Министерство образования и науки Российской Федерации
Филиал Федерального государственного автономного образовательного учреждения
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
в г. Нижневартовске

Кафедра «Информатика»

РАБОТА ПРОВЕРЕНА

РЕЦЕНЗЕНТ
Начальник цеха
_____/ С.Н.Находов/_____
«____»_____2018 г

ДОПУСТИТЬ К ЗАЩИТЕ

И.о.зав.кафедрой «Информатика»
к.ф.-м.н, доцент
_____/ А.В.Ялаев /_____
«____»_____2018 г.

Автоматизация системы заявок на плановые и внеплановые работы на месторождении

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-09.03.01.2018.361.ПЗ ВКР

Консультанты
Экономическая часть
к.э.н., доцент
_____/А.В.Прокопьев/___
«____»_____2018г.

Безопасность жизнедеятельности
к.ф.-м.н., доцент
_____/ А.В.Ялаев/_____
«___»_____2018 г.
_2018г.

Руководитель работы
к.т.н., доцент
_____/В.А.Парасич/
«____»_____2018 г.

Автор работы
обучающийся группы НвФл-526
_____/ Ю.Ю.Юрченко/
«___ »_____

Нормоконтролер
старший преподаватель
_____/Л.Н.Буйлушкина/
«____»_____
2018г.

Нижневартовск 2018

АННОТАЦИЯ

Юрченко Ю.Ю. Автоматизация системы заявок на плановые и внеплановые работы на месторождении. – Нижневартовск: филиал ЮУрГУ, Информатика: 2018, 78с., 11 ил., 7 табл., библиогр. список – 20 наим., 2 прил.

Данная выпускная квалификационная работа представляет собой описание автоматизации системы заявок на плановые и внеплановые работы на месторождении.

Цель работы – повышение оперативности работы службы Контрольно Измерительных Приборов и Автоматизации и диспетчерской службы.

Для достижения поставленной цели выполнены следующие задачи:

Исследована предметная область по учёту и обработке заявок. Изучены процессы передачи заявок. Изучены требования заказчика. Выбран инструмент и средства для разработки. Разработан программный продукт по учёту и обработке заявок на месторождении.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Каждый день предприятия сталкиваются с компьютеризацией различных систем. Они во многом улучшают нашу жизнь, освобождают время, упрощают нашу работу, и облегчают наш труд. Самым важным ресурсом в наше время является информация. Работа с потоками не структурированной информации ведёт к увеличению затрат времени и ресурсов на обработку этой информации. Поэтому рациональное хранение, преумножение и использование информации является особенно актуальной темой.

Целью выпускной квалификационной работы является разработка приложения для автоматизации системы по учёту и обработке заявок на плановые и внеплановые работы на месторождении.

Задачи:

1. анализ предметной области;
2. определение функциональных требований;
3. анализ существующих разработок;
4. выбор средства реализации;
5. разработка баз данных (далее – БД);
6. организация условий труда;
7. оценка экономической эффективности.

Объектом выпускной квалификационной работы является процесс автоматизации обработки и учёта заявок на нефтяном предприятии .

Разработанное приложение позволяет оперативно обрабатывать заявки на плановые и внеплановые работы на месторождении.

1 АНАЛИТИЧЕСКИЙ РАЗДЕЛ

1.1 Описание предметной области

В целях проведения оперативного планирования, регулирования хода производства и систематического контроля за выполнением графиков создаются органы оперативного регулирования производства (диспетчерские службы). Диспетчеризация производства – действенное средство оперативного контроля за ходом производственного процесса во всех подразделениях предприятия.

Диспетчерская служба предназначена для оперативного управления. Основные задачи диспетчерской службы следующие:

– контроль, анализ и оперативное управление режимами работы;

– ликвидация нарушений режима работы и их последствий;

– обеспечение заданного объема закачки газа в период закачки и заданного дебита в режиме отбора газа.

– обеспечение выполнения на каждом участке суточных (сменных) заданий и подготовка производства к выполнению плана на следующие сутки.

Диспетчерская служба тесно связана с цехом по обслуживанию контрольно-измерительных приборов и автоматизации (далее – КИП и А). Связь осуществляется посредством передачи и обработки заявок на плановые и внеплановые работы (мероприятия) по обслуживанию нефтегазовых месторождений. Передача заявок осуществляется посредством заполнения файла MS Excel находящегося на общем сетевом диске. Передачу осуществляет дежурный диспетчер, после чего обслуживающий персонал КИП и А может лично ознакомиться с планом выполнения работ по месторождению. Так же, обслуживающий персонал может внести заявку для необходимых проведений работ другими цехами в целях ликвидации неполадок препятствующих работе КИП и А или же препятствующих обслуживающему персоналу для выполнения поставленных задач.

1.2 Функциональные задачи будущих пользователей

Администратор:

- создание новых пользователей;
- изменение структуры ПО;
- редактирование БД.

Диспетчер:

- добавление заявок;
- подтверждение заявок;
- удаление заявок;
- подтверждение выполнения заявок.

Обслуживающий персонал:

- передача заявок;
- подтверждение выполнения заявок.

1.3 Анализ аналогов и прототипов

В ходе работы изучены следующие аналоги и прототипы программных продуктов для учёта и обработки заявок:

- «Кларис»
- «ITSM 365»
- «Okdesk»

Программный продукт «Кларис» позволяет быстро и с минимальными затратами организовать работу с заявками на обслуживание и техподдержку.

Возможности программы:

- единую точку обращения за помощью к нашим специалистам. интуитивный и удобный для пользователей интерфейс даёт возможность делать запросы в службу поддержки быстро и легко;

– круглосуточный доступ к системе. вход возможен с любого аппарата, подключенного к интернету. вы не привязаны больше к офису, главное иметь компьютер и доступ в Интернет;

– стандартный способ оформления запросов и передачи заданий службе поддержки;

– доступ к данным ограничивается ролью пользователей в системе, каждый из которых видит только то, что необходимо для их работы;

– контроль над выполнением работ, учет затраченных ресурсов и времени;

– отправка задействованным сотрудникам уведомлений по e-mail и sms;

– сохранение всех предыдущих заявок в информационной базе, позволяет быстро разрешить две схожие проблемы;

– отчетность по затратам средств и времени на исполнение заявок.

Отклонено по причине:

– сложность в освоении;

– коммерческий программный продукт 36000 рублей.

«ITSM 365» – сервис для внутренней и внешней поддержки. Представляет собой платформу Naumen Service Desk, преднастроенную для малого / среднего бизнеса. Включает в себя Service Desk, портал самообслуживания, личные кабинеты бизнес-пользователей, каталог услуг (внешних или внутренних), базу знаний, каталог оборудования, ПО и ИТ услуг, внутренние задачи, а также инструменты для управления изменениями, проблемами и конфигурациями и модуль отчетности.

Возможности программы:

– управление службой service desk, включая управление заявками, инцидентами, событиями;

– управление проблемами и изменениями;

– управление корпоративными и порученными задачами;

– управление доступом к данным;

– управление конфигурациями;
– управление сервисами (slm) и уровнями предложения сервисов (sla);
– управление портфелем услуг;
– управление базой знаний;
– ведение учёта трудозатрат ит-специалистов;
– удобные дешборды;
– учет ит-активов;
– портал самообслуживания;
– отчёты и аналитика;
– уведомления о событиях;
– интеграция с ldap-каталогами и ms active directory.

Отклонено по причине:

– сложность в освоении;
– платный программный продукт.

«Okdesk» – облачное решение для автоматизации процессов поддержки и взаимодействия с юридическими лицами в малых и средних сервисных компаниях. Обладает функциональностью Help Desk + CRM (учет и обработка заявок, учет договоров, клиентов и контактных лиц с индивидуальными условиями обслуживания, история взаимодействия). Бесплатный встроенный клиентский портал. Регистрация обращений по почте и с сайта. Переписка с клиентом. Функциональность отчетов и дашбордов для руководителей.

Возможности программы:

– регистрация заявок через email, по телефону, с настраиваемой веб-формой на сайте и из клиентского портала;
– управление заявками и распределение ответственности «на лету»;
– мобильное приложение с offline режимом;
– база знаний;
– настройка бизнес процессов службы поддержки;
– предпросмотр файлов;

– функциональный клиентский портал для компаний (юр. лица) и контактных лиц – регистрация и контроль выполнения заявок;

– ведение договоров и индивидуальных sla для клиента;

– учет сервисных периодов в рамках абонентского обслуживания

– стилизация интерфейса и оповещений, парковка домена;

– настраиваемые оповещения (email и sms);

– история взаимодействия с клиентом;

– модуль учета трудозатрат;

– модуль «прайс лист», позволяет учитывать стоимость разовых работ и услуг компании;

– интеграция с более чем 20 атс

– api.

Отклонено по причине:

– сложность в освоении;

– коммерческий программный продукт 15000 рублей.

1.4 Постановка задачи проектирования

Целью является разработка программного продукта для реализации передачи достоверной информации, исключения дублирования необходимых мероприятий, упрощения производственного процесса.

Исходя из поставленной цели, были сформулированы следующие задачи:

– провести сбор информации об объекте исследования;

– провести анализ предметной области;

– рассмотреть функциональные задачи будущих пользователей;

– изучить литературу по данной теме;

– выполнить технико-экономическое обоснование;

– разработать программный продукт;

– протестировать программный продукт;

– внедрить программный продукт.

На основе анализа предметной области к программному продукту были предъявлены следующие требования:

– простой и удобный пользовательский интерфейс;
– ввод и хранение данных;
– редактирование, изменение, удаление существующих данных отчетов.

Выводы по разделу один:

В данном разделе проанализирована предметная область, выделены основные проблемы и задачи, которые предлагается решить с помощью разработанного приложения. Определены требования, как к самому приложению, так и к программно-техническим средствам для корректной работы приложения.

2 ПРОЕКТНЫЙ РАЗДЕЛ

2.1 Выбор средств разработки

Так как программное обеспечение (далее – ПО) предназначено для работы в среде Windows, и должно иметь интерфейс пользователя, работающего в домене с каталогом Active Directory, из популярных языков программирования наиболее подходящим является язык программирования C# входящий в инструмент разработки Visual Studio 2015 Express, являющийся бесплатным продуктом, позволяющий реализовывать и коммерческие проекты. Для работы с Active Directory в Visual Studio 2015 Express имеется встроенная стандартная библиотека «System.DirectoryServices», которая включает в себя класс «DirectoryEntry», позволяющий достаточно просто получить информацию о любой учетной записи, хранящейся в каталоге Active Directory, и осуществить необходимые операции с ней.

Еще до написания проекта встал выбор между языками программирования, были рассмотрены: Borland Delphi, C++, и на то время сравнительно молодой C#. После поиска информации и ее оценки, появились следующие заключения: Borland Delhi потерял свое развитие, поддержку, поддерживаются лишь старые, ранее написанные проекты, подавляющее большинство разработчиков для разработки приложений на платформе .Net для реализации новых проектов используют либо язык программирования C++ либо C#. Оценивая плюсы и минусы данных языков, на C# легче стартовать, интуитивнее и понятнее код, отличная поддержка языка и среды разработки Visual Studio, возможность написания Web и мобильных приложений. Для малого и среднего уровня разработки не уступает в производительности C++, реализация проектов занимает меньше времени. C++ более подходит для масштабных, крупных приложений, требующих большой производительности.

Характеристики данного продукта.

Visual Studio – это интегрированная среда разработки ПО от компании Microsoft. С помощью Visual Studio можно создавать приложения для Windows, iOS, Android и других платформ. В Visual Studio включены инструменты не только для создания desktop приложений, но и web, мобильные и облачные инструменты разработки. Имеется возможность написания кода на таких языках как: C++, C#, Visual Basic, F#, JavaScript, Python, TypeScript. Помимо всего этого она включает в себя конструкторы, редакторы, отладчики, профилировщики, а также огромное количество расширений для разных областей применения – от PHP до игр.

20 июля 2015 года компания Microsoft объявила о выходе новой версии Visual Studio, а именно Visual Studio 2015, она включает в себя следующие новые возможности и обновления:

- выпускается в трех редакциях (ранее было четыре);
- кросс-платформенная поддержка мобильных устройств (Android, IOS, и Windows);
- улучшения в C ++;
- изменения в отладке и диагностике;
- .NET Framework 4.6;
- улучшена интеграция Visual Studio и GitHub.

Системные требования для установки Visual Studio 2015:

Visual Studio 2015 поддерживает следующие операционные системы:

- Windows 7 с пакетом обновления 1;
- Windows 8;
- Windows 8.1;
- Windows Server 2008 R2 с пакетом обновления 1 (SP1);
- Windows Server 2012;
- Windows Server 2012 R2;
- Windows 10 (после выхода окончательной версии).

Также есть требования к оборудованию:

– процессор с частотой 1.6 ГГц (или выше);

– 1 Гб оперативной памяти (1,5 ГБ при работе на виртуальной машине);

– 10 Гб свободного пространства на жестком диске;

– жесткий диск со скоростью вращения шпинделя 5400 оборотов в минуту и выше;

– видеокарта с поддержкой DirectX 9 и разрешения дисплея 1024x768 (или более высокого).

2.2 Проектирование БД

На основе предметной области и структуры приложения, приступаем к проектированию БД разрабатываемого программного продукта. Методом проектирования БД выбраны ER-диаграммы. Данный метод является одним из наиболее понятных и практически используемых методов проектирования реляционных БД, в основу которых положена модель «сущность-связь». Ключевыми элементами модели являются сущности, их свойства (атрибуты) и связи между ними.

Сущности с указанием ключевого атрибута:

1. Куст (id_kust).
2. Скважина (id_hole).
3. Подразделения (id_subjects).
4. Пользователи (id_users).

Имеются сущности, образованные в результате связывания нескольких сущностей из списка выше:

Не ключевые атрибуты в сущности:

1. Причина простоя (reason_stop).
2. Дата/время (date_create).
3. Выполнение (is_done).
4. Ф.И.О. (FIO).
5. Логин (login).

6. Пароль (password).

На рисунке 2.1 представлена логическая модель спроектированной БД.

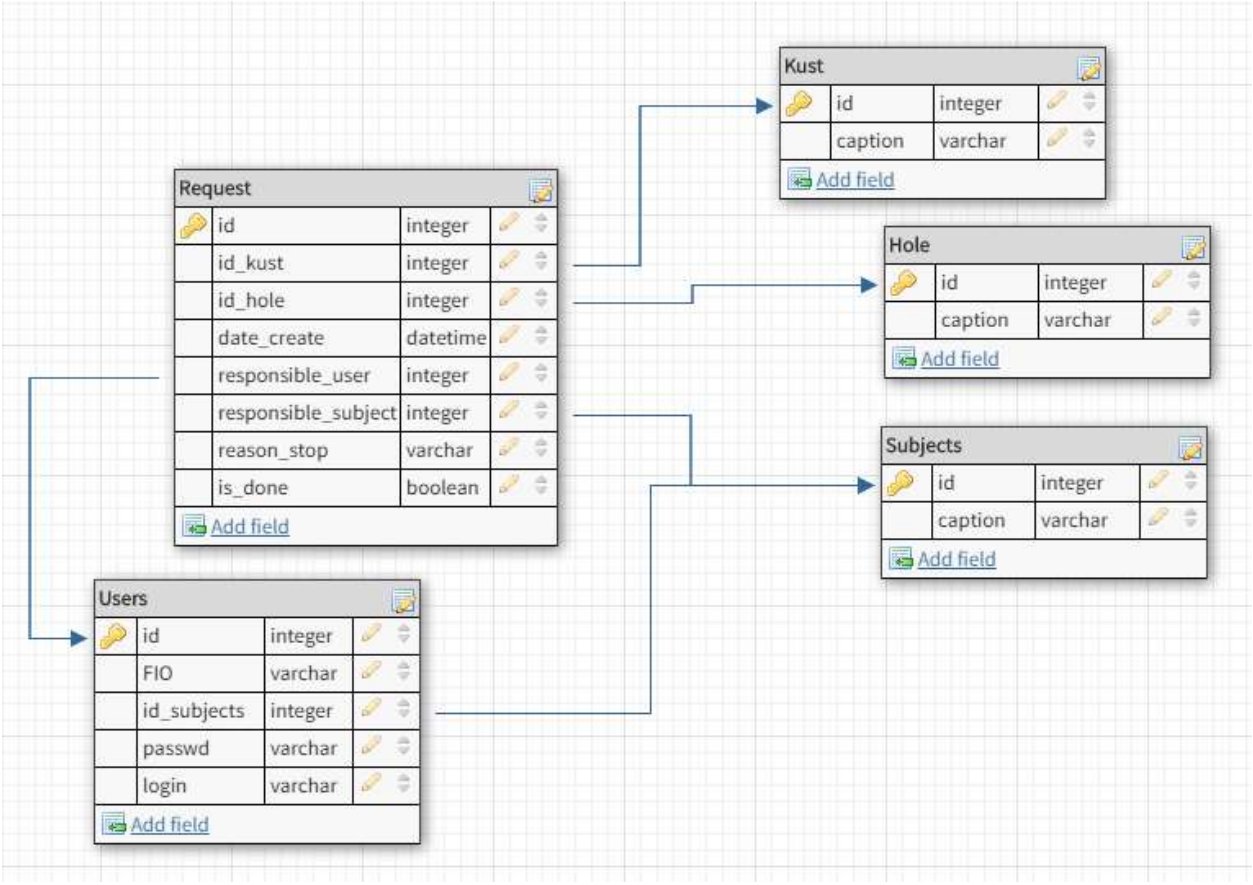

Рисунок 2.1 – Сущности и связи между таблицами БД

2.2.1 Система управления базами данных

Система управления базами данных (далее – СУБД) представляет собой ПО, которое управляет всем доступом к базе данных.

Для разрабатываемой системой надежность и производительность СУБД является критическим моментом, поскольку приложение предполагает наличие большого количества записей в БД, а также большое количество пользователей, чьи действия так или иначе будут связаны с манипулированием информации из БД.

Основные требования к СУБД:

– управление данными во внешней памяти (на дисках);

– управление данными в оперативной памяти с использованием дискового кэша;

– журнал изменений, резервное копирование и восстановление БД после сбоев;

– поддержка языков БД (язык определения данных, язык манипулирования данными);

– поддержка реляционной модели хранения данных;

– поддержка распределенного режима;

– поддержка клиент-серверной архитектуры взаимодействия;

– наличие внешних утилит для управления СУБД;

– низкая стоимость.

2.2.2 Обзор современных СУБД

В мире существует множество СУБД, обзор их всех не входит в рамки настоящей выпускной квалификационной работы. В данном раздел рассмотрены наиболее популярные СУБД, которые имеют многолетнюю историю разработки и поддержки.

Oracle Database 12c Standard Edition – полнофункциональная серверная база для управления данными в сфере среднего бизнеса. Данная редакция подходит для всех типов данных и приложений, совместима с ОС Windows, Linux и Unix. Решение Oracle Database отличают доступность и гибкость, простота в установке и конфигурировании. Oracle Database 12c Standard Edition можно приобрести в минимальной конфигурации, а по мере расширения задач организации масштабировать её с помощью кластеров Oracle Real Application Clusters. При масштабировании IT-инфраструктуры организации Oracle Database 12c Standard Edition легко расширяется до версии Enterprise.

Стоимость решения – от 20 000 рублей.

PostgreSQL – свободная объектно-реляционная СУБД.

У PostgreSQL множество возможностей. Созданный с использованием объектно-реляционной модели, он поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных. Он обеспечивает расширенную ёмкость данных, а так же особое внимание уделяет к целостности данных.

Свободно распространяемое ПО.

MySQL – свободная СУБД. MySQL является собственностью компании Sun Microsystems, осуществляющей разработку и поддержку приложения. Распространяется под GNU General Public License и под собственной коммерческой лицензией, на выбор. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей, именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

Стоимость простой лицензии MySQL Enterprise на год – 18 000 рублей.

Свободно распространяемое ПО.

Выбор СУБД в соответствии с предъявленными требованиями

Решение Oracle является неприемлемым с точки зрения стоимости лицензии, а также необходимой аппаратной части для работы этой СУБД.

СУБД PostgreSQL по сравнению с MySQL похожи по функциональности, однако при простых операциях чтения PostgreSQL может значительно замедлить сервер и быть медленнее своих конкурентов, таких как MySQL.

MySQL полностью совместима с выбранной ОС Ubuntu Server 16.04. Поддерживается ветвь MySQL 5.7.x, которая является наиболее стабильной на данный момент.

MySQL имеет двойное лицензирование. MySQL может распространяться в соответствии с условиями лицензии GPL. Однако по условиям GPL, если какая-либо программа включает исходные коды MySQL, то она тоже должна распространяться по лицензии GPL. Это может расходиться с планами разработчиков, не желающих открывать исходные тексты своих программ. Для таких случаев предусмотрена коммерческая лицензия, которая также обеспечивает качественную сервисную поддержку.

Среда SQL Server Management Studio 17.X (далее – SSMS) основана на изолированной оболочке Visual Studio 2015, которая была выпущена до Windows Server 2016. Корпорация Майкрософт уделяет большое внимание совместимости приложений и гарантирует, что уже выпущенные приложения продолжат работать в последних выпусках Windows. Чтобы минимизировать проблемы с запуском SSMS в Windows Server 2016, убедитесь, что для SSMS установлены все последние обновления. При возникновении каких-либо проблем с SSMS в Windows Server 2016, обратитесь в службу поддержки. Служба поддержки определит, связана ли проблема с SSMS, с Visual Studio или с совместимостью SSMS и Windows. Затем ваш запрос будет перенаправлен соответствующей группе для дальнейшего изучения.

Свободно распространяемое ПО.

Выбор СУБД в соответствии с предъявленными требованиями:

Решение Oracle является неприемлемым с точки зрения стоимости лицензии, а также необходимой аппаратной части для работы этой СУБД.

СУБД PostgreSQL по сравнению с MySQL похожи по функциональности, однако при простых операциях чтения PostgreSQL может значительно замедлить сервер и быть медленнее своих конкурентов, таких как MySQL.

MySQL имеет двойное лицензирование. MySQL может распространяться в соответствии с условиями лицензии GPL. Однако по условиям GPL, если какая-либо программа включает исходные коды MySQL, то она тоже должна распространяться по лицензии GPL. Это может расходиться с планами разработчиков, не желающих открывать исходные тексты своих программ. Для таких случаев предусмотрена коммерческая лицензия, которая также обеспечивает качественную сервисную поддержку.

Это означает, что в рамках разрабатываемого приложения, возможно, использовать некоммерческую версию SSMS.

2.3 Руководство пользователя

2.3.1 Функции и интерфейс взаимодействия программы с пользователем.

При запуске программы открывается окно авторизации (рисунок 2.1), которое позволяет получить доступ к функциям программы для обработки заявок после ввода логина и пароля. Для успешной авторизации необходимо иметь логин и пароль пользователя, который присваивается администратором. Пользователю необходимо ввести их в соответствующие поля и затем нажать кнопку «Войти». При введении не правильных данных пользователь может повторить попытку или же выйти из программы.

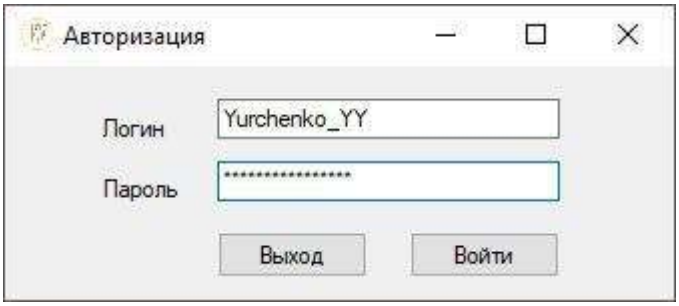

Рисунок 2.1 – Окно авторизации программы

Главное окно программы (рисунок 2.2) представляет собой информационную панель на которой располагается вся информация необходимая для работы с заявками на нефтяном месторождении, а именно:

1. Панель инструментов, с помощью которой осуществляется доступ к главному окну и к справочникам.
2. Фильтры для поиска необходимой информации по таблице заявок.
3. Элемент управления для добавления заявок.
4. Элемент управления для редактирования заявок.
5. Элемент управления для удаления заявок.
6. Элемент управления скрывающий выполненные заявки.

Для того чтобы создать заявку пользователю необходимо заполнить данные таблицы соответственно названиям столбцов и нажать кнопку добавления, находящуюся в правом нижнем углу, которая сохранит данные. Выделив строку и затем нажав кнопку редактирования, находящуюся в одном ряду с кнопкой добавления и удаления, пользователь имеет возможность изменения текущих данных по заявкам. Для удаления заявки пользователю необходимо выделить необходимую строку и нажать кнопку удаления.

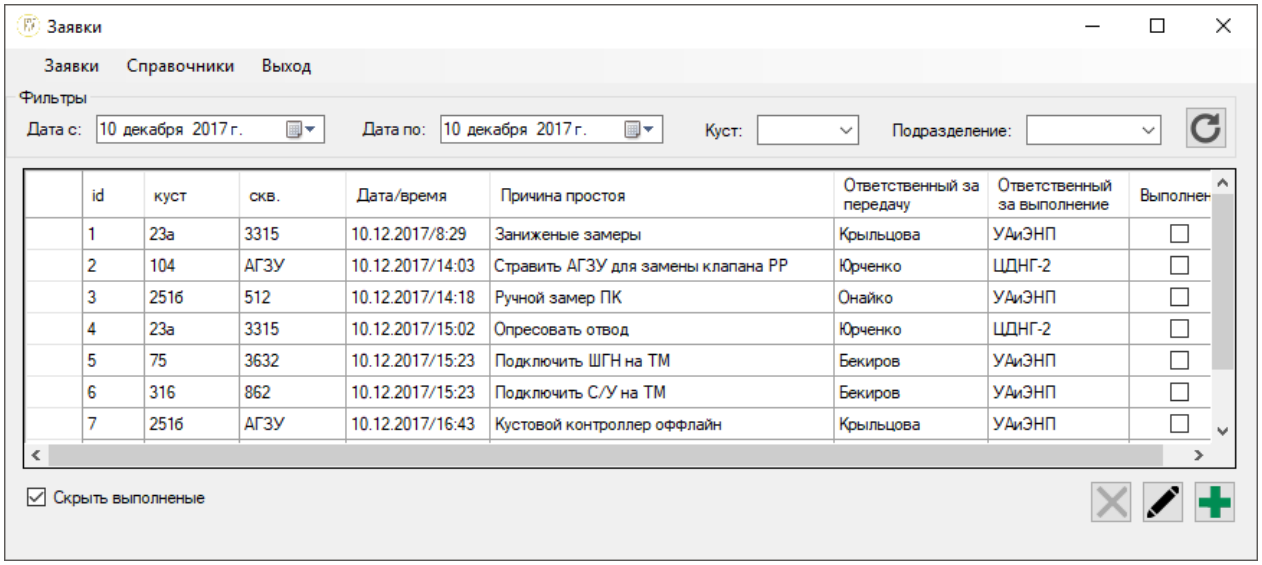

Рисунок 2.2 – Главное окно программы

При использовании вкладки «справочники» пользователь получает доступ к меню с помощью которого может редактировать необходимую информацию в базах данных. При её нажатии появляются такие пункты как «Куст», «Скважины», «Подразделения» и «Пользователи».

При переходе на вкладку «Куст» (рисунок 2.3) справочников открывается окно, которое содержит инструменты для внесения кустов находящихся в работе на месторождении. Для того чтобы внести кусты, пользователю необходимо записать их в таблицу, и нажать кнопку добавления, находящуюся в правом нижнем углу окна на ряду с кнопками редактирования и удаления. Для того чтобы изменить или удалить номер куста необходимо выделить строку содержащую куст и нажать кнопку редактирования или удаления в правом нижнем углу окна.

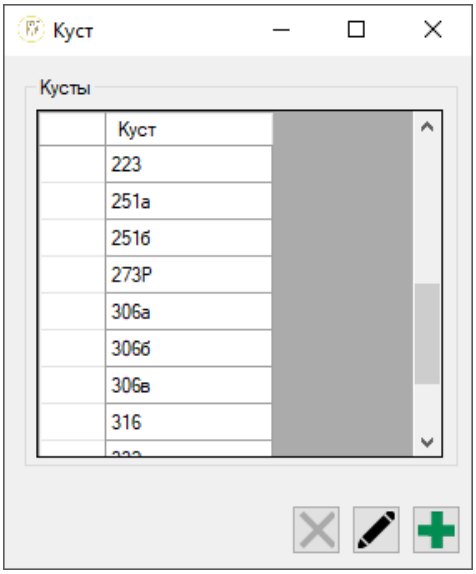

Рисунок 2.3 – Окно «Куст»

При переходе на вкладку «Скважины» (рисунок 2.4) справочников открывается окно, которое содержит инструменты для внесения скважин находящихся на определённых кустах. Для того чтобы внести скважины, пользователю необходимо записать их в таблицу, и нажать кнопку добавления, находящуюся в правом нижнем углу окна на ряду с кнопками редактирования и удаления. Для того чтобы изменить или удалить номер скважины необходимо выделить строку содержащую необходимую скважину и нажать кнопку редактирования или удаления в правом нижнем углу окна.

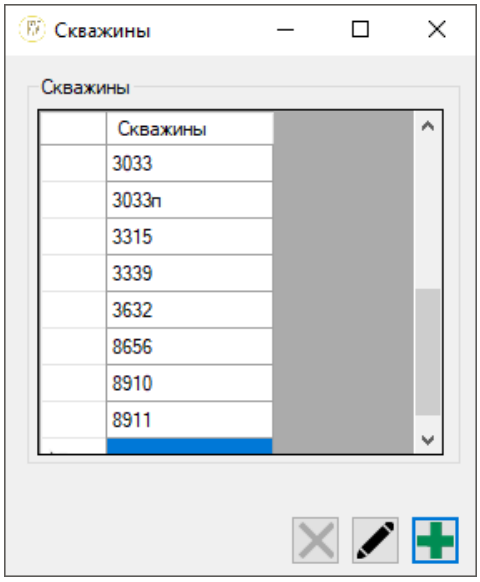

Рисунок 2.4 – Окно «Скважины»

При переходе на вкладку «Подразделения» (рисунок 2.5) справочников открывается окно, которое содержит инструменты для внесения подразделений выполняющих работы на месторождении. Для того чтобы внести подпазделения, пользователю необходимо записать их в таблицу, и нажать кнопку добавления, находящуюся в правом нижнем углу окна на ряду с кнопками редактирования и удаления. Для того чтобы изменить или удалить подразделение необходимо выделить строку содержащую необходимое подразделение и нажать кнопку редактирования или удаления в правом нижнем углу окна.

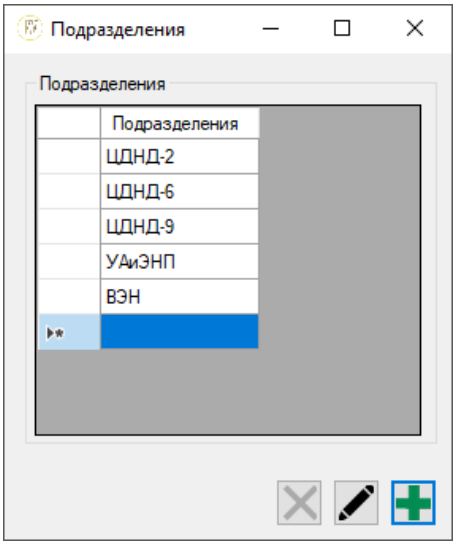

Рисунок 2.5 – Окно «Подразделения»

При переходе на вкладку «Пользователи» (рисунок 2.6) справочников открывается окно, которое содержит инструменты для регистрации пользователей имеющих доступ к программе. Для того чтобы добавить, удалить или редактировать информацию о новых или имеющихся пользователях, необходимо нажать соответствующую кнопку в нижнем правом углу окна «Пользователи».

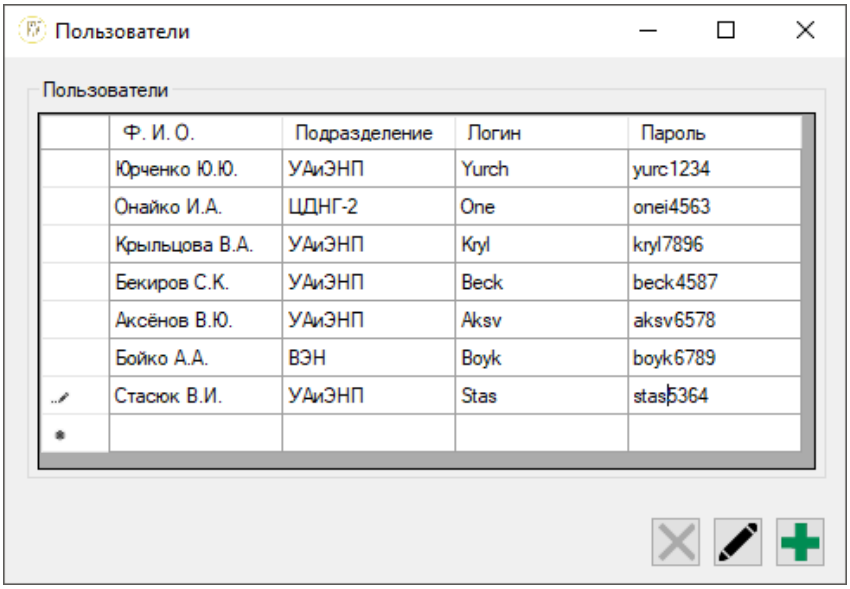

Рисунок 2.6 – Окно «Пользователи»

При создании пользователей открывается окно «Редактирование пользователей» (рисунок 2.7) для внесения необходимой информации о персонале имеющем доступ к программе и, соответственно, обработке заявок на месторождении. Для добавления необходимо заполнить личные данные находящиеся в окне «Редактирование пользователей» а так же присвоить оригинальные данные авторизации как логин и пароль, каждый из которых состоит из 16 символов. После внесения всех данных необходимо нажать кнопку добавления/обновления данных, для последующего создания пользователя.

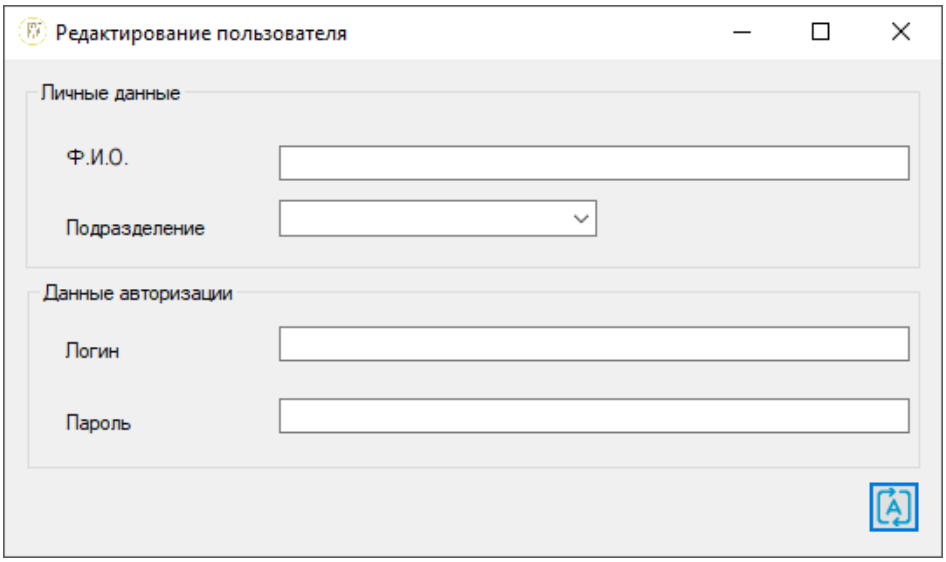

Рисунок 2.7 – Окно «Редактирование пользователей»

2.3.2 Разграничение пользователей

Права доступа — совокупность правил, регламентирующих порядок и условия доступа субъекта к объектам информационной системы (информации, её носителям, процессам и другим ресурсам) установленных правовыми документами или собственником, владельцем информации.

Права доступа определяют набор действий (например, чтение, запись, выполнение), разрешённых для выполнения субъектам (например, пользователям системы) над объектами данных. Для этого требуется некая система для предоставления субъектам различных прав доступа к объектам. Это система разграничения доступа субъектов к объектам, которая рассматривается в качестве главного средства защиты от несанкционированного доступа к информации или порче самой системы.

Каждому пользователю определена роль. Это позволяет создать границы между пользователями, за счёт которых им будут отведены выполнение определённых им задач. Роль пользователя определяется при регистрации пользователя администратором. В качестве примера можно привести заполнение информации БД и заполнение окна заявок.

В ходе разработки выделены три вида пользователей с отведёнными для них возможностями:

- администратор;
- диспетчер;
- обслуживающий персонал.

Решено выделить такие функции системы разграничения доступа, как

- реализация правил разграничения доступа (ПРД) субъектов и их процессов к данным;
- изоляция программ процесса, выполняемого в интересах субъекта, от других субъектов;
- реализация правил обмена данными между субъектами.

– идентификацию и опознание (аутентификацию) субъектов и поддержание привязки субъекта к процессу, выполняемому для субъекта;

– регистрацию действий субъекта и его процесса;

– предоставление возможностей исключения и включения новых субъектов и объектов доступа, а также изменение полномочий субъектов;

– контроль целостности программной и информационной части.

При авторизации пользователя с правами доступа администратора, ему доступны такие возможности как заполнение справочников. При заполнении справочников администратор заполняет БД содержащие информацию о кустах, скважинах, подразделениях и пользователях. При заполнении БД пользователей администратор вносит информацию о будущих пользователях и определяет права доступа пользователей. Функции администратора изображены на следующей диаграмме (рисунок 2.8).

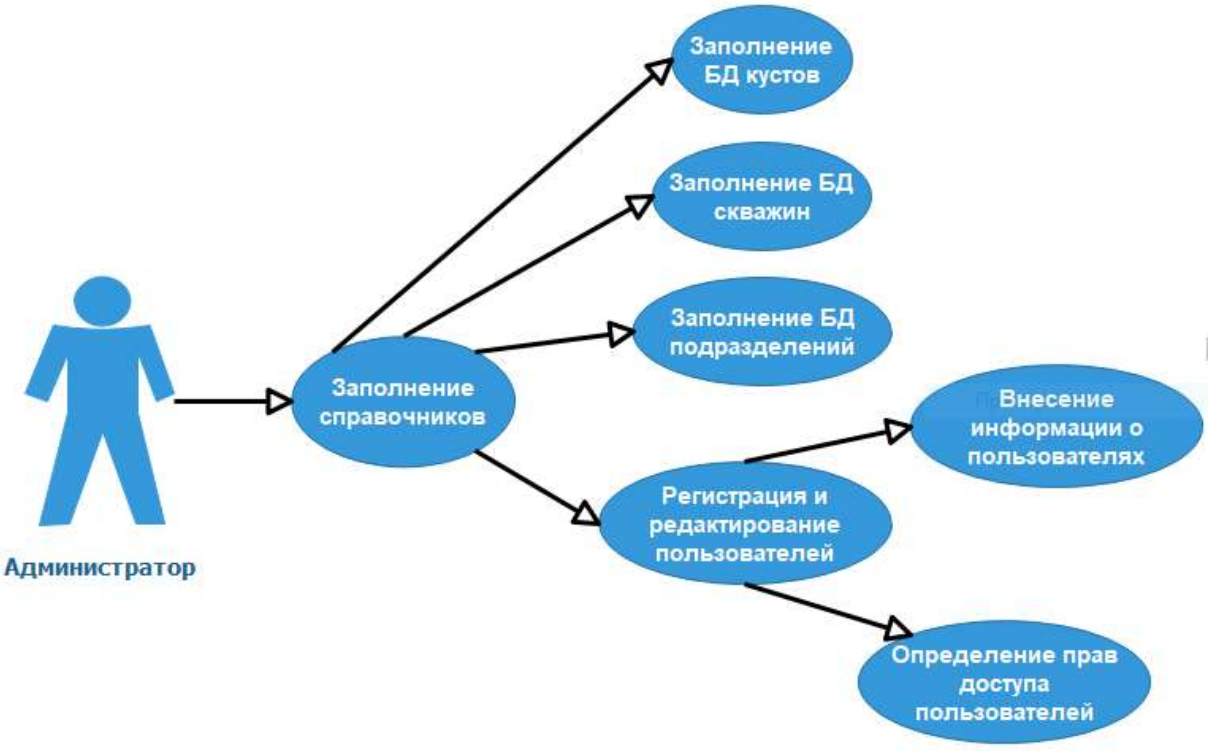

Рисунок 2.8 – ER-диаграмма «администратор»

При авторизации пользователя с правами доступа диспетчера, ему доступны такие возможности как редактирование заявок. При заполнении таблицы заявок диспетчеру доступны функции добавления, удаления и подтверждения заявок предложенных на добавление обслуживающим персоналом. Функции диспетчера изображены на следующей диаграмме (рисунок 2.9).

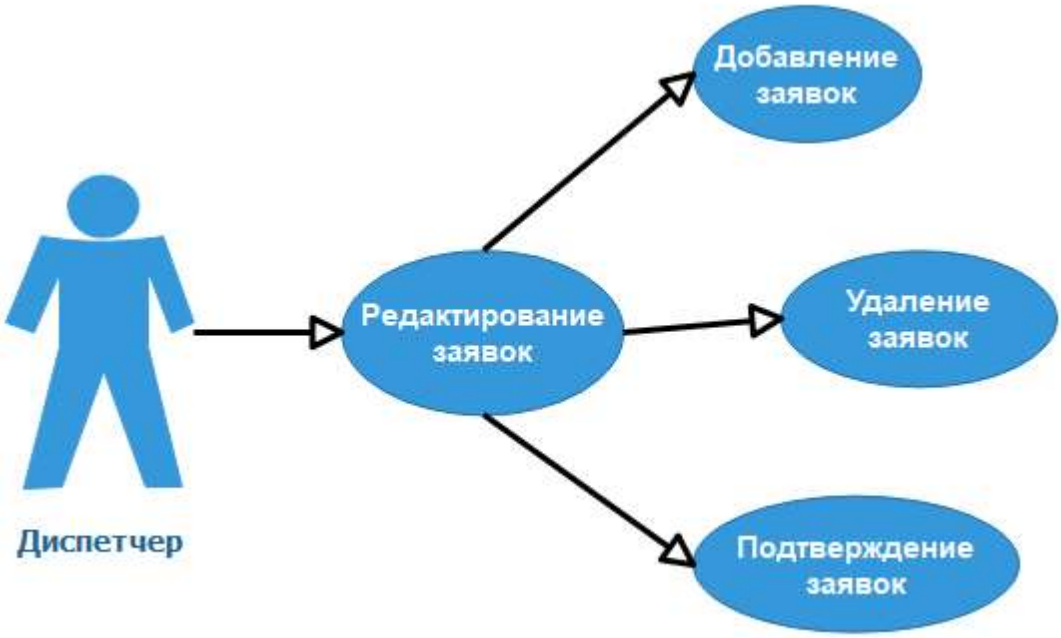

Рисунок 2.9 – ER-диаграмма «диспетчер»

При авторизации пользователя с правами доступа обслуживающего персонала, ему доступны такие возможности как добавление заявок. При создании заявки пользователь вносит её в таблицу, где в следствии диспетчер должен её подтвердить. Функции обслуживающего персонала изображены на следующей диаграмме (рисунок 2.10).

Рисунок 2.10 – ER-диаграмма «обслуживающий персонал»

Выводы по разделу два:

В ходе написания проектного раздела выпускной квалифицированной работы были выполнены следующие задачи:

1. Произведён выбор средств разработки.
2. Спроектированы БД.
3. Разработано руководство пользователя.

3 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

Целью выпускной квалификационной работы является разработка программного продукта предоставляющего информационную систему для решения задач связанных с работоспособностью приборов на нефтегазовом месторождении. Снижение трудовых затрат позволит уменьшить и финансовые затраты, что приведет к общему увеличению производительности и экономии.

Рассчитаем экономический эффект от использования приложения организацией.

Основной задачей является определение величины затрат на проведение исследований, себестоимость для определения экономического эффекта от использования в общественном производстве основных и сопутствующих результатов, получаемых при решении поставленной технической задачи в выпускной квалификационной работе. Оценка эффективности принятого научно-технического решения должна учитывать все необходимые расходы и затраты, для этого потребуется провести ряд необходимых расчетов.

3.1 Расходы на приобретение, содержание и эксплуатацию программного и аппаратного обеспечения

Стоимость затрат определяется из списков цен производителя, и представлена в таблице 3.1.

Таблица 3.1 – Стоимость ПО и аппаратных средств

Наименование	Цена в руб.
Visual Studio 2015 Express	30058
Ноутбук MSI CX70 QD	19999
Microsoft Windows Server 2014 R2	46845
Microsoft Office 2010 x32	16890
Итого:	113792

Таблица 3.2 – Материалы

Наименование	Кол-во	Цена, руб.
Kingston DataTraveler SE9 16GB16Gb (DTSE9H)	1	380
Диск DVD+R TDK 4.7Gb 16x Slim	1	46
Итого		426

Затраты на электроэнергию находятся исходя из продолжительности периода разработки ПО, количества кВт/ч, затраченных на проектирование ПО и тарифа за 1 кВт/ч. Тариф по городу Радужному для юридических лиц составляет 4.60 руб. за кВт/ч. Затраты отражены в таблице 3.3.

Таблица 3.3 – Затраты на электроэнергию

Элемент системы	Установленная мощность, кВт	Стоимость 1кВт в час (руб.)	Кол-во часов работы	Общая стоимость, руб.
Ноутбук MSI CX70 QD	0,057	4,60	348	91,24
Итого				91,24

Затраты на амортизацию оборудования проводятся за период их использования, т.е. за период внедрения и создания дополнений к программному обеспечению.

Денежное выражение амортизации является амортизационным отчислением, которое входит в текущие затраты.

Величина амортизационных отчислений определяется на основе норм амортизации.

Норма амортизации – это установленный размер амортизационных отчислений на полное восстановление, выраженное в %. Норма амортизации устанавливается на основе экономически целесообразного срока службы и должна обеспечить возмещение износа основных средств к моменту возможного их морального и физического износа и создать экономическую основу для замены.

Амортизационные отчисления, приходящиеся на 1 час работы системы, рассчитываются по формуле (1).

$$Ач = Ф_{перв} \cdot \frac{a}{F_д}, \quad (1)$$

где $Ф_{перв}$ – первоначальная стоимость системы или отдельных элементов;

a – норма амортизации (0.2);

$F_д$– фонд времени работы за год (2500 часов).

Таблица 3.4 – Расчет амортизационных отчислений

п/п	Элемент КТС	*Ф перв*	*Fд*	*Ач*	Кол-во часов работы	Общая стоимость (руб.)
1	Ноутбук MSI CX70 QD	19 999	2500	1.5999	348	556.7
Итого						556.7

Просуммировав расчет, мы получили расходы на приобретение, содержание и эксплуатацию программного и аппаратного обеспечение равного 1270,97 рублей.

3.2 Затраты на заработную плату

Для расчета расходов на заработанную платы необходимо умножить среднюю часовую ставку программиста на трудоемкость работы, чел/час по каждому из этапов разработки системы.

Средняя часовая ставка находиться по формуле (2):

$$З_ч = \frac{З_м}{168}, \quad (2)$$

где $З_ч$– средняя часовая ставка программиста;

$З_м$ – средняя месячная ставка техника-программиста (12556 рублей).

$З_ч$ = 12556 /168 = 74 (руб.).

Исходя из полученных данных, можно рассчитать заработную плату по всем этапам разработки, результат в таблице 3.5.

Таблица 3.5 – Расчет основной и дополнительной заработной платы

Содержание работы	Трудоёмкость работы, чел/час	Основная заработная плата (руб.)
Анализ предметной области	10	592
Постановка задачи	10	592
Разработка технического задания	15	888
Разработка интерфейса программы	66	3907
Разработка программы	95	5624
Тестирование системы	137	8288
Документирование	15	888
Итого:	348	20779

3.3 Расчёт затрат на дополнительную заработную плату

Дополнительную заработную плату разработчиков определяют в процентах от итоговой суммы основной заработной платы (15 %).

ЗП доп.= 20779 · 0,15 = 3116 (руб.).

3.4 Отчисления на социальные нужды

Отчисления на социальные нужды рассчитывают в процентах от суммы основной и дополнительной заработных плат, в пенсионный фонд, в ФСС и мед. страхование. На 2018 год данный процент составляет 30%, рассчитывается по формуле (3).

$$ОСН = 30\% \cdot (ЗП_{осн} + ЗП_{доп}), \quad (3)$$

где *ОСН* – отчисления на социальные нужды;

$ЗП_{осн}$ – основная заработная плата;

$ЗП_{доп}$ – дополнительная заработная плата.

ОСН = 0,30 · (20779+3116) =7168 (руб.)

3.5 Общая смета затрат на внедрение приложения

Таблица 3.6 – Общая смета затрат

п/п	Элементы затрат	Сумма, руб.
1.	Затраты на основную заработную плату	20779
2.	Затраты на дополнительную заработную плату	3116
3.	Отчисления на социальные нужды	7168
4.	Материалы	426
5.	Амортизационные отчисления	556
ИТОГО		32045

3.6 Оценка экономической эффективности

На предприятии существует проблема передачи заявок связанная с достоверностью передаваемой информации и с своевременным документированием плана выполненных работ. На основе проблем было решено разработать программный продукт по учёту и обработке заявок цеха КИП и А.

Разрабатываемое ПО предназначено для эффективной работы диспетчерской службы и цеха КИП и А, из чего следует что дополнительных затрат не планируется.

Для нахождения средней стоимости часа $З_ч$, необходимо разделить среднюю заработную плату диспетчера за день (1200 рублей) на количество рабочих часов (8 часов).

$З_ч$ = 1200 /8 = 150 руб.

Для внесения заявки в список необходимо редактировать файл находящийся на общем сетевом диске, исходя из этого возможность редактирования доступна

лишь одному пользователю. Цеху КИП и А также, необходимо вносить информацию в список для подтверждения выполнения работ или же для внесения заявок связанными с другими отделами предприятия. Цех КИП и А имеет 3 участка которым доступна возможность редактирования списка заявок. Время внесения в данном случае составляет t_{n1} = 40 минут. С помощью разрабатываемого програмного продукта время сократится до t_{n2} = 10 минут.

$t_{n1} - t_{n2}$ = 40мин. – 10мин. = 30мин. в день.

За месяц экономия времени сотрудника составит: $T_{рм1}$ = 26 · 30мин. = 13 ч. Значит, за счет экономии времени, возможно увеличение часов рабочего времени в выполнении других задач, выигрыш в денежном эквиваленте составит: $P_{м1}$ =200 · 13 ч = 2600 руб. в месяц.

Выводы по разделу три:

Произведён анализ разработанного программного приложения с точки зрения актуальности и экономической выгодности его использования взамен используемых средств. Прежде всего, была определена трудоёмкость и сроки выполнения работ. Следующим шагом стал подсчёт затрат, необходимых для реализации разработанного. Завершающим этапом стал расчёт экономического эффекта от данного приложения.

Произведённые расчёты и показатели позволяют говорить о целесообразности и экономической эффективности с точки зрения временных затрат данного приложения для учёта данных промышленной экспертизы нефтепромыслового оборудования.

4 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

4.1 Охрана труда на рабочем месте программиста

Охрана труда – система законодательных актов, социально-экономических, организационных, технических, гигиенических и лечебно-профилактических мероприятий и средств, обеспечивающих безопасность, сохранение здоровья и работоспособности человека в процессе труда. Научно-технический прогресс внес серьезные изменения в условия производственной деятельности работников умственного труда. Их труд стал более интенсивным, напряженным, требующим значительных затрат умственной, эмоциональной и физической энергии. Это потребовало комплексного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

Охрана здоровья трудящихся, обеспечение безопасности условий труда, ликвидация профессиональных заболеваний и производственного травматизма составляет одну из главных забот человеческого общества. Обращается внимание на необходимость широкого применения прогрессивных форм научной организации труда, сведения к минимуму ручного, малоквалифицированного труда, создания обстановки, исключающей профессиональные заболевания и производственный травматизм.

Данный раздел выпускной квалификационной работы посвящен рассмотрению следующих вопросов:

- организация рабочего места программиста;
- определение оптимальных условий труда программиста.

4.2 Описание рабочего места программиста

Рабочее место – это часть пространства, в котором инженер осуществляет трудовую деятельность, и проводит большую часть рабочего времени. Рабочее место, хорошо приспособленное к трудовой деятельности инженера, правильно и це-

лесообразно организованное, в отношении пространства, формы, размера обеспечивает ему удобное положение при работе и высокую производительность труда при наименьшем физическом и психическом напряжении.

При правильной организации рабочего места производительность труда инженера возрастает с 8 до 20 процентов.

Согласно ГОСТ 12.2.032-78 конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия:

– оптимальное размещение оборудования, входящего в состав рабочего места;

– достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;

– необходимо естественное и искусственное освещение для выполнения поставленных задач;

– уровень акустического шума не должен превышать допустимого значения.

Главными элементами рабочего места программиста являются письменный стол и кресло. Основным рабочим положением является положение сидя. Рабочее место для выполнения работ в положении сидя организуется в соответствии с ГОСТ 12.2.032-78 [4].

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Моторное поле – пространство рабочего места, в котором могут осуществляться двигательные действия человека.

Максимальная зона досягаемости рук – это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

Оптимальная зона - часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом.

При проектировании письменного стола следует учитывать следующее:

– высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;

– нижняя часть стола должна быть сконструирована так, чтобы программист мог удобно сидеть, не был вынужден поджимать ноги;

– поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения программиста;

– конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, листингов, канцелярских принадлежностей, личных вещей).

Параметры рабочего места выбираются в соответствии с антропометрическими характеристиками. При использовании этих данных в расчетах следует исходить из максимальных антропометрических характеристик (М+2).

При работе в положении сидя рекомендуются следующие параметры рабочего пространства:

– ширина не менее 700 мм;

– глубина не менее 400 мм;

– высота рабочей поверхности стола над полом 700-750 мм.

Оптимальными размерами стола являются:

– высота 710 мм;

– длина стола 1300 мм;

– ширина стола 650 мм.

Поверхность для письма должна иметь не менее 400 мм в глубину и не менее 600 мм в ширину.

Под рабочей поверхностью должно быть предусмотрено пространство для ног:

- высота не менее 600 мм;
- ширина не менее 500 мм;
- глубина не менее 400 мм.

Важным элементом рабочего места программиста является кресло. Оно выполняется в соответствии с ГОСТ 21.889-76 [7]. При проектировании кресла исходят из того, что при любом рабочем положении программиста его поза должна быть физиологически правильно обоснованной, т.е. положение частей тела должно быть оптимальным. Для удовлетворения требований физиологии, вытекающих из анализа положения тела человека в положении сидя, конструкция рабочего сидения должна удовлетворять следующим основным требованиям:

- допускать возможность изменения положения тела, т.е. обеспечивать свободное перемещение корпуса и конечностей тела друг относительно друга;
- допускать регулирование высоты в зависимости от роста работающего человека (в пределах от 400 до 550 мм);
- иметь слегка вогнутую поверхность,
- иметь небольшой наклон назад.

Исходя из вышесказанного, приведем параметры стола программиста:

- высота стола 710 мм;
- длина стола 1300 мм;
- ширина стола 650 мм;
- глубина стола 400 мм.

Поверхность для письма:

- в глубину 400 мм;
- в ширину 600 мм.

Важным моментом является также рациональное размещение на рабочем месте документации, канцелярских принадлежностей, что должно обеспечить работающему удобную рабочую позу, наиболее экономичные движения и минимальные траектории перемещения работающего и предмета труда на данном рабочем месте.

Создание благоприятных условий труда и правильное эстетическое оформление рабочих мест на производстве имеет большое значение как для облегчения труда, так и для повышения его привлекательности, положительно влияющей на производительность труда. Окраска помещений и мебели должна способствовать созданию благоприятных условий для зрительного восприятия, хорошего настроения. В служебных помещениях, в которых выполняется однообразная умственная работа, требующая значительного нервного напряжения и большого сосредоточения, окраска должна быть спокойных тонов - малонасыщенные оттенки холодного зеленого или голубого цветов

При разработке оптимальных условий труда программиста необходимо учитывать освещенность, шум и микроклимат.

4.3 Освещенность рабочего места

Рациональное освещение рабочего места является одним из важнейших факторов, влияющих на эффективность трудовой деятельности человека, предупреждающих травматизм и профессиональные заболевания. Правильно организованное освещение создает благоприятные условия труда, повышает работоспособность и производительность труда. Освещение на рабочем месте программиста должно быть таким, чтобы работник мог без напряжения зрения выполнять свою работу. Утомляемость органов зрения зависит от ряда причин:

- недостаточность освещенности;
- чрезмерная освещенность;
- неправильное направление света.

Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение и резь в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям, поэтому столь важен правильный расчет освещенности.

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Процесс работы программиста в таких условиях, когда естественное освещение недостаточно или отсутствует. Исходя из этого, рассчитаем параметры искусственного освещения [8].

Искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют существенные преимущества:

– по спектральному составу света они близки к дневному, естественному освещению;

– обладают более высоким КПД (в 1.5-2 раза выше, чем КПД ламп накаливания);

– обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);

– более длительный срок службы.

Расчет освещения производится для комнаты площадью 36 м^2 , ширина которой 4.9 м, высота – 4.2 м. Воспользуемся методом светового потока.

Для определения количества светильников определим световой поток, падающий на поверхность по формуле:

$$F = \frac{E \cdot K \cdot S \cdot Z}{n} \ , \qquad (4)$$

где F – рассчитываемый световой поток, Лм;

E – нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет E = 300 Лк при газоразрядных лампах;

S – площадь освещаемого помещения (в нашем случае S = 36 м^2);

Z – отношение средней освещенности к минимальной (обычно принимается равным 1.1-1.2 , пусть Z = 1.1);

K – коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение определяется по таблице коэффициентов запаса для различных помещений и в нашем случае K = 1.5);

n – коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемых коэффициентами отражения от стен (Pc) и потолка (Pn)), значение коэффициентов Pc и Pn определим по таблице зависимостей коэффициентов отражения от характера поверхности: Pc=30%, Pn=50%. Значение n определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

$$I = \frac{S}{h \cdot (A + B)} \quad , \tag{5}$$

где S – площадь помещения, S = 36 м^2;

h – расчетная высота подвеса, h = 3.39 м;

A – ширина помещения, A = 4.9 м;

B – длина помещения, B = 7.35 м.

Подставив значения, получим:

$$I = \frac{3.6}{3.39 \cdot (4.9+7.35)} = 0.8$$

Зная индекс помещения *I*, *Pc* и *Pn*, по таблице находим *n* = 0.28

Подставим все значения в формулу для определения светового потока *F*:

$$F = \frac{300 \cdot 1.5 \cdot 36 \cdot 1.1}{0.28} = 63642.857 \quad \text{Лм}$$

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световой поток которых *F* = 4320 Лк.

Рассчитаем необходимое количество ламп по формуле:

$$N = \frac{F}{F_л}, \qquad (6)$$

где *N* – определяемое число ламп;

F – световой поток, *F* = 63642,857 Лм;

$F_л$ – световой поток лампы, $F_л$ = 4320 Лм.

$$N = \frac{63642.857}{4320} = 15 \text{ шт.}$$

При выборе осветительных приборов используем светильники типа ОД. Каждый светильник комплектуется двумя лампами. Размещаются светильники двумя рядами, по четыре в каждом ряду.

4.4 Нормирование шума

Установлено, что шум ухудшает условия труда, оказывая вредное воздействие на организм человека. При длительном воздействии шума на человека происходят нежелательные явления: снижается острота зрения, слуха, повышается кровяное давление, понижается внимание. Сильный продолжительный шум может стать причиной функциональных изменений сердечно-сосудистой и нервной систем.

Согласно ГОСТ 12.1.003-88 («Шум. Общие требования безопасности») характеристикой постоянного шума на рабочих местах являются среднеквадратичные уровни давлений в октавных полосах частот со среднегеометрическими стандартными частотами: 63, 125, 250, 500, 1000, 2000, 4000 и 8000 Гц [6]. В этом ГОСТе указаны значения предельно допустимых уровней шума на рабочих местах предприятий. Для помещений конструкторских бюро, расчетчиков и программистов уровни шума не должны превышать соответственно: 71, 61, 54, 49, 45, 42, 40, 38 дБ. Эта совокупность восьми нормативных уровней звукового давления называется предельным спектром.

4.5 Вентиляция

Системы отопления и системы кондиционирования следует устанавливать так, чтобы ни теплый, ни холодный воздух не направлялся на людей. На производстве рекомендуется создавать динамический климат с определенными перепадами показателей. Температура воздуха у поверхности пола и на уровне головы не должна отличаться более, чем на 5 градусов [6]. В производственных помещениях помимо естественной вентиляции предусматривают приточно-вытяжную вентиля-

цию. Основным параметром, определяющим характеристики вентиляционной системы, является кратность обмена, т.е. сколько раз в час сменится воздух в помещении.

Выводы по разделу четыре:

В этой части выпускной квалификационной работы были изложены требования к рабочему месту программиста (пользователя). Созданные условия должны обеспечивать комфортную работу. На основании изученной литературы по данной проблеме, были указаны оптимальные размеры рабочего стола и кресла, рабочей поверхности, а также проведен выбор системы и расчет оптимального освещения производственного помещения, а также расчет информационной нагрузки. Соблюдение условий, определяющих оптимальную организацию рабочего места программиста, позволит сохранить хорошую работоспособность в течение всего рабочего дня, повысит, как в количественном, так и в качественном отношениях производительность труда программиста.

ЗАКЛЮЧЕНИЕ

При выполнении выпускной квалификационной работы был разработан программный продукт для автоматизации системы заявок на плановые внеплановые работы на месторождении. Оперативность диспетчерской службы и цеха КИП и А ООО «Управление Автоматизации и Энергетики Нефтяного Предприятия» повысилась за счёт беспрепятственного доступа к сводке содержащей заявки на проведение работ на месторождении.

Решены следующие задачи:

- исследована предметная область;
- исследована существующая модель данных;
- изучены требования заказчика;
- проанализированы имеющиеся аналоги разрабатываемого ПО.
- выбраны инструменты и средства для разработки, исходя из требований заказчика;
- разработано ПО, отвечающее требованиям заказчика.

Разработанный программный продукт соответствует всем требованиям заказчика. Осуществлён весь заявленный функционал. Программный продукт готов к внедрению на производство.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Макконнелл, Стив. Совершенный код. / Стив Макконнелл. – СПб.: «Питер», 2005. – 896 с.
2. Брауде, Эрик. Технология разработки ПО. / Эрик Брауде. – СПб.: «Питер», 2004. – 655 с.
3. Microsoft Corporation. Основы Microsoft Visual Studio.NET 2008. Пер. с англ. – М.: Издательско-торговый дом «Русская Редакция», 2008. – 464 с.
4. ГОСТ 12.2.032-78 Рабочее место при выполнении работ сидя. Общие эргономические требования. // Справочно-правовая система «КонсультантПлюс». – http://www.consultant.ru/cons/cgi/online.cgi?req=doc;base=STR;n=6342#0 [дата обращения – 26.11.2017]
5. СП 52.13330.2011 Свод правил. Естественное и искусственное освещение. – М.: Минрегион России, 2011. // Справочно-правовая система «КонсультантПлюс». http://www.consultant.ru/cons/cgi/online.cgi?req=doc&base=STR&n=13780#0 [дата обращения – 26.11.2017]
6. ГОСТ 12.1.005-88 Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны. // Справочно-правовая система «КонсультантПлюс». – http://www.consultant.ru/document/cons_doc_LAW [дата обращения – 26.11.2017]
7. ГОСТ 12.1.003-83 ССБТ. Шум. Общие требования безопасности. // Справочно-правовая система «Гарант» http://base.garant.ru/3922239/ [дата обращения – 27.11.2017]
8. ГОСТ 22269-76. Система «Человек-машина». Рабочее место оператора. Взаимное расположение элементов рабочего места. Общие эргономические требования. // Единая база ГОСТОВ РФ «ГОСТ Эксперт» http://gostexpert.ru/gost/gost-22269-76 [дата обращения – 27.11.2017]
9. Беляков, Г.И. Безопасность жизнедеятельности. Охрана труда: Учебник для бакалавров / Г.И. Беляков. – М.: Юрайт, 2013. – 572 с.

10. ГОСТ 12.1.038-82 Система стандартов безопасности труда. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов. // Справочно-правовая система «КонсультантПлюс». http://www.consultant.ru/cons/cgi/online.cgi?req=doc&base=STR&n=6167#0 [дата обращения – 27.11.2017]

11. Скит, Джон. C# для профессионалов: тонкости программирования. 3-е издание: учебник./ Джон Скит. – М.: «Вильямс», 2014. – 608 с.

12. Шарп, Джон. Microsoft Visual C#. Подробное руководство. 8-е издание./ Джон Шарп. – М.: «Питер», 2017. – 848 с.

13. Виссер, Джуст. Разработка обслуживаемых программ на языке C#. / Джуст Виссер. – М.: «ДМК», 2017. – 194 с.

14. Албахари, Джозеф. C# 3.0. Справочник. / Джозеф Албахари, Бен Албахари. – М.: БХВ-Петербург, 2013. – 944 с.

15. Троелсен, Эндрю. Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. – М.: Вильямс, 2015. – 486 с.

16. Зиборов, В. Visual C# 2010 на примерах / В. Зиборов. – М.: "БХВ-Петербург", 2011. – 432 с.

17. Рихтер, Джефри. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#./ Джефри Рихтер. – СПб.: «Питер», 2007. – 656 с.

18. Робинсон, С. C# для профессионалов. / С. Робинсон, О. Корнес, Д. Глинн и др. – М.: Лори, 2005. – 396 с.

19. СанПиН 2.2.1/2.1.1.1278-03 «Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий» (Зарегистрировано в Минюсте России 08.08.2016 № 43153) // Справочно-правовая система «Гарант» http://base.garant.ru/12174919/ [дата обращения – 27.11.2017]

20. Методические рекомендации по подготовке и оформлению выпускной квалификационной работы (проекта) для технических направлений подготовки 09.03.01 Информатика и вычислительная техника, 09.03.04 Программная инжене-

рия, 12.03.01 Приборостроение, 23.03.01 Технология транспортных процессов / сост. Л.Н.Буйлушкина. – Нижневартовск, 2017. – 35 с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ПРОГРАММНОГО ПРОДУКТА «АВТОМАТИЗАЦИЯ СИСТЕМЫ ЗАЯВОК НА ПЛАНОВЫЕ И ВНЕПЛАНОВЫЕ РАБОТЫ НА МЕСТОРОЖДЕНИИ»

1. Наименование программного продукта

Приложение «Заявки».

2. Основание для разработки

Разработка ведется на основании задания выпускной квалификационной работы по направлению 09.03.01 «Информатика и вычислительная техника».

3. Исполнитель

Исполнитель – Юрченко Юрий Юрьевич.

4. Назначение и цель разработки

Разрабатываемое приложение предназначено для учёта и обработки заявок на месторождении

5. Содержание работы

5.1. Задачи, подлежащие решению:

- анализ предметной области и создание ее формального описания;
- определение функций, выполняемых системой;

– выбор основной технологии и средств разработки и реализации программных модулей;

– проектирование БД;

– составление структуры программного комплекса;

– разработка интерфейса взаимодействия программы с пользователем;

– оформление документации.

5.2 Требования к программному продукту

5.2.1 Требования к функциональным характеристикам:

Программный продукт для учёта и обработки заявок должен выполнять следующие функции:

– редактирование БД содержащих необходимую информацию по месторождению;

– содержать информацию о пользователях программного продукта;

– содержать список кустов, скважин, подразделений;

– иметь возможность диспетчеру редактировать, удалять, добавлять, подтверждать заявки;

– иметь возможность обслуживающему персоналу вносить заявки для последующего подтверждения.

5.2.2 Требования к интерфейсу диспетчера:

Диспетчер должен иметь возможность:

– авторизоваться в приложении.

– чётко понимать, каким образом необходимо вводить информациюо заявках.

Администратор должен иметь возможность:

– редактирования информации о пользователях.

– редактирования информации справочников.

5.3 Требование к программному обеспечению

ПО клиентской части должно удовлетворять следующим требованиям:

- Microsoft Windows 7, Microsoft Windows 8, Microsoft Windows 8.1, Microsoft Windows 10;

5.4 Требования к архитектуре системы

Система должна включать в себя:

- Приложение, производящее работу с БД и реализующую интерфейс с пользователем;
- БД для хранения информации;
- приложение, ответственное за выполнение функций системы.

5.5 Требования к БД

Приложение использует файловую СУБД для хранения следующей информации:

- список кустов;
- список пользователей;
- список скважин;
- список подразделений;

5.6 Требования к входным и выходным данным

5.6.1 Входные данные

Входными данными является информация о заявках, кустах, скважинах, подразделениях.

5.6.2 Выходные данные

Выходными данным является информация о заявках на месторождении.

5.7 Требования к составу и характеристикам технических средств

Приложение должно функционировать на IBM–совместимой ЭВМ следующей конфигурации:

- процессор семейства Intel Pentium IV и выше;
- оперативная память не менее 1 Гбайт;
- дисковая подсистема со свободным дисковым пространством;
- не менее 2 Гбайт;
- цветной монитор с поддержкой SVGA–режимов;
- стандартная русифицированная клавиатура;
- манипулятор мышь.

5.7.1 Требования к лингвистическому обеспечению

Интерфейс пользователя должен быть реализован на русском языке.

5. Этапы разработки

Таблица А.1 – Этапы разработки

№	Этапы разработки	Срок начала	Срок выполнения
1	Разработка технического задания	01.11.2017	04.11.2017
2	Разработка технического проекта	05.11.2017	20.11.2017
3	Проектирование БД	21.11.2017	30.11.2017
4	Разработка интерфейса программы	01.12.2017	04.12.2017
5	Разработка программного обеспечения	05.12.2017	08.12.2017
6	Отладка программы	09.12.2017	14.12.2017
7	Разработка программной документации	15.12.2017	20.12.2017
8	Оформление и представление документации	20.12.2017	25.12.2017

6. Требования к документации

Для приема программного продукта должны быть предоставлены следующие документы:

7.1. Техническое задание.

7.2. Пояснительная записка.

8. Дополнительные условия

Данное техническое задание может уточняться и изменяться в установленном порядке.

ПРИЛОЖЕНИЕ Б.КОМПАКТ-ДИСК

Содержание:

1. Пояснительная записка.
2. Презентация.
3. Исходный код программы.

ПРИЛОЖЕНИЕ В. ИСХОДНЫЙ КОД ПРОГРАММНОГО ПРОДУКТА «АВТОМАТИЗАЦИЯ СИСТЕМЫ ЗАЯВОК НА ПЛАНОВЫЕ И ВНЕПЛАНОВЫЕ РАБОТЫ НА МЕСТОРОЖДЕНИИ»

Заявки

```
using System;
using System.Data;
using System.Linq;
using System.Windows.Forms;
using Nwuram.Framework.Settings.Connection;
using Nwuram.Framework.Settings.User;
using Nwuram.Framework.ToExcelNew;
using Nwuram.Framework.Logging;

namespace RepairRequest
{
    public partial class Form1 : Form
    {
        private readonly Sql _sql = new
Sql(ConnectionSettings.GetServer(), ConnectionSettings.GetDatabase(),
            ConnectionSettings.GetUsername(), ConnectionSet-
tings.GetPassword(), ConnectionSettings.ProgramName);

        private DataTable _dt;
        private int _inConfirm;
        private int _inWork;
        private int indexColSort = 0;

        public Form1()
        {
            InitializeComponent();

            dateTimePicker_From.MaxDate = DateTime.Now;
            dateTimePicker_Till.MaxDate = DateTime.Now;
```

```
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Init();

            // Настройки дополнений
            dataGridView.AllowUserToResizeColumns = true;
        }

        private void Init()
        {
            InitDateTimePickers();
            InitCombobox();
            InitDataGrid();
            InitToolTips();
        }

        private void InitToolTips()
        {
            toolTip1.SetToolTip(button_Exit, "Выход");
            toolTip1.SetToolTip(button_Report, "");
            toolTip1.SetToolTip(button_Comment, "");
            toolTip1.SetToolTip(button_HwList, "Справочник");
            toolTip1.SetToolTip(button_ResList, "Справочник
}

        private void InitDateTimePickers()
        {
            dateTimePicker_From.Value = DateTime.Now.AddDays(-
1);
        }

        private void InitCombobox()
        {
```

```
        var dt = new DataTable();
        dt.Columns.Add("");
        dt.Columns.Add("");
        dt.Rows.Add(0, "");
        dt.Rows.Add(1, "");
        dt.Rows.Add(2, "");
        dt.Rows.Add(3, "");
        dt.Rows.Add(4, "");
        comboBox1.DataSource = dt;
        comboBox1.DisplayMember = "str";
        comboBox1.ValueMember = "id";
        comboBox1.SelectedValue = 4;
    }

    private string NumberToStatus(int num)
    {
        switch (num)
        {
            case 0:
                return "";
            case 1:
                return "";
            case 2:
                return "";
            case 3:
                return "";
        }
        return string.Empty;
    }

    private void InitDataGrid()
    {
        dataGridView.Columns.Add("reqNum", ");
        dataGridView.Columns.Add("dateSub", "");
        dataGridView.Columns.Add("timeSub", "");
```

```
                dataGridView.Columns.Add("", ");
                dataGridView.Columns.Add("", "");
                dataGridView.Columns.Add("", "");
                dataGridView.Columns.Add("", "");
                dataGridView.Columns.Add("", "");
                dataGridView.Columns.Add("", "");

                dataGridView.Columns[""].Visible = false;
                dataGridView.Columns[""].Visible = false;

                dataGridView.Columns["dateSub"].SortMode = Data-
GridViewColumnSortMode.Programmatic;
                dataGridView.Columns["timeSub"].SortMode = Data-
GridViewColumnSortMode.Programmatic;

                dataGridView.Columns["dateDone"].SortMode = Data-
GridViewColumnSortMode.Programmatic;
                dataGridView.Columns["timeDone"].SortMode = Data-
GridViewColumnSortMode.Programmatic;
            }

            private void Report()
            {
                var report = new ExcelUnLoad();
                var dateStr = " " + dateTimePick-
er_From.Value.ToShortDateString() + " по: " +
                 dateTimePicker_Till.Value.ToShortDateString();
                 var datePrintStr = " " + DateTime.Now;
var whoPrinted = "Выгрузил: " + UserSettings.User.FullUsername;

                Logging.StartFirstLevel(79);
                Logging.Comment("");
                Logging.Comment(dateStr);
                Logging.Comment(" ID:" + Nwu-
ram.Framework.Settings.User.UserSettings.User.Id
```

```
                                    + " :" + Nwu-
ram.Framework.Settings.User.UserSettings.User.FullUsername);
                    Logging.StopFirstLevel();

                    report.AddSingleValue(dateStr, 1, 1);
                    report.AddSingleValue(datePrintStr, 1, 9);
                    report.AddSingleValue(whoPrinted, 2, 9);
                    report.AddSingleValue("", 4, 1);
                    report.AddSingleValue("", 4, 2);
                    report.AddSingleValue("", 4, 3);
                    report.AddSingleValue("", 4, 4);
                    report.AddSingleValue("i", 4, 5);
                    report.AddSingleValue("", 4, 6);
                    report.AddSingleValue("", 4, 7);
                    report.AddSingleValue("", 4, 8);
                    report.AddSingleValue("", 4, 9);
                    report.AddSingleValue("\", 4, 10);
                    report.AddSingleValue("\", 4, 11);
                    report.AddSingleValue(",\", 4, 12);
                    report.AddSingleValue("", 4, 13);
                    report.AddSingleValue("", 4, 14);
                    report.SetBorders(4, 1, 4, 14);
                    report.SetFontBold(4, 1, 4, 14);
                    var rowCount = 5;
                    foreach (DataGridViewRow row in dataGridView.Rows)
                        if ((int) row.Cells["status"].Tag == 3)
                        {
                            var rowCountStart = rowCount;
report.AddSingleValue(row.Cells["reqNum"].Value.ToString(), rowCount,
1);
report.AddSingleValue(row.Cells["status"].Value.ToString(), rowCount,
2);
report.AddSingleValue(row.Cells["dateSub"].Value + " " +
row.Cells["timeSub"].Value, rowCount,
                                    3);
```

```
      report.AddSingleValue(row.Cells["cab"].Value.ToString(), row-
Count, 4);
      report.AddSingleValue(row.Cells["ip"].Value.ToString(), row-
Count, 5);
      report.AddSingleValue(row.Cells["fio"].Value.ToString(), row-
Count, 6);
      report.AddSingleValue(row.Cells["hw"].Value.ToString(), row-
Count, 7);
                           var id = (int) row.Cells["reqNum"].Tag;
                           var tmpRow = FindRowById(id);
      report.AddSingleValue(tmpRow["Fault"].ToString().Trim(), row-
Count, 8);
      report.AddSingleValue(row.Cells["res"].Value.ToString(), row-
Count, 9);
      report.AddSingleValue(tmpRow["DataStart"].ToString().Trim(),
rowCount, 10);
      report.AddSingleValue(tmpRow["DateEnd"].ToString().Trim(), row-
Count, 11);
                           report.AddSingleValue(
                               row.Cells["timeCount"].Value == null ?
0.ToString() : row.Cells["timeCount"].Value.ToString(),
                               rowCount, 12);
      report.AddSingleValue(tmpRow["endName"].ToString().Trim(), row-
Count, 13);
                           var comments = _sql.GetComments(id);
                           foreach (DataRow comment in comments.Rows)
                           {
report.AddSingleValue(comment["Comment"].ToString().Trim(), rowCount,
14);
                               rowCount++;
                           }
                           for (var i = 1; i < 13; i++)
                       report.Merge(rowCountStart, i, rowCount - 1, i);
                           report.SetBorders(rowCountStart, 1, rowCount
- 1, 14);
```

```
                    }
                report.SetColumnAutoSize(4, 1, rowCount, 14);
                report.SetColumnWidth(4, 2, rowCount, 2, 30);
                report.SetColumnWidth(4, 14, rowCount, 14, 30);
                report.SetWrapText(4, 1, rowCount, 14);
                report.Show();
            }

            private void UpdateData()
            {
                int current = -1;
                if (dataGridView.CurrentRow != null && data-
GridView.RowCount > 0 && dataGridView.CurrentRow.Index > 0)
                {
                    current = dataGridView.CurrentRow.Index;
                }
                dataGridView.Rows.Clear();
                _dt =
_sql.GetRepairRequests(Convert.ToDateTime(dateTimePicker_From.Value.To
ShortDateString()), Con-
vert.ToDateTime(dateTimePicker_Till.Value.ToShortDateString()).AddDays
(1));
                _inWork = 0;
                _inConfirm = 0;
                foreach (DataRow row in _dt.Rows)
                {
                    var rowIndex = dataGridView.Rows.Add();
                    data-
GridView.Rows[rowIndex].Cells["reqNum"].Value = row["Number"];
                    dataGridView.Rows[rowIndex].Cells["reqNum"].Tag
= int.Parse(row["id"].ToString());
                    var dateSub =
DateTime.Parse(row["DateSubmission"].ToString());
```

```
                    data-
GridView.Rows[rowIndex].Cells["dateSub"].Value =
dateSub.ToShortDateString();
                    data-
GridView.Rows[rowIndex].Cells["timeSub"].Value =
dateSub.ToLongTimeString();
                    dataGridView.Rows[rowIndex].Cells["cab"].Value =
row["Cabinet"].ToString().Trim();
                    dataGridView.Rows[rowIndex].Cells["ip"].Value =
row["ip_address"].ToString().Trim();
                    dataGridView.Rows[rowIndex].Cells["fio"].Value =
row["FIO"].ToString().Trim();
                    dataGridView.Rows[rowIndex].Cells["hw"].Value =
row["cName"].ToString().Trim();
                    dataGridView.Rows[rowIndex].Cells["res"].Value =
row["resName"].ToString().Trim();
                    data-
GridView.Rows[rowIndex].Cells["cComments"].Value =
row["strComment"].ToString().Trim();
                    DateTime dateDone;
                    if (DateTime.TryParse(row["DateEnd"].ToString(),
out dateDone))
                    {
                        int hourCount = getHourCount(dateSub,
dateDone);

                        data-
GridView.Rows[rowIndex].Cells["dateDone"].Value =
dateDone.ToShortDateString();
                        data-
GridView.Rows[rowIndex].Cells["timeDone"].Value =
dateDone.ToLongTimeString();
                        //var hourCount = (dateDone -
dateSub).TotalHours;
```

```
//dataGridView.Rows[rowIndex].Cells["timeCount"].Value = (int) hour-
Count;
                        data-
GridView.Rows[rowIndex].Cells["timeCount"].Value = hourCount;
                    }
                    var status =
int.Parse(row["Status"].ToString());
                    data-
GridView.Rows[rowIndex].Cells["status"].Value = NumberToSta-
tus(status);
                    dataGridView.Rows[rowIndex].Cells["status"].Tag
= status;
                    switch (status)
                    {
                        case 1:
                            data-
GridView.Rows[rowIndex].DefaultCellStyle.BackColor = pan-
el_InWork.BackColor;
                            _inWork++;
                            break;
                        case 2:
                            data-
GridView.Rows[rowIndex].DefaultCellStyle.BackColor = pan-
el_NotConfirmed.BackColor;
                            _inConfirm++;
                            break;
                    }

                    data-
GridView.Rows[rowIndex].Cells["dateStr"].Value = dateSub;
                    data-
GridView.Rows[rowIndex].Cells["dateFin"].Value = dateDone;
```

```
                }

                if (current != -1 && dataGridView.Rows.Count < cur-
rent)
                    dataGridView.Rows[current].Selected = true;

                if (_dt != null && _dt.Rows.Count > 0)
                {
                    DataRow cur = (current != -1) ?
_dt.Rows[current] : _dt.Rows[0];

                    textBox_Create.Text =
cur["startName"].ToString();
                    textBox_DateCreate.Text =
cur["DataStart"].ToString();
                    textBox_Done.Text = cur["endName"].ToString();
                    textBox_DateDone.Text =
cur["DateEnd"].ToString();
                    textBox_Confirm.Text =
cur["confirmName"].ToString();
                    textBox_DateConfirm.Text =
cur["DateConfirm"].ToString();
                }

                textBox_InWork.Text = _inWork.ToString();
                textBox_NotConfirmed.Text = _inConfirm.ToString();
                HideFunc();
            }

            private void HideRowsBasedOnCombobox()
            {
                var selectedStatus =
int.Parse(comboBox1.SelectedValue.ToString());
                if (selectedStatus == 4)
                {
```

```
                    foreach (DataGridViewRow row in data-
GridView.Rows)
                        row.Visible = true;
                    return;
                }
                    foreach (DataGridViewRow row in data-
GridView.Rows)
                        if (row.Visible)
                        {
                            row.Visible = (int)
row.Cells["status"].Tag == selectedStatus;
                        }
            }

            private void HideFunc()
            {
                foreach (DataGridViewRow row in dataGridView.Rows)
                {
                    row.Visible = true;
                }
                HideRowsBasedOnCombobox();
                HideRowsBasedOnCheckbox();
                FilterCabinet();
                FilterFio();
                FilterHw();
                FilterNumber();
            }

            private void HideRowsBasedOnCheckbox()
            {
                foreach (DataGridViewRow row in dataGridView.Rows)
                    if (row.Visible)
                    {
                        if ((int) row.Cells["status"].Tag == 2)
                            row.Visible = !checkBox_Done.Checked;
```

```
            if((int) row.Cells["status"].Tag == 3)
            row.Visible = !checkBox_Ok.Checked;
            }
            }

            private void button1_Click(object sender, EventArgs e)
            {
                UpdateData();
            }

            private void button_HwList_Click(object sender, Even-
tArgs e)
            {
                var form = new HardwareListForm();
                form.ShowDialog();
            }

            private void button_Exit_Click(object sender, EventArgs
e)
            {
                Close();
            }

            private DataRow FindRowById(int id)
            {
                return _dt.Rows.Cast<DataRow>().FirstOrDefault(row
=> int.Parse(row["id"].ToString()) == id);
            }

            private void dataGridView_SelectionChanged(object send-
er, EventArgs e)
            {
                if (dataGridView.CurrentRow == null || data-
GridView.CurrentRow.Cells["reqNum"].Tag == null) return;
```

```
                var selectedId = (int) data-
GridView.CurrentRow.Cells["reqNum"].Tag;
                var row = FindRowById(selectedId);
                textBox_Create.Text = row["startName"].ToString();
                textBox_DateCreate.Text =
row["DataStart"].ToString();
                textBox_Done.Text = row["endName"].ToString();
                textBox_DateDone.Text = row["DateEnd"].ToString();
                textBox_Confirm.Text =
row["confirmName"].ToString();
                textBox_DateConfirm.Text =
row["DateConfirm"].ToString();
            }

            private void checkBox_Done_CheckedChanged(object sender,
EventArgs e)
            {
                HideFunc();
            }

            private void comboBox1_SelectionChangeCommitted(object
sender, EventArgs e)
            {
                HideFunc();
            }

            private void dataGridView_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
            {
                try
                {
                    int select = dataGridView.CurrentRow.Index;
                    var id =
(int)dataGridView.Rows[e.RowIndex].Cells["reqNum"].Tag;
                    var row = FindRowById(id);
```

```
                        var form = new RepairWorkForm(row);
                        form.ShowDialog();
                        SortOrder tempSort = dataGridView.SortOrder;
                        //dataGridView.Sort(dataGridView.Columns[0],
System.ComponentModel.ListSortDirection.Ascending);
                        UpdateData();
                        if (tempSort == SortOrder.Ascending)
                            data-
GridView.Sort(dataGridView.Columns[indexColSort], Sys-
tem.ComponentModel.ListSortDirection.Ascending);
                        else
                            data-
GridView.Sort(dataGridView.Columns[indexColSort], Sys-
tem.ComponentModel.ListSortDirection.Descending);
                                        data-
GridView.FirstDisplayedScrollingRowIndex = select;

//dataGridView.SelectedRows.Clear();
                        foreach (DataGridViewRow gr in data-
GridView.Rows)
                        {
                            if (id == (int)gr.Cells["reqNum"].Tag)
                            {
                                dataGridView.CurrentCell = gr.Cells[0];
                                dataGridView.Rows[gr.Index].Selected =
true;
                                break;
                            }
                        }
                        //dataGridView.CurrentCell = data-
GridView.Rows[select].Cells[0];
                        //dataGridView.Rows[select].Selected = true;
                        ////dataGridView.CurrentRow = data-
GridView.Rows[select];
```

```
                    Con-
sole.WriteLine(dataGridView.CurrentRow.Index);
                }
                catch { }
            }
            private void button_Comment_Click(object sender, Even-
tArgs e)
            {
                if (dataGridView.CurrentRow == null) return;
                var id = (int) data-
GridView.CurrentRow.Cells["reqNum"].Tag;
                var form = new ViewCommentsForm(id);
                form.ShowDialog();
            }
            private void textBox1_TextChanged(object sender, Even-
tArgs e)
            {
                HideFunc();
            }
            private void FilterNumber()
            {
                    foreach (DataGridViewRow row in data-
GridView.Rows)
                        if (row.Visible)
                            row.Visible =
row.Cells["reqNum"].Value.ToString().Contains(textBox_Number.Text);
            }
            private void textBox_Number_KeyPress(object sender, Key-
PressEventArgs e)
            {
                if (char.IsDigit(e.KeyChar) || e.KeyChar == '\b')
                    e.Handled = false;
                else
                    e.Handled = true;
            }
```

```
            private void textBox4_TextChanged(object sender, Even-
tArgs e)
            {
                HideFunc();
            }
            private void FilterCabinet()
            {
                    foreach (DataGridViewRow row in data-
GridView.Rows)
                        if (row.Visible)
                            row.Visible =
row.Cells["cab"].Value.ToString().Contains(textBox_Cabinet.Text);
            }
            private void textBox_Cabinet_KeyPress(object sender,
KeyPressEventArgs e)
            {
                if (char.IsLetterOrDigit(e.KeyChar) || e.KeyChar ==
'\b')
                    e.Handled = false;
                else
                    e.Handled = true;
            }
            private void textBox_Fio_TextChanged(object sender,
EventArgs e)
            {
                HideFunc();
            }
            private void FilterFio()
            {
                    foreach (DataGridViewRow row in data-
GridView.Rows)
                        if (row.Visible)
                            row.Visible =
row.Cells["fio"].Value.ToString().Contains(textBox_Fio.Text);
            }
```

```
            private void textBox_Fio_KeyPress(object sender, Key-
PressEventArgs e)
            {
                if (char.IsLetter(e.KeyChar) || e.KeyChar == '\b')
                    e.Handled = false;
                else
                    e.Handled = true;
            }
            private void textBox_Hw_TextChanged(object sender, Even-
tArgs e)
            {
                HideFunc();
            }
            private void textBox_Hw_KeyPress(object sender, KeyPres-
sEventArgs e)
            {
                if (char.IsLetterOrDigit(e.KeyChar) || e.KeyChar ==
'\b')
                    e.Handled = false;
                else
                    e.Handled = true;
            }
            private void FilterHw()
            {
                    foreach (DataGridViewRow row in data-
GridView.Rows)
                        if (row.Visible)
                            row.Visible =
row.Cells["hw"].Value.ToString().Contains(textBox_Hw.Text);
            }
            private void button_Report_Click(object sender, Even-
tArgs e)
            {
                Report();
            }
```

```
            private void button_ResList_Click(object sender, Even-
tArgs e)
            {
                var form = new ResponsibleListForm();
                form.ShowDialog();
            }
            private int getHourCount(DateTime start, DateTime end)
            {
                            DateTime tempStart = new DateTime(0000,
00);
                DateTime tempEnd = new DateTime(0000, 00);
                DataTable holiday = _sql.getHoliday(start,
DateTime.Now);
                tempStart = tempStart.AddYears(start.Year -
1).AddMonths(start.Month - 1).AddDays(start.Day - 1);
                if (start.Hour < 9 || (start.Hour == 9 &&
start.Minute < 30))
                    tempStart =
tempStart.AddHours(9).AddMinutes(30);
                else
                    tempStart =
tempStart.AddHours(start.Hour).AddMinutes(start.Minute);
                tempEnd = tempEnd.AddYears(start.Year -
1).AddMonths(start.Month - 1).AddDays(start.Day - 1);
                if (tempEnd.Year <= end.Year && tempEnd.Month <=
end.Month && tempEnd.Day < end.Day)
                    tempEnd = tempEnd.AddHours(17).AddMinutes(30);
                else if (end.Hour > 17 || (end.Hour == 17 &&
end.Minute > 30))
                    tempEnd = tempEnd.AddHours(17).AddMinutes(30);
                else
                    tempEnd =
tempEnd.AddHours(end.Hour).AddMinutes(end.Minute);
                double hourCount = 0;
                if (tempStart < end)
```

```
                {
                    if ((int)tempStart.DayOfWeek == 0 ||
(int)tempStart.DayOfWeek == 6 ||
                        (holiday != null && holiday.Rows.Count > 0
&& holiday.Select("Date_Holiday = #" + tempStart.ToString("yyyy-MM-
dd") + "#").Count() > 0))
                    {
                        tempStart = tempStart.AddDays(1);
                        tempEnd = tempEnd.AddDays(1);
                    }
                    else
                    {
                        tempStart = tempStart.AddDays(1);
                        tempEnd = tempEnd.AddDays(1);
                        if (tempEnd > tempStart)
                            hourCount += (tempEnd -
tempStart).TotalHours;
                    }
                    tempStart = tempStart.AddHours(-
tempStart.Hour).AddMinutes(-tempStart.Minute);
                    tempStart =
tempStart.AddHours(9).AddMinutes(30);
                    while (tempStart.Year < end.Year ||
                            tempStart.Year == end.Year &&
tempStart.Month < end.Month ||
                            tempStart.Year == end.Year &&
tempStart.Month == end.Month && tempStart.Day < end.Day)
                    {
                        if ((int)tempStart.DayOfWeek == 0 ||
(int)tempStart.DayOfWeek == 6 ||
                            (holiday != null && holiday.Rows.Count >
0 && holiday.Select("Date_Holiday = #" + tempStart.ToString("yyyy-MM-
dd") + "#").Count() > 0))
                        {
                            tempStart = tempStart.AddDays(1);
```

```
                    tempEnd = tempEnd.AddDays(1);
                }
                else
                {
                    hourCount += (tempEnd -
tempStart).TotalHours;
                    tempStart = tempStart.AddDays(1);
                    tempEnd = tempEnd.AddDays(1);
                }
            }
            if (tempStart.Year == end.Year &&
tempStart.Month == end.Month && tempStart.Day == end.Day)
            {
                tempEnd = tempEnd.AddHours(-
tempEnd.Hour).AddMinutes(-tempEnd.Minute);
                if (end.Hour > 17 || (end.Hour == 17 &&
end.Minute > 30))
                    tempEnd =
tempEnd.AddHours(17).AddMinutes(30);
                else
                    tempEnd =
tempEnd.AddHours(end.Hour).AddMinutes(end.Minute);
                if ((int)tempStart.DayOfWeek == 0 ||
(int)tempStart.DayOfWeek == 6 ||
                    (holiday != null && holiday.Rows.Count >
0 && holiday.Select("Date_Holiday = #" + tempStart.ToString("yyyy-MM-
dd") + "#").Count() > 0))
                {
                    tempStart = tempStart.AddDays(1);
                    tempEnd = tempEnd.AddDays(1);
                }
                else
                {
                    hourCount += (tempEnd -
tempStart).TotalHours;
```

```
                            tempStart = tempStart.AddDays(1);
                            tempEnd = tempEnd.AddDays(1);
                        }
                    }
                }
                return (int)hourCount;
            }
            private void dataGridView_ColumnHeaderMouseClick(object
sender, DataGridViewCellMouseEventArgs e)
            {
                indexColSort = e.ColumnIndex;
                if (((Data-
GridView)sender).Columns[e.ColumnIndex].Name == "dateSub" || ((Data-
GridView)sender).Columns[e.ColumnIndex].Name == "timeSub")
                {
                    if (((DataGridView)sender).SortOrder != SortOr-
der.Ascending)
                        ((Data-
GridView)sender).Sort(((DataGridView)sender).Columns["dateStr"], Sys-
tem.ComponentModel.ListSortDirection.Ascending);
                    else
                        ((Data-
GridView)sender).Sort(((DataGridView)sender).Columns["dateStr"], Sys-
tem.ComponentModel.ListSortDirection.Descending);
                    //indexColSort = (Data-
GridView)sender).Columns["dateStr"];
                    indexColSort = data-
GridView.Columns["dateStr"].Index;
                }
                else if (((Data-
GridView)sender).Columns[e.ColumnIndex].Name == "dateDone" || ((Data-
GridView)sender).Columns[e.ColumnIndex].Name == "timeDone")
                {
                    if (((DataGridView)sender).SortOrder != SortOr-
der.Ascending)
```

```
                        ((Data-
GridView)sender).Sort(((DataGridView)sender).Columns["dateFin"], Sys-
tem.ComponentModel.ListSortDirection.Ascending);
                    else
                        ((Data-
GridView)sender).Sort(((DataGridView)sender).Columns["dateFin"], Sys-
tem.ComponentModel.ListSortDirection.Descending);
                    //indexColSort =
(int)((DataGridView)sender).Sort(((DataGridView)sender).Columns["dateF
in"].Index.ToString();
                    indexColSort = data-
GridView.Columns["dateFin"].Index;
                }

                return;
            }
        }
        public enum Modes
        {
            Create,
            Edit
        }
    }
```