ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения очная

«К ЗАЩИТЕ ДОПУЩЕН(А)»
Заведующий кафедрой

(подпись)
Черняева С. Н.
(ФИО)
2024

Выпускная квалификационная работа

Обучающегося Килимнюк Антона Олеговича
(фамилия, имя, отчество)
Вид работы выпускная квалификационная работа бакалавра
(выпускная квалификационная работа бакалавра, специалиста, магистра)

Пояснительная записка

Тема Разработка АИС распределения работ
(на примере АО «Кольская ГМК»)
(полное название темы квалификационной работы, в соответствии с приказом об утверждении тематики ВКР)

Руководитель работы к. т. н., доцент Матыцина И. А.
(должность, подпись, фамилия, инициалы, дата)
Консультант
(при наличии) (должность, подпись, фамилия, инициалы, дата)
Консультант
(должность, подпись, фамилия, инициалы, дата)
Обучающийся Килимнюк А. О.
(подпись, фамилия, инициалы, дата)

Воронеж
2024

ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения очная

УТВЕРЖДАЮ
Заведующий кафедрой
_____
(подпись)
Черняева С.Н.
(ФИО)
_____ 2024

Задание
на выпускную квалификационную работу

Вид работы ВКР бакалавра
(ВКР бакалавра, ВКР специалиста, ВКР магистра)

Обучающемуся Килимнюк Антону Олеговичу
(фамилия, имя, отчество)

Тема Разработка АИС распределения работ (на примере АО «Кольская ГМК»)

Утверждена приказом ректора университета от _____ 20___, № _____

Срок сдачи законченной работы _____ 20___

Исходные данные (или цель ВКР):
Целью данной выпускной квалификационной работы является изучение существующих методов распределения работ, при помощи автоматизированной информационной системы или руководителя проекта, а также разработка собственной АИС распределения работ

Перечень подлежащих исследованию, разработке, проектированию вопросов (краткое содержание ВКР)
(актуальность темы, цели и задачи ВКР; аналитический обзор литературных источников; постановка задачи исследования, разработки, проектирования;

содержание процедуры исследования, разработки, проектирования; обсуждение результатов; дополнительные вопросы, подлежащие разработке; заключение – выводы по работе в целом, оценка степени решения поставленных задач, практические рекомендации; и др.)

– Введение.Актуальность выбранной темы, цель и задачи ВКР, объект и предмет

(наименование вопроса, раздела и его краткое содержание)

– Исследовательский раздел. Обзор литературы и научных исследований

(наименование вопроса, раздела и его краткое содержание)

– Проектный раздел. Разработка и внедрение АИС

(наименование вопроса, раздела и его краткое содержание)

– Заключение. Выводы по работе в целом. Оценка степени решения поставленных задач

(наименование вопроса, раздела и его краткое содержание)

Практические рекомендации

Перечень графического материала (или презентационного материала):

1. Титульный лист
2. Цель и задачи ВКР
3. Описание предметной области
4. Проектирование АИС
5. Проектирование АИС (продолжение)
6. Процесс разработки информационной системы
7. Интерфейс пользователя
8. Будущие перспективы данного направления
9. Результаты ВКР

Консультанты по разделам ВКР (при наличии):

1. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

2. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

3. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

Дата выдачи задания: _____ 20___

Задание согласовано и принято к исполнению: _____ 20___

Руководитель ВКР: к. т. н., доцент Матыцина И. А. _____

(должность, ученая степень, ученое звание, ФИО) (подпись)

Обучающийся: ИТ-4, Килимнюк Антон Олегович _____

(учебная группа, ФИО) (подпись)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность данной работы заключается в том, что на данный момент автоматизированные системы применяются во многих отраслях для автоматизации различных процессов производства. При увеличении количества задач и проектов данные системы становятся всё более востребованными. Автоматизированные информационные системы распределения работ помогают в разделении труда, упрощении работы и сокращении издержек, связанных с распределением.

Целью данной работы является изучение предыдущих АИС распределения работ, а также схожих систем для определения необходимого функционала для разработки востребованной автоматизированной информационной системы. Для достижения цели дипломной работы были поставлены следующие задачи: изучить процесс распределения работ и выявить основные проблемы, и сложности, с которыми сталкиваются работники в организации; проанализировать существующие автоматизированные информационные системы, которые применяются для управления рабочим процессом; спроектировать структуру и функционал АИС, учитывая потребности сотрудников; разработать ПО, интерфейс для пользователя и базу данных; тестирование и отладочный процесс разработанной АИС; внедрение и тестирование системы; оценка эффективности системы; проведение анализа и предложение рекомендаций по дальнейшему улучшению системы.

Объектом в данной дипломной работе «Разработка АИС распределения работ» является информационная система, её функционал, принцип работы, а также структура. Предмет исследования – процесс распределения работ, методы оптимизации данного процесса и разработка программного обеспечения для автоматизации процесса.

1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1. Основы распределения работ

В большинстве ИТ-компаний существует проблема нормирования и распределения работ среди сотрудников. Данный процесс происходит на различных этапах работы над проектом, начиная с процесса разработки и внедрения, заканчивая процессами сопровождения и развития [1]. Оценка уровня производительности, а также компетенций сотрудников, которые принимают участие в решении важных задач, является субъективной. Причиной субъективности является сложность оценки эффективной работы и компетенций работника относительно проекта, а также пристрастностью оценки его мотивации [14].

В проблеме мотивации существует два основополагающих фактора – доход, получаемый сотрудником с учётом выполненной работы, а также карьерный рост, который показывает повышение уровня компетенций и показателя профессионализма, доверие руководства и коллег [5].

Мотивированные на достижение отличного результата, очень важной частью является самореализация в своей профессиональной деятельности [1]. Часто могут возникать ситуации, в ходе которых сотрудник делает всё, чтобы «заявить о себе» в процессе работы, показывая свои навыки, что приводит к проблеме объективности распределения работ [6].

Работники, которые имеют высокий уровень знаний, умений, навыков и желающие продвижение по «карьерной лестнице» получают большее количество работ с более высоким уровнем мотивации [3].

Работники, которые имеют высокий уровень компетенций в ИТ-компаниях, ощущают себя перегруженными и ссылаясь на теорию Джевонса, удовлетворение потребности будет приводить к уменьшению мотивированности достижения. Это в свою очередь приводит к снижению темпов развития системы и реализации поставленных задач [2].

В IT-компаниях на данный момент ещё актуальна проблема неправильности распределения работы среди сотрудников, возникая на одном этапе проекта, она оказывает существенное влияние на последующие. Самое значимое влияние неправильного распределения можно наблюдать на этапе сопровождения проектов, в тот момент, когда исполнитель коммуницирует с человеком, использующим данный продукт [13].

Чтобы решать такого рода задачи применяется алгоритм теории расписаний Джонсона – правило, которое задаёт порядок запуска изделий в обработку.

Данный алгоритм, строящийся на основе правила, который обеспечивает оптимальность загрузки рабочих центров, уменьшает продолжительность производственного цикла [10].

Многие авторы рассматривали проблему распределения работ между исполнителями. Например, некоторые пытались рассмотреть инновационный подход, в основе которого была работа нейронных сетей, регулирующих загрузку ресурса/оборудования. В данной работе применялись нейронные сети двух типов:

– Многослойный персептрон;
– Сеть Кохонена.

Методология, которая используется в IT-компаниях позволяет применять «жадные» алгоритмы для того, чтобы принимать более оптимальные решения на локальном уровне в процессе реализации проекта. Самый большой эффект данного подхода будет проявляться для этапа технической поддержки [1].

Также были авторы, пытающиеся оптимально распределить работы среди сотрудников, когда процесс разнесён на простейшие задачи указанной продолжительности, а интервал планирования известен [16].

Основной смысл, который можно вынести из работ авторов – подход алгоритма заключается в распределении работы тому исполнителю, который сможет выполнить работу, учитывая необходимую скорость и качество в рамках данной работы. Проблема такого подхода заключается в том, что при

его применении на производстве один сотрудник получает набор всех задач. С течением времени и работы такого сотрудника в компании он становится тяжело или вовсе незаменимой её частью, что в свою очередь создаёт проблемы для производственных процессов. Решение такого рода задачи требует формирования группы специалистов и распределения задач среди них, что даёт возможность формирования необходимых навыков у работников. Это достигается за счёт ведения базы знаний специалистов, имеющих достаточно высокий уровень компетенций, а также с помощью взаимодействия сотрудников в кросс-функциональных командах [12, 18].

Обычно в компаниях решение о распределении каких-либо работ среди сотрудников принимается руководителем проекта или проектной группы.

Процесс распределения должен осуществляться на основании уровня знаний и умений сотрудников, их работоспособности, а также нагрузки и мотивации [11]. Важным аспектом является наличие сотрудников различных уровней компетенции [19]. Также существует проблема, связанная с принятием решений, т.к. это решение принимает руководитель проекта, то он может не учесть или учесть, но не в полной степени уровень загруженности, компетентности, многозадачности и т.д., в связи с этим появляется вероятность неверного распределения [15].

Один из авторов предлагает систему распределения между исполнителями в рамках алгоритмов, учитывающих не-факторы: нечётко-продукционную модель, а также подход основывающийся на методе Г.С. Поспелова.

Данные предстают в виде правил, которые доступны для трактовки экспертами. Важной задачей при синтезе такого рода систем является корректная идентификация базы нечётких правил. Такие системы лежат в основе машинного обучения, моделирование систем, распознание образов.

Система в рамках нечётко-продукционной модели, предназначенная для распределения работ между сотрудниками должна иметь в себе алгоритм

воспроизводящий рассуждения работника с высоким уровнем компетенций – профессионала при выборе из n альтернатив назначения $A = \{A_1, A_2, \dots, A_j, \dots, A_n\}$. Выбор осуществляется с учётом m критериев $K = \{k_1, k_2, k_3, \dots, k_m\}$, которые определяются за счёт предпочтений человека, принимающего решения. В каждом случае из альтернатив A_j, обязательно должно быть известно значение показателей pk соответствующих критериям $PK_j = \{pk_{j1}, pk_{j2}, \dots, pk_{jm}\}$. К ним относят уровень загруженности, наличие у работника нужного уровня знаний и умений, опыта, а также работоспособность. Оценивание производится человеком, принимающим решения и основывающимся на субъективности анализа, и нечётких знаниях [1].

1.2. Существующие АИС для управления задачами и работами, преимущества и недостатки

В компании «Кольская ГМК» применяются следующие системы для распределения работ и управления задачами:

- «Регистратор неисправностей»;
- «Sap ERP»;
- «Сириус».

Программное обеспечение «Регистратор неисправностей» было разработано в компании АО «Кольская ГМК» в городе Мончегорск для распределения заявок по сотрудникам для исправления и/или ремонта.

Ниже представлена блок-схема регистратора неисправностей (рисунок 1.1).

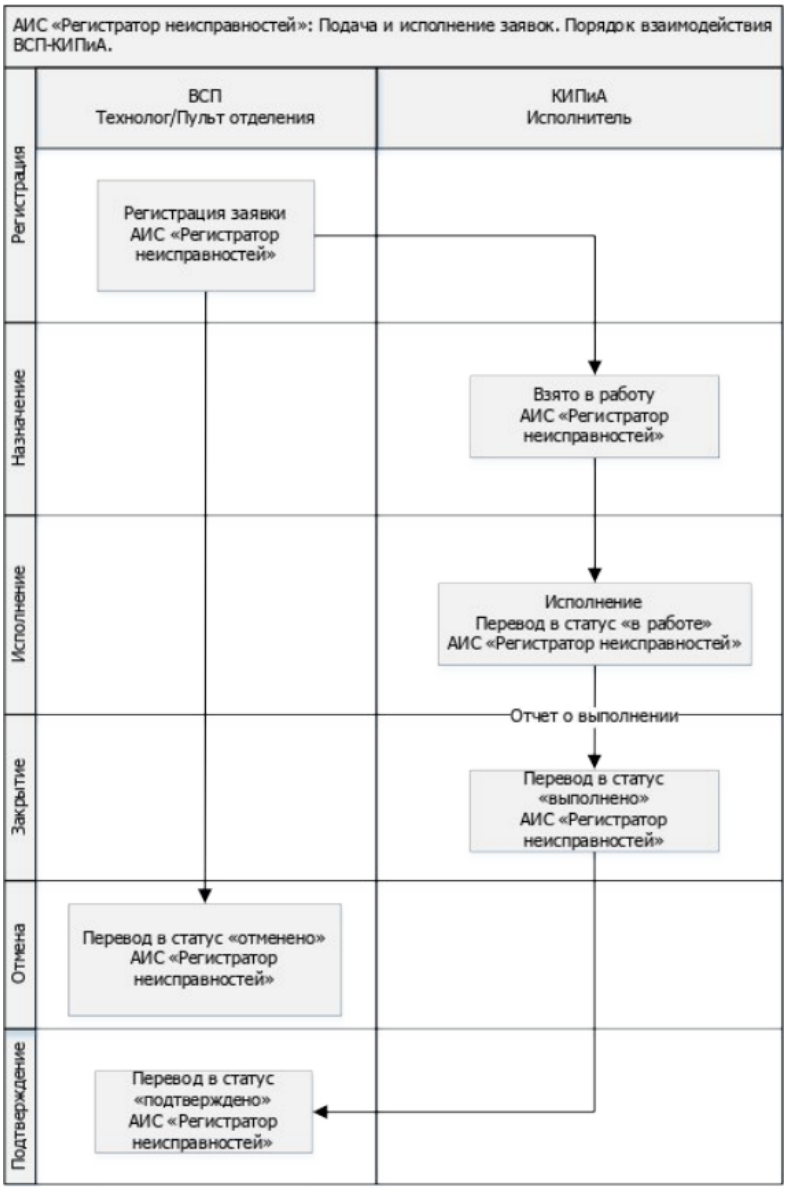


Рисунок 1.1 – Блок-схема взаимодействия ВСП и КИПиА

Далее показано взаимодействие пользователя с программой «Регистратор неисправностей» (рисунок 1.2)

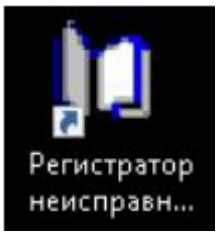


Рисунок 1.2 – Ярлык системы «Регистратор неисправностей»

После запуска программного обеспечения путём двойного нажатия левой кнопки мыши по ярлыку перед пользователем открывается форма для входа в свою учётную запись (рисунок 1.3).

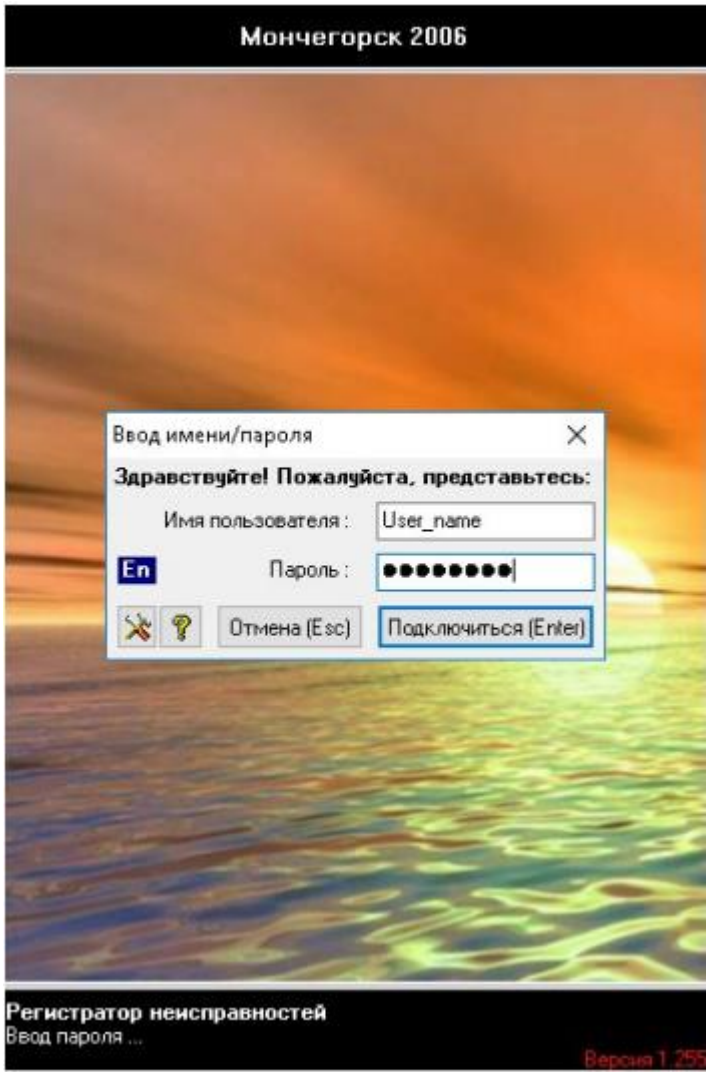


Рисунок 1.3 – Форма входа пользователя в свою учётную запись

После входа перед сотрудником открывается окно со списком заявок, а также кнопки и поля ввода используемые для создания заявок (рисунок 1.4).

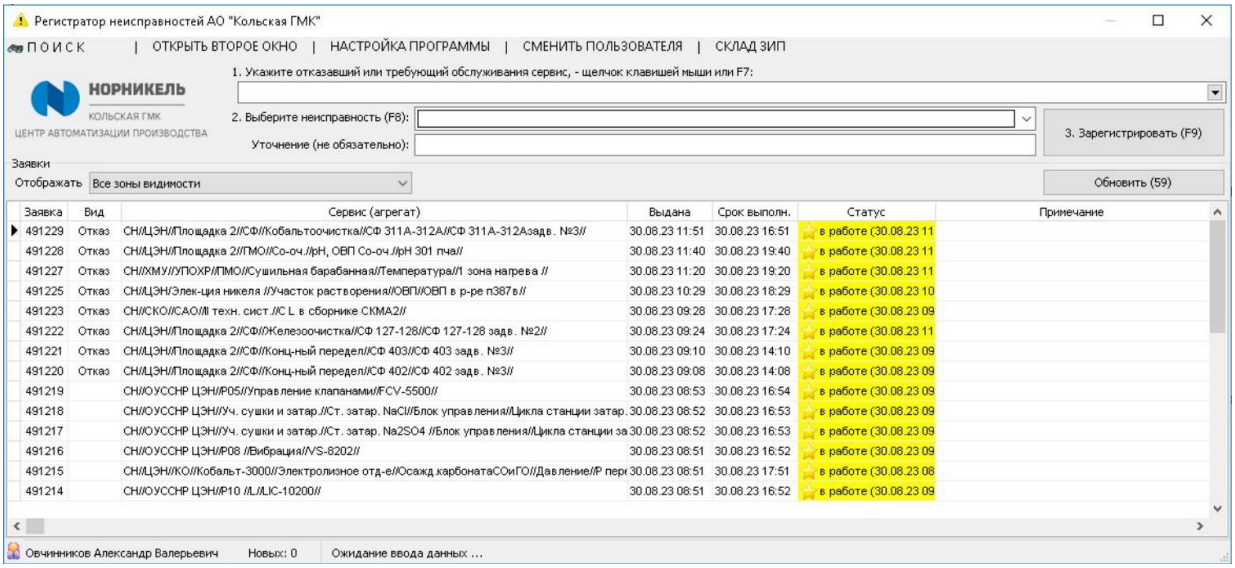


Рисунок 1.4 – Панель навигации

Для понимания работы отдельных элементов управления существует таблица описания элементов интерфейса (таблица 1.1).

Таблица 1.1 Описание элементов интерфейса.

Графическое изображение	Назначение
ПОИСК	Кнопка поиска заявки по номеру
ОТКРЫТЬ ВТОРОЕ ОКНО	Кнопка вызова второго окна
НАСТРОЙКА ПРОГРАММЫ	Кнопка перехода в меню настроек
СМЕНИТЬ ПОЛЬЗОВАТЕЛЯ	Кнопка смены пользователя
СКЛАД ЗИП	Кнопка перехода в базу ЗИП
1. Укажите отказавший или требующий обслуживания се 2. Выберите неисправность (F8): Уточнение (не обязательно):	Область создания новой заявки
3. Зарегистрировать (F9)	Кнопка регистрации заявки
Обновить (59)	Кнопка обновления
Заявки Отображать Все зоны видимости CH// Заявка Все зоны видимости	Фильтр зон видимости (определяется правами доступа)
Заявка Вид 491229 Отказ CH//ЦЭН//Площадка 2//СФ//Кобал 491228 Отказ CH//ЦЭН//Площадка 2//ПМО//Со-о 491227 Отказ CH//ХМУ//УПОХР//ПМО//Сушильн 491225 Отказ CH//ЦЭН//Элек-ция никеля //Учас 491223 Отказ CH//СКО//САО//II техн. сист.//С L 491222 Отказ CH//ЦЭН//Площадка 2//СФ//Желе	Список заявок

Далее, в таблице описаны статусы заявок:

Таблица 1.2. Статусы заявок

Изображение	Статус	Описание
	новая	зарегистрированная заявка в регистраторе неисправностей
	переадресована	заявка переведена на исполнителя
	принята	исполнитель принял на себя заявку
	в работе	выполнение заявки
	выполнена	заявка выполнена, неисправность устранена
	отложена	невозможность выполнения заявки в отведённый период времени на заявку
	отменена	ложная заявка
	доведена до	эскалация заявки, передача другому исполнителю
	подтверждена	подтверждение выполнения заявки

Также присутствует ролевая модель, в зависимости от того, какая роль у пользователя будет предоставлен соответствующий ей интерфейс.

Таблица 1.3. Ролевая модель.

	Технолог	Исполнитель	Инженер
Модуль заявки	+	+	+
Регистрация заявок	+	-	-
Перевод в статус «Принята»	-	+	+
Перевод в статус «В работе»	-	+	+
Перевод в статус «Отложена»	-	+	+
Перевод в статус «Переадресована»	-	+	+
Перевод в статус «Выполнена»	-	+	+
Перевод в статус «Отмена»	+	-	-
Перевод в статус «Подтверждена»	+	-	-
Перевод в статус «Не подтверждена»	+	-	-
Модуль «Каталог сервисов» (объекты обслуживания)	-	-	+

Для регистрации заявки необходимо выбрать в выпадающем списке сервис или узел (рисунок 1.5).

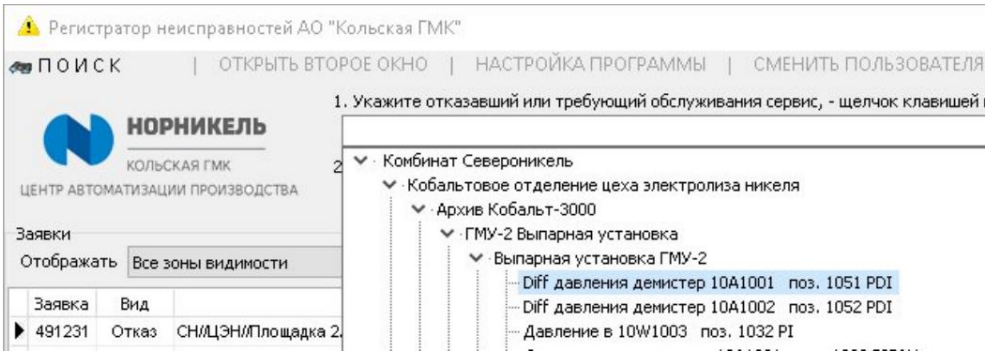


Рисунок 1.5 – Окно выбора списка сервисов (объектов обслуживания)

Затем следует выбрать конкретную неисправность, которая наиболее подходит для описания отказа. В случае отсутствия в списке подходящего вида неисправности – выбрать пункт «неисправность – нет в списке» (рисунок 1.6).

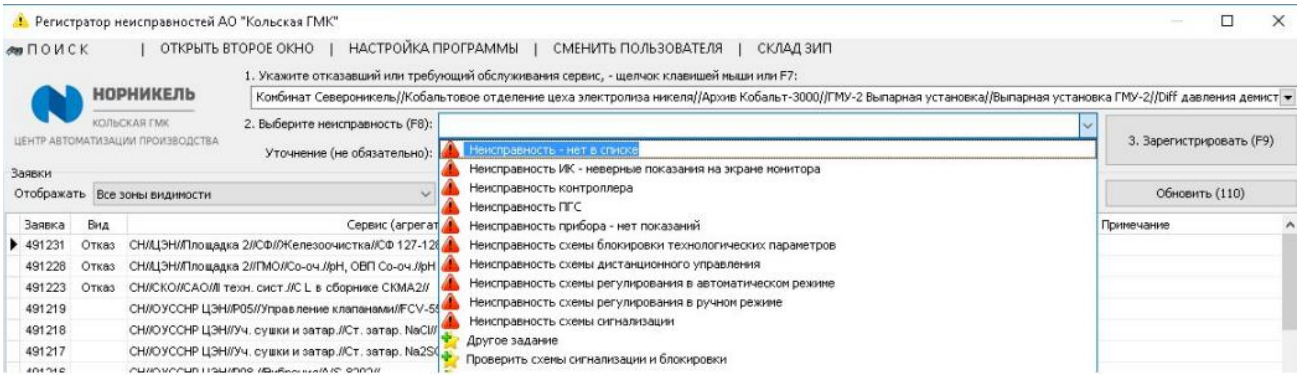


Рисунок 1.6 – Выбор вида неисправности

Также существует поле, в котором необходимо указать комментарий, который описывает проблематику отказа, и зарегистрировать заявку, нажав F9 или кнопку с названием «Зарегистрируйте (F9)» (рисунок 1.7).

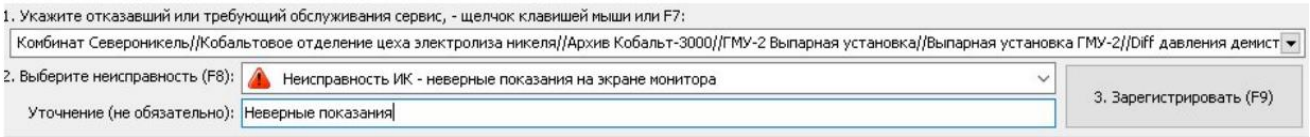


Рисунок 1.7 – Регистрация заявки

До момента регистрации заявки пользователь может вносить изменения или уточнения в заявку, после регистрации внесение каких-либо изменений невозможно.

После регистрации появится запись в верхней части экрана со статусом «новая» (рисунок 1.8).

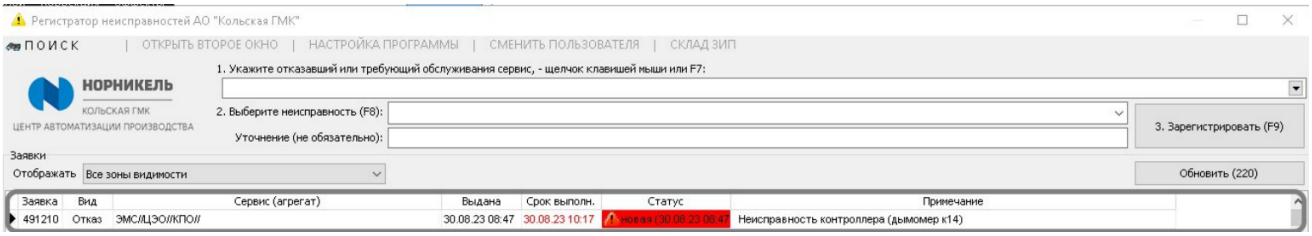


Рисунок 1.8 – Новая заявка

Созданная запись появляется у всех пользователей, имеющих доступ к ней. Появление сопровождается звуковым сигналом. После клика по левой кнопке мыши по заявке будет открыта «Карточка заявки» (рисунок 1.9).

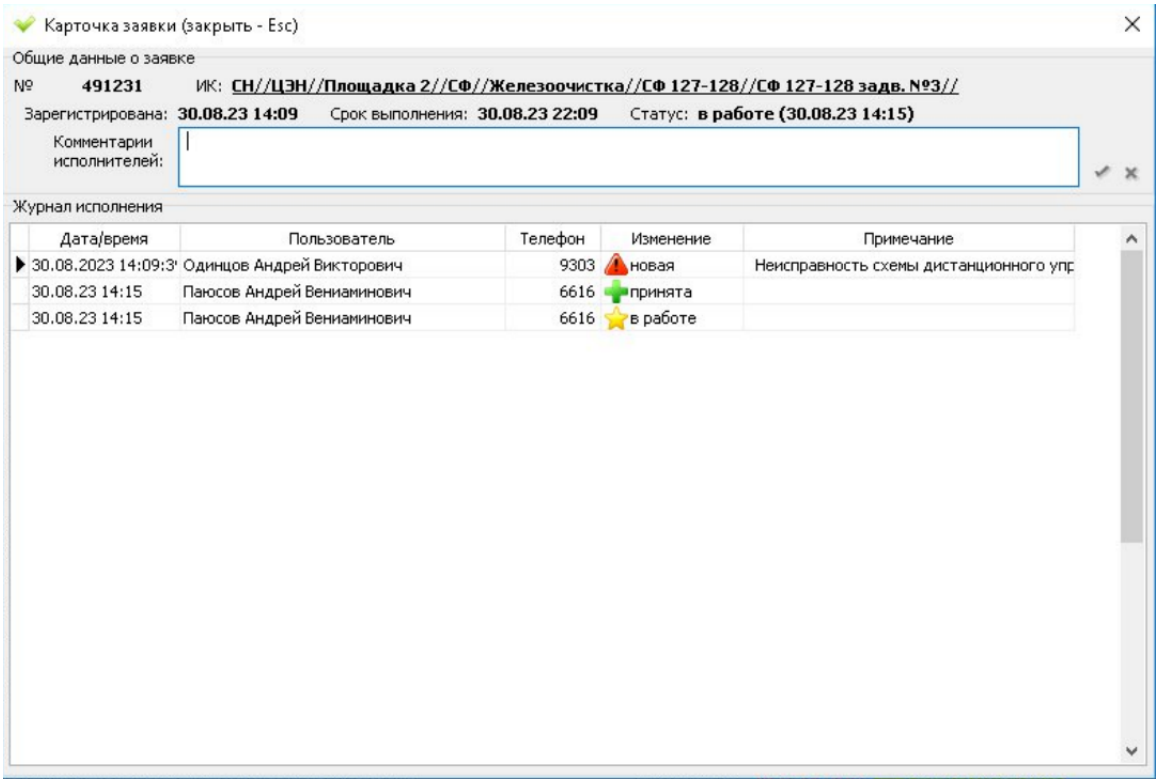


Рисунок 1.9 – Карточка заявки

В карточке заявки ведется история изменений статуса заявки.

Для отмены нажмите ЛКМ по нужной заявке и выберете «Отменяю заявку – выполнение не требуется» (рисунок 1.10).

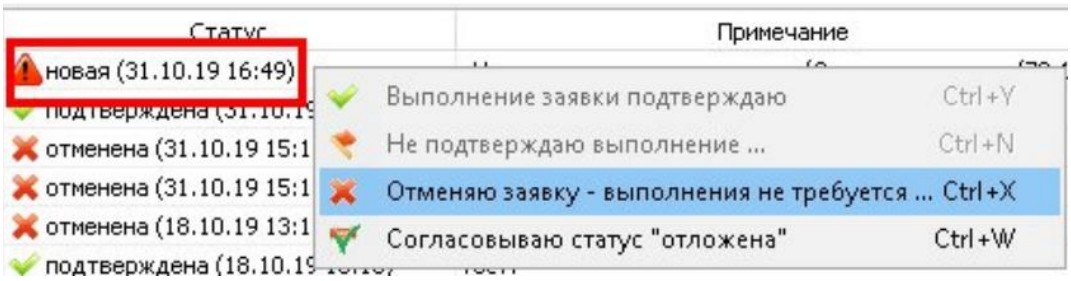


Рисунок 1.10 – Отмена заявки

Заявка получит статус «Отменена» (рисунок 1.11).

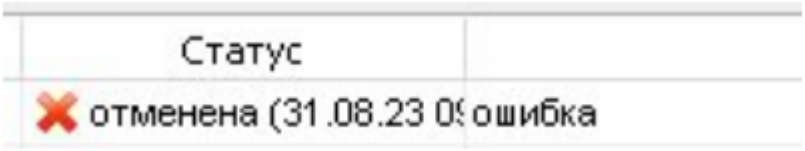


Рисунок 1.11 – Вид заявки при отмене

После устранения неисправности заявка переводится в статус «выполнена». Задача пользователя с ролью Технолог (таблица 1.3) выбрать статус заявки. Для этого требуется нажать ЛКМ по заявке, и в появившемся меню выбрать «Выполнение заявки подтверждаю» или «Не подтверждаю выполнение» (рисунок 1.12).

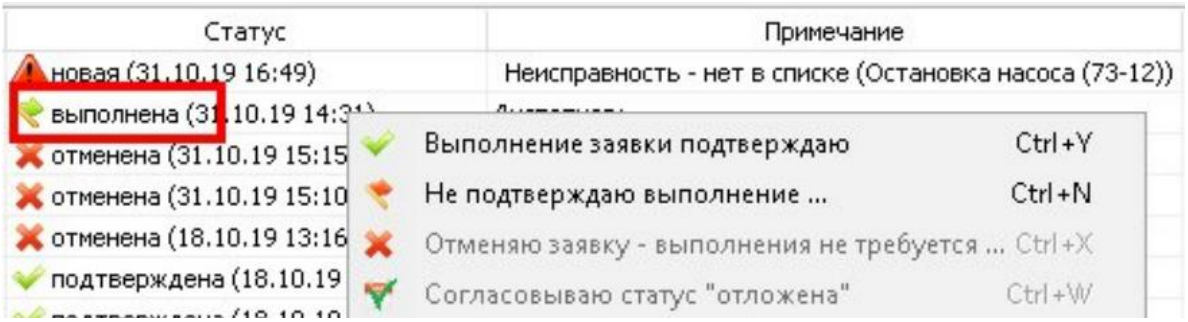


Рисунок 1.12 –Подтверждение заявки

Заявка изменит статус на «Подтверждена» (рисунок 1.13).

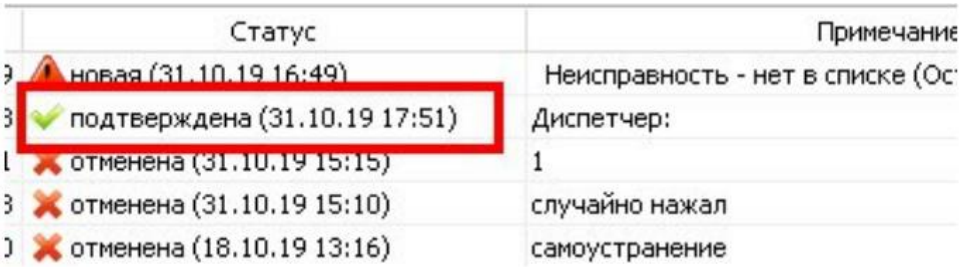


Рисунок 1.13 – Заявка подтверждена

В зависимости от роли (таблица 1.3) карточка заявки имеет разный вид. Ролям Исполнитель и Инженер доступен (рисунок 1.14).

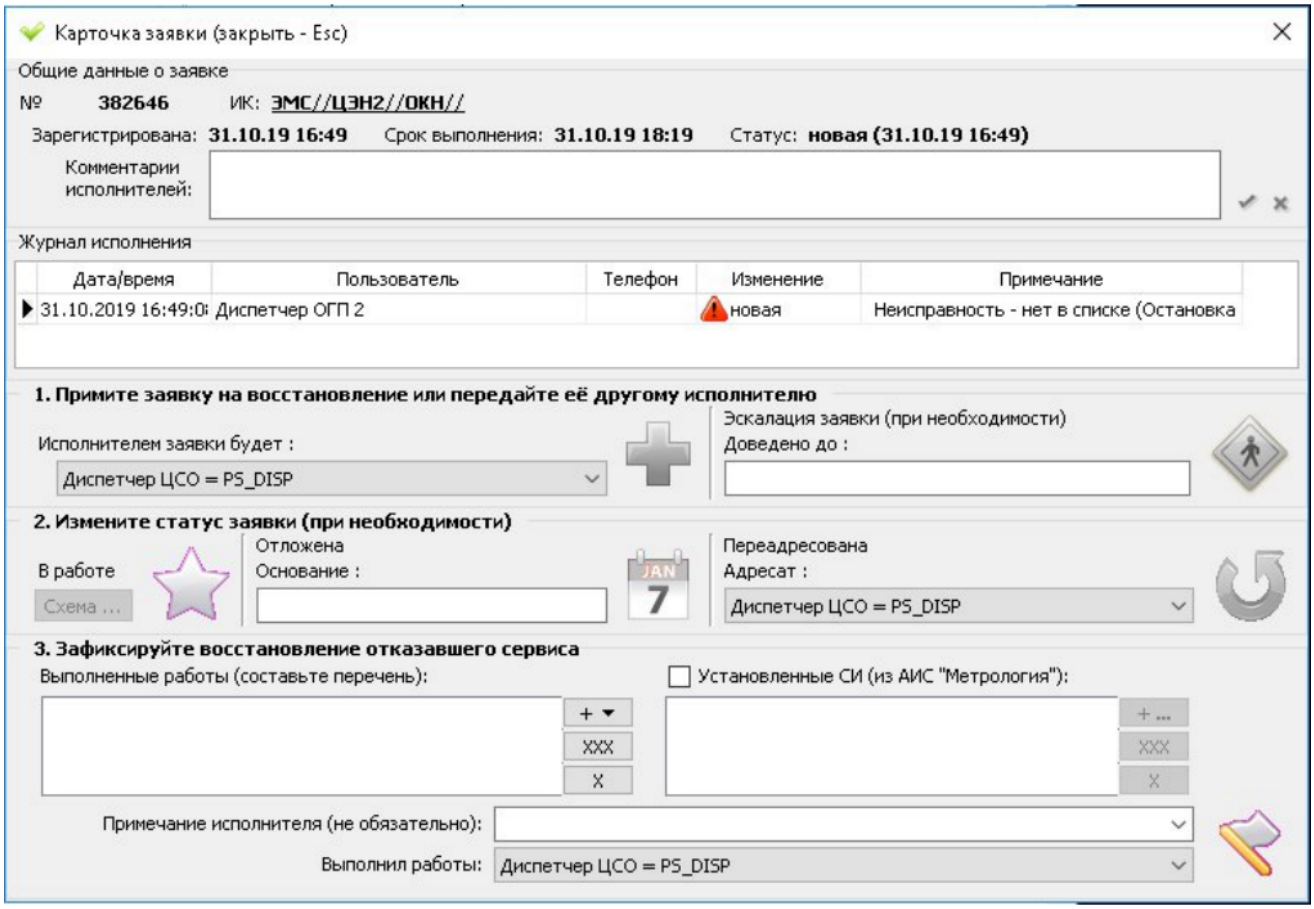


Рисунок 1.14 – Вид карточки заявки у пользователя с ролью «Исполнитель» или «Инженер»

После выполнения работ заявка переводится в статус «Выполнена». Для этого потребуется открыть карточку заявки, заполнить поле «Примечание исполнителя» и кликнуть ЛКМ по кнопке «Зарегистрировать восстановление» (рисунок 1.15).

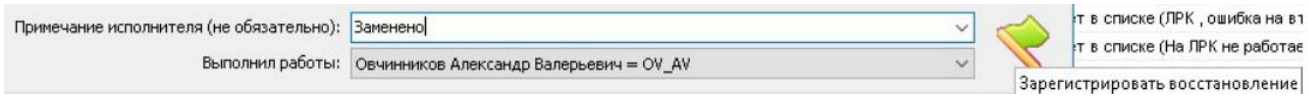


Рисунок 1.15 – Выполнение заявки

Отдел эксплуатации программного обеспечения АСУТП (ЭО ПО АСУТП) сопровождает четыре АИС:

АИС «Теплоузлы» – учёт потребления теплоносителя (горячей воды) и холодной воды объектов, зданий, принадлежащих Кольской ГМК (далее

КГМК), но расположенных удалённо от территории комбината, на территории г. Мончегорска.

АСКУЭ «Альфа Центр» – автоматическая система коммерческого учёта электроэнергии. По показаниям данной системы происходит учёт и оплата потребления КГМК электроэнергии в целом. Система имеет необходимую сертификацию, внесена в реестр ПО, раз в полгода проходит инспекцию контролирующими Гос. органами.

АСТУЭ «Астер2016» – автоматическая система технического учёта электроэнергии. Используется для контроля состояния оборудования и учёта потребления электроэнергии на пл. Мончегорск КГМК. Собственная разработка отдела ПО АСУТП, не имеет строгой сертификации.

АИС «КПО» – автоматизированная система котельно-парового отделения (ТЭЦ) производит сбор показаний датчиков КИП, предоставляет информацию операторам КПО в виде Web-приложения.

В ходе ознакомления с работой отдела, перед началом выполнения индивидуального задания, меня познакомили с функционалом и принципом работы АСТУЭ «Астер2016».

Электроэнергия на КГМК поступает от АЭС (атомной электростанции), расположенные на территории КГМК в количестве 4 штук. На ГПП происходит трансформация высокого напряжения и частоты до стандартных 380В. Далее электроэнергия поступает на РП (распределительная подстанция), расположенные каждая рядом с соответствующим цехом, в количестве 20 штук. От РП происходит питание металлургических агрегатов внутри цеха, питание административно-бытовых помещений цеха. В РП расположено оборудование ЦЭО (цех энергообеспечения) – энергоячейки и счётчики потребления электроэнергии.

Основная задача ЭО ПО АСУТП: получение данных с оборудования РП; обработка информации; предоставление и архивация информации. «Астер2016» выполняет подсчёт электроэнергии, предоставление оператору ЦЭО (цеха

энергообеспечения) показаний положений энергетических ячеек, сигнализирует об аварийных ситуациях.

Бесперебойная работа «Астер2016» обеспечивает контроль за работой оборудования РП, а значит бесперебойную подачу питания агрегатам цеха, минимизацию простоев оборудования путём быстрой локализации проблемы на РП, в конечном счёте – стабильный выпуск продукции КГМК. Бесперебойная работа «Астер2016» обеспечивает правильность учёта потребления электроэнергии цехом, не приводит к срыву взаиморасчётов подрядных организаций и КГМК за потреблённую энергию.

До 2016 года на каждой из подстанций стоял PLC Schneider Micro (Шнайдер Микро) и модем (телефонная линия), также счётчик электроэнергии ЦЭ6823М или его разновидность. В 2016 году произошла модернизация системы, до каждой РП проложили линию связи (оптику), в случае невозможности прокладки линии связи были использованы Wi-Fi мосты, а также поставили современные PLC Schneider M340.

Сотрудниками ЭО ПО АСУТП в 2016 году было произведено конфигурирование каждого PLC в количестве 24, написана логика работы каждого PLC. Организован сбор и архивация данных, для этого потребовалась создание нового сервера, создание виртуальных машин под разные задачи. В «Астер2016» используется 1 центральный сервер, на котором расположены 3 виртуальные машины (сбор данных, архивация данных, среда разработки). Заново создана SCADA-система и выполнено архивирование параметров. Выполнены работы по пуско-наладке (установка PLC, подключение сигналов, проверка отображения в SCADA-систему) на каждой из РП. На данном этапе жизненного цикла системы происходит обслуживание системы, вносятся изменения по заявкам пользователей. Система масштабируема: возможно увеличение РП, увеличение сигналов на РП, возможен опрос оборудования ЦЭО по цифровым интерфейсам, возможно (не реализовано на данный момент) удалённое управление.

Для программирования микроконтроллеров применяется программное обеспечение от производителя Shneider – UnityPro. (рисунок 1.16). Среда программирования UnityPro проверяет программный код на ошибки, при отсутствии ошибок программирования компилирует код программы в бинарный файл, загружает\выгружает исполняемый файл в указанный PLC. Используемый PLC Schneider M340 позволяет вносить изменения online, без перезагрузки PLC, а значит без потери данных, без ложного срабатывания сигнализации. ПО Unity, программирование PLC происходит на виртуальной машине №1 с ролью «разработка» (рисунки 1.16, 1.17).

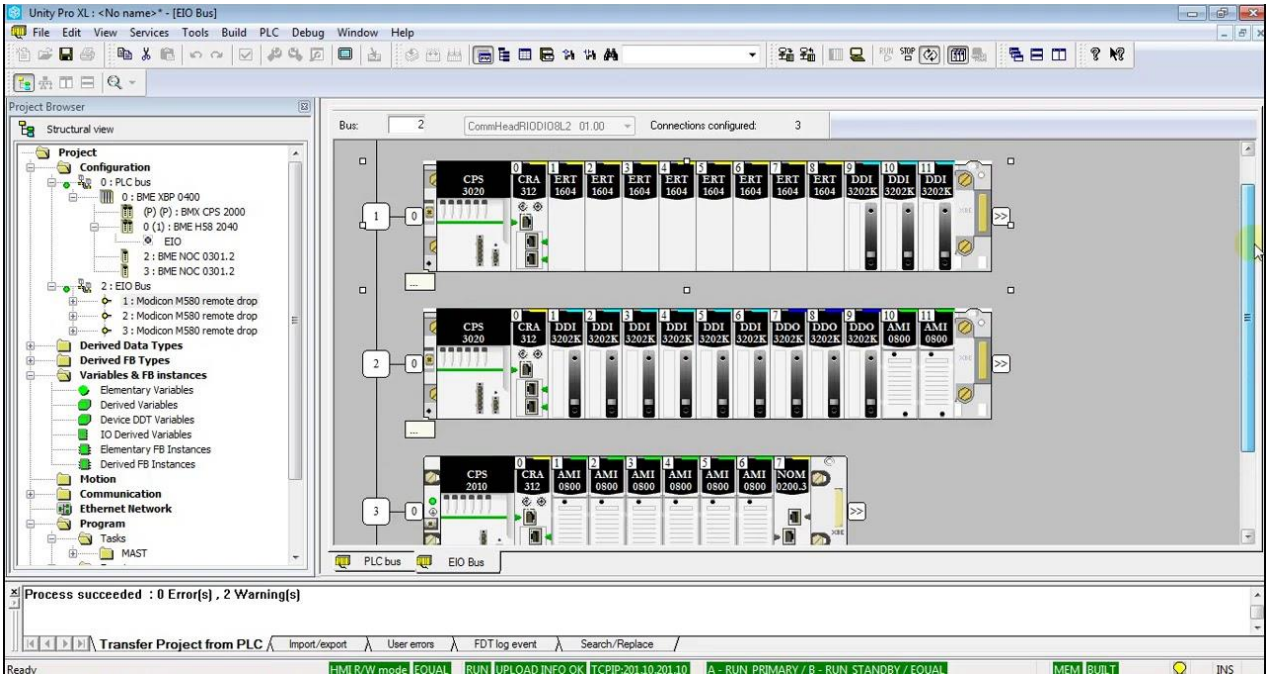


Рисунок 1.16 –Unity PRO XL, параметрирование контроллера

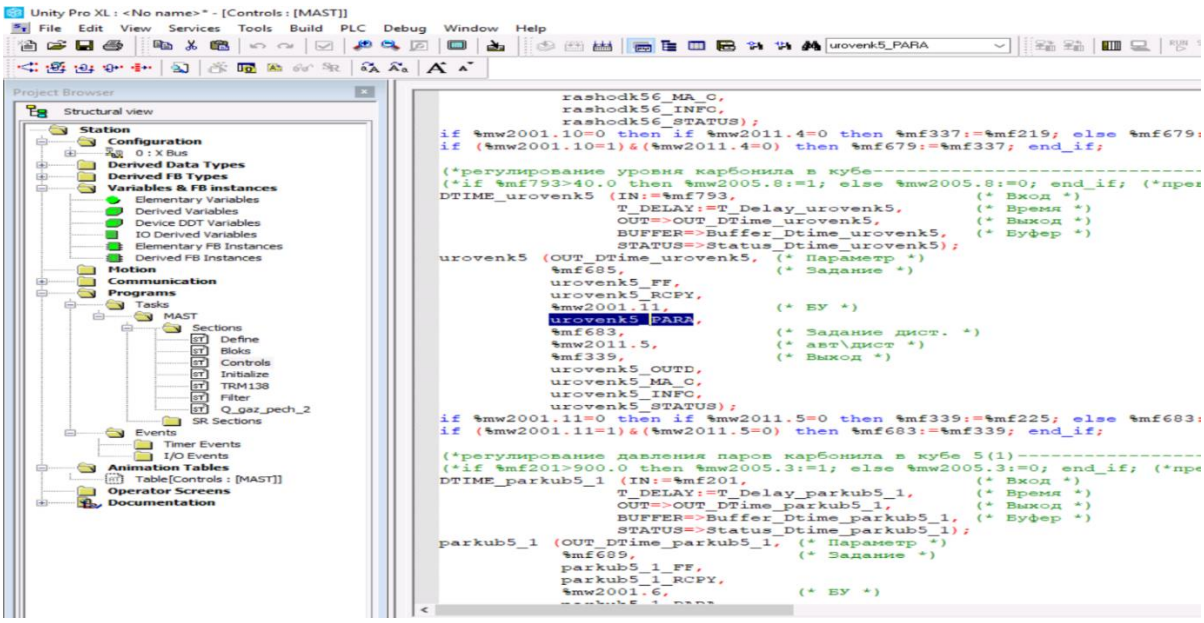


Рисунок 1.17 – Unity Pro XL, написание логики (пример)

После программирования, запуска в работу PLC, необходимо получить данные для отображения их в SCADA и архивации.

Опрос всех PLC производит так же фирменное ПО производителя Sheider – OFS (OPC Factory Server) (рисунок 1.18). В OFS конфигурируется кол-во, тип и адрес каждого PLC. Существует возможность диагностики получения данных от PLC - посмотреть список тегов, посмотреть значение каждого тега. ПО OFS (рисунок 1.18) расположено на виртуальной машине №2 с ролью «сбор данных».

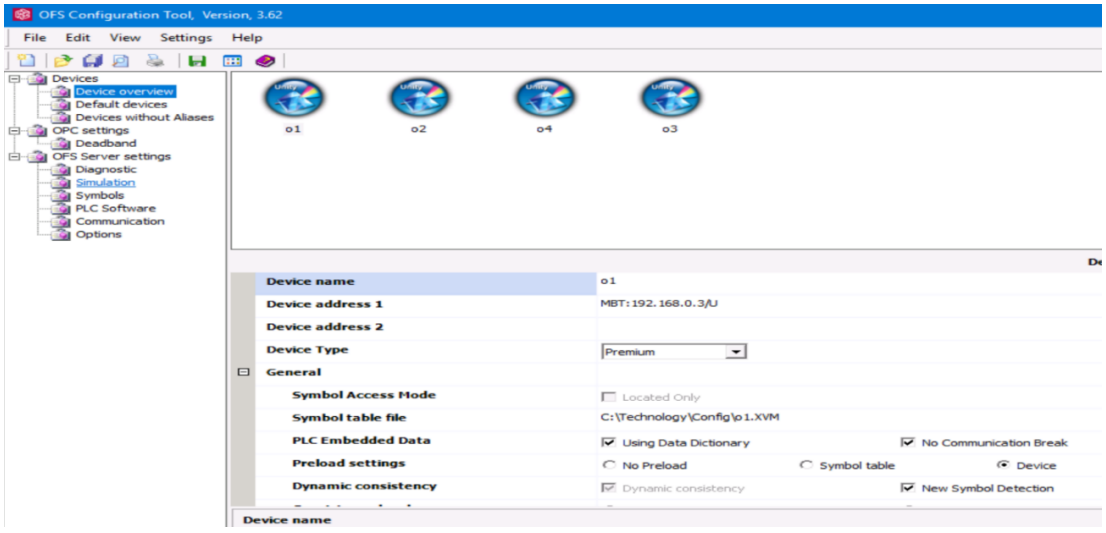


Рисунок 1.18 – OFS

После опроса контроллера, необходимо отобразить её пользователям. Отображение информации происходит в SCADA-систему Intouch 2014R2. Используется клиент-серверная конфигурация SCADA.

Среда разработки Intouch расположена на виртуальной машине №1 с ролью «разработка». В данном ПО создаются, конфигурируются теги, которые получают данные из PLC (через OFS); пишутся скрипты для сигнализации; создаются мнемосхемы. Есть возможность диагностики – запуск клиентской части локально на сервере для проверки корректности написания самой мнемосхемы, кнопок перехода, работы динамических элементов. После внесения изменений есть возможность передать новый проект толстым клиентам средствами самого Intouch автоматически. При запуске ПО клиентом или переходе клиентом между мнемосхемами, появляется уведомление об изменении проекта, происходит скачивание и запуск изменённого ПО (рисунок 1.19).

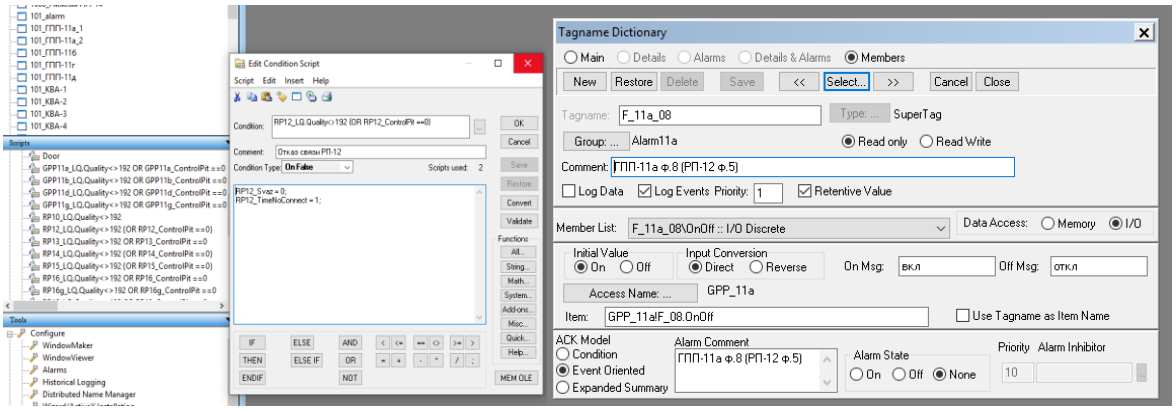


Рисунок 1.19 –Примеры экранов SCADA-системы, написание скрипта, настройка тега

Схема электроснабжения АО «Комбинат Североникель» выглядит следующим образом (рисунок 1.20):

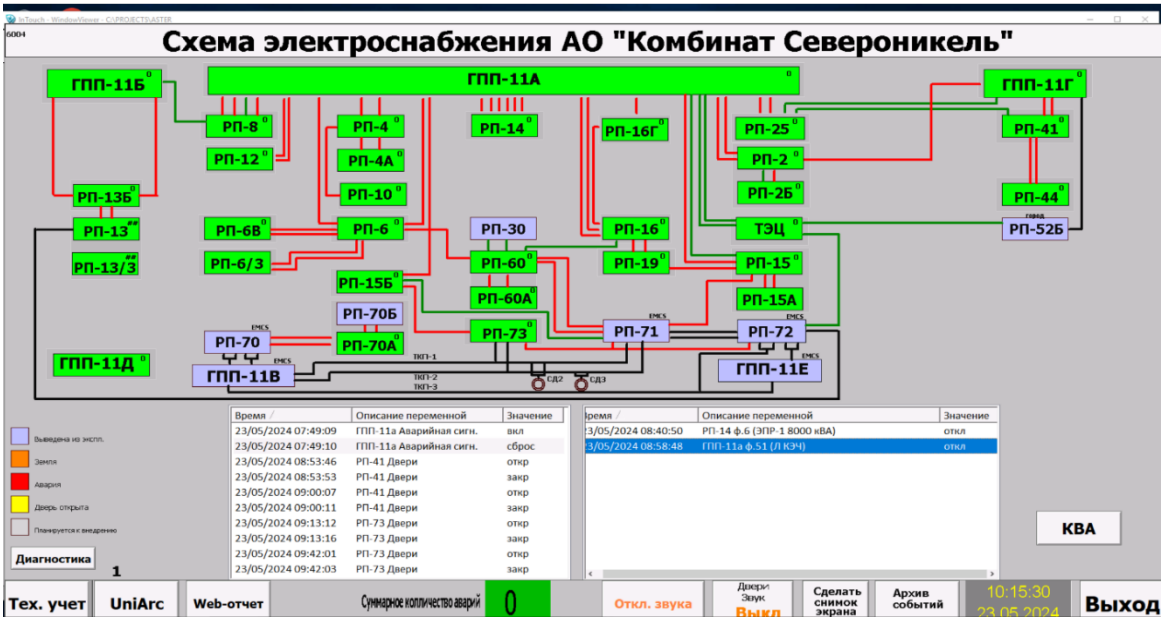


Рисунок 1.20 – Мнемосхема главный экран АСТУЭ «Астер2016»

Ниже представлен скриншот диспетчерского управления, реализованного для удобного управления (рисунок 1.21).

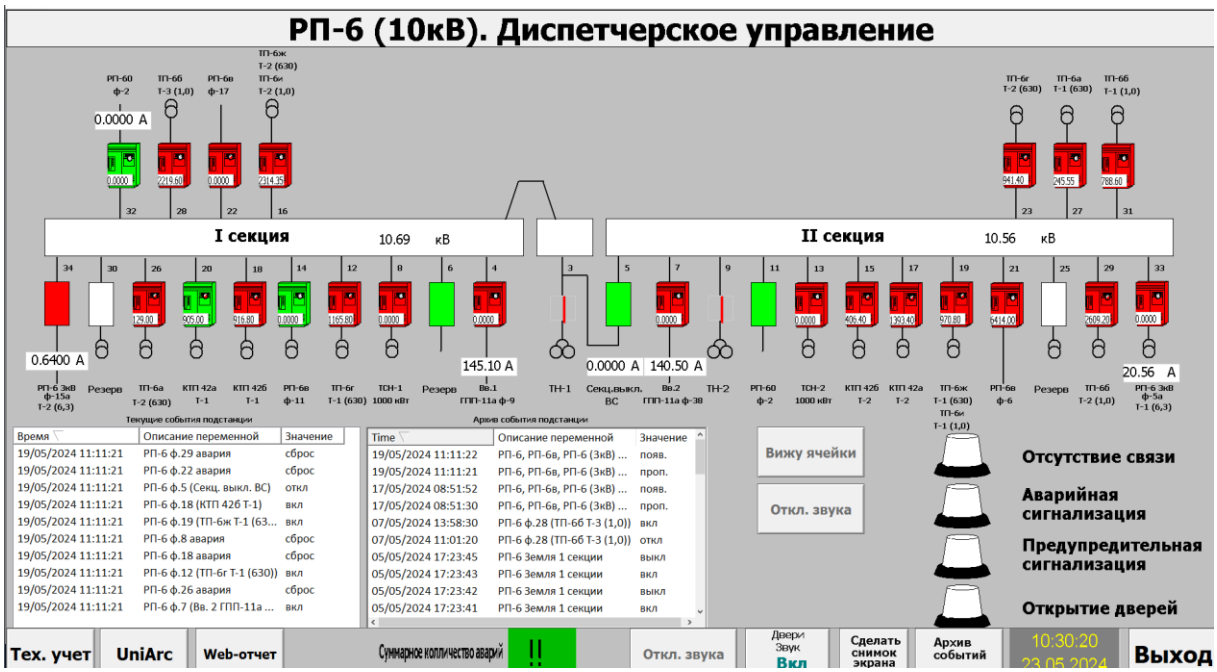


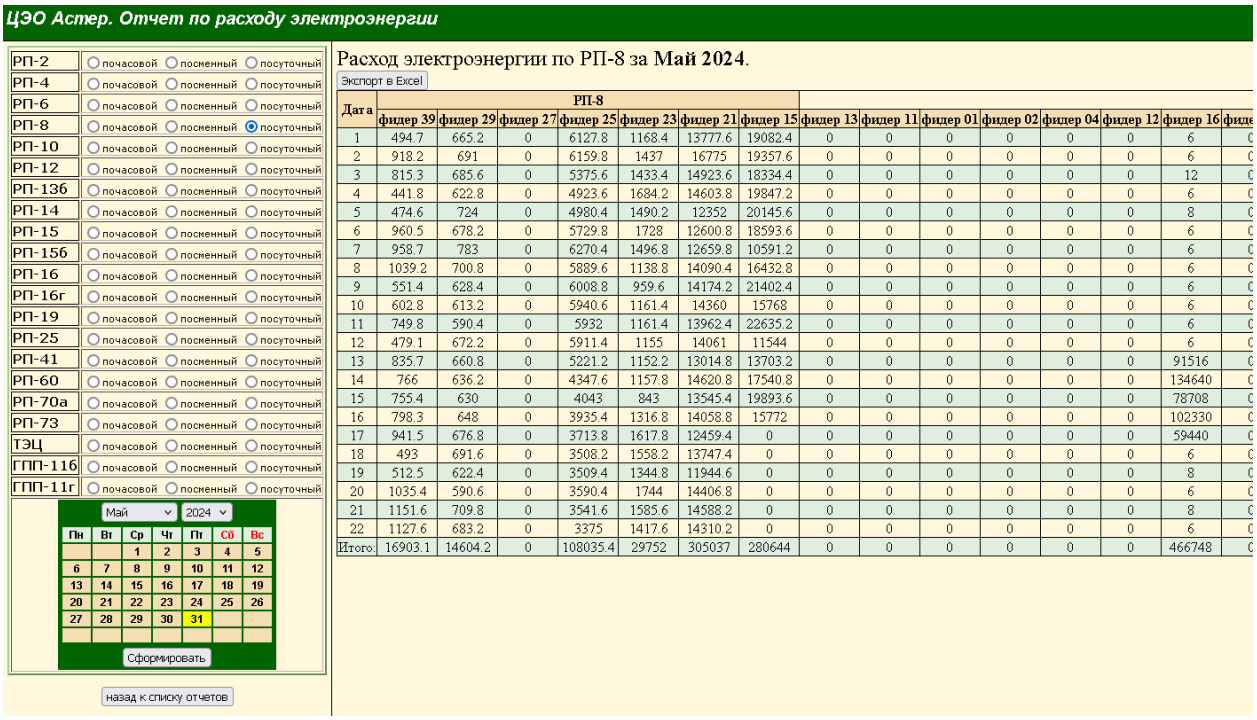
Рисунок 1.21 – Мнемосхема РП-6

С помощью InTouch можно создавать мощные, функционально развитые приложения, использующие основные преимущества операционной системы Microsoft Windows, включая объекты ActiveX, OLE, графические возможности и многое другое. Возможности InTouch могут быть расширены за счет

добавления созданных пользователем элементов ActiveX, мастеров, генерируемых объектов, а также и путем создания Quick-сценариев InTouch.

Приложения InTouch применяются во всем мире в различных отраслях, включая пищевую, бумажную промышленность, производство полупроводников, добычу газа и нефти, автомобилестроение, химию, фармацевтику, транспорт, коммунальные службы и другие отрасли.

Отдельной задачей стоит архивация данных (стандарт хранения данных КГМК составляет 3 года). Для этой цели используется виртуальная машина №3 с ролью «архивация». База данных, используемая ЭО ПО АСУТП во всех проектах – SQL (Structured Query Language). Данные в SQL попадают из OFS. На основании архивированных данных выполнен WEB-отчёт для учёта электроэнергии. Средствами SQL происходит выборка данных по подстанции, разбивка по времени (рисунок 1.22).



Дата	РП-8															
	фидер 39	фидер 29	фидер 27	фидер 25	фидер 23	фидер 21	фидер 15	фидер 13	фидер 11	фидер 01	фидер 02	фидер 04	фидер 12	фидер 16	фиде
1	494.7	665.2	0	6127.8	1168.4	13777.6	19082.4	0	0	0	0	0	0	6	0
2	918.2	691	0	6159.8	1437	16775	19357.6	0	0	0	0	0	0	6	0
3	815.3	685.6	0	5375.6	1433.4	14923.6	18334.4	0	0	0	0	0	0	12	0
4	441.8	622.8	0	4923.6	1684.2	14603.8	19847.2	0	0	0	0	0	0	6	0
5	474.6	724	0	4980.4	1490.2	12352	20145.6	0	0	0	0	0	0	8	0
6	960.5	678.2	0	5729.8	1728	12600.8	18593.6	0	0	0	0	0	0	6	0
7	958.7	783	0	6270.4	1496.8	12659.8	10591.2	0	0	0	0	0	0	6	0
8	1039.2	700.8	0	5889.6	1138.8	14090.4	16432.8	0	0	0	0	0	0	6	0
9	551.4	628.4	0	6008.8	959.6	14174.2	21402.4	0	0	0	0	0	0	6	0
10	602.8	613.2	0	5940.6	1161.4	14360	15768	0	0	0	0	0	0	6	0
11	749.8	590.4	0	5932	1161.4	13962.4	22635.2	0	0	0	0	0	0	6	0
12	479.1	672.2	0	5911.4	1155	14061	11544	0	0	0	0	0	0	6	0
13	835.7	660.8	0	5221.2	1152.2	13014.8	13703.2	0	0	0	0	0	0	91516	0
14	766	636.2	0	4347.6	1157.8	14620.8	17540.8	0	0	0	0	0	0	134640	0
15	755.4	630	0	4043	843	13545.4	19893.6	0	0	0	0	0	0	78708	0
16	798.3	648	0	3935.4	1316.8	14058.8	15772	0	0	0	0	0	0	102330	0
17	941.5	676.8	0	3713.8	1617.8	12459.4	0	0	0	0	0	0	0	59440	0
18	493	691.6	0	3508.2	1558.2	13747.4	0	0	0	0	0	0	0	6	0
19	512.5	622.4	0	3509.4	1344.8	11944.6	0	0	0	0	0	0	0	8	0
20	1035.4	590.6	0	3590.4	1744	14406.8	0	0	0	0	0	0	0	6	0
21	1151.6	709.8	0	3541.6	1585.6	14588.2	0	0	0	0	0	0	0	8	0
22	1127.6	683.2	0	3375	1417.6	14310.2	0	0	0	0	0	0	0	6	0
Итого:	16903.1	14604.2	0	108035.4	29752	305037	280644	0	0	0	0	0	0	466748	0

Рисунок 1.22 – Web-отчет Астер2016

Данные Астер2016 поступают в MES-систему КГМК.

Также есть информационная система, собирающая и хранящая данные с удобным интерфейсом, другими словами, архив с интерфейсом. Она называется «UniARC» (рисунок 1.23).

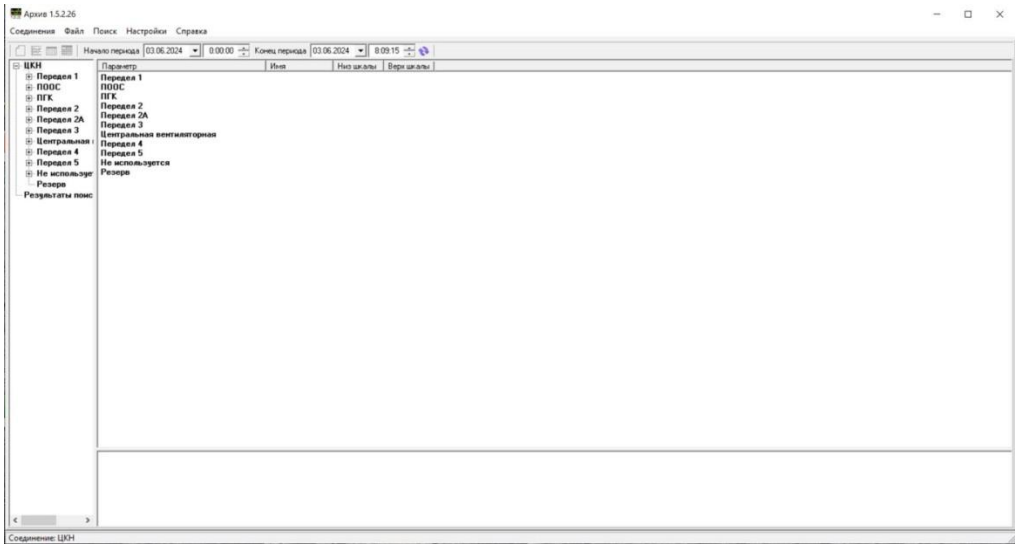


Рисунок 1.23 – Основное окно «Архив 1.5.2.26» в программе «UniARC»

Если сотруднику требуется более детально получить информацию, он может раскрыть слева любой из разделов с помощью кнопки «+». И указать необходимую дату и время для отображения данных (рисунок 1.24).

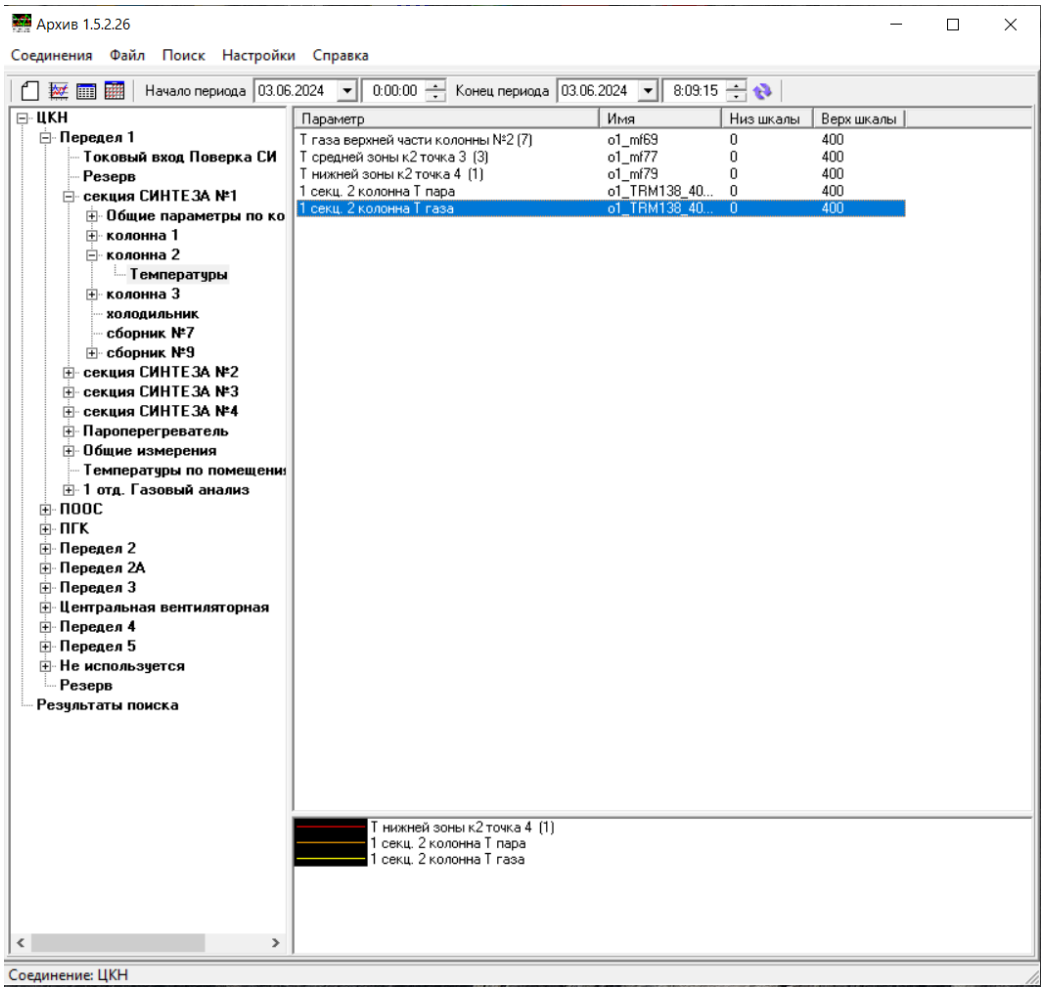


Рисунок 1.24 – Просмотр информации более конкретно

1.3. Описание алгоритмов и подходов для разработки АИС распределения работ

После прочитанных и изученным материалов был создан вариант для реализации задания (рисунок 1.25):

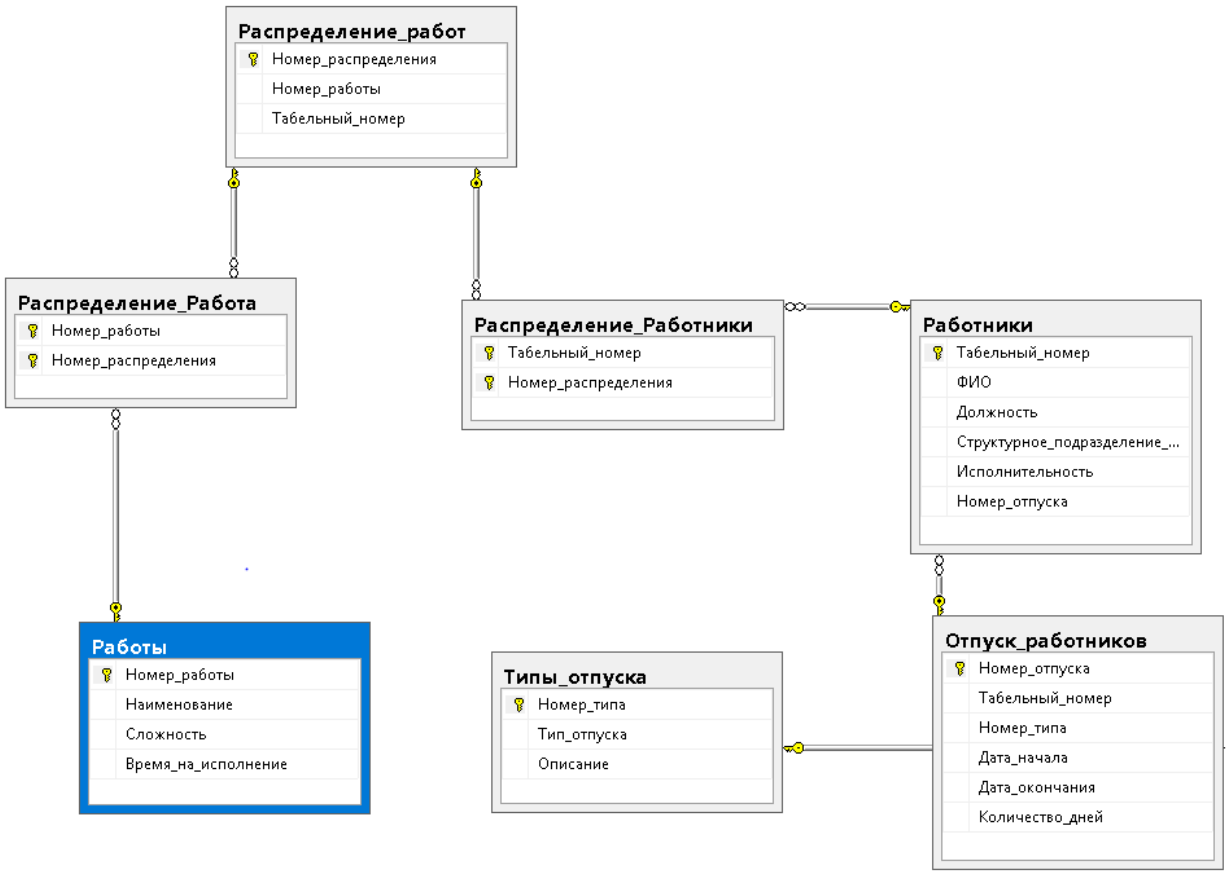


Рисунок 1.25 – Схема данных

Для создания схемы данных был написан следующий код: `CREATE DATABASE WorkDistribution;`

Данная строчка кода позволяет создать базу данных с наименованием «WorkDistribution». Затем требуется перейти к этой базе с помощью: `GO USE WorkDistribution;`

Затем были созданы таблицы, в первую очередь было указано название таблицы, затем, были написаны наименования, тип данных, и, при необходимости добавление «ключа», уникального идентифицируемого столбца,

данные в котором недолжны повторяться, чтобы идентифицировать остальные значения других столбцов конкретной записи.

```
CREATE TABLE Типы_отпуска
(
     Номер_типа INT IDENTITY PRIMARY KEY,
     Тип_отпуска NVARCHAR(200),
     Описание NVARCHAR(200)
);
```

Таким же образом были созданы остальные таблицы.

Для создания связей указывается столбец в скобках «()» в связываемой таблице после `FOREIGN KEY`, затем пишется `REFERENCES`, далее наименование связываемой таблицы и столбец в скобках, который должен быть связан с таким же столбцом в связываемой таблице, важно соблюдать тот же тип данных и желательно создать столбцы с одинаковым наименованием.

```
CREATE TABLE Отпуск_работников
(
     Номер_отпуска INT,
     Табельный_номер INT,
     Номер_типа INT,
     Дата_начала DATE,
     Дата_окончания DATE,
     Количество_дней INT,
     PRIMARY KEY (Номер_отпуска),
     FOREIGN    KEY (Номер_типа)    REFERENCES    Типы_отпуска
(Номер_типа)
);
```

После, было необходимо занести данные в каждую из таблиц чтобы можно было работать с информацией с помощью АИС распределения работ, учитывая связи таблиц.

Для вставки данных используется команда `INSERT INTO` затем наименование таблицы, куда будут вставлены данные, ключевое слово `VALUES` и далее в скобках указывается значение каждого столбца, исключение

составляют только те столбцы, где указана генерация. Важным моментом является правильное соблюдение типа данных, а также длина указываемого значения.

```
INSERT INTO Типы_отпуска VALUES
(
    N'Ежегодный дополнительный отпуск без сохранения заработной платы',
    N'Дополнительный отпуск, допустим, по переработкам'
)
```

Таким же образом заполняются остальные столбцы (рисунок 1.26).

```
USE WorkDistribution;
INSERT INTO Типы_отпуска VALUES
(
    'Ежегодный дополнительный отпуск без сохранения заработной платы',
    'Дополнительнй отпуск, допустим, по переработкам'
),
(
    'Отпуск без сохранения заработной платы',
    'Ещё так называемый "отпуск за свой счёт", берётся человеком по причинам, которые не подпадают под остальные виды отпусков'
),
(
    'Отпуск по беременности и родам',
    'Иногда называемый "дикретным", отпуск, который берёт женщина для того, чтобы процесс вынашивания и родов не мешал работе'
),
(
    'Отпуск при совмещении работы с обучением',
    'Требуется, когда совмещение работы с учёбой без потери часов учёбы или работы - невозможно'
),
(
    'Отпуск по уходу за ребенком',
    'Требуется для сохранения рабочего места в связи с необходимостью ухода за ребёнком'
);
```

Рисунок 1.26 – Внесение данных в таблицу «Типы_отпуска»

Список внесённых данных представлен ниже (рисунок 1.27).

```
SELECT * FROM Типы_отпуска;
```

100 %

Результаты | Сообщения

	Номер_типа	Тип_отпуска	Описание
1	1	Ежегодный дополнительный отпуск без сохранения з...	Дополнительнй отпуск, допустим, по переработкам
2	2	Отпуск без сохранения заработной платы	Ещё так называемый "отпуск за свой счёт", берётся человеком по причинам, которые не подпад...
3	3	Отпуск по беременности и родам	Иногда называемый "дикретным", отпуск, который берёт женщина для того, чтобы процесс вына...
4	4	Отпуск при совмещении работы с обучением	Требуется, когда совмещение работы с учёбой без потери часов учёбы или работы - невозможно
5	5	Отпуск по уходу за ребенком	Требуется для сохранения рабочего места в связи с необходимостью ухода за ребёнком

Рисунок 1.27 – Внесённые данные в таблицу «Типы_отпуска»

Далее были внесены данные в таблицу «Работники» (рисунок 1.28).

```
USE WorkDistribution;
INSERT INTO Работники VALUES
(
    2876095, 'Васнецов Игорь Николаевич', 'Инженер-программист 1 категории', 'ЦАП', 10, 1
),
(
    2877000, 'Блинина Любовь Михайловна', 'Главный специалист', 'ЦАП', 10, 2
),
(
    2865123, 'Улочкин Илья Вадимович', 'Главный специалист', 'ЦЭН', 8, 3
),
(
    2835010, 'Палкин Дмитрий Олегович', 'Главный специалист', 'ЦЭН', 7, 4
),
(
    2871111, 'Бзденко Роман Александрович', 'Инженер-программист 2 категории', 'КАЦ', 9, 5
),
(
    2876911, 'Отпусков Олег Васильевич', 'Инженер-программист 1 категории', 'КАЦ', 8, 6
),
(
    2874222, 'Маладцова Валентина Даниловна', 'Инженер-программист 1 категории', 'ХМЦ', 9, 7
),
(
    2876311, 'Быстреньков Алексей Закурпатович', 'Инженер-программист 1 категории', 'ХМЦ', 8, 8
),
(
    2873333, 'Владеленко Даниил Булатович', 'Инженер-программист 1 категории', 'РАФ', 7, 9
),
(
    2855321, 'Блинков Антон Копатович', 'Инженер-программист 2 категории', 'РАФ', 10, 10
),
(
    2854321, 'Братков Булат Замкадович', 'Инженер-программист 2 категории', 'ЦМТО', 10, 11
);
```

Рисунок 1.28 – Внесение данных в таблицу «Работники»

Внесённые данные в таблицу «Работники» представлены ниже (рисунок 1.29).

```
SELECT * FROM Работники;
```

100 %

Результаты | Сообщения

	Табельный_номер	ФИО	Должность	Структурное_подразделение_цех	Исполнительность	Номер_отпуска
1	2835010	Палкин Дмитрий Олегович	Главный специалист	ЦЭН	7	4
2	2854321	Братков Булат Замкадович	Инженер-программист 2 категории	ЦМТО	10	11
3	2855321	Блинков Антон Копатович	Инженер-программист 2 категории	РАФ	10	10
4	2865123	Улочкин Илья Вадимович	Главный специалист	ЦЭН	8	3
5	2871111	Бзденко Роман Александрович	Инженер-программист 2 категории	КАЦ	9	5
6	2873333	Владеленко Даниил Булатович	Инженер-программист 1 категории	РАФ	7	9
7	2874222	Маладцова Валентина Даниловна	Инженер-программист 1 категории	ХМЦ	9	7
8	2876095	Васнецов Игорь Николаевич	Инженер-программист 1 категории	ЦАП	10	1
9	2876311	Быстреньков Алексей Закурпатович	Инженер-программист 1 категории	ХМЦ	8	8
10	2876911	Отпусков Олег Васильевич	Инженер-программист 1 категории	КАЦ	8	6
11	2877000	Блинина Любовь Михайловна	Главный специалист	ЦАП	10	2

Рисунок 1.29 – Внесённые данные в таблицу «Работники»

Затем были внесена информация в таблицу «Отпуск_работников» (рисунок 1.30).

```
INSERT INTO Отпуск_работников VALUES
(
    1, 2876095, 1, '2024-03-07', '2023-04-01', 26
),
(
    2, 2877000, 2, '2024-01-25', '2024-02-26', 33
),
(
    3, 2865123, 2, '2024-01-31', '2024-02-08', 40
),
(
    4, 2835010, 3, '2024-01-06', '2024-02-01', 25
),
(
    5, 2871111, 4, '2024-03-18', '2024-04-26', 39
),
(
    6, 2876911, 5, '2024-02-12', '2024-03-14', 33
),
(
    7, 2874222, 5, '2024-04-27', '2024-05-22', 26
),
(
    8, 2876311, 5, '2024-04-04', '2024-05-05', 33
),
(
    9, 2873333, 1, '2024-02-04', '2024-03-29', 26
),
(
    10, 2855321, 3, '2024-03-10', '2024-04-11', 32
),
(
    11, 2854321, 4, '2024-05-10', '2024-06-04', 26
);
```

Рисунок 1.30 – Внесение отпусков работников в соответствующую таблицу «Отпуск_работников»

Ниже представлен результат запроса к базе данных для отображения данных из таблицы «Отпуск_работников» (рисунок 1.31).

```
1  SELECT * FROM Отпуск_работников;
```

110 % Проблемы не найдены.

T-SQL | Результаты | Сообщение

	Номер_отпуска	Табельный_номер	Номер_типа	Дата_начала	Дата_окончания	Количество_дней
1	1	2876095	1	2024-03-07	2023-04-01	26
2	2	2877000	2	2024-01-25	2024-02-26	33
3	3	2865123	2	2024-01-31	2024-02-08	40
4	4	2835010	3	2024-01-06	2024-02-01	25
5	5	2871111	4	2024-03-18	2024-04-26	39
6	6	2876911	5	2024-02-12	2024-03-14	33
7	7	2874222	5	2024-04-27	2024-05-22	26
8	8	2876311	5	2024-04-04	2024-05-05	33
9	9	2873333	1	2024-02-04	2024-03-29	26
10	10	2855321	3	2024-03-10	2024-04-11	32
11	11	2854321	4	2024-05-10	2024-06-04	26

Рисунок 1.31 – Занесённая информация в «Отпуск_работников»

Также была заполнена таблица с работами (рисунок 1.32):

```
USE WorkDistribution;
INSERT INTO Работы VALUES
(
    'Оперативная диагностика неисправностей', 6, '08:00:00', 'Определение и измерение признаков, параметров объектов (технических систем и их компонентов) для оперативной оценки их состояния на заданном этапе
),
(
    'Выполнение заявок подразделений', 5, '07:00:00', 'Выполнение заявок подразделений по сопровождению и модификации программных компонентов АСУТП'
),
(
    'Внесение изменений в прикладное ПО контроллера', 8, '03:00:00', 'Внесение изменений в прикладное ПО контроллеров и серверов АСУТП, программы человеко-машинного интерфейса (ЧМИ)'
),
(
    'Доработка прикладного программного обеспечения АСУТП', 10, '10:00:00', 'Доработка прикладного программного обеспечения АСУТП в процессе опытной и промышленной эксплуатации'
),
(
    'Консультирование работников подразделений в части использования ПО', 7, '02:00:00', 'Консультирование, при необходимости, работников подразделений и подрядных организаций в части использования ПО, выполне
),
(
    'Выполнение подготовки планов-графиков внедрения проектов АСУТП', 8, '03:00:00', 'Выполнение подготовки планов-графиков внедрения проектов АСУТП, оценку нагрузку по выполняемым работам, выдачу рекомендации
),
(
    'Разработку и внедрение программных компоненты АСУТП', 10, '07:00:00', 'Осуществляется разработка программных компонентов для АСУТП и их внедрение'
),
(
    'Осуществление подготовки нормативно-справочной документации', 7, '02:00:00', 'Осуществление подготовки нормативно-справочной документации, обновлений текущей технической документации АСУТП'
),
(
    'Идентификация и экспертная оценка рисков', 8, '04:00:00', 'Идентификация и экспертная оценка рисков в соответствии с компетенциями по направлениям деятельности. Проведение анализа факторов, оказывающих вл
),
(
    'Участие в анализе проектов', 6, '05:00:00', 'Участие в анализе проектов, разрабатываемых сторонними организациями для ВСП, подготовка замечаний по проектной документации.'
),
(
    'Авторский надзор за разработанными программными компонентами', 10, '09:00:00', 'Осуществление авторского надзора за разработанными программными компонентами, совершенствование их при необходимости'
),
(
    'Постановка задач в области автоматизации технологических процессов', 9, '01:00:00', 'Осуществление постановки задач в области автоматизации технологических процессов'
),
(
    'Поиск оптимальных решений по автоматизации технологического производства', 6, '08:00:00', 'Поиск оптимальных решений по автоматизации технологического производства с целью стандартизации программных проду
),
(
    'Подготовка предложений по планированию деятельности отдела', 8, '05:00:00', 'Контроль норм производительности, расчёт норм запасов по готовой продукции, сырью и комплектующим с учётом сезонности; Контроль
),
```

Рисунок 1.32 – Занесение данных в таблицу «Работы»

Результат запроса к базе данных на получение информации из таблицы «Работы» (рисунок 1.33).

```
SELECT * FROM Работы;
```

100 %

Результаты | Сообщения

	Номер_работы	Наименование	Сложность	Время_на_исполнение	Описание
1	1	Оперативная диагностика неисправностей	6	08:00:00.0000000	Определение и измерение признаков, параметров объ...
2	2	Выполнение заявок подразделений	5	07:00:00.0000000	Выполнение заявок подразделений по сопровождению ...
3	3	Внесение изменений в прикладное ПО контроллера	8	03:00:00.0000000	Внесение изменений в прикладное ПО контроллеров и ...
4	4	Доработка прикладного программного обеспечения АС...	10	10:00:00.0000000	Доработка прикладного программного обеспечения АС...
5	5	Консультирование работников подразделений в части и...	7	02:00:00.0000000	Консультирование, при необходимости, работников под...
6	6	Выполнение подготовки планов-графиков внедрения пр...	8	03:00:00.0000000	Выполнение подготовки планов-графиков внедрения п...
7	7	Разработку и внедрение программных компоненты АС...	10	07:00:00.0000000	Осуществляется разработка программных компоненто...
8	8	Осуществление подготовки нормативно-справочной до...	7	02:00:00.0000000	Осуществление подготовки нормативно-справочной до...
9	9	Идентификация и экспертная оценка рисков	8	04:00:00.0000000	Идентификация и экспертная оценка рисков в соответ...
10	10	Участие в анализе проектов	6	05:00:00.0000000	Участие в анализе проектов, разрабатываемых сторон...
11	11	Авторский надзор за разработанными программными ...	10	09:00:00.0000000	Осуществление авторского надзора за разработанным...
12	12	Постановка задач в области автоматизации технологич...	9	01:00:00.0000000	Осуществление постановки задач в области автоматиз...
13	13	Поиск оптимальных решений по автоматизации технол...	6	08:00:00.0000000	Поиск оптимальных решений по автоматизации технол...
14	14	Подготовка предложений по планированию деятельнос...	8	05:00:00.0000000	Контроль норм производительности, расчёт норм запа...
15	15	Внести изменения в прикладное ПО контроллеров и се...	10	12:00:00.0000000	Внесение изменений в прикладное ПО контроллеров и ...
16	16	Доработка прикладного программного обеспечения АС...	9	11:00:00.0000000	Доработка прикладного программного обеспечения АС...
17	17	Идентификация и экспертная оценка рисков	8	08:00:00.0000000	Идентификация и экспертная оценка рисков в соответ...
18	18	Осуществление руководства	10	08:00:00.0000000	Осуществление руководства и установление обязанно...
19	19	Обеспечение и контроль выполнения	7	08:00:00.0000000	Обеспечение и контроль выполнения работниками отд...
20	20	Контроль выполнения заявок	7	08:00:00.0000000	Контроль выполнения заявок, находящиеся в работе у ...
21	21	Проведение аналитической работы по предотвращению...	6	06:00:00.0000000	Проведение аналитической работы по предотвращени...

Рисунок 1.33 – Занесённые данные в таблицу «Работы»

Далее было разработана АИС распределения работ. Сначала был создан вход в приложение для контроля возможностей пользователей.

2. ПРОЕКТНЫЙ РАЗДЕЛ

2.1. Проектирование АИС распределения работ

2.1.1. Описание функционала АИС

Автоматизированная информационная система имеет функционал для проверки правильности распределения работ и отправки отчётов о неправильном распределении.

Сама система подразумевает, что у работников есть некий, уже установленный уровень «Загруженности», исходя из количества и сложности работы, которая уже ему поручена, а также уровень «Исполнительности» на основании работоспособности конкретного сотрудника за определённый промежуток времени и качестве выполненных работ.

АИС даёт возможность получения показателя «Выполнимость», который получается путём вычитания из «Нагруженности» показателя «Исполнительность». Результат вычитания даёт возможность оценить уровень выполнимости работ.

Проверка введение табельного номера и указания верного режима работы, доступного для указанного работника осуществляется за счёт запроса в базу данных, сначала для проверки наличия табельного номера, а затем проверки правильного указания режима работы. Это происходит, т.к. при введении неправильного табельного номера будет выскакивать своя ошибка, указывающая на неправильность его указания. Режим «Список работников» доступен только работнику, имеющему должность «Главный специалист». А в режим «Конкретный работник» могут входить как «Главный специалист», так и остальные сотрудники.

Главному специалисту доступны «checkedlistbox» – список работников, «listbox» – работы, которые им назначены.

На форме просмотра списка сотрудников и работ присутствует система отчётов. Если специалист видит, что конкретный работник по каким -либо причинам не может выполнить данный вид деятельности, то он может выбрать

сотрудника и нажать кнопку «Отчёт», далее система спросит уверен ли он в отправке отчёта об ошибке в базу данных, это сделано для того, чтобы отчёт не отправлялся по ошибке. Также присутствует ещё одна проверка, она заключается в считывании количества выбранных работников, если количество выбранных работников становиться 0, то программа убирает возможность отправки отчёта, в ином случае такая возможность появляется.

Принцип распределения работ состоит в считывании «нагруженности», «исполнимости» и вычитании второго из первого, на основании этих расчётов получается «Выполнимость». Затем сотрудники заносятся в специально созданных класс и сортируются по «Выполнимости». В работах уже есть сложность, поэтому работы сразу заносятся в класс и сортируются по ней. Отсортированные работники по выполнимости и работы по сложности и составляют само распределение, которое заносится в таблицу «Распределение_работ» базы данных. Также в программе реализована проверка на доступность подключения к базе данных, если это невозможно, то программа начинает работу исключительно с файлами, которые постоянно обновляются при каждом подключении к базе данных. Если же отсутствует файл, то программа выдаёт сообщение об ошибке подключения к базе данных и работе с файлами.

Ниже представлена диаграмма «Функциональный блок» (рисунок 2.1).

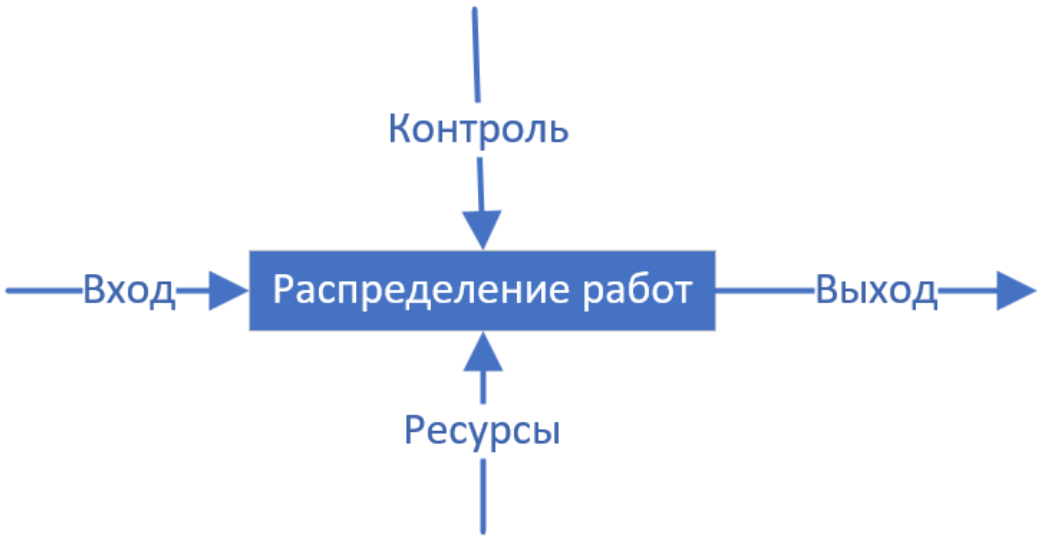


Рисунок 2.1 – Диаграмма «Функциональный блок»

Диаграмма функционального блока (DFB) представляет собой графическое изображение структуры и связи между функциональными составляющими, которые выполняют определенные функции в системе. На примере автоматизированной информационной системы (АИС) распределения работ, DFB может выглядеть следующим образом:

Вход – этот блок отвечает за прием и обработку информации введённых данных пользователем при входе, а также выбор его режима доступа.

Контроль – данный блок отвечает за контроль администратором распределения работ среди сотрудников.

Выход – этот блок отвечает за определение исполнителей и сроков выполнения задач. Он осуществляет автоматическое распределение задач между членами команды в зависимости от их нагрузки и исполнительности.

Ресурсы – блок, в который входит сотрудники и работы.

Вот как выглядит декомпозиция функционального блока (рисунок 2.2):

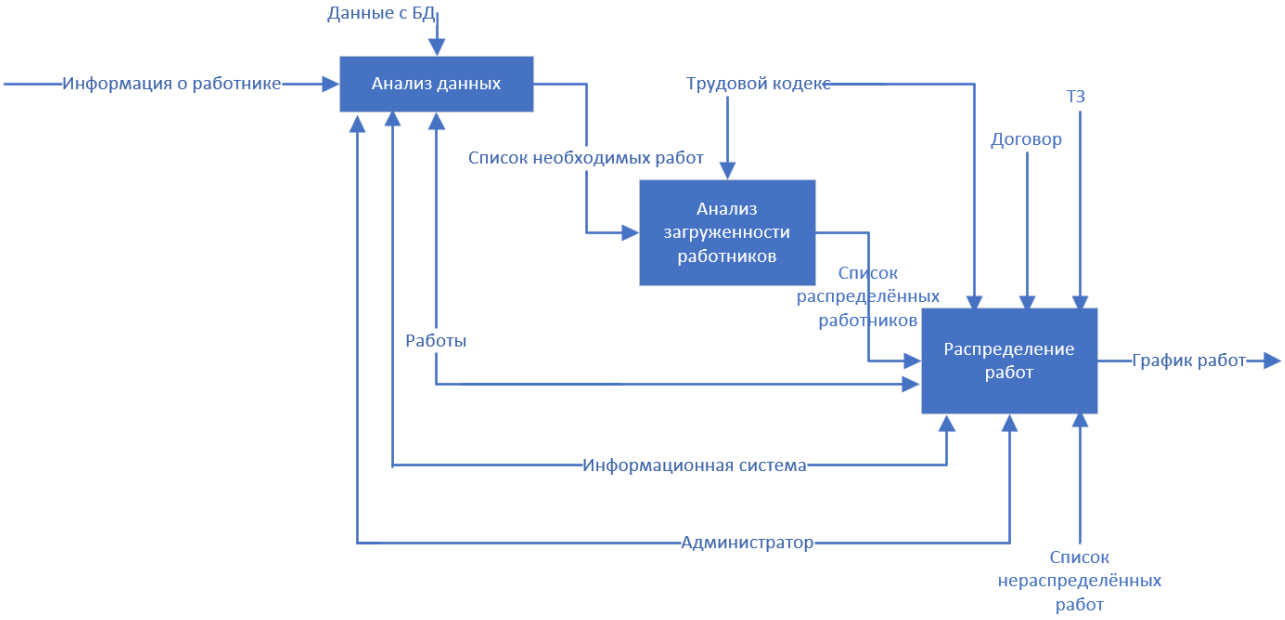


Рисунок 2.2 – Декомпозиция функционального блока

Анализ данных – в данный блок входит анализ полученной информации из заполненной базы данных.

Анализ загруженности работников – анализируется загруженность каждого из работников и выставляется конкретное значение этой нагрузки для последующего распределения с учётом данного параметра.

Распределение работ – этап распределения работы с учётом полученных значений нагруженности, исполнительности работников, а также сложности каждой из работ.

Далее идёт представление в виде диаграммы DFD (рисунок 2.3).

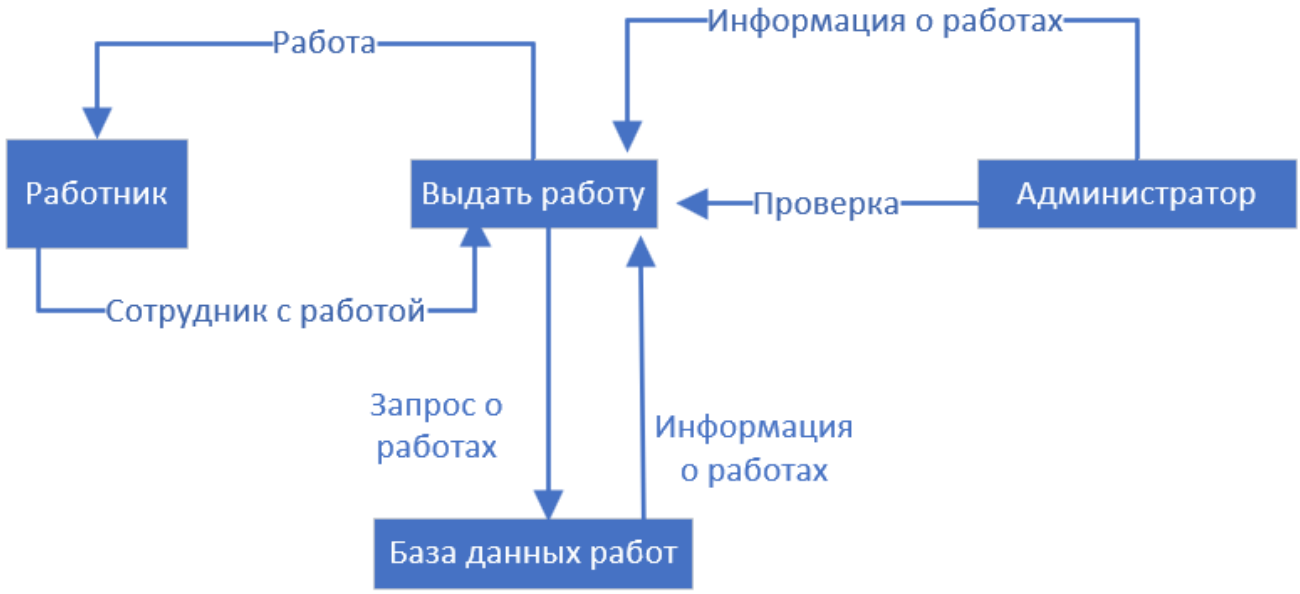


Рисунок 2.3 – Диаграмма DFD

Данная диаграмма DFD (Data Flow Diagram) представляет собой схему распределения работ в АИС (автоматизированной информационной системе) компании. На диаграмме изображены основные процессы, потоки данных и участники, вовлеченные в работу системы.

На верхнем уровне диаграммы находятся основные процессы системы, такие как "Выдать работу", "Работник" и "Администритор". Каждый из этих процессов имеет связанные с ним входные и выходные потоки данных, которые отображаются стрелками на диаграмме.

Участниками в системе являются сотрудники, АИС и администраторы. Сотрудники получают номер на выполнение работы, администраторы могут следить за выполнением распределения. А сама АИС получает данные и распределяет работы среди сотрудников.

Диаграмма DFD помогает наглядно представить взаимодействие между процессами и участниками системы, а также позволяет выявить потенциальные узкие места и оптимизировать процессы для более эффективной работы компании.

Техническое задание

Техническое задание на разработку приложения учета рабочего времени на основе распознавания лиц с учетом требований ГОСТ 34.602-89

1. Наименование разрабатываемого программного продукта

"WorkDistributor"

2. Назначение разрабатываемого программного продукта

Данное ПО разрабатывается для автоматизации распределения работ на производстве.

3. Требования к программному продукту

3.1.Функциональные требования

– Занесение и хранение данных о сотрудниках (табельный номер, ФИО, должность, загруженность, исполнительность);

– Занесение и хранение списка работ (номер, наименование, сложность);

– Распределение работ (№ записи, номер работы, табельный номер);

– Административная панель управления (просмотр, выбор, отправка отчёта об ошибке);

3.2.Требования к надежности

– Защита данных от несанкционированного доступа;

– Резервное копирование данных;

– Восстановление информации;

3.3.Требования к интерфейсу

Интуитивно понятный интерфейс для пользователей всех уровней владения;

3.4.Технические требования

– Компьютер или сервер с установленной ОС Windows;

– База данных, с занесённой информацией;

– Доступ в интернет для синхронизации данных.

4. Требования к документации

– Руководство пользователя;

– Инструкция по установке;

– Инструкция по настройке;

– Руководство системного администратора;

5. Стадии и этапы разработки

5.1. Техническое задание;

5.2. Проектирование программного продукта;

5.3. Кодирование;

5.4. Тестирование;

5.5. Внедрение;

5.6. Сопровождение и обслуживание;

6. Требования к квалификации и численности исполнителей

– Руководитель проекта (1 чел.);

– Проектировщик интерфейса (1 чел.);

– Программисты (1 чел.);

– Тестировщики (1 чел.);

– Администратор (1 чел.).

7. Источники и порядок финансирования

Финансирование проекта осуществляется за счет средств заказчика.

8. Сроки выполнения проекта

Проект должен быть выполнен в течение 3 месяцев с момента утверждения технического задания.

Для проектирования АИС распределения работ была составлена диаграмма IDEF3 (рисунок 2.4):

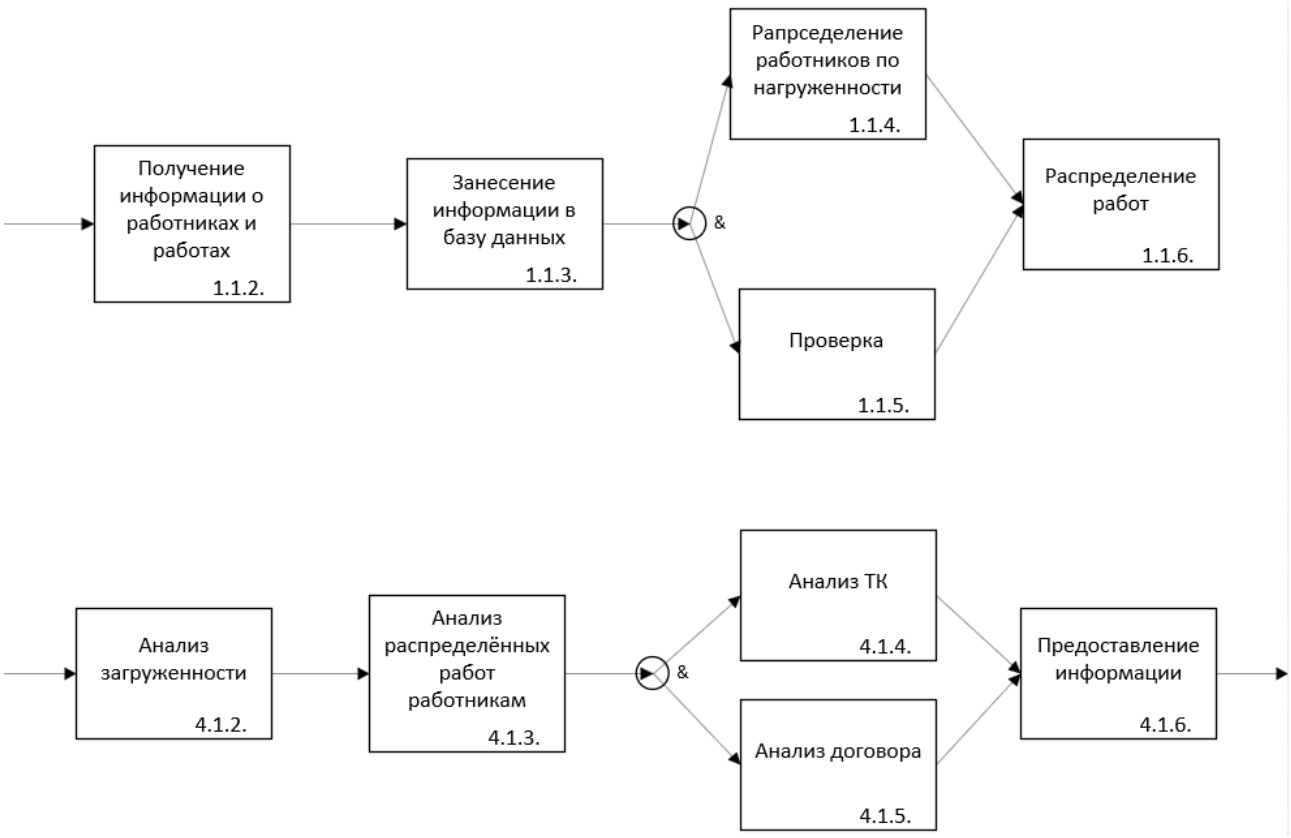


Рисунок 2.4 – Диаграмма IDEF3

Диаграмма IDEF3 представляет собой схему распределения работ в автоматизированной информационной системе (АИС). На диаграмме показаны основные этапы выполнения задачи, а также последовательность действий и ресурсы, необходимые для ее выполнения. Каждый этап представлен отдельным блоком, который соединен линиями, показывающими порядок выполнения работ.

Диаграмма IDEF3 для АИС распределения работ может включать следующие этапы:

1. Получение запроса на вход пользователя;
2. Проверка правильности введённых данных;
3. Вход через учётную запись в доступный режим просмотра;
4. Отправка отчёта о неправильности распределения;
5. Распределение работ среди сотрудников.
6. Предоставление доступа к просмотру и функционалу.

Каждый этап выполнения задачи представлен блоком на диаграмме IDEF3, а линии между этапами показывают последовательность действий. Такая диаграмма помогает лучше понять структуру процесса и оптимизировать его выполнение в АИС.

Для более простого понимания возможностей каждого из участников проекта была создана UML диаграмма (рисунок 2.5).

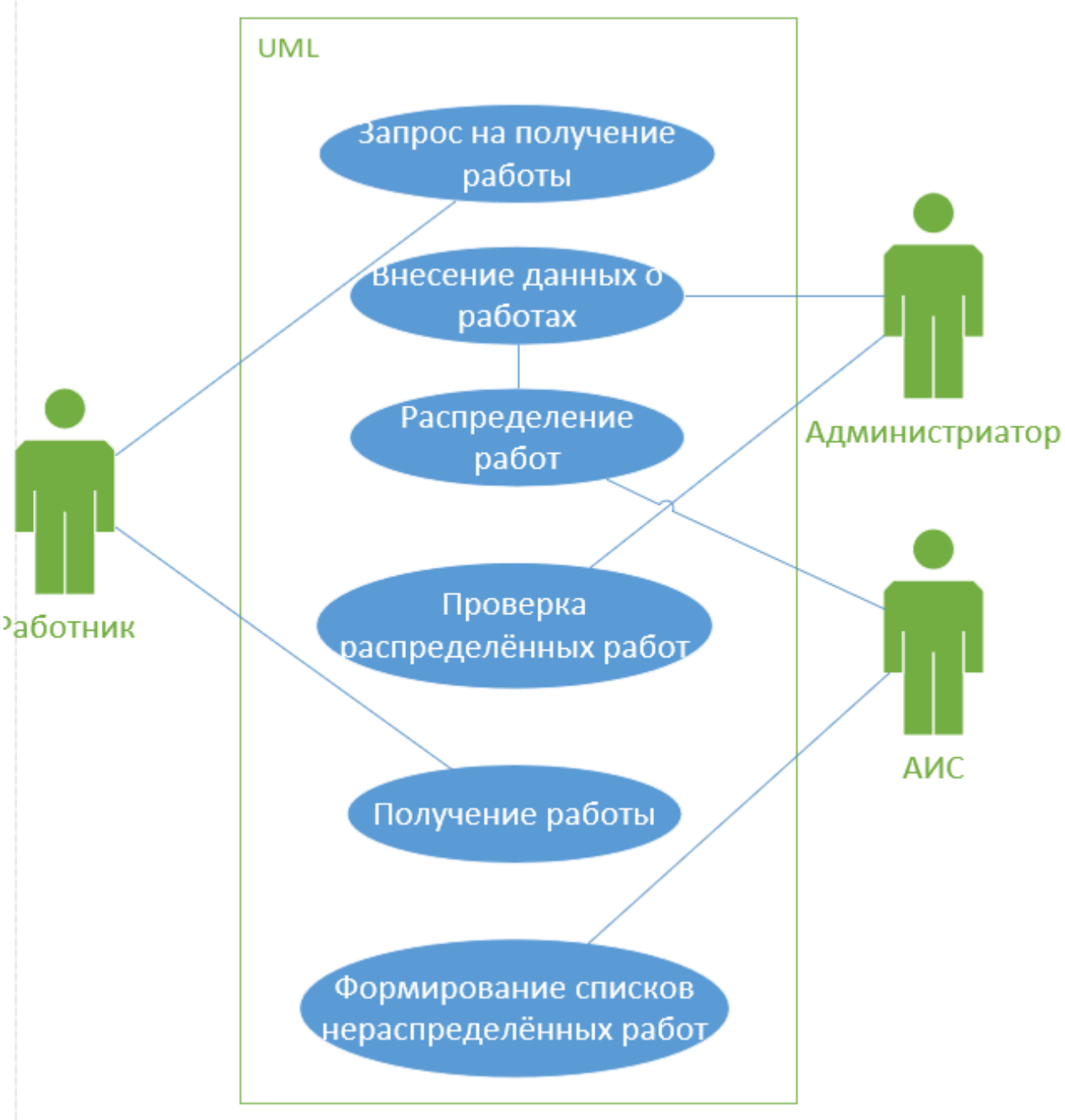


Рисунок 2.5 – Диаграмма UML

UML диаграмма распределения работ представляет собой графическую модель, которая показывает различные компоненты и подсистемы в информационной системе, их взаимосвязи и способы взаимодействия между ними. Диаграмма включает в себя работника, администратора, а также АИС и связи между ними (запрос на получение работы, внесение данных о работах, Распределение работ, проверка распределённых работ, получение работы, формирование списка нераспределённых работ).

В рамках диаграммы распределения работ можно определить, какие компоненты системы находятся на каких узлах, какие протоколы используются для связи между компонентами, и какие данные передаются через эти связи.

В целом, UML диаграмма распределения работ помогает визуализировать архитектуру информационной системы, ее компоненты и связи между ними.

Также для проектирования системы была создана диаграмма последовательности (рисунок 2.6).

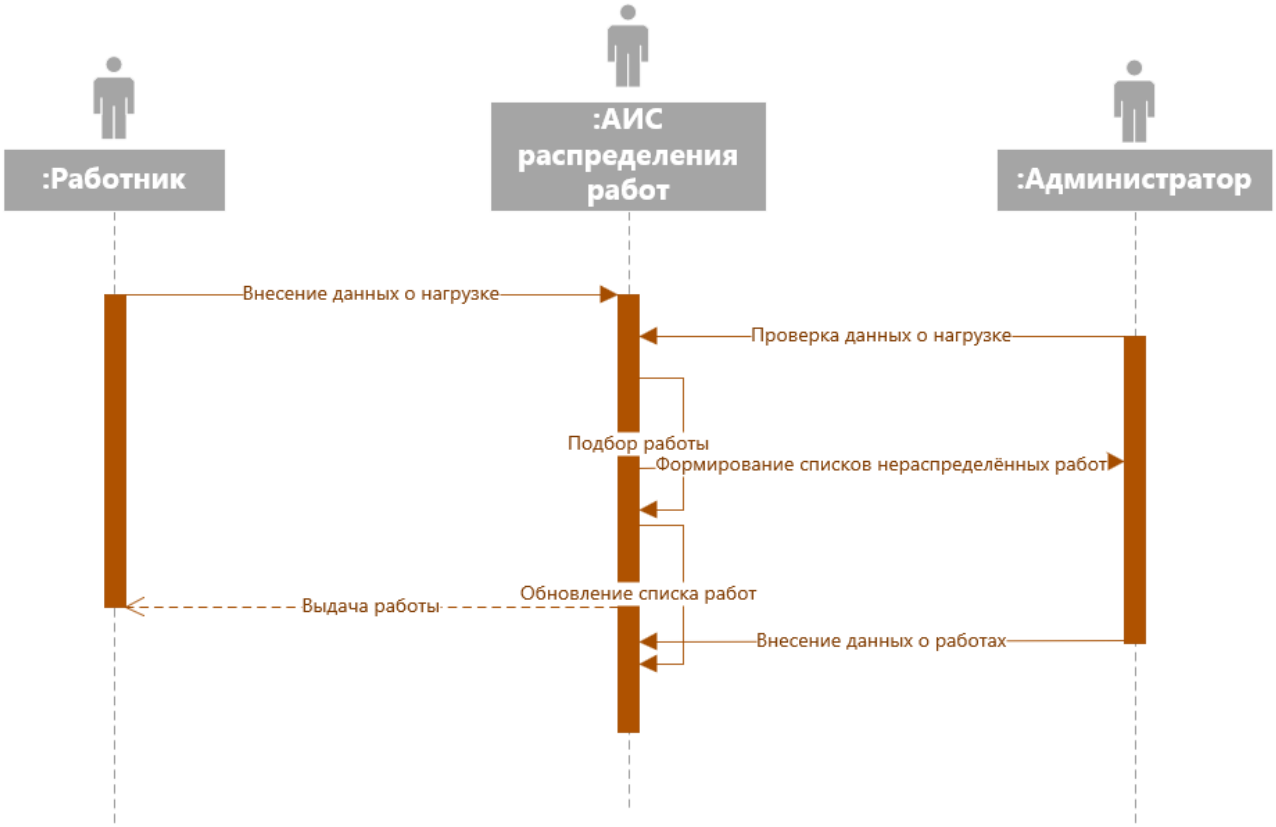


Рисунок 2.6 – Диаграмма последовательности

Диаграмма последовательности АИС распределения работ представляет собой графическое изображение последовательности и обмена между компонентами системы.

На диаграмме отображены блоки, как различные этапы выполнения работ, и стрелки, обозначающие передачу данных между этими этапами.

На диаграмме представлена следующая последовательность работ:

1. Получение информации от работника в виде табельного номера
2. Поиск необходимых данных в базе данных
3. Анализ полученной информации
4. Проверка правильности введения и указания режима доступа
5. Отправка отчета о неправильности распределения

Стрелки между блоками показывают направление передачи данных и информации между этапами работы.

Диаграмма последовательности АИС распределения работ помогает визуализировать и оптимизировать процессы автоматизированных систем.

Далее была составлена диаграмма деятельности (рисунок 2.7).

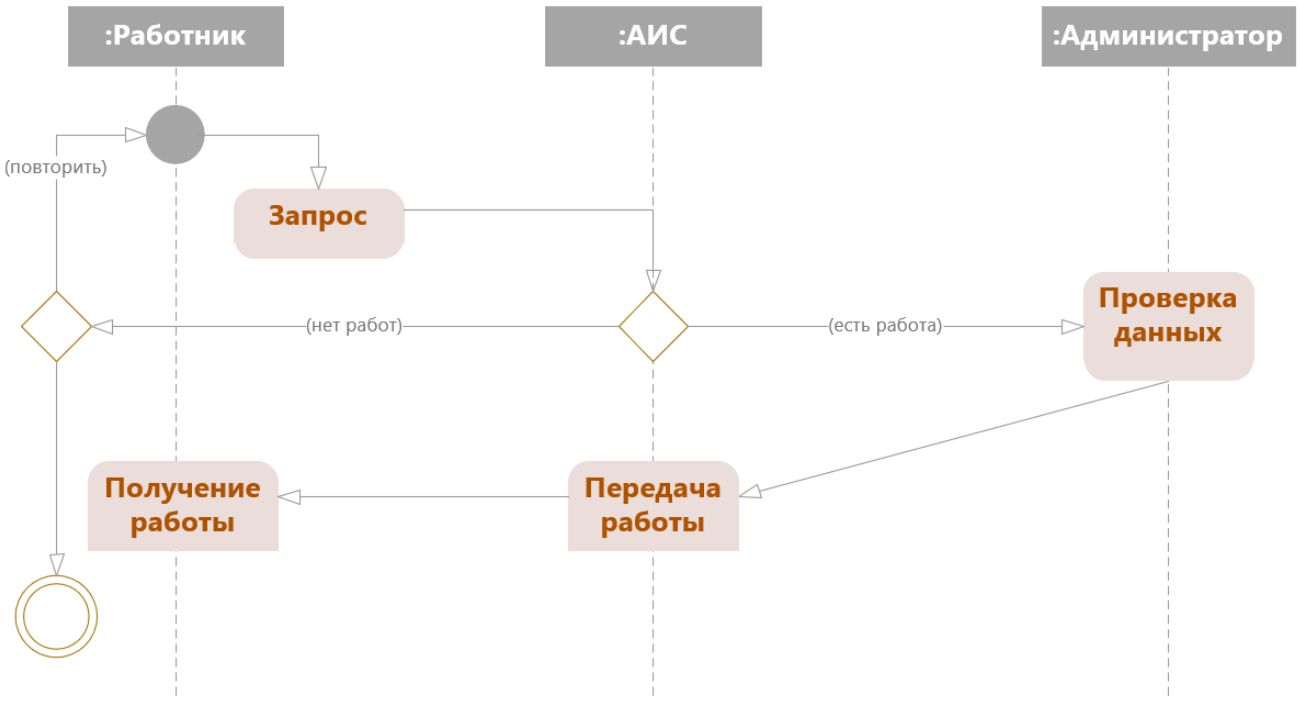


Рисунок 2.7 – Диаграмма деятельности

Данная диаграмма последовательности представляет собой визуализацию процесса распределения работ в автоматизированной информационной системе (АИС). На диаграмме изображены этапы, а именно:

Пользователь инициирует вход в АИС.

АИС принимает запрос, проверяет корректность введения данных и проводит анализ содержания работ для определения их приоритетности.

Далее АИС автоматически распределяет работы между исполнителями в соответствии с их загруженностью и исполнительностью.

Исполнители получают возможность просмотра номера работы, которая им назначена.

АИС предоставляет возможность отправки отчётности о неправильном распределения конкретной работы конкретному сотруднику.

Таким образом, данная диаграмма позволяет наглядно представить последовательность действий, необходимых для эффективного распределения и контроля выполнения работ в АИС.

2.1.2. Анализ требований

У пользователей есть большой спектр требований к программному продукту. Основное требование – автоматизированное распределение работ среди сотрудников с помощью системы. Далее идёт нужда в более удобном интерфейсе, в максимально интуитивно понятном виде. Сам интерфейс должен содержать насколько это возможно наименьшее количество элементов для взаимодействия, таких как кнопки (buttons), поля для ввода (textbox), текста (label), полосы прокрутки (scrollbar), а также элементы содержащие данные – список элементов (listbox) и список элементов с флаговыми кнопками (checkedlistbox).

Сотрудникам необходимо иметь возможность обращения в поддержку по каким-либо причинам связанным с использованием автоматизированной системы распределения работ.

2.1.3. Проектирование пользовательского интерфейса

В самом начале была добавлена возможность входа под своим конкретным табельным номером (рисунок 2.8):

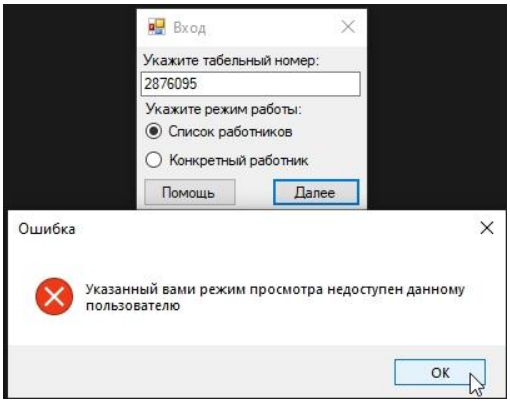


Рисунок 2.8 – Окно ошибки, при указании неверных данных

Проверка введение табельного номера и указания верного режима работы, доступного для указанного работника осуществляется за счёт запроса в базу данных, сначала для проверки наличия табельного номера, а затем проверки правильного указания режима работы. Это происходит, т.к. при введении неправильного табельного номера будет выскакивать своя ошибка, указывающая на неправильность его указания. Режим «Список работников» доступен только работнику, имеющему должность «Главный специалист». А в режим «Конкретный работник» могут входить как «Главный специалист», так и остальные сотрудники (рисунок 2.9).

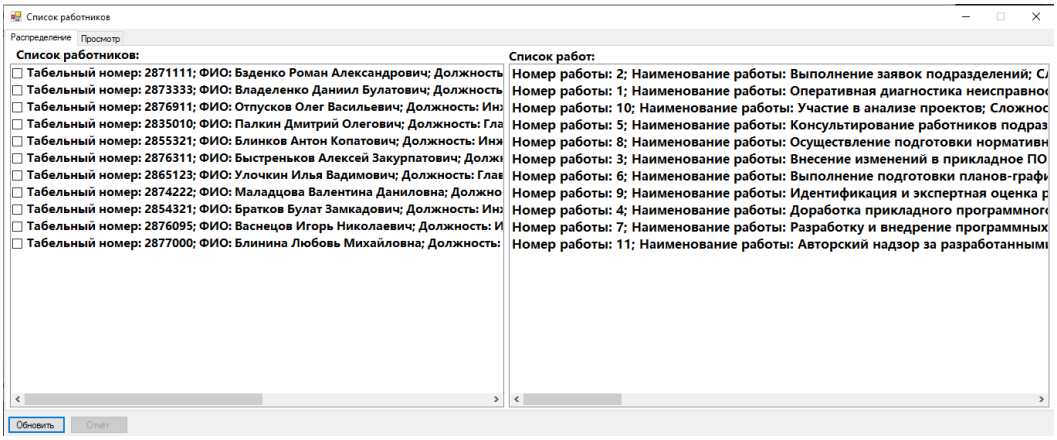


Рисунок 2.9 – Просмотр окна в режиме работы «Список работников»

Главному специалисту доступны «checkedlistbox» – список работников, «listbox» – работы, которые им назначены (рисунок 2.10).

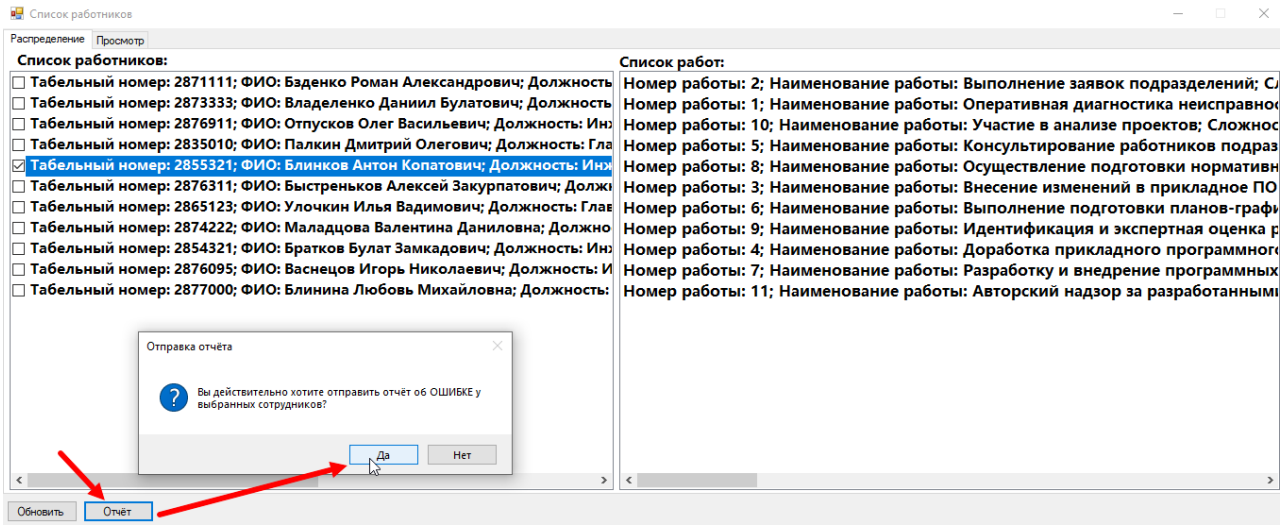


Рисунок 2.10 – Возможность отправки отчёта о неправильном распределении сотрудника по работе

На форме просмотра списка сотрудников и работ присутствует система отчётов. Если специалист видит, что конкретный работник по каким -либо причинам не может выполнить данный вид деятельности, то он может выбрать сотрудника и нажать кнопку «Отчёт» (рисунок 2.9), далее система спросит уверен ли он в отправке отчёта об ошибке в базу данных, это сделано для того, чтобы отчёт не отправлялся по ошибке. Также присутствует ещё одна проверка, она заключается в считывании количества выбранных работников, если количество выбранных работников становиться 0, то программа убирает возможность отправки отчёта, в ином случае такая возможность появляется (рисунок 2.10).

Также главному специалисту доступен просмотр списка распределённых работ, с указанием правильности распределения (рисунки 2.11, 2.12).

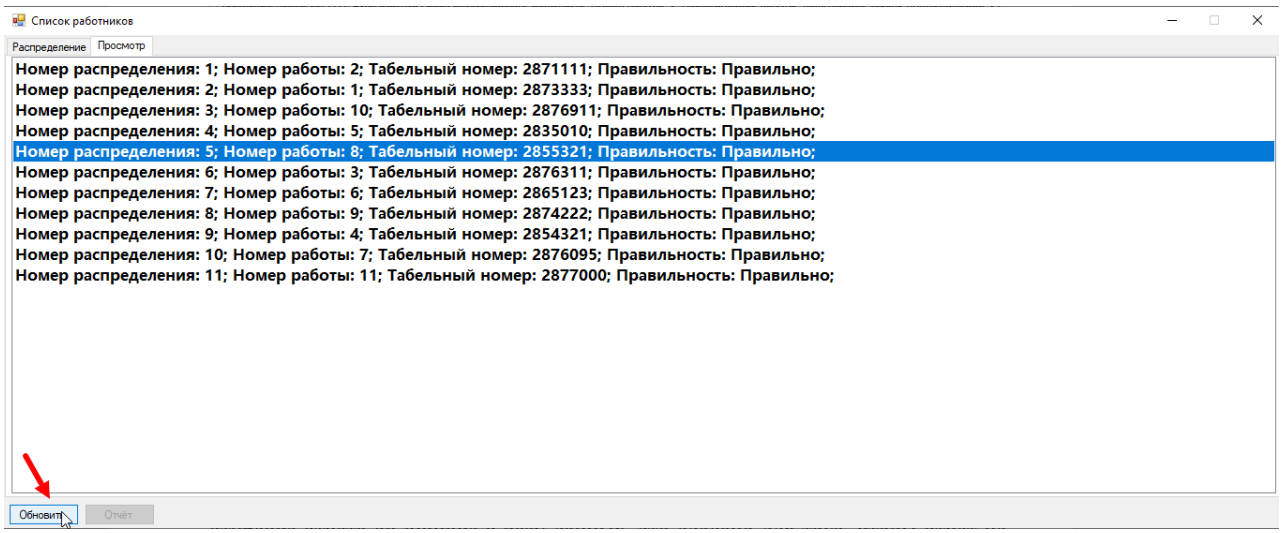


Рисунок 2.11 – Форма для просмотра информации о конкретном сотруднике, предоставленной ему работе, а также правильности занесения

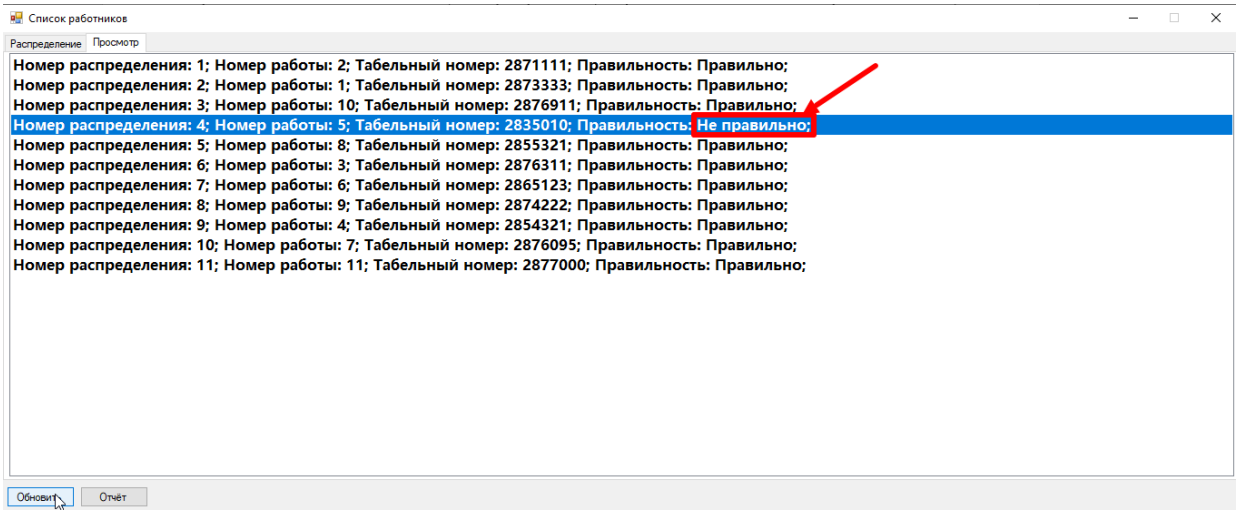


Рисунок 2.12 – Изменённые данные после обновления таблицы списков распределения

Обычному работнику доступен простой просмотр номера конкретной работы, которую ему выдала система (рисунок 2.13).

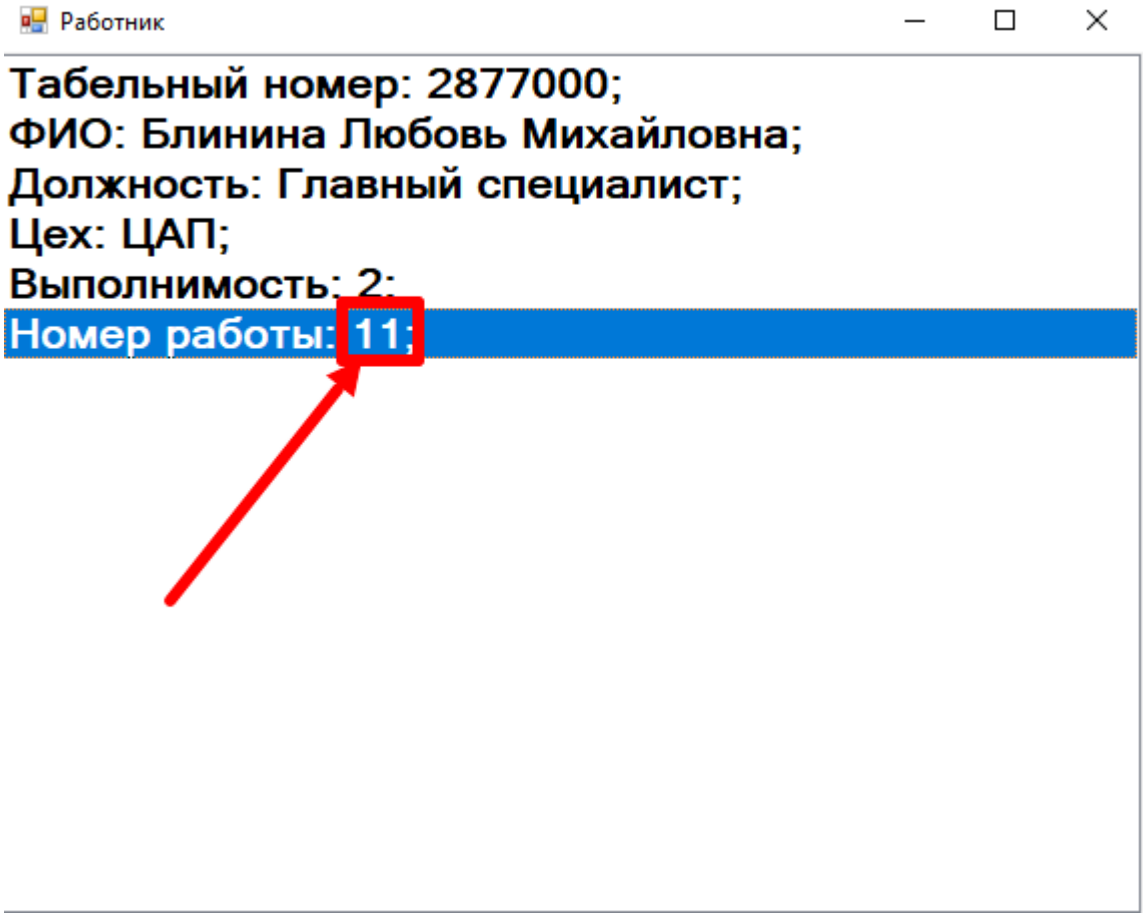


Рисунок 2.13 – Просмотр конкретного работника по его табельному номеру через соответствующий режим доступа

2.2. Реализация автоматизированной информационной системы распределения работ

2.2.1. Выбор технологий и инструментов для реализации

Visual Studio — это средство для разработчика, которое используется для исполнения полного цикла разработок. Это комплексная и интегрированная среда разработки по-другому называемая IDE – integrated development environment, которая используется для написания, изменения, отладки, а также компиляции кода, и, в конечном итоге сборки приложения. Помимо редактирования и отладки кода VS включает компиляторы, средства завершения кода, управление версиями, расширения и многое другое, чтобы улучшить каждый этап процесса разработки программного обеспечения.

Visual Studio предоставляет разработчикам широкие возможности среды разработки для эффективного и совместного написания высококачественного кода.

– Установщик, позволяющий гибко настраивать компоненты для последующей загрузки

– Средства для написания кода и функции — всё необходимое для создания приложений

– Поддержка нескольких языков программирования C++, C#, JavaScript, TypeScript, Python и многое другое

– Разработка кроссплатформенных приложений для любой платформы

– Интеграция управления версиями — совместная работа над проектами

– Разработка с поддержкой ИИ

Среда разработки VS имеет функции, которые упрощают написание, а также управление кодом. С помощью средств разработки и ИИ, таких как GitHub Copilot и IntelliCode, можно повысить точность кода, а также его производительность с помощью лампочек, при нажатии на которые предлагаются действия, или развернуть и свернуть блоки кода с помощью структурирования.

Интегрированная среда разработки имеет следующие функции:

– Редактор кода

– Персонализация интегрированной среды разработки и редактора

– Упорядочение кода

– Советы и рекомендации [7]

Для работы были подключены следующие библиотеки:

```
using System.Data.SqlClient;
using System.IO;
using System.Linq [9];
```

Первая библиотека позволяет создавать подключение к базе данных, читать в DataSet и записывать данные, чем я и воспользовался.

Вторая библиотека даёт возможность работать с файлами, читать содержимое, редактировать, просто записывать новые данные, удалять и

создавать файлы. В случае программы данная библиотека использовалась для работы с файлами, в которых хранилась временная информация.

Третья библиотека представляет собой простой и удобный язык запросов к источнику данных. В качестве источника может выступать объект, реализующий интерфейс IEnumerable (коллекции, массивы), набор данных DataSet, а также документ XML. Но вне зависимости от типа источника LINQ позволяет применить ко всем один и тот же подход для выборки данных [8].

2.2.2. Разработка программного обеспечения

Для корректного входа в автоматизированную информационную систему распределения работ были реализованы проверки на внесение корректных данных (рисунок 2.14):

```
if ((radio_Admin.Checked == false && radio_Worker.Checked == false) || string.IsNullOrEmpty(textBox_TabellNum.Text))
{
    MessageBox.Show("Вы не ввели нобходимые данные, либо не выбрали вид просмотра!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

Рисунок 2.14 – Проверка на введение данных или отсутствие выбора режима просмотра

Создана строка подключения, для подключения к базе данных, которая была создана ранее, а также открытие подключения (рисунок 2.15).

```
name_persnum = textBox_TabellNum.Text;
string connString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\QualityBrain\\source\\repos\\WorkDistributor" +
    "\\WorkDistributor\\WorkDistribution.mdf;Integrated Security=True";
sqlConnection = new SqlConnection(connString);
sqlConnection.Open();
```

Рисунок 2.15 – Строка подключения и установление подключения по ней

Проверка на возможность установления подключения к базе данных (рисунок 2.16).

```
if (sqlConnection.State == ConnectionState.Open)
```

Рисунок 2.16 – Проверка наличия открытого подключения

Ниже представлен запрос на получение информации о работниках из базы данных (рисунок 2.17).

```
SqlCommand sqlCommandWorker = new SqlCommand("SELECT * FROM Работники ", sqlConnection);
sqlDataReaderWorker = sqlCommandWorker.ExecuteReader();
```

Рисунок 2.17 – Задание sql запроса к базе данных, где хранятся все «Работники»

Затем идёт запись в файл «WorkDistribution_Worker» (рисунок 2.18).

```
StreamWriter strWrite_Worker = new StreamWriter("WorkDistribution_Worker");
```

Рисунок 2.18 – Создание файла с именем «WorkDistribution_Worker»

Далее создание файла и запись в последующих итерациях цикла (рисунок 2.19):

```
StreamWriter strWrite_Worker = new StreamWriter("WorkDistribution_Worker");
int i = 0;
while (sqlDataReaderWorker.Read())
{
    DBWorker[i] = new DataBase_Worker();
    DBWorker[i].Personnel_Numb = sqlDataReaderWorker[0].ToString();
    DBWorker[i].Fio = sqlDataReaderWorker[1].ToString();
    DBWorker[i].Post = sqlDataReaderWorker[2].ToString();
    DBWorker[i].Ceh = sqlDataReaderWorker[3].ToString();
    DBWorker[i].Diligence = sqlDataReaderWorker[4].ToString();
    DBWorker[i].Workload = sqlDataReaderWorker[5].ToString();
    DBWorker[i].Vacation_Numb = sqlDataReaderWorker[6].ToString();

    strWrite_Worker.WriteLine(DBWorker[i].Personnel_Numb + ';' + DBWorker[i].Fio + ';'
        + DBWorker[i].Post + ';' + DBWorker[i].Ceh + ';' + DBWorker[i].Diligence + ';' + DBWorker[i].Workload
        + ';' + DBWorker[i].Vacation_Numb);

    if (textBox_TabellNum.Text == sqlDataReaderWorker[0].ToString())
    {
        if (sqlDataReaderWorker[2].ToString() == "Главный специалист")
        {
            TabellNumbWorkerClass.NameWorker = "Главный специалист";
        }
        ok = true;
        TabellNumbWorkerClass.TabellNumbWorker = textBox_TabellNum.Text;
    }
    i++;
}
i = 0;
```

Рисунок 2.19 – Цикл для чтения данных и записи в класс и в файл

Подключение к базе данных, а именно к таблице «Работы», осуществляется запись в класс и в файл (рисунок 2.20).

```
StreamWriter strWrite_Work = new StreamWriter("WorkDistribution_Work");
SqlCommand sqlCommandWork = new SqlCommand("SELECT * FROM Работы ", sqlConnection);
sqlDataReaderWork = sqlCommandWork.ExecuteReader();
while (sqlDataReaderWork.Read())
{
    DBWork[i] = new DataBase_Work();
    DBWork[i].Number_work = sqlDataReaderWork[0].ToString();
    DBWork[i].Name_work = sqlDataReaderWork[1].ToString();
    DBWork[i].Difficult = Convert.ToInt32(sqlDataReaderWork[2]);
    DBWork[i].Time = sqlDataReaderWork[3].ToString();
    DBWork[i].Comment = sqlDataReaderWork[4].ToString();

    strWrite_Work.WriteLine(DBWork[i].Number_work + ';' + DBWork[i].Name_work + ';'
        + DBWork[i].Difficult + ';' + DBWork[i].Time + ';' + DBWork[i].Comment);
    i++;
}
i = 0;
```

Рисунок 2.20 – Цикл для чтения данных и записи в класс и в файл

Созданы массивы для сортировки данных таблиц «Работники» и «Работы» (рисунок 2.21).

```
while (!strReadWorker.EndOfStream)
{
    sort_Worker[i] = new Sort();
    Worker = strReadWorker.ReadLine();
    MassWorker = Worker.Split(sp);
    int neW = Convert.ToInt32(MassWorker[4]) - Convert.ToInt32(MassWorker[5]);
    sort_Worker[i].Personnel_Numb = Convert.ToInt32(MassWorker[0]);
    sort_Worker[i].Fio = MassWorker[1];
    sort_Worker[i].Post = MassWorker[2];
    sort_Worker[i].Ceh = MassWorker[3];
    sort_Worker[i].Complete = neW;
    i++;
}
i = 0;
```

Рисунок 2.21 – Чтение данных из файла

Осуществлена сортировка и результаты этой сортировки записались в файл (рисунок 2.22):

```
var sortByComplete = from j in sort_Worker orderby j.Complete select j;
foreach (Sort sort in sortByComplete)
{
    sort_Worker[i] = sort;
    strWrite_Worker_Next.WriteLine(sort_Worker[i].Personnel_Numb + ";" + sort_Worker[i].Fio + ";" + sort_Worker[i].Post + ";"
    i++;
}
i = 0;
strReadWorker.Close();
strWrite_Worker_Next.Close();
```

Рисунок 2.22 – Сортировка и запись данных в файл

Абсолютно такой же шаг сортировки и записи был применён к данным из таблицы «Работы» (рисунок 2.23).

```
if (TabellNumbWorkerClass.NameWorker == "Главный специалист" && radio_Admin.Checked == true)
{
    this.Hide();
    AdminForm adminform = new AdminForm();
    adminform.Show();
}
else if (TabellNumbWorkerClass.NameWorker != "Главный специалист" && radio_Admin.Checked == true)
{
    MessageBox.Show("Указанный вами режим просмотра недоступен данному пользователю", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
else
{
    this.Hide();
    Worker_Form workerform = new Worker_Form();
    workerform.Show();
}
sqlConnection.Close();
```

Рисунок 2.23 – Проверки на правильность указания табельного номера и режима доступа

Создан запрос к базе данных на удаление информации из таблицы чтобы занести новую, изменённую, если же этого не происходит, то программа просто сохраняет в файл, до следующей работы (рисунок 2.24).

```
SqlCommand sqlCommand_Delete = new SqlCommand("DELETE Распределение_работ");
int k = 0;
string connString2 = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\QualityBrain\\source\\repos\\WorkDistributor" +
    "\\WorkDistributor\\WorkDistribution.mdf;Integrated Security=True";
sqlConnection = new SqlConnection(connString2);
sqlConnection.Open();
sqlCommand_Delete.Connection = sqlConnection;
sqlCommand_Delete.ExecuteNonQuery();
SqlCommand sqlCommand_Work_and_Worker = new SqlCommand($"INSERT INTO Распределение_работ VALUES ( {k + 1}, " +
    $"{work_sort[k].Number_work}, {sort_Worker[k].Personnel_Numb}, 'Правильно')", sqlConnection);
while (k != sort_Worker.Length)
{
    sqlCommand_Work_and_Worker.CommandText = $"INSERT INTO Распределение_работ VALUES ( {Convert.ToInt32(k + 1)}, " +
        $"{work_sort[k].Number_work}, {sort_Worker[k].Personnel_Numb}, N'Правильно')";
    sqlCommand_Work_and_Worker.Connection = sqlConnection;
    sqlCommand_Work_and_Worker.ExecuteNonQuery();
    k++;
}
sqlConnection.Close();
```

Рисунок 2.24 – Запросы для изменения данных

Осуществляется чтение данных из таблицы с распределением работ (рисунок 2.25):

```
string connString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\QualityBrain\\source\\repos\\WorkDistributor" +
        "\\WorkDistributor\\WorkDistribution.mdf;Integrated Security=True";
SqlConnection sqlConnection1 = new SqlConnection(connString);
sqlConnection1.Open();
SqlCommand sqlCommandDistr = new SqlCommand("SELECT * FROM Распределение_работ ", sqlConnection1);
SqlDataReader sqlDataReaderDistr = sqlCommandDistr.ExecuteReader();
while(sqlDataReaderDistr.Read())
{
    listBox_WorkDistr.Items.Add("Номер распределения: " + sqlDataReaderDistr[0] + "; " + "Номер работы: " + sqlDataReaderDistr[1] + "; "
        + "Табельный номер: " + sqlDataReaderDistr[2] + "; " + "Правильность: " + sqlDataReaderDistr[3] + ";");
}
sqlConnection1.Close();
```

Рисунок 2.25 – Чтение данных из базы данных и занесение в соответствующие элементы для упрощения взаимодействия с информацией

Запрос на добавление в таблицу с распределёнными работами указания, что распределение было ошибочным и если человек нажимает да то код выполняется, если же нет, то ничего не происходит (рисунок 2.26).

```
var result = MessageBox.Show("Вы действительно хотите отправить отчёт об ОШИБКЕ у выбранных сотрудников?", "Отправка отчёта",
    MessageBoxButtons.YesNo, MessageBoxIcon.Question);
if (result == DialogResult.Yes)
{
    SqlCommand sqlCommand_Update = new SqlCommand("UPDATE Распределение_работ ");
    string connString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\QualityBrain\\source\\repos\\WorkDistributor" +
        "\\WorkDistributor\\WorkDistribution.mdf;Integrated Security=True";
    SqlConnection sqlConnection = new SqlConnection(connString);
    sqlConnection.Open();
    for (int i = 0; i < checkedListBox_Worker.Items.Count; i++)
    {
        if (checkedListBox_Worker.GetItemChecked(i))
        {
            sqlCommand_Update.CommandText = $"UPDATE Распределение_работ SET Правильность = N'Не правильно' WHERE Номер_распределения = {i}";
            sqlCommand_Update.Connection = sqlConnection;
            sqlCommand_Update.ExecuteNonQuery();
        }
    }
    sqlConnection.Close();
}
else if (result == DialogResult.No)
{

}
```

Рисунок 2.26 – Отправка отчёта об ошибке

В форме, предназначенной для работника, осуществляется просто чтение и выведение информации из файла, в который ранее заносится информация (рисунок 2.27).

```
string[] strMass = new string[7];
char[] sp = { ';' }; string s = "";
string[] strMassWorker = new string[7];
string[] strMassWork = new string[7];
StreamReader strRead = new StreamReader("WorkDistribution_Worker_Next");
StreamReader strRead2 = new StreamReader("WorkDistribution_Work_Next");
int i = 0;
while(!strRead.EndOfStream)
{
    s = strRead.ReadLine();
    strMass = s.Split(sp);
    s = strRead2.ReadLine();
    strMassWork = s.Split(sp);
    i++;
    if (strMass[0] == TabellNumbWorkerClass.TabellNumbWorker)
    {
        listBox_Information.Items.Add("Табельный номер: " + strMass[0] + ";");
        listBox_Information.Items.Add("ФИО: " + strMass[1] + ";");
        listBox_Information.Items.Add("Должность: " + strMass[2] + ";");
        listBox_Information.Items.Add("Цех: " + strMass[3] + ";");
        listBox_Information.Items.Add("Выполнимость: " + strMass[4] + ";");
        listBox_Information.Items.Add("Номер работы: " + strMassWork[0] + ";");
    }
}
strRead.Close();
```

Рисунок 2.27 – Выведение информации из файла в элемент

2.2.3. Тестирование и отладка

В процессе разработки на каждом этапе проводилось тестирование каждого фрагмента кода на работоспособность и выполнение поставленных задач.

После разработки системы были проведены тесты и осуществлены проверки.

Была проведена проверка занесения данных в файл из базы данных, а именно из таблицы «Работники» (рисунок 2.28).

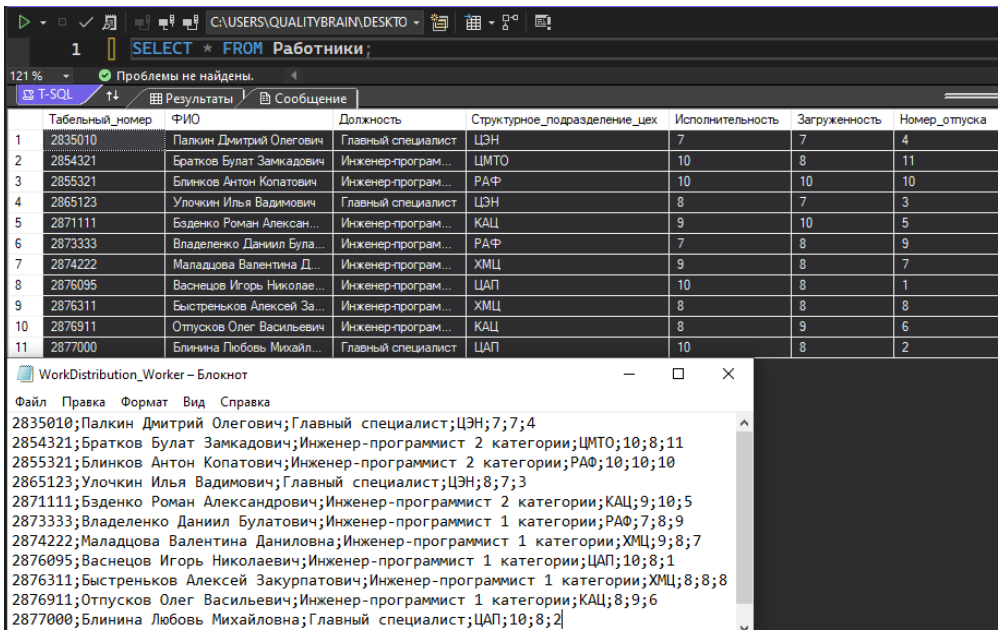


Рисунок 2.28 – Сверка результатов запроса с данными в файле «WorkDistribution_Worker»

Затем такая же проверка была осуществлена с данными из другой таблицы «Работы» (рисунок 2.29).

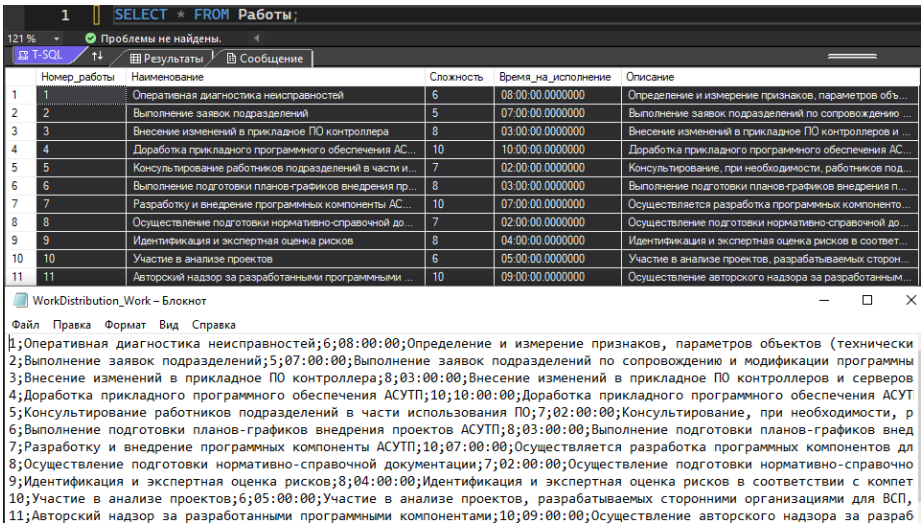


Рисунок 2.29 – Сверка результатов запроса с данными в файле «WorkDistribution_Work»

Далее была проверена таблица с записями работников и работ, назначенных данным работникам (рисунок 2.30).

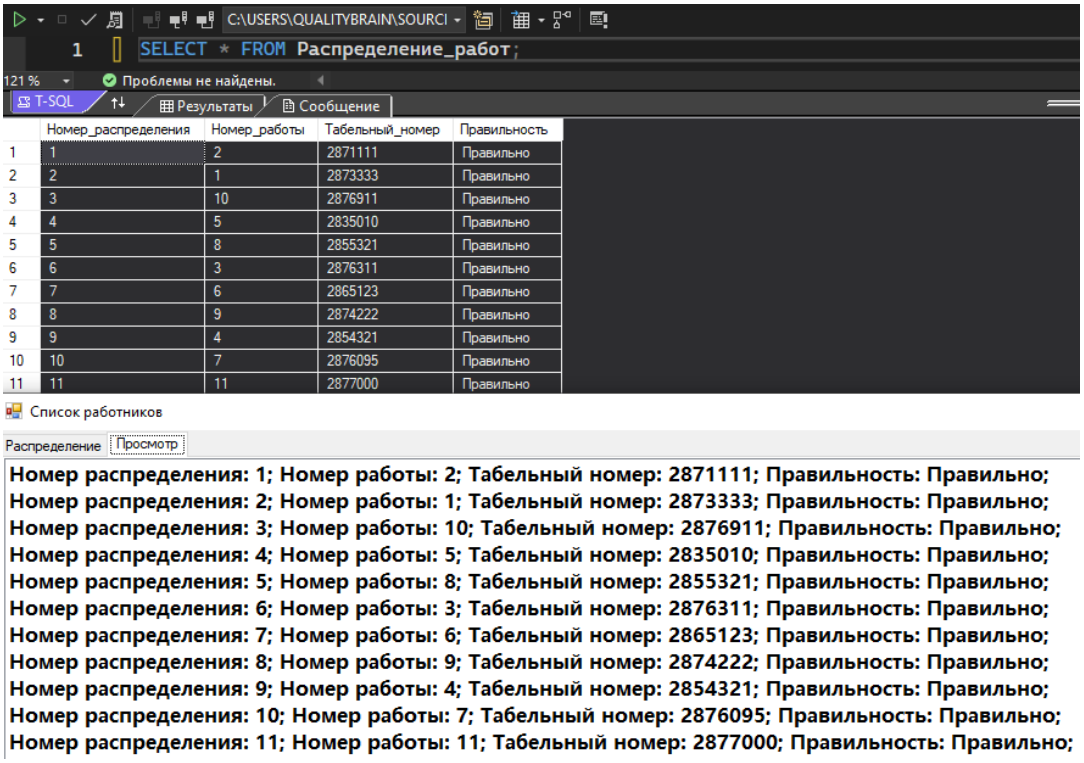


Рисунок 2.30 – Проверка данных из программы с данными после запроса

Также требовалось проверить отправляется ли отчёт (рисунок 2.31).

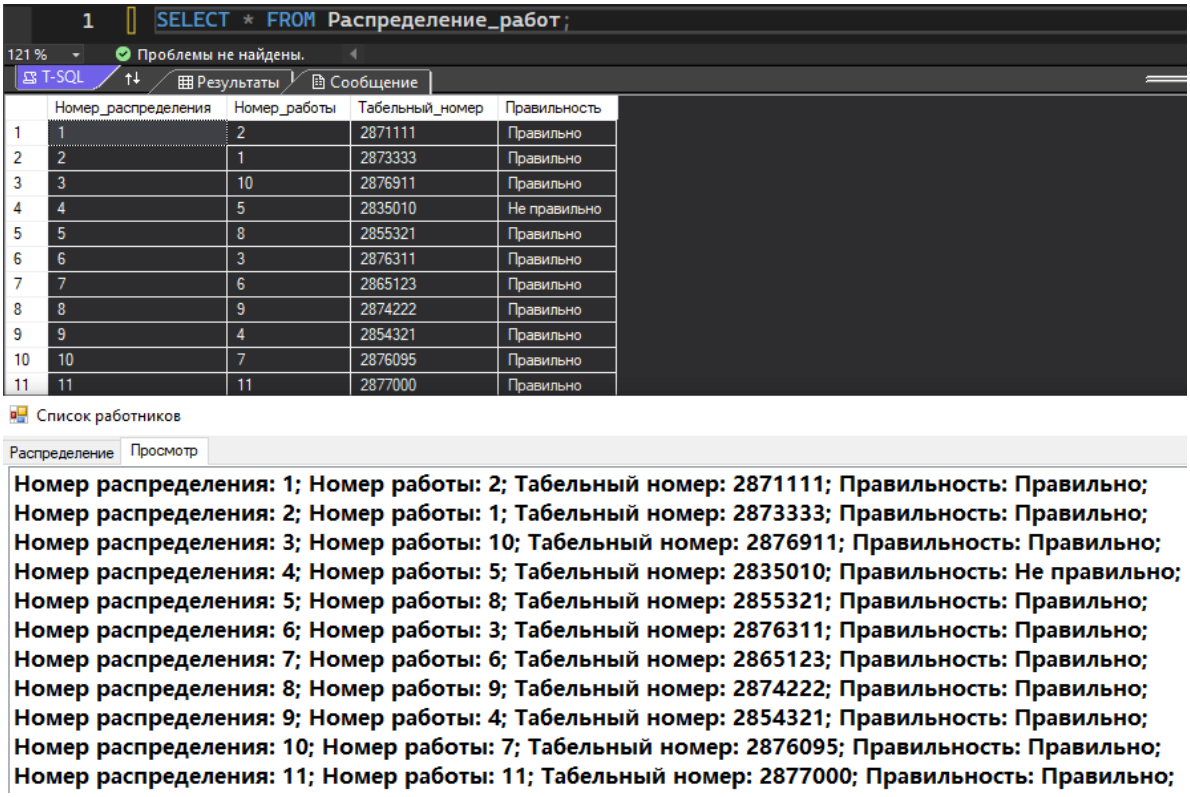
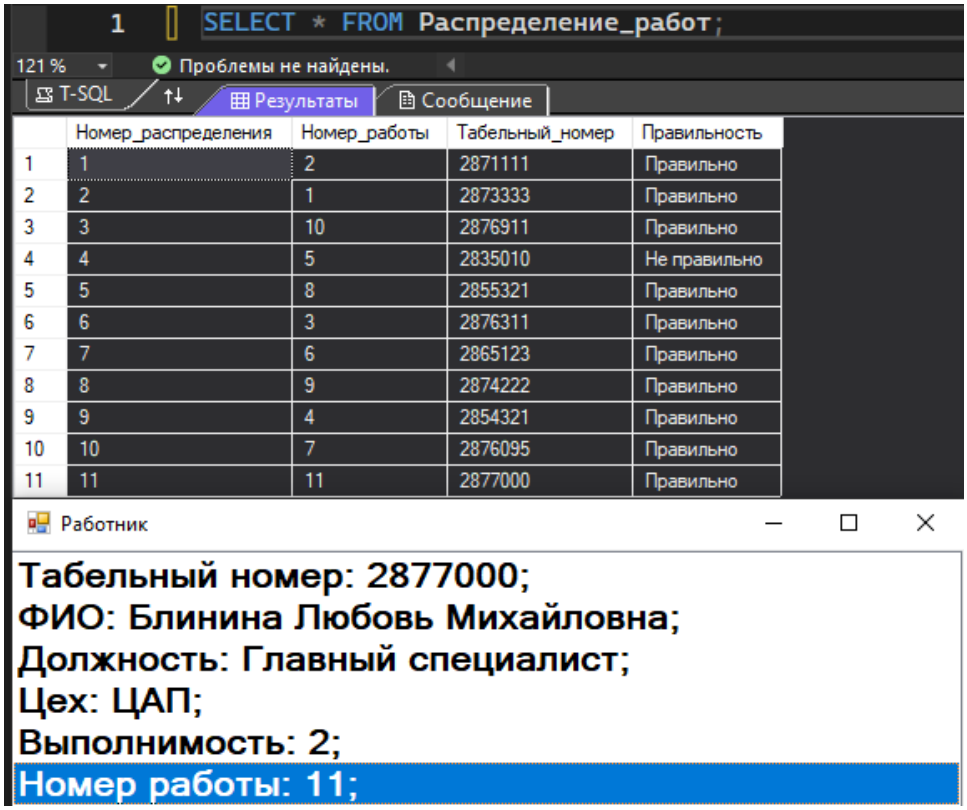


Рисунок 2.31 – Проверка результатов выведения программы с результатами в базе данных

Осуществление проверки данных о выводимых данных конкретному сотруднику с данными из базы данных (рисунок 2.32).



	Номер_распределения	Номер_работы	Табельный_номер	Правильность
1	1	2	2871111	Правильно
2	2	1	2873333	Правильно
3	3	10	2876911	Правильно
4	4	5	2835010	Не правильно
5	5	8	2855321	Правильно
6	6	3	2876311	Правильно
7	7	6	2865123	Правильно
8	8	9	2874222	Правильно
9	9	4	2854321	Правильно
10	10	7	2876095	Правильно
11	11	11	2877000	Правильно

Рисунок 2.32 – Проверка данных о конкретном сотруднике

2.3. Внедрение и апробация АИС

2.3.1. Подготовка к внедрению

Для внедрения на производство требуется обучить сотрудников работе с данной автоматизированной информационной системой распределения работ. Также следует указать значения необходимых показателей для распределения, таких как «Загруженность» и «Исполнительность» исходя из объективных нагрузок на сотрудника, а также уровень его исполнения свои должностных обязанностей [20]. Затем следует определить степень сложности каждой отдельной работы. Далее необходимо занести все полученные значения в базу данных для дальнейшего распределения работ среди сотрудников. Также требуется обучить сотрудников отдельно доступным именно ему функциям, а также проследить за их работой [17].

Для запуска программного обеспечения нужно занести необходимые данные в таблицы базы данных, а именно, в таблицу «Работы» заносится список в котором для каждой работы должен быть номер, её наименование, сложность, а также время на исполнение. Далее информация заносится в таблицу «Типы_отпуска»: его номер, тип и описание. Далее необходимо занести информацию о работниках и их отпусках в таблицу «Отпуск_работников»: первым указывается номер, затем табельный номер, номер типа отпуска, дата начала, дата окончания, а также количество дней, которое сотрудник пробудет в отпуске. Затем следует занесение данных в таблицу «Работнки»: табельный номер, ФИО, должность, структурное подразделение или цех, исполнительность и номер отпуска.

2.3.2. Тестирование на реальных задачах

После занесения данных в таблицы, программа считывает информацию и заносит в файлы, на случай если доступ к базе данных на момент использования может пропасть. Далее идёт сортировка и занесение отсортированных данных в файлы, на основании той информации, которая была ранее получена из базы данных. За основу были взяты реальные данные с места прохождения практики, и были лишь изменены табельные номера сотрудников, а также их фамилии, имена и отчества, для соблюдения конфиденциальности, но длина табельного номера, а также его тип остались без изменения.

После проведённых тестов и распределения работ, за счёт сортировки по полученным показателям система позволяет распределить работы по сотрудникам в необходимой последовательности, позволяя получить распределённый список работ и работников.

2.3.3. Оценка эффективности и результатов использования системы

Система показала себя очень хорошо в рамках отдела, где было произведено тестирование и проверка функций программы. Так как были использованы реальные данные с незначительными изменениями, которые не изменили структуру. Так как программа учитывает сразу несколько критериев для распределения конкретному работнику конкретной работы, это позволяет ускорить сам процесс распределения, снизить количество ошибок и охватить очень большое количество как сотрудников так и работ, которые должны быть направлены данным сотрудникам.

ЗАКЛЮЧЕНИЕ

В результате проведённого исследования литературы, а также уже существующих автоматизированных информационных систем, были выявлены алгоритмы, а также основные, применяемые на практике методы по распределению работ среди сотрудников.

Был рассмотрен основной принцип, в основе которого лежит распределение работы путём субъективной оценки сотрудников, их уровня компетенций, мотивации, а также опыта. Имеет место быть предвзятость на основании некоторых черт человека или взаимодействия с ним руководителя проекта. Также на одного сотрудника может быть возложено слишком много должностных обязанностей из-за оценки только высокого уровня компетенций или ответственности. Всё это в совокупности снижает эффективность производства за счёт не оптимального распределения труда.

В процессах автоматизации принятия такого рода решений есть множество плюсов, таких как непредвзятость, оценка сотрудника непосредственно по его профессиональным навыкам и снижение вероятности ошибки из-за человеческого фактора, также шанс на возникновение проблем при распределении снижается за счёт автоматизации данного процесса.

Работа представляет практическую значимость для производственных процессов, когда необходимо распределить много работы среди большого количества сотрудников при этом потратив на данный процесс оптимальное время, уменьшить влияние человеческого фактора, а также упростив получение работы, что существенно сократит время, потраченное на получение этого задания.

В данной системе имеет место быть множество улучшений, которые ещё больше снизят время на распределение, а также сложность на получение задач работниками.

Для упрощения пользования программным обеспечением, а также для сокращения количества возможных ошибок необходимо добавить ряд проверок

на случайное нажатие, а также для предотвращения случайного внесения данных.

Также можно реализовать более простой алгоритм задействующий меньшее количество файлов, таблиц.

Для создания большего уровня конфиденциальности необходимо усложнить вход и сделать проверку пользователя, также можно привязать рабочую почту к аккаунту.

Можно реализовать восстановление пароля не с помощью службы поддержки, а за счёт внутреннего функционала программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Хитрова Т.И. Проблемы распределения работ в процессе реализации инновационных задач / Т.И. Хитрова, А.С. Низовцева. — DOI: 10.17150/2411-6262.2020.11(2).15 // Baikal Research Journal. — 2020. — Т. 11, № 2.

2. Хитрова Т.И. Методы формирования состава исполнителе IT-проекта / Т.И. Хитрова, С.С. Ованесян, А.С. Низовцева. — DOI: 10.17150/2411-6262.2020.11(4).7 // Baikal Research Journal. — 2020. — Т. 11, № 4.

3. Никитин Н.С. История научных исследований в области автоматизации производства / Никитин Н.С. — DOI 10.23672/SAE.2019.2019.44451 // Санкт-Петербургский политехнический университет Петра Великого. — 2019.

4. Бойко С. В., Покровская Н. Н., Слободской А. Л., Спивак В. А. Социально-экономические вопросы мотивации сотрудников в условиях удалённой и распределённой работы / Бойко С. В., Покровская Н. Н., Слободской А. Л., Спивак В. А. — DOI: 10.35854/2219-6242-2021-1-6-17 // «Научная мысль». — 2021.

5. Баева О.Н. Особенности мотивации и стимулирования труда IT-персонала / О.Н. Баева, С.А Туренко // Исследование, систематизация, кооперация, развитие, анализ социально-экономических систем в области экономики и управления : сб. тр. II Всерос. школы-симпозиума молодых ученых / под ред. В.М. Ячменевой. — Симферополь, 2019. — С. 219–223.

6. Былков В. Повышение эффективности использования трудового потенциала в процессе актуализации системы оценки квалификаций / В. Былков // Global and Regional Research. — 2019. — Т. 1, № 3. — С. 76–81.

7. [Электронный ресурс] / Блог компании Microsoft; — Режим доступа: https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022, свободный. (Дата обращения: 18.05.2024)

8. [Электронный ресурс] / Metanit сайт о программировании; — Режим доступа: https://metanit.com/sharp/tutorial/15.1.php, свободный. (Дата обращения: 15.05.2024).

9. [Электронный ресурс] / System.Data.SqlClient Namespace; — Режим доступа: https://learn.microsoft.com/en-en/dotnet/api/system.data.sqlclient?view=dotnet-plat-ext-8.0, свободный. (Дата обращения: 20.05.2024)

10. [Электронный ресурс] / Johnson’s Rule in Sequencing Problems; — Режим доступа: https://www.geeksforgeeks.org/johnsons-rule-in-sequencing-problems, свободный. (Дата обращения: 01.05.2024)

11. [Электронный ресурс] / How to Create a Work Breakdown Structure in 6 Steps; — Режим доступа: https://toggl.com/blog/work-breakdown-structure, свободный (Дата обращения: 01.05.2024)

12. [Электронный ресурс] / Распределение задач; — Режим доступа: https://www.leadertask.ru/blog/raspredelenie-zadach, свободный (Дата обращения: 06.05.2024)

13. [Электронный ресурс] / Как эффективно распределять нагрузку среди сотрудников; — Режим доступа: https://vc.ru/hr/485130-kak-effektivno-raspredelyat-nagruzku-sredi-sotrudnikov, свободный (Дата обращения: 04.05.2024)

14. [Электронный ресурс] / Распределение обязанностей: распределение должностных обязанностей работников разного уровня, порядок составления и подписания приказов, руководство для руководителя; — Режим доступа: https://www.seeneco.com/ru/blog/raspredelenie-dolzhnostnyh-obyazannostej, свободный (Дата обращения: 09.05.2024)

15. [Электронный ресурс] / Workload Management: A Complete Guide For Creative Agencies; — Режим доступа: https://toggl.com/blog/workload-management, свободный (Дата обращения: 11.05.2024)

16. [Электронный ресурс] / Распределение задач между сотрудниками; — Режим доступа: https://www.kck.ru/solutions/raspredelenie-zadach, свободный (Дата обращения: 11.05.2024)

17. [Электронный ресурс] / What is distributed work and how can it benefit your enterprise?; — Режим доступа: https://envoy.com/workplace-compliance-security-safety/what-is-distributed-work, свободный (Дата обращения: 12.05.2024)

18. [Электронный ресурс] / Work Breakdown Structures in PM: Everything You Need to Know [Free Templates]; — Режим доступа: https://hubstaff.com/blog/work-breakdown-structure, свободный (Дата обращения 13.05.2024)

19. [Электронный ресурс] / What is a Work Breakdown Structure (WBS)?; — Режим доступа: https://project-management.com/work-breakdown-structure-wbs, свободный (Дата обращения: 15.05.2024)

20. [Электронный ресурс] / What Is a Work Breakdown Structure in Project Management?; — Режим доступа: https://www.smarttask.io/blog/work-breakdown-structure, свободный (Дата обращения: 10.05.2024)

ПРИЛОЖЕНИЕ

Приложение

```
CREATE DATABASE WorkDistribution;
GO
USE WorkDistribution;
CREATE TABLE Типы_отпуска
(
     Номер_типа INT IDENTITY PRIMARY KEY,
     Тип_отпуска NVARCHAR(200),
     Описание NVARCHAR(200)
);

CREATE TABLE Отпуск_работников
(
     Номер_отпуска INT,
     Табельный_номер INT,
     Номер_типа INT,
     Дата_начала DATE,
     Дата_окончания DATE,
     Количество_дней INT,
     PRIMARY KEY (Номер_отпуска),
     FOREIGN KEY (Номер_типа) REFERENCES Типы_отпуска
(Номер_типа)
);

CREATE TABLE Работники
(
     Табельный_номер INT PRIMARY KEY,
     ФИО NVARCHAR(200),
     Должность NVARCHAR(200),
     Структурное_подразделение_цех NVARCHAR(200),
     Исполнительность INT,
     Загруженность INT,
     Номер_отпуска INT,
     FOREIGN KEY (Номер_отпуска) REFERENCES Отпуск_работников
(Номер_отпуска)
);

CREATE TABLE Работы
(
     Номер_работы INT IDENTITY PRIMARY KEY,
     Наименование NVARCHAR(200),
     Сложность INT,
     Время_на_исполнение TIME,
     Описание NVARCHAR(300)
);
```

```
CREATE TABLE Распределение_работ
(
    Номер_распределения INT,
    Номер_работы INT,
    Табельный_номер INT,
    Правильность NVARCHAR(20),
    PRIMARY KEY (Номер_распределения)
);

CREATE TABLE Распределение_Работники
(
    Табельный_номер INT,
    Номер_распределения INT,
    PRIMARY KEY (Табельный_номер, Номер_распределения),
    FOREIGN KEY (Номер_распределения) REFERENCES
Распределение_работ(Номер_распределения),
    FOREIGN KEY (Табельный_номер) REFERENCES
Работники(Табельный_номер)
);

CREATE TABLE Распределение_Работа
(
    Номер_работы INT,
    Номер_распределения INT,
    PRIMARY KEY (Номер_работы, Номер_распределения),
    FOREIGN KEY (Номер_работы) REFERENCES
Работы(Номер_работы),
    FOREIGN KEY (Номер_распределения) REFERENCES
Распределение_работ(Номер_распределения)
);
INSERT INTO Типы_отпуска VALUES
(
    N'Ежегодный дополнительный отпуск без сохранения
заработной платы',
    N'Дополнительный отпуск, допустим, по переработкам'
),
(
    N'Отпуск без сохранения заработной платы',
    N'Ещё так называемый "отпуск за свой счёт", берётся
человеком по причинам, которые не подпадают под остальные виды
отпусков'
),
(
    N'Отпуск по беременности и родам',
    N'Иногда называемый "декретным", отпуск, который берёт
женщина для того, чтобы процесс вынашивания и родов не мешал
работе'
),
(
    N'Отпуск при совмещении работы с обучением',
    N'Требуется, когда совмещение работы с учёбой без потери
часов учёбы или работы - невозможно'
),
```

```
    (
        N'Отпуск по уходу за ребенком',
        N'Требуется для сохранения рабочего места в связи с
необходимостью ухода за ребёнком'
    );
    INSERT INTO Отпуск_работников VALUES
    (
        1, 2876095, 1, '2024-03-07', '2023-04-01', 26
    ),
    (
        2, 2877000, 2, '2024-01-25', '2024-02-26', 33
    ),
    (
        3, 2865123, 2, '2024-01-31', '2024-02-08', 40
    ),
    (
        4, 2835010, 3, '2024-01-06', '2024-02-01', 25
    ),
    (
        5, 2871111, 4, '2024-03-18', '2024-04-26', 39
    ),
    (
        6, 2876911, 5, '2024-02-12', '2024-03-14', 33
    ),
    (
        7, 2874222, 5, '2024-04-27', '2024-05-22', 26
    ),
    (
        8, 2876311, 5, '2024-04-04', '2024-05-05', 33
    ),
    (
        9, 2873333, 1, '2024-02-04', '2024-03-29', 26
    ),
    (
        10, 2855321, 3, '2024-03-10', '2024-04-11', 32
    ),
    (
        11, 2854321, 4, '2024-05-10', '2024-06-04', 26
    );


    INSERT INTO Работники VALUES
    (
        2876095,  N'Васнецов  Игорь  Николаевич',  N'Инженер-
программист 1 категории', N'ЦАП', 10, 8, 1
    ),
    (
        2877000,  N'Блинина  Любовь  Михайловна',  N'Главный
специалист', N'ЦАП', 10, 8, 2
    ),
    (
        2865123,  N'Улочкин  Илья  Вадимович',  N'Главный
специалист', N'ЦЭН', 8, 7, 3
```

```
    ),
    (
        2835010, N'Палкин Дмитрий Олегович', N'Главный специалист', N'ЦЭН', 7, 7, 4
    ),
    (
        2871111, N'Бзденко Роман Александрович', N'Инженер-программист 2 категории', N'КАЦ', 9, 10, 5
    ),
    (
        2876911, N'Отпусков Олег Васильевич', N'Инженер-программист 1 категории', N'КАЦ', 8, 9, 6
    ),
    (
        2874222, N'Маладцова Валентина Даниловна', N'Инженер-программист 1 категории', N'ХМЦ', 9, 8, 7
    ),
    (
        2876311, N'Быстреньков Алексей Закурпатович', N'Инженер-программист 1 категории', N'ХМЦ', 8, 8, 8
    ),
    (
        2873333, N'Владеленко Даниил Булатович', N'Инженер-программист 1 категории', N'РАФ', 7, 8, 9
    ),
    (
        2855321, N'Блинков Антон Копатович', N'Инженер-программист 2 категории', N'РАФ', 10, 10, 10
    ),
    (
        2854321, N'Братков Булат Замкадович', N'Инженер-программист 2 категории', N'ЦМТО', 10, 8, 11
    );
    INSERT INTO Работы VALUES
    (
        N'Оперативная диагностика неисправностей', 6, '08:00:00', N'Определение и измерение признаков, параметров объектов (технических систем и их компонентов) для оперативной оценки их состояния на заданном этапе функционирования'
    ),
    (
        N'Выполнение заявок подразделений', 5, '07:00:00', N'Выполнение заявок подразделений по сопровождению и модификации программных компонентов АСУТП'
    ),
    (
        N'Внесение изменений в прикладное ПО контроллера', 8, '03:00:00', N'Внесение изменений в прикладное ПО контроллеров и серверов АСУТП, программы человеко-машинного интерфейса (ЧМИ)'
    ),
    (
```

```
        N'Доработка прикладного программного обеспечения АСУТП',
10, '10:00:00', N'Доработка прикладного программного обеспечения
АСУТП в процессе опытной и промышленной эксплуатации'
    ),
    (
        N'Консультирование работников подразделений в части
использования ПО', 7, '02:00:00', N'Консультирование, при
необходимости, работников подразделений и подрядных организаций в
части использования ПО, выполнение работ по актуализации
технической документации'
    ),
    (
        N'Выполнение подготовки планов-графиков внедрения
проектов АСУТП', 8, '03:00:00', N'Выполнение подготовки планов-
графиков внедрения проектов АСУТП, оценку нагрузку по выполняемым
работам, выдачу рекомендации по назначению исполнителей, оценку
сроков реализации этапов внедрения'
    ),
    (
        N'Разработку и внедрение программных компоненты АСУТП',
10, '07:00:00', N'Осуществляется разработка программных
компонентов для АСУТП и их внедрение'
    ),
    (
        N'Осуществление подготовки нормативно-справочной
документации', 7, '02:00:00', N'Осуществление подготовки
нормативно-справочной документации, обновлений текущей технической
документации АСУТП'
    ),
    (
        N'Идентификация и экспертная оценка рисков', 8,
'04:00:00', N'Идентификация и экспертная оценка рисков в
соответствии с компетенциями по направлениям деятельности.
Проведение анализа факторов, оказывающих влияние на возникновение
негативных событий. Своевременное информирование о рисках
непосредственного руководителя'
    ),
    (
        N'Участие в анализе проектов', 6, '05:00:00', N'Участие в
анализе проектов, разрабатываемых сторонними организациями для
ВСП, подготовка замечаний по проектной документации.'
    ),
    (
        N'Авторский надзор за разработанными программными
компонентами', 10, '09:00:00', N'Осуществление авторского надзора
за разработанными программными компонентами, совершенствование их
при необходимости'
    ); (
        N'Постановка задач в области автоматизации
технологических процессов', 9, '01:00:00', N'Осуществление
постановки задач в области автоматизации технологических
процессов'
    ),
```

```
    (
        N'Поиск оптимальных решений по автоматизации технологического производства', 6, '08:00:00', N'Поиск оптимальных решений по автоматизации технологического производства с целью стандартизации программных продуктов и реализации единой политики построения проектов'
    ),
    (
        N'Подготовка предложений по планированию деятельности отдела', 8, '05:00:00', N'Контроль норм производительности, расчёт норм запасов по готовой продукции, сырью и комплектующим с учётом сезонности; Контроль простоев оборудования; Автоматизация процессов планирования; Подготовка отчётов'
    ),
    (
        N'Внести изменения в прикладное ПО контроллеров и серверов АСУТП', 10, '12:00:00', N'Внесение изменений в прикладное ПО контроллеров и серверов АСУТП, программы человеко-машинного интерфейса (ЧМИ)'
    ),
    (
        N'Доработка прикладного программного обеспечения АСУТП', 9, '11:00:00', N'Доработка прикладного программного обеспечения АСУТП в процессе опытной и промышленной эксплуатации'
    ),
    (
        N'Идентификация и экспертная оценка рисков', 8, '08:00:00', N'Идентификация и экспертная оценка рисков в соответствии с компетенциями по направлениям деятельности. Проведение анализа факторов, оказывающих влияние на возникновение негативных событий. Своевременное информирование о рисках непосредственного руководителя'
    ),
    (
        N'Осуществление руководства', 10, '08:00:00', N'Осуществление руководства и установление обязанностей работников, находящихся в его непосредственном подчинении'
    ),
    (
        N'Обеспечение и контроль выполнения', 7, '08:00:00', N'Обеспечение и контроль выполнения работниками отдела их должностных обязанностей, отдельных разовых служебных заданий и поручений'
    );
```

JoinForm

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Linq;
using System.Windows.Forms;

namespace WorkDistributor
{
    public partial class JoinForm : Form
    {
        public JoinForm()
        {
            InitializeComponent();
        }

        private SqlConnection sqlConnection = null;
        public string DBread = ""; public string name_persnum
= "";
        SqlDataReader sqlDataReaderWorker = null;
SqlDataReader sqlDataReaderWork = null;
        DataBase_Worker[] DBWorker = new DataBase_Worker[11];
DataBase_Work[] DBWork = new DataBase_Work[11];
        public bool ok = false;
        private void radioWorker_CheckedChanged(object
sender, EventArgs e)
        {
            textBox_TabellNum.Text = "";
        }
        private void radioAdmin_CheckedChanged(object sender,
EventArgs e)
        {
            textBox_TabellNum.Text = "";
        }
```

```
            private void Next_Click(object sender, EventArgs e)
            {
                if ((radio_Admin.Checked == false &&
radio_Worker.Checked == false) ||
string.IsNullOrEmpty(textBox_TabellNum.Text))
                {
                    MessageBox.Show("Вы не ввели нобходимые
данные, либо не выбрали вид просмотра!", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);

                }
                else
                {
                    name_persnum = textBox_TabellNum.Text;
                    string connString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\Quality
Brain\\source\\repos\\WorkDistributor" +

"\\WorkDistributor\\WorkDistribution.mdf;Integrated
Security=True";
                    sqlConnection = new
SqlConnection(connString);
                    sqlConnection.Open();

                    if (sqlConnection.State ==
ConnectionState.Open)
                    {
                        SqlCommand sqlCommandWorker = new
SqlCommand("SELECT * FROM Работники ", sqlConnection);
                        sqlDataReaderWorker =
sqlCommandWorker.ExecuteReader();
                        StreamWriter strWrite_Worker = new
StreamWriter("WorkDistribution_Worker");
                        int i = 0;
                        while (sqlDataReaderWorker.Read())
```

```
                        {
                            DBWorker[i] = new DataBase_Worker();
                            DBWorker[i].Personnel_Numb          =
sqlDataReaderWorker[0].ToString();
                            DBWorker[i].Fio                     =
sqlDataReaderWorker[1].ToString();
                            DBWorker[i].Post                    =
sqlDataReaderWorker[2].ToString();
                            DBWorker[i].Ceh                     =
sqlDataReaderWorker[3].ToString();
                            DBWorker[i].Diligence               =
sqlDataReaderWorker[4].ToString();
                            DBWorker[i].Workload                =
sqlDataReaderWorker[5].ToString();
                            DBWorker[i].Vacation_Numb           =
sqlDataReaderWorker[6].ToString();


strWrite_Worker.WriteLine(DBWorker[i].Personnel_Numb   +   ';'   +
DBWorker[i].Fio + ';'
                                +   DBWorker[i].Post   +   ';'   +
DBWorker[i].Ceh   +   ';'   +   DBWorker[i].Diligence   +   ';'   +
DBWorker[i].Workload
                                +              ';'              +
DBWorker[i].Vacation_Numb);

                            if      (textBox_TabellNum.Text      ==
sqlDataReaderWorker[0].ToString())
                            {
                                if
(sqlDataReaderWorker[2].ToString() == "Главный специалист")
                                {

TabellNumbWorkerClass.NameWorker = "Главный специалист";
                                }
                                ok = true;
```

```
TabellNumbWorkerClass.TabellNumbWorker = textBox_TabellNum.Text;
                    }
                    i++;
                }
                i = 0;
                sqlDataReaderWorker.Close();
                StreamWriter strWrite_Work = new
StreamWriter("WorkDistribution_Work");
                SqlCommand sqlCommandWork = new
SqlCommand("SELECT * FROM Работы ", sqlConnection);
                sqlDataReaderWork =
sqlCommandWork.ExecuteReader();
                while (sqlDataReaderWork.Read())
                {
                    DBWork[i] = new DataBase_Work();
                    DBWork[i].Number_work =
sqlDataReaderWork[0].ToString();
                    DBWork[i].Name_work =
sqlDataReaderWork[1].ToString();
                    DBWork[i].Difficult =
Convert.ToInt32(sqlDataReaderWork[2]);
                    DBWork[i].Time =
sqlDataReaderWork[3].ToString();
                    DBWork[i].Comment =
sqlDataReaderWork[4].ToString();


strWrite_Work.WriteLine(DBWork[i].Number_work + ';' +
DBWork[i].Name_work + ';'
                        + DBWork[i].Difficult + ';' +
DBWork[i].Time + ';' + DBWork[i].Comment);
                    i++;
                }
                i = 0;
                strWrite_Worker.Close();
```

```
                        strWrite_Work.Close();
                        sqlDataReaderWork.Close();
                        Sort[]       sort_Worker       =       new
Sort[DBWorker.Length];
                        string[] MassWorker = new string[11];
                        string Worker = ""; char[] sp = {';'};
                        StreamWriter  strWrite_Worker_Next  =  new
StreamWriter("WorkDistribution_Worker_Next");
                        StreamReader    strReadWorker    =    new
StreamReader("WorkDistribution_Worker");
                        while (!strReadWorker.EndOfStream)
                        {
                            sort_Worker[i] = new Sort();
                            Worker = strReadWorker.ReadLine();
                            MassWorker = Worker.Split(sp);
                            int                neW                =
Convert.ToInt32(MassWorker[4]) - Convert.ToInt32(MassWorker[5]);
                            sort_Worker[i].Personnel_Numb        =
Convert.ToInt32(MassWorker[0]);
                            sort_Worker[i].Fio = MassWorker[1];
                            sort_Worker[i].Post = MassWorker[2];
                            sort_Worker[i].Ceh = MassWorker[3];
                            sort_Worker[i].Complete = neW;
                            i++;
                        }
                        i = 0;
                        var   sortByComplete   =   from   j   in
sort_Worker orderby j.Complete select j;
                        foreach (Sort sort in sortByComplete)
                        {
                            sort_Worker[i] = sort;

strWrite_Worker_Next.WriteLine(sort_Worker[i].Personnel_Numb  + ";"
+  sort_Worker[i].Fio  +  ";"  +  sort_Worker[i].Post  +  ";"  +
sort_Worker[i].Ceh + ";" + sort_Worker[i].Complete + ";");
                            i++;
```

```
                        }
                        i = 0;
                        strReadWorker.Close();
                        strWrite_Worker_Next.Close();
                        Work_Sort[]      work_sort      =      new
Work_Sort[DBWork.Length];
                        StreamReader     strRead_Work     =     new
StreamReader("WorkDistribution_Work");
                        StreamWriter    strWrite_Work_2    =    new
StreamWriter("WorkDistribution_Work_Next");
                        string str_Work = ""; string[] MassWork =
new string[11];
                        while (!strRead_Work.EndOfStream)
                        {
                            work_sort[i] = new Work_Sort();
                            str_Work = strRead_Work.ReadLine();
                            MassWork = str_Work.Split(sp);
                            work_sort[i].Number_work             =
Convert.ToInt32(MassWork[0]);
                            work_sort[i].Name_work = MassWork[1];
                            work_sort[i].Difficult               =
Convert.ToInt32(MassWork[2]);
                            work_sort[i].Time = MassWork[3];
                            work_sort[i].Comment = MassWork[4];
                            i++;
                        }
                        i = 0;
                        var sortByDifficult = from j in work_sort
orderby j.Difficult select j;
                        foreach(Work_Sort       worksort       in
sortByDifficult)
                        {
                            work_sort[i] = worksort;

strWrite_Work_2.WriteLine(work_sort[i].Number_work    +    ";"    +
```

```
work_sort[i].Name_work + ";" + work_sort[i].Difficult + ";" + work_sort[i].Time + ";" + work_sort[i].Comment + ";");
                            i++;
                        }
                        strWrite_Work_2.Close();
                        if (TabellNumbWorkerClass.NameWorker == "Главный специалист" && radio_Admin.Checked == true)
                        {
                            this.Hide();
                            AdminForm adminform = new AdminForm();
                            adminform.Show();
                        }
                        else if (TabellNumbWorkerClass.NameWorker != "Главный специалист" && radio_Admin.Checked == true)
                        {
                            MessageBox.Show("Указанный вами режим просмотра недоступен данному пользователю", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                            strRead_Work.Close();
                            strWrite_Work_2.Close();
                        }
                        else
                        {
                            this.Hide();
                            Worker_Form workerform = new Worker_Form();
                            workerform.Show();
                        }
                        sqlConnection.Close();

                        SqlCommand sqlCommand_Delete = new SqlCommand("DELETE Распределение_работ");
                        int k = 0;
```

```
                        string connString2 = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\Quality
Brain\\source\\repos\\WorkDistributor" +

"\\WorkDistributor\\WorkDistribution.mdf;Integrated
Security=True";
                        sqlConnection = new
SqlConnection(connString2);
                        sqlConnection.Open();
                        sqlCommand_Delete.Connection =
sqlConnection;
                        sqlCommand_Delete.ExecuteNonQuery();
                        SqlCommand sqlCommand_Work_and_Worker =
new SqlCommand($"INSERT INTO Распределение_работ VALUES ( {k + 1},
{work_sort[k].Number_work}, {sort_Worker[k].Personnel_Numb},
'Правильно')", sqlConnection);
                        while (k != sort_Worker.Length)
                        {

sqlCommand_Work_and_Worker.CommandText = $"INSERT INTO
Распределение_работ VALUES ( {Convert.ToInt32(k + 1)},
{work_sort[k].Number_work}, {sort_Worker[k].Personnel_Numb},
N'Правильно')";
                            sqlCommand_Work_and_Worker.Connection
= sqlConnection;

sqlCommand_Work_and_Worker.ExecuteNonQuery();
                            k++;
                        }
                        sqlConnection.Close();
                    }
                    else
                    {
                        MessageBox.Show("Доступ к базе данных не
был получен, информация будет подгружена с файла",
"Предупредление", MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

```
                        if
(!File.Exists("WorkDistribution_Worker"))
                        {
                            MessageBox.Show("Доступ к базе данных
не был получен, также отсутствует ФАЙЛ!", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                        else
                        {
                            StreamReader strRead = new
StreamReader("WorkDistribution_Worker");
                            DBread = "";
                            while (!strRead.EndOfStream)
                            {
                                DBread =
strRead.Read().ToString();
                            }

                            this.Hide();
                            if (radio_Admin.Checked == true)
                            {
                                AdminForm adminForm = new
AdminForm();
                                adminForm.Show();
                            }
                            else if (radio_Worker.Checked ==
true)
                            {
                                Worker_Form workerForm = new
Worker_Form();
                                workerForm.Show();
                            }
                        }
                    }

                }
```

```
            }

            private   void   textBox_TabellNum_TextChanged(object
sender, EventArgs e)
            {
                if (textBox_TabellNum.Text != null)
                {
                    Next.Enabled = true;
                }
                else
                {
                    Next.Enabled = false;
                }
            }

            private   void   JoinForm_FormClosing(object   sender,
FormClosingEventArgs e)
            {
                JoinForm joinForm = new JoinForm();
                joinForm.Close();
            }
        }
    }
```

AdminForm

```
using System;
using System.Data.SqlClient;
using System.IO;
using System.Windows.Forms;

namespace WorkDistributor
{
   public partial class AdminForm : Form
   {
      public AdminForm()
      {
         InitializeComponent();
      }
      char[] sp = { ';' };
      DataBase_Work[] DBWork= new DataBase_Work[11];
      DataBase_Worker[] DBWorker = new DataBase_Worker[11];
      Sort[] sort = new Sort[11];
      private void AdminForm_Load(object sender, EventArgs e)
      {
         if(File.Exists("WorkDistribution_Worker") && File.Exists("WorkDistribution_Work"))
         {
```

```
            char[] sp = { ';' };
            string[] MassWorker = new string[11];
            string Worker = "";
            StreamReader strReadWorker = new StreamReader("WorkDistribution_Worker_Next");
            while(!strReadWorker.EndOfStream)
            {
               Worker = strReadWorker.ReadLine();
               MassWorker = Worker.Split(sp);
               checkedListBox_Worker.Items.Add("Табельный номер: " + MassWorker[0] + "; ФИО: " + MassWorker[1] + "; Должность: "
                  + MassWorker[2] + "; Цех: " + MassWorker[3] + "; Выполнимость: " + MassWorker[4] + ';');
            }
            strReadWorker.Close();
            StreamReader strReadWork = new StreamReader("WorkDistribution_Work_Next");
            string Work = ""; string[] MassWork = new string[11];
            while (!strReadWork.EndOfStream)
            {
               Work = strReadWork.ReadLine();
               MassWork = Work.Split(sp);
               listBox_Work.Items.Add("Номер работы: " + MassWork[0] + "; Наименование работы: " + MassWork[1] + "; Сложность: "
                  + MassWork[2] + "; Время: " + MassWork[3] + "; Описание: " + MassWork[4] + ';');
            }
            strReadWork.Close();
         }
         else
         {
            MessageBox.Show("Отсутствует ФАЙЛ! Пожалуйста перезайдите в программу для отгрузки информации из базы данных", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
         }
         string connString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\QualityBrain\\source\\repos\\WorkDistributor" +
               "\\WorkDistributor\\WorkDistribution.mdf;Integrated Security=True";
         SqlConnection sqlConnection1 = new SqlConnection(connString);
         sqlConnection1.Open();
         SqlCommand sqlCommandDistr = new SqlCommand("SELECT * FROM Распределение_работ ", sqlConnection1);
         SqlDataReader sqlDataReaderDistr = sqlCommandDistr.ExecuteReader();
         while(sqlDataReaderDistr.Read())
         {
            listBox_WorkDistr.Items.Add("Номер распределения: " + sqlDataReaderDistr[0] + "; " + "Номер работы: " + sqlDataReaderDistr[1] + "; " + "Табельный номер: " + sqlDataReaderDistr[2] + "; " + "Правильность: " + sqlDataReaderDistr[3] + ";");
         }
         sqlConnection1.Close();
      }

      private void checkedListBox_Distr_SelectedIndexChanged(object sender, EventArgs e)
      {
         if(checkedListBox_Worker.CheckedItems.Count == 0)
         {
            button_Order.Enabled = false;
         }
         else
         {
            button_Order.Enabled = true;
         }
      }

      private void AdminForm_FormClosing(object sender, FormClosingEventArgs e)
      {
         Application.Exit();
      }

      private void button_Order_Click(object sender, EventArgs e)
      {
```

```
        var result = MessageBox.Show("Вы действительно хотите отправить отчёт об ОШИБКЕ у выбранных
сотрудников?", "Отправка отчёта", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (result == DialogResult.Yes)
        {
            SqlCommand sqlCommand_Update = new SqlCommand("UPDATE Распределение_работ ");
            string connString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\QualityBrain\\source\\repos\\WorkDistributor" +
                "\\WorkDistributor\\WorkDistribution.mdf;Integrated Security=True";
            SqlConnection sqlConnection = new SqlConnection(connString);
            sqlConnection.Open();
            for (int i = 0; i < checkedListBox_Worker.Items.Count; i++)
            {
                if (checkedListBox_Worker.GetItemChecked(i))
                {
                    sqlCommand_Update.CommandText = $"UPDATE Распределение_работ SET Правильность = N'Не
правильно' WHERE Номер_распределения = {i + 1}";
                    sqlCommand_Update.Connection = sqlConnection;
                    sqlCommand_Update.ExecuteNonQuery();
                }
            }
            sqlConnection.Close();
        }
        else if (result == DialogResult.No)
        {

        }
    }

    private void button_Update_Click(object sender, EventArgs e)
    {
        checkedListBox_Worker.Items.Clear();
        listBox_Work.Items.Clear();
        if (File.Exists("WorkDistribution_Worker_Next") && File.Exists("WorkDistribution_Work_Next"))
        {
            char[] sp = { ';' };
            string[] MassWorker = new string[11];
            string Worker = "";
            StreamReader strReadWorker = new StreamReader("WorkDistribution_Worker_Next");
            while (!strReadWorker.EndOfStream)
            {
                Worker = strReadWorker.ReadLine();
                MassWorker = Worker.Split(sp);
                checkedListBox_Worker.Items.Add("Табельный номер: " + MassWorker[0] + "; ФИО: " + MassWorker[1] + "
Должность: "
                    + MassWorker[2] + "; Цех: " + MassWorker[3] + "; Выполнимость: " + MassWorker[4] + ';');
            }
            strReadWorker.Close();
            StreamReader strReadWork = new StreamReader("WorkDistribution_Work_Next");
            string Work = ""; string[] MassWork = new string[11];
            while (!strReadWork.EndOfStream)
            {
                Work = strReadWork.ReadLine();
                MassWork = Work.Split(sp);
                listBox_Work.Items.Add("Номер работы: " + MassWork[0] + "; Наименование работы: " + MassWork[1] + "
Сложность: "
                    + MassWork[2] + "; Время: " + MassWork[3] + "; Описание: " + MassWork[4] + ';');
            }
            strReadWork.Close();
            listBox_WorkDistr.Items.Clear();
            string connString = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\QualityBrain\\source\\repos\\WorkDistributor" +
                "\\WorkDistributor\\WorkDistribution.mdf;Integrated Security=True";
            SqlConnection sqlConnection1 = new SqlConnection(connString);
            sqlConnection1.Open();
            SqlCommand sqlCommandDistr = new SqlCommand("SELECT * FROM Распределение_работ ",
sqlConnection1);
```

```
            SqlDataReader sqlDataReaderDistr = sqlCommandDistr.ExecuteReader();
            while (sqlDataReaderDistr.Read())
            {
                listBox_WorkDistr.Items.Add("Номер распределения: " + sqlDataReaderDistr[0] + "; " + "Номер работы: " +
sqlDataReaderDistr[1] + "; " + "Табельный номер: " + sqlDataReaderDistr[2] + "; " + "Правильность: " +
sqlDataReaderDistr[3] + ";");
            }
            sqlConnection1.Close();
        }
        else
        {
            MessageBox.Show("Отсутствует ФАЙЛ! Пожалуйста перезайдите в программу для отгрузки информации из
базы данных", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
  }
}
```

WorkerForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WorkDistributor
{
    public partial class Worker_Form : Form
    {
        public Worker_Form()
        {
            InitializeComponent();
        }
        private    void    Worker_Form_Load(object    sender,
EventArgs e)
        {
            string[] strMass = new string[7];
            char[] sp = { ';' }; string s = "";
            string[] strMassWorker = new string[7];
            StreamReader         strRead         =         new
StreamReader("WorkDistribution_Worker");
            while(!strRead.EndOfStream)
            {
                s = strRead.ReadLine();
                strMass = s.Split(sp);
                if                (strMass[0]                ==
TabellNumbWorkerClass.TabellNumbWorker)
                {
                    //strMass = strMassWorker;
                    listBox_Information.Items.Add("Табельный
номер: " + strMass[0] + ";");
                    listBox_Information.Items.Add("ФИО:   "   +
strMass[1] + ";");
                    listBox_Information.Items.Add("Должность:
" + strMass[2] + ";");
                    listBox_Information.Items.Add("Цех:   "   +
strMass[3] + ";");

listBox_Information.Items.Add("Выполнимость:   "   +   strMass[4]   +
";");
                }
            }
            strRead.Close();
        }
        private  void  Worker_Form_FormClosing(object  sender,
FormClosingEventArgs e)
        {
```

```
                Application.Exit();
            }
        }
    }
```

DataBase_Worker.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WorkDistributor
{
    internal class DataBase_Worker
    {
        private string personnel_Numb; //табельный номер
        private string fio; //ФИО
        private string post; //должность
        private string ceh; //цех
        private string diligence; //исполнительность
        private string workload; //загруженность
        private string vacation_Numb; //номер отпуска

        public DataBase_Worker()
        {
        }

        public DataBase_Worker(string personnel_Numb, string
fio, string post, string ceh, string diligence, string workload,
string vacation_Numb)
        {
            this.personnel_Numb = Personnel_Numb;
            this.fio = Fio;
            this.post = Post;
            this.ceh = Ceh;
            this.diligence = Diligence;
            this.workload = Workload;
            this.vacation_Numb = Vacation_Numb;
        }

        public string Personnel_Numb
        {
            get
            {
                return personnel_Numb;
            }
            set
            {
                personnel_Numb = value;
            }
        }
        public string Fio
        {
```

```
        get
        {
            return fio;
        }
        set
        {
            fio = value;
        }
    }

    public string Post
    {
        get
        {
            return post;
        }
        set
        {
            post = value;
        }
    }

    public string Ceh
    {
        get
        {
            return ceh;
        }
        set
        {
            ceh = value;
        }
    }
    public string Diligence
    {
        get
        {
            return diligence;
        }
        set
        {
            diligence = value;
        }
    }

    public string Workload
    {
        get
        {
            return workload;
        }
        set
        {
```

```
                workload = value;
            }
        }

        public string Vacation_Numb
        {
            get
            {
                return vacation_Numb;
            }
            set
            {
                vacation_Numb = value;
            }
        }
    }
}
```

NameWorkerClass.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WorkDistributor
{
    static class TabellNumbWorkerClass
    {
        public static string TabellNumbWorker;
        public static string NameWorker;
        public static string Work;
    }
}
```

DataBase_Work.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WorkDistributor
{
    internal class DataBase_Work
    {

        private string number_work; //номер
        private string name_work; //наименование работы
        private int difficult; //сложность
        private string time; //время на исполнение
        private string comment; //описание

        public DataBase_Work()
        {
        }

        public   DataBase_Work(string   number_work,   string
name_work, int difficult, string time, string comment)
        {
            this.number_work = Number_work;
            this.name_work = Name_work;
            this.difficult = Difficult;
            this.time = Time;
            this.comment = Comment;
        }

        public string Number_work
        {
            get
            {
                return number_work;
            }
            set
            {
                number_work = value;
            }
        }
        public string Name_work
        {
            get
            {
                return name_work;
            }
            set
            {
                name_work = value;
            }
```

```
        }

        public int Difficult
        {
            get
            {
                return difficult;
            }
            set
            {
                difficult = value;
            }
        }

        public string Time
        {
            get
            {
                return time;
            }
            set
            {
                time = value;
            }
        }
        public string Comment
        {
            get
            {
                return comment;
            }
            set
            {
                comment = value;
            }
        }
    }
}
```

Sort.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WorkDistributor
{
    internal class Sort
    {
        public int Personnel_Numb { get; set; }
        public string Fio { get; set; }
        public string Post { get; set; }
        public string Ceh { get; set; }
        public int Complete { get; set; }
        public Sort(int personell_numb, string fio, string
post, string ceh, int complete)
        {
            Personnel_Numb = personell_numb;
            Fio = fio;
            Post = post;
            Ceh = ceh;
            Complete = complete;
        }
        public Sort()
        {

        }
    }
}
```

DataBase_Work.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WorkDistributor
{
    internal class DataBase_Work
    {

        private string number_work; //номер
        private string name_work; //наименование работы
        private int difficult; //сложность
        private string time; //время на исполнение
        private string comment; //описание

        public DataBase_Work()
        {
        }

        public   DataBase_Work(string   number_work,   string
name_work, int difficult, string time, string comment)
        {
            this.number_work = Number_work;
            this.name_work = Name_work;
            this.difficult = Difficult;
            this.time = Time;
            this.comment = Comment;
        }

        public string Number_work
        {
            get
            {
                return number_work;
            }
            set
            {
                number_work = value;
            }
        }
        public string Name_work
        {
            get
            {
                return name_work;
            }
            set
            {
                name_work = value;
            }
```

```
        }

        public int Difficult
        {
            get
            {
                return difficult;
            }
            set
            {
                difficult = value;
            }
        }

        public string Time
        {
            get
            {
                return time;
            }
            set
            {
                time = value;
            }
        }
        public string Comment
        {
            get
            {
                return comment;
            }
            set
            {
                comment = value;
            }
        }
    }
}
```