета. В связи с неудобством форматирования, сложности организации совместного доступа начальником НТЦ было принято решение об автоматизации ведения контроля производства изделий специального
назначения.

Цель выполнения работы автоматизация процесса контроля изготовления изделий специального назначения.

Основной задачей автоматизации является построение автоматизированной системы контроля изготовления изделий специального назначения, упрощающей доступ к данным, их хранение и изменение.

1 Автоматизированная система контроля изготовления изделий специального назначения

1.1 Анализ предметной области

1.1.1 Актуальность разработки

Основным местом внедрения данного приложения являются предприятия направлениями деятельности, которых являются разработка, производство и модернизация стационарных и мобильных автоматизированных систем и комплексов управления, информационных систем специального назначения, в том числе специальных машин на колесной и гусеничной базе. Разработкой изделий специального назначения занимаются научно-технические центры (НТЦ) предприятия.

Такие центры состоят из подразделений, выполняющих полный цикл изготовления указанных изделий.

Изготовление изделия происходит в соответствии с договором на изготовлении партии. Для изготовления изделий планово-экономическая группа на основании технической карты проводит анализ состава изделия и определяет необходимость закупки у сторонних производителей и изготовления определенных деталей собственными силами. Изготовление изделий собственными силами осуществляется на основании наряда, который оформляется на определенное количество рабочих часов для производственного отдела.

Комплектующие от сторонних производителей и из производственных цехов регистрируются сотрудником НТЦ ответственным за их расход.

Сборка изделий производится в соответствии с их технологическими картами и строго установленными этапами, включающими настройки и проверки изделий. Основными этапами и работами изготовления изделия являются:

– назначение ответственного за изделие;

– проверка шасси;

– прогон шасси;

– доработка шасси;

– укладка жгутов;

– установка каркасов;

– установка приборов;

– предъявление отделу технического контроля (ОТК);

– устранение замечаний;

– наладка питания;

– наладка связи;

– наладка комплексов;

– прогон изделия;

– дождевание;

– испытания в тепловой камере;

– испытание непрерывной работы на протяжении 72 часов;

– навигационная наладка;

– испытания на дальность;

– специальные проверки ПРИ, ПСИ.

Выполненные этапы фиксируются в техническом журнале изделия.

Основным отчетным документом подразделения НТЦ является таблица «ГОЗ», отражающая основные этапы изготовления изделий. Технический журнал изделия ведется ответственным. В журнале указываются все работы, проведенные на изделии, в хронологическом порядке. Начальник отдела или подразделения проверяет состояние изделия по техническому журналу и на его основе составляет оперативные доклады руководству и таблицу «Изготовление изделий серии ГОЗ».

Для контроля изготовления изделия начальнику подразделения или начальнику НТЦ предоставляются следующие данные:

– фамилия, имя, отчество ответственного за сборку изделия;

– подразделение сотрудника;

– номер изделия;

– тип изделия;

– перечень выполненных этапов изготовления изделия;

– наименование организации заказчика.

Для контроля расхода деталей от стороннего производителя, предоставляется информация о установленных на изделие деталях:

– наименование детали;

– дата изготовления детали,

– паспортный номер детали;

– инвентарный номер детали;

– дата установки на изделие;

– производитель детали;

– номер изделия в состав, которого входит деталь.

1.1.2 Обоснование актуальности разработки программы

В настоящее время основной сложностью ведения контроля производства в отделе является отсутствие автоматизированной обработки данных по изготовлению изделий и представления их в установленной форме.

В частности, учет выполняемых этапов, работ ведется путем регистрации изменений в файле в текстовом редакторе офисного пакета. В связи с неудобством форматирования, сложности организации совместного доступа начальником НТЦ было принято решение об автоматизации ведения контроля производства изделий специального назначенияс целью перехода на электронный документооборот.

Цель работы – автоматизация процесса контроля изготовления изделий специального назначения.

Основной задачей автоматизации является построение автоматизированной системы контроля изготовления изделий специального назначения, упрощающей доступ к данным, их хранение и изменение.

Предлагаемая для разработки автоматизированная система будет предназначена для начальников подразделений, начальника НТЦ и позволит автоматизировать выполнение следующих операций:

– учет и хранение информации о изделиях, сотрудниках, деталях, от стороннего производителя;

– оформление сводной таблицы «Изготовление серии ГОЗ»;

– оформление перечня деталей от стороннего производителя, установленных на изделие;

– оформление графика выполнения этапов изготовления по каждому изделию.

Ожидаемая эффективность от внедрения специализированного программного обеспечения заключается в сокращении временных затрат на поиск данных, формированиесводной таблицы «Изготовление ГОЗ», а также оперативной информации по изделиям в удобном виде.

1.1.3 Постановка задачи

Программное обеспечение должно обеспечить выполнение следующих функций:

– ввод и хранение данных о сотруднике: номер сотрудника, Ф.И.О., отдел;

– ввод и хранение данных о готовом изделии: номер изделия, тип изделия, организация заказчик, ответственный, наименование изделия, код изделия;

– ввод и хранение данных о деталях: номер детали, наименование, дата изготовления, паспортный номер детали, инвентарный номер детали, дата установки на изделие, производитель детали, изделие в состав которого входит деталь, номер вида детали, номер доработки детали;

– ввод и хранение данных о организации заказчике: номер организации, наименование, номер договора, юридический адрес;

– ввод и хранение данных о этапах сборки изделия: номер сборки, сборка жгутов дата начала, сборка жгутов дата окончания, сборка каркаса дата начала, сборка каркаса дата окончания, сборка приборов дата начала, сборка приборов

дата окончания, предъявление отделу технического контроля дата начала, предъявление отделу технического контроля дата окончания, устранение замечаний дата начала, дата окончания, номер изделия;

– ввод и хранение данных о этапах проверки изделия: номер проверки, проверка шасси дата начала, проверка шасси дата окончания, прогон дата начала, прогон дата окончания, доработка дата начала, доработка дата окончания, номер изделия;

– ввод и хранение данных о этапах наладки изделия: номер наладки, наладка питания дата начала, наладка питания дата окончания, наладка автоматизированных систем дата начала, наладка автоматизированных систем дата окончания, наладка связи дата начала, наладка связи дата окончания, комплексная наладка дата начала, комплексная наладка дата окончания, прогон изделия дата начала, прогон изделия дата окончания, дождевание дата начала, дождевание дата окончания, испытания в тепловой камере дата начала, испытание в тепловой камере дата окончания, наладка протоколов ЭТГ дата начала, наладка протоколов ЭТГ дата окончания, функционирование 72 часа дата начала, функционирование 72 часа дата окончания, навигация дата начала, навигация дата окончания, дальность дата начала, дальность дата окончания, специальные проверки дата начала, специальные проверки дата окончания, номер изделия;

– ввод и хранение данных о этапах сдачи изделия: номер сдачи, Предъявительские испытания (ПРИ) дата начала, ПРИ дата окончания, приемо-сдаточные испытания (ПСИ) дата начала, ПСИ дата окончания, отгрузка дата начала, отгрузка дата окончания, номер изделия;

– ввод и хранение данных о примечаниях по изделию: номер примечания, текст примечания, номер изделия;

– формирование и вывод на печать: сводной таблицы «Изготовление серии ГОЗ», перечня деталей от сторонних производителей, графика выполнения этапов по каждому изделию;

– входными данными программного обеспечения должны быть сведения о сотрудниках, изделиях и их характеристиках, организациях-заказчиках, деталях, работах по сборке, проверке, наладке изделия, сдаче изделия.

Выходными данными программного обеспечения должны быть отчёты:

– «Изготовление серии ГОЗ» имеющая форму сводной таблицы и содержащей данные о этапах, сотрудниках, изделиях, датах начала выполнения этапов, датах окончания выполнения этапов;

– «Перечень деталей от стороннего производителя», содержащие сведения о деталях от стороннего производителя и изделия, на которые они установлены;

– «График выполнения этапов изделия», содержащие сведения о прохождении этапов изготовления по каждому изделию.

Также программное обеспечение должно выдавать сообщения о возникших ошибках, при неверном задании исходных данных, поддерживать диалоговый режим в рамках предоставляемых пользователю возможностей.

В программном обеспечении должна быть реализована защита хранимых данных от несанкционированного доступа с использованием механизма идентификации пользователя.

Разрабатываемая автоматизированная система должна соответствовать следующим техническим требованиям:

– удобный и интуитивно понятный интерфейс для работы с программой;

– разработанная программа является десктопным решением;

– наличие возможности сохранения данных локально, пока отсутствует подключения к серверу;

– операционная система: Windows 7 и более поздние версии;

– оперативная память: не менее 512 МБ ОЗУ;

– незанятное дисковое пространство: не менее 1,5 ГБ ПЗУ;

– процессор: Intel Pentium III 800 MHz;

– разрешение экрана: 1024 x 728.

Автоматизированная система должна предоставлять функционал в соответствии с UML-диаграммой вариантов использования, представленной на рисунке 1.

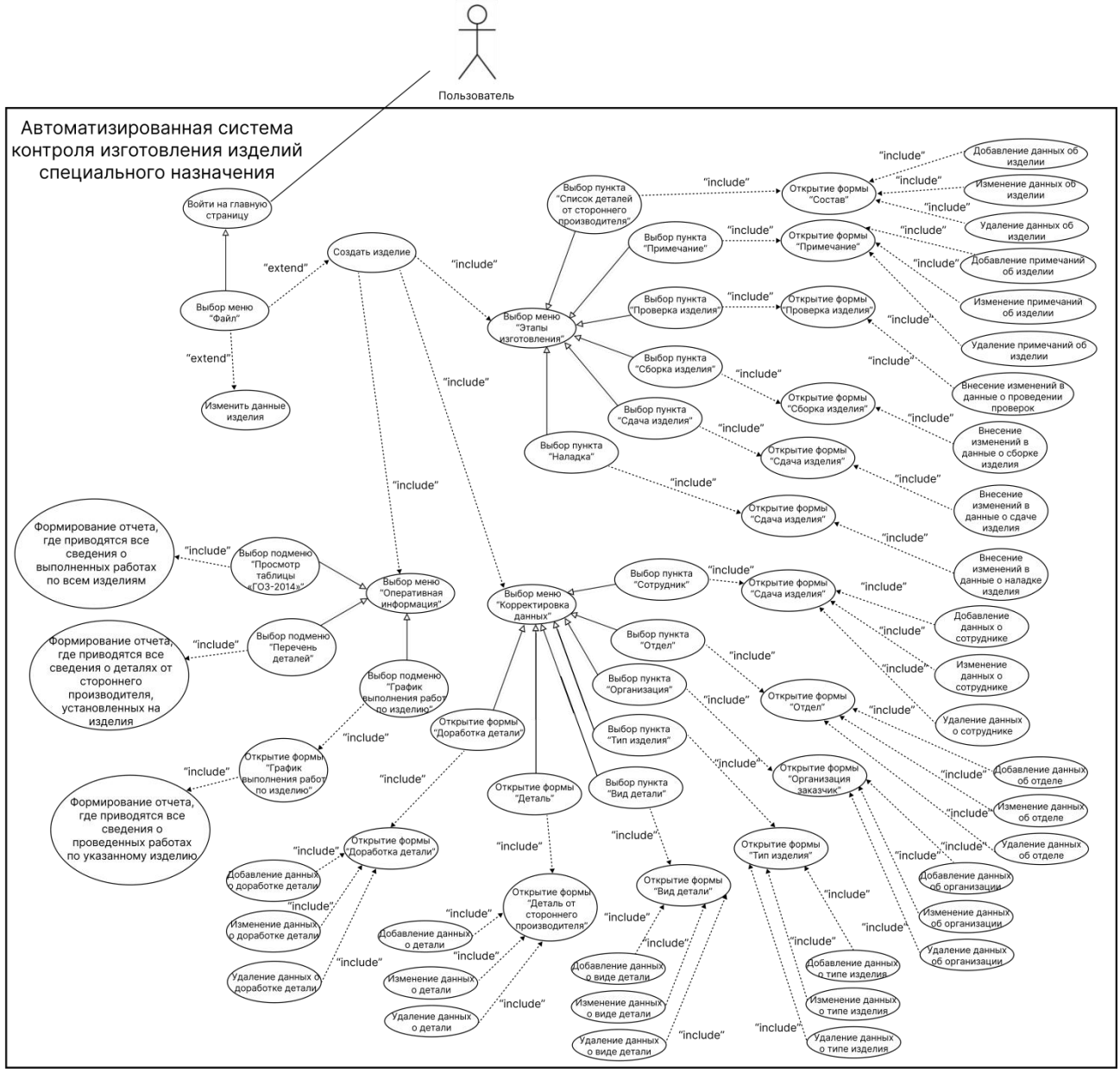


Рисунок 1 – UML-диаграмма вариантов использования

1.1.4 Стадии и этапы разработки

Программное обеспечение должно разрабатываться в соответствии со следующими этапами:

– анализ предметной области;

– разработка технического задания;

– функциональное проектирование модели системы;

– разработка БД системы;

– тестирование и отладка программного обеспечения;

– разработка программной документации;

– оформление пояснительной записки.

1.2 Выбор технологий и средств разработки приложения

Программное обеспечение должно быть написано в среде QtCreator на языке C++ с использованием фреймворка Qt. База данных должна быть создана с использованием PostgresSQL для ОС AstraLinux.

Для обеспечения нормальной работоспособности приложения на персональном компьютере должны быть установлены следующие программные продукты:

– операционная система AstraLinux;

– СУБД PostgresSQL;

– QtCreator.

1.2.1 Выбор языка программирования

Язык программирования C++.

C++ – это высокоуровневый язык программирования, обладающий возможностями как низкоуровневых языков (например, C), так и объектно-ориентированных языков [1]. Данный язык позволяет писать эффективный и быстродействующий код благодаря возможности работы с указателями, множеством библиотек и шаблонами. Он также поддерживает многопоточность и множество парадигм программирования. Однако C++ не является достаточно гибким для создания графического интерфейса приложения.

Язык программирования Python.

Python – высокоуровневый интерпретируемый язык программирования с динамической типизацией [2]. Python известен своей простотой и читаемостью кода, богатой стандартной библиотекой, поддержкой различных парадигм

программирования (функциональное, объектно-ориентированное). Он также широко используется в области науки о данных и искусственного интеллекта.

Самой простой библиотекой для создания графического интерфейса в Python является Tkinter. Эта библиотека предоставляет базовые средства для создания простых GUI-приложений, однако для решения более серьезных задач не пригодна.

Язык программирования Java.

Java – это объектно-ориентированный язык программирования, который изначально разрабатывался для создания кроссплатформенных приложений [3].

Java обладает простым синтаксисом, автоматическим управлением памятью, механизмом сборки мусора. Этот язык позволяет создавать надежные и безопасные приложения благодаря своей виртуальной машине JVM (JavaVirtualMachine).

Для создания графического интерфейса в Java часто применяется библиотека Swing, предоставляющая широкие возможности для создания различных элементов интерфейса.

1.2.2 Выбор СУБД

СУБД PostgreSQL.

В качестве СУБД была PostgreSQL – свободно распространяемая объектно-реляционная система управления базами данных, предоставляющая широкий спектр функций для хранения, организации и обработки данных [4]. Данная СУБД распространяется на бесплатной основе (OpenSource) и имеет открытый исходной код.

Среди преимуществ PostgreSQL можно выделить:

– позволяет работать с большими размерами данных;

– кроссплатформенность;

– расширяемость и богатый набор типов данных;

– наследование;

– активное сообщество пользователей;

– наличие официальной исчерпывающей документации.

Решением использовать PostgreSQL стало то, что данную СУБД можно быстро и бесплатно установить на любую необходимую платформу. Кроме того, для PostgreSQL были созданы различные графические библиотеки и приложения для удобной работы с ним.

СУБД MySQL.

MySQL – это одна из самых популярных открытых реляционных систем управления базами данных (СУБД). Она широко используется для хранения данных веб-приложений, бизнес-приложений и других типов приложений.

Преимущества MySQL:

– доступна бесплатно для использования, что делает ее доступной для широкого круга разработчиков;

– предлагает простой и интуитивно понятный язык запросов SQL для работы с данными;

– обладает хорошей производительностью и способна эффективно обрабатывать большие объёмы данных;

– легко масштабируется как вертикально (добавление ресурсов на одном сервере), так и горизонтально (добавление дополнительных серверов).

Однако стоит помнить, что MySQL не всегда поддерживает хранимые процедуры и триггеры так эффективно, как другие СУБД. Кроме того, могут возникнуть сложности с масштабированием БД на большие объемы данных.

СУБД SQLite.

SQLite – это компактная встраиваемая реляционная система управления базами данных (СУБД). Она отличается от клиент-серверных СУБД, таких как MySQL, тем, что база данных SQLite хранится в одном файле, который не требует отдельного сервера для работы. SQLite широко используется в мобильных приложениях, встроенных системах и других проектах, где требуется легкая и простая в использовании база данных.

1.3 Проектирование и разработка автоматизированной системы контроля изготовления изделий специализированного назначения

1.3.1 Архитектура приложения

Для описания архитектуры приложения далее будут представлены UML-диаграммы.

Рисунок 2 демонстрирует UML-диаграмму развёртывания, которая показывает, как устроен проект: где находятся его части (на пользовательском ПК) и как они связаны между собой.

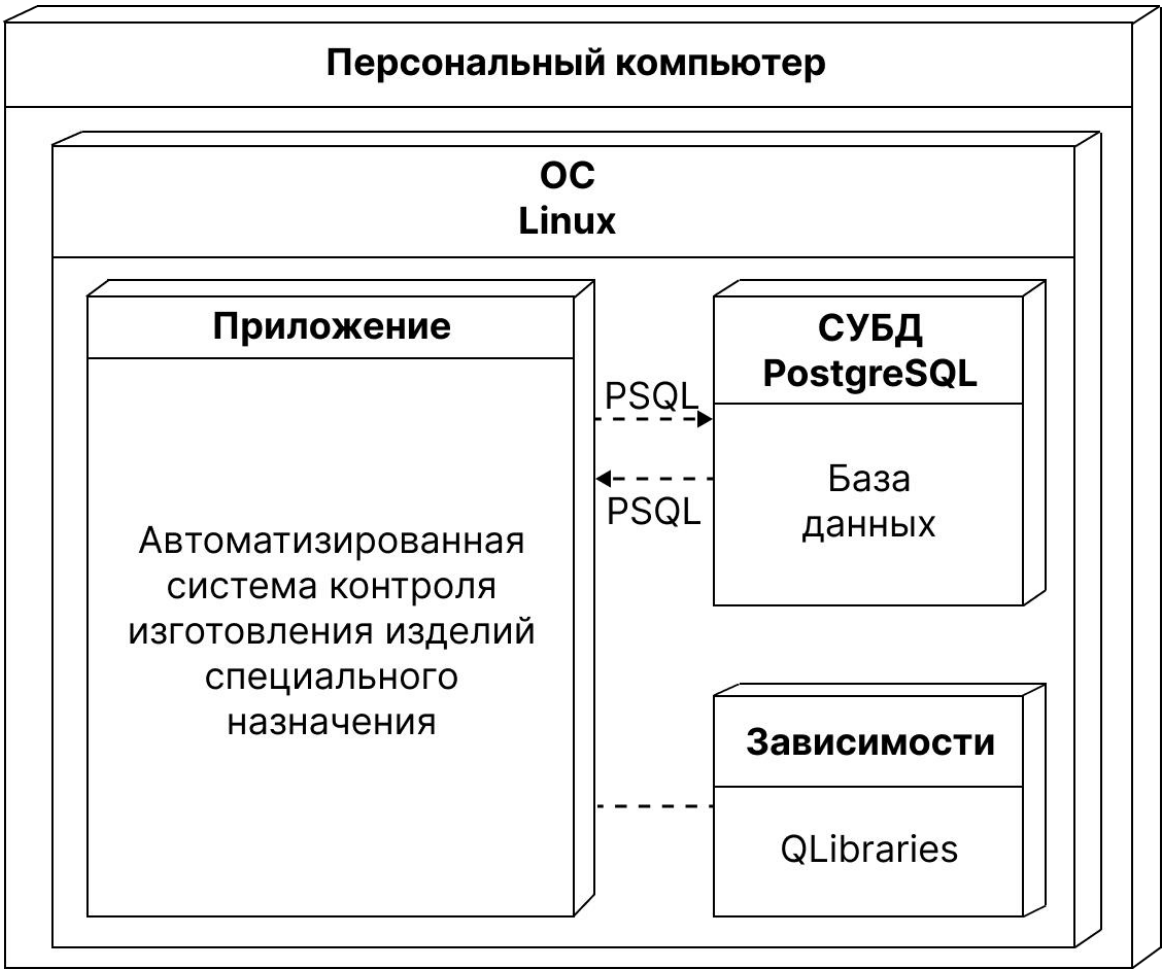


Рисунок 2 – UML-диаграмма развёртывания

А на рисунке 3 представлена UML-диаграмма последовательностей, по которой видно, как пользователь взаимодействует с приложением, а также как приложение обращается к базе данных для получения или занесения данных.

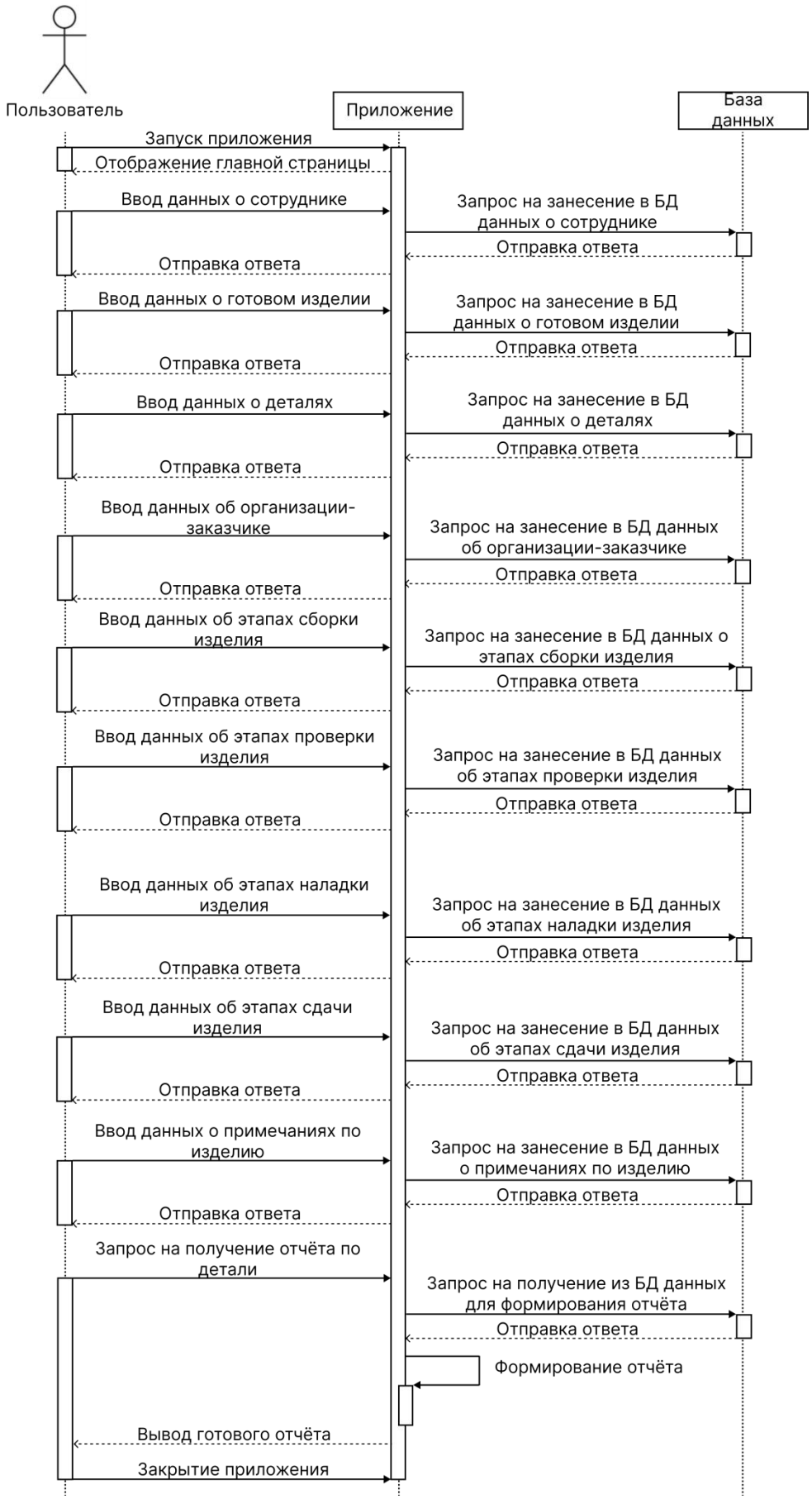


Рисунок 3 – UML-диаграмма последовательностей

Входными данными системы «Контроль изготовления изделий специального назначения» являются:

– информация о детали;

– информация о сотруднике;

– информация о изделии;

– информация о выполненных работах;

– информация о организации-заказчике.

Выходными данными для бизнес-процесса являются:

– таблица «Изготовление серии ГОЗ»;

– перечень деталей, установленных на изделие;

– график выполнения этапов изготовления по каждому изделию.

Реализация функций контроля изготовления изделий специального назначения ведется в соответствии с технологической картой и ГОСТ (стрелка управления), начальником отдела или подразделения (стрелка-механизм).

Контекстная диаграмма функциональной модели системы представлена на рисунке 4.

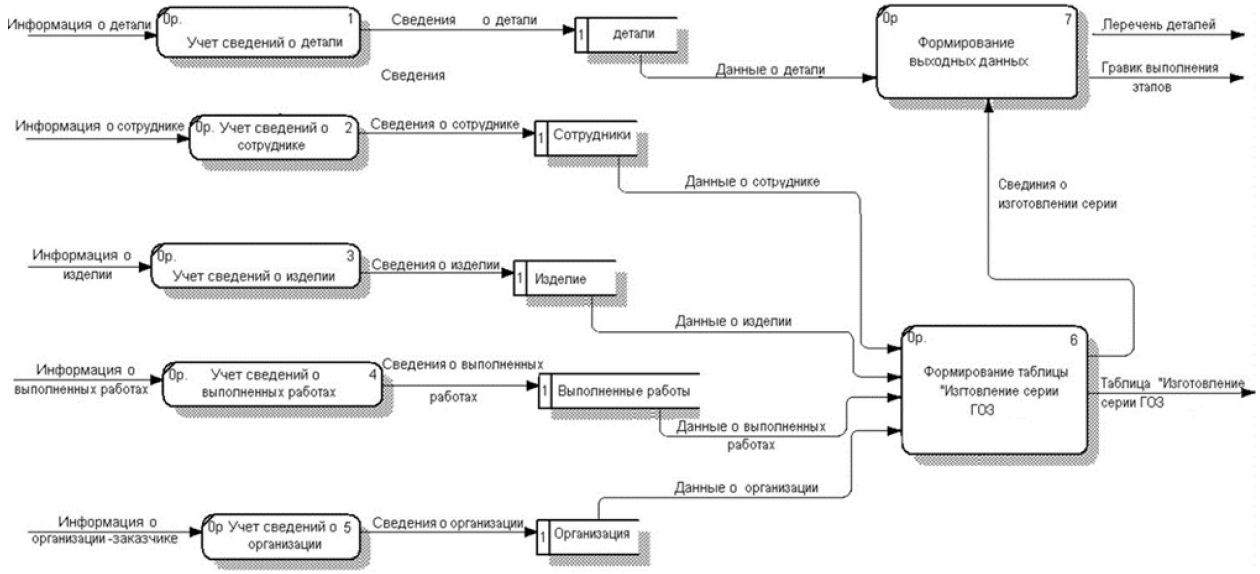


Рисунок 4 – Контекстная диаграмма

Функциональная декомпозиция процесса «Контроль изготовления изделий специального назначения» проведена на основе результатов анализа предметной области. В процессе декомпозиции выделено 7 процессов:

– учет сведений о детали;

– учет сведений осотруднике;

– учет сведений о изделии;

– учет сведений о выполненных работах;

– учет сведений о организации;

– формирование таблицы «Изготовление серии ГОЗ»;

– формирование выходных данных.

В процессе декомпозиции выделено 5 накопителей, предназначенных для хранения соответствующих данных: детали, сотрудники, о изделие, выполненные работы, о организации.

– Процесс «Учет сведений о детали» предназначен для учета и корректировки данных, о деталях, поступающих от стороннего производителя, установленных на изделиях;

– Процесс «Учет сведений о сотруднике» предназначен для учета и корректировки данных по сотрудникам ответственным за изделия;

– Процесс «Учет сведений о изделии» предназначен для учета и корректировки данных о изделии;

– Процесс «Учет сведений о выполненных работах» предназначен для учета и корректировки сведений о выполненных этапах изготовления изделий и корректировки данных;

– Процесс «Учет сведений об организации» предназначен для учета и корректировки данных об организациях-заказчиках.

Процесс «Формирование таблицы «Изготовление серии ГОЗ» предназначен для формирования отчета по изготовленным изделиям в виде таблицы с целью контроля за выполнением этапов работ изготовления всех изделий.

Процесс «Формирование выходных данных» предназначен для формирования отчетов НТЦ и его подразделений.

Входные и выходные данные для выделенных процессов представлены в таблице 1.

Таблица 1 – Входные и выходные данные процессов

Процесс	Входные данные	Выходные данные
Учет сведений о детали	Информация о детали	Сведения о детали
Учет сведений о сотруднике	Информация о сотруднике	Сведения о сотруднике
Учет сведений о изделии	Информация о изделии	Сведения о изделии
Учет сведений о выполненных работах	Информация о выполненных работах	Сведения о выполненных работах
Учет сведений о организации	Информация об организации-заказчике	Сведения об организации
Формирование таблицы «Изготовление серии ГОЗ»	Данные о сотруднике	Таблица «Изготовление серии ГОЗ»
	Данные о изделии	
	Данные о выполненных работах	
	Данные об организации	
Формирование выходных данных	Данные о детали	Перечень деталей
	Данные о изготовлении серии	График выполнения этапов

В результате анализа диаграммы декомпозиции выявлено, что бизнес-процесс «Контроль изготовления изделий специального назначения» декомпозирован до элементарного уровня, и этого достаточного для разработки структуры приложения и реализации автоматизированной системы контроля изготовления изделий специального назначения.

1.3.2 Разработка структуры приложения

Разработка структуры приложения осуществлена на основе функциональной модели системы и модели данных системы. На основе

функциональной модели системы выделены основные функции системы. Иерархия функций приложения приведена на рисунке 5.

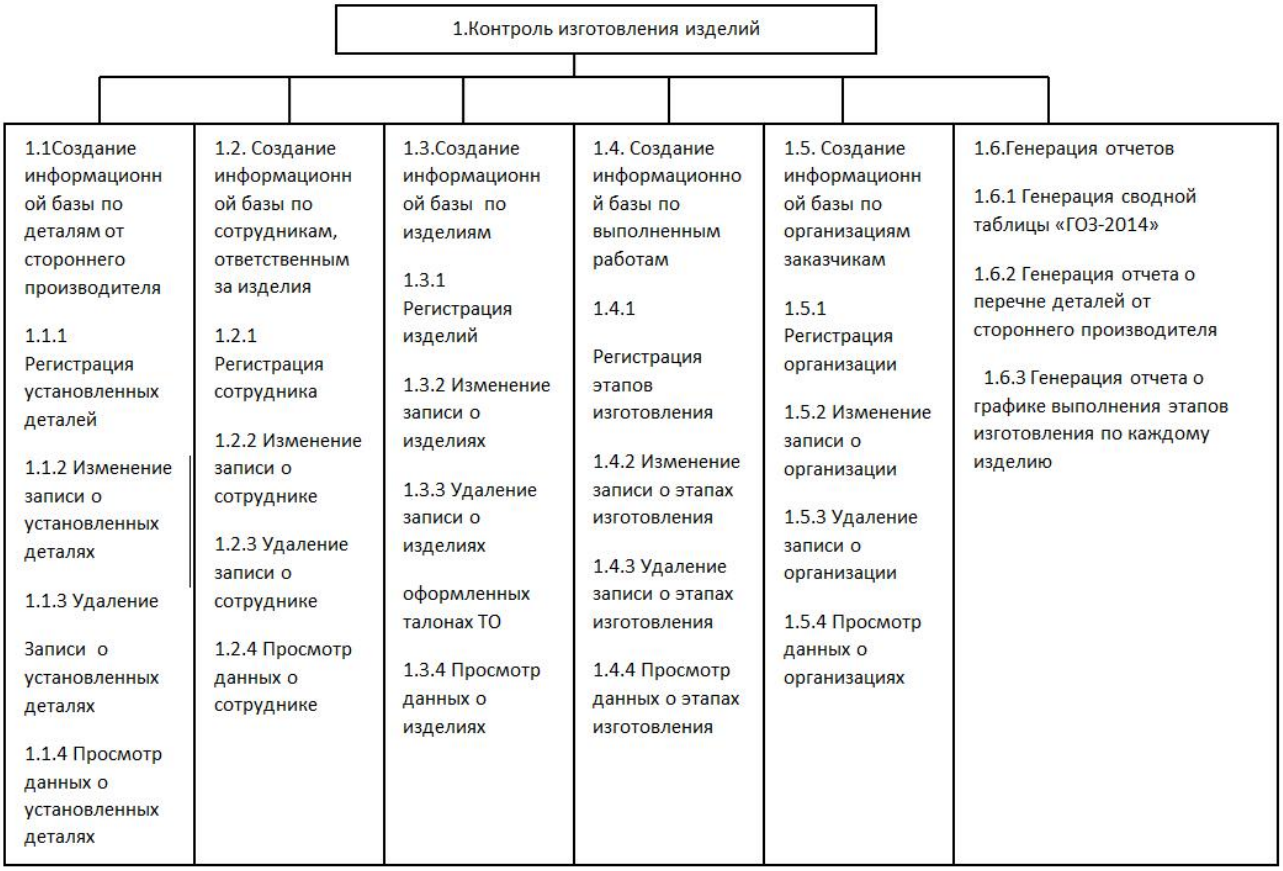


Рисунок **5.** – Иерархия функций приложения

Описание функций, которые должна выполнять система, представлено в таблице 2.

Таблица 2 – Функции системы

Уровень	Наименование функции	Описание функции
1.1	Создание информационной базы по деталям от стороннего производителя	Регистрация установленных деталей Изменение записи о установленных деталях Удаление записи о установленных деталях (DET, VID_DET, DORAB_DET,

Уровень	Наименование функции	Описание функции
		SOSTAV)
1.2	Создание информационной базы по сотрудникам, ответственным за изделия	Регистрация сотрудника Изменение записи о сотруднике Удаление записи о сотруднике (SOTR, OTD)
1.3	Создание информационной базы по изделиям	Регистрация изделий Изменение записи о изделиях Удаление записи о изделиях (IZD, TIP_IZD, PRIM)
1.4	Создание информационной базы по выполненным работам	Регистрация этапов изготовления Изменение записи о этапах изготовления Удаление записи о этапах изготовления (PROV, NALADKA, SBORKA, SDACHA)
1.5	Создание информационной базы по организациям заказчикам	Регистрация организации Изменение записи о организации Удаление записи о организации (ORG)
1.6	Генерация отчётов	Генерация сводной таблицы «ГОЗ» Генерация отчета о перечне деталей от стороннего производителя Генерация отчета о графике выполнения этапов изготовления по каждому изделию

Каждой выделенной функции поставлен в соответствие модуль. Описание списка функций и модулей представлено в таблице 3.

Таблица 3 – Списки функций и модулей

Уровень	Функция	Модуль
1.1	Создание информационной базы по деталям от стороннего производителя	Детали от стороннего производителя, Доработка детали, Вид детали
1.2	Создание информационной базы по сотрудникам, ответственным за изделия	Сотрудник, Отдел
1.3	Создание информационной базы по изделиям	Главная страница, Примечание, Тип изделия
1.4	Создание информационной базы по выполненным работам	Наладка, Сборка, Сдача, Проверка
1.5	Создание информационной базы по организациям заказчикам	Организация
1.6	Генерация отчётов	Модуль главной формы, Сводная таблица «ГОЗ», Перечень деталей от стороннего производителя, График выполнения этапов изготовления по каждому изделию

Разработанной схеме «функции-модули» сопоставлена реляционная модель базы данных. Описание фрагмента схемы модуля данных представлено в таблице 4.

Таблица 4 – Фрагмент схемы «Модули-данные»

Модуль	Таблица	Поле	Состояние колонки
Сведения о деталях от стороннего производителя	DET	N_DET N_DORAB NAME_DET N_VID	Чтение, Вставка, Удаление Чтение, Вставка, Редактирование, Удаление Чтение, Вставка, Редактирование, Удаление Чтение, Вставка, Редактирование, Удаление
Сведения о сотрудниках, ответственных за изделия	SOTR	N_SOTR F I O N_OTD	Чтение, Вставка, Удаление Чтение, Вставка, Редактирование, Удаление Чтение, Вставка, Редактирование, Удаление Чтение, Вставка, Редактирование, Удаление Чтение, Вставка, Редактирование, Удаление
Сведения о изделиях	IZD	N_IZD N_SOTR N_ORG N_TIP NAME_IZD	Чтение, Вставка, Удаление Чтение, Вставка, Редактирование, Удаление Чтение, Вставка, Редактирование, Удаление Чтение, Вставка, Редактирование, Удаление Чтение, Вставка, Редактирование, Удаление
Сведения о выполненных работах	PROV	N_PROV N_IZD SHASSIN SASSIK	Чтение, Вставка, Удаление Чтение, Вставка, Редактирование, Удаление Чтение, Вставка, Редактирование,

Модуль	Таблица	Поле	Состояние колонки
		PROGONN	Удаление
		PROGONK	Чтение, Вставка, Редактирование, Удаление
		DORABN	Чтение, Вставка, Редактирование, Удаление
		DORABK	Чтение, Вставка, Редактирование, Удаление
			Чтение, Вставка, Редактирование, Удаление
			Чтение, Вставка, Редактирование, Удаление
			Чтение, Вставка, Редактирование, Удаление
Сведения о организациях заказчиках	ORG	N_ORG	Чтение, Вставка, Удаление
		NAME_ORG	Чтение, Вставка, Редактирование, Удаление
		ADRES	Чтение, Вставка, Редактирование, Удаление
		DOGOV	Чтение, Вставка, Редактирование, Удаление

1.4 Разработка базы данных для автоматизированной системы контроля изготовления изделий специализированного назначения

1.4.1 Разработка логической модели данных системы

В процессе логического проектирования модели данных системы на основе результатов анализа предметной области выделено 14 сущностей:

– сотрудник;

– отдел;

– сборка;

– проверка;

– наладка;

– сдача;

– примечание;

– изделие;

– организация;

– деталь;

– доработка детали;

– тип изделия;

– вид детали;

– состав.

Описание сущностей представлено в таблице 5.

Таблица 5 – Описание сущностей логической модели данных системы

Сущность	Атрибут	Ключ	
Сотрудник	Номер сотрудника	PK	
	Фамилия		
	Имя		
	Отчество		
	Номер отдела		FK
Отдел	Номер отдела	PK	
	Наименование отдела		
	Код отдела		
Сборка	Номер сборки	PK	
	Номер изделия	PK	FK
	Сборка жгутов дата начала		
	Сборка жгутов дата окончания		
	Сборка каркаса дата начала		

Сущность	Атрибут	Ключ	
	Сборка каркаса дата окончания		
	Сборка приборов дата начала		
	Сборка приборов дата окончания		
	Предъявление отделу технического контроля дата начала		
	Предъявление отделу технического контроля дата окончания		
	Устранение замечаний дата начала		
	Устранение замечаний дата окончания		
Проверка	Номер проверки	PK	
	Номер изделия	PK	FK
	Проверка шасси дата начала		
	Проверка шасси дата окончания		
	Прогон дата начала		
	Прогон дата окончания		
	Доработка дата начала		
	Доработка дата окончания		
Наладка изделия	Номер наладки	PK	
	Номер изделия	PK	FK
	Наладка питания дата начала		
	Наладка питания дата окончания		
	Наладка автоматизированных систем дата начала		
	Наладка автоматизированных систем дата окончания		
	Наладка связи дата начала		
	Наладка связи дата окончания		
	Комплексная наладка дата начала		
	Комплексная наладка дата окончания		

Сущность	Атрибут	Ключ	
	Прогон изделия дата начала		
	Прогон изделия дата окончания		
	Дождевание дата начала		
	Дождевание дата окончания		
	Испытания в тепловой камере дата начала		
	Испытание в тепловой камере дата окончания		
	Наладка протоколов ЭТГ дата начала		
	Наладка протоколов ЭТГ дата окончания		
	Функционирование 72 часа дата начала		
	Функционирование 72 часа дата окончания		
	Навигация дата начала		
	Навигация дата окончания		
	Дальность дата начала		
	Дальность дата окончания		
	Специальные проверки дата начала		
	Специальные проверки дата окончания		
Сдача изделия	Номер сдачи	PK	
	Номер изделия	PK	FK
	ПРИ дата начала		
	ПРИ дата окончания		
	ПСИ дата начала		
	ПСИ дата окончания		
	Отгрузка дата начала		
	Отгрузка дата окончания		
Примечание	Номер примечания	PK	

Сущность	Атрибут	Ключ	
	Номер изделия	PK	FK
	Текст примечания		
Изделие	Номер изделия	PK	
	Наименование изделия		
	Номер сотрудника		FK
	Номер организации		FK
	Номер типа		FK
Организация	Номер организации	PK	
	Наименование организации		
	Адрес		
	Договор		
Тип изделия	Номер типа	PK	
	Наименование изделия код		
	Наименование изделия буквенное		
Деталь	Номер детали	PK	
	Наименование детали		
	Номер вида детали		FK
	Номер доработки детали		FK
Доработка детали	Номер доработки	PK	
	Наименование доработки		
Вид детали	Номер вида детали	PK	
	Наименование вида детали		
Состав	Номер состава	PK	
	Номер изделия	PK	FK
	Номер детали		FK

1.4.2 Разработка структуры связей

Между объектами предметной области существуют связи, которые отражаются в виде связей объектов логической модели. Графически связь обозначается линией, соединяющей связываемые объекты. В каждом направлении связи можно выделить главный объект, от которого идет связь, и подчиненный.

Различают:

– идентифицирующую связь;

– неидентифицирующую связь.

Идентифицирующая связь – это отношение между двумя сущностями, в котором 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. Атрибуты первичного ключа родительской сущности мигрируют в атрибуты подчиненной, чтобы стать там атрибутами первичного ключа.

Неидентифицирующая связь – это отношение между двумя сущностями, в котором каждый экземпляр подчиненной сущности не зависит от значений атрибутов родительской сущности. Это означает, что экземпляр подчиненной сущности может существовать без экземпляра родительской сущности. В не идентифицирующем отношении единственный экземпляр родительской сущности связан с множеством экземпляров подчиненной. Атрибуты первичного ключа родительской сущности мигрируют в подчиненную, чтобы стать там не ключевыми атрибутами [5].

В логической структуре данных системы между сущностями присутствуют связи мощностью только 1: N. Такая структура говорит о том, что, например, к одному типу детали может относится несколько различных деталей и т.п.

Параметры связей между сущностями логической модели данных автоматизированной системы контроля изготовления изделий специального назначения представлены в таблице 6.

Таблица 6 – Структура связей

Сущность-родитель	Сущность-потомок	Мощность связи	Тип связи
Отдел	Сотрудник	1 :N	Неидентифицирующая
Сотрудник	Изделие	1: N	Неидентифицирующая
Организация	Изделие	1 : N	Неидентифицирующая
Тип изделия	Изделие	1 : N	Неидентифицирующая
Изделие	Состав	1 : N	Идентифицирующая
Деталь	Доработка детали	1 : N	Неидентифицирующая
Вид детали	Деталь	1 : N	Неидентифицирующая
Деталь	Состав	1 : N	Неидентифицирующая
Изделие	Сборка	1 : N	Идентифицирующая
Изделие	Проверка	1 : N	Идентифицирующая
Изделие	Наладка	1 : N	Идентифицирующая
Изделие	Сдача	1 : N	Идентифицирующая
Изделие	Примечание	1 : N	Идентифицирующая

Ссылочная целостность для всех связей одинакова. Любые операции с дочерней сущностью никак не повлияют на родительскую сущность. При удалении или обновлении родительской сущности – дочерние сущности удаляются или обновляются каскадом, при добавлении новой записи в родительские сущности реакция отсутствует.

1.4.3 Нормализация отношений

Процесс нормализации выполняется путем анализа отношений с учетом их первичного ключа и существующих функциональных зависимостей. Он

включает ряд правил, которые используются для проверки отношений, чтобы вся база данных могла быть нормализована до желаемой степени нормализации. Если некоторое требование не удовлетворяется, то должна быть проведена декомпозиция отношения на отношения, каждое из которых удовлетворяет всем требованиям нормализации.

При работе с реляционными базами данных обязательным является удовлетворение только требованиям первой нормальной формы (1НФ). Все остальные формы нормализации могут использоваться по желанию проектировщиков. Однако рекомендуется выполнять нормализацию как минимум до 3НФ, чтобы получить хорошее качество отношений (приемлемую избыточность отношений).

Ненормализованное отношение характеризуется отношением, содержащим в одном или нескольких атрибутах, повторяющиеся группы данных – сложные атрибуты (массив или вектор значений).

Отношение, в котором на пересечении каждогокортежа и каждого атрибута содержится атомарное (или единственное) значение, находится в 1НФ. При этом необходимо, чтобы отношение имело первичный ключ. Для преобразования ненормализованного отношения в первую нормальную форму, следует найти в исходном отношении устранить все повторяющиеся группы данных, используя декомпозицию этого отношения.

Вторая нормальная форма применяется к отношениям с составными ключами, т.е. к таким отношениям, первичный ключ которых состоит из двух или больше атрибутов. Отношение с первичным ключом на основе единственного атрибута всегда находится в 2НФ.

Отношение, которое находится в 1НФ и каждый атрибут которого, не входящий в состав первичного ключа, зависит только от полного значения ключа и не зависит ни от какого отдельного атрибута, входящего в состав первичного ключа, имеет вторую нормальную форму (каждый неключевой атрибут функционально полно зависит от ключа).

Для преобразования отношения из 1НФ в 2НФ необходимо из исходного отношения удалить функционально зависимые атрибуты и поместить их в новое отношение вместе с копией детерминанта.

Отношение в 2НФ может иметь транзитивную зависимость своих атрибутов.

Отношение находится в 3НФ, если оно представлено в 2НФ и не имеет не входящих в первичный ключ атрибутов, которые находились бы в транзитивной функциональной зависимости друг от друга.

Нормализация 2НФ-отношения с образованием 3НФ-отношения осуществляется путем устранения транзитивных зависимостей – транзитивно-зависимые атрибуты удаляются из отношения и помещаются в новое отношение вместе с их детерминантом [6].

Все отношения логической модели данных автоматизированной системы контроля изготовления изделий специального назначения нормализованы, и находятся в третьей нормальной форме.

Все отношения логической модели данных автоматизированной системы контроля изготовления изделий специального назначения находятся в 1НФ, поскольку все атрибуты атомарны и каждое из данных отношений имеет первичный ключ.

Отношение «Сотрудник» имеет первичный ключ «Номер сотрудника» и атрибуты «Фамилия», «Имя», «Отчество», «Номер отдела», которые являются атомарными.

Отношение «Отдел» имеет первичный ключ «Номер отдела» и атрибуты «Наименование отдела», «Код отдела», которые являются атомарными.

Отношение «Сборка» имеет составной первичный ключ «Номер сборки», «Номер изделия» и атрибуты «Сборка жгутов дата начала», «Сборка жгутов дата окончания», «Сборка каркаса дата начала», «Сборка каркаса дата окончания», «Сборка приборов дата начала», «Сборка приборов дата окончания», «Предъявление отделу технического контроля дата начала», «Предъявление

отделу технического контроля дата окончания», «Устранение замечаний дата начала», «Устранение замечаний дата окончания», которые являются атомарными.

Отношение «Проверка» имеет составной первичный ключ «Номер проверки», «Номер изделия» и атрибуты «Проверка шасси дата начала», «Проверка шасси дата окончания», «Прогон дата начала», «Прогон дата окончания», «Доработка дата начала», «Доработка дата окончания», которые являются атомарными.

Отношение «Изделие» имеет первичный ключ «Номер изделия» и атрибуты «Номер сотрудника», «Номер организации», «Номер типа», «Наименование изделия», которые являются атомарными.

Отношение «Организация» имеет первичный ключ «Номер организации» и атрибуты «Наименование организации», «Адрес», «Договор», которые являются атомарными.

Отношение «Тип изделия» имеет первичный ключ «Номер типа» и атрибут «Наименование типа», который является атомарным.

Отношение «Наладка изделия» имеет составной первичный ключ «Номер наладки», «Номер изделия» и атрибуты «Наладка питания дата начала», «Наладка питания дата окончания», «Наладка автоматизированных систем дата начала», «Наладка автоматизированных систем дата окончания», «Наладка связи дата начала», «Наладка связи дата окончания», «Комплексная наладка дата начала», «Комплексная наладка дата окончания», «Прогон изделия дата начала», «Прогон изделия дата окончания», «Дождевание дата начала», «Дождевание датаокончания», «Испытания в тепловой камере дата начала», «Испытание в тепловой камере дата окончания», «Наладка протоколов ЭТГ дата начала», «Наладка протоколов ЭТГ дата окончания», «Функционирование 72 часа дата начала», «Функционирование 72 часа дата окончания», «Навигация дата начала», «Навигация дата окончания», «Дальность дата начала», «Дальность дата

окончания», «Специальные проверки дата начала», «Специальные проверки дата окончания», которые являются атомарными.

Отношение «Сдача изделия» имеет составной первичный ключ «Номер сдачи», «Номер изделия» и атрибуты «ПРИ дата начала», «ПРИ дата окончания», «ПСИ дата начала», «ПСИ дата окончания», «Отгрузка дата начала», «Отгрузка дата окончания», которые являются атомарными.

Отношение «Доработка детали» имеет первичный ключ «Номер доработки», и атрибут «Наименование доработки», который является атомарным.

Отношение «Вид детали» имеет первичный ключ «Номер вида детали» и атрибут «Наименование вида», который является атомарным.

Отношение «Деталь» имеет первичный ключ «Номер детали» и атрибуты «Наименование», «Номер доработки», «Номер вида детали», которые являются атомарными.

Отношение «Примечание» имеет составной первичный ключ «Номер примечания», «Номер изделия» и атрибут «Текст примечания», который является атомарным.

Отношение «Состав» имеет составной первичный ключ «Номер состава», «Номер изделия» и атрибут «Номер детали», который является атомарным.

Отношения «Сотрудник», «Отдел», «Изделие», «Организация», «Тип изделия», «Доработка детали», «Деталь», «Вид детали» находятся во 2 НФ, так как не имеют составных первичных ключей.

В отношении «Состав» первичный ключ состоит из атрибутов «Номер состава» и «Номер изделия». В данном отношении есть неключевой атрибут «Номер детали», который функционально полно зависит от первичного ключа в целом. Можно сделать вывод, что данное отношение находится во 2НФ.

В отношении «Сборка» первичный ключ состоит из атрибутов «Номер сборки» и «Номер изделия». В данном отношении есть неключевые атрибуты «Сборка жгутов дата начала», «Сборка жгутов дата окончания», «Сборка каркаса дата начала», «Сборка каркаса дата окончания», «Сборка приборов дата начала»,

«Сборка приборов дата окончания», «Предъявление отделу технического контроля дата начала», «Предъявление отделу технического контроля дата окончания», «Устранение замечаний дата начала», «Устранение замечаний дата окончания», которые функционально полно зависят от первичного ключа в целом. Можно сделать вывод, что данное отношение находится во 2НФ.

В отношении «Проверка» первичный ключ состоит из атрибутов «Номер проверки» и «Номер изделия». В данном отношении есть неключевые атрибуты «Проверка шасси дата начала», «Проверка шасси дата окончания», «Прогон дата начала», «Прогон дата окончания», «Доработка дата начала», «Доработка дата окончания», которые функционально полно зависят от первичного ключа в целом. Можно сделать вывод, что данное отношение находится во 2НФ.

В отношении «Наладка изделия» первичный ключ состоит из атрибутов «Номер наладки» и «Номер изделия». В данном отношении есть неключевые атрибуты «Наладка питания дата начала», «Наладка питания дата окончания», «Наладка автоматизированных систем дата начала», «Наладка автоматизированных систем дата окончания», «Наладка связи дата начала», «Наладка связи дата окончания», «Комплексная наладка дата начала», «Комплексная наладка дата окончания», «Прогон изделия дата начала», «Прогон изделия дата окончания», «Дождевание дата начала», «Дождевание дата окончания», «Испытания в тепловой камере датаначала», «Испытание в тепловой камере дата окончания», «Наладка протоколов ЭТГ дата начала», «Наладка протоколов ЭТГ дата окончания», «Функционирование 72 часа дата начала», «Функционирование 72 часа дата окончания», «Навигация дата начала», «Навигация дата окончания», «Дальность дата начала», «Дальность дата окончания», «Специальные проверки дата начала», «Специальные проверки дата окончания», которые функционально полно зависят от первичного ключа в целом.Можно сделать вывод, что данное отношение находится во 2НФ.

В отношении «Сдача изделия» первичный ключ состоит из атрибутов «Номер сдачи» и «Номер изделия». В данном отношении есть неключевые

атрибуты «ПРИдата начала», «ПРИ дата окончания», «ПСИ дата начала», «ПСИ дата окончания», «Отгрузка дата начала», «Отгрузка дата окончания», которые функционально полно зависят от первичного ключа в целом. Можно сделать вывод, что данное отношение находится во 2НФ.

В отношении «Примечание» первичный ключ состоит из атрибутов «Номер примечания» и «Номер изделия». В данном отношении есть неключевой атрибут «Текст примечания», который функционально полно зависит от первичного ключа в целом. Можно сделать вывод, что данное отношение находится во 2НФ.

Таким образом, логическая модель данных разрабатываемой системы находится во 2 НФ.

Ни в одном из отношений логической модели данных не существует транзитивных зависимостей, т.е. не ключевые атрибуты не зависят функционально друг от друга, поэтому отношения находятся в 3НФ.

Отношение «Сотрудник» имеет не ключевые атрибуты «Фамилия», «Имя», «Отчество», «Номер отдела», которые не зависят функционально друг от друга.

Отношение «Отдел» имеет не ключевые атрибуты «Наименование отдела», «Код», которые не зависят функционально друг от друга.

Отношение «Изделие» имеет не ключевые атрибуты «Номер сотрудника», «Номер организации», «Номер типа», «Наименование изделия», которые не зависят функционально друг от друга.

Отношение «Организация» имеет не ключевые атрибуты «Наименование организации», «Расположение», «Договор», которые не зависят функционально друг от друга.

Отношение «Тип изделия» имеет не ключевые атрибуты «Наименование цифр», «Наименование буквенное», которые не зависят функционально друг от друга.

Отношение «Доработка детали» имеет один не ключевой атрибут «Наименование доработки».

Отношение «Вид детали» имеет один не ключевой атрибут «Наименование вида».

Отношение «Состав» имеет один не ключевой атрибут «Номер детали».

Отношение «Сборка» имеет не ключевые атрибуты «Сборка жгутов дата начала», «Сборка жгутов дата окончания», «Сборка каркаса дата начала», «Сборка каркаса дата окончания», «Сборка приборов дата начала», «Сборка приборов дата окончания», «Предъявление отделу технического контроля дата начала», «Предъявление отделу технического контроля дата окончания», «Устранение замечаний дата начала», «Устранение замечаний дата окончания», которые не зависят функционально друг от друга.

Отношение «Деталь» имеет не ключевые атрибуты «Наименование детали», «Номер доработки», «Номер вида детали», которые не зависят функционально друг от друга.

Отношение «Проверка» имеет не ключевые атрибуты «Проверка шасси дата начала», «Проверка шасси дата окончания», «Прогон дата начала», «Прогон дата окончания», «Доработка дата начала», «Доработка дата окончания», которые не зависят функционально друг от друга.

Отношение «Наладка» имеет не ключевые атрибуты «Наладка питания дата начала», «Наладка питания дата окончания», «Наладка автоматизированных систем дата начала», «Наладка автоматизированных систем дата окончания», «Наладка связи дата начала», «Наладка связи дата окончания», «Комплексная наладка дата начала», «Комплексная наладка дата окончания», «Прогон изделия дата начала», «Прогон изделия дата окончания», «Дождевание дата начала», «Дождевание дата окончания», «Испытания в тепловой камере дата начала», «Испытание в тепловой камере дата окончания», «Наладка протоколов ЭТГ дата начала», «Наладка протоколов ЭТГ дата окончания», «Функционирование 72 часа дата начала», «Функционирование 72 часа дата окончания», «Навигация дата начала», «Навигация дата окончания», «Дальность дата начала», «Дальность

дата окончания», «Специальные проверки дата начала», «Специальные проверки дата окончания», которые не зависят функционально друг от друга.

Отношение «Сдача» имеет не ключевые атрибуты «ПРИдата начала», «ПРИ дата окончания», «ПСИ дата начала», «ПСИ дата окончания», «Отгрузка дата начала», «Отгрузка дата окончания», которые не зависят функционально друг от друга.

Отношение «Примечание» имеет один не ключевой атрибут «Текст примечания».

Таким образом, логическая модель данных разрабатываемой системы находится в 3 НФ, что достаточно для реализации базы данных и обеспечения нормальной ее работоспособности.

Логическая модель данных автоматизированной системы контроля изготовления изделий специального назначения представлен на рисунке 7.

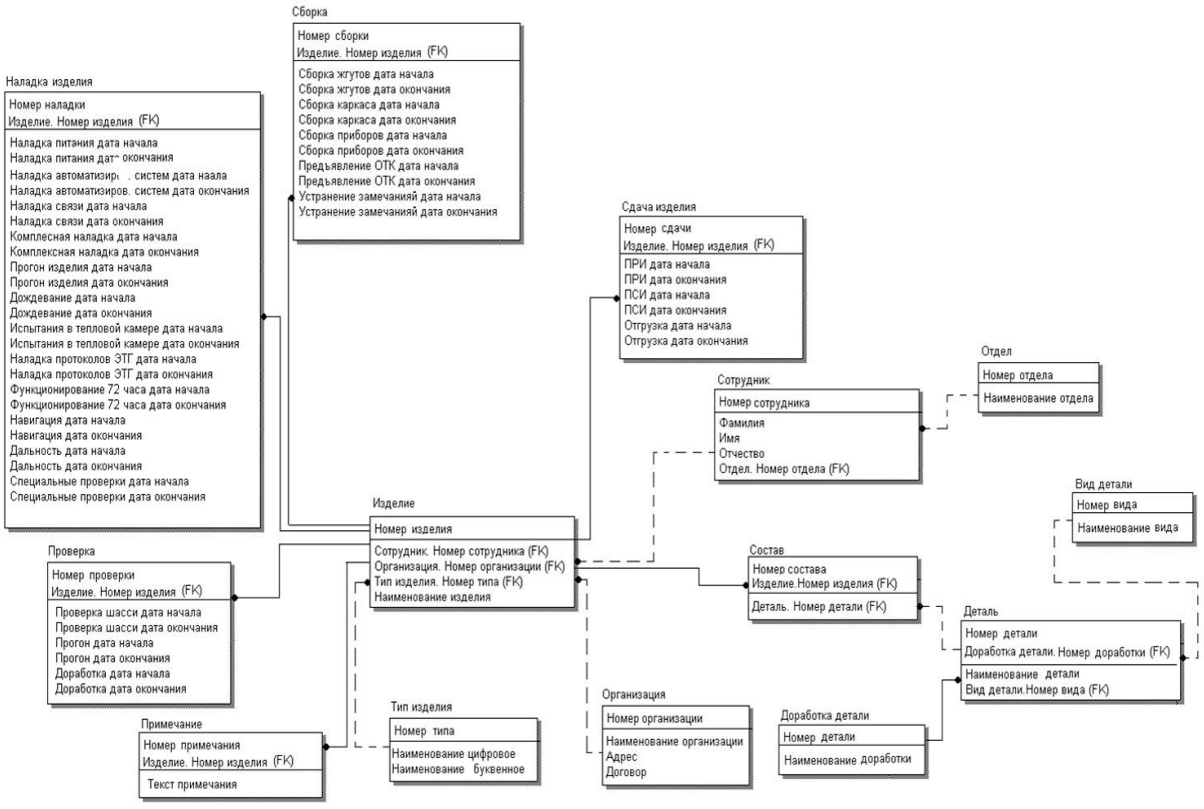


Рисунок 7 – Логическая модель данных

1.4.4 Разработка физической модели данных системы

Построение физической модели данных автоматизированной системы контроля изготовления изделий специального назначения осуществлена путем автоматического перехода от логической модели к физической модели, т.к. при создании логической модели данных системы был выбран логико – физический тип модели для ее реализации в СУБД PostgresSQL.

В процессе физического проектирования каждой сущности логической модели данных системы поставлена в соответствие таблица, каждому атрибуту – поле таблицы.

Соответствие таблиц физической модели сущностям логической представлено в таблице 7.

Таблица 7 – Соответствие таблиц физической модели данных сущностям

Имя сущности	Имя таблицы
Наладка изделия	NALADKA
Проверка	PROV
Примечание	PRIM
Сборка	SBORKA
Изделие	IZD
Тип изделия	TIP_IZD
Сдача изделия	SDACHA
Сотрудник	SOTR
Отдел	OTD
Состав	SOSTAV
Организация	ORG
Деталь	DET
Вид детали	VID_DET
Доработка детали	DORAB_DET

Описание таблиц физической модели данных представлено в таблице 8.

Таблица 8 – Описание таблиц физической модели данных

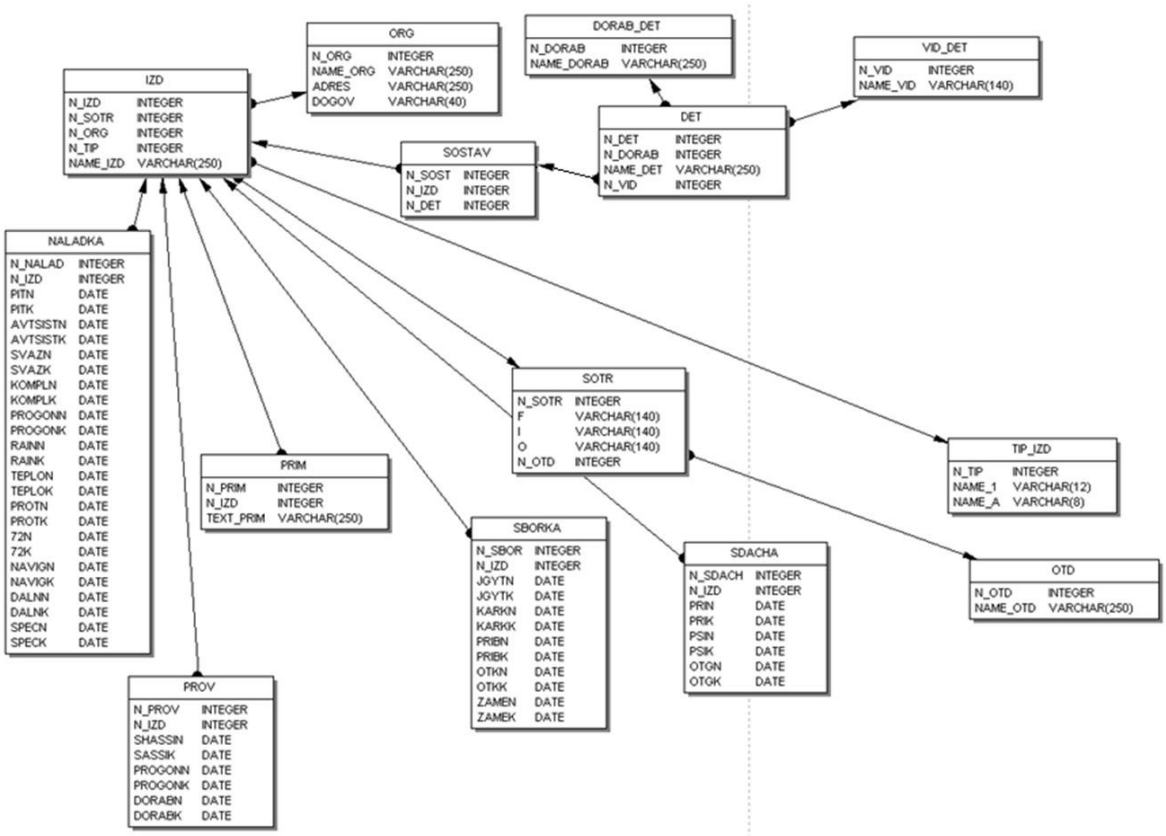
Таблица	Поля таблицы	Тип данных	Ключ	
SOTR	PK_SOTR	Integer	PK	
	F	Varchar(140)		
	I	Varchar(140)		
	O	Varchar(140)		
	N_OTD	Integer		
OTDEL	PK_OTD	Integer	PK	
	NAME_OTD	Varchar(250)		
	KOD	Integer		
IZDEL	PK_IZD	Integer	PK	
	N_SOTR	Integer		FK
	N_ORG	Integer		FK
	N_TIP	Integer		FK
	NAME_IZD	Varchar(250)		
ORG	PK_ORG	Integer	PK	
	NAME	Varchar(250)		
	ADRES	Varchar(250)		
	DOG	Varchar(40)		
TIPIZD	PK_TIZD	Integer	PK	
	NAME_C	Varchar(12)		
	NAME_B	Varchar(8)		
DETAL	PK_DET	Integer	PK	
	N_VDET	Integer		FK

Таблица	Поля таблицы	Тип данных	Ключ	
	N_DDET	Integer		FK
	NAME_DET	Varchar(250)		
DORABDET	PK_DDET	Integer	PK	
	NAME_DORAB	Varchar(250)		
VIDDET	PK_VDET	Integer	PK	
	NAME_VID	Varchar(100)		
SOSTAV	PK_SOST	Integer	PK	
	N_IZD	Integer		FK
	N_DET	Integer		FK
SBORKA	N_SBOR	Integer	PK	
	N_IZD	Integer		FK
	JGYTN	Date		
	JGETK	Date		
	KARKN	Date		
	KARKK	Date		
	PRIBN	Date		
	PRIBK	Date		
	OTKN	Date		
	OTKK	Date		
	ZAMEN	Date		
	ZAMEK	Date		
PROVER	N_PROV	Integer	PK	
	N_IZD	Integer		FK
	SHASSIN	Date		
	SHASSIK	Date		
	PROGONN	Date		

Таблица	Поля таблицы	Тип данных	Ключ	
	PROGONK	Date		
	DORABN	Date		
	DORABK	Date		
NALADKA	N_NALAD	Integer	PK	
	N_IZD	Integer		FK
	PITN	Date		
	PITK	Date		
	AVTN	Date		
	AVTK	Date		
	SVAZN	Date		
	SVAZK	Date		
	KOMPLN	Date		
	KOMPLK	Date		
	PROGONN	Date		
	PROGONK	Date		
	DOGDN	Date		
	DOGDK	Date		
	TEPLON	Date		
	TEPLOK	Date		
	ETGN	Date		
	ETGK	Date		
	72HOUERN	Date		
	72HOUERK	Date		
	NAVIGN	Date		
	NAVIGK	Date		
	DALNN	Date		

Таблица	Поля таблицы	Тип данных	Ключ	
	DALNK	Date		
	SPECN	Date		
	SPECK	Date		
SDACHA	N_SD	Integer	PK	
	N_IZD	Integer		FK
	PRIN	Date		
	PRIK	Date		
	PSIN	Date		
	PSIK	Date		
	OTKN	Date		
	OTKK	Date		
	ZAKN	Date		
	ZAKK	Date		
PRIM	N_PRIM	Integer	PK	
	N_IZD	Integer		FK
	TEXT	Vrchar(250)		

Построена физическая модель данных приведена на рисунке



8.

Рисунок 8 – Физическая модель данных

1.5 Описание программы

1.5.1 Общие сведения

Программное обеспечение автоматизированной системы контроля изготовления изделий специального назначения разработано кроссплатформенная свободная IDE QTCreator. База данных реализована с использованием СУБД PostgresSQL.

1.5.2 Описание логической структуры

Программа разработана в среде QtCreator. В процессе разработки автоматизированной системы были созданы модуль главной формы MainWindow.cpp, модуль данных Data.cpp, модули для работы с данными об изделии: MainWindow.cpp, Tizd.cpp, Org.cpp, Prim.cpp;модули для работы с данными о сотруднике ответственном за изделие:Sotr.cpp, Otd.cpp; модули для

работы с данными о деталях от стороннего производителя: Det.cpp, DD.cpp, Vdet.cpp, Sost.cpp; модуля для работы с данными о изготовлении изделий: Nalad.cpp, Prov.cpp, Sbor.cpp, Sd.cpp.

Иерархия модулей представлена на рисунке 9.

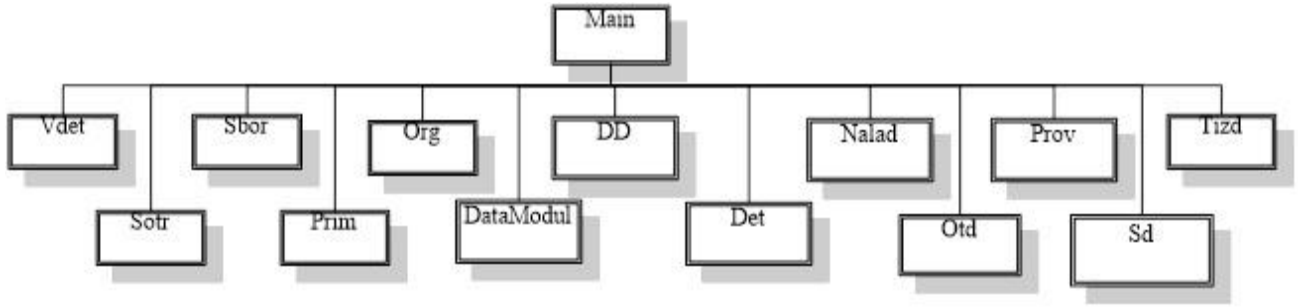


Рисунок 9 – Иерархия модулей программы

UML-диаграмму классов, демонстрирующая общую структуру иерархии классов, представлена на рисунке 10.

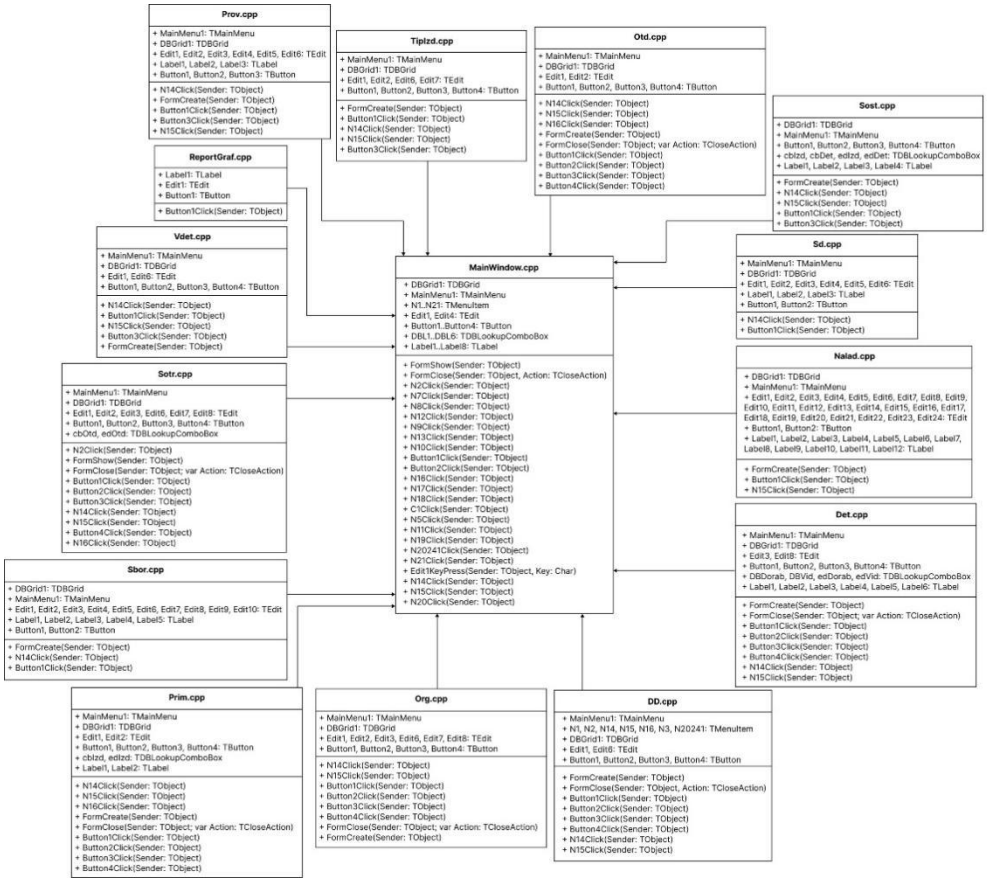


Рисунок 10 – UML-диаграмма классов

Листинг программы приведён в приложении Б.

Описание обработчиков событий и действий и их функции представлены в таблице 9.

Таблица 9 – Структура программы

Модуль	Функции	Назначение
MainWindow	voidFormShow(Sender:TObject)	Установление соединения с базой данных
	voidFormClose(Sender:TObject	Закрытие соединения с базой данных
	void N2Click(Sender:TObject)	Закрытие формы «Главная страница», выход из программы
	oid N7Click(Sender:TObject)	Создание и открытие формы «Сотрудник»
	void N8Click(Sender:TObject)	Создание и открытие формы «Отдел»
	void N12Click(Sender:TObject)	Создание и открытие формы «Деталь»
	void N9Click(Sender:TObject)	Создание и открытие формы «Организация»
	void N13Click(Sender:TObject)	Создание и открытие формы «Доработка детали»
	void N10Click(Sender:TObject)	Создание и открытие формы «Тип изделия»
	void Button2Click(Sender:TObject)	Отмена ввода новой записи
	void Button1Click(Sender:TObject)	Добавление новой записи в таблицу
	void N16Click(Sender:TObject)	Создание и открытие формы «Примечание»
	void N17Click(Sender:TObject)	Создание и открытие формы «Проверка изделия»

Модуль	Функции	Назначение
	void N18Click(Sender:TObject)	Создание и открытие формы «Сборка изделия»
	void C1Click(Sender:TObject)	Создание и открытие формы «Сдача изделия»
	void N5Click(Sender:TObject)	Создание и открытие формы «Состав»
	void N11Click(Sender:TObject)	Создание и открытие формы «Вид детали»
	void N19Click(Sender:TObject)	Отображение полей для добавления новой записи
	void N20141Click(Sender:TObject)	Формирование отчета «ГОЗ»
	void N21Click(Sender:TObject)	Создание и открытие формы «Наладка»
	void Edit1KeyPress(Sender:TObject	Проверка корректности ввода
	void N14Click(Sender:TObject)	Формирование отчета «Перечень деталей от стороннего производителя»
	void N15Click(Sender:TObject)	Создание и открытие формы «График выполнения работ по изделию»
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TStart.Edit1KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
DD	voidFormCreate(Sender:TObject)	Установление соединения с базой данных, заполнение полей формы
	voidFormClose(Sender:TObject	Закрытие соединения с базой данных
	void Button1Click(Sender:TObject)	Добавление записи в таблицу

Модуль	Функции	Назначение
	void Button2Click(Sender:TObject)	Отмена ввода новой записи в таблицу
	void Button3Click(Sender:TObject)	Изменение записи в таблице
	void Button4Click(Sender:TObject)	Отмена ввода изменений в таблицу
	void N14Click(Sender:TObject)	Активация отображения полей для ввода новой записи
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
	void TStart.Edit1KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
	void TStart.Edit2KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
Otd	void N14Click(Sender:TObject)	Активация отображения полей для ввода новой записи
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	voidFormCreate(Sender:TObject)	Установление соединения с базой данных, заполнение полей формы
	voidFormClose(Sender:TObject	Закрытие соединения с базой данных
	void Button1Click(Sender:TObject)	Добавление записи в таблицу
	void Button2Click(Sender:TObject)	Отмена ввода новой записи в таблицу
	void Button3Click(Sender:TObject)	Изменение записи в таблице

Модуль	Функции	Назначение
	void Button4Click(Sender:TObject)	Отмена ввода изменений в таблицу
	Void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
	void TStart.Edit1KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
	void TStart.Edit2KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
Org	oid N14Click(Sender:TObject)	Активация отображения полей для ввода новой записи
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	void Button1Click(Sender:TObject)	Добавление записи в таблицу
	void Button2Click(Sender:TObject)	Отмена ввода новой записи в таблицу
	void Button3Click(Sender:TObject)	Изменение записи в таблице
	void Button4Click(Sender:TObject)	Отмена ввода изменений в таблицу
	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
	void TStart.Edit1KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле

Модуль	Функции	Назначение
	void TStart.Edit2KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
	void TStart.Edit3KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
Det	oid N14Click(Sender:TObject)	Активация отображения полей для ввода новой записи
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	void Button1Click(Sender:TObject)	Добавление записи в таблицу
	void Button2Click(Sender:TObject)	Отмена ввода новой записи в таблицу
	void Button3Click(Sender:TObject)	Изменение записи в таблице
	void Button4Click(Sender:TObject)	Отмена ввода изменений в таблицу
	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
	void TStart.Edit1KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
Nalad	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных
	void N15Click(Sender:TObject)	Активация отображения полей для

Модуль	Функции	Назначение
		ввода изменений в базу данных
	voidButton1Click(Sender:TObject)	Изменение записи в таблице
	voidButton2Click(Sender:TObject)	Отмена ввода изменений в таблицу
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
Prim	oid N14Click(Sender:TObject)	Активация отображения полей для ввода новой записи
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	void Button1Click(Sender:TObject)	Добавление записи в таблицу
	void Button2Click(Sender:TObject)	Отмена ввода новой записи в таблицу
	void Button3Click(Sender:TObject)	Изменение записи в таблице
	void Button4Click(Sender:TObject)	Отмена ввода изменений в таблицу
	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
	void TStart.Edit1KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле

Модуль	Функции	Назначение
Prov	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	voidButton1Click(Sender:TObject)	Изменение записи в таблице
	voidButton2Click(Sender:TObject)	Отмена ввода изменений в таблицу
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
Sbor	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	voidButton1Click(Sender:TObject)	Изменение записи в таблице
	voidButton2Click(Sender:TObject)	Отмена ввода изменений в таблицу
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
Sd	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных

Модуль	Функции	Назначение
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	voidButton1Click(Sender:TObject)	Изменение записи в таблице
	voidButton2Click(Sender:TObject)	Отмена ввода изменений в таблицу
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
Sost	oid N14Click(Sender:TObject)	Активация отображения полей для ввода новой записи
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	void Button1Click(Sender:TObject)	Добавление записи в таблицу
	void Button2Click(Sender:TObject)	Отмена ввода новой записи в таблицу
	void Button3Click(Sender:TObject)	Изменение записи в таблице
	void Button4Click(Sender:TObject)	Отмена ввода изменений в таблицу
	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
Sotr	oid N14Click(Sender:TObject)	Активация отображения полей для ввода новой записи
	void N15Click(Sender:TObject)	Активация отображения полей для

Модуль	Функции	Назначение
		ввода изменений в базу данных
	void Button1Click(Sender:TObject)	Добавление записи в таблицу
	void Button2Click(Sender:TObject)	Отмена ввода новой записи в таблицу
	void Button3Click(Sender:TObject)	Изменение записи в таблице
	void Button4Click(Sender:TObject)	Отмена ввода изменений в таблицу
	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
	void TStart.Edit1KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
	void TStart.Edit2KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
	void TStart.Edit3KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
	void TStart.Edit4KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
	void TStart.Edit5KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
	void TStart.Edit6KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
TipIzd	void N14Click(Sender:TObject)	Активация отображения полей для ввода новой записи

Модуль	Функции	Назначение
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	void Button1Click(Sender:TObject)	Добавление записи в таблицу
	void Button2Click(Sender:TObject)	Отмена ввода новой записи в таблицу
	void Button3Click(Sender:TObject)	Изменение записи в таблице
	void Button4Click(Sender:TObject)	Отмена ввода изменений в таблицу
	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
Vdet	oid N14Click(Sender:TObject)	Активация отображения полей для ввода новой записи
	void N15Click(Sender:TObject)	Активация отображения полей для ввода изменений в базу данных
	void Button1Click(Sender:TObject)	Добавление записи в таблицу
	void Button2Click(Sender:TObject)	Отмена ввода новой записи в таблицу
	void Button3Click(Sender:TObject)	Изменение записи в таблице
	void Button4Click(Sender:TObject)	Отмена ввода изменений в таблицу
	voidFormClose(Sender:TObject	Установление соединения с базой данных, заполнение полей формы
	voidFormCreate(Sender:TObject)	Закрытие соединения с базой данных

Модуль	Функции	Назначение
	void TfmPrim.N16Click(Sender: TObject);	Удаление записи
	void TfmPrim.N20141Click(Sender: TObject);	Формирование отчета «ГОЗ»
	void TfmPrim.N2Click(Sender: TObject);	Закрытие формы
	void TStart.Edit1KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле
ReportGraf	void Button1Click(Sender:TObject);	Формирование отчета «График выполнения работ по изделию»
	void TStart.Edit1KeyPress(Sender: TObject; var Key: Char);	Проверка правильности ввода в поле

С помощью инструмента PostgresSQL была создана база данных. При этом в качестве сервера базы данных использовался локальный сервер с именем localhost (локальный сервер).

Были созданы 14 таблиц: «NALADKA», «SBORKA», «PROV», «PRIM», «IZD», «TIP_IZD», «SDACHA», «SOTR», «SOSTAV», «OTD», «VID_DET», «DET», «DORAB_DET», «ORG».

Таблицы «AVTO» и «COLOR» соединены между собой внешним ключом N_COLOR.

Таблицы «IZD» и «PRIM» соединены между собой внешним ключом N_IZD.

Таблицы «IZD» и «PROV» соединены между собой внешним ключом N_IZD.

Таблицы «IZD» и «NALADKA» соединены между собой внешним ключом N_IZD.

Таблицы «IZD» и «SBORKA» соединены между собой внешним ключомN_IZD.

Таблицы «IZD» и «SDACHA» соединены между собой внешним ключом N_IZD.

Таблицы «IZD» и «SOSTAV» соединены между собой внешним ключом N_IZD.

Таблицы «SOTR» и «IZD» соединены между собой внешним ключом N_SOTR.

Таблицы «ORG» и «IZD» соединены между собой внешним ключом N_ORG.

Таблицы «TIP_IZD» и «IZD» соединены между собой внешним ключом N_TIP.

Таблицы «DET» и «SOSTAV» соединенымеждусобойвнешнимключом N_DET.

Таблицы «OTD» и «SOTR» соединены между собой внешним ключом N_OTD.

Таблицы «DORAB_DET» и «DET» соединены между собой внешним ключом N_DORAB.

Таблицы «VID_DET» и «DET» соединены между собой внешним ключом N_VID.

Для всех внешних ключей было установлено правило каскадного удаления и обновления данных, которое влечет за собой удаление/обновление во всех строках дочерних таблиц при удалении/обновлении соответствующих строк в родительских таблицах.

В процессе разработки программного обеспечения автоматизированной системы были написаны следующие запросы:

– запрос для формирования сводной таблицы «Изготовление серии ГОЗ»;

– запрос для формирования списка деталей от стороннего производителя;

– запрос для формирования графика выполнения работ по каждому изделию;

– тексты запросов приведены в приложении В.

1.5.3 Входные данные

Входными данными для программного обеспечения являются:

– информация о сотруднике ответственном за изделие;

– информация оизделии;

– информация о деталях от стороннего производителя;

– информация оэтапах сборки;

– информация оэтапах проверки;

– информация оэтапах наладки;

– информация оэтапах сдачи;

– информация о примечаниях.

1.5.4 Выходные данные

Выходными данными для программного обеспечения являются:

– сводная таблица «ГОЗ»;

– перечень деталей от стороннего производителя;

– график выполнения работ по каждому изделию.

1.6 Тестирование

1.6.1 Объект испытаний

Объектом испытаний является программное обеспечение автоматизированной системы контроля изготовления изделий специального назначения.

1.6.2 Цель испытаний

Цель испытания программного обеспечения состоит в проверке его функциональных возможностей, соответствия требованиям технического задания и правильности решения поставленной задачи при любых условиях.

1.6.3 Требования к программе

Программное обеспечение должно работать корректно: организация диалога должна предусматривать защиту от ввода некорректных данных.

1.6.4 Методы испытаний

Испытание программы должно проводиться на основе тестовых примеров, приведенных в таблице 10.

Таблица 10 – Методы испытаний

Методы испытаний	Ожидаемый результат
Добавить отдел – контролирующий	В таблице «Отдел» должна добавиться новая запись – контролирующий
Изменить отдел на проверяющий	В таблице «Отдел» должен измениться отдел на проверяющий
Добавить вид детали – корпус	В таблице «Вид детали» должна добавиться новая запись – корпус
Изменить вид детали на корпус тип 2	В таблице «Вид детали» должен измениться вид детали на корпус тип 2
Добавить доработку детали – шлифофовка	В таблице «Доработка детали» должна добавиться новая запись – шлифофовка
Изменить доработку детали на шлифовка	В таблице «Доработка детали» должна измениться доработка детали на шлифовка
Добавить новую деталь: доработка детали – шлифовка, вид детали – связь, наименование – блок – 312	В таблице «Деталь» должна добавиться запись доработка детали – шлифовка, вид детали – связь, наименование – блок – 312
Изменить деталь на блок-А12	В таблице «Деталь» должна измениться деталь на блок-А12
Добавить нового сотрудника: фамилия – Евстигнеев, имя – Иван, отчество – Семёнович, отдел – проверяющий	В таблицу «Сотрудник» должна добавиться новая запись: фамилия – Евстигнеев, имя – Иван, отчество – Семёнович, отдел – проверяющий
Изменить для сотрудника Евстигнеев Иван Семёнович имя на Александр	В таблице «Сотрудник» для сотрудника Евстигнеев Иван Семёнович должно измениться имя на Александр
Добавить новую организацию:	В таблицу «Организация» должна

Методы испытаний	Ожидаемый результат
наименование организации – Звезда, адрес – Москва, Покровский пер., д.8, договор – 112233100	добавиться новая запись: наименование организации – Звезда, адрес – Москва, Покровский пер., д.8, договор – 112233100
Изменить для организации звезда договор на 10010033	В таблице «Организация» для организации Звезда договор должен измениться на 10010033
Добавить новый тип изделия: наименование цифровое – 25, наименование буквенное – РТ	В таблицу «Тип изделия» должна добавиться новая запись: наименование цифровое – 25, наименование буквенное – РТ
Изменить для типа изделия 25 буквенное наименование на МРТ	В таблице «Тип изделия» для типа изделия 25 буквенное наименование должно измениться на МРТ
Добавить новое изделие: сотрудник – Евстигнеев, организация – Звезда, тип изделия – 25, наименование изделия – 9Д156СК	В таблицу «Изделие» должна добавиться запись: сотрудник – Евстигнеев, организация – Звезда, тип изделия – 25, наименование изделия – 9Д156СК; в таблицу «Проверка» должна добавиться запись с номером изделия; в таблицу «Наладка изделия» должна добавиться запись с номером изделия, в таблицу «Сборка» должна добавиться запись с номером изделия, в таблицу «Сдача» должна добавиться запись с номером изделия
Изменить для изделия 9Д156СК фамилию сотрудника на Иванов	В таблице «Изделие» для изделия 9Д156СК фамилия сотрудника должна измениться на Иванов
Добавить новое вхождение в состав: номер	В таблицу «Состав» должна добавиться

Методы испытаний	Ожидаемый результат
изделия – 4, деталь – двигатель	запись: номер изделия – 4, номер детали – 2
Изменить вхождение в состав деталь на блок связи	В таблице «Состав» номер детали должен изменится на 1
Добавить примечание для изделия 4: текст примечания – затянуть все болты, проверить ремни.	В таблицу «Примечание» должна добавиться запись: текст примечания – затянуть все болты, проверить ремни.
Изменить примечание затянуть все болты, проверить ремни на затянуть все болты	В таблице «Примечание» текст примечания должен измениться на затянуть все болты.
Изменить проверку изделия 4: Проверка шасси, дата начала – 12.05.2024	В таблице «Проверка» проверка шасси, дата начала должна измениться на 12.05.2024
Изменить наладку изделия для изделия 4: наладка питания, дата начала – 13.05.2024, наладка питания, дата окончания – 13.05.2024	В таблице «Наладка изделия» наладка питания, дата начала должна измениться на 13.05.2024, наладка питания, дата окончания изменится на 13.05.2024
Изменить сборку изделия 4: Сборка жгутов, дата начала – 14.05.2024	В таблице «Сборка» сборка жгутов, дата начала должна измениться на 14.05.2024
Изменить сдачу изделия 4: ПРИ, дата начала – 15.05.2024	В таблице «Сдача изделия» ПРИ, дата начала должна измениться на 15.05.2024
Сформировать отчет «Изготовление серии «ГОЗ»»	Должен быть сформирован отчет «Изготовление серии «ГОЗ», содержащий данные о изделиях и этапах изготовления
Сформировать отчет «Перечень деталей от стороннего производителя»	Должен быть сформирован отчет «Перечень деталей от стороннего производителя», содержащий данные о деталях от стороннего производителя установленные на изделия
Сформировать отчет «График выполнения работ по каждому изделию»	Должен быть сформирован отчет «График выполнения работ по каждому изделию», содержащий данные о выполнении наладки,

Методы испытаний	Ожидаемый результат
	проверки, сборки, сдачи каждого изделия

1.6.5 Результаты испытаний системы

Результаты испытания программного обеспечения системы приведены в приложении Д.

Результаты испытаний показали, что программное обеспечение автоматизированной системы контроля изготовления изделий специального назначения работает верно и соответствует требованиям технического задания.

1.7 Описание применения

1.7.1 Назначение применения

Программное обеспечение автоматизированной системы контроля изготовления изделий специального назначения предназначено для автоматизации работы сотрудников НТЦ при оформлении сводной таблицы «ГОЗ», хранения и обработки оперативной информации.

1.7.2 Описание задачи

Работа с программным обеспечением должна осуществляться путем запуска исполняемого файла start.exe. В результате на экране отобразится «Главная страница», которая позволяет работать с даннымио изделиях, сотрудниках, организациях, этапах изготовления изделий, деталях от стороннего производителя и отчетной документации.

На форме «Главная страница» отображаются сведения о изделиях. При выборе меню «Файл» и пункта меню «Новое изделие» в правой части экрана будут доступны поля для ввода данных (рисунок 11). При выборе пункта меню «Изменить данные» в правой части формы будут доступны поля для изменения данных о изделии.

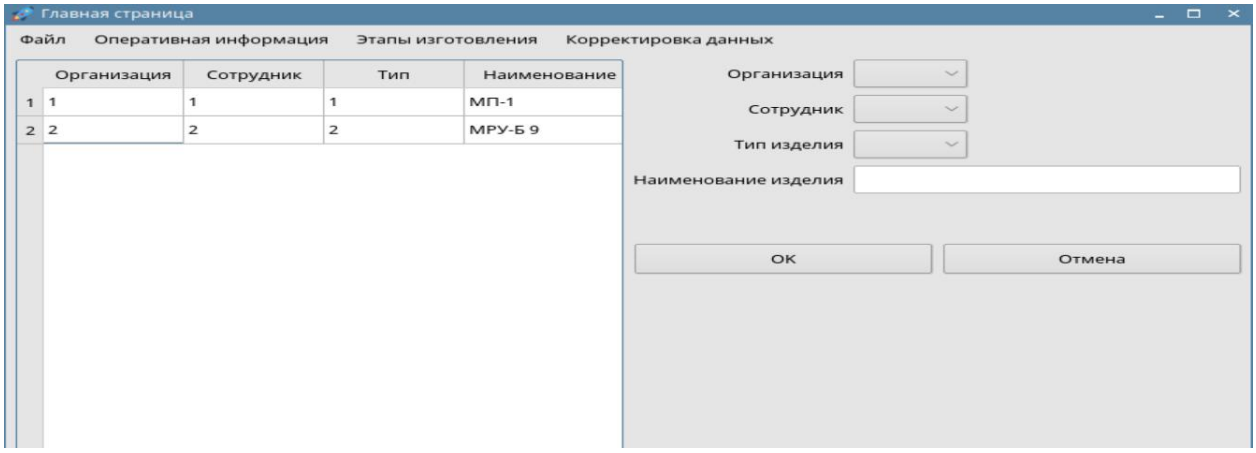


Рисунок 11 – Форма «Главная страница»

При выборе из меню «Этапы изготовления» пункта «Список деталей от стороннего производителя» открывается форма «Состав» (рисунок 12) на которой доступно добавление, изменение и удаление данных о изделиях от стороннего производителя, установленных на изделия.

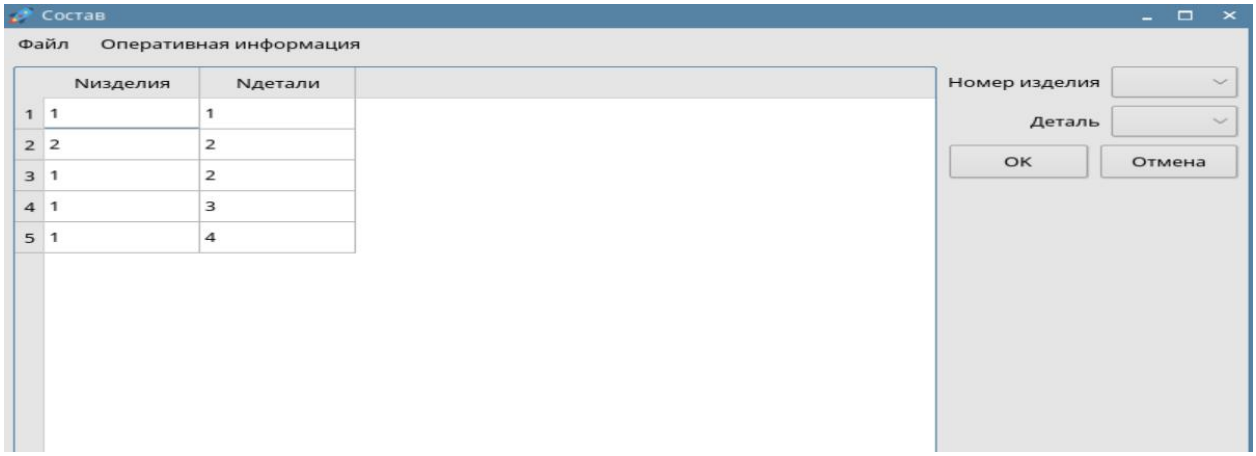


Рисунок 12 – Форма «Состав»

При выборе из меню «Этапы изготовления» пункта «Примечание» открывается форма «Примечание» (рисунок 13) на которой доступно добавление, изменение и удаление текстовых примечаний о изделиях.

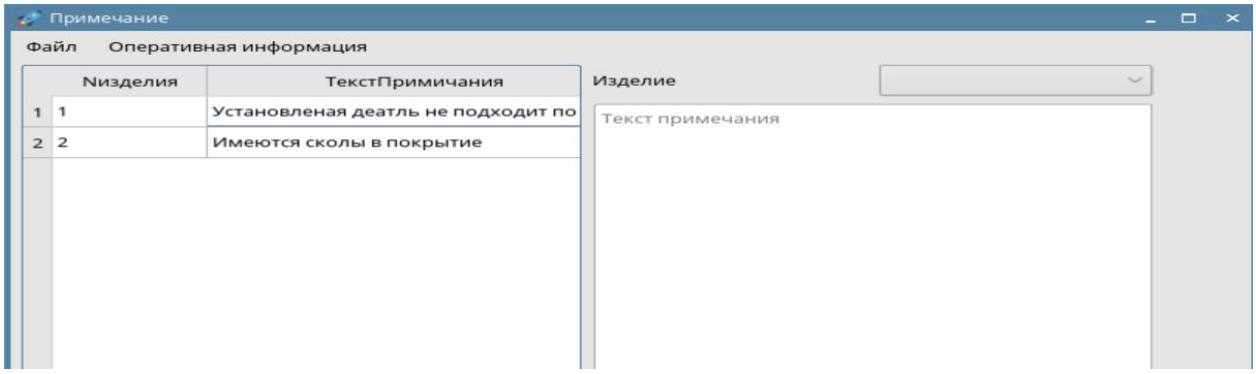


Рисунок 13 – Форма «Примечание»

При выборе из меню «Этапы изготовления» пункта «Сборка изделия» открывается форма «Сборка изделия» (рисунок 14) на которой доступно внесение изменений в данные о сборке изделия.

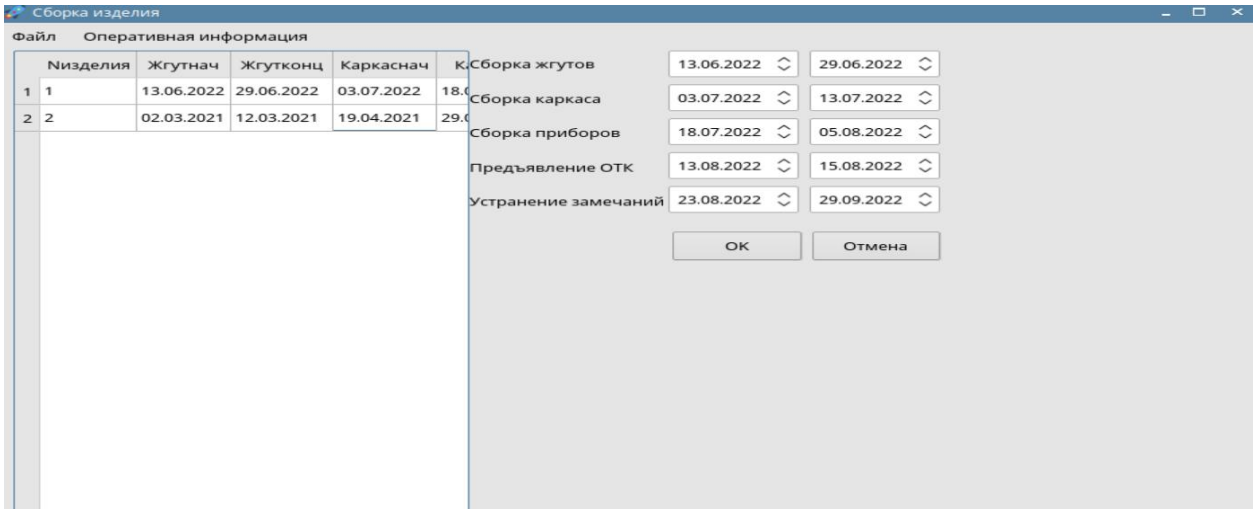


Рисунок 14 – Форма «Сборка изделия»

При выборе из меню «Этапы изготовления» пункта «Сдача изделия» открывается форма «Сдача изделия» (рисунок 15) на которой доступно внесение изменений в данные о сдаче изделия.

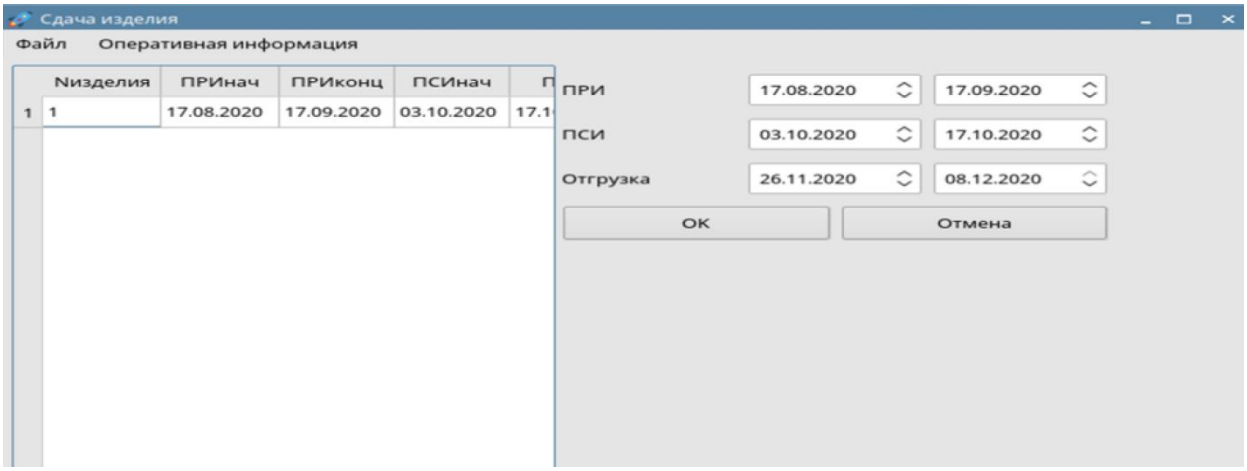


Рисунок 15 – Форма «Сдача изделия»

При выборе из меню «Этапы изготовления» пункта «Наладка» открывается форма «Наладка» (рисунок 16) на которой доступно внесение изменений в данные о наладке изделия.

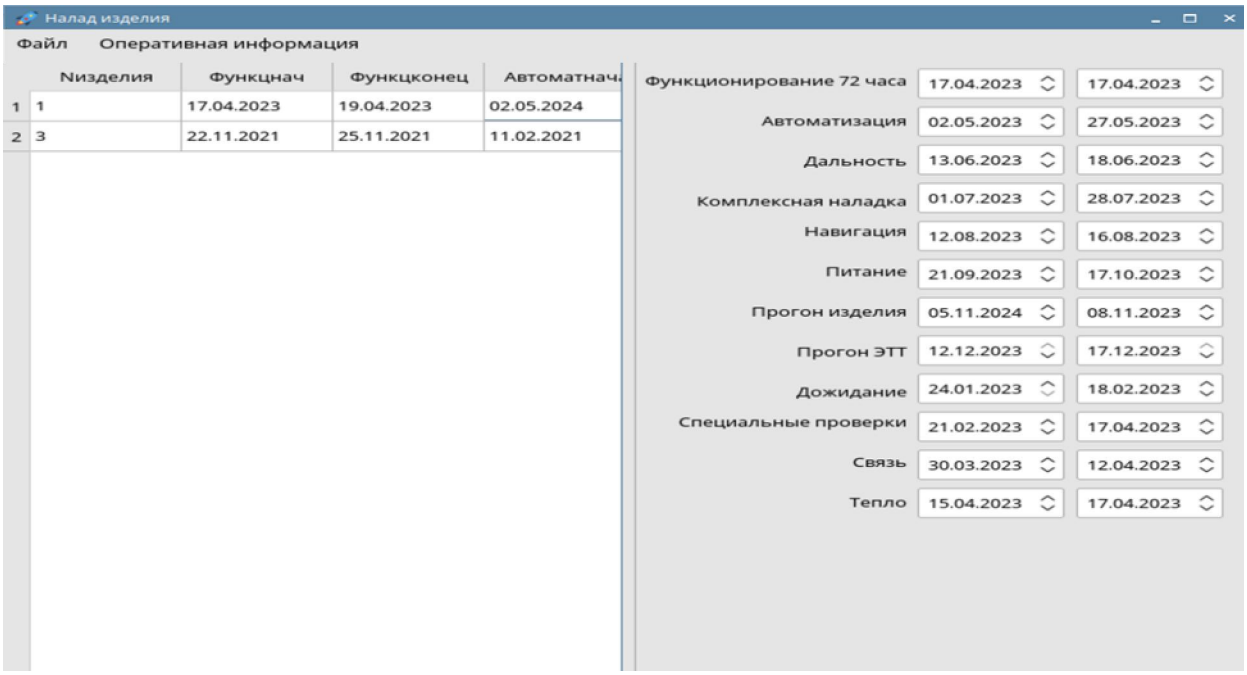


Рисунок 16 – Форма «Наладка»

При выборе из меню «Корректировка данных» пункта «Сотрудник» открывается форма «Сотрудник» (рисунок 17), где указываются фамилия, имя, отчество и отдел сотрудника. Пользователю доступно добавление, изменение и удаление данных о сотруднике.

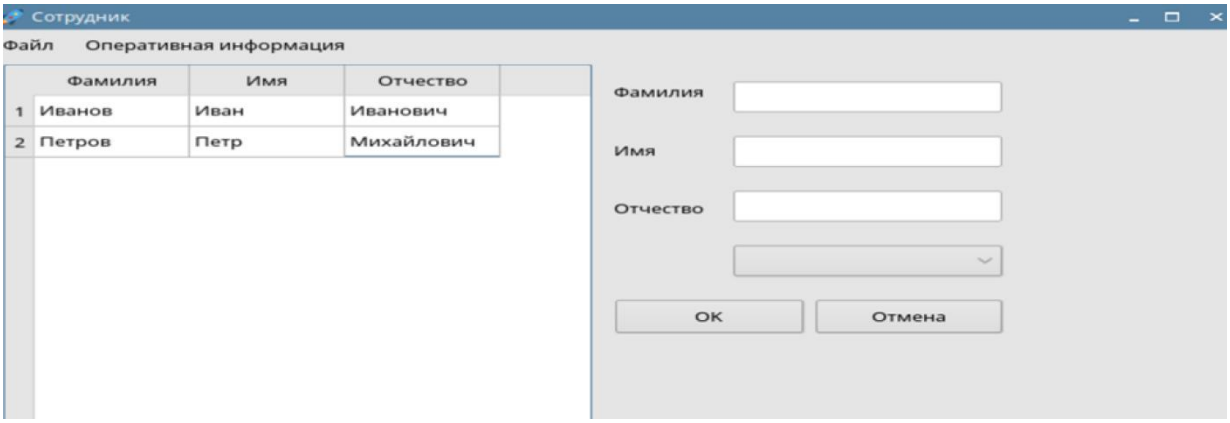


Рисунок 17 – Форма «Сотрудник»

При выборе из меню «Корректировка данных» пункта «Отдел» открывается форма «Отдел» (рисунок 18), где указывается наименование отдела. Пользователю доступно добавление, изменение и удаление данных о отделе.

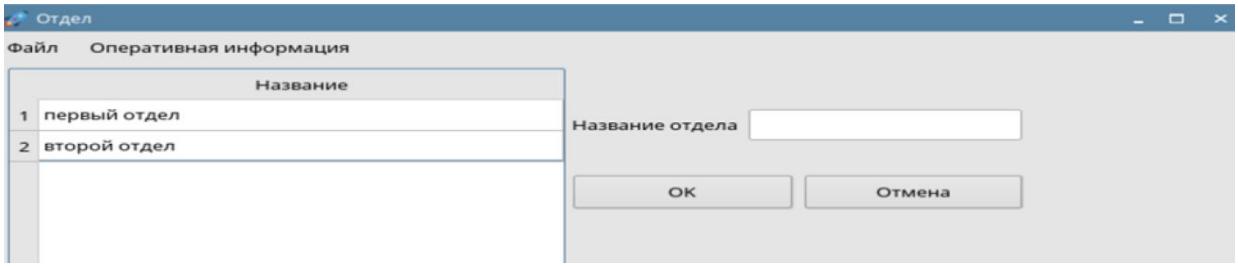


Рисунок 18 – Форма «Отдел»

При выборе из меню «Корректировка данных» пункта «Тип изделия» открывается форма «Тип изделия» (рисунок 19), где указываются цифровое и буквенное обозначения. Пользователю доступно добавление, изменение и удаление данных о типе.

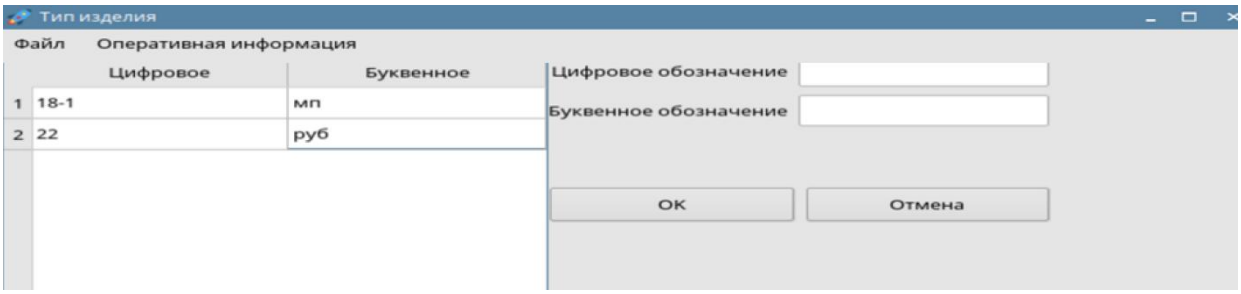
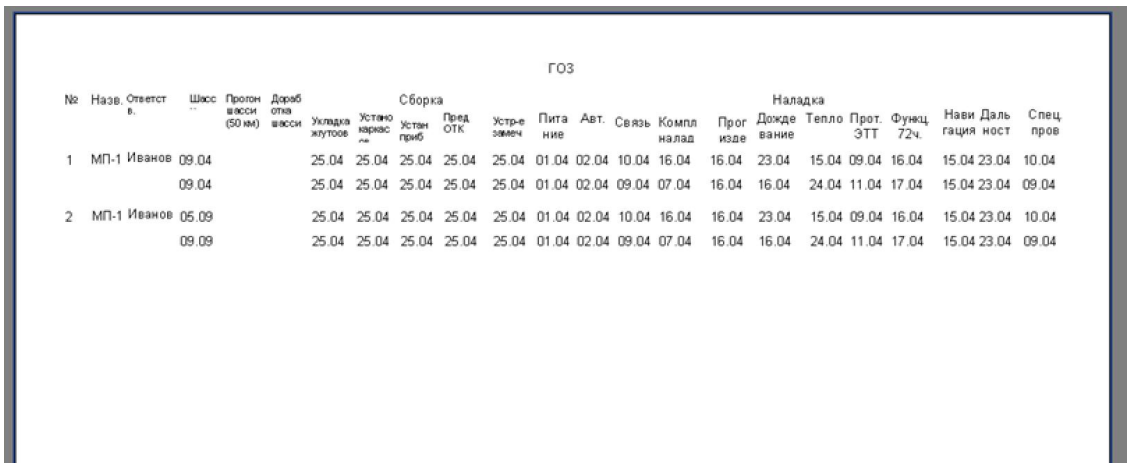


Рисунок 19 – Форма «Тип изделия»

При выборе главного меню «Оперативная информация» и подменю «Просмотр таблицы «ГОЗ» формируется отчёт, где приводятся все сведения о выполненных работах по всем изделиям (рисунок 20).

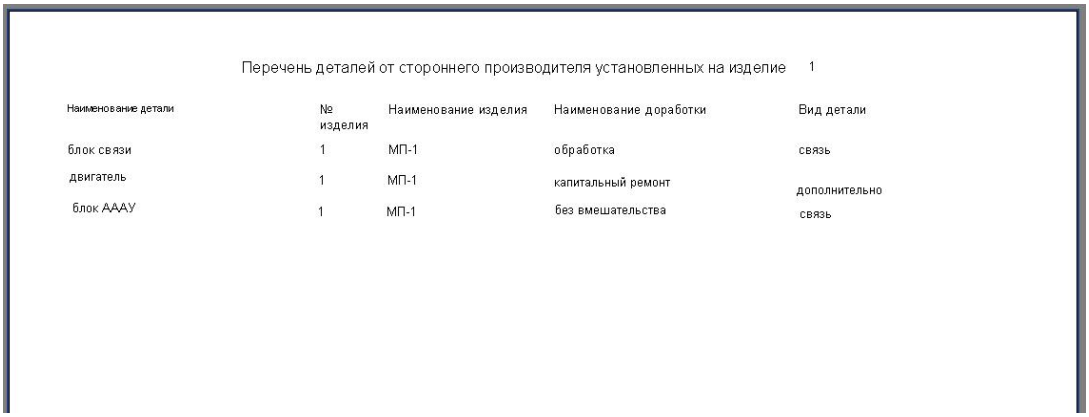


ГОЗ

№	Назв.	Ответст в.	Шасс и	Протон шасси (50 км)	Дораб отка шасси	Сборка								Наладка								
						Укладка жгутоов	Устано каркас	Устан приб	Пред. ОТК	Устр-е замеч	Пита ние	Авт.	Связь	Компл налад	Прог изде	Дожде вание	Тепло	Прот. ЭТТ	Функц. 72ч.	Нави гация	Даль ност	Спец. пров
1	МП-1	Иванов	09.04			25.04	25.04	25.04	25.04	25.04	01.04	02.04	10.04	16.04	16.04	23.04	15.04	09.04	16.04	15.04	23.04	10.04
			09.04			25.04	25.04	25.04	25.04	25.04	01.04	02.04	09.04	07.04	16.04	16.04	24.04	11.04	17.04	15.04	23.04	09.04
2	МП-1	Иванов	05.09			25.04	25.04	25.04	25.04	25.04	01.04	02.04	10.04	16.04	16.04	23.04	15.04	09.04	16.04	15.04	23.04	10.04
			09.09			25.04	25.04	25.04	25.04	25.04	01.04	02.04	09.04	07.04	16.04	16.04	24.04	11.04	17.04	15.04	23.04	09.04

Рисунок 20 – Отчёт «Изготовление серии «ГОЗ»»

При выборе главного меню «Оперативная информация» и подменю «Перечень деталей» формируется отчёт, где приводятся все сведения о деталях от стороннего производителя, установленных на изделия (рисунок 21).



Перечень деталей от стороннего производителя установленных на изделие 1

Наименование детали	№ изделия	Наименование изделия	Наименование доработки	Вид детали
блок связи	1	МП-1	обработка	связь
двигатель	1	МП-1	капитальный ремонт	дополнительно
блок АДАУ	1	МП-1	без вмешательства	связь

Рисунок 21 – Отчёт «Перечень деталей от стороннего производителя, установленных на изделие»

При выборе главного меню «Оперативная информация» и подменю «График выполнения работ по изделию» открывается форма «График выполнения работ по изделию».

1.7.3 Входные и выходные данные

Входными данными программного обеспечения являются: информация о изделии, информация о деталях от стороннего производителя, информация о сотрудниках, информация о этапах изготовления изделий.

Выходными данными программного обеспечения являются: сводная таблица «ГОЗ», график выполнения работ по каждому изделию, перечень деталей от стороннего производителя.

2 Охрана труда

2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения

Работа с вычислительной техникой, совершается в условиях ограниченной подвижности, с длительным статическим мышечным напряжением. Также происходит перенапряжение зрительных органов. Воздействие на человека выше указанных неблагоприятных факторов приводит к снижению работоспособности, переутомлению и психологическому раздражению, появлению физических недомогания, общей усталости.

Для снижения вредного воздействия вычислительной техники на человека, важно соблюдать ряд правил использования ПЭВМ. Основными из которых являются:

– правильная организация рабочего места;

– освещенность рабочего места;

– соблюдение правил электробезопасности;

– соблюдение правил пожарной безопасности:

– режим труда и отдыха.

2.1.1 Оборудование рабочего места

Ниже представлены основные требования, которым должно удовлетворять рабочее место:

– расстояние от глаз до экрана должно быть в пределах 600 – 700 мм;

– монитор должен располагаться ниже уровня глаз. Угол между направлением при взгляде прямо и направлением на центр экрана должен составлять 10-15 градусов;

– рабочий стол должен регулироваться по высоте в пределах 680 – 800 мм, при отсутствии такой возможности его высота должна составлять 725мм;

– клавиатуру следует располагать на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной регулируемой по высоте рабочей поверхности, отделенной от основной столешницы;

– рабочий стол должен иметь пространство для ног высотой не менее 600 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

На рисунке 21 изображена организация рабочего места, которая отвечает указанным требованиям.

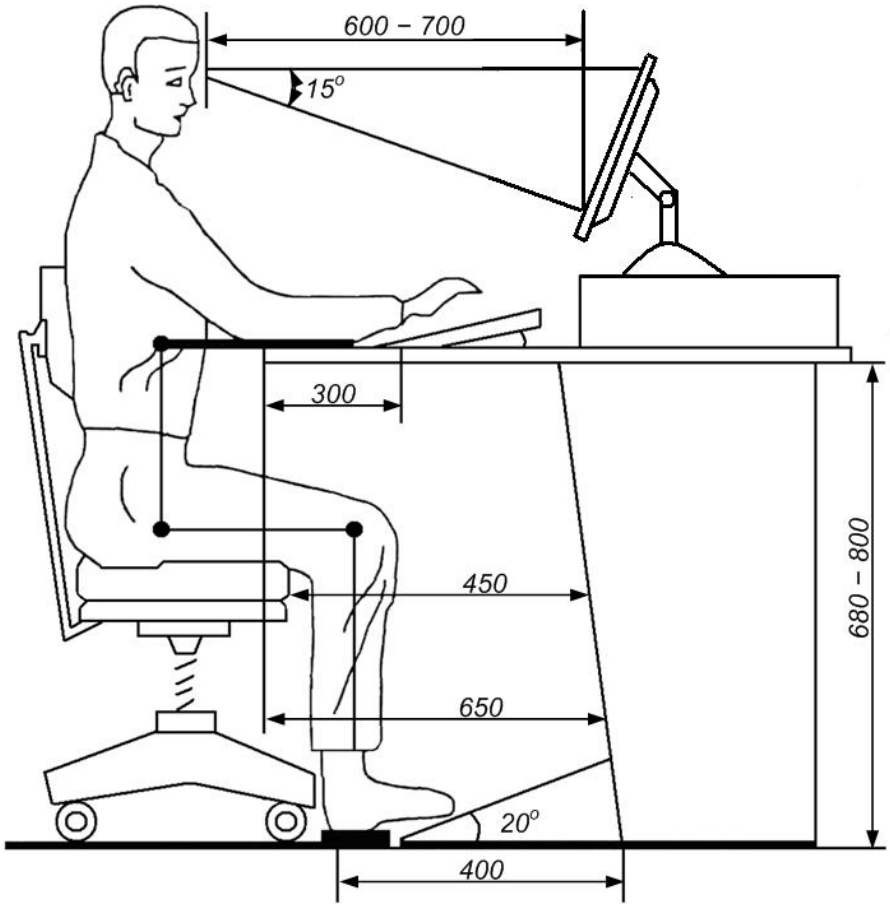


Рисунок 21 – Организация рабочего места

Рабочее место с дисплеем должно обеспечивать оператору возможность удобного выполнения работ в положении сидя и не создавать перегрузки костно-мышечной системы.

При правильно организованном рабочем месте должна обеспечиваться правильная посадка пользователя ПК, изображённая на рисунке 22

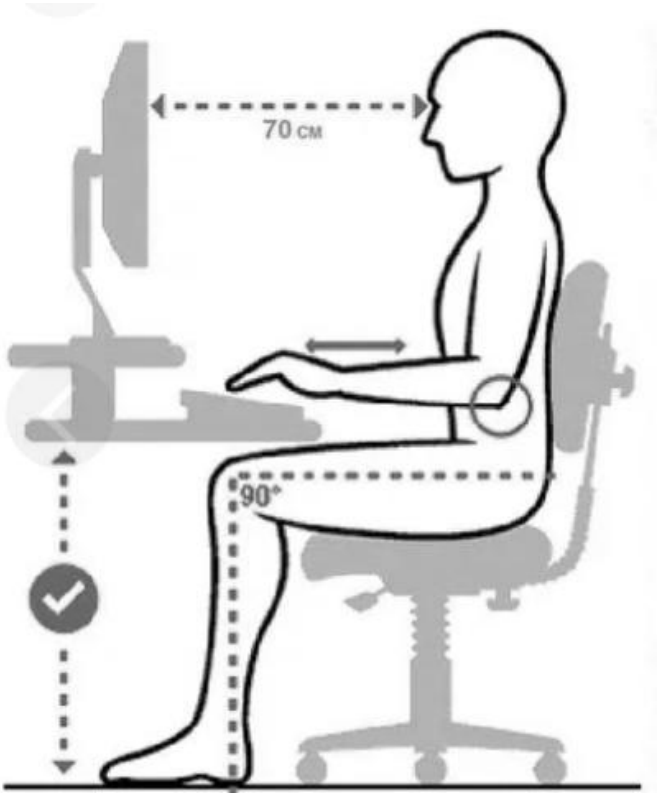


Рисунок 22 – Правильная посадка за ПК

Посадка должна обеспечивать выполнение следующих условий:

– запястья должны находиться в прямом положении;

– плечи находятся в расслабленном состоянии;

– локти должны полностью лежать на столе, не свисая и располагаться перпендикулярно полу (угол в локтевых суставах должен составлять 90-100 градусов).

2.1.2 Освещенность рабочего места

Нормы освещенности устанавливают с учетом характера выполняемой работы, места и поверхности. Для пультов ЭВМ и дисплеев – комбинированное освещение не менее 400 люкс, а общее освещение не менее 200 люкс.

На рабочем месте необходимо обеспечивать большую равномерность яркости, исключая наличие ярких и блестящих предметов, для снижения монотонности в поле зрения рекомендуется отдельные пестрые поверхности.

Для освещения используются в основном потолочные или встроенные светильники с люминесцентными лампами. Для исключения засвечивания

экранов прямыми световыми потоками светильники общего освещения располагают сбоку от рабочего места, параллельно линии зрения оператора.

2.2 Электробезопасность

К электробезопасности предъявляются требования по ГОСТ 12.1.019-2017 «ЭЛЕКТРОБЕЗОПАСНОСТЬ».

ПК – это комплекс устройств, работающий от сети переменного тока частотой 50 Гц с напряжением 230 В. Электрический ток при таком напряжении опасен для жизни. Поэтому при работе на ПК необходимо соблюдать следующие требования электробезопасности:

– все устройства должны питаться от одной фазы электрической сети;

– сетевое электропитание устройств ПК должно производиться только от розеток типа «Европа» с заземляющими контактами;

– все электрические розетки, предназначенные для подключения к ним устройств ПК, должны иметь маркировку по напряжению;

– значение номинального напряжения сети необходимо наносить яркой краской, крупными символами (высотой не менее 50 мм) на стене или щите, возле розетки.

В процессе использования средств вычислительной техники и периферийного оборудования каждый работник должен внимательно относиться к электропроводке, приборам и аппаратам, всегда помнить, что несоблюдение правил безопасности угрожает здоровью и жизни человека.

Необходимо постоянно следить на своем рабочем месте за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть.

После обнаружения неисправности необходимо незамедлительно обесточить электрооборудование. Продолжение работы возможно только после устранения всех неисправностей.

2.2.1 Пожарная безопасность

При эксплуатации электроустановок и ПК наиболее частыми причинами пожаров, возникающих при использовании электроустановок, являются:

– короткие замыкания в проводках и электрических устройствах;

– воспламенение горючих материалов вблизи от электрических устройств;

– большие переходные сопротивления в местах контактных соединений.

Пожарные меры профилактики проводятся согласно федеральному закону от 21.12.1994 №69-ФЗ (ред. от 28.09.2010) "О пожарной безопасности". Для пожарной безопасности требуется выполнить ряд организационных мероприятий, направленных на обеспечение безопасности людей и предотвращение пожара с ограничением его распространения. А также создать условия для успешного тушения пожаров. Для повышения противопожарной безопасности помещения необходимы следующие мероприятия:

– проходы в помещения и запасные выходы не должны быть загромождены посторонними предметами, которые препятствуют свободному прохождению;

– к средствам пожаротушения всегда должен быть свободный доступ;

– система вентиляции должна быть оборудована устройствам, обеспечивающим автоматическое отключение ее при пожаре, а также огне задерживающим клапаном;

– установленная автоматическая пожарная сигнализация и средства пожаротушения должны содержаться в исправном состоянии и в соответствии с типовыми правилами технического содержания установок пожарной автоматики.

2.2.2 Режим труда и отдыха

Работник, работающий с ПК, обязан соблюдать режим труда и отдыха в зависимости от продолжительности, вида и категории трудовой деятельности:

Существуют следующие виды трудовой деятельности, связанные с персональным компьютером:

– группа А – работа по считыванию информации с экрана видео дисплейного терминала с предварительным запросом;

– группа Б – работа по вводу информации;

– группа В – творческая работа в режиме диалога с ПЭВМ.

При выполнении работ, относящихся к различным видам трудовой деятельности, за основную работу с компьютером следует принимать тот вид деятельности, который занимает не менее 50% времени в течение всей рабочей смены или рабочего дня.

Для видов трудовой деятельности устанавливается 3 категории тяжести и напряженности работы с компьютером, которые определяются следующим образом:

– для группы А – по суммарному числу считываемых знаков за рабочую смену (не более 60000 знаков за смену);

– для группы Б – по суммарному числу считываемых или вводимых знаков за рабочую смену (не более 40000 знаков за смену);

– для группы В – по суммарному времени непосредственной работы с компьютером за рабочую смену (не более 6 часов за смену).

Суммарное время перерывов для каждой категории и видов работы указаны в таблице 11.

Таблица 11 – Время перерывов

Категория работы с ПЭВМ или ВДТ	Уровень нагрузки за рабочую смену при видах работы с ВДТ			Суммарное время регламентированных перерывов, мин	
	группа А, количество знаков	группа Б, количество знаков	группа В час	при 8-часовой смене	при 12-часовой смене
1	До 20000	До 15000	До 2	50	80
2	До 40000	До 30000	До 4	70	110

3	До 60000	До 40000	До 6	90	140

Заключение

В результате проделанной работы было разработано и отлаженопрограммное обеспечение автоматизированной системы контроля изготовления изделий специального назначения, позволяющая обрабатывать и хранить данные о сотрудниках, изделиях, деталях от стороннего производителя, организациях заказчиках, этапах изготовления изделий и т.д. Была разработана модель данных системы на основе которой спроектирована структура базы данных, разработано программное обеспечение автоматизированной системы, отлажено и протестировано.

В разделе по безопасности жизнедеятельности вкр изложены возможные вредные факторы на рабочем месте разработчика автоматизированной системы контроля изготовления изделий специального назначения. На основании изученной литературы по данной проблеме, выделены физиологические, психофизиологические опасные и вредные факторы. И разработаны технические, организационные и профилактические мероприятия по каждому из опасного и вредного фактора.

На этапе проектирования была разработана модель автоматизированной системы контроля изготовления изделий специального назначения, также проведено физическое проектирование и разработана структура базы данных.

Для разработки автоматизированной системы использовались СУБД PostgresSQL и в качестве средства разработки программного обеспечения для работы с базой данных выбрана среда QtCreator.

Проведено тестирование показавшее, что система работает верно, без системных сбоев.

Таким образом, заданиена дипломное проектирование выполнено в полном объеме.

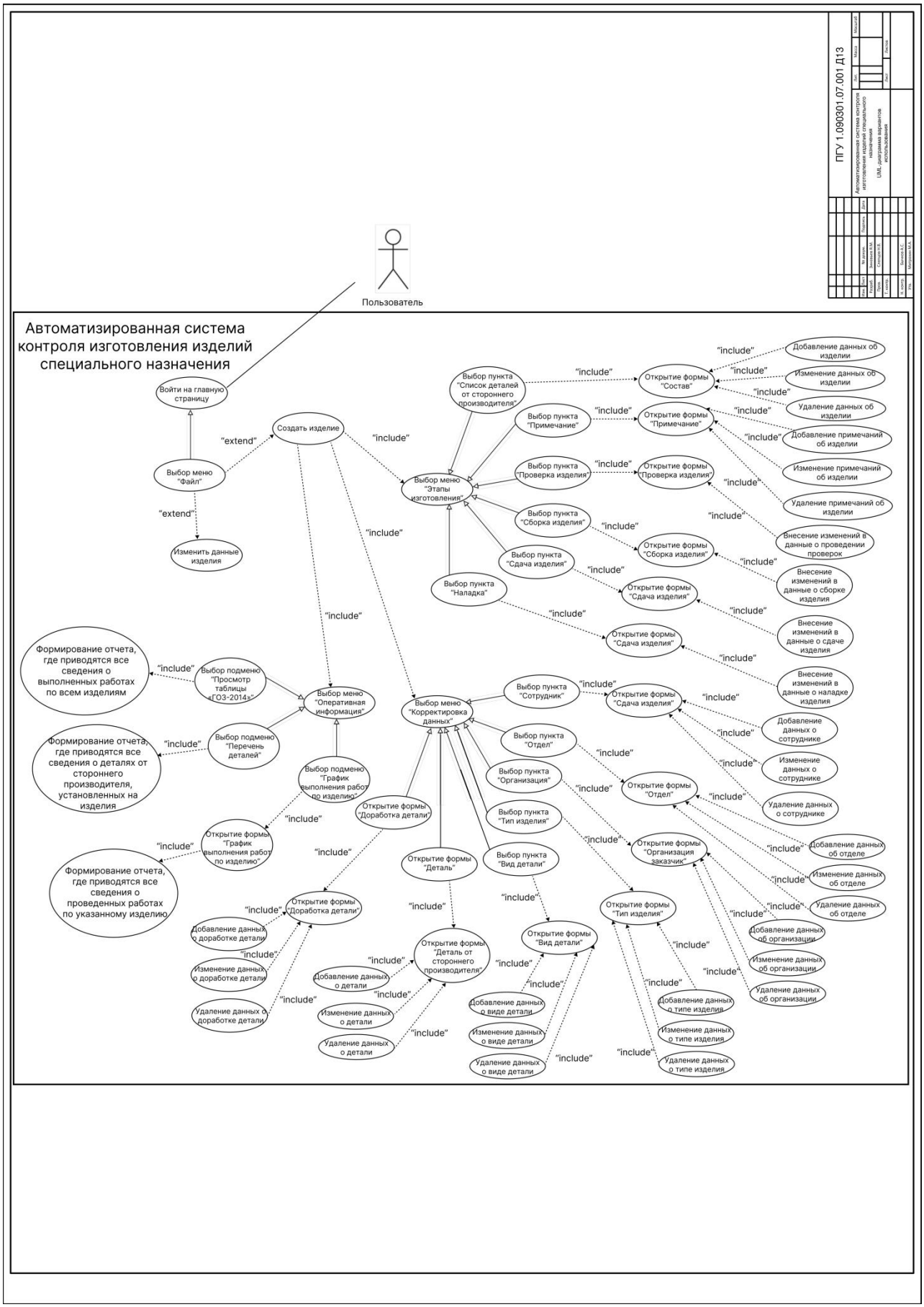
Список использованных источников

1. Сиддхартха Рао. Освой самостоятельно C++ за 21 день. 7-е издание. – М.: Издательство «Вильямс», 2013. – 685 с. 15.05.2024
2. Доусон М. Программируем на Python. –СПб.: Питер, 2024. – 416 с.23.05.2024
3. Хорстманн, Кей С., Корнелл, Гари. Java – Библиотека проффесионала, том 1. Основы. 9-е изд.: Пер. с англ. – М: ООО «И.Д. Вильямс», 2024. – 864 с. 03.06.2024
4. Рогов Е. В. PostgreSQL 16 изнутри. – М.: ДМК Пресс, 2024. – 664 с.10.06.2024
5. Громаков Е.И., Разработка логической и физической моделей БД в CASE- системе проектирования ERWIN: Методические указания – М.: 2000. 14.06.2024
6. Ерёменко А.В. Проектирование приложения БД. – Пенза, ПГУ, 2009 17.07.2024

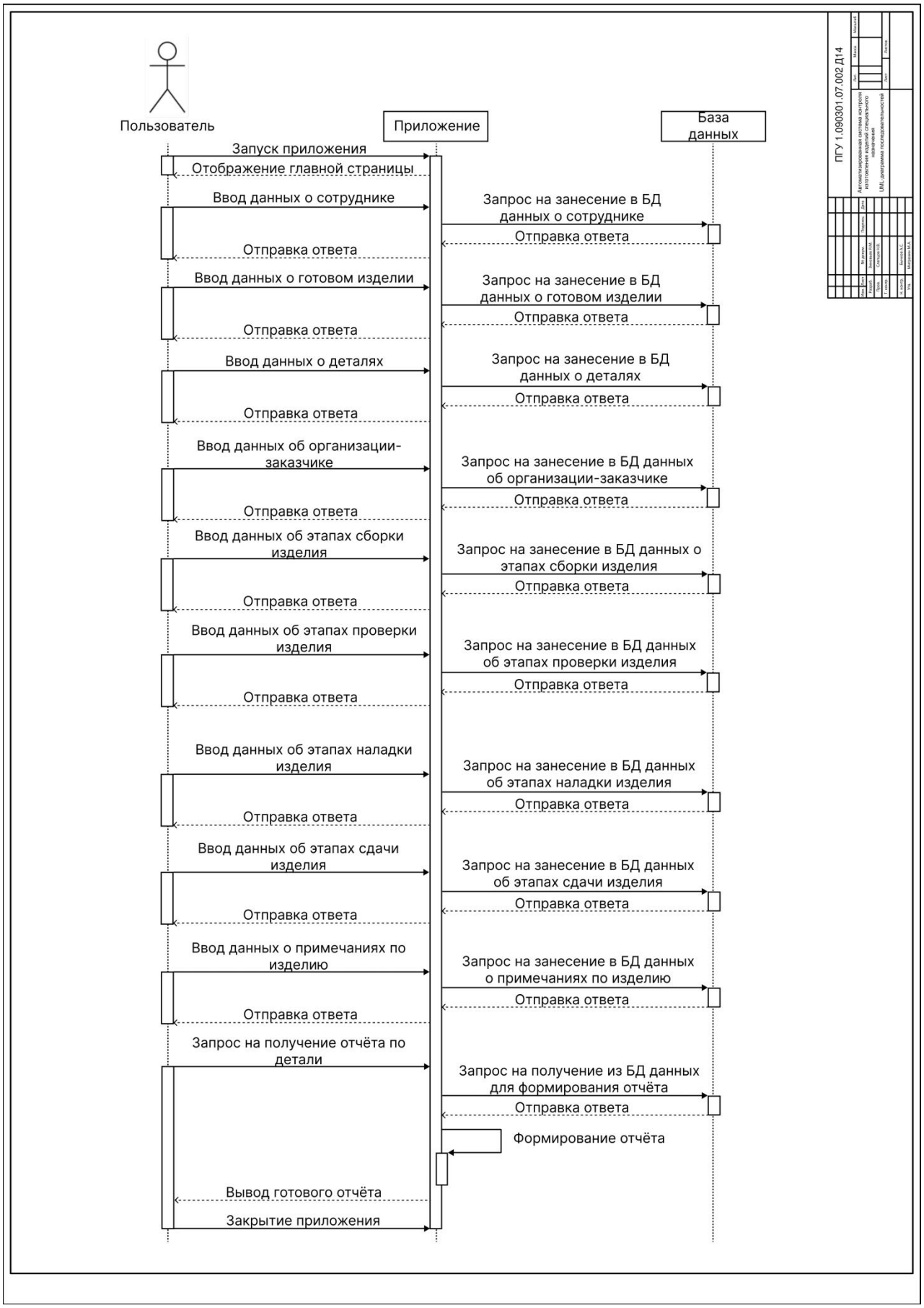
Приложение А – UML-диаграммы

Автоматизированная система контроля изготовления изделий специального назначения

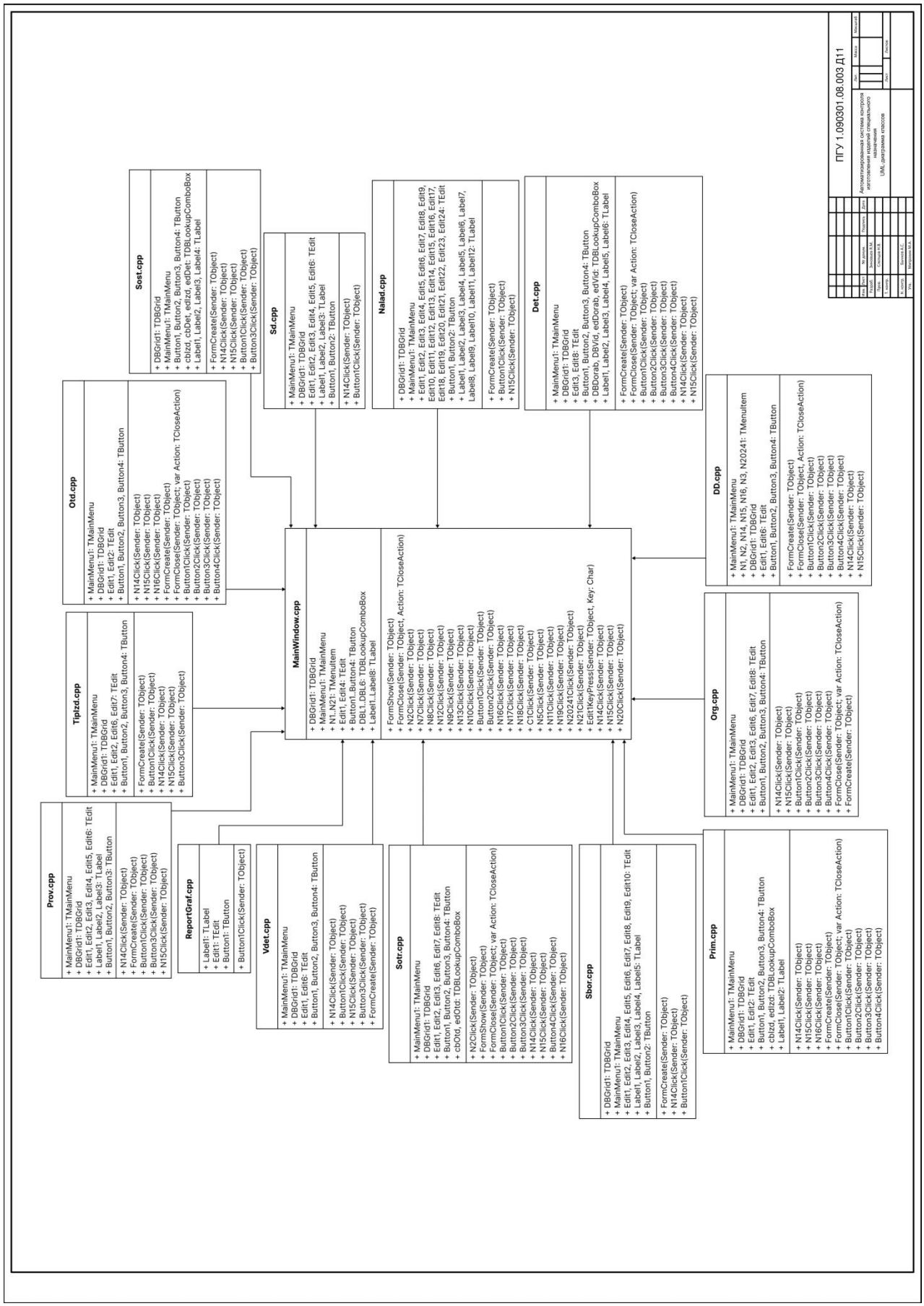
Приложение А.1 – UML-диаграмма вариантов использования



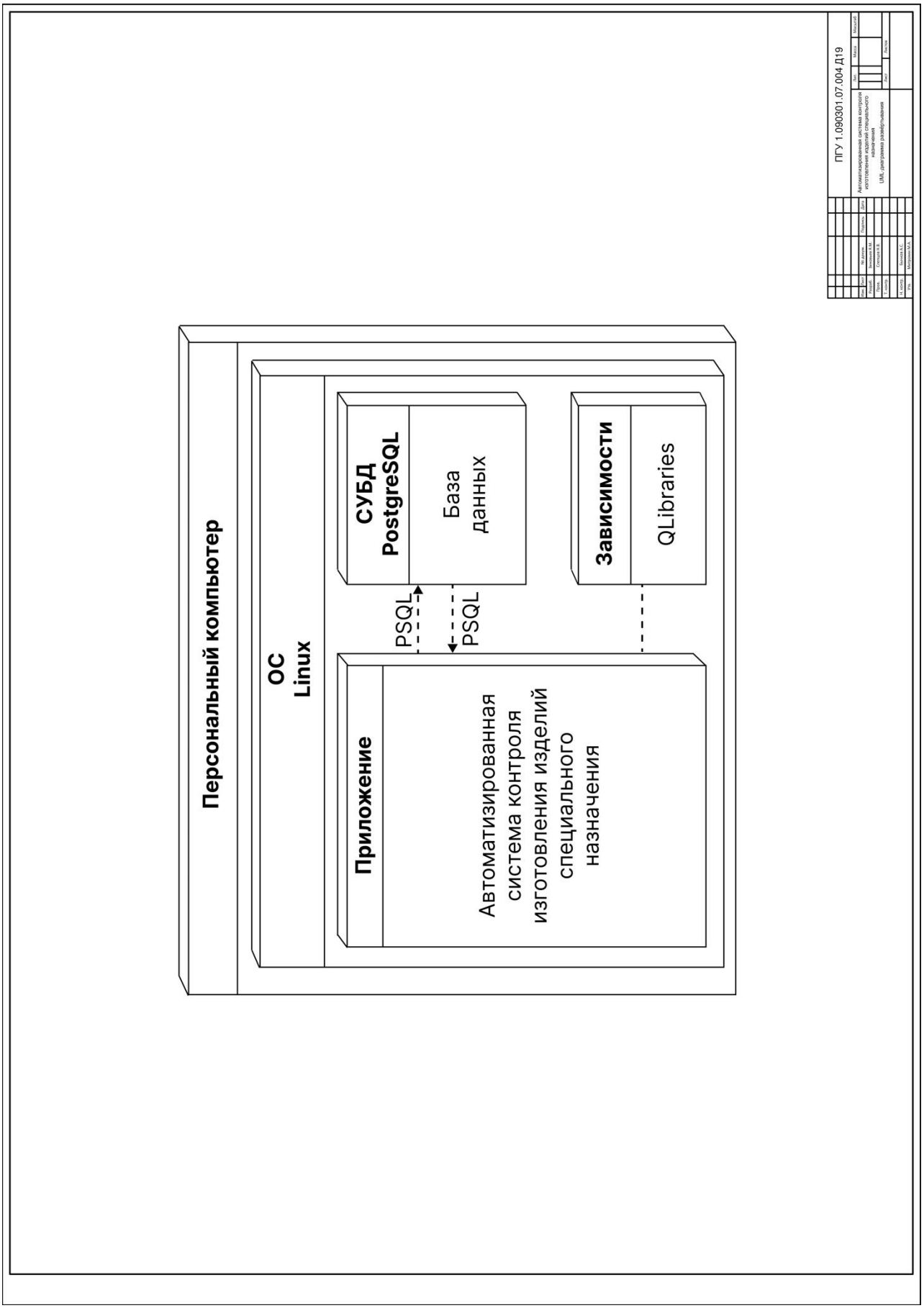
Приложение А.2 – UML-диаграмма последовательностей



Приложение А.3 – UML-диаграмма классов



Приложение А.4 – UML-диаграмма развёртывания



Приложение Б Листинг программы

Автоматизированная система контроля изготовления изделий специального назначения

Файл mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QtWidgets>

class MainWindow : public QMainWindow {
  Q_OBJECT

public:
  explicit MainWindow(QWidget *parent = nullptr);

private slots:
  void openDatabase();
  void closeDatabase();

private:
  Ui::MainWindow *ui;
};

#endif
```

Файл mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QtSql>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent), ui(new Ui::MainWindow) {
  ui->setupUi(this);

  connect(ui->openDatabaseAction, &QAction::triggered, this,
&MainWindow::openDatabase);
  connect(ui->closeDatabaseAction, &QAction::triggered, this,
&MainWindow::closeDatabase);
}

void MainWindow::openDatabase() {
  QSqlDatabase db = QSqlDatabase::addDatabase("QPSQL");
  db.setHostName("localhost");
  db.setDatabaseName("database");
  db.setUserName("Yaroslav");
  db.setPassword("password");

  if (!db.open()) {
    QMessageBox::critical(nullptr, "Ошибка подключения",
                          "Не удалось подключиться к базе
данных.");
    return;
  }
```

```
}

void MainWindow::closeDatabase() {
  QSqlDatabase db = QSqlDatabase::database("QPSQL");
  if (db.isOpen()) {
    db.close();
  }

}
```

Файл otd.h

```
#ifndef OTD_H
#define OTD_H

#include <QtWidgets>
#include <QtSql>

class Otd : public QWidget {
  Q_OBJECT

public:
  explicit Otd(QWidget *parent = nullptr);

private slots:
  void addOtd();
  void editOtd();
  void deleteOtd();

private:
  QSqlTableModel *otdModel;
  QTableView *otdView;

};

#endif
```

Файл otd.cpp

```
#include "otd.h"
#include "ui_otd.h"

Otd::Otd(QWidget *parent) : QWidget(parent), ui(new Ui::Otd) {
  ui->setupUi(this);

  otdModel = new QSqlTableModel(this);
  otdModel->setTable("otd");
  otdModel->select();

  otdView = new QTableView(this);
```

```
  otdView->setModel(otdModel);

  ui->otdView->setModel(otdModel);
  connect(ui->otdView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &Otd::onCurrentRowChanged);
}

void Otd::addOtd() {
  otdModel->insertRows(otdModel->rowCount(), 1);
  otdModel->setData(otdModel->index(otdModel->rowCount() - 1, 0),
                    ui->otdNameEdit->text());
  otdModel->submitAll();
  otdView->update();
}

void Otd::editOtd() {
  otdModel->setData(otdModel->index(otdView-
>currentIndex().row(), 0),
                    ui->otdNameEdit->text());
  otdModel->submitAll();
  otdView->update();
}

void Otd::deleteOtd() {
  otdModel->removeRow(otdView->currentIndex().row());
  otdModel->submitAll();
  otdView->update();
}

void Otd::onCurrentRowChanged(const QModelIndex &current,
                              const QModelIndex &previous) {
  if (current.isValid()) {
    ui->otdNameEdit->setText(
        otdModel->data(otdModel->index(current.row(),
0)).toString());
  }
}
```

Файл det.h

```
#ifndef DET_H
#define DET_H

#include <QtWidgets>
#include <QtSql>

class Det : public QWidget {
  Q_OBJECT

public:
  explicit Det(QWidget *parent = nullptr);
```

```
private slots:
  void addDet();
  void editDet();
  void deleteDet();

private:
  QSqlTableModel *detModel;
  QTableView *detView;
  QComboBox *dorabComboBox;
  QComboBox *vidComboBox;

};

#endif
```

Файл det.cpp

```
#include "det.h"
#include "ui_det.h"

Det::Det(QWidget *parent) : QWidget(parent), ui(new Ui::Det) {
  ui->setupUi(this);

  detModel = new QSqlTableModel(this);
  detModel->setTable("det");
  detModel->select();

  detView = new QTableView(this);
  detView->setModel(detModel);

  ui->detView->setModel(detModel);
  connect(ui->detView->selectionModel(),
&QItemSelectionModel::currentRowChanged, this,
&Det::onCurrentRowChanged);

  QSqlTableModel *dorabModel = new QSqlTableModel(this);
  dorabModel->setTable("dorab");
  dorabModel->select();

  QSqlTableModel *vidModel = new QSqlTableModel(this);
  vidModel->setTable("vid");
  vidModel->select();

  dorabComboBox = new QComboBox(this);
  dorabComboBox->setModel(dorabModel);
  dorabComboBox->setModelColumn(1);

  vidComboBox = new QComboBox(this);
  vidComboBox->setModel(vidModel);
  vidComboBox->setModelColumn(1);
```

```
  ui->dorabComboBox->setModel(dorabModel);
  ui->dorabComboBox->setModelColumn(1);
  ui->vidComboBox->setModel(vidModel);
  ui->vidComboBox->setModelColumn(1);

  connect(ui->dorabComboBox, SIGNAL(currentIndexChanged(int)),
this, SLOT(addDet()));
  connect(ui->vidComboBox, SIGNAL(currentIndexChanged(int)),
this, SLOT(addDet()));
}

void Det::addDet() {
  detModel->insertRows(detModel->rowCount(), 1);
  detModel->setData(detModel->index(detModel->rowCount() - 1, 0),
                    ui->detNameEdit->text());
  detModel->setData(detModel->index(detModel->rowCount() - 1, 1),
                    ui->dorabComboBox->currentData().toInt());
  detModel->setData(detModel->index(detModel->rowCount() - 1, 2),
                    ui->vidComboBox->currentData().toInt());
  detModel->submitAll();
  detView->update();
}

void Det::editDet() {
  detModel->setData(detModel->index(detView-
>currentIndex().row(), 0),
                    ui->detNameEdit->text());
  detModel->setData(detModel->index(detView-
>currentIndex().row(), 1),
                    ui->dorabComboBox->currentData().toInt());
  detModel->setData(detModel->index(detView-
>currentIndex().row(), 2),
                    ui->vidComboBox->currentData().toInt());
  detModel->submitAll();
  detView->update();
}

void Det::deleteDet() {
  detModel->removeRow(detView->currentIndex().row());
  detModel->submitAll();
  detView->update();
}

void Det::onCurrentRowChanged(const QModelIndex &current,
                              const QModelIndex &previous) {
  if (current.isValid()) {
    ui->detNameEdit->setText(
        detModel->data(detModel->index(current.row(),
0)).toString());
    ui->dorabComboBox->setCurrentIndex(
```

```
        detModel->data(detModel->index(current.row(),
1)).toInt());
    ui->vidComboBox->setCurrentIndex(
        detModel->data(detModel->index(current.row(),
2)).toInt());
  }
}
```

org.h

```
#ifndef ORG_H
#define ORG_H

#include <QtWidgets>
#include <QtSql>

class Org : public QWidget {
  Q_OBJECT

public:
  explicit Org(QWidget *parent = nullptr);

private slots:
  void addOrg();
  void editOrg();
  void deleteOrg();

private:
  QSqlTableModel *orgModel;
  QTableView *orgView;

};

#endif
```

Файл org.cpp

```
#include "org.h"
#include "ui_org.h"

Org::Org(QWidget *parent) : QWidget(parent), ui(new Ui::Org) {
  ui->setupUi(this);

  orgModel = new QSqlTableModel(this);
  orgModel->setTable("org");
  orgModel-> select();

  orgView = new QTableView(this);
  orgView->setModel(orgModel);

  ui->orgView->setModel(orgModel);
```

```
  connect(ui->orgView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &Org::onCurrentRowChanged);
}

void Org::addOrg() {
  orgModel->insertRows(orgModel->rowCount(), 1);
  orgModel->setData(orgModel->index(orgModel->rowCount() - 1, 0),
                    ui->orgNameEdit->text());
  orgModel->setData(orgModel->index(orgModel->rowCount() - 1, 1),
                    ui->orgAddressEdit->text());
  orgModel->setData(orgModel->index(orgModel->rowCount() - 1, 2),
                    ui->orgDogovEdit->text());
  orgModel->submitAll();
  orgView->update();
}

void Org::editOrg() {
  orgModel->setData(orgModel->index(orgView-
>currentIndex().row(), 0),
                    ui->orgNameEdit->text());
  orgModel->setData(orgModel->index(orgView-
>currentIndex().row(), 1),
                    ui->orgAddressEdit->text());
  orgModel->setData(orgModel->index(orgView-
>currentIndex().row(), 2),
                    ui->orgDogovEdit->text());
  orgModel->submitAll();
  orgView->update();
}

void Org::deleteOrg() {
  orgModel->removeRow(orgView->currentIndex().row());
  orgModel->submitAll();
  orgView->update();
}

void Org::onCurrentRowChanged(const QModelIndex &current,
                              const QModelIndex &previous) {
  if (current.isValid()) {
    ui->orgNameEdit->setText(
        orgModel->data(orgModel->index(current.row(),
0)).toString());
    ui->orgAddressEdit->setText(
        orgModel->data(orgModel->index(current.row(),
1)).toString());
    ui->orgDogovEdit->setText(
        orgModel->data(orgModel->index(current.row(),
2)).toString());
  }
}
```

Файл dd.h

```
#ifndef DD_H
#define DD_H

#include <QtWidgets>
#include <QtSql>

class DD : public QWidget {
  Q_OBJECT

public:
  explicit DD(QWidget *parent = nullptr);

private slots:
  void addDD();
  void editDD();
  void deleteDD();

private:
  QSqlTableModel *ddModel;
  QTableView *ddView;
};

#endif
```

Файл dd.cpp

```
#include "dd.h"
#include "ui_dd.h"

DD::DD(QWidget *parent) : QWidget(parent), ui(new Ui::DD) {
  ui->setupUi(this);

  ddModel = new QSqlTableModel(this);
  ddModel->setTable("dd");
  ddModel->select();

  ddView = new QTableView(this);
  ddView->setModel(ddModel);

  ui->ddView->setModel(ddModel);
  connect(ui->ddView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &DD::onCurrentRowChanged);
}

void DD::addDD() {
  ddModel->insertRows(ddModel->rowCount(), 1);
  ddModel->setData(ddModel->index(ddModel->rowCount() - 1, 0),
                   ui->ddNameEdit->text());
  ddModel->submitAll();
```

```
  ddView->update();
}

void DD::editDD() {
  ddModel->setData(ddModel->index(ddView->currentIndex().row(),
0),
                   ui->ddNameEdit->text());
  ddModel->submitAll();
  ddView->update();
}

void DD::deleteDD() {
  // Delete selected record from ddModel
  ddModel->removeRow(ddView->currentIndex().row());
  ddModel->submitAll();
  ddView->update();
}

void DD::onCurrentRowChanged(const QModelIndex &current,
                             const QModelIndex &previous) {
  if (current.isValid()) {
    ui->ddNameEdit->setText(
        ddModel->data(ddModel->index(current.row(),
0)).toString());
  }
}
```

Файл tipizd.h

```
#ifndef TIPIZD_H
#define TIPIZD_H

#include <QtWidgets>
#include <QtSql>

class TipIzd : public QWidget {
  Q_OBJECT

public:
  explicit TipIzd(QWidget *parent = nullptr);

private slots:
  void addTipIzd();
  void editTipIzd();
  void deleteTipIzd();

private:
  QSqlTableModel *tipizdModel;
  QTableView *tipizdView;
};
```

```
#endif
```

Файл tipizd.cpp

```
#include "tipizd.h"
#include "ui_tipizd.h"

TipIzd::TipIzd(QWidget *parent) : QWidget(parent), ui(new
Ui::TipIzd) {
  ui->setupUi(this);

  tipizdModel = new QSqlTableModel(this);
  tipizdModel->setTable("tipizd");
  tipizdModel->select();

  tipizdView = new QTableView(this);
  tipizdView->setModel(tipizdModel);

  ui->tipizdView->setModel(tipizdModel);
  connect(ui->tipizdView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &TipIzd::onCurrentRowChanged);
}

void TipIzd::addTipIzd() {
  // Add new record to tipizdModel
  tipizdModel->insertRows(tipizdModel->rowCount(), 1);
  tipizdModel->setData(tipizdModel->index(tipizdModel->rowCount()
- 1, 0),
                       ui->tipizdName1Edit->text());
  tipizdModel->setData(tipizdModel->index(tipizdModel->rowCount()
- 1, 1),
                       ui->tipizdNameAEdit->text());
  tipizdModel->submitAll();
  tipizdView->update();
}

void TipIzd::editTipIzd() {
  // Edit selected record in tipizdModel
  tipizdModel->setData(tipizdModel->index(tipizdView-
>currentIndex().row(), 0),
                       ui->tipizdName1Edit->text());
  tipizdModel->setData(tipizdModel->index(tipizdView-
>currentIndex().row(), 1),
                       ui->tipizdNameAEdit->text());
  tipizdModel->submitAll();
  tipizdView->update();
}

void TipIzd::deleteTipIzd() {
  // Delete selected record from tipizdModel
```

```
  tipizdModel->removeRow(tipizdView->currentIndex().row());
  tipizdModel->submitAll();
  tipizdView->update();
}

void TipIzd::onCurrentRowChanged(const QModelIndex &current,
                                 const QModelIndex &previous) {
  if (current.isValid()) {
    ui->tipizdName1Edit->setText(
        tipizdModel->data(tipizdModel->index(current.row(),
0)).toString());
    ui->tipizdNameAEdit->setText(
        tipizdModel->data(tipizdModel->index(current.row(),
1)).toString());
  }
}
```

Файл nalad.h

```
#ifndef NALAD_H
#define NALAD_H

#include <QtWidgets>
#include <QtSql>

class Nalad : public QWidget {
  Q_OBJECT

public:
  explicitNalad(QWidget *parent = nullptr);

private slots:
  void editNalad();

private:
  QSqlTableModel *naladModel;
  QTableView *naladView;
};

#endif
```

Файл nalad.cpp

```
#include "nalad.h"
#include "ui_nalad.h"

Nalad::Nalad(QWidget *parent) : QWidget(parent), ui(new
Ui::Nalad) {
  ui->setupUi(this);

  naladModel = new QSqlTableModel(this);
```

```
  naladModel->setTable("nalad");
  naladModel->select();

  naladView = new QTableView(this);
  naladView->setModel(naladModel);

  ui->naladView->setModel(naladModel);
  connect(ui->naladView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &Nalad::onCurrentRowChanged);
}

void Nalad::editNalad() {
  // Edit selected record in naladModel
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 1),
                      ui->nalad72NEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 2),
                      ui->nalad72KEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 3),
                      ui->naladAvtsistNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 4),
                      ui->naladAvtsistKEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 5),
                      ui->naladDalNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 6),
                      ui->naladDalKEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 7),
                      ui->naladKomplNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 8),
                      ui->naladKomplKEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 9),
                      ui->naladNavigNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 10),
                      ui->naladNavigKEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 11),
                      ui->naladPitNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 12),
                      ui->naladPitKEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 13),
```

```
                    ui->naladProgonNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 14),
                    ui->naladProgonKEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 15),
                    ui->naladProtNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 16),
                    ui->naladProtKEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 17),
                    ui->naladRainNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 18),
                    ui->naladRainKEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 19),
                    ui->naladSpecNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 20),
                    ui->naladSvazNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 21),
                    ui->naladSvazKEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 22),
                    ui->naladTeploNEdit->text());
  naladModel->setData(naladModel->index(naladView-
>currentIndex().row(), 23),
                    ui->naladTeploKEdit->text());
  naladModel->submitAll();
  naladView->update();
}

void Nalad::onCurrentRowChanged(const QModelIndex &current,
                                const QModelIndex &previous) {
  if (current.isValid()) {
    ui->nalad72NEdit->setText(
        naladModel->data(naladModel->index(current.row(),
1)).toString());
    ui->nalad72KEdit->setText(
        naladModel->data(naladModel->index(current.row(),
2)).toString());
    ui->naladAvtsistNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
3)).toString());
    ui->naladAvtsistKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
4)).toString());
    ui->naladDalNEdit->setText(
```

```
        naladModel->data(naladModel->index(current.row(),
5)).toString());
    ui->naladDalKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
6)).toString());
    ui->naladKomplNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
7)).toString());
    ui->naladKomplKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
8)).toString());
    ui->naladNavigNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
9)).toString());
    ui->naladNavigKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
10)).toString());
    ui->naladPitNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
11)).toString());
    ui->naladPitKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
12)).toString());
    ui->naladProgonNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
13)).toString());
    ui->naladProgonKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
14)).toString());
    ui->naladProtNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
15)).toString());
    ui->naladProtKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
16)).toString());
    ui->naladRainNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
17)).toString());
    ui->naladRainKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
18)).toString());
    ui->naladSpecNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
19)).toString());
    ui->naladSpecKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
20)).toString());
    ui->naladSvazNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
21)).toString());
    ui->naladSvazKEdit->setText(
```

```
        naladModel->data(naladModel->index(current.row(),
22)).toString());
    ui->naladTeploNEdit->setText(
        naladModel->data(naladModel->index(current.row(),
23)).toString());
    ui->naladTeploKEdit->setText(
        naladModel->data(naladModel->index(current.row(),
24)).toString());
  }
}
```

Файл sost.h

```
#ifndef SOST_H
#define SOST_H

#include <QtWidgets>
#include <QtSql>

class Sost : public QWidget {
  Q_OBJECT

public:
  explicit Sost(QWidget *parent = nullptr);

private slots:
  void addSost();
  void editSost();
  void deleteSost();

private:
  QSqlTableModel *sostModel;
  QTableView *sostView;
  QComboBox *izdComboBox;
  QComboBox *detComboBox;
};

#endif
```

Файл sost.cpp

```
#include "sost.h"
#include "ui_sost.h"

Sost::Sost(QWidget *parent) : QWidget(parent), ui(new Ui::Sost)
{
  ui->setupUi(this);

  sostModel = new QSqlTableModel(this);
  sostModel->setTable("sost");
  sostModel->select();
```

```
  sostView = new QTableView(this);
  sostView->setModel(sostModel);

  ui->sostView->setModel(sostModel);
  connect(ui->sostView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &Sost::onCurrentRowChanged);

  QSqlTableModel *izdModel = new QSqlTableModel(this);
  izdModel->setTable("izd");
  izdModel->select();

  QSqlTableModel *detModel = new QSqlTableModel(this);
  detModel->setTable("det");
  detModel->select();

  izdComboBox = new QComboBox(this);
  izdComboBox->setModel(izdModel);
  izdComboBox->setModelColumn(1); // Display "NAME_IZD" column

  detComboBox = new QComboBox(this);
  detComboBox->setModel(detModel);
  detComboBox->setModelColumn(1); // Display "NAME_DET" column

  ui->izdComboBox->setModel(izdModel);
  ui->izdComboBox->setModelColumn(1);
  ui->detComboBox->setModel(detModel);
  ui->detComboBox->setModelColumn(1);

  connect(ui->izdComboBox, SIGNAL(currentIndexChanged(int)),
this, SLOT(addSost()));
  connect(ui->detComboBox, SIGNAL(currentIndexChanged(int)),
this, SLOT(addSost()));
}

void Sost::addSost() {
  sostModel->insertRows(sostModel->rowCount(), 1);
  sostModel->setData(sostModel->index(sostModel->rowCount() - 1,
0),
                     ui->izdComboBox->currentData().toInt());
  sostModel->setData(sostModel->index(sostModel->rowCount() - 1,
1),
                     ui->detComboBox->currentData().toInt());
  sostModel->submitAll();
  sostView->update();
}

void Sost::editSost() {
  sostModel->setData(sostModel->index(sostView-
>currentIndex().row(), 0),
                     ui->izdComboBox->currentData().toInt());
```

```
  sostModel->setData(sostModel->index(sostView-
>currentIndex().row(), 1),
                     ui->detComboBox->currentData().toInt());
  sostModel->submitAll();
  sostView->update();
}

void Sost::deleteSost() {
  sostModel->removeRow(sostView->currentIndex().row());
  sostModel->submitAll();
  sostView->update();
}

void Sost::onCurrentRowChanged(const QModelIndex &current,
                               const QModelIndex &previous) {
  if (current.isValid()) {
    ui->izdComboBox->setCurrentIndex(
        sostModel->data(sostModel->index(current.row(),
0)).toInt());
    ui->detComboBox->setCurrentIndex(
        sostModel->data(sostModel->index(current.row(),
1)).toInt());
  }
}
```

Файл prim.h

```
#ifndef PRIM_H
#define PRIM_H

#include <QtWidgets>
#include <QtSql>

class Prim : public QWidget {
  Q_OBJECT

public:
  explicit Prim(QWidget *parent = nullptr);

private slots:
  void addPrim();
  void editPrim();
  void deletePrim();

private:
  QSqlTableModel *primModel;
  QTableView *primView;
  QComboBox *izdComboBox;
};

#endif
```

Файл prim.cpp

```
#include "prim.h"
#include "ui_prim.h"

Prim::Prim(QWidget *parent) : QWidget(parent), ui(new Ui::Prim)
{
  ui->setupUi(this);

  primModel = new QSqlTableModel(this);
  primModel->setTable("prim");
  primModel->select();

  primView = new QTableView(this);
  primView->setModel(primModel);

  ui->primView->setModel(primModel);
  connect(ui->primView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &Prim::onCurrentRowChanged);

  QSqlTableModel *izdModel = new QSqlTableModel(this);
  izdModel->setTable("izd");
  izdModel->select();

  izdComboBox = new QComboBox(this);
  izdComboBox->setModel(izdModel);
  izdComboBox->setModelColumn(1); // Display "NAME_IZD" column

  ui->izdComboBox->setModel(izdModel);
  ui->izdComboBox->setModelColumn(1);

  connect(ui->izdComboBox, SIGNAL(currentIndexChanged(int)),
this, SLOT(addPrim()));
}

void Prim::addPrim() {
  primModel->insertRows(primModel->rowCount(), 1);
  primModel->setData(primModel->index(primModel->rowCount() - 1,
0),
                     ui->primTextEdit->text());
  primModel->setData(primModel->index(primModel->rowCount() - 1,
1),
                     ui->izdComboBox->currentData().toInt());
  primModel->submitAll();
  primView->update();
}

void Prim::editPrim() {
```

```
  primModel->setData(primModel->index(primView-
>currentIndex().row(), 0),
                     ui->primTextEdit->text());
  primModel->setData(primModel->index(primView-
>currentIndex().row(), 1),
                     ui->izdComboBox->currentData().toInt());
  primModel->submitAll();
  primView->update();
}

void Prim::deletePrim() {
  primModel->removeRow(primView->currentIndex().row());
  primModel->submitAll();
  primView->update();
}

void Prim::onCurrentRowChanged(const QModelIndex &current,
                               const QModelIndex &previous) {
  if (current.isValid()) {
    ui->primTextEdit->setText(
        primModel->data(primModel->index(current.row(),
0)).toString());
    ui->izdComboBox->setCurrentIndex(
        primModel->data(primModel->index(current.row(),
1)).toInt());
  }
}
```

Файл prov.h

```
#ifndef PROV_H
#define PROV_H

#include <QtWidgets>
#include <QtSql>

class Prov : public QWidget {
  Q_OBJECT

public:
  explicit Prov(QWidget *parent = nullptr);

private slots:
  void addProv();
  void editProv();
  void deleteProv();

private:
  QSqlTableModel *provModel;
  QTableView *provView;
};
```

```
#endif
```

Файл prov.cpp

```
#include "prov.h"
#include "ui_prov.h"

Prov::Prov(QWidget *parent) : QWidget(parent), ui(new Ui::Prov)
{
  ui->setupUi(this);

  provModel = new QSqlTableModel(this);
  provModel->setTable("prov");
  provModel->select();

  provView = new QTableView(this);
  provView->setModel(provModel);

  ui->provView->setModel(provModel);
  connect(ui->provView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &Prov::onCurrentRowChanged);
}

void Prov::addProv() {
  provModel->insertRows(provModel->rowCount(), 1);
  provModel->setData(provModel->index(provModel->rowCount() - 1,
1),
                     ui->provShassinEdit->text());
  provModel->setData(provModel->index(provModel->rowCount() - 1,
2),
                     ui->provSassikEdit->text());
  provModel->setData(provModel->index(provModel->rowCount() - 1,
3),
                     ui->provProgonNEdit->text());
  provModel->setData(provModel->index(provModel->rowCount() - 1,
4),
                     ui->provProgonKEdit->text());
  provModel->setData(provModel->index(provModel->rowCount() - 1,
5),
                     ui->provDorabNEdit->text());
  provModel->setData(provModel->index(provModel->rowCount() - 1,
6),
                     ui->provDorabKEdit->text());
  provModel->submitAll();
  provView->update();
}

void Prov::editProv() {
```

```
  provModel->setData(provModel->index(provView-
>currentIndex().row(), 1),
                     ui->provShassinEdit->text());
  provModel->setData(provModel->index(provView-
>currentIndex().row(), 2),
                     ui->provSassikEdit->text());
  provModel->setData(provModel->index(provView-
>currentIndex().row(), 3),
                     ui->provProgonNEdit->text());
  provModel->setData(provModel->index(provView-
>currentIndex().row(), 4),
                     ui->provProgonKEdit->text());
  provModel->setData(provModel->index(provView-
>currentIndex().row(), 5),
                     ui->provDorabNEdit->text());
  provModel->setData(provModel->index(provView-
>currentIndex().row(), 6),
                     ui->provDorabKEdit->text());
  provModel->submitAll();
  provView->update();
}

void Prov::deleteProv() {
  provModel->removeRow(provView->currentIndex().row());
  provModel->submitAll();
  provView->update();
}

void Prov::onCurrentRowChanged(const QModelIndex &current,
                               const QModelIndex &previous) {
  if (current.isValid()) {
    ui->provShassinEdit->setText(
        provModel->data(provModel->index(current.row(),
1)).toString());
    ui->provSassikEdit->setText(
        provModel->data(provModel->index(current.row(),
2)).toString());
    ui->provProgonNEdit->setText(
        provModel->data(provModel->index(current.row(),
3)).toString());
    ui->provProgonKEdit->setText(
        provModel->data(provModel->index(current.row(),
4)).toString());
    ui->provDorabNEdit->setText(
        provModel->data(provModel->index(current.row(),
5)).toString());
    ui->provDorabKEdit->setText(
        provModel->data(provModel->index(current.row(),
6)).toString());
  }
}
```

Файл sbor.h

```
#ifndef SBOR_H
#define SBOR_H

#include <QtWidgets>
#include <QtSql>

class Sbor : public QWidget {
  Q_OBJECT

public:
  explicit Sbor(QWidget *parent = nullptr);

private slots:
  void editSbor();

private:
  QSqlTableModel *sborModel;
  QTableView *sborView;
};

#endif
```

Файл sbor.cpp

```
#include "sbor.h"
#include "ui_sbor.h"

Sbor::Sbor(QWidget *parent) : QWidget(parent), ui(new Ui::Sbor)
{
  ui->setupUi(this);

  sborModel = new QSqlTableModel(this);
  sborModel->setTable("sbor");
  sborModel->select();

  sborView = new QTableView(this);
  sborView->setModel(sborModel);

  ui->sborView->setModel(sborModel);
  connect(ui->sborView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &Sbor::onCurrentRowChanged);
}

void Sbor::editSbor() {
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 1),
                     ui->sborJgytNEdit->text());
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 2),
```

```
                    ui->sborJgytKEdit->text());
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 3),
                    ui->sborKarkNEdit->text());
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 4),
                    ui->sborKarkKEdit->text());
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 5),
                    ui->sborPribNEdit->text());
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 6),
                    ui->sborPribKEdit->text());
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 7),
                    ui->sborOtkNEdit->text());
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 8),
                    ui->sborOtkKEdit->text());
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 9),
                    ui->sborZamenEdit->text());
  sborModel->setData(sborModel->index(sborView-
>currentIndex().row(), 10),
                    ui->sborZamekEdit->text());
  sborModel->submitAll();
  sborView->update();
}

void Sbor::onCurrentRowChanged(const QModelIndex &current,
                               const QModelIndex &previous) {
  if (current.isValid()) {
    ui->sborJgytNEdit->setText(
        sborModel->data(sborModel->index(current.row(),
1)).toString());
    ui->sborJgytKEdit->setText(
        sborModel->data(sborModel->index(current.row(),
2)).toString());
    ui->sborKarkNEdit->setText(
        sborModel->data(sborModel->index(current.row(),
3)).toString());
    ui->sborKarkKEdit->setText(
        sborModel->data(sborModel->index(current.row(),
4)).toString());
    ui->sborPribNEdit->setText(
        sborModel->data(sborModel->index(current.row(),
5)).toString());
    ui->sborPribKEdit->setText(
        sborModel->data(sborModel->index(current.row(),
6)).toString());
    ui->sborOtkNEdit->setText(
```

```
        sborModel->data(sborModel->index(current.row(),
7)).toString());
    ui->sborOtkKEdit->setText(
        sborModel->data(sborModel->index(current.row(),
8)).toString());
    ui->sborZamenEdit->setText(
        sborModel->data(sborModel->index(current.row(),
9)).toString());
    ui->sborZamekEdit->setText(
        sborModel->data(sborModel->index(current.row(),
10)).toString());
  }
}
```

Файл sd.h

```
#ifndef SD_H
#define SD_H

#include <QtWidgets>
#include <QtSql>

class Sd : public QWidget {
  Q_OBJECT

public:
  explicit Sd(QWidget *parent = nullptr);

private slots:
  void editSd();

private:
  QSqlTableModel *sdModel;
  QTableView *sdView;
};

#endif
```

Файл sd.cpp

```
#include "sd.h"
#include "ui_sd.h"

Sd::Sd(QWidget *parent) : QWidget(parent), ui(new Ui::Sd) {
  ui->setupUi(this);

  sdModel = new QSqlTableModel(this);
  sdModel->setTable("sd");
  sdModel->select();

  sdView = new QTableView(this);
```

```
  sdView->setModel(sdModel);

  ui->sdView->setModel(sdModel);
  connect(ui->sdView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &Sd::onCurrentRowChanged);
}

void Sd::editSd() {
  // Edit selected record in sdModel
  sdModel->setData(sdModel->index(sdView->currentIndex().row(),
1),
                   ui->sdPrinEdit->text());
  sdModel->setData(sdModel->index(sdView->currentIndex().row(),
2),
                   ui->sdPrikEdit->text());
  sdModel->setData(sdModel->index(sdView->currentIndex().row(),
3),
                   ui->sdPsinEdit->text());
  sdModel->setData(sdModel->index(sdView->currentIndex().row(),
4),
                   ui->sdPsikEdit->text());
  sdModel->setData(sdModel->index(sdView->currentIndex().row(),
5),
                   ui->sdOtgNEdit->text());
  sdModel->setData(sdModel->index(sdView->currentIndex().row(),
6),
                   ui->sdOtgKEdit->text());
  sdModel->submitAll();
  sdView->update();
}

void Sd::onCurrentRowChanged(const QModelIndex &current,
                             const QModelIndex &previous) {
  if (current.isValid()) {
    ui->sdPrinEdit->setText(
        sdModel->data(sdModel->index(current.row(),
1)).toString());
    ui->sdPrikEdit->setText(
        sdModel->data(sdModel->index(current.row(),
2)).toString());
    ui->sdPsinEdit->setText(
        sdModel->data(sdModel->index(current.row(),
3)).toString());
    ui->sdPsikEdit->setText(
        sdModel->data(sdModel->index(current.row(),
4)).toString());
    ui->sdOtgNEdit->setText(
        sdModel->data(sdModel->index(current.row(),
5)).toString());
    ui->sdOtgKEdit->setText(
```

```
        sdModel->data(sdModel->index(current.row(),
6)).toString());
  }
}
```

Файл vdet.h

```
#ifndef VDET_H
#define VDET_H

#include <QtWidgets>
#include <QtSql>

class Vdet : public QWidget {
Q_OBJECT

public:
  explicit Vdet(QWidget *parent = nullptr);

private slots:
  void addVdet();
  void editVdet();
  void deleteVdet();

private:
  QSqlTableModel *vdetModel;
  QTableView *vdetView;
};

#endif
```

Файл vdet.cpp

```
#include "vdet.h"
#include "ui_vdet.h"

Vdet::Vdet(QWidget *parent) : QWidget(parent), ui(new Ui::Vdet)
{
  ui->setupUi(this);

  vdetModel = new QSqlTableModel(this);
  vdetModel->setTable("vdet");
  vdetModel->select();

  vdetView = new QTableView(this);
  vdetView->setModel(vdetModel);

  ui->vdetView->setModel(vdetModel);
  connect(ui->vdetView->selectionModel(),
&QItemSelectionModel::currentRowChanged,
          this, &Vdet::onCurrentRowChanged);
```

```
}

void Vdet::addVdet() {
  vdetModel->insertRows(vdetModel->rowCount(), 1);
  vdetModel->setData(vdetModel->index(vdetModel->rowCount() - 1,
0),
                     ui->vdetNameEdit->text());
  vdetModel->submitAll();
  vdetView->update();
}

void Vdet::editVdet() {
  vdetModel->setData(vdetModel->index(vdetView-
>currentIndex().row(), 0),
                     ui->vdetNameEdit->text());
  vdetModel->submitAll();
  vdetView->update();
}

void Vdet::deleteVdet() {
  vdetModel->removeRow(vdetView->currentIndex().row());
  vdetModel->submitAll();
  vdetView->update();
}

void Vdet::onCurrentRowChanged(const QModelIndex &current,
                               const QModelIndex &previous) {
  if (current.isValid()) {
    ui->vdetNameEdit->setText(
        vdetModel->data(vdetModel->index(current.row(),
0)).toString());
  }
}
```

Файл reportgraf.h

```
#ifndef REPORTGRAF_H
#define REPORTGRAF_H

#include <QtWidgets>
#include <QtSql>
#include <QtCharts>

class ReportGraf : public QWidget {
  Q_OBJECT

public:
  explicit ReportGraf(QWidget *parent = nullptr);

private slots:
  void generateReport();
```

```
private:
  QSqlDatabase db;
  QChart *chart;
  QChartView *chartView;

};

#endif
```

Файл reportgraf.cpp

```
#include "reportgraf.h"
#include "ui_reportgraf.h"

ReportGraf::ReportGraf(QWidget *parent) : QWidget(parent),
ui(new Ui::ReportGraf) {
  ui->setupUi(this);

  // Connect to PostgreSQL database
  db = QSqlDatabase::addDatabase("QPSQL");
  db.setHostName("localhost");
  db.setDatabaseName("your_database");
  db.setUserName("your_user");
  db.setPassword("your_password");

  if (!db.open()) {
    QMessageBox::critical(nullptr, "Ошибка подключения",
                          "Не удалось подключиться к базе
данных.");
    return;
  }
}

void ReportGraf::generateReport() {
}
```

Приложение Г Презентация

Автоматизированная система контроля изготовления изделий специального назначения

Автоматизированная система контроля изготовления изделий специального назначения

Автор работы:
студент гр. 20ВВП1 Зиновьев Ярослав Михайлович
Научный руководитель:
к.т.н., доцент Слепцов Николай Владимирович

Пенза, 2024

Актуальность проекта

Основной задачей автоматизации является построение автоматизированной системы контроля изготовления изделий специального назначения, упрощающей доступ к данным, их хранение и изменение.

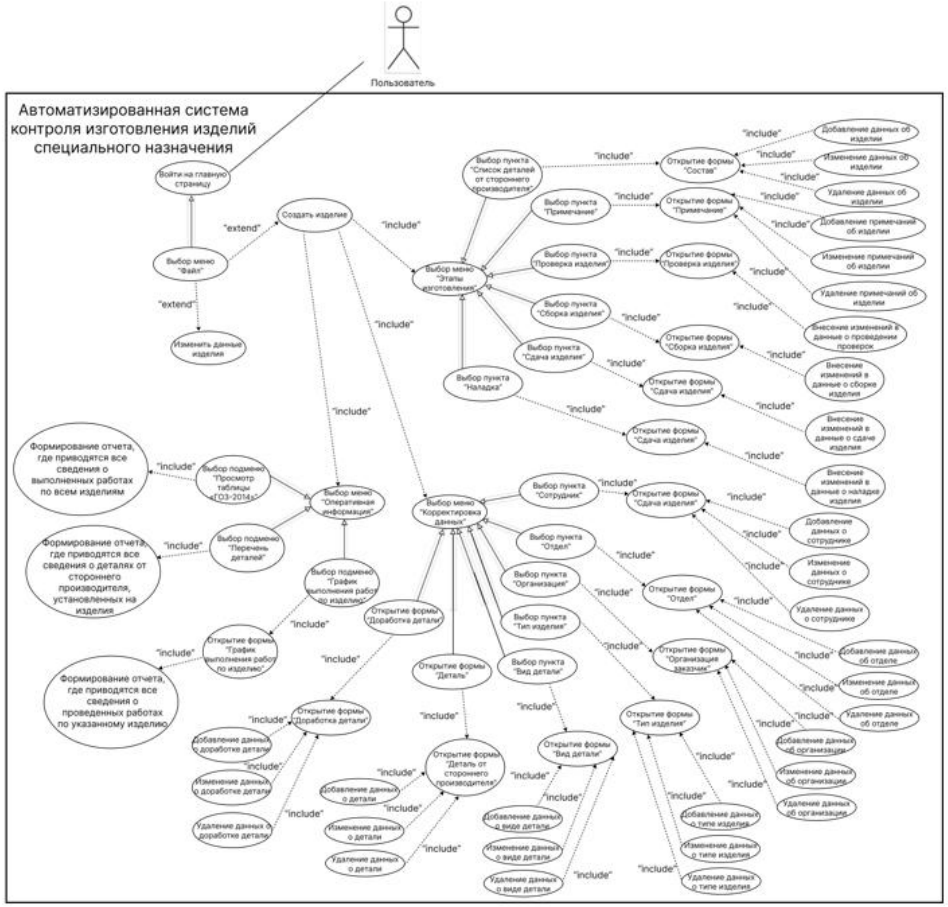
Целевая аудитория проекта: Основным местом внедрения данного приложения являются предприятия направлениями деятельности, которых являются разработка, производство и модернизация стационарных и мобильных автоматизированных систем и комплексов управления, информационных систем специального назначения, в том числе специальных машин на колесной и гусеничной базе. Разработкой изделий специального назначения занимаются научно-технические центры (НТЦ) предприятия.

Цель работы и объекты

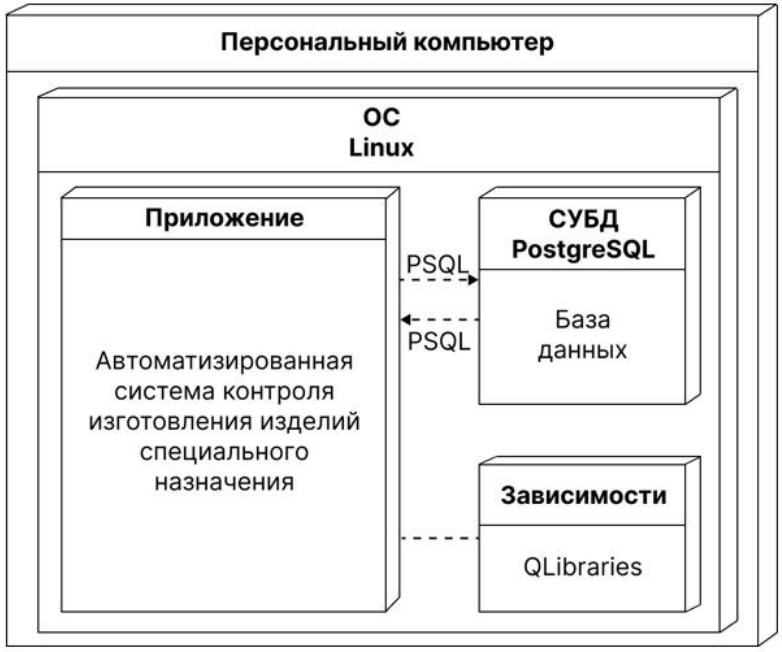
Объектом исследования является процесс контроля изготовления изделий в научно-техническом центре.

Цель работы – автоматизация процесса контроля изготовления изделий специального назначения.

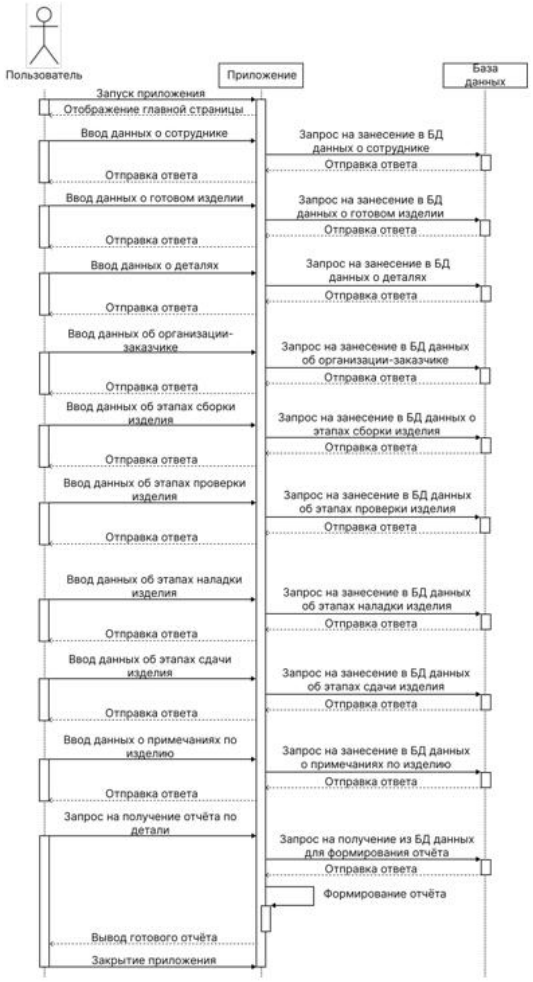
UML-диаграммой вариантов использования



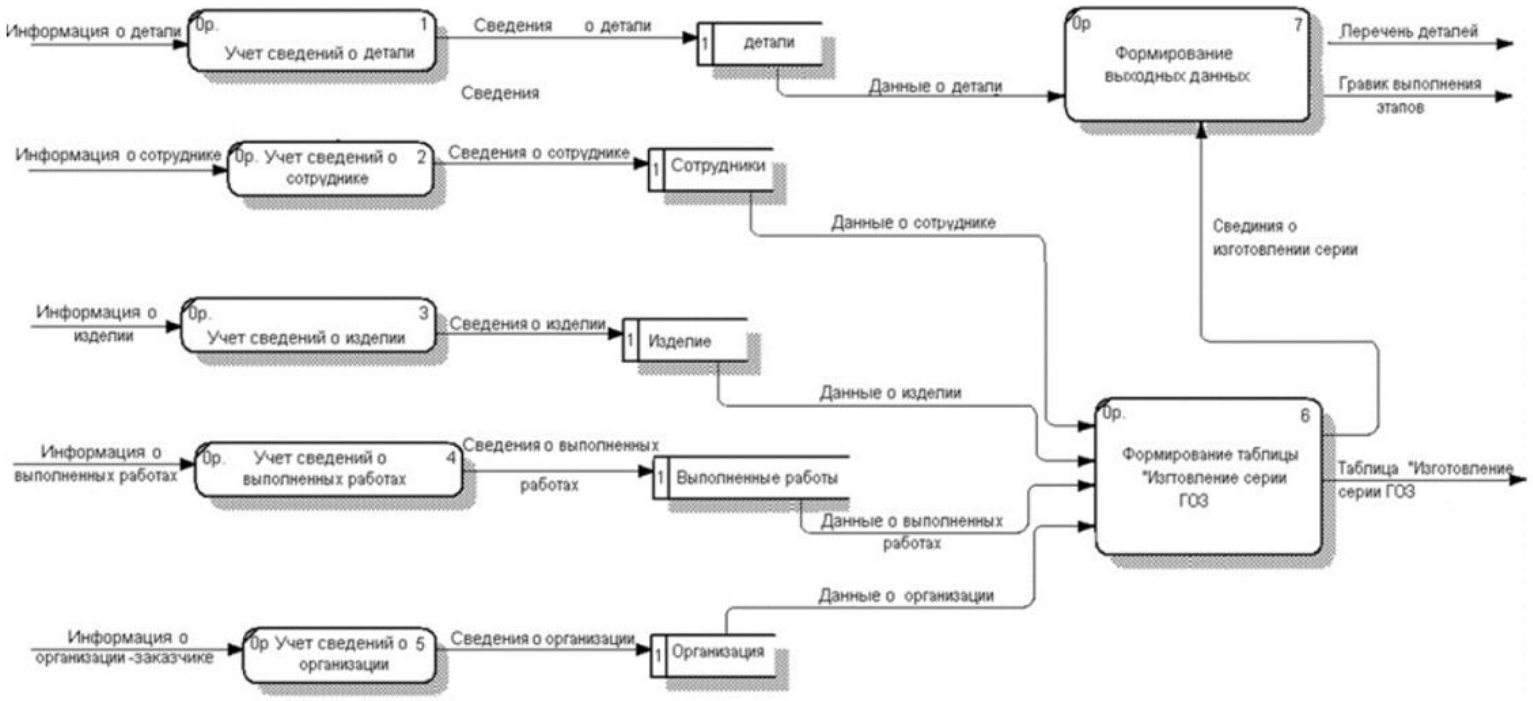
UML-диаграмма развёртывания



UML-диаграмма последовательностей



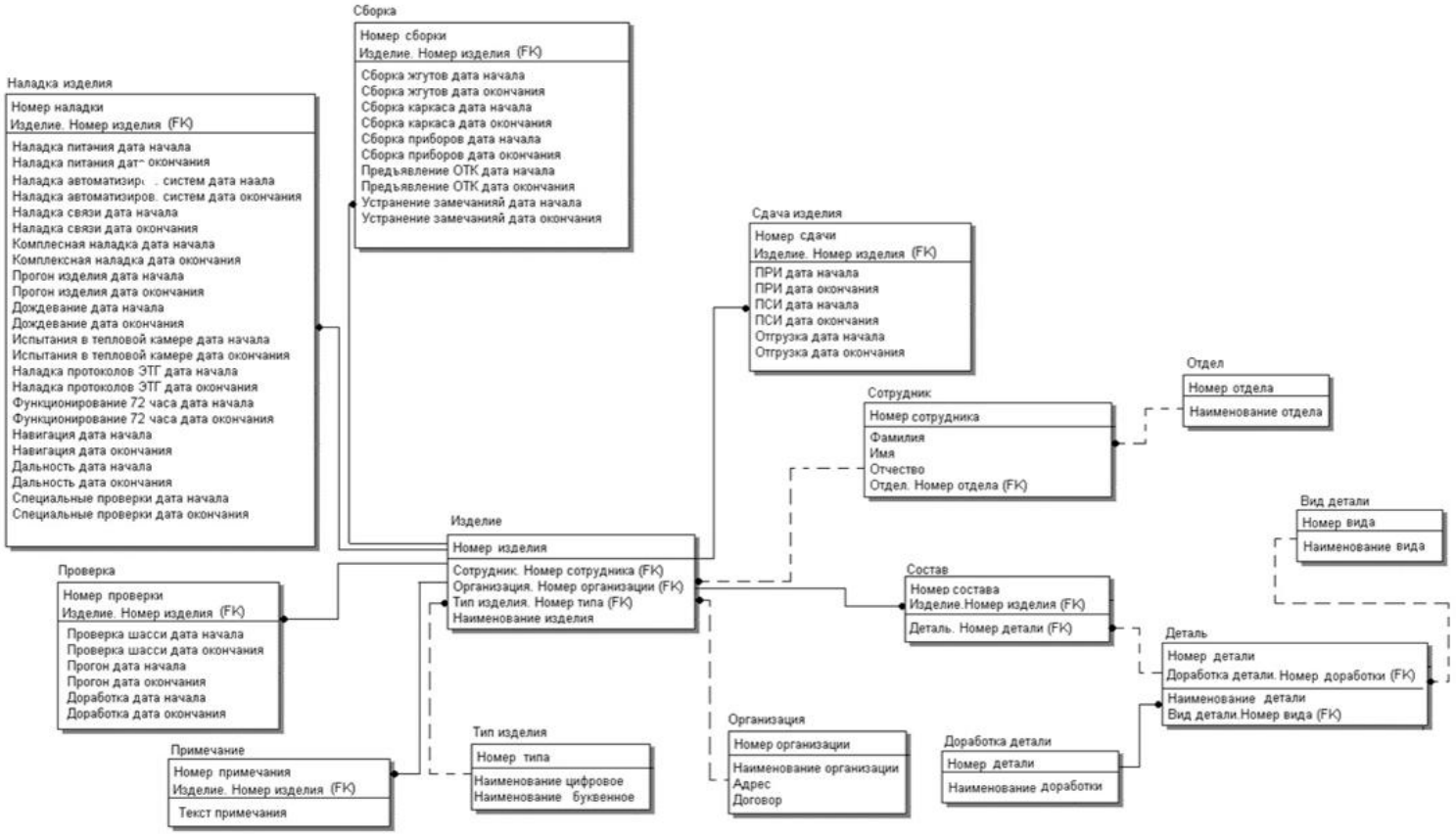
Контекстная диаграмма



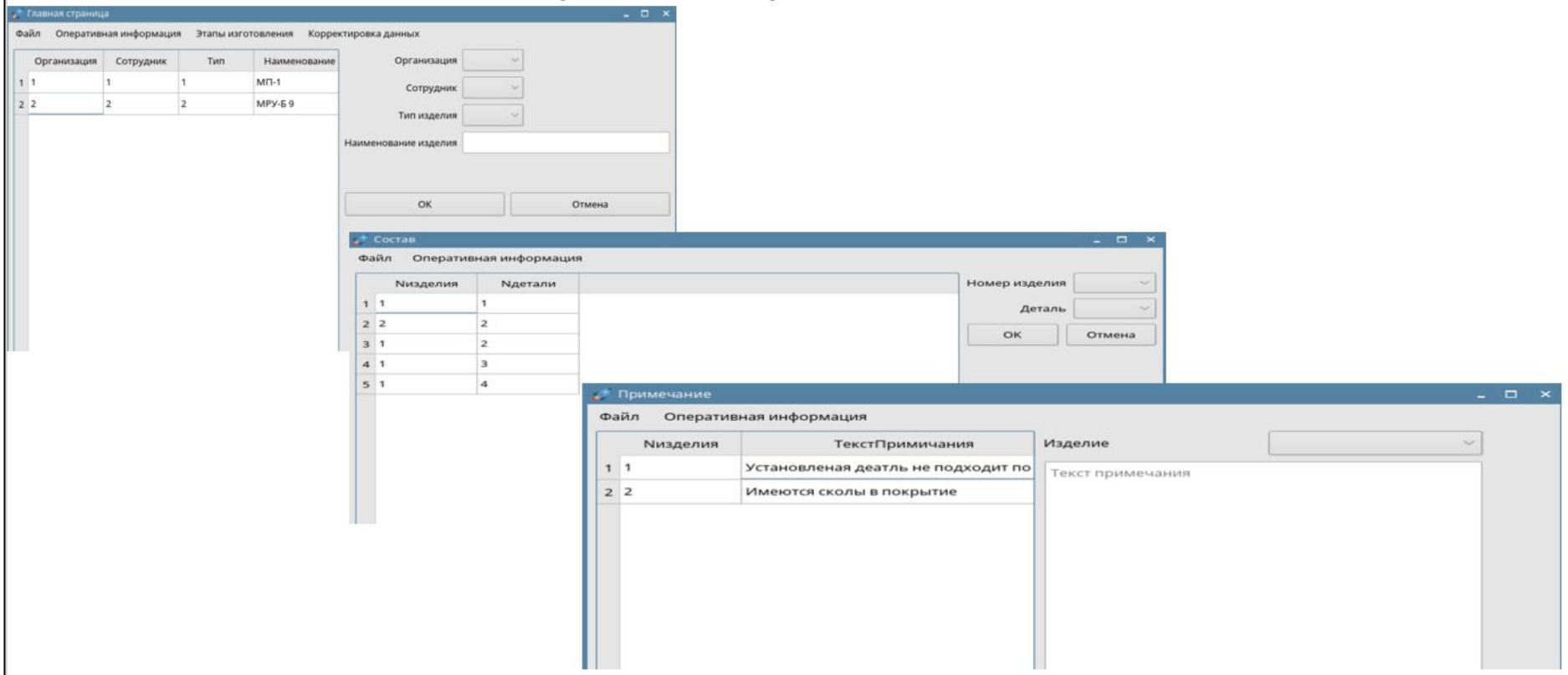
7

ПГУ1.090301.07.001 ПЗ

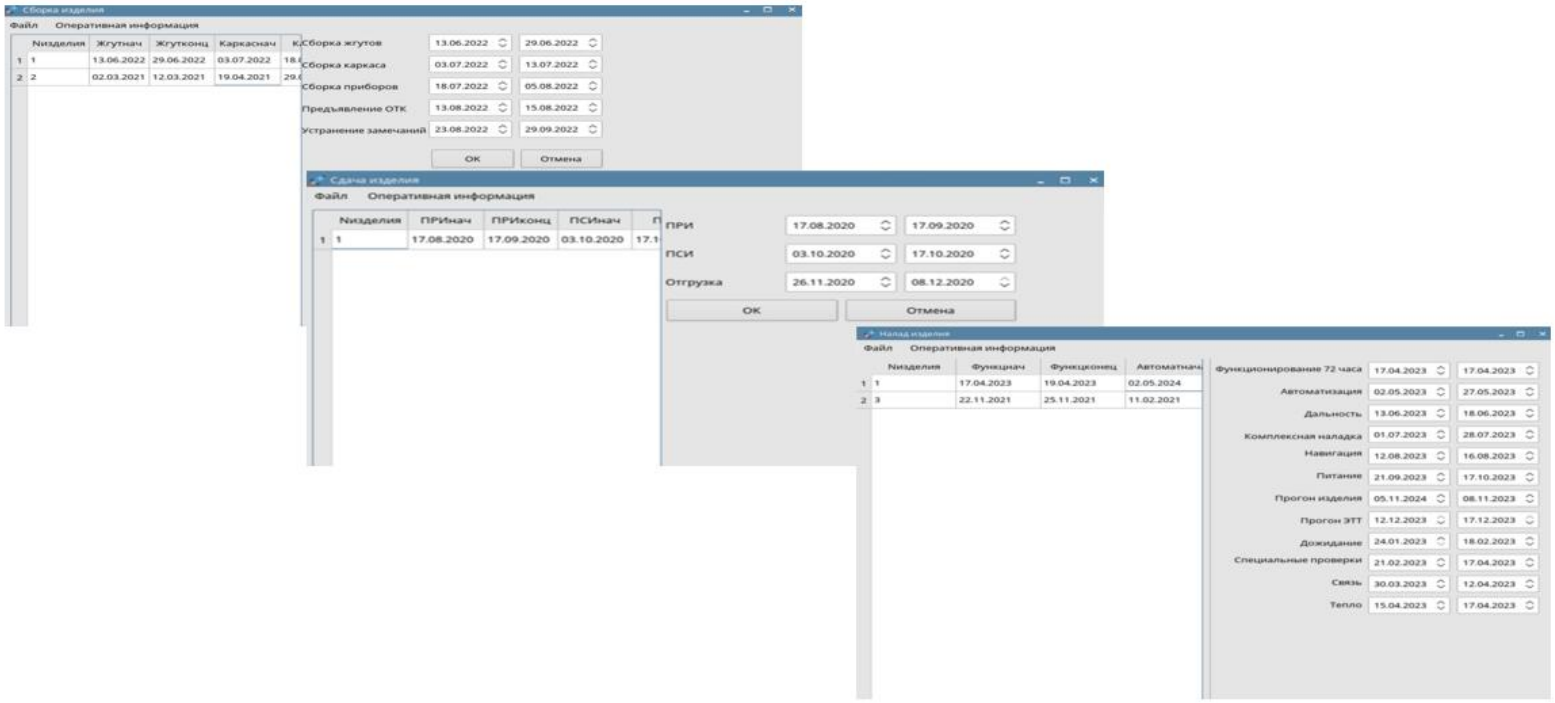
Логическая модель данных



Экраны приложения



Экраны приложения



Результаты работы

В результате проделанной работы было разработано и отлаженопрограммное обеспечение автоматизированной системы контроля изготовления изделий специального назначения, позволяющая обрабатывать и хранить данные о сотрудниках, изделиях, деталях от стороннего производителя, организациях заказчиках, этапах изготовления изделий и т.д. Была разработана модель данных системы на основе которой спроектирована структура базы данных, разработано программное обеспечение автоматизированной системы, отлажено и протестировано.
Проведено тестирование показавшее, что система работает верно, без системных сбоев.
Таким образом, заданиена дипломное проектирование выполнено в полном объеме.