СОДЕРЖАНИЕ

Введение 4

1 Технический проект 7

1.1 Анализ предметной области 7

1.2 Постановка задачи 10

1.3 Проектирование базы данных 11

1.4 Проектирование функциональности программы 15

1.5 Проектирование структуры программы 20

1.6 Выбор средств реализации 22

1.7 Требования к программному обеспечению и техническим средствам 24

2 Рабочий проект 26

2.1 Физическая модель данных 26

2.2 Функциональное взаимодействие модулей программы 28

2.3 Входные и выходные данные 30

2.4 Установка и настройка программы 32

2.5 Работа с программой 33

Заключение 42

Список использованных источников 45

Приложение А – Код основных модулей 46

ВВЕДЕНИЕ

В деловой и личной сфере часто приходится работать с данными из разных источников, каждый из которых связан с определенным видом деятельности. Для координации всех этих данных необходимы определенные знания и организационные навыки, а также специальные системы для хранения данных – базы данных (БД). Система БД в SQL Server Management Studio представляет собой совокупность инструментов для ввода, хранения, просмотра, выборки и управления информацией. К этим средствам относятся таблицы, формы, отчеты, запросы.

Система управления БД (СУБД) являются едва ли не самым распространенным видом программного обеспечения. СУБД имеют более чем тридцатилетнюю историю развития с сохранением преемственности и устойчивых традиций. Идеологическая ценность СУБД объясняется тем, что в основе программ такого рода лежит концепция модели данных, то есть некоторой абстракции представления данных.

Организация учета специальной одежды (спецодежды) – один из наиболее сложных участков учетной работы. На промышленном предприятии номенклатура материальных ценностей исчисляется десятками тысяч наименований, а информация по учету спецодежды составляет около 30 процентов всей информации по управлению производством. Поэтому организация учета и контроля за движением, сохранностью и использованием спецодежды связана с большими трудностями. Важное значение имеет автоматизация всех учетных работ, начиная от выписки учетных документов и заканчивая составлением необходимой отчетности.

Обработка больших объемов информации стала не под силу человеку, поэтому для быстрой и достоверной обработки данных используются персональные компьютеры. Информация в компьютере структурируется и хранится, как правило, в виде таблиц. В свою очередь отдельные таблицы объединяются в БД.

Неотъемлемой частью учета и контроля спецодежды является анализ её использования.

Темой данного дипломного проекта является «Разработка и создание приложения для учета выдачи спецодежды на предприятии».

Цель дипломного проекта – изучить информационную систему учета спецодежды на складе, спроектировать и создать информационную систему, используя инструменты моделирования.

Объектом исследования будет система учета выдачи спецодежды на предприятии.

Предметом исследования будут являться инструментальные средства, с помощью которых будет происходить проектирование и реализация приложения для учета выдачи спецодежды.

Для достижения поставленных целей будут решены следующие задачи:

* изучена предметная область и спроектированы концептуальная и логическая модели БД для хранения необходимой информации;
* спроектирована функциональность и структура приложения, работающего с созданной БД;
* произведен выбор средств реализации;
* описаны минимальные требования к техническим и программным средствам;
* разработана физическая модель БД;
* спроектирован и разработан алгоритм программы;
* описаны входные и выходные данные;
* разработано руководство пользователя;
* создана инсталляция.

Написание данного дипломного проекта целесообразно, потому что приложение облегчит работу сотрудника склада, отвечающего за выдачу спецодежды.

Теоретическими основами и методами решения поставленных задач будут методологии проектирования функциональности приложения, структуры БД, особенности реализации программного кода с помощью конкретного языка программирования, принципы и алгоритмы применения инструментальных средств на различных этапах разработки программного продукта.

Практическая значимость проекта заключается в возможности использования приложения в рабочем процессе для автоматизации учета выдачи спецодежды на предприятии.

1 ТЕХНИЧЕСКИЙ ПРОЕКТ

1.1 Анализ предметной области

Важную роль на современном производстве играет специальная одежда. Она защищает работника от вредных условий. Хотя эстетические и гигиенические свойства спецодежды немаловажны, главная ее функция – защитная, поэтому при закупках спецодежды очень важно обращать внимание на качество предоставляемой защиты.

Под спецодеждой подразумевают специальные носимые устройства, а также виды одежды и обуви, предназначенные для полной или частичной индивидуальной защиты человека от различных поражающих воздействий и загрязнений, в том числе, вызванных опасными и вредными для здоровья производственными факторами.

Право работников на труд в условиях, отвечающих требованиям охраны труда, а также гарантии государства по защите этого права, закреплены в Трудовом кодексе. Для обеспечения этого законного права на работах с вредными или опасными условиями труда, а также на работах, выполняемых в особых температурных условиях или связанных с загрязнением, работникам бесплатно выдается сертифицированная специальная одежда.

Министерством труда разработан ряд документов, которые являются обязательными для организаций всех форм собственности. К таковым, в частности, относятся:

* типовые нормы бесплатной выдачи средств индивидуальной защиты работникам общих профессий и должностей для всех отраслей экономики, утвержденные постановлением Министерства труда и социальной защиты Российской Федерации от 22.09.2006 № 110;
* перечень средств индивидуальной защиты, непосредственно обеспечивающих безопасность труда, утвержденный постановлением Министерства труда и социальной защиты Российской Федерации от 19.04.2000 № 65;
* правила обеспечения работников средствами индивидуальной защиты, утвержденные постановлением Министерства труда и социальной защиты Российской Федерации от 28.05.1999 № 67.

Спецодежда обязана иметь соответствующие характеристики для определенного типа работы. Она должна обладать следующими качествами:

* терморегуляция – современные материалы, в частности, мембранная ткань, и особенности кроя (отстегивающийся жилет, съемный капюшон) не допускают как переохлаждения, так и перегрева работника;
* безопасность – спецодежда надежно защищает работника от контакта с вредными и опасными веществами;
* удобство и легкий вес – спецодежда спроектирована таким образом, что работать в ней легко и комфортно;
* прочность и простота в уходе – специальные пропитки и ткани, использованные в изготовлении спецодежды, защищают ее от преждевременного изнашивания и предохраняют от загрязнения;
* компактность и несминаемость – такую одежду удобно хранить;
* аккуратный и современный вид спецодежды – спецодежда придает работнику презентабельный вид и дает возможность использовать такую одежду не только в производстве, но и в быту.

Защитные свойства обеспечиваются следующими обязательными показателями защитной эффективности спецодежды:

* от механических воздействий и общих производственных загрязнений – сопротивление взрыву детали;
* от повышенных температур, теплового излучения и пониженных температур – воздухо- и паропроницаемость пакета;
* от открытого пламени – воздухопроницаемость пакета;
* от радиоактивных веществ – коэффициент защиты, способность к дезактивации;
* от рентгеновских излучений – свинцовый эквивалент;
* от нетоксичной пыли, асбеста, пыли стекловолокна – пылепроницаемость, устойчивость к обеспыливанию;
* от кислоты – кислотопроницаемость;
* от щелочи – щелочепроницаемость;
* от электрических зарядов – электрическое сопротивление, коэффициент защиты;
* от электронных полей – коэффициент защиты;
* от магнитных полей – коэффициент защиты;
* от воды – водопроницаемость;
* от лаков и красок – проницаемость лаков и красок;
* от органических растворителей – проницаемость органических растворителей;
* от насекомых – проницаемость насекомых, устойчивость к дезинсекции;
* от сырой нефти – проницаемость нефти;
* от масел и жиров – проницаемость масел и жиров.

Зимняя спецодежда создается для того, чтобы сохранить здоровье и работоспособность, обеспечить комфорт и безопасность для работников различных профессий, выполняющих работу в неотапливаемых помещениях и на открытом воздухе. Такая одежда отличается высокой степенью теплоизоляции, ветрозащиты, водоупорностью, износостойкостью, а также не сковывает движения.

Для защиты от ветра используются ветрозащитные мембраны, кулисы и современная ветрозащитная ткань. Повышенная прочность изделий обеспечивается двойными строчками и защитными вставками.

Современный ассортимент спецодежды позволяет выбрать оптимальную модель в зависимости от предполагаемых условий эксплуатации:

* полукомбинезон и куртка;
* брюки и куртка;
* брюки и жилет [13].

При учете выдачи спецодежды на предприятии нужно отслеживать вид и тип выдаваемой спецодежды, периодичность ее обновления, стоимость комплекта, дату выдачи, кому выдается (так как учитывается скидка конкретного рабочего).

Работать с информацией о выдаче спецодежды будет кладовщик. Просматривать данную информацию будет также директор предприятия. Следовательно, при разработке базы данных и приложения нужно будет предусмотреть возможность авторизации и разделения функций для различных пользователей.

Периодически надо будет формировать различные отчеты с информацией о выдаче спецодежды, следовательно, надо в приложении реализовать функцию создания и сохранения в файлах отчетов.

Текущая работа с информацией также предполагает поиск и фильтрацию данных по определенным критериям, обновление, удаление и добавление данных, следовательно, разрабатываемое приложение должно предоставлять такие возможности.

1.2 Постановка задачи

Необходимо разработать приложение, которое облегчит выполнение работы кладовщика и директора. Оно должно использовать информацию, хранимую в БД, и позволять пользователям оперативно получать необходимую им информацию о выдаче спецодежды в удобном виде, а также изменять ее. Приложение должно обеспечивать хранение, накопление и предоставление всей необходимой информации об униформе, работниках цеха, получении спецодежды. С помощью него сотрудники должны будут оперативно получать ответы на запросы, затрачивая на это минимум времени.

Приложение должно быть простым в использовании, иметь удобный интерфейс и обеспечивать следующие возможности:

* авторизацию пользователей;
* доступ к окнам и функциям системы через основное меню;
* внесение информации;
* редактирование информации;
* поиск информации;
* фильтрация информации по запросам пользователя;
* сортировку информации;
* автоматический расчет стоимости;
* создание отчетов в pdf, excel, word;
* печать необходимой информации;
* работу с информационной БД.

Вся информация должна быть распределена по таблицам БД, находящимся в третьей нормальной форме.

Приложение должно состоять из окон, на которых будет располагаться вся необходимая информация.

Способ решения поставленных задач зависит от выбора средств разработки.

1.3 Проектирование базы данных

Первым шагом на пути реализации проекта будет создание БД.

В широком смысле БД – это хранилище элементов данных, называемых «записями», имеющее определенную физическую и логическую структуру, а также программный интерфейс, позволяющий пользователю взаимодействовать с сохраняемой в ней информацией. Чтобы универсальным способом извлекать из нее группы записей, обрабатывать их, изменять и удалять, требуются специальные программы, которые называются СУБД [2].

Наиболее удобной на данный момент является реляционная БД, которая представляет собой совокупность отношений, связанных между собой. Исходя из этого, будет проектироваться реляционная БД.

В теории БД, связь – это ассоциирование двух или более отношений. Существуют следующие основные виды связей:

* один-к-одному (каждой записи из одной таблицы соответствует одна запись в другой таблице);
* один-ко-многим (каждой записи из одной таблицы соответствует несколько записей другой таблицы);
* многие-к-одному (множеству записей из одной таблицы соответствует одна запись в другой таблице);
* многие-ко-многим (множеству записей из одной таблицы соответствует несколько записей в другой таблице (в явном виде в реляционных базах данных не поддерживается) [2].

Одним из важных этапов процесса проектирования БД является нормализация схем отношений. При этом каждая следующая нормальная форма обладает свойствами, в некотором смысле, лучшими, чем предыдущая, и при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

Идея нормализации заключается в следующем: каждая таблица в реляционной БД удовлетворяет условию, в соответствии с которым в позиции на пересечении каждой строки и столбца таблицы всегда находится единственное значение, и никогда не может быть множества таких значений.

Обычно выделяется следующая последовательность нормальных форм:

* первая нормальная форма (1НФ);
* вторая нормальная форма (2НФ);
* третья нормальная форма (3НФ);
* нормальная форма Бойса-Кодда (БКНФ);
* четвертая нормальная форма (4НФ);
* пятая нормальная форма, или нормальная форма проекции-соединения (5НФ) [1].

В данном дипломном проекте предметной областью является система учета выдачи спецодежды на предприятии. БД будет предназначена для хранения данных о спецодежде и ее получении работниками цеха, о работниках, цехе, должностях.

Первый этап процесса проектирования БД заключается в создании концептуальной модели данных.

Компонентами концептуальной модели являются объекты и взаимосвязи. Она служит средством общения между различными пользователями, и поэтому разрабатывается без учета особенностей физического представления данных. При проектировании концептуальной модели все усилия разработчика должны быть направлены в основном на структуризацию данных и выявление взаимосвязей между ними без рассмотрения особенностей реализации и вопросов эффективности обработки. Проектирование концептуальной модели базируется на основе анализа решаемых на этом предприятии задач по обработке данных [2].

На основе анализа предметной области и пункта 1.2 были выделены основные сущности:

* спецодежда (uniform);
* авторизация (Authorization);
* персона (Person);
* должность (Position);
* получение (Receipt);
* работники (Workers);
* цех (Workshop);
* работники цеха (WorkshopWorkers).

Затем была разработана концептуальная модель БД (рисунок 1) и логическая модель, которая отражает связи между сущностями (рисунок 2). Данная модель находится в 3НФ.

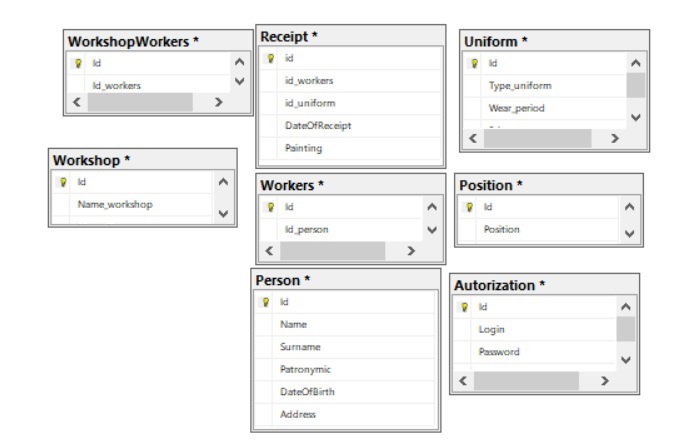


Рисунок 1 – Концептуальная модель БД

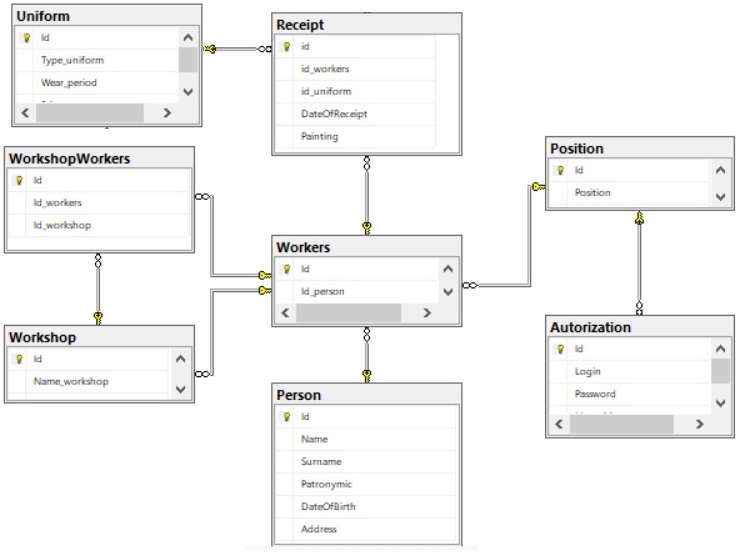


Рисунок 2 – Логическая модель БД

1.4 Проектирование функциональности программы

Для описания основных функций проектируемой программы воспользуемся методологией SADT с использованием диаграмм IDEF0.

Сначала строим самую общую, контекстную диаграмму (рисунок 3). На ней описываем процесс учета информации о спецодежде одним функциональным блоком, определяем входные и выходные данные, управляющие воздействия и механизмы.

Входные данные:

* информация из БД о спецодежде (I1);
* данные пользователя (I2);

Выходные данные:

* отображенная информация (О1);
* данные клиента в БД (О2);

Управление:

* рекомендации по оформлению интерфейса (С1);
* особенности хранения данных в СУБД (С2);

Механизмы:

* приложение (М1);
* СУБД (М2).

Субъект: приложение для учета выдачи спецодежды.

Точка зрения: процесс рассматривается с точки зрения корректного выполнения основных функций (отображения, изменения данных, формирования нужных отчетов).

Цель: рассмотрение функций системы, которые позволяют корректно отобразить и изменить данные, сформировать нужные отчеты.

Далее производим декомпозицию функционального блока А000, выделяя подфункции «Создание записей о клиентах и спецодежде» – блок А100, и «Предоставление информации о спецодежде» – блок А200 (рисунок 4).

Выполнение функций блока «Создание записей о клиентах и спецодежде» будет производиться в соответствии с методическими рекомендациями по оформлению интерфейса (С1) и особенностями хранения данных в СУБД (С2).

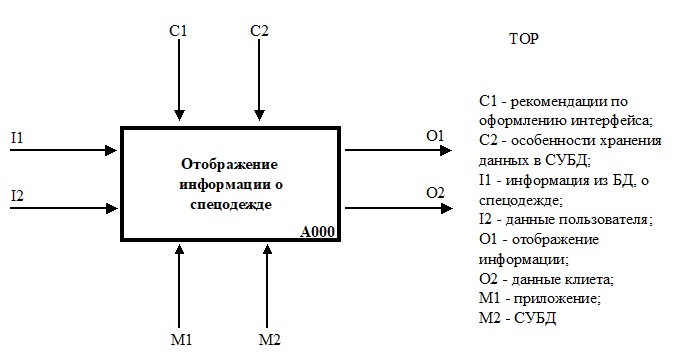


Рисунок 3 – Контекстная IDEF0-диаграмма

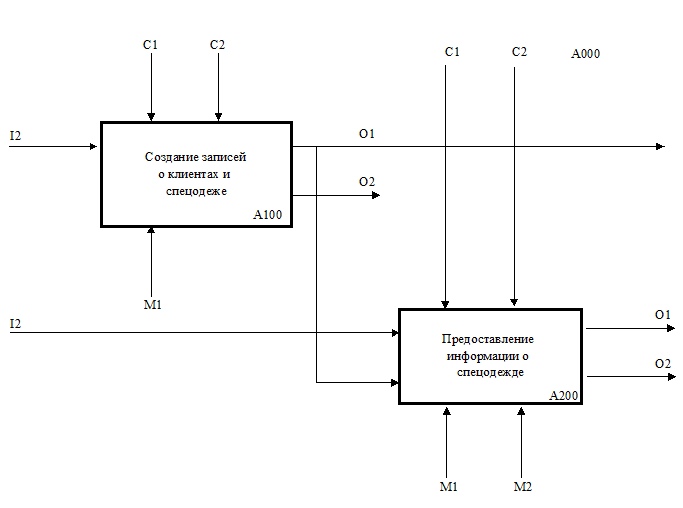


Рисунок 4 − IDEF0-диаграмма основных функций программы

Обеспечивать выполнение функций «Создание записей о клиентах и спецодежде» и «Предоставление информации о спецодежде» будут разрабатываемое приложение (М1) и СУБД (М2).

Теперь на отдельных диаграммах произведем декомпозицию функционального блока «Создание записей о клиентах и спецодежде» на блоки «Создание записи о клиенте» и «Создание записи о спецодежде» (рисунок 5).

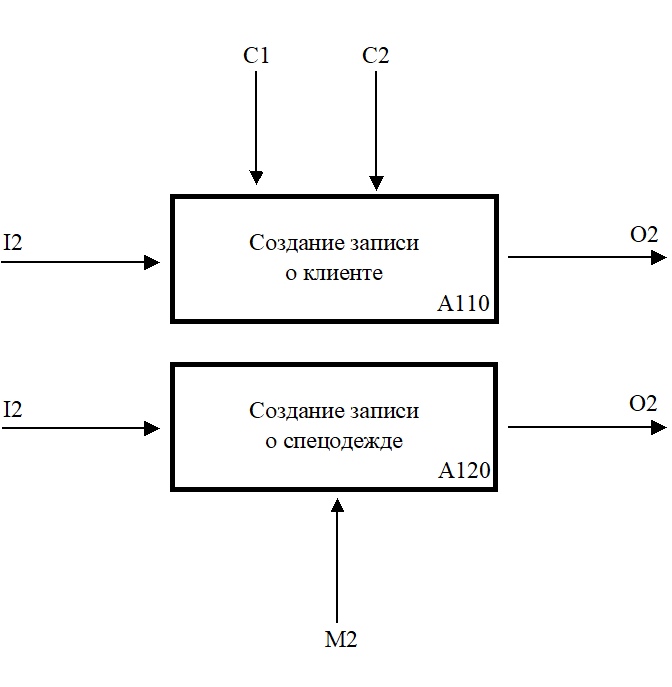


Рисунок 5 − IDEF0-диаграмма основных функций блока «Создание записей о клиентах и спецодежде»

На диаграмме А110 (рисунок 6) представлены основные функции, осуществляемые проектируемым приложением, которые позволят корректно ввести информацию о клиенте:

* «Ввод ФИО» (А111);
* «Ввод идентификатора клиента» (А112);
* «Ввод адреса клиента» (A113);
* «Ввод даты рождения клиента» (А114);
* «Сохранение в БД» (А115).

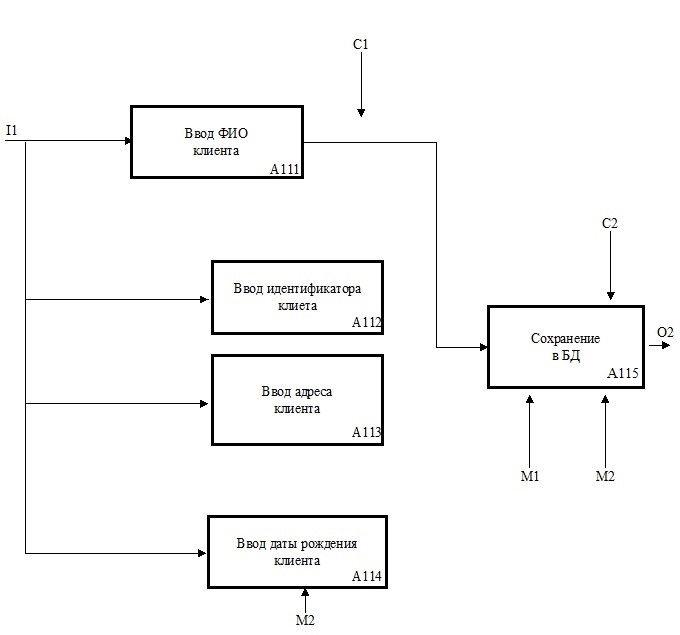


Рисунок 6 − IDEF0-диаграмма основных функций блока «Создание записи о клиенте»

На диаграмме А120 (рисунок 7) представлены основные функции, осуществляемые проектируемым приложением, которые позволят корректно ввести информацию о спецодежде:

* «Ввод идентификатора спецодежды» (А121);
* «Ввод вида спецодежды» (А122);
* «Ввод времени использования спецодежды» (A123);
* «Ввод цены спецодежды» (А124);
* «Сохранение в БД» (А125).

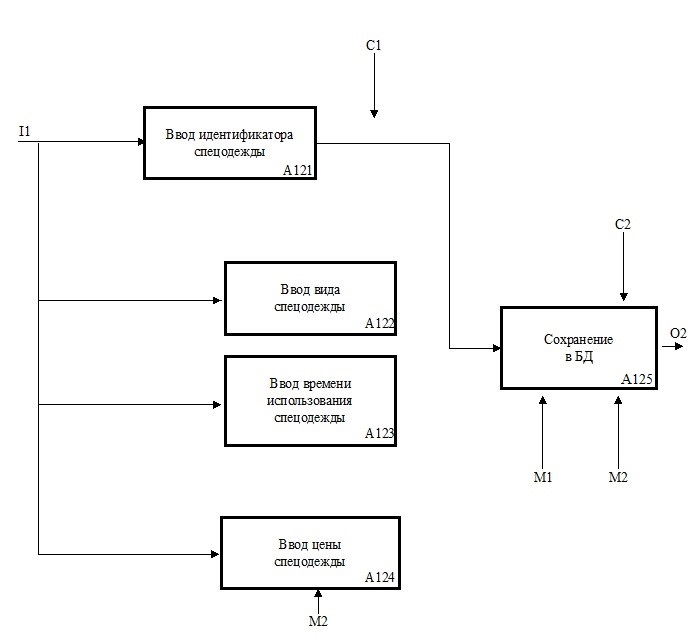


Рисунок 7 − IDEF0-диаграмма основных функций блока «Создание записи о спецодежде»

На диаграмме А200 (рисунок 8) представлены основные функции, осуществляемые проектируемым приложением, которые позволят корректно предоставить информации о спецодежде:

* «Ввод критериев поиска и фильтрации» (А210);
* «Отображение найденной информации» (А220).

На рисунке 9 представлена декомпозиция функционального блока А220, уточняющая функции разрабатываемой системы.

Полученная модель позволяет не только выделить основные функции проектируемой системы, но и демонстрирует последовательность их выполнения. Также она определяет, кто именно будет выполнять основные функции системы.

Для создания диаграмм использовалось приложение для построения структурированных диаграмм Dia.

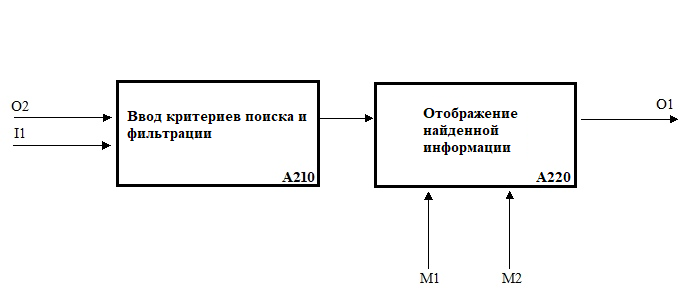


Рисунок 8 − IDEF0-диаграмма основных функций блока «Предоставление информации о спецодежде»

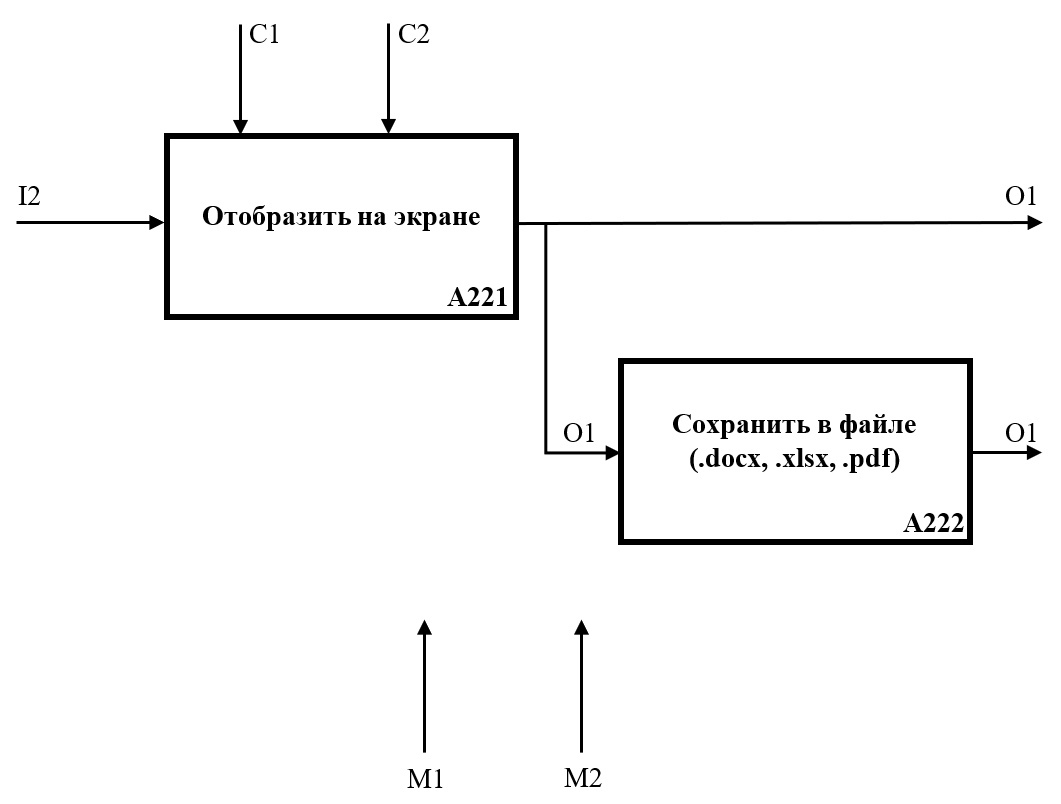


Рисунок 9 − IDEF0-диаграмма основных функций блока «Отображение найденной информации»

1.5 Проектирование структуры программы

С целью удовлетворения предъявляемым к ней требованиям, разрабатываемая программа будет состоять из ряда взаимосвязанных окон, основными из которых являются четыре, представляющие основную рабочую область:

* главное окно кладовщика, которое будет обеспечивать доступ к основным функциям приложения, позволять отображать, сортировать, составлять отчеты, добавлять, редактировать, удалять основную информацию;
* главное окно директора, которое будет обеспечивать просмотр основной информации, составлять отчеты;
* окно составления отчетов, которое будет обеспечивать возможность составления отчетов в Microsoft Word, Microsoft Excel, PDF, и печатать всю необходимую информацию;
* окно спецодежды, которое будет обеспечивать добавление, редактирование спецодежды, сортировку по цене, поиск спецодежды по неполному совпадению.

Переход между окнами будет происходить при выборе соответствующей кнопки или пункта меню.

Дополнительно будет реализовано окно авторизации для обеспечения разграничения доступа к данным и их защиты.

Также будет разработана система помощи с информацией о работе с приложением.

Проектируемая структура приложения представлена на рисунке 10.

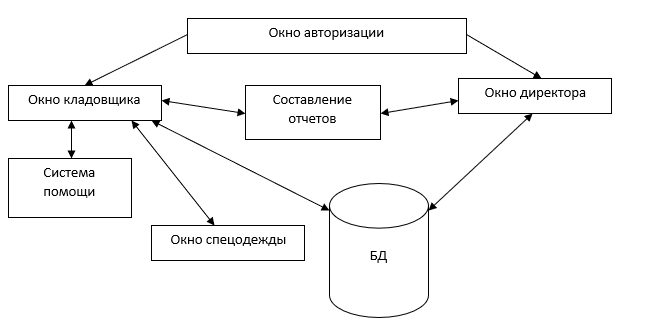


Рисунок 10 – Схема проектируемой структуры приложения

Тестирование программы планируется проводить путем выбора каждого пункта меню и проверки корректности получаемого отклика программы.

1.6 Выбор средств реализации

Программа может быть написана на следующих языках программирования: С++, С#, Python, Java. Все они позволяют работать с файлами, текстом и графикой, с использованием объектно-ориентированного подхода.

Выбор был остановлен на языке программирования C# и интегрированной среде разработки Microsoft Visual Studio 2019.

Visual Studio обеспечивает поддержку новейших языковых функций на протяжении всего процесса разработки. Данная среда разработки поддерживает C#, Visual Basic, C++, TypeScript, F# и даже сторонние языки, например, JavaScript [3].

Язык С# актуален потому, что:

* позволяет более рационально создавать популярные на сегодня типы приложений;
* интегрировал в себе преимущества языков Java и С++, при этом в объединённом языке исключены некоторые спорные директивы, макросы, отменены глобальные переменные;
* является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
* является мощным объектным языком с возможностями наследования и универсализации;
* мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из БД и других хранилищ данных [4].

Для проектирования физической модели БД будет использоваться СУБД Microsoft SQL Server Express 2017 и утилита SQL Server Management Studio 2018. Данная СУБД является высокоэффективной и надежной, в ней есть возможности для использования ее в любых критичных бизнес-приложениях. Современная версия SQL Server на равных конкурирует с СУБД таких производителей, как Oracle, IBM, и Sybase.

Основные преимущества SQL Server:

* многопоточность, поддержка нескольких одновременных запросов;
* оптимизация связей с присоединением многих данных за один проход;
* работа с записями фиксированной и переменной длины;
* поддержка ODBC-драйвера;
* поддержка гибкой системы привилегий и паролей;
* гибкая поддержка форматов чисел, строк переменной длины и меток времени;
* интерфейс с языками C и Perl, PHP;
* быстрая работа, масштабируемость;
* совместимость с ANSI SQL;
* бесплатна в большинстве случаев;
* хорошая поддержка со стороны провайдеров услуг хостинга;
* быстрая поддержка транзакций через механизм InnoDB;
* удобный графический пользовательский интерфейс и доступ к функциям через SQL Server Management Studio [5].

Для реализации справочной системы был выбран HTML Help WorkShop. Этот инструмент имеет легкий для использования и интуитивно понятный графический интерфейс, с помощью которого можно создавать проекты, гипертекстовые разделы, файлы содержаний, заголовочные файлы, а также прочие элементы, которые могут понадобиться для последующего использования в справочной системе [7].

Для создания инсталляции было выбрано средство Smart InstallMaker 5.04. Оно предоставляет собой среду для настройки проекта, создания сценария и конечного релиза дистрибутива – как в MSI, так и в различных виртуальных форматах. Это одно из наиболее известных решений в области создания инсталляторов для платформы Windows.

Инсталлятор Smart InstallMaker является программным обеспечением, которое может:

* непосредственно конвертировать виртуальные пакеты MSIS в Microsoft App-V;
* поддерживает все современные технологии – Windows и мультиплатформенный;
* идеально подходит для начинающих и опытных разработчиков;
* поддерживает гибкие и распределенные среды разработки и установки.

Утилита предоставляет мощный алгоритм для сжатия и создания компактных инсталляторов с маленьким размером, поддерживает создание многоязычного инсталлятора (до 25 языков), а также предоставляет средства для управления внешним видом программы-установки. Среди прочих возможностей Smart InstallMaker, можно выделить создание деинсталлятора, ярлыков в указанных директориях системы, регистрацию новых шрифтов или расширений в операционной системе, поддержку ActiveX и переменных, проверку установленного .NET Framework, разделение инсталлятора на установочные диски, детальную настройку информации об установочном файле, системных требований, а также его внешнего вида и многое другое [8].

1.7 Требования к программному обеспечению и техническим средствам

Программное обеспечение должно удовлетворять всем реализуемым функциям системы, а также иметь определенный набор средств для организации всех требуемых процессов обработки данных, которые позволяют своевременно выполнять все функции во всех режимах функционирования системы.

Системное программное обеспечение должно реализовывать комплекс задач управления. Для корректной работы разрабатываемой программы необходимо, чтобы компьютер с целью удовлетворения минимальным требованиям был оснащен:

* операционной системой Windows7 / 8 / 10;
* сервером SQL Server Express 2017;
* NetFramework 4.5+.

При выборе технических средств, применяемых для функционирования разрабатываемой программы, должны учитываться следующие требования:

* выбор технических средств должен обеспечивать рациональное соотношение между затратами на создание системы и достигаемым эффектом;
* технические параметры системы управления не должны налагать ограничения на регламент технологического процесса функционирования системы.

Для реализации вышеуказанных требований необходим следующий состав технических средств:

* центральный процессор фирмы INTEL или AMD с тактовой частотой не менее 1 ГГц;
* оперативная память не менее 2 ГБ;
* дисплей с разрешением не менее 1024x768 точек;
* 130 МБ свободного места на диске;
* клавиатура;
* манипулятор типа «мышь».

2 РАБОЧИЙ ПРОЕКТ

* 1. Физическая модель данных

Физическая модель данных описывает данные средствами конкретной СУБД. Ограничения, имеющиеся в логической модели данных, реализуются различными средствами СУБД, например, при помощи индексов, декларативных ограничений целостности, триггеров, хранимых процедур. При этом опять-таки решения, принятые на уровне логического моделирования определяют некоторые границы, в пределах которых можно развивать физическую модель данных. Точно также, в пределах этих границ можно принимать различные решения. Например, отношения, содержащиеся в логической модели данных, должны быть преобразованы в таблицы, но для каждой таблицы можно дополнительно объявить различные индексы, повышающие скорость обращения к данным. Многое тут зависит от конкретной СУБД.

Если физическая модель данных реализована средствами реляционной СУБД, то отношения, разработанные на стадии формирования логической модели данных, преобразуются в таблицы, атрибуты становятся столбцами таблиц, для ключевых атрибутов создаются уникальные индексы, домены преображаются в типы данных, принятые в конкретной СУБД.

На основе логической модели, представленной в пункте 1.3, была создана физическая модель БД. Она описывает то, как данные хранятся в компьютере, представляя информацию о структуре записей, их упорядоченности и существующих путях доступа. Также описываются типы, идентификаторы и разрядность полей.

Физическая модель разработанной БД состоит из следующих таблиц:

* Receipt (таблица 1);
* Uniform (таблица 2);
* Workers (таблица 3);
* WorkshopWorkers (таблица 4);
* Workshop (таблица 5);
* Position (таблица 6);
* Autorization (таблица 7);
* Person (таблица 8).

Таблица 1 – Структура таблицы Receipt (Получение)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Длина | Ключ |
| Id | Код получения | int | 100 | PK |
| Id\_workers | Код работника | int | 100 | FK |
| Id\_uniform | Код спецодежды | int | 100 | FK |
| DateOfReceipt | Дата получения | date | - | - |

Таблица 2 – Структура таблицы Uniform (Спецодежда)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Длина | Ключ |
| Id | Код спецодежды | int | 100 | PK |
| Type\_uniform | Вид спецодежды | string | 50 | - |
| Wear\_period | Время использования | int | 100 | - |
| Price | Цена | int | 200 | - |

Таблица 3 – Структура таблицы Workers (Работники)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Длина | Ключ |
| Id | Код работника | int | 100 | PK |
| Id\_person | Код человека | int | 100 | FK |
| Id\_position | Код должности | int | 100 | FK |

Таблица 4 – Структура таблицы WorkshopWorkers (Работники цеха)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Длина | Ключ |
| Id | Код работника цеха | int | 100 | PK |
| Id\_ workers | Код работника | int | 100 | FK |
| Id\_ workshop | Код цеха | int | 100 | FK |

Таблица 5 – Структура таблицы Workshop (Цех)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Длина | Ключ |
| Id | Код цеха | int | 100 | PK |
| Name\_workshop | Имя цеха | string | 40 | - |

Таблица 6 – Структура таблицы Position (Должность)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Длина | Ключ |
| Id | Код должности | int | 100 | PK |
| Position | Должность | string | 40 | - |
| Discount | Скидка | int | 100 | - |

Таблица 7 – Структура таблицы Autorization (Авторизация)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Длина | Ключ |
| Id | Код авторизации | int | 100 | PK |
| Login | Логин | string | 255 | - |
| Password | Пароль | string | 60 | - |
| Id\_position | Код должности | int | 100 | FK |

Таблица 8 – Структура таблицы Person (Персона)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Длина | Ключ |
| Id | Код человека | int | 100 | PK |
| Name | Имя | string | 60 | - |
| Surname | Фамилия | string | 60 | - |
| Patronymic | Отчество | string | 60 | - |
| DateOfBirth | Дата рождения | date | - | - |
| Address | Адрес | string | 255 | - |

2.2 Функциональное взаимодействие модулей программы

На основе технического проекта была разработана и создана программа «Спецодежда». Для создания приложения была использована среда программирования Visual Studio 2019 и язык программирования С#. В программе используются следующие языковые модули:

* System − содержит фундаментальные и базовые классы, определяющие часто используемые типы значений и ссылочных данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки;
* System.Diagnostics − содержит типы, позволяющие взаимодействовать с системными процессами, журналами событий и счётчиками производительности;
* System.Windows − содержит типы, используемые в приложениях Windows Presentation Foundation (WPF), включая клиенты анимации, элементы управления пользовательского интерфейса, привязку данных и преобразование типов;
* CefSharp − содержит типы, поддерживающие компиляцию и создание кода на языке C#, а также поддерживающие взаимодействие между средой DLR и языком C#;
* System.IO − cодержит типы, поддерживающие ввод и вывод, включая возможности чтения и записи данных в потоках как синхронно, так и асинхронно, реализации пользовательского ведения журнала и обработки входящих и исходящих потоков данных в последовательных портах [7];
* iTextSharp.text – позволяет динамически создавать и манипулировать PDF документами [10];
* Microsoft.Office.Interop.Excel – позволяет работать с файлами \*.xls и \*.xlsx, используется для загрузки данных из файлов MS Excel 2003 в базу. Также данное средство может извлекать данные из файлов с расширением \*.xlsx (формат Microsoft Office Open XML) [9].

Также были созданы следующие пользовательские модули:

* Authorization.cs – реализует окно авторизации пользователя программы;
* BarcodeForm.cs – реализует окно создания штрихкодов;
* QRCode.cs – реализует окно создания QR кодов;
* DirectorsForm.cs – реализует окно директора, в котором есть возможность просматривать данные, создавать отчеты, штрихкоды и QR коды;
* Otchet.cs, Reports.cs – окна добавления, просмотра и редактирования данных отчетов;
* WorkersForm.cs – реализует главное окно кладовщика;
* Uniform.cs – реализует окно просмотра информации о спецодежде;
* Receipt.cs – реализует окно для работы с информацией о виде спецодежды;
* ReceiptUniform1 – реализует окно вывода информации о виде спецодежды;
* Spravka.chm – реализует окно со справочной информацией о приложении.

Схема функционального взаимодействия модулей программы представлена на рисунке 11.

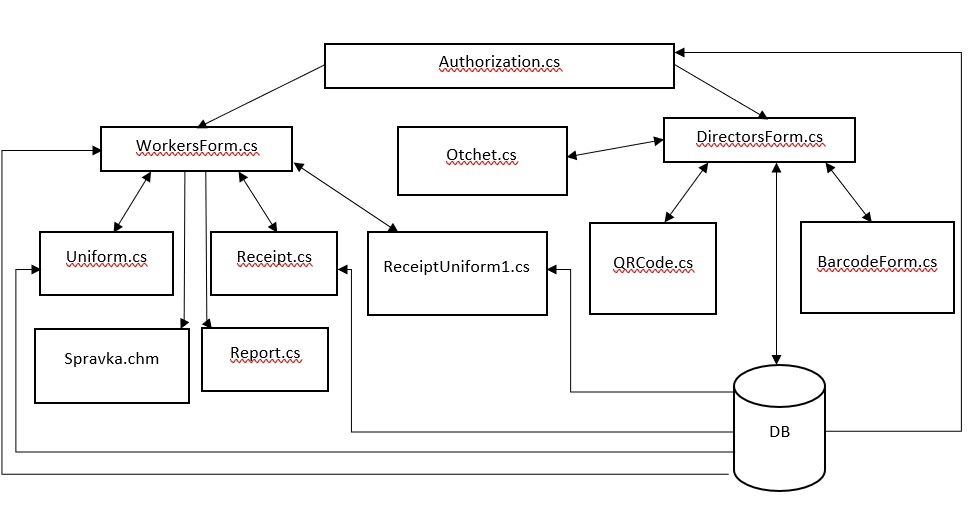


Рисунок 11 – Схема функционального взаимодействия модулей

2.3 Входные и выходные данные

Входными данными в модуле Uniform.cs являются данные из таблицы «Спецодежда».

Выходными данными для модуля Uniform.cs является информация, вводимая в различные компоненты на главной форме. Изменения автоматически сохраняются в соответствующих полях таблицы «Спецодежда» в БД.

Входными данными в модуле DirectorsForm.cs являются данные из таблиц «Спецодежда», «Работники», «Получение», «Цеха».

Выходными данными для модуля DirectorsForm.cs являются отчеты, которые можно сформировать и сохранить в файлах docx, pdf и xlsx.

Входными данными в модулях Otchet.cs и Report.cs являются данные из таблиц «Спецодежда», «Работники», «Получение», «Цеха».

Выходными данными для модулей Otchet.cs и Report.cs является информация, вводимая в различные компоненты форм добавления и редактирования отчетов.

Входными данными в модуле AddReport.cs являются данные из таблицы «Report».

Выходными данными для модуля AddReport.cs является информация, вводимая в различные компоненты форм добавления данных.

Входными данными в модуле FormQRCode.cs являются данные из поля для ввода текста.

Выходными данными для модуля FormQRCode.cs является сформированный QR-код.

Входными данными в модуле BarcodeForm.cs являются данные из поля для ввода текста.

Выходными данными для модуля BarcodeForm.cs является сформированный штрих-код.

При авторизации пользователя входные и выходные данные хранятся в следующих переменных:

* string Login – содержит введенный логин пользователя при регистрации;
* string Password – содержит введенный пароль пользователя при регистрации;
* string LoginTextBox.Text – содержит введенный логин пользователя при авторизации;
* string PasswordTextBox.Text – содержит введенный пароль пользователя при авторизации.

2.4 Установка и настройка программы

Программа устанавливается на компьютер при запуске файла «Setup.exe». При установке программы происходит создание папки «Спецодежда» в папке программ системы и осуществляется копирование файлов необходимых для запуска и корректной работы программы.

В каталоге «Справка» расположены следующие файлы:

* «Справка.chm» – система помощи, вызываемая приложением;
* «О программе.htm» – файл с гипертекстом, используемый системой помощи.

Для корректной работы приложения на компьютере также должны быть установлены Microsoft SQL Server 2017 и база данных uniform.mdf в папке C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA.

Программу можно удалить, выбрав пункт меню «Деинсталлировать» в меню Пуск/Программы или запустив файл uninstall.exe в папке программы.

Для нормальной работы программы необходимо:

* 130 МБ свободного места на жестком диске;
* операционная система Windows XP/Vista/7/8/10;
* оперативная память не менее 2 ГБ;
* разрешение экрана не менее 1024х768 точек;
* манипулятор «мышь» и клавиатура.

Для создания инсталляции была использована программа Smart Install Maker 5.04. В ходе инсталляции пользователю показываются следующие окна:

* приветствие – в нем предлагается закрыть все запущенные приложения;
* краткая информация об устанавливаемом приложении – назначение, минимальные требования к программному обеспечению и техническим средствам;
* выбор папки установки;
* создание дополнительных ярлык – позволяет создать ярлык на рабочем столе или отказаться от создания ярлыков;
* ход выполнения процесса установки;
* завершение установки – позволяет выбрать файлы, которые нужно запустить после установки.

2.5 Работа с программой

Для запуска программы нужно нажать на иконку с названием «Uniform.exe», двойным кликом. После запуска программы, открывается окно авторизации приложения, в котором необходимо указать логин и пароль для входа в программу (рисунок 12).

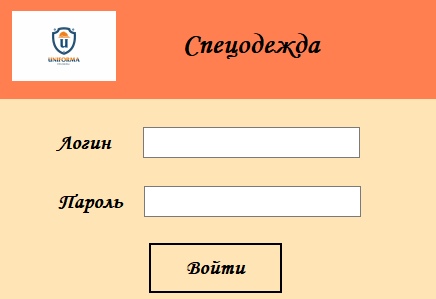
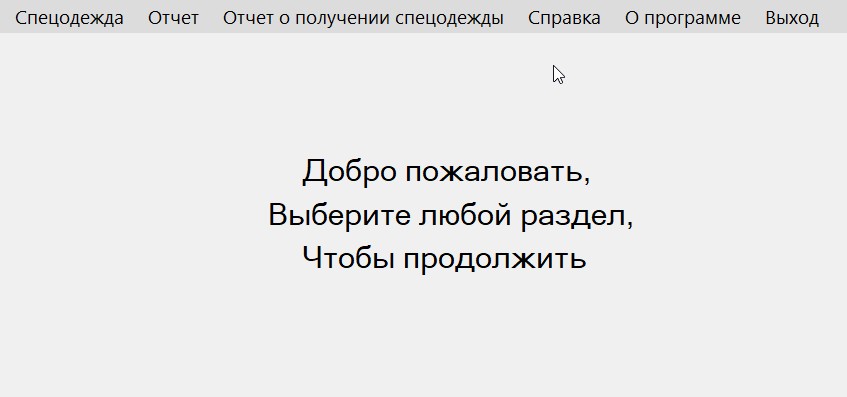


Рисунок 12 – Окно авторизации

После авторизации под логином и паролем кладовщика, открывается главное окно кладовщика (рисунок 13).

  
Рисунок 13 – Главное окно кладовщика

При выборе пункта меню «Спецодежда», открывается новое окно с информацией о спецодежде (рисунок 14).

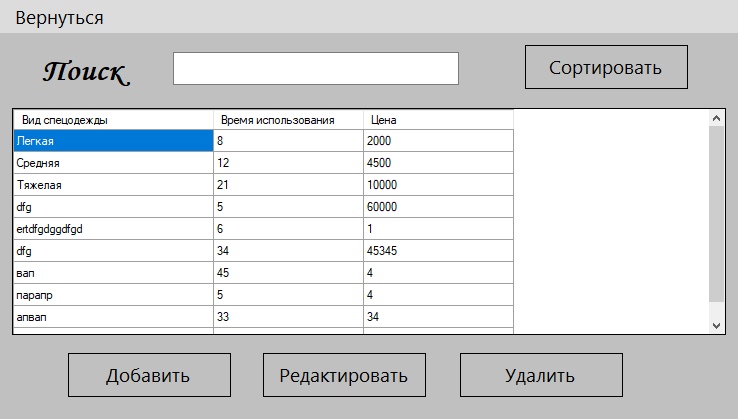


Рисунок 14 – Окно информации о спецодежде

Также на каждой форме есть возможность поиска нужной информации (рисунок 15).

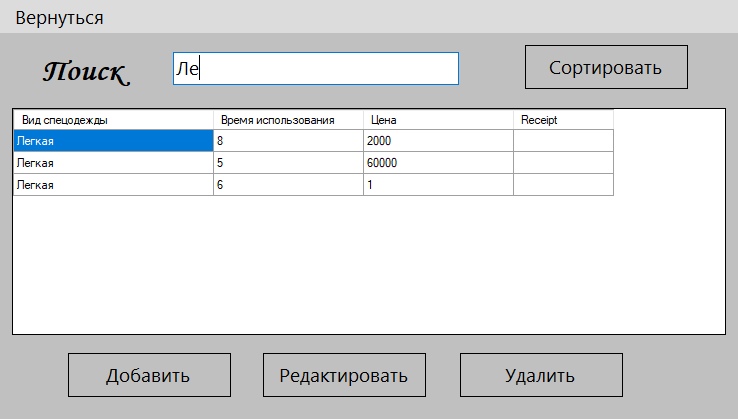


Рисунок 15 – Поиск информации

Нажатие на кнопку «Сортировать» приведет к расположению записей в порядке возрастания цены (рисунок 16).

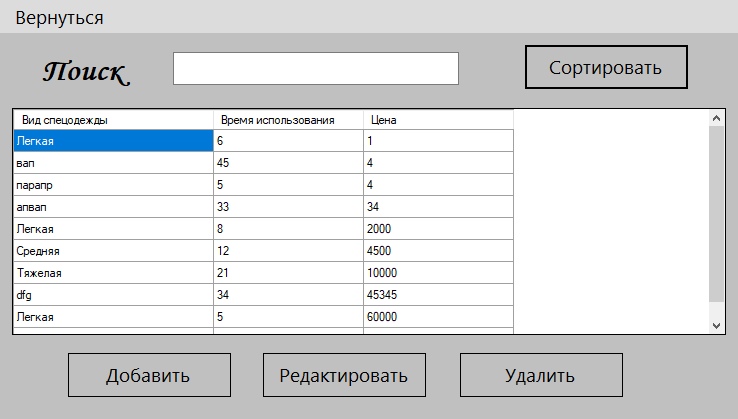


Рисунок 16 – Сортировка по возрастанию цены

После нажатия на кнопку «Добавить» откроется окно добавления информации о спецодежде (рисунок 17). При вводе осуществляется автоматическая проверка корректности значений. Все поля являются обязательными для заполнения.

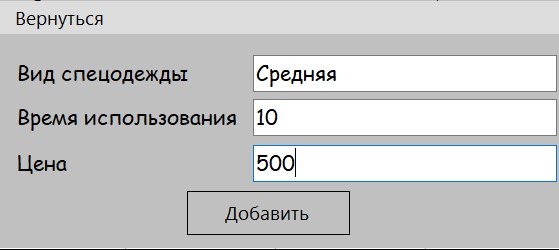


Рисунок 17 –Добавление спецодежды

После нажатия на кнопку «Редактировать» откроется окно редактирования спецодежды (рисунок 18). Здесь также осуществляется автоматическая проверка корректности значений. Все поля являются обязательными для заполнения.

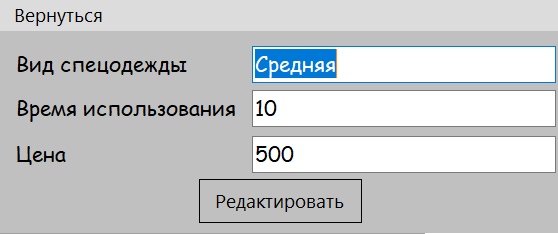


Рисунок 18 –Редактирование спецодежды

Если в главном окне кладовщика выбрать пункт меню «Отчёт», то откроется окно со сводной информацией о выдаче спецодежды (рисунок 19).

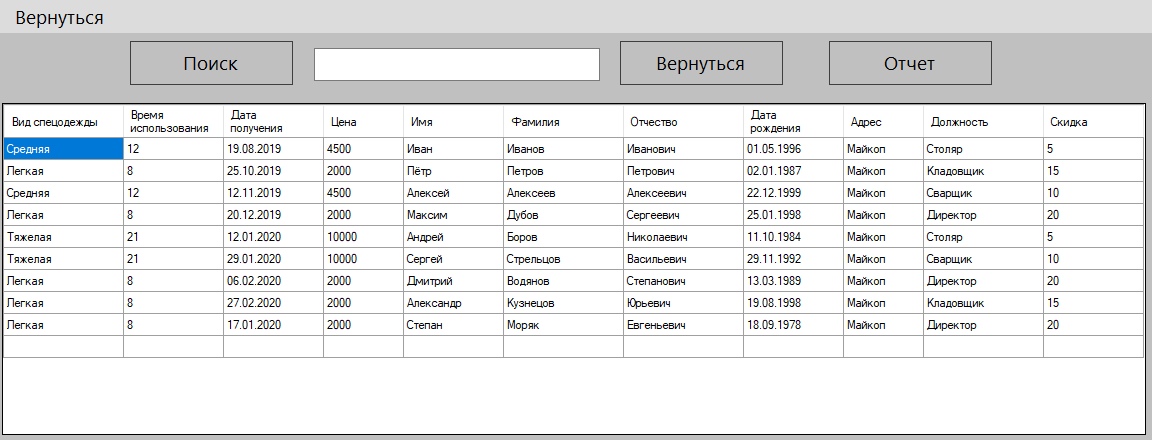


Рисунок 19 – Окно отчёта

В окне отчёта реализован поиск информации (рисунок 20).

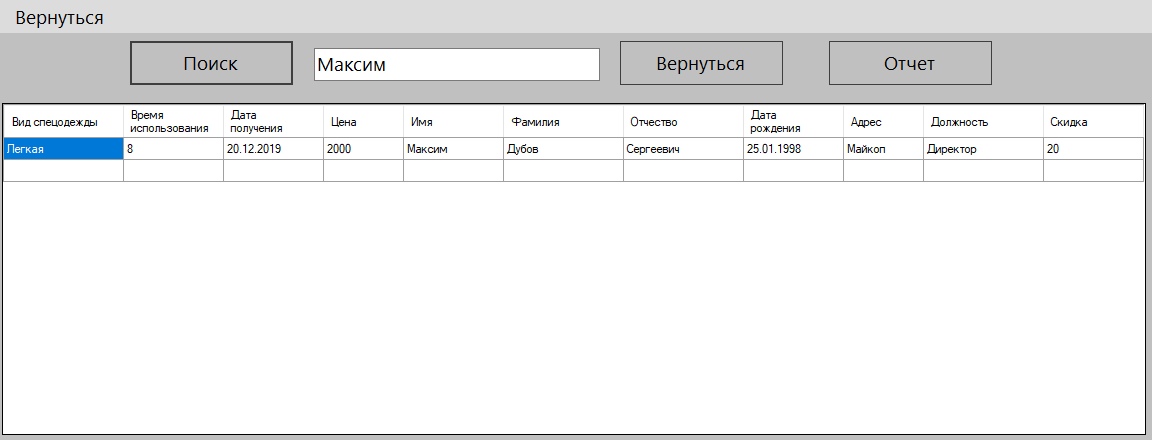


Рисунок 20 – Поиск информации

Если в окне отчета нажать на кнопку «Отчёт», то откроется программа Microsoft Excel, в которой будет отображена вся информация из таблицы (рисунок 21).

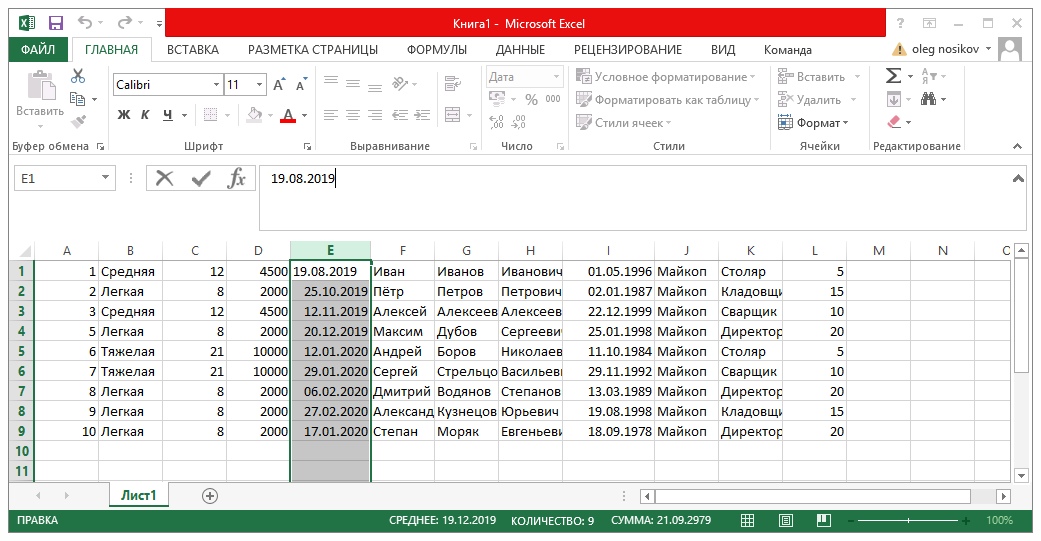


Рисунок 21 – Отчёт

При выборе в главном окне кладовщика пункта меню «Справка» откроется окно справочной системы, в котором будет информация о работе с программой (рисунок 22).

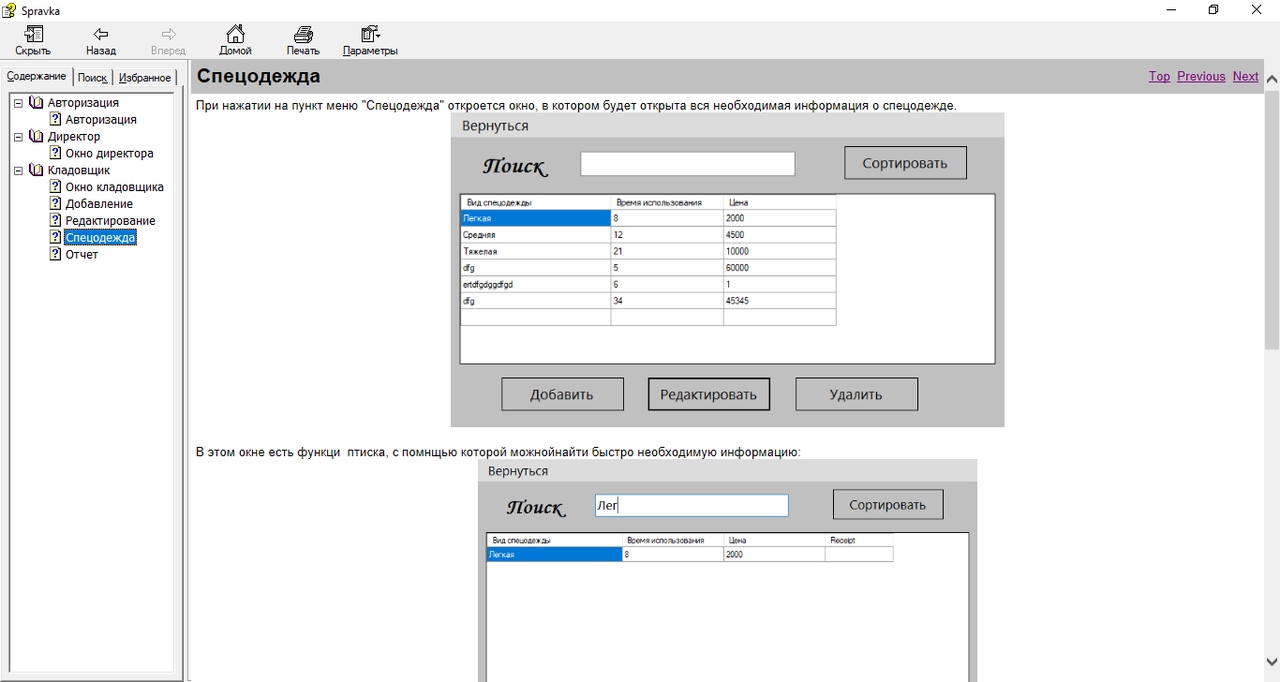


Рисунок 22 – Окно справочной системы

После авторизации под логином и паролем директора, открывается главное окно директора, на котором расположена информация о спецодежде (рисунок 23).

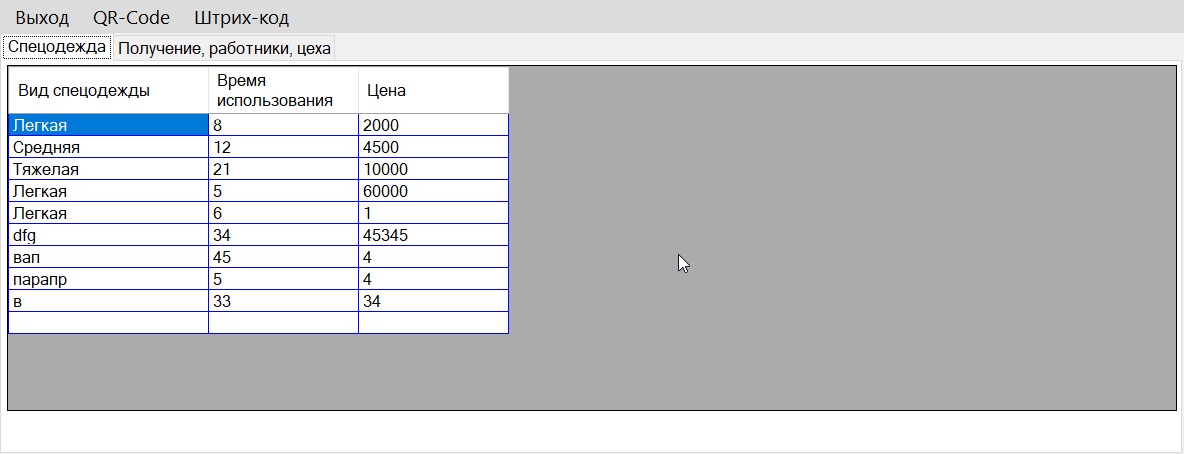


Рисунок 23 – Главное окно директора

При выборе пункта меню «Получение, работники, цеха» появится сводная информация о получении спецодежды работниками каждого цеха (рисунок 24).

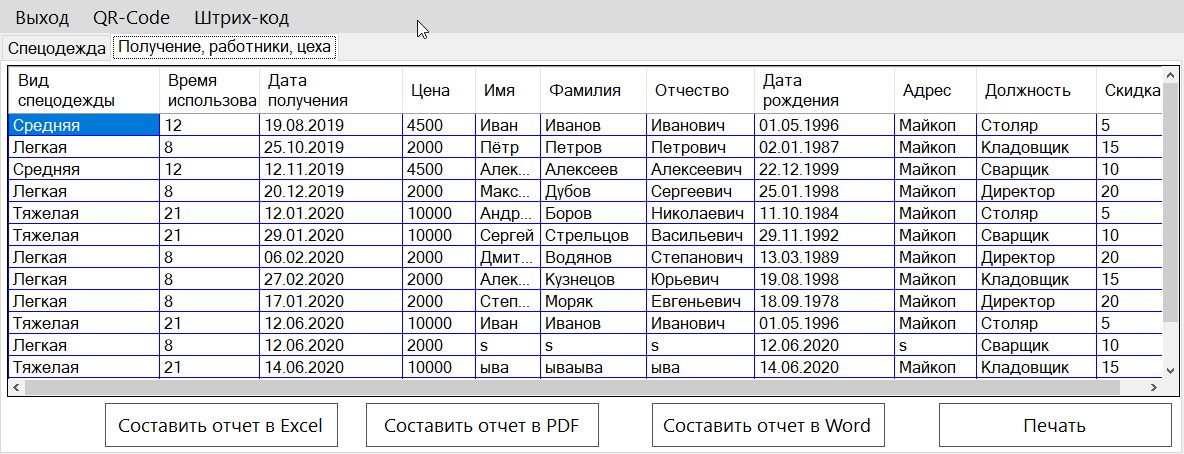


Рисунок 24 –Получение, работники, цеха

При нажатии на кнопку «Составить отчёт в Excel» откроется программа Microsoft Excel, в которой будет выведена вся информация из таблицы (рисунок 25).

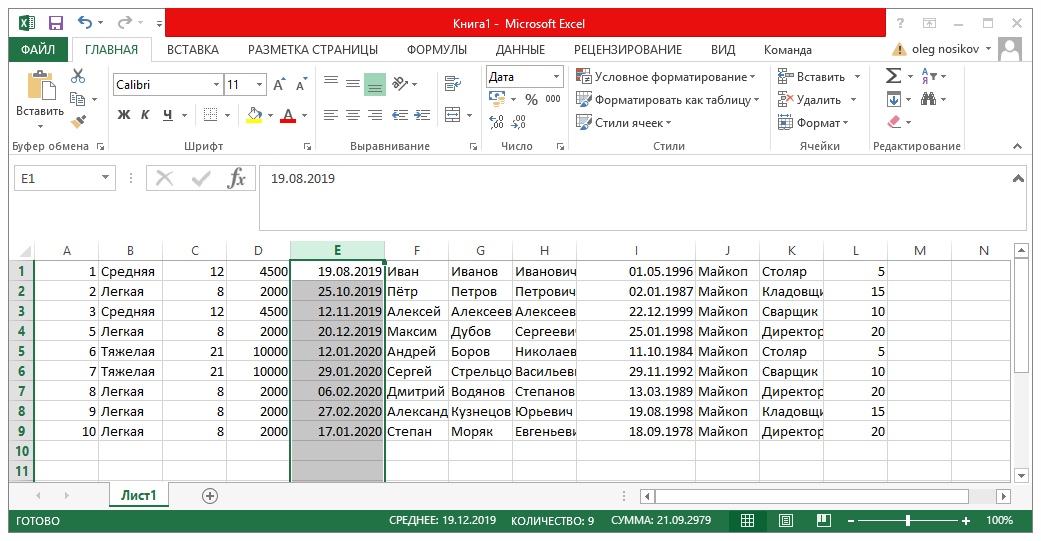


Рисунок 25 –Отчёт Excel

При нажатии на кнопку «Составить отчёт в PDF» отчёт сохранится в формате PDF в папке программы (рисунок 26).

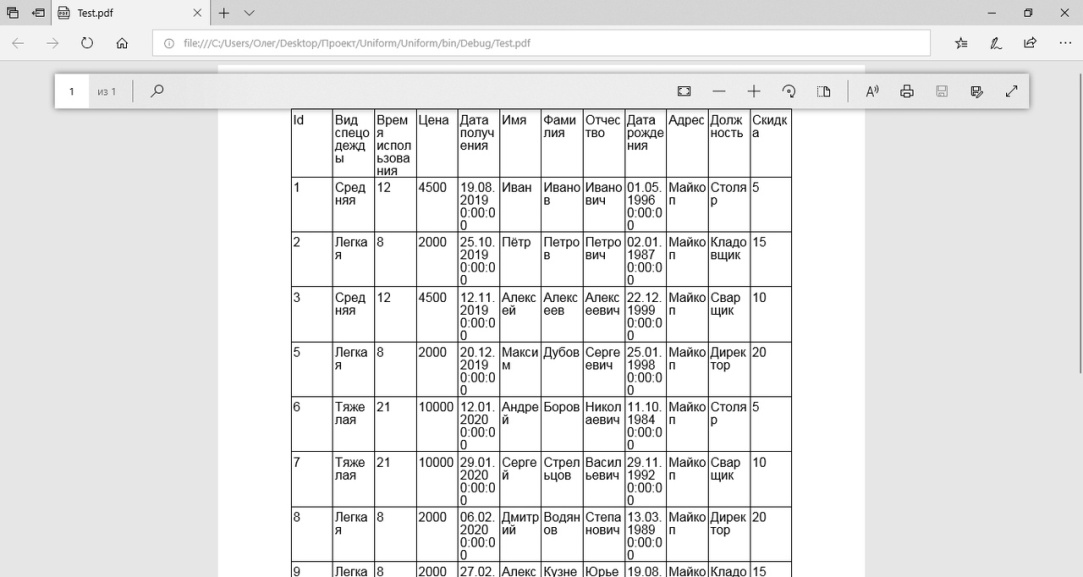


Рисунок 26 –Отчёт PDF

При нажатии на кнопку «Составить отчёт в Word» отчёт сохранится в новом документе Word в папке программы.

При нажатии на кнопку «Составить отчёт в Word» откроется программа Microsoft Word, в которой будет выведена вся информация из таблицы (рисунок 27).

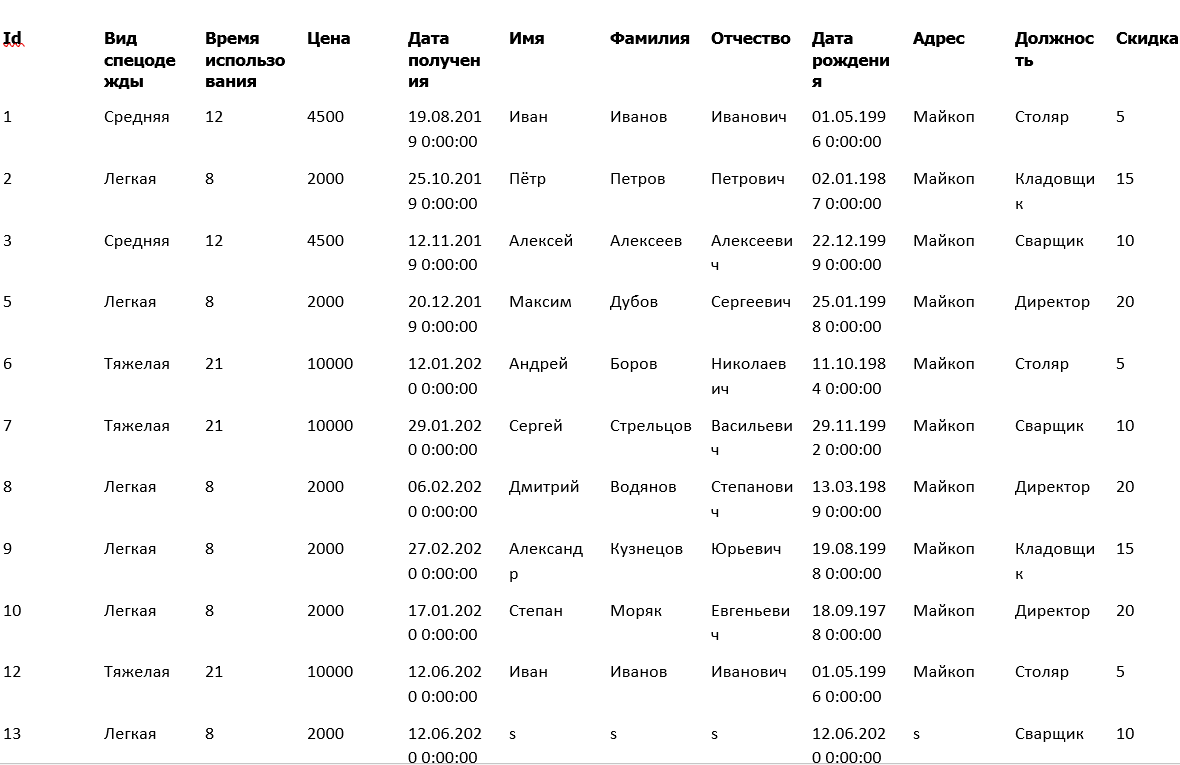


Рисунок 27 –Отчёт Word

Отчет также можно распечатать на принтере, нажав на кнопку «Печать».

При нажатии на пункт меню «QR-code» откроется окно создания QR-кода, в котором можно создавать, загружать, сохранять и напечатать QR-код (рисунок 28).

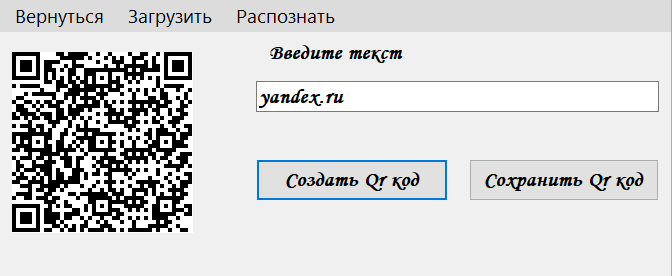


Рисунок 28 – QR-код

При нажатии на пункт меню «Штрих-код» откроется окно создания штрих-кода, в котором можно создавать, загружать, сохранять и напечатать штрих-код (рисунок 29).

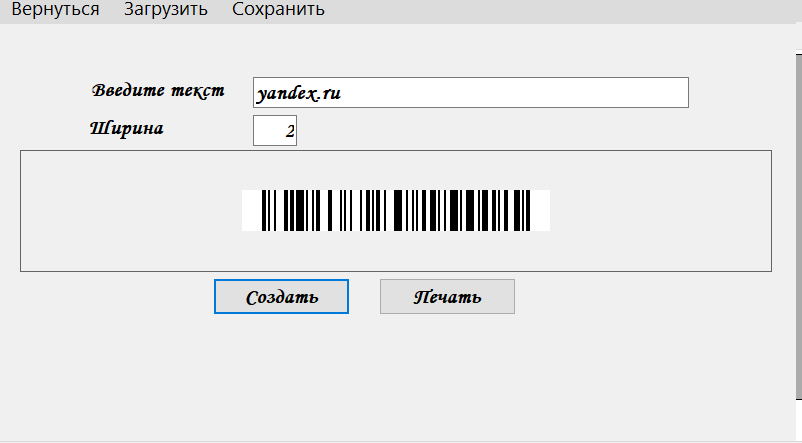


Рисунок 29 – Штрих-код

С каждой формы можно вернуться на предыдущую, выбрав пункт «Вернуться».

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была разработана БД для хранения информации о выдаче спецодежды и приложение для работы с ней. Также было разработано документальное сопровождение программы, включающее в себя технический и рабочий проекты.

В ходе выполнения дипломного проекта была достигнута поставленная цель и решены следующие задачи:

* выполнен анализ предметной области и спроектированы концептуальная и логическая модели БД для хранения необходимой информации;
* спроектирована функциональность и структура приложения, работающего с созданной БД;
* произведен выбор средств реализации БД, приложения, справочной системы и инсталляции;
* определены минимальные требования к техническим и программным средствам;
* создана физическая модель БД с помощью СУБД Microsoft SQL Server и утилиты SQL Server Management Studio;
* с помощью средств интегрированной среды разработки Visual Studio 2019 и языка программирования С# был разработан интерфейс приложения, позволяющий получить доступ ко всем необходимым функциям;
* с помощью средств интегрированной среды разработки Visual Studio 2019 и языка программирования С# были разработаны модули, обеспечивающие выполнение всех необходимых функций;
* создана пользовательская система помощи с помощью средств HTML Help WorkShop;
* создана инсталляция с помощью средств InstallMaker 5.04.

Полученные в результате анализа предметной области данные были распределены по таблицам, которые логически были связаны между собой. Основываясь на логической связи таблиц, строились запросы к БД.

Разработанная БД находится в 3НФ, что позволяет быстро и эффективно работать с информацией данной предметной области.

Для доступа к БД были использованы компоненты среды программирования C#, реализующие технологию ADO (Entity Framework).

Удобный интерфейс программы, с одной стороны, позволяет легко ориентироваться в приложении, не требуя от пользователя каких-либо специальных навыков работы, с другой стороны – предоставляет пользователю оперативную информацию.

Разработанное приложение обеспечивает следующие возможности:

* авторизацию пользователей;
* доступ к окнам и функциям системы через основное меню;
* внесение информации;
* редактирование информации;
* поиск информации;
* фильтрация информации по запросам пользователя;
* сортировку информации;
* автоматический расчет стоимости;
* создание отчетов в pdf, excel, word;
* печать необходимой информации;
* работу с информационной БД.

Основные достоинства разработанного приложения:

* низкая требовательность к ресурсам системы;
* простота установки и настройки;
* удобство эксплуатации;
* возможность оперативного получения необходимых отчетов;
* возможность сохранения отчетов в виде файлов на компьютере в различных форматах (docx, pdf, xlsx).

К недостаткам данного приложения можно отнести отсутствие возможности выбора информации для включения в отчет. В будущем это планируется исправить.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Карпова Т.С. Базы данных: модели, разработка, реализация. СПб.: Питер, 2016, 304 с.
2. Крёнке Д. Теория и практика построения баз данных. 8-е изд. СПб.: Питер, 2017. 800 с.
3. Павловская Т. А. Программирование на языке высокого уровня С#. Учебник для вузов. СПб.: Питер, 2018. 432 с.
4. Рихтер Д. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#. СПб.: Питер, 2017. 407 с.
5. Хомоненко А. Д. Базы данных: учебник для ВУЗов. М.: Корона-Принт, 2017. 736 с.
6. Шилдт Г. C# 4.0: полное руководство. Пер. с англ. М.: Вильямс, 2018. 1056 с.
7. Официальный сайт HTML HelpWorkshop. URL: https://www.microsoft. com (дата обращения: 18.05.2021).
8. Официальный сайт Smart Install Maker. URL: http://ru.sminstall.com (дата обращения 18.05.2021).
9. Перенос данных из DataGridView в Microsoft Excel. URL: https://www.cyberforum.ru/windows-forms/thread166299.html (дата обращения 28.04.2021);
10. Работа с библиотекой ItextSharp для вывода информации. URL: http://billibook.blogspot.com/2012/08/iTextSharp.html (дата обращения 25.04.2021);
11. Работа с Entity Framework C#. URL: https://professorweb.ru/my/entity-framework/6/level1/ (дата обращения 06.04.2021);
12. Руководство по Entity. URL: https://metanit.com/sharp/entityframework/ (дата обращения 05.04.2021);
13. Спецодежда как средство индивидуальной защиты. URL: https://forma-odezhda.ru/encyclopedia/specodezhda-kak-sredstvo-individualnoj-zacshity/

ПРИЛОЖЕНИЕ А

Код основных модулей

ReportLogic.cs

public class ReportLogic : IDisposable

{

EntityContext context;

public List<Position> Position { get; set; }

public List<Uniform1> Uniforms { get; set; }

public List<ReportModels> Report { get; set; }

public ReportModels SelectedAvtor { get; set; }

public ReportLogic()

{

context = new EntityContext();

Report = new List<ReportModels>();

Position = new List<Position>();

Uniforms = new List<Uniform1>();

}

public async Task Fill()

{

await context.Receipt.LoadAsync();

foreach (var zakaz in context.Receipt.Local)

{

Report.Add(new ReportModels().Fill(zakaz));

}

}

public async Task FillPosition()

{

await context.Position.LoadAsync();

foreach (var position in context.Position.Local)

Position.Add(position);

}

public async Task FillUniform()

{

await context.Uniform.LoadAsync();

foreach (var uniform in context.Uniform.Local)

Uniforms.Add(uniform);

}

public void Dispose()

{

throw new NotImplementedException();

}

public void Add(ReportModels model)

{

context.Person.Add(new Person

{

Surname = model.Surname,

Name = model.Name,

Patronymic = model.Patronymic,

DateOfBirth = model.DateOfBirth,

Address = model.Address,

});

context.SaveChanges();

context.Position.Add(new Position

{

Position1 = model.Position,

Discount = model.Discount,

});

context.SaveChanges();

context.Workers.Add(new Workers

{

Id\_person = context.Person.Local.Last().Id,

Id\_position = model.Id\_position,

});

context.SaveChanges();

context.Receipt.Add(new Receipt

{

id\_workers = context.Workers.Local.Last().Id,

id\_uniform = model.Id\_unidorm,

DateOfReceipt = model.DateOfReceipt,

Painting = model.Painting,

});

context.SaveChanges();

}

public void Update(ReportModels model)

{

var orderForUpdate = context.Receipt.FirstOrDefault(order =>

order.id.Equals(model.Id));

orderForUpdate.id = model.Id;

orderForUpdate.id\_uniform = model.Id\_unidorm;

orderForUpdate.DateOfReceipt = model.DateOfReceipt;

orderForUpdate.Painting = model.Painting;

orderForUpdate.id\_workers = Convert.ToInt32(model.Id\_workers);

int personID = Convert.ToInt32(model.IdPerson);

Person person = context.Person.FirstOrDefault(p => p.Id.Equals(personID));

person.Surname = model.Surname;

person.Patronymic = model.Patronymic;

person.Name = model.Name;

person.Address = model.Address;

person.DateOfBirth = model.DateOfBirth;

Workers workers = context.Workers.FirstOrDefault(p =>

p.Id.Equals(model.Id));

workers.Id\_position = model.Id\_position;

workers.Id\_person = personID;

Position position = context.Position.FirstOrDefault(p =>

p.Position1.Equals(model.Position));

Uniform1 uniform = context.Uniform.FirstOrDefault(p =>

p.Type\_uniform.Equals(model.Type\_uniform));

context.SaveChanges();

}

public void Delete(ReportModels model)

{

Person person = context.Person.FirstOrDefault(p =>

p.Id.Equals(model.Id));

Workers workers = context.Workers.FirstOrDefault(p =>

p.Id.Equals(model.Id));

Position position = context.Position.FirstOrDefault(p =>

p.Position1.Equals(model.Position));

Uniform1 uniform = context.Uniform.FirstOrDefault(p =>

p.Type\_uniform.Equals(model.Type\_uniform));

Receipt receipt = context.Receipt.FirstOrDefault(p =>

p.id.Equals(model.Id));

context.Person.Remove(person);

context.Workers.Remove(workers);

context.Position.Remove(position);

context.Uniform.Remove(uniform);

context.Receipt.Remove(receipt);

context.SaveChanges();

}

}

}

ReportLogicDirector.cs

class ReportLogicDirector : IDisposable

{

EntityContext context;

public List<Position> Position { get; set; }

public List<Uniform1> Uniforms { get; set; }

//public List<Uniform> Uniforms { get; set; }

public List<ReportModelDirector> Report { get; set; }

public ReportModels SelectedAvtor { get; set; }

public ReportLogicDirector()

{

context = new EntityContext();

Report = new List<ReportModelDirector>();

Position = new List<Position>();

Uniforms = new List<Uniform1>();

}

public async Task Fill()

{

await context.Receipt.LoadAsync();

foreach (var zakaz in context.Receipt.Local)

{

Report.Add(new ReportModelDirector().Fill(zakaz));

}

}

public async Task FillPosition()

{

await context.Position.LoadAsync();

foreach (var position in context.Position.Local)

Position.Add(position);

}

public async Task FillUniform()

{

await context.Uniform.LoadAsync();

foreach (var uniform in context.Uniform.Local)

Uniforms.Add(uniform);

}

public void Dispose()

{

throw new NotImplementedException();

}

}

}

ReportUniform.cs

public class ReportUniform : IDisposable

{

EntityContext context;

public List<ReportUniformModel> Report { get; set; }

public ReportUniform()

{

context = new EntityContext();

Report = new List<ReportUniformModel>();

}

public async Task Fill()

{

await context.Receipt.LoadAsync();

foreach (var zakaz in context.Receipt.Local)

{

Report.Add(new ReportUniformModel().Fill(zakaz));

}

}

public void Dispose()

{

context.Dispose();

}

}

}

UniformLogic.cs

public class UniformLogic : IDisposable

{

EntityContext context;

public List<UniformModels> Uniforms { get; set; }

public UniformModels SelectedUniform { get; set; }

public UniformLogic()

{

context = new EntityContext();

Uniforms = new List<UniformModels>();

}

public async Task Fill()

{

await context.Uniform.LoadAsync();

foreach (var zakaz in context.Uniform.Local)

{

Uniforms.Add(new UniformModels().Fill(zakaz));

}

}

public void Dispose()

{

context.Dispose();

}

public void Add(UniformModels model)

{

try

{

context.Uniform.Add(new Uniform1

{

Type\_uniform = model.Type\_uniform,

Wear\_period = model.Wear\_period,

Price = model.Price,

});

context.SaveChanges();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка");

}

}

public void Update(UniformModels model)

{

Uniform1 uniform = context.Uniform.FirstOrDefault(p =>

p.Id.Equals(model.Id));

uniform.Type\_uniform = model.Type\_uniform;

uniform.Wear\_period = model.Wear\_period;

uniform.Price = model.Price;

context.SaveChanges();

}

public void Delete(UniformModels model)

{

Uniform1 uniform = context.Uniform.FirstOrDefault(p =>

p.Id.Equals(model.Id));

context.Uniform.Remove(uniform);

context.SaveChanges();

}

}

Report.cs

public partial class Report : Form

{

EntityContext context;

private ReportLogic reportLogic;

BindingList<ReportModels> bindingList;

static string connectionString = @"Data Source=

DESKTOP-ILA26Q7\SQLEXPRESS01;

Initial Catalog=Uniform;

Integrated Security=True";

public Report()

{

InitializeComponent();

reportLogic = new ReportLogic();

context = new EntityContext();

CreateBinding();

}

private void Report\_Load(object sender, EventArgs e)

{

FillDataGrid();

}

private async void FillDataGrid()

{

await reportLogic.Fill();

FillBindingList(reportLogic.Report);

dataGridView1.DataSource = new BindingSource {DataSource = bindingList};

}

private void FillBindingList(IEnumerable<ReportModels> reports)

{

foreach (var report in reports)

{

bindingList.Add(report);

}

}

public void UpdateDAtaSet()

{

if (dataGridView1.DataSource != null)

{

dataGridView1.Rows.Clear();

dataGridView1.Refresh();

reportLogic.Report = new List<ReportModels>();

bindingList = new BindingList<ReportModels>();

FillDataGrid();

}

}

private void CreateBinding()

{

try

{

bindingList = new BindingList<ReportModels>();

Id.DataPropertyName = nameof(ReportModels.Id);

Type\_uniform.DataPropertyName =

nameof(ReportModels.Type\_uniform);

Wear\_period.DataPropertyName = nameof(ReportModels.Wear\_period);

DateOfReceipt.DataPropertyName =

nameof(ReportModels.DateOfReceipt);

Price.DataPropertyName = nameof(ReportModels.Price);

Name.DataPropertyName = nameof(ReportModels.Name);

Surname.DataPropertyName = nameof(ReportModels.Surname);

Patronymic.DataPropertyName = nameof(ReportModels.Patronymic);

Address.DataPropertyName = nameof(ReportModels.Address);

DateOfBirth.DataPropertyName = nameof(ReportModels.DateOfBirth);

Position.DataPropertyName = nameof(ReportModels.Position);

Discount.DataPropertyName = nameof(ReportModels.Discount);

Id\_uniform.DataPropertyName = nameof(ReportModels.Id\_unidorm);

Id\_workers.DataPropertyName = nameof(ReportModels.Id\_workers);

Id\_position.DataPropertyName = nameof(ReportModels.Id\_position);

IdPerson.DataPropertyName = nameof(ReportModels.IdPerson);

Name\_workshop.DataPropertyName =

nameof(ReportModels.Name\_workshop);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

}

}

private void Recipe\_Click(object sender, EventArgs e)

{

this.Close();

this.Hide();

}

private void Button1\_Click(object sender, EventArgs e)

{

var col = reportLogic.Report.Where(client =>

client.Name.Equals(textBox1.Text) || client.Surname.Equals(textBox1.Text) || client.Patronymic.Equals(textBox1.Text));

dataGridView1.Rows.Clear();

dataGridView1.Refresh();

FillBindingList(col);

dataGridView1.DataSource = new BindingSource {DataSource = bindingList};

}

private void Button2\_Click(object sender, EventArgs e)

{

textBox1.Text = "";

UpdateDAtaSet();

}

private void Button3\_Click(object sender, EventArgs e)

{

string sqlExpression = "Sort";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

// указываем, что команда представляет хранимую процедуру

command.CommandType = System.Data.CommandType.StoredProcedure;

UpdateDAtaSet();

}

}

private void Button3\_Click\_2(object sender, EventArgs e)

{

Microsoft.Office.Interop.Excel.Application ExcelApp =

new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook ExcelWorkBook;

Microsoft.Office.Interop.Excel.Worksheet ExcelWorkSheet;

//Книга.

ExcelWorkBook =

ExcelApp.Workbooks.Add(System.Reflection.Missing.Value);

//Таблица.

ExcelWorkSheet =

(Microsoft.Office.Interop.Excel.Worksheet)ExcelWorkBook.

Worksheets.get\_Item(1);

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

ExcelApp.Cells[i + 1, j + 1] = dataGridView1.Rows[i].Cells[j].Value;

}

}

ExcelApp.Visible = true; //Вызываем нашу созданную эксельку.

ExcelApp.UserControl = true;

}

private void Button4\_Click\_2(object sender, EventArgs e)

{

AddReport report = new AddReport();

try

{

if (report.ShowDialog(this).Equals(DialogResult.OK))

{

if (dataGridView1.DataSource != null) UpdateDAtaSet();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private async void Button5\_Click(object sender, EventArgs e)

{

try

{

if (dataGridView1.CurrentRow != null)

{

DataGridViewRow row = dataGridView1.CurrentRow;

using (var context = new EntityContext())

{

ReportModels model = new ReportModels()

{

Name = row.Cells["Name"].Value.ToString(),

Surname = row.Cells["Surname"].Value.ToString(),

Patronymic = row.Cells["Patronymic"].Value.ToString(),

Address = row.Cells["Address"].Value.ToString(),

DateOfBirth =

DateTime.Parse(row.Cells["DateOfBirth"].Value.ToString()),

Position = row.Cells["Position"].Value.ToString(),

DateOfReceipt =

DateTime.Parse(row.Cells["DateOfReceipt"].Value.ToString()),

Painting = row.Cells["Painting"].Value.ToString(),

Type\_uniform = row.Cells["Type\_uniform"].Value.ToString(),

Id = Int32.Parse(row.Cells["Id"].Value.ToString()),

IdPerson = Int32.Parse(row.Cells["IdPerson"].Value.ToString()),

Id\_workers =

Int32.Parse(row.Cells["Id\_workers"].Value.ToString()),

Id\_position =

Int32.Parse(row.Cells["Id\_position"].Value.ToString()),

Id\_unidorm =

Int32.Parse(row.Cells["Id\_uniform"].Value.ToString()),

};

await context.Receipt.LoadAsync();

reportLogic.SelectedAvtor = model;

}

}

UpdateReceipt receipt = new UpdateReceipt(reportLogic.SelectedAvtor);

receipt.ShowDialog(this);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void Button6\_Click(object sender, EventArgs e)

{

try

{

if (dataGridView1.CurrentRow != null)

{

DataGridViewRow row = dataGridView1.CurrentRow;

ReportModels model = new ReportModels()

{

Name = row.Cells["Name"].Value.ToString(),

Surname = row.Cells["Surname"].Value.ToString(),

Patronymic = row.Cells["Patronymic"].Value.ToString(),

Address = row.Cells["Address"].Value.ToString(),

DateOfBirth =

DateTime.Parse(row.Cells["DateOfBirth"].Value.ToString()),

Position = row.Cells["Position"].Value.ToString(),

DateOfReceipt =

DateTime.Parse(row.Cells["DateOfReceipt"].Value.ToString()),

Painting = row.Cells["Painting"].Value.ToString(),

Type\_uniform = row.Cells["Type\_uniform"].Value.ToString(),

Id = Int32.Parse(row.Cells["Id"].Value.ToString()),

IdPerson = Int32.Parse(row.Cells["IdPerson"].Value.ToString()),

Id\_workers = Int32.Parse(row.Cells["Id\_workers"].Value.ToString()),

Id\_position = Int32.Parse(row.Cells["Id\_position"].Value.ToString()),

Id\_unidorm = Int32.Parse(row.Cells["Id\_uniform"].Value.ToString()),

};

reportLogic.Delete(model);

UpdateDAtaSet();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ОбновитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

UpdateDAtaSet();

}

}

}

Uniforms.cs

public partial class Uniform : Form

{

EntityContext context;

private UniformLogic uniformLogic;

BindingList<UniformModels> bindingList;

static string connectionString = @"Data Source=

DESKTOP-ILA26Q7\SQLEXPRESS01;

Initial Catalog=Uniform;Integrated Security=True";

public Uniform()

{

InitializeComponent();

uniformLogic = new UniformLogic();

context = new EntityContext();

CreateBinding();

}

private void Recipe\_Click(object sender, EventArgs e)

{

this.Hide();

this.Close();

}

private void Button1\_Click(object sender, EventArgs e)

{

AddUniform uniform = new AddUniform();

try

{

if (uniform.ShowDialog(this).Equals(DialogResult.OK))

{

if (dataGridView1.DataSource != null)

{

UpdateDAtaSet();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + " button1\_Click", "Error");

}

}

public void UpdateDAtaSet()

{

if (dataGridView1.DataSource != null)

{

dataGridView1.Rows.Clear();

dataGridView1.Refresh();

uniformLogic.Uniforms = new List<UniformModels>();

bindingList = new BindingList<UniformModels>();

FillDataGrid();

}

}

private void Uniform\_Load(object sender, EventArgs e)

{

FillDataGrid();

}

private async void FillDataGrid()

{

await uniformLogic.Fill();

FillBindingList(uniformLogic.Uniforms);

dataGridView1.DataSource = new BindingSource { DataSource = bindingList };

}

private void FillBindingList(IEnumerable<UniformModels> uniforms)

{

foreach (var uniform in uniforms)

{

bindingList.Add(uniform);

}

}

private void CreateBinding()

{

try

{

bindingList = new BindingList<UniformModels>();

Id.DataPropertyName = nameof(UniformModels.Id);

Type\_Uniform.DataPropertyName =

nameof(UniformModels.Type\_uniform);

Wear\_period.DataPropertyName = nameof(UniformModels.Wear\_period);

Price.DataPropertyName = nameof(UniformModels.Price);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error");

}

}

private async void Button2\_Click(object sender, EventArgs e)

{

try

{

if (dataGridView1.CurrentRow != null)

{

DataGridViewRow row = dataGridView1.CurrentRow;

using (var context = new EntityContext())

{

UniformModels uniform = new UniformModels()

{

Id = Int32.Parse(row.Cells["Id"].Value.ToString()),

Type\_uniform = row.Cells["Type\_uniform"].Value.ToString(),

Wear\_period =

Convert.ToInt32(row.Cells["Wear\_period"].Value.ToString()),

Price = Convert.ToDouble(row.Cells["Price"].Value.ToString()),

};

await context.Uniform.LoadAsync();

uniform.Id = context.Uniform.Local.First(c =>

c.Id.Equals(uniform.Id)).Id;

uniformLogic.SelectedUniform = uniform;

UpdateDAtaSet();

}

}

new UpdateUniform(uniformLogic.SelectedUniform).ShowDialog();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void Button3\_Click(object sender, EventArgs e)

{

try

{

if (dataGridView1.CurrentRow != null)

{

DataGridViewRow row = dataGridView1.CurrentRow;

UniformModels uniform = new UniformModels()

{

Id = Int32.Parse(row.Cells["Id"].Value.ToString()),

Type\_uniform = row.Cells["Type\_uniform"].Value.ToString(),

Wear\_period =

Convert.ToInt32(row.Cells["Wear\_period"].Value.ToString()),

Price = Convert.ToDouble(row.Cells["Price"].Value.ToString()),

};

uniformLogic.Delete(uniform);

UpdateDAtaSet();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void TextBox1\_TextChanged(object sender, EventArgs e)

{

dataGridView1.DataSource =

context.Uniform.Where(x=>

x.Type\_uniform.Contains(textBox1.Text)).ToList();

}

private void Button4\_Click(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

DataSet ds = new DataSet();

using (SqlConnection myConnection =

new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand("Sort",

myConnection))

{

SqlDataAdapter myAdapter = new SqlDataAdapter();

myAdapter.SelectCommand = command;

myAdapter.Fill(ds, "Uniform");

}

dataGridView1.DataSource = ds.Tables["Uniform"].DefaultView;

}

}

}

}