ОГЛАВЛЕНИЕ

[1 СИСТЕМНЫЙ АНАЛИЗ 5](#_Toc89287441)

[1.1 Название и цель разработки информационной системы 5](#_Toc89287442)

[1.2 Характеристика предприятия 5](#_Toc89287443)

[1.2.1 Сведения о предприятии 5](#_Toc89287444)

[1.2.2 Сведения об отношениях предприятия с действующими лицами из окружающей среды 6](#_Toc89287445)

[1.2.3 Детализация отношений 7](#_Toc89287446)

[1.2.4 Описание сценариев реализации процессов 8](#_Toc89287447)

[1.3 Сводные данные о категориях пользователей и их потребностях 11](#_Toc89287448)

[1.4 Концептуальная модель данных предметной области 14](#_Toc89287449)

[1.5 Описание специфических требований к программному продукту, опосредованно связанных с предметной областью 18](#_Toc89287450)

[2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 19](#_Toc89287451)

[2.1 Иерархия функций информационной системы 19](#_Toc89287452)

[2.2 Структура системы 20](#_Toc89287453)

[2.2.1 Структурная схема системы 20](#_Toc89287454)

[2.2.2 Описание подсистем, из которых состоит программный продукт 21](#_Toc89287455)

[2.3 Проектирование пользовательского интерфейса 22](#_Toc89287456)

[2.3.1 Схема интерфейса 22](#_Toc89287457)

[2.4 Проектирование БД 27](#_Toc89287458)

[2.4.1 Логическая модель данных 27](#_Toc89287459)

[2.4.2 Составление таблиц со списком полей логической модели. 28](#_Toc89287460)

[2.4.3 Описание работ проделанных по нормализации 30](#_Toc89287461)

[2.4.4 Описание используемых средств обеспечения целостности данных 32](#_Toc89287462)

[3 ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 33](#_Toc89287463)

[3.1 Выбор языка, среды, которые будут использоваться в ходе разработки приложения. 33](#_Toc89287464)

[3.2 Выбор СУБД 35](#_Toc89287465)

[3.3 Физическая модель данных 36](#_Toc89287466)

[Составление таблиц со списком полей физической модели 36](#_Toc89287467)

[3.4 Разработка триггеров, хранимых процедур, запросов на выборку, представлений 40](#_Toc89287468)

[ЗАКЛЮЧЕНИЕ 41](#_Toc89287469)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 43](#_Toc89287470)

[ПРИЛОЖЕНИЕ А. SQL СКРИП БАЗЫ ДАННЫХ 44](#_Toc89287471)

[ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММЫ 47](#_Toc89287472)

# 1 СИСТЕМНЫЙ АНАЛИЗ

## 1.1 Название и цель разработки информационной системы

Название проектируемой информационной системы: Информационная система «Курьерская служба».

Результатами внедрения информационной системы являются:

-повышение производительности труда сотрудников;

-уменьшение времени ожидания получателя;

-уменьшение времени поиска, редактирования и удаления необходимой информации о сотрудниках.

ИС «Курьерская служба» должна обеспечивать выполнение следующих основных функций:

-хранение и поиск данных о сотрудниках службы;

-хранение и поиск информации о получателях и отправителях;

-хранение и поиск информации о наличии в службе доставки тех или иных посылок;

-поиск информации по следующим основным позициям: отдел, сотрудник, категория посылки, должность, посылка, оплата, отправитель, получатель.

## 1.2 Характеристика предприятия

### 1.2.1 Сведения о предприятии

Курьерская служба представляет собой профессиональную организацию, специализирующуюся на оперативной доставке грузов, деловой корреспонденции и документов. Курьерская служба является прекрасной альтернативой традиционным почтовым услугам, которые уже не могут удовлетворить коммерческие предприятия и частных граждан с точки зрения надежности и срочности доставки их отправлений. В современных условиях компаниям приходится быстро и адекватно реагировать на постоянно меняющиеся условия внешней среды, принимать правильные решения, направленные на развитие своего бизнеса. Поэтому коммерческим предприятиям необходим эффективный документооборот, а также своевременное оповещение сотрудников подразделений, партнеров по бизнесу и потребителей о своих новых предложениях или акциях. И, несмотря на развитие высоких технологий, и коммуникационных средств, эта важная информация не всегда может быть передана посредством электронной почты или факса. Единственный вариант - это курьерская служба, благодаря услугам которой можно быть уверенным в том, что деловая корреспонденция или документы будут доставлены к назначенному часу и переданы прямо в руки адресата. Организационная структура курьерской службы представлена на рисунке 1.1.

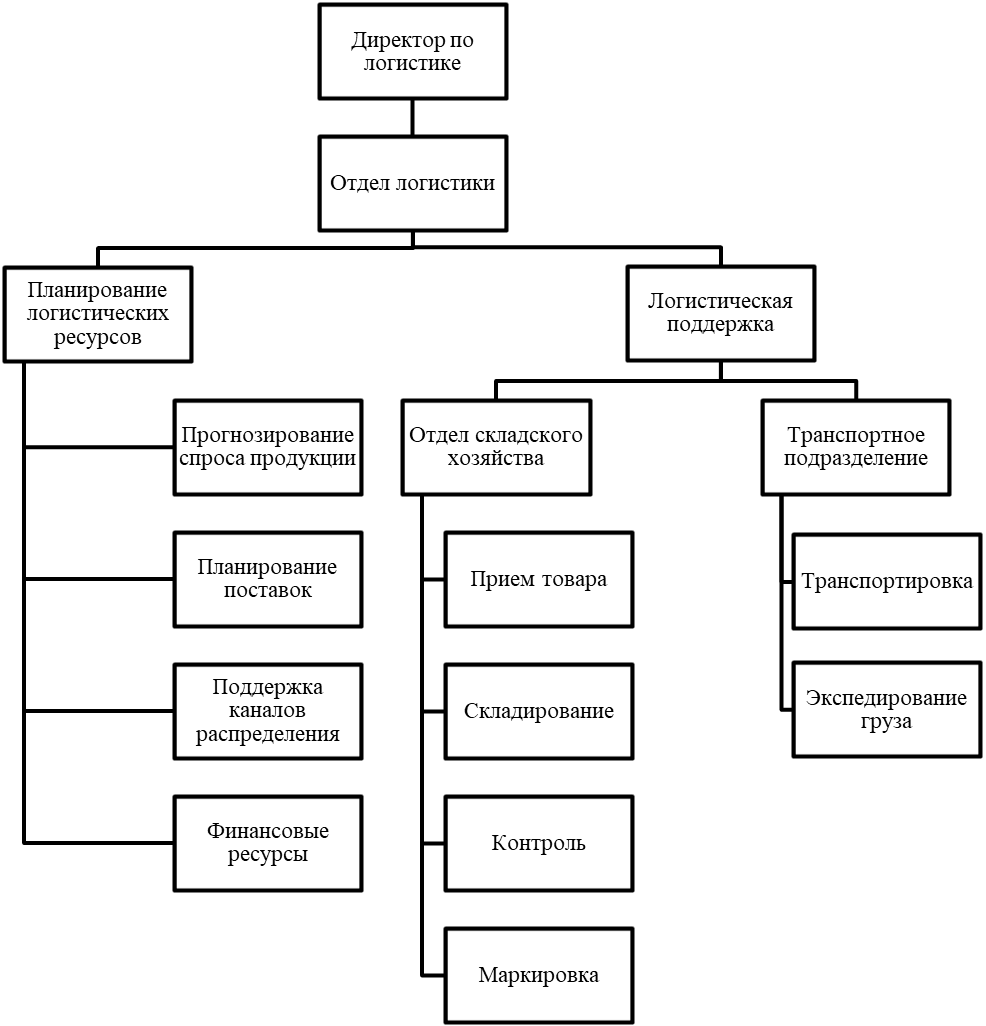


Рисунок 1.1 – Организационная структура курьерской службы

### 1.2.2 Сведения об отношениях предприятия с действующими лицами из окружающей среды

Курьерская служба взаимодействует со следующими типами лиц из окружающей среды:

* получатели;
* отправители;
* юридические лица.

Курьерская служба может вступать в следующие отношения с лицами из окружающей среды:

* осуществление доставки посылок клиентам;
* осуществление принятия посылок и отправка получателям;
* предоставление необходимой информации правоохранительным органам;
* предоставление информации налоговой службе.

### 1.2.3 Детализация отношений

1. Отношение «Осуществление доставки посылок клиентам». Процессы:

* получение менеджером запроса на доставку от отправителя;
* выдача менеджером результата на запрос отправителя.

2) Отношение «Осуществление принятия посылок и отправка получателям».

Процессы:

* принятие менеджером посылок и отправка их получателям;
* формирование и отправка менеджером заявки на доставку посылки клиенту.

1. Отношение «Предоставление необходимой информации правоохранительным органам».

Процессы:

* предоставление всей необходимой информации правоохранительным органам.

4) Отношение «Предоставление информации налоговой службе».

Процессы:

- предоставление всей необходимой информации налоговой службе.

### 1.2.4 Описание сценариев реализации процессов

Процесс №1 «Получение менеджером запроса на доставку от отправителя». Сценарий №1.

Отправитель заходит на сайт или приложение курьерской службы, заполняет бланк с информацией какую посылку нужно доставить, а также вносить данные, где нужно забрать и куда доставить. Менеджер, проверив правильность заполнения данных, вносить заказ в базу данных и передает его сотрудникам на доставку. При необходимости менеджер в диалоговом окне дает консультации отправителю, о том какую выбрать категорию посылки, по потребительским свойствам и особенностям грузов, а также как правильно заполнить данные, для успешной доставки посылки. Далее менеджер окончательно заполняет все данные для доставки и вносит посылку в документ «Журнал учета посылок».

Выявленные категории пользователей: менеджер, отправитель.

Выявленные типы документов: «Журнал учета посылок».

Заключение: Требуется автоматизировать поиск посылок у отправителя.

Процесс №2 «Выдача менеджером результата на запрос отправителя». Сценарий №1.

Если отправитель пришел на пункт приема посылок уже с посылкой, то менеджер оформляет все необходимые документы, называет клиенту сумму доставки, принимает от клиента деньги за доставку и передает посылку сотрудникам для передачи в доставку. При этом менеджер должен сделать запись об отправки посылки в документе «Журнал учета активных посылок». Изменение количества посылок происходит автоматически в документе «Журнал учета посылок» в соответствии с реальным количеством доставленных посылок.

Выявленные категории пользователей: менеджер, отправитель.

Выявленные типы документов: «Журнал учета активных посылок», «Журнал учета посылок».

Заключение: Требуется автоматизировать учет сведений о принятых к доставке посылках в документе «Журнал учета активных посылок» и уменьшение количества посылок в документе «Журнал учета посылок» в соответствии с реальным количеством доставленных посылок.

Процесс №2 «Выдача менеджером результата на запрос отправителя». Сценарий №2.

Если посылка не может быть доставлена в связи техническими неполадками или иными причинами, то посылка удаляется из документа «Журнал учета посылок», и менеджер отказывает клиенту в услуге по доставке запрашиваемой посылки.

Выявленные категории пользователей: менеджер, отправитель.

Выявленные типы документов: «Журнал учета посылок».

Заключение: Требуется автоматизировать учет сведений о не принятых к доставке посылках в документе «Журнал учета посылок».

Процесс №3 «Принятие менеджером посылок и отправка их клиентам». Сценарий №1.

Менеджер осуществляет принятие и отправку посылок необходимых для клиента, взаимодействуя с популярными производителями, имея доступ к их интернет-магазинам, либо самостоятельно в офисах курьерской службы по приему посылок для отправки. Если получателя устраивают условия курьерской службы, менеджер вносит сведения об этом отправителе в документ «Журнал учета отправителей».

Выявленные категории пользователей: менеджер, отправитель.

Выявленные типы документов: «Журнал учета отправителей».

Заключение: Требуется автоматизировать учет сведений об отправителях в документе «Журнал учета отправителей».

Процесс №4 «Формирование и отправка менеджером заявки на доставку посылки клиенту». Сценарий №1.

Менеджер, просматривая «Журнал учета посылок», принимает решение о необходимости отправки некоторых посылок в соответствии со сроками доставки. Для этого он формирует заявку из всех необходимых для отправки посылок. Затем менеджер просматривает «Журнал учета отправителей» и сверяет данные отправителей с посылками. После выполнения всех необходимых действий передает посылки сотрудникам для доставки их получателям.

Выявленные категории пользователей: менеджер.

Выявленные типы документов: «Журнал учета посылок», «Журнал учета отправителей».

Заключение: Требуется автоматизировать поиск отправителей в документе «Журнал учета отправителей».

Процесс №5 «Предоставление всей необходимой информации правоохранительным органам». Сценарий №1.

Управляющий при запросе правоохранительных органов предоставить им всю необходимую информацию о сотрудниках или о клиентах, с которыми они сотрудничают, просматривает «Журнал учета сотрудников», «Журнал учета клиентов» и после этого выдает необходимую информацию.

Выявленные категории пользователей: директор, правоохранительные органы.

Выявленные типы документов: «Журнал учета сотрудников», «Журнал учета клиентов».

Заключение: Требуется автоматизировать поиск сотрудников в «Журнал учета сотрудников», и клиентов в «Журнал учета клиентов».

Процесс №6 «Предоставление всей необходимой информации налоговой службе» Сценарий №1.

Директор при запросе налоговой службы предоставить им всю необходимую информацию о доставленных посылках, и выплаченных по ним налогах просматривает «Журнал учета посылок» после этого предоставляет необходимую информацию.

Выявленные категории пользователей: директор, налоговая служба.

Выявленные типы документов: «Журнал учета посылок».

Заключение: Требуется автоматизировать поиск информации о посылках в «Журнале учета посылок».

## 1.3 Сводные данные о категориях пользователей и их потребностях

Перечень действующих лиц, с которыми предприятие взаимодействует во внешней среде:

* ДЛ1. Отправители;
* ДЛ2. Получатели;
* ДЛ3. Юридические лица;

Перечень действующих лиц, которые являются сотрудниками предприятия и принимают участие в бизнес-процессах:

* ДЛ4. Менеджер;
* ДЛ5. Директор.

Перечень документов:

* Д1. «Журнал учета посылок»;
* Д2. «Журнал учета активных посылок»;
* Д3. «Журнал учета отправителей»;
* Д4. «Журнал учета сотрудников»;
* Д5. «Журнал учета клиентов».

Сводные данные о схеме взаимодействия пользователей с документами при осуществлении запроса на доставку представлены в таблице 1.1

Таблица 1.1 – Сводные данные о потребностях пользователей при получения запроса на доставку посылки

| **Процесс и сценарий** | **Операция и действие** | **Действующее лицо** | **Документы и данные, вход** | **Документы и данные, выход** |
| --- | --- | --- | --- | --- |
| Процесс №1 «Получение менеджером запроса на доставку от отправителя». Сценарий №1. | Консультация отправителя | ДЛ4.Менеджер | ФИО отправителя | Д1.Журнал учета посылок |
| Процесс №2 «Выдача менеджером результата на запрос отправителя». Сценарий №1. | Ответ менеджера на запрос отправителя | ДЛ4.Менеджер | - | Д1.Журнал учета посылок |
| Процесс №2 «Выдача менеджером результата на запрос отправителя». Сценарий №2. | Ответ менеджера на запрос отправителя | ДЛ4.Менеджер | - | - |

Сводные данные о схеме взаимодействия пользователей при принятии менеджером посылок представлены в таблице 1.2

Таблица 1.2 – Сводные данные о потребностях пользователей при принятии менеджером посылок

| **Процесс и сценарий** | **Операция и действие** | **Действующее лицо** | **Документы и данные, вход** | **Документы и данные, выход** |
| --- | --- | --- | --- | --- |
| Процесс №3 «Принятие менеджером посылок и отправка их клиентам». Сценарий №1. | Поиск посылок для отправки. | ДЛ4. Менеджер | - | Д3. «Журнал учета отправителей» |
| Процесс №4 «Формирование и отправка менеджером заявки на доставку посылки клиенту». Сценарий №1. | Выбор отправителей и посылок. Отправка посылки. | ДЛ4. Менеджер | Заявка на отправку посылок | Д1. «Журнал учета посылок»  Д4. «Журнал учета сотрудников» |

Сводные данные о схеме взаимодействия пользователей с информацией представлены в таблице 1.3

Таблица 1.3 – Сводные данные о потребностях юридических лиц в поиске информации о посылках

| **Процесс и**  **сценарий** | **Операция и действие** | **Действующее лицо** | **Документы и данные, вход** | **Документы и данные, выход** |
| --- | --- | --- | --- | --- |
| Процесс №5 «Предоставление всей необходимой информации правоохранительным органам». Сценарий №1. | Предоставление информации | ДЛ5. Директор | - | Д4,Д5. «Журнал учета сотрудников», «Журнал учета клиентов» |
| Процесс №6 «Предоставление всей необходимой информации налоговой службе» Сценарий №1. | Предоставление информации | ДЛ5. Директор | - | Д1. «Журнал учета посылок» |

## 1.4 Концептуальная модель данных предметной области

В ходе исследования предметной области составлена концептуальная модель данных, представленная на рисунке 1.5.



Рисунок 1.5 – Концептуальная модель предметной области

Сводные данные о сущностях концептуальной модели данных и их описание представлены в таблице 1.4

Таблица 1.4 – Справочник сущностей концептуальной модели данных

| **Номер сущности** | **Название сущности** | **Описание сущности** | **Ссылка на документ** |
| --- | --- | --- | --- |
| E1 | Категория посылки | Разделение посылки в зависимости от того, что требуется доставить | Д1 |
| E2 | Отдел | Место, где работают определенные сотрудники | Д4 |
| E3 | Сотрудники | Люди, работающие в курьерской службе | Д4 |
| E4 | Должность | Ступень, которую сотрудник занимает в курьерской службе | Д4 |
| E5 | Посылка | Хранит всю информацию о посылках | Д1,Д2,Д3 |
| E6 | Вид оплаты | Определяет вид оплаты услуг курьерской службы | Д1,Д2,Д3 |
| E7 | Отправитель | Хранит информацию о отправителях посылки | Д2 |
| E8 | Получатель | Хранит данные о том, кому необходимо доставить заказ | Д3 |

Сводные данные об атрибутах концептуальной модели данных и их описании представлены в таблице 1.5.

Таблица 1.5 – Справочник атрибутов концептуальной модели

| **Номер атрибута** | **Название атрибута** | **Тип атрибута** | **Название сущности** | **Описание** |
| --- | --- | --- | --- | --- |
| A1 | ID Категории посылки | Целое число | E1.Категория посылки | Для определения уникального номера категории посылки |
| A2 | Название категории посылки | Текст | E1.Категория посылки | Для определения категории посылки |
| A3 | ID Отдела | Целое число | E2.Отдел | Для присвоения отделу своего уникального кода |
| A4 | Название отдела | Текст | E2.Отдел | Для присвоения отделу своего названия |
| A5 | ID Сотрудника | Целое число | E3.Сотрудники | Уникальный номер сотрудника |
| A6 | Имя сотрудника | Текст | E3.Сотрудники | Имя сотрудника |
| A7 | ID Должности | Целое число | E4.Должности | Уникальный номер должности сотрудника |
| A8 | Название должности | Текст | E4.Должности | Должность сотрудника |
| A9 | ID Посылки | Целое число | E5.Посылка | Уникальный номер посылки |
| A10 | Название | Текст | E5. Посылка | Название посылки |
| A11 | Стоимость | Целое число | E5. Посылка | Для определения ценности посылки |
| A12 | Вес | Целое число | E5. Посылка | Для определения способа доставки |
| A13 | Количество | Целое число | E5. Посылка | Для определения кол-ва товара |
| A14 | ID Отправителя | Целое число | E7.Отправитель | Уникальный номер отправителя |
| A15 | Название | Текст | E7. Отправитель | Название получателя посылки |
| A16 | Адрес | Текст | E7. Отправитель | Адрес отправителя |
| A17 | Статус посылки | Текст | E5. Посылка | Статус посылки во время доставки |
| A18 | ID Получателя | Целое число | E8.Получатель | Уникальный номер получателя |
| A19 | ФИО | Текст | E8. Получатель | ФИО получателя |
| A20 | Адрес | Текст | E8. Получатель | Адрес фирмы или офиса получателя |
| A21 | ID Вида оплаты | Целое число | E6.Вид оплаты | Уникальный код вида оплаты |
| A22 | Название вида оплаты | Текст | E6.Вид оплаты | Название вида оплаты(наличный, безналичный) |

Сводные данные о связях концептуальной модели данных представлены в таблице 1.6.

Таблица 1.6 – Справочник отношений концептуальной модели

| **Номер** | **Название** | **Сущности** | **Тип отношений** | **Документы** |
| --- | --- | --- | --- | --- |
| R1 | Относятся | E1.Категории – E2.Отделы | 1:1, 1:n | Д4 |
| R2 | Работать | E2.Отделы – E3.Сотрудники | 1:1, 1:n | Д4 |
| R3 | Занимать | E3.Сотрудники – E4. Должность | 1:1, 1:n | Д4 |
| R4 | Относится | E1.Категория -E5.Посылка | 1:1, 1:n | Д1 |
| R5 | Отправлять | E5.Посылка – E7.Отправитель | 1:1, 1:n | Д2 |
| R6 | Получать | E5.Посылка – E8.Получатель | 1:1, 1:n | Д3 |
| R7 | Принимать | E5.Посылка – E6.Вид оплаты | 1:1, 1:n | Д1 |
| R8 | Доставлять | Е3.Сотрудник-Е5.Посылка | 1:1, 1:n | Д4 |

## 1.5 Описание специфических требований к программному продукту, опосредованно связанных с предметной областью

В ходе исследования предметной области были составлены следующие специфические требования к программному продукту:

- сотрудник может иметь только одну должность;

- при поступлении на работу сотруднику выдается уникальный табельный номер;

- на любую доставляемую посылку у курьерской службы устанавливается определенная наценка;

- поле стоимость посылки не должно быть пустым.

# 2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

## 2.1 Иерархия функций информационной системы

В разрабатываемой информационной системе «Курьерская служба» можно выделить ряд следующих обобщенных функций:

* учет сотрудников. Данная функция информационной системы позволяет осуществлять работу с информацией о сотрудниках и должностях;
* учет отправителей. Данная функция информационной системы позволяет осуществлять работу с информацией об отправителей;
* учет посылок. Данная функция информационной системы позволяет осуществлять работу с информацией о посылках.

Разработанная функциональная иерархия информационной системы «Курьерская служба» представлена на рисунках 2.1, 2.2, 2.3.



Рисунок 2.1 – Иерархия функций информационной системы, раздел   
«Учет сотрудников»



Рисунок 2.2 – Иерархия функций информационной системы, раздел   
«Учет отправителей»



Рисунок 2.3 – Иерархия функций информационной системы, раздел   
«Учет посылок»

## 2.2 Структура системы

### 2.2.1 Структурная схема системы

На основании описанных функций информационной системы разработана структурная схема системы, представленная на рисунке 2.4.



Рисунок 2.4 - Структурная схема информационной системы

### 2.2.2 Описание подсистем, из которых состоит программный продукт

В состав информационной системы входят:

- клиентское приложение;

* подсистема учета сотрудников и должностей, предназначена для обеспечения добавления сотрудников, изменение сведений о них, увольнения, поиска;

- подсистема учета отправителей, предназначена для:

1. добавления отправителей;

1. изменения сведений об отправителях;
2. удаление данных;
3. поиск.

- подсистема учета посылок, с помощью которой осуществляется учет сведений о доставке посылок и их поиске;

- база данных, в которой хранятся сведения о сотрудниках, должностях, посылках, получателях, категориях посылок, отправителях и др.

## 2.3 Проектирование пользовательского интерфейса

### 2.3.1 Схема интерфейса

Схема пользовательского интерфейса представлена на рисунке 2.5.



Рисунок 2.5 - Схема пользовательского интерфейса

При входе в приложение пользователю отображается главная форма (рисунок 2.6). Из главной формы пользователь может перейти в следующие разделы: «Вид оплаты», «Получатели», «Отправители», «Должности», «Категории посылок», «Отделы», «Сотрудники» и «Посылки».



Рисунок 2.6 - Главное меню информационной системы

С помощью раздела «Вид оплаты» (рисунок 2.7) можно посмотреть все виды оплаты, имеющиеся у курьерской службы, а также произвести поиск, внести новые и удалить их.

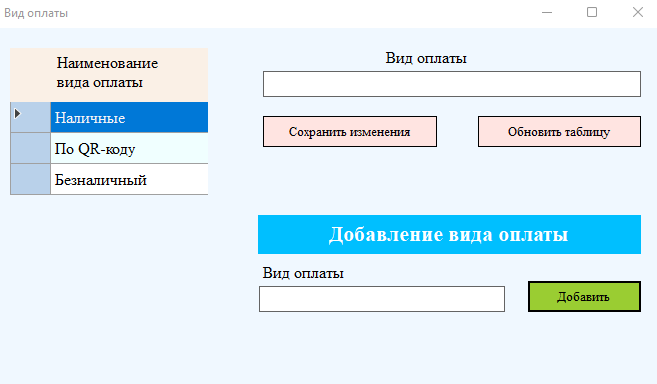


Рисунок 2.7 – Раздел программы «Виды оплаты»

С помощью раздела «Получатели» (рисунок 2.8) можно посмотреть список получателей, добавить новых, изменить данные о уже существующих и осуществить поиск по ним.

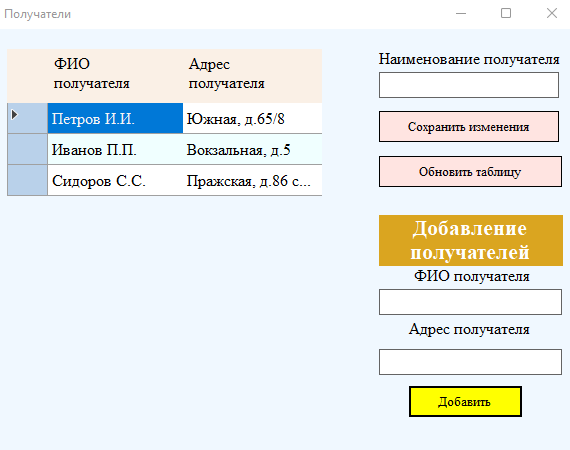


Рисунок 2.8 - Раздел «Получатели»

С помощью раздела «Должности» (рисунок 2.9) мы добавляем список должностей, которые будут в данной курьерской службе. Имеем возможность их редактирования, поиска и удаления.

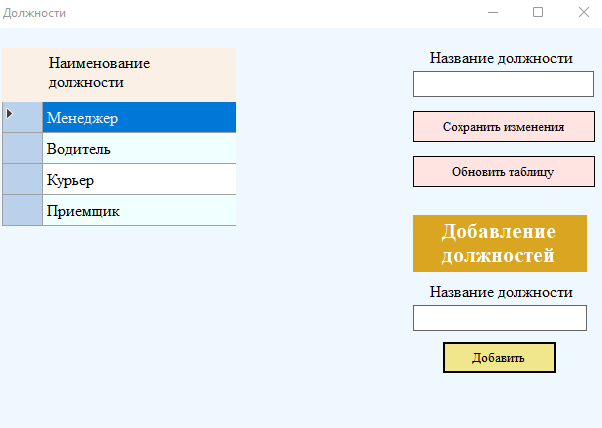


Рисунок 2.9 – Раздел «Должности»

С помощью раздела «Категории» (рисунок 2.10) мы имеем возможность добавить категории посылок, которые будут доставляться, а также произвести их поиск, удаление и редактирование.

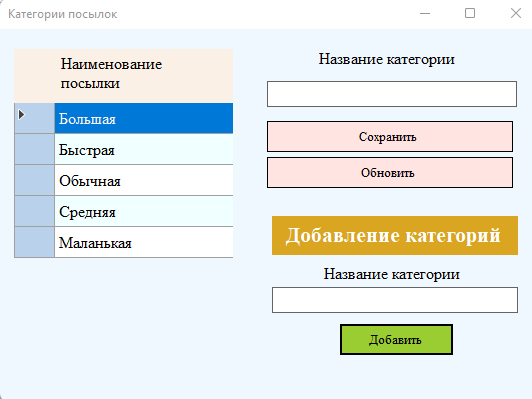


Рисунок 2.10 – Раздел «Категории посылок»

С помощью раздела «Сотрудники» (рисунок 2.11) мы имеем возможность всегда иметь сведения о своем рабочем штате, принимать на работу новых сотрудников, производить поиск в каком отделе они работают, и какую должность занимают.

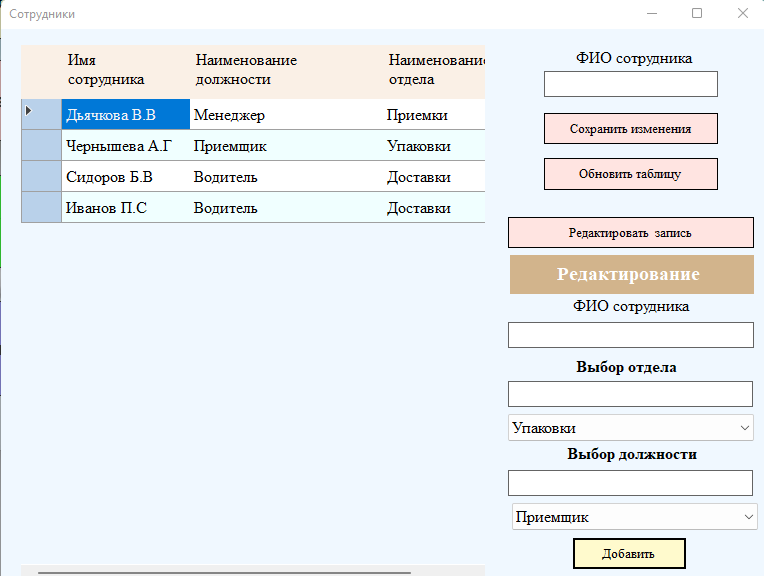


Рисунок 2.11 – Раздел «Сотрудники»

С помощью раздела «Посылки» (рисунок 2.12) мы имеем полный доступ ко всем посылкам, которые находятся в курьерской службе. Также есть возможность добавлять новые посылки, записывать стоимость, вес, количество, редактировать имеющиеся данные и производить поиск.

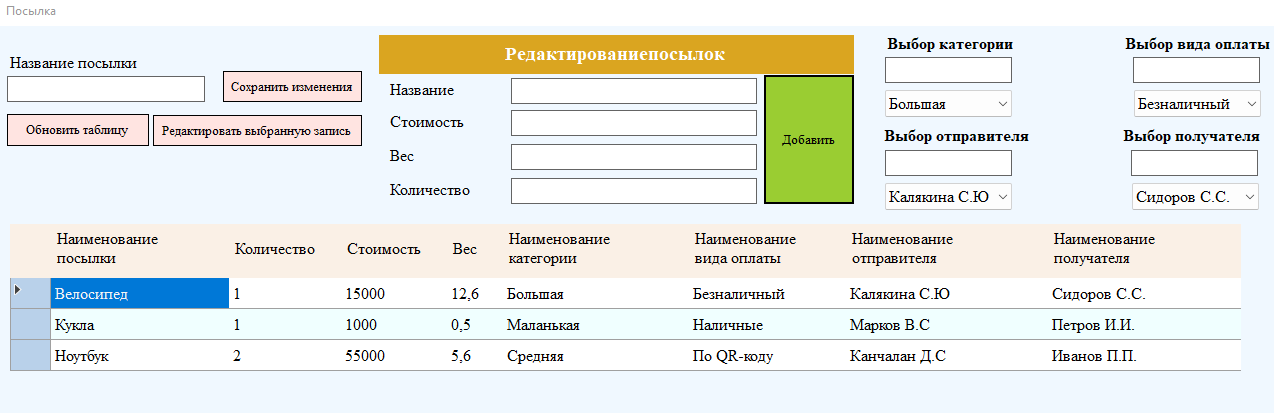


Рисунок 2.12 – Раздел «Посылки»

## 2.4 Проектирование БД

### 2.4.1 Логическая модель данных

На основе концептуальной модели данных получена логическая модель данных, представленная на рисунке 2.14.

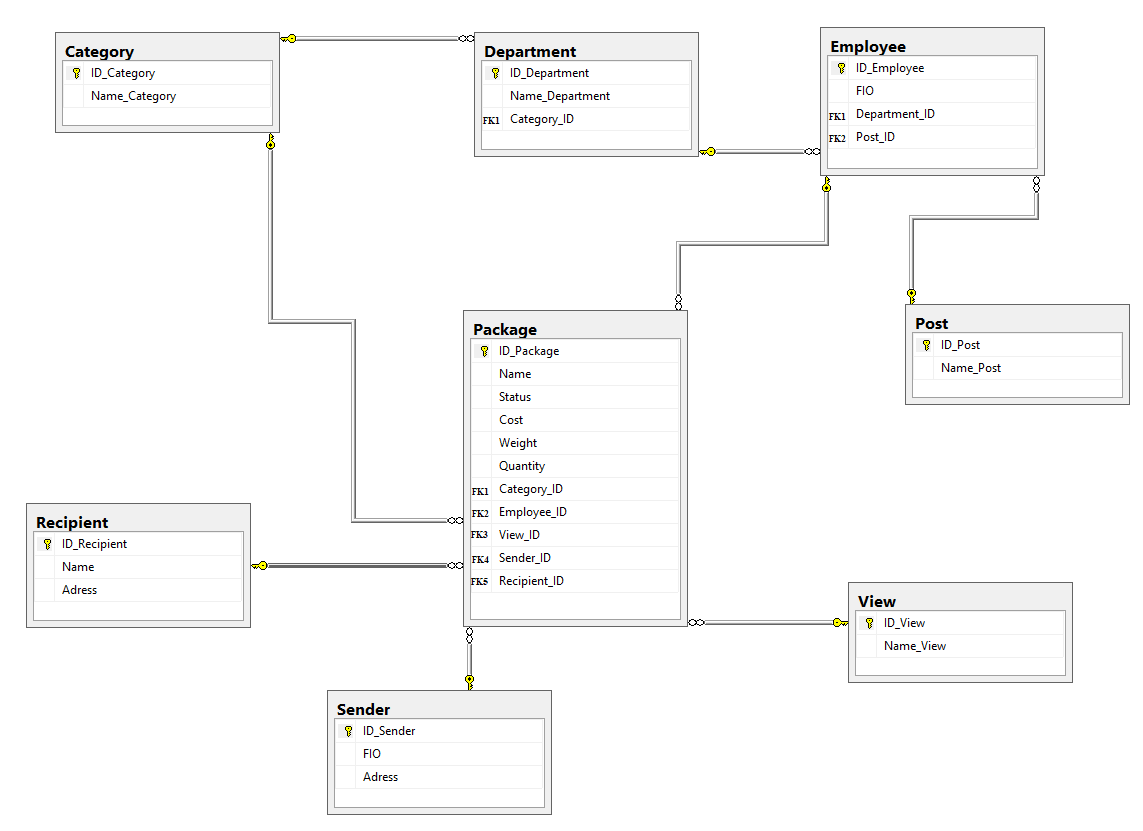


Рисунок 2.14 – Логическая модель данных

### 2.4.2 Составление таблиц со списком полей логической модели.

Таблица 2.1 – Список полей для таблицы Категория посылок.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип поля | Признаки | Описание |
| 1 | ID Категории | Целое число | PK | Уникальный код категории заказа |
| 2 | Название категории посылок | Текст | M, u | Название категории |

Таблица 2.2 – Список полей для таблицы Отдел.

| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| --- | --- | --- | --- | --- |
| 1 | ID Отдела | Целое число | PK | Уникальный код отдела |
| 2 | Название отдела | Текст | M, u | Название отдела продаж |
| 3 | ID Категории | Целое число | FK, m,u | Уникальный код категории |

Таблица 2.3 – Список полей для таблицы Сотрудники.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| 1 | ID Сотрудника | Целое число | PK | Уникальный код сотрудника |
| 2 | ФИО сотрудника | Текст | M, u | Присваивается имя сотрудника |
| 3 | ID Должности | Целое число | FK, m,u | Уникальный код должности |
| 4 | ID Отдела | Целое число | FK, m,u | Уникальный код отдела |

Таблица 2.4 – Список полей для таблицы Должности.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| 1 | ID Должности | Целое число | PK | Уникальный код должности |
| 2 | Название должности | Текст | M, u | Название занимаемой должности |

Таблица 2.5 – Список полей для таблицы Посылки.

| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| --- | --- | --- | --- | --- |
| 1 | ID Посылки | Целое число | PK | Уникальный код посылки |
| 2 | Наименование посылки | Текст | M, u | Наименование каждого товара |
| 3 | Статус посылки | Текст | M, u | Статус отслеживаемой посылки |
| 4 | Стоимость посылки | Целое число | M, u | Цена доставляемой посылки |
| 5 | Вес | Целое число | M, u | Вес доставляемой посылки |
| 6 | Количество | Целое число | M, u | Мера посылки |
| 7 | ID Вида оплаты | Целое число | FK, m u | Уникальный код вида оплаты |
| 8 | ID Категории посылки | Целое число | FK, m u | Уникальный код категории посылки |
| 9 | ID Отправителя | Целое число | FK, m u | Уникальный код отправителя |
| 10 | ID Получателя | Целое число | FK, m u | Уникальный код получателя |
| 11 | ID Сотрудника | Целое число | FK, m u | Уникальный код сотрудника |

Таблица 2.6 – Список полей для таблицы Отправители.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| 1 | ID Отправителя | Целое число | PK | Уникальный код отправителя |
| 2 | Название отправителя | Текст | M, u | Название отправителя |
| 3 | Адрес отправителя | Текст | M, u | Адрес, где находится отправитель |

Таблица 2.7 – Список полей для таблицы Получатели.

| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| --- | --- | --- | --- | --- |
| 1 | ID Получателя | Целое число | PK | Уникальный код получателя |
| 2 | ФИО получателя | Текст | M, u | ФИО получателя |
| 3 | Адрес получателя | Текст | M, u | Адрес, где проживает клиент, которому необходимо доставить посылку |

Таблица 2.8 – Список полей для таблицы Вид оплаты.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| 1 | ID Вида оплаты | Целое число | PK | Уникальный номер вида оплаты |
| 2 | Название вида оплаты | Текст | M, u | Название вида оплаты определенной посылки |

### 2.4.3 Описание работ проделанных по нормализации

Выявлены следующие функциональные зависимости:

1. Отношение «Категории посылок»:

- ID Категории=> Наименование категории.

1. Отношение «Отделы»:

- ID Отдела=> Наименование отдела.

- ID Категории => ID Категории.

1. Отношение «Сотрудники»:

- ID Сотрудника=> Имя сотрудника;

- ID Сотрудника => ID Должности;

- ID Сотрудника => ID Отдела.

1. Отношение «Должности»:

- ID Должности=> Наименование должности.

1. Отношение «Посылка»:

- ID Посылки => Наименование посылки;

- ID Посылки => Статус;

- ID Посылки => Стоимость;

- ID Посылки => Вес;

- ID Посылки => Количество;

- ID Посылки => ID Вида оплаты;

- ID Посылки => ID Категории;

- ID Посылки => ID Сотрудника;

- ID Посылки => ID Отправителя;

- ID Посылки => ID Получателя.

1. Отношение «Отправители»:

- ID Производителя=> Название отправителя;

- ID Производителя => Адрес отправителя.

1. Отношение «Получатель»:

- ID Получателя=> ФИО получателя;

- ID Получателя => Адрес получателя.

1. Отношение «Вид оплаты»:

- ID Вида оплаты => Название вида оплаты.

Все приведенные выше отношения находятся в первой нормальной форме, так как атрибуты этих отношений не упорядочены и различаются по наименованию и все значения атрибутов атомарны.

Приведенные выше отношения находятся также и во второй нормальной форме, так как нет не ключевых атрибутов, зависящих от части сложного ключа.

И наконец, можно сказать, что приведенные выше отношения находятся также и в третьей нормальной форме, так как все не ключевые атрибуты этих отношений взаимно независимы.

Таким образом, логическая модель данных изначально находилась в третьей нормальной форме.

### 2.4.4 Описание используемых средств обеспечения целостности данных

Правильно спроектированная и поддерживаемая база данных не допускает возможности нарушения ссылочной целостности. В разрабатываемой базе данных для обеспечения целостности внешних ключей используются механизмы автоматического поддержания ссылочной целостности.

Так, при операции редактирования записи невозможно изменить значение первичного ключа. А при операции изменения внешнего ключа, хранящегося в записи, или при операции удаления записи, на которую имеются ссылки, происходит запрет выполнения этой операции.

Например, при удалении должности «Водитель» из таблицы «Должности» приложение не удалит данную запись из-за ошибки, так как в таблице «Сотрудники» имеются записи, содержащие ссылки на удаляемую запись (рисунок 2.16).

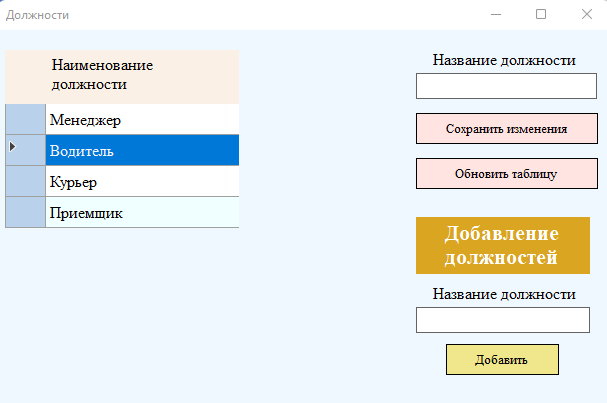


Рисунок 2.16 – Удаление записи

# 3 ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

## 3.1 Выбор языка, среды, которые будут использоваться в ходе разработки приложения.

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей. По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла в феврале 2002 года. Текущей версией языка является версия C# 7.0, которая вышла в марте 2017 года.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла в феврале 2002 года. Текущей версией языка является версия C# 7.0, которая вышла в марте 2017 года.

C# является объектно-ориентированным языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных возможностей языка.

C++ является одним из самых старых объектно-ориентированных языков, построенных на основе классического для многих программистов языка C. Многие плюсы C++ одновременно являются его минусами, поскольку за свою относительно долгую историю этот язык вобрал в себя столько всего, что порой пугает свой сложностью и избыточностью. Управление памятью, в отличие от языков C# и Java, находится в зоне ответственности разработчика, а значит, программистам приходится уделять 32 этому много внимания, отвлекаясь от основных тем. C# и C++ отличаются и по сфере применения. Если C# больше ориентирован на разработку классического прикладного программного обеспечения и веб-приложений, то C++ - это прекрасный инструмент для написания системного ПО, в том числе самих операционных систем.

После сравнения двух языков программирования: С++ и С#, предпочтение было отдано языку C#. Таким образом, в ходе разработки клиентского приложения для информационной системы будет использоваться язык программирования C#.

Для того чтобы облегчить написание, а также тестирование и отладку программного кода обычно используют специальные среды разработки. В ходе разработки приложения будет использоваться среда разработки Visual Studio 2013.

## 3.2 Выбор СУБД

На сегодняшний момент известно большое число различных СУБД. При выборе конкретной СУБД очень важно выбрать такую, которая в наибольшей степени соответствует предъявляемым к информационной системе требованиям.

СУБД Oracle наделена самым развитым набором функций для работы с языком Java и доступа к данным через Интернет, системой оптимизации одновременного доступа. Единственным недостатком данной СУБД является сложность администрирования, однако все затраты на ее внедрение и освоение в последствии окупятся эффективной и надежной работой. Среди основных свойств СУБД Oracle следует отметить такие, как:

* высочайшая надежность;
* возможность разбиения крупных баз данных на разделы;
* наличие универсальных средств защиты информации;
* эффективные методы максимального повышения скорости обработки запросов;
* распараллеливание операций в запросе.
* наличие широкого спектра средств разработки, мониторинга и администрирования.

Microsoft SQL Server - система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия. Важнейшими характеристиками данной СУБД являются:

* простота администрирования;
* возможность подключения к Web;
* быстродействие и функциональные возможности механизма сервера СУБД;
* наличие средств удаленного доступа.

В комплект средств административного управления данной СУБД входит целый набор специальных мастеров и средств автоматической настройки параметров конфигурации. Входящий в комплект поставки сервер OLAP дает возможность сохранять и анализировать все имеющиеся у пользователя данные. В принципе MS SQL Server представляет собой современную полнофункциональную СУБД, которая идеально подходит для малых и средних организаций.

Необходимо отметить, что MS SQL Server уступает другим рассматриваемым СУБД по двум важным показателям: программируемость и средства работы. При разработке клиентских приложений на основе языков Java, HTML часто возникает проблема недостаточности программных средств MS SQL Server и пользоваться этой СУБД будет труднее, чем системами DB2, Informix, Oracle или Sybase. Общемировой тенденцией в XXI веке стал практически повсеместный переход на платформу LINUX, а SQL Server функционирует только в среде Windows.

После сравнения двух СУБД: Oracle и Microsoft SQL Server, предпочтение было отдано MS SQL Server. Таким образом, для работы с базой данных будет использоваться СУБД Microsoft SQL Server 2012.

## 3.3 Физическая модель данных

На основе логической модели данных с учетом выбранной СУБД была получена физическая модель данных, представлена на рисунке 3.1.

### Составление таблиц со списком полей физической модели

В таблицах 3.1 ÷ 3.8 приведены списки полей всех таблиц физической модели, приведенной выше.

Таблица 3.1 – Список полей для таблицы Category.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип поля | Признаки | Описание |
| 1 | ID Category | uniqueidentifier | PK | Уникальный код категории посылки |
| 2 | Name Category | nvarchar(MAX) | M, u | Название категории посылки |

Таблица 3.2 – Список полей для таблицы Department.

| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| --- | --- | --- | --- | --- |
| 1 | ID Department | uniqueidentifier | PK | Уникальный код отдела |
| 2 | Name | nvarchar(MAX) | M, u | Название отдела |
| 3 | ID Category | uniqueidentifier | FK, m,u | Уникальный код категории посылки |

Таблица 3.3 – Список полей для таблицы Employee.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| 1 | ID Employee | uniqueidentifier | PK | Уникальный код сотрудника |
| 2 | FIO | nvarchar(MAX) | M, u | Присваивается фио сотрудника |
| 3 | ID Posts | uniqueidentifier | FK, m,u | Уникальный код должности |
| 4 | ID Department | uniqueidentifier | FK, m,u | Уникальный код отдела |

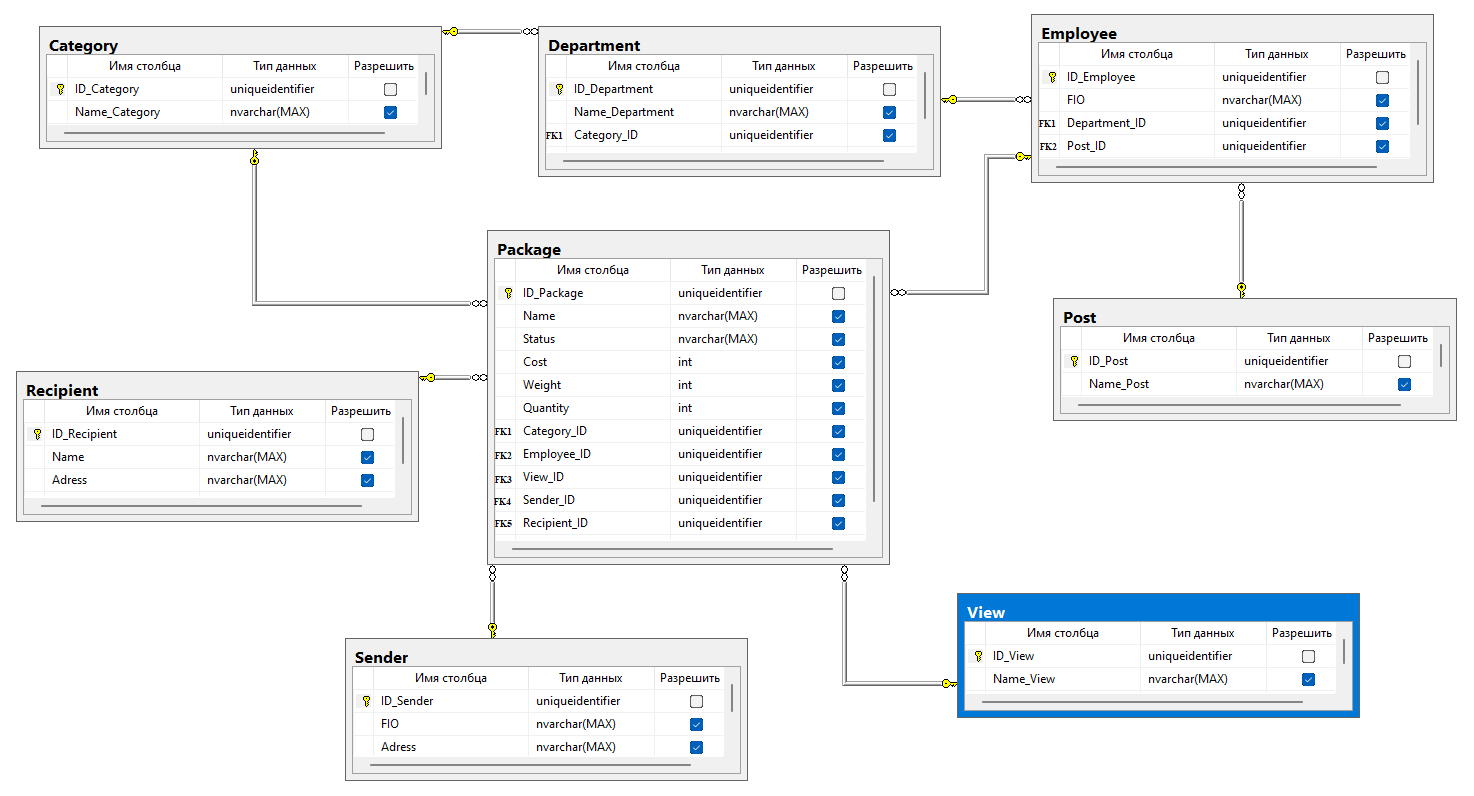


Рисунок 3.1 – Физическая модель данных

Таблица 3.4 – Список полей для таблицы Post.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| 1 | ID Post | uniqueidentifier | PK | Уникальный код должности |
| 2 | Name\_Post | nvarchar(MAX) | M, u | Название занимаемой должности |

Таблица 3.5 – Список полей для таблицы Package.

| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| --- | --- | --- | --- | --- |
| 1 | ID Package | uniqueidentifier | PK | Уникальный код посылки |
| 2 | Name | nvarchar(MAX) | M, u | Наименование каждой посылки |
| 3 | Status | nvarchar(MAX) | M, u | Название статуса посылки |
| 4 | Cost | real | M, u | Стоимость посылки |
| 5 | Weight | real | M, u | Вес посылки |
| 6 | Quantity | int | M, u | Мера посылки |
| 7 | View\_ID | uniqueidentifier | FK, m u | Уникальный код вида оплаты |
| 8 | Category\_ID | uniqueidentifier | FK, m u | Уникальный код категории посылки |
| 9 | Sender\_ID | uniqueidentifier | FK, m u | Уникальный код отправителя |
| 10 | Recipient\_ID | uniqueidentifier | FK, m u | Уникальный код получателя |
| 11 | Caterory\_ID | uniqueidentifier | FK, m u | Уникальный код категории |

Таблица 3.6 – Список полей для таблицы Providers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| 1 | ID Sender | uniqueidentifier | PK | Уникальный код отправителя |
| 2 | Name\_ Sender | nvarchar(MAX) | M, u | ФИО отправителя |
| 3 | Address | nvarchar(MAX) | M, u | Адрес, где находится отправитель |

Таблица 3.7 – Список полей для таблицы Recipient.

| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| --- | --- | --- | --- | --- |
| 1 | ID Recipient | uniqueidentifier | PK | Уникальный код получателя |
| 2 | Name | nvarchar(MAX) | M, u | Имя получателя |
| 3 | Address | nvarchar(MAX) | M, u | Адрес, куда нужно доставить посылку |

Таблица 3.8 – Список полей для таблицы View.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Поле** | **Тип поля** | **Признаки** | **Описание** |
| 1 | ID\_View | uniqueidentifier | PK | Уникальный номер вида оплаты |
| 2 | Name\_ View | nvarchar(MAX) | M, u | Название вида оплаты определенной посылки |

## 3.4 Разработка триггеров, хранимых процедур, запросов на выборку, представлений

В ходе проектирования базы данных был разработан триггер для удаления всех посылок, у которых есть ID удаляемого отправителя. То есть если был удален отправитель с ID (1,2,3,..n), то триггер ищет все посылки, у которых в поле Sender есть такие ID и удаляет их.

Скрипт создания данного триггера приведен ниже.

CREATE TRIGGER DeletePackageOnDeleteSender  
ON Sender for Delete as  
BEGIN  
DELETE FROM Package WHERE SenderId in (select Id from deleted);  
END;

# ЗАКЛЮЧЕНИЕ

В курсовой работе на основе данных, полученных при анализе предметной области и анализе особенностей работы предприятия, была разработана и реализована база данных для ведения необходимой информации, составлен и реализован ряд алгоритмов для выполнения функций обработки информации, соответствующих специфике работы предприятия.

Информационная система предназначена для ведения информации о посылках, отделах, категориях посылок, отправителях, должностях, получателях и видах оплаты.

ИС «Курьерская служба» должна обеспечивать выполнение следующих основных функций:

-хранение и поиск данных о сотрудниках службы;

-хранение и поиск информации о получателях и отправителях;

-хранение и поиск информации о наличии в службе доставки тех или иных посылок;

-поиск информации по следующим основным позициям: отдел, сотрудник, категория посылки, должность, посылка, оплата, отправитель, получатель.

Преимуществами созданной системы является повышение производительности труда сотрудников службы доставки, систематизация данных предприятия, повышение уровня понятности информации.

В первой главе работы составлена характеристика предприятия, описаны данные о категориях пользователей и их потребностях, разработана концептуальная модель данных предметной области.

Во второй главе работы составлена функциональная схема разрабатываемого приложения, спроектированы схема информационной системы, а также схема пользовательского интерфейса, разработана логическая модель данных и проведена работа по нормализации логической модели, описаны используемые средства обеспечения целостности данных.

В третьей главе работы рассмотрены вопросы технического проектирования информационной системы. Обоснован выбор комплекса технических средств и СУБД, построена физическая модель данных проектируемой базы данных. При разработке системы были использованы следующие приложения:

* Visual Studio 2017;
* Microsoft SQL Server 2014.

При оформлении пояснительной записки использовались:

* Microsoft Word 2010;
* Microsoft Visio 2007.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Пушников Ф. Ю. Введение в системы управления базами данных – Уфа: 1999 г.
2. Мартин Г. Понимание SQL – М.: 1993 г., 291 стр.
3. Полное руководство по языку программирования С# 7.0 и платформе .NET 4.7, интернет ресурс https://metanit.com/sharp/tutorial/ (дата обращения 19.12.18)
4. Руководство по Entity Framework, интернет ресурс https://metanit.com/sharp/entityframework/ (дата обращения 17.12.18)
5. [Julia Lerman](https://www.ozon.ru/person/27438013/), [Rowan Miller](https://www.ozon.ru/person/32018180/) Programming Entity Framework: Code First - 2012 г
6. Джозеф Албахари, [Бен Албахари](https://www.ozon.ru/person/4166242/) C# 7.0. Карманный справочник - 2018

# ПРИЛОЖЕНИЕ А. SQL СКРИП БАЗЫ ДАННЫХ

USE [master]

GO

/\*\*\*\*\*\* Object: Database [Service] Script Date: 01.12.2022 18:58:18 \*\*\*\*\*\*/

CREATE DATABASE [Service]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'Service', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\Service.mdf' , SIZE = 5120KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )

LOG ON

( NAME = N'Service\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\Service\_log.ldf' , SIZE = 2048KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)

GO

ALTER DATABASE [Service] SET COMPATIBILITY\_LEVEL = 120

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [Service].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [Service] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [Service] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [Service] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [Service] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [Service] SET ARITHABORT OFF

GO

ALTER DATABASE [Service] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [Service] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [Service] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [Service] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [Service] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [Service] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [Service] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [Service] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [Service] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [Service] SET DISABLE\_BROKER

GO

ALTER DATABASE [Service] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [Service] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [Service] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [Service] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [Service] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [Service] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [Service] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [Service] SET RECOVERY SIMPLE

GO

ALTER DATABASE [Service] SET MULTI\_USER

GO

ALTER DATABASE [Service] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [Service] SET DB\_CHAINING OFF

GO

ALTER DATABASE [Service] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [Service] SET TARGET\_RECOVERY\_TIME = 0 SECONDS

GO

ALTER DATABASE [Service] SET DELAYED\_DURABILITY = DISABLED

GO

USE [Service]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Category] Script Date: 01.12.2022 18:58:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Category](

[ID\_Category] [uniqueidentifier] NOT NULL,

[Name\_Category] [nvarchar](max) NULL,

CONSTRAINT [PK\_Category] PRIMARY KEY CLUSTERED

(

[ID\_Category] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Department] Script Date: 01.12.2022 18:58:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Department](

[ID\_Department] [uniqueidentifier] NOT NULL,

[Name\_Department] [nvarchar](max) NULL,

[Category\_ID] [uniqueidentifier] NULL,

CONSTRAINT [PK\_Department] PRIMARY KEY CLUSTERED

(

[ID\_Department] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Employee] Script Date: 01.12.2022 18:58:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Employee](

[ID\_Employee] [uniqueidentifier] NOT NULL,

[FIO] [nvarchar](max) NULL,

[Department\_ID] [uniqueidentifier] NULL,

[Post\_ID] [uniqueidentifier] NULL,

CONSTRAINT [PK\_Employee] PRIMARY KEY CLUSTERED

(

[ID\_Employee] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Package] Script Date: 01.12.2022 18:58:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Package](

[ID\_Package] [uniqueidentifier] NOT NULL,

[Name] [nvarchar](max) NULL,

[Status] [nvarchar](max) NULL,

[Cost] [int] NULL,

[Weight] [int] NULL,

[Quantity] [int] NULL,

[Category\_ID] [uniqueidentifier] NULL,

[Employee\_ID] [uniqueidentifier] NULL,

[View\_ID] [uniqueidentifier] NULL,

[Sender\_ID] [uniqueidentifier] NULL,

[Recipient\_ID] [uniqueidentifier] NULL,

CONSTRAINT [PK\_Package] PRIMARY KEY CLUSTERED

(

[ID\_Package] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Post] Script Date: 01.12.2022 18:58:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Post](

[ID\_Post] [uniqueidentifier] NOT NULL,

[Name\_Post] [nvarchar](max) NULL,

CONSTRAINT [PK\_Post] PRIMARY KEY CLUSTERED

(

[ID\_Post] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Recipient] Script Date: 01.12.2022 18:58:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Recipient](

[ID\_Recipient] [uniqueidentifier] NOT NULL,

[Name] [nvarchar](max) NULL,

[Adress] [nvarchar](max) NULL,

CONSTRAINT [PK\_Recipient] PRIMARY KEY CLUSTERED

(

[ID\_Recipient] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Sender] Script Date: 01.12.2022 18:58:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Sender](

[ID\_Sender] [uniqueidentifier] NOT NULL,

[FIO] [nvarchar](max) NULL,

[Adress] [nvarchar](max) NULL,

CONSTRAINT [PK\_Sender] PRIMARY KEY CLUSTERED

(

[ID\_Sender] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[View] Script Date: 01.12.2022 18:58:18 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[View](

[ID\_View] [uniqueidentifier] NOT NULL,

[Name\_View] [nvarchar](max) NULL,

CONSTRAINT [PK\_View] PRIMARY KEY CLUSTERED

(

[ID\_View] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Department] WITH CHECK ADD CONSTRAINT [FK\_Department\_Category] FOREIGN KEY([Category\_ID])

REFERENCES [dbo].[Category] ([ID\_Category])

GO

ALTER TABLE [dbo].[Department] CHECK CONSTRAINT [FK\_Department\_Category]

GO

ALTER TABLE [dbo].[Employee] WITH CHECK ADD CONSTRAINT [FK\_Employee\_Department] FOREIGN KEY([Department\_ID])

REFERENCES [dbo].[Department] ([ID\_Department])

GO

ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT [FK\_Employee\_Department]

GO

ALTER TABLE [dbo].[Employee] WITH CHECK ADD CONSTRAINT [FK\_Employee\_Post] FOREIGN KEY([Post\_ID])

REFERENCES [dbo].[Post] ([ID\_Post])

GO

ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT [FK\_Employee\_Post]

GO

ALTER TABLE [dbo].[Package] WITH CHECK ADD CONSTRAINT [FK\_Package\_Category] FOREIGN KEY([Category\_ID])

REFERENCES [dbo].[Category] ([ID\_Category])

GO

ALTER TABLE [dbo].[Package] CHECK CONSTRAINT [FK\_Package\_Category]

GO

ALTER TABLE [dbo].[Package] WITH CHECK ADD CONSTRAINT [FK\_Package\_Employee] FOREIGN KEY([Employee\_ID])

REFERENCES [dbo].[Employee] ([ID\_Employee])

GO

ALTER TABLE [dbo].[Package] CHECK CONSTRAINT [FK\_Package\_Employee]

GO

ALTER TABLE [dbo].[Package] WITH CHECK ADD CONSTRAINT [FK\_Package\_Recipient] FOREIGN KEY([Recipient\_ID])

REFERENCES [dbo].[Recipient] ([ID\_Recipient])

GO

ALTER TABLE [dbo].[Package] CHECK CONSTRAINT [FK\_Package\_Recipient]

GO

ALTER TABLE [dbo].[Package] WITH CHECK ADD CONSTRAINT [FK\_Package\_Sender] FOREIGN KEY([Sender\_ID])

REFERENCES [dbo].[Sender] ([ID\_Sender])

GO

ALTER TABLE [dbo].[Package] CHECK CONSTRAINT [FK\_Package\_Sender]

GO

ALTER TABLE [dbo].[Package] WITH CHECK ADD CONSTRAINT [FK\_Package\_View] FOREIGN KEY([View\_ID])

REFERENCES [dbo].[View] ([ID\_View])

GO

ALTER TABLE [dbo].[Package] CHECK CONSTRAINT [FK\_Package\_View]

GO

USE [master]

GO

ALTER DATABASE [Service] SET READ\_WRITE

GO

# ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММЫ

**CommonMethods.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Service.Context;

using Service.Interfaces;

namespace Service.Methods

{

public class CommonMethods

{

/// <summary>

/// Удалеие записей по ID

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="ids">список id</param>

/// <returns></returns>

public bool RemoveEntity<T>(List<Guid> ids) where T : class,Common

{

try

{

using(ProjectContext context = new ProjectContext())

{

var source = context.Set<T>().Where(o => ids.Contains(o.Id)).ToList();

context.Set<T>().RemoveRange(source);

context.SaveChanges();

return true;

}

}

catch(Exception exc)

{

MessageBox.Show(exc.ToString());

return false;

}

}

/// <summary>

/// Достать запись из БД по ИД

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="id"></param>

/// <param name="context"></param>

/// <returns></returns>

public T GetEntityById<T>(Guid id, ProjectContext context) where T : class, Common

{

try

{

var source = context.Set<T>().Where(o=> o.Id == id).FirstOrDefault();

return source;

}

catch(Exception exc)

{

MessageBox.Show(exc.ToString());

return default(T);

}

}

/// <summary>

/// Найти все записи с пожим именем

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="search">строка поиска</param>

/// <returns></returns>

public List<T> SearchEntities<T>(string search) where T : class, Common

{

try

{

using (ProjectContext context = new ProjectContext())

{

var source = context.Set<T>().ToList();

if(!string.IsNullOrEmpty(search) && !string.IsNullOrEmpty(search))

{

source = source.Where(o => o.Name.ToLower().Contains(search.ToLower())).ToList();

}

return source;

}

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

return new List<T>();

}

}

/// <summary>

/// Найти все записи с пожим именем и взять определенное количество

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="search">строка поиска</param>

/// <param name="take">сколько записей взять</param>

/// <returns></returns>

public List<T> SearchEntities<T>(string search,int take) where T : class, Common

{

try

{

using (ProjectContext context = new ProjectContext())

{

List<T> result = new List<T>();

var source = context.Set<T>().Where(o=> true);

if (!string.IsNullOrEmpty(search) && !string.IsNullOrEmpty(search))

{

source = source.Where(o => o.Name.ToLower().Contains(search.ToLower()));

}

result = source.Take(take).ToList();

return result;

}

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

return new List<T>();

}

}

/// <summary>

/// Найти все записи и вернуть типом IQueryable

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="search">строка поиска</param>

/// <returns></returns>

public IQueryable<T> SearchEntitiesIQueryable<T>(string search) where T : class, Common

{

try

{

using (ProjectContext context = new ProjectContext())

{

var source = context.Set<T>().Where(o=> true);

if (!string.IsNullOrEmpty(search) && !string.IsNullOrEmpty(search))

{

source = source.Where(o => o.Name.ToLower().Contains(search.ToLower()));

}

return source;

}

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

return null;

}

}

/// <summary>

/// Найти все записи, но не создавать новый context

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="context"></param>

/// <param name="search"></param>

/// <returns></returns>

public List<T> SearchEntities<T>(ProjectContext context, string search) where T : class, Common

{

try

{

var deletedIds = context.ChangeTracker.Entries<T>().Where(o => o.State == System.Data.Entity.EntityState.Deleted).Select(o => o.Entity.Id).ToList();

var source = context.Set<T>().Where(o=> !deletedIds.Contains(o.Id)).ToList();

if (!string.IsNullOrEmpty(search) && !string.IsNullOrEmpty(search))

{

source = source.Where(o => o.Name.ToLower().Contains(search.ToLower())).ToList();

}

return source;

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

return new List<T>();

}

}

/// <summary>

/// Найти все записи, которые не были удалены

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="context"></param>

/// <param name="search"></param>

/// <returns></returns>

public List<T> SearchEntitiesWithoutContains<T>(ProjectContext context, string search) where T : class, Common

{

try

{

var deletedIds = context.ChangeTracker.Entries<T>().Where(o => o.State == System.Data.Entity.EntityState.Deleted).Select(o => o.Entity.Id).ToList();

var source = context.Set<T>().Where(o => !deletedIds.Contains(o.Id)).ToList();

if (!string.IsNullOrEmpty(search) && !string.IsNullOrEmpty(search))

{

source = source.Where(o => o.Name.ToLower() == search.ToLower()).ToList();

}

return source;

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

return new List<T>();

}

}

/// <summary>

/// Записать данные в тиблицу

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="context"></param>

/// <param name="search">строка поиска</param>

/// <param name="table">таблица</param>

/// <returns></returns>

public bool BindDataTable<T>(ProjectContext context, string search, DataGridView table) where T : class, Common

{

try

{

var models = SearchEntities<T>(context,search);

table.DataSource = new BindingList<T>(models);

return true;

}catch(Exception exc)

{

MessageBox.Show(exc.ToString());

return false;

}

}

/// <summary>

/// Пометить запись как удаленную

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="entity">запись</param>

/// <param name="context"></param>

/// <returns></returns>

public bool SetEntityAsRemoved<T>(object entity,ProjectContext context) where T : class, Common

{

try

{

T item = (T)entity;

context.Entry(item).State = System.Data.Entity.EntityState.Deleted;

return true;

}catch(Exception exc)

{

MessageBox.Show(exc.ToString());

return false;

}

}

/// <summary>

/// Найти запись по id и пометить как удаленную

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="id"></param>

/// <param name="context"></param>

/// <returns></returns>

public bool SetEntityAsRemovedById<T>(Guid id, ProjectContext context) where T : class, Common

{

try

{

var item = GetEntityById<T>(id,context);

context.Entry(item).State = System.Data.Entity.EntityState.Deleted;

return true;

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

return false;

}

}

/// <summary>

/// Заполнить данные комбобокса в соответствии с переданным поиском

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="search"></param>

/// <param name="select">комбобокс</param>

/// <returns></returns>

public bool FillCombobox<T>(string search, ComboBox select) where T : class, Common

{

try

{

var entities = SearchEntities<T>(search,10);

select.DisplayMember = "Name";

select.ValueMember = "Id";

select.DataSource = entities;

return true;

}catch(Exception exc)

{

MessageBox.Show(exc.ToString());

return false;

}

}

/// <summary>

/// Заполнить данные комбобокса в соответствии с переданным поиском и добавить туда переданную запись

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="search"></param>

/// <param name="select">комбобокс</param>

/// <param name="needToAdd">запись, которую необходимо добавить</param>

/// <returns></returns>

public bool FillCombobox<T>(string search, ComboBox select,T needToAdd) where T : class, Common

{

try

{

var entities = SearchEntities<T>(search, 10);

select.DisplayMember = "Name";

select.ValueMember = "Id";

entities.Add(needToAdd);

select.DataSource = entities;

return true;

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

return false;

}

}

/// <summary>

/// Добавить новую запись в БД

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="context"></param>

/// <param name="entity">сущность, которую нужно добавить</param>

/// <returns></returns>

public bool AddNewEntity<T>(ProjectContext context,T entity)where T: class, Common

{

try

{

if(string.IsNullOrEmpty(entity.Name) || string.IsNullOrWhiteSpace(entity.Name))

{

MessageBox.Show("Необходимо указать наименование!");

}

if(SearchEntitiesWithoutContains<T>(context,entity.Name).Count > 0)

{

MessageBox.Show("Нельзя создавать записи с одинаковым именем!");

return false;

}

context.Set<T>().Add(entity);

context.SaveChanges();

return true;

}catch(Exception exc)

{

MessageBox.Show(exc.ToString());

return false;

}

}

/// <summary>

/// Добавить новую запись и очистить поля ввода на переданной панеле

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="context"></param>

/// <param name="entity">сущность, которую нужно добавить</param>

/// <param name="panel">панель</param>

/// <returns></returns>

public bool AddNewEntity<T>(ProjectContext context, T entity,Panel panel) where T : class, Common

{

try

{

if (string.IsNullOrEmpty(entity.Name) || string.IsNullOrWhiteSpace(entity.Name))

{

MessageBox.Show("Необходимо указать наименование!");

return false;

}

if (SearchEntitiesWithoutContains<T>(context, entity.Name).Count > 0)

{

MessageBox.Show("Нельзя создавать записи с одинаковым именем!");

return false;

}

context.Set<T>().Add(entity);

context.SaveChanges();

foreach (Control contr in panel.Controls)

{

TextBox tb = contr as TextBox;

if (tb != null)

{

tb.Text = string.Empty;

}

}

return true;

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString());

return false;

}

}

/// <summary>

/// Получить ид записи, которая выбрана в таблице

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="table">таблица</param>

/// <returns></returns>

public Guid GetIdFromTable<T>(DataGridView table) where T : class, Common

{

try

{

var selectedItem = table.SelectedRows[0].DataBoundItem as T;

return selectedItem.Id;

}catch(Exception exc)

{

MessageBox.Show(exc.ToString());

return Guid.Empty;

}

}

/// <summary>

/// Очистить поля ввода на панеле

/// </summary>

/// <param name="panel">панель</param>

public void ClearControl(Panel panel)

{

foreach (Control contr in panel.Controls)

{

TextBox tb = contr as TextBox;

if (tb != null)

{

tb.Text = string.Empty;

}

}

}

/// <summary>

/// Поменять текст в кнопке редактирования и добавления

/// </summary>

/// <param name="editButton"></param>

/// <param name="addButton"></param>

/// <param name="isEditing"></param>

public void SwitchButtonsStates(Button editButton,Button addButton, bool isEditing)

{

if (isEditing)

{

editButton.Text = "Отменить редактирование";

addButton.Text = "Подтвердить";

}

else

{

editButton.Text = "Редактировать выбранную запись";

addButton.Text = "Добавить";

}

}

/// <summary>

/// Поменять текст в кнопке редактирования и добавления и очистить переданную панель

/// </summary>

/// <param name="editButton"></param>

/// <param name="addButton"></param>

/// <param name="panel"></param>

/// <param name="isEditing"></param>

public void SwitchEditButtonStates(Button editButton, Button addButton, Panel panel, ref bool isEditing)

{

isEditing = !isEditing;

SwitchButtonsStates(editButton, addButton, isEditing);

ClearControl(panel);

}

/// <summary>

/// Заполнить комбобокс записями, но если нет необходимой записи добавить ее (нужно для редактирования, т.к. из в 10 загружаемых записях может не быть связанной с сущностью)

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="Table"></param>

/// <param name="context"></param>

/// <param name="comboBox"></param>

/// <param name="searchId"></param>

/// <param name="searchTextBox"></param>

/// <returns></returns>

public bool SetValuesToComboBox<T>(ProjectContext context, ComboBox comboBox, Guid searchId, TextBox searchTextBox) where T : class, Common

{

bool wasSelected = false;

foreach (var item in comboBox.Items)

{

var converted = item as T;

if (converted.Id == searchId)

{

comboBox.SelectedItem = item;

wasSelected = true;

break;

}

}

if (!wasSelected)

{

var entity = GetEntityById<T>(searchId, context);

FillCombobox<T>(searchTextBox.Text, comboBox, entity);

comboBox.SelectedItem = entity as object;

}

return true;

}

}

}

**ModelsMethods.cs**

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Service.Context;

using Service.Models;

using Service.ViewModels;

namespace Service.Methods

{

public class ModelsMethods

{

CommonMethods commonMethods = new CommonMethods();

public Func<Department,DepartmentsViewModel> ConvertDepartmentModel(ProjectContext context)

{

return result => new DepartmentsViewModel

{

Id= result.Id,

Name = result.Name,

GroupName = commonMethods.GetEntityById<Group>(result.GroupId,context).Name

};

}

public Func<Employee, EmployeeViewModel> ConvertEmployeeModel(ProjectContext context)

{

return result => new EmployeeViewModel

{

Id = result.Id,

Name = result.Name,

DepartmentName = commonMethods.GetEntityById<Department>(result.DepartmentId, context).Name,

PostName = commonMethods.GetEntityById<Post>(result.PostId, context).Name,

PostId = result.PostId,

DepartmentId = result.DepartmentId

};

}

public Func<Product, ProductViewModel> ConvertProductModel(ProjectContext context)

{

return result => new ProductViewModel

{

Id = result.Id,

Name = result.Name,

Amount = result.Amount.ToString(),

BuyingPrice = result.BuyingPrice.ToString(),

GroupName = commonMethods.GetEntityById<Group>(result.GroupId, context).Name,

ManufacturerName = commonMethods.GetEntityById<Manufacturer>(result.ManufacturerId, context).Name,

ProviderName = commonMethods.GetEntityById<Provider>(result.ProviderId, context).Name,

SellingPrice = result.SellingPrice.ToString(),

UnitName = commonMethods.GetEntityById<Unit>(result.UnitId, context).Name

};

}

public Func<Product, ProductViewModel> ConvertProductModel()

{

return result => new ProductViewModel

{

Id = result.Id,

Name = result.Name,

/\*DepartmentName = commonMethods.GetEntityById<Department>(result.DepartmentId, context).Name,

PostName = commonMethods.GetEntityById<Post>(result.PostId, context).Name,

PostId = result.PostId,

DepartmentId = result.DepartmentId\*/

Amount = result.Amount.ToString(),

BuyingPrice = result.BuyingPrice.ToString(),

GroupName = result.Group.Name,

ManufacturerName = result.Manufacturer.Name,

ProviderName = result.Provider.Name,

SellingPrice = result.SellingPrice.ToString(),

UnitName = result.Unit.Name

};

}

#region Подразделения

public bool SetDepartmentComboBox(DataGridView Table,ProjectContext context, ComboBox Departments,Employee currentEntity, TextBox SearchDepartmentName)

{

bool wasSelected = false;

foreach (var item in Departments.Items)

{

var converted = item as Department;

if (converted.Id == currentEntity.DepartmentId)

{

Departments.SelectedItem = item;

wasSelected = true;

break;

}

}

if (!wasSelected)

{

var entity = commonMethods.GetEntityById<Department>(currentEntity.DepartmentId, context);

commonMethods.FillCombobox<Department>(SearchDepartmentName.Text, Departments, entity);

Departments.SelectedItem = entity as object;

}

return true;

}

#endregion

#region Должности

public bool SetPostsComboBox(DataGridView Table, ProjectContext context, ComboBox Posts, Employee currentEntity, TextBox SearchPostName)

{

bool wasSelected = false;

foreach (var item in Posts.Items)

{

var converted = item as Post;

if (converted.Id == currentEntity.PostId)

{

Posts.SelectedItem = item;

wasSelected = true;

break;

}

}

if (!wasSelected)

{

var entity = commonMethods.GetEntityById<Post>(currentEntity.DepartmentId, context);

commonMethods.FillCombobox<Post>(SearchPostName.Text, Posts, entity);

Posts.SelectedItem = entity as object;

}

return true;

}

#endregion

#region Подразделения

public bool SetUnitComboBox(DataGridView Table, ProjectContext context, ComboBox Units, Product currentEntity, TextBox searchTextBox)

{

bool wasSelected = false;

foreach (var item in Units.Items)

{

var converted = item as Unit;

if (converted.Id == currentEntity.UnitId)

{

Units.SelectedItem = item;

wasSelected = true;

break;

}

}

if (!wasSelected)

{

var entity = commonMethods.GetEntityById<Unit>(currentEntity.UnitId, context);

commonMethods.FillCombobox<Unit>(searchTextBox.Text, Units, entity);

Units.SelectedItem = entity as object;

}

return true;

}

#endregion

#region Подразделения

public bool SetProviderComboBox(DataGridView Table, ProjectContext context, ComboBox Providers, Product currentEntity, TextBox searchTextBox)

{

bool wasSelected = false;

foreach (var item in Providers.Items)

{

var converted = item as Provider;

if (converted.Id == currentEntity.ProviderId)

{

Providers.SelectedItem = item;

wasSelected = true;

break;

}

}

if (!wasSelected)

{

var entity = commonMethods.GetEntityById<Provider>(currentEntity.ProviderId, context);

commonMethods.FillCombobox<Provider>(searchTextBox.Text, Providers, entity);

Providers.SelectedItem = entity as object;

}

return true;

}

#endregion

#region Подразделения

public bool SetManufacturerComboBox(DataGridView Table, ProjectContext context, ComboBox Manufacturers, Product currentEntity, TextBox searchTextBox)

{

bool wasSelected = false;

foreach (var item in Manufacturers.Items)

{

var converted = item as Manufacturer;

if (converted.Id == currentEntity.ManufacturerId)

{

Manufacturers.SelectedItem = item;

wasSelected = true;

break;

}

}

if (!wasSelected)

{

var entity = commonMethods.GetEntityById<Manufacturer>(currentEntity.ManufacturerId, context);

commonMethods.FillCombobox<Manufacturer>(searchTextBox.Text, Manufacturers, entity);

Manufacturers.SelectedItem = entity as object;

}

return true;

}

#endregion

#region Подразделения

public bool SetGroupComboBox(DataGridView Table, ProjectContext context, ComboBox Groups, Product currentEntity, TextBox searchTextBox)

{

bool wasSelected = false;

foreach (var item in Groups.Items)

{

var converted = item as Group;

if (converted.Id == currentEntity.GroupId)

{

Groups.SelectedItem = item;

wasSelected = true;

break;

}

}

if (!wasSelected)

{

var entity = commonMethods.GetEntityById<Group>(currentEntity.GroupId, context);

commonMethods.FillCombobox<Group>(searchTextBox.Text, Groups, entity);

Groups.SelectedItem = entity as object;

}

return true;

}

#endregion

}

}

**Package.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Service.Interfaces;

namespace Service.Models

{

public class Package: Common

{

public Guid Id { get; set; }

public string Name { get; set; }

public int Amount { get; set; }

/// <summary>

/// Стоимость

/// </summary>

public float BuyingPrice { get; set; }

/// <summary>

/// Вес

/// </summary>

public float SellingPrice { get; set; }

public Guid GroupId { get; set; }

public Group Group { get; set; }

public Guid UnitId { get; set; }

public Unit Unit { get; set; }

public Guid ProviderId { get; set; }

public Provider Provider { get; set; }

public Guid ManufacturerId { get; set; }

public Manufacturer Manufacturer { get; set; }

}

}

**PackageForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Entity;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Service.Context;

using Service.Methods;

using Service.Models;

using Service.ViewModels;

namespace Service.Forms

{

public partial class ProductsForm : Form

{

public ProductsForm()

{

InitializeComponent();

}

ProjectContext context = new ProjectContext();

CommonMethods commonMethods = new CommonMethods();

ModelsMethods modelsMethods = new ModelsMethods();

public bool isEditing = false;

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

{

LoadTable();

}

private void LoadTable()

{

var deletedIds = context.ChangeTracker.Entries<Product>().Where(o => o.State == System.Data.Entity.EntityState.Deleted).Select(o => o.Entity.Id).ToList();

var updatedIds = context.ChangeTracker.Entries<Product>().Where(o => o.State == System.Data.Entity.EntityState.Modified).Select(o => o.Entity.Id).ToList();

List<ProductViewModel> list = new List<ProductViewModel>();

if (updatedIds.Count > 0)

{

var updatedEntities = context.ChangeTracker.Entries<Product>().Where(o => o.State == System.Data.Entity.EntityState.Modified).Select(o => o.Entity).Select(modelsMethods.ConvertProductModel(context)).ToList();

list = context.Products.Where(o => !deletedIds.Contains(o.Id) && !updatedIds.Contains(o.Id)).Include(o => o.Provider).Include(o => o.Unit).Include(o => o.Manufacturer).Include(o => o.Group)

.Select(modelsMethods.ConvertProductModel()).ToList();

list.AddRange(updatedEntities);

}

else

{

list = context.Products.Where(o => !deletedIds.Contains(o.Id)).Include(o => o.Provider).Include(o => o.Unit).Include(o => o.Manufacturer).Include(o => o.Group)

.Select(modelsMethods.ConvertProductModel()).ToList();

}

if (!string.IsNullOrEmpty(SearchNameInput.Text) || !string.IsNullOrWhiteSpace(SearchNameInput.Text))

{

list = list.Where(o => o.Name.ToLower().Contains(SearchNameInput.Text.ToLower())).ToList();

}

Table.DataSource = new BindingList<ProductViewModel>(list);

}

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

{

context.SaveChanges();

}

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

{

var unit = Units.SelectedItem as Unit;

if (unit == null)

{

MessageBox.Show("Некорректная единица измерения!");

return;

}

var provider = Providers.SelectedItem as Provider;

if (provider == null)

{

MessageBox.Show("Некорректный получатель!");

return;

}

var manufacturer = Manufacturers.SelectedItem as Manufacturer;

if (manufacturer == null)

{

MessageBox.Show("Некорректный производитель!");

return;

}

var Group = Groups.SelectedItem as Group;

if (Group == null)

{

MessageBox.Show("Некорректная категория!");

return;

}

if (!isFieldsValid())

{

return;

}

var amount = int.Parse(Amount.Text);

var buyingPrice = float.Parse(BuyingPrice.Text);

var sellingPrice = float.Parse(SellingPrice.Text);

if (isEditing)

{

Guid currentId = commonMethods.GetIdFromTable<ProductViewModel>(Table);

var currentEntity = commonMethods.GetEntityById<Product>(currentId, context);

currentEntity.Name = EntityName.Text;

currentEntity.Amount = int.Parse(Amount.Text);

currentEntity.SellingPrice = float.Parse(SellingPrice.Text);

currentEntity.BuyingPrice = float.Parse(BuyingPrice.Text);

currentEntity.UnitId = unit.Id;

currentEntity.ProviderId = provider.Id;

currentEntity.ManufacturerId = manufacturer.Id;

currentEntity.GroupId = Group.Id;

context.Entry<Product>(currentEntity).State = EntityState.Modified;

commonMethods.SwitchEditButtonStates(editEntity, AddNewEntity, AddEntityPanel, ref isEditing);

}

else

{

Product newEntity = new Product() { Id = Guid.NewGuid(), Name = EntityName.Text, UnitId = unit.Id, ProviderId = provider.Id, GroupId = Group.Id,ManufacturerId = manufacturer.Id,

Amount = amount, BuyingPrice = buyingPrice, SellingPrice = sellingPrice};

commonMethods.AddNewEntity<Product>(context, newEntity, AddEntityPanel);

}

commonMethods.ClearControl(AddEntityPanel);

LoadTable();

}

public bool isFieldsValid()

{

try

{

bool res = true;

int result;

float fl;

var amount = int.TryParse(Amount.Text,out result);

if (!amount)

{

res = false;

throw new Exception("Неверно введено количество!");

}

var buyingPrice = float.TryParse(BuyingPrice.Text,out fl);

if (!buyingPrice)

{

res = false;

throw new Exception("Неверно введена стоимость!");

}

var sellingPrice = float.TryParse(SellingPrice.Text,out fl);

if (!sellingPrice)

{

res = false;

throw new Exception("Неверно введен вес!");

}

return res;

}

catch(Exception exc)

{

MessageBox.Show(exc.Message);

return false;

}

}

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

{

commonMethods.FillCombobox<Unit>(string.Empty, Units);

commonMethods.FillCombobox<Group>(string.Empty, Groups);

commonMethods.FillCombobox<Manufacturer>(string.Empty, Manufacturers);

commonMethods.FillCombobox<Provider>(string.Empty, Providers);

LoadTable();

}

private void Table\_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)

{

var currentEntity = e.Row.DataBoundItem as ProductViewModel;

commonMethods.SetEntityAsRemovedById<Product>(currentEntity.Id, context);

}

private void Groups\_TextUpdate(object sender, EventArgs e)

{

commonMethods.FillCombobox<Group>(Groups.Text, Groups);

}

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

{

if (Table.SelectedRows.Count == 0)

{

MessageBox.Show("Ничего не выбрано!");

return;

}

commonMethods.SwitchEditButtonStates(editEntity, AddNewEntity, AddEntityPanel, ref isEditing);

if (isEditing)

{

Guid currentId = commonMethods.GetIdFromTable<ProductViewModel>(Table);

var currentEntity = context.Products.Where(o => o.Id == currentId).Include(o => o.Provider).Include(o => o.Unit).Include(o => o.Manufacturer).Include(o => o.Group).FirstOrDefault();

EntityName.Text = currentEntity.Name;

Amount.Text = currentEntity.Amount.ToString();

BuyingPrice.Text = currentEntity.BuyingPrice.ToString();

SellingPrice.Text = currentEntity.SellingPrice.ToString();

modelsMethods.SetUnitComboBox(Table, context, Units, currentEntity, UnitName);

modelsMethods.SetGroupComboBox(Table, context, Groups, currentEntity, GroupName);

modelsMethods.SetProviderComboBox(Table, context, Providers, currentEntity, ProviderName);

modelsMethods.SetManufacturerComboBox(Table, context, Manufacturers, currentEntity, ManufacturerName);

}

}

private void Table\_RowStateChanged(object sender, DataGridViewRowStateChangedEventArgs e)

{

if (e.StateChanged == DataGridViewElementStates.Displayed) return;

if (!isEditing) return;

commonMethods.SwitchEditButtonStates(editEntity, AddNewEntity, AddEntityPanel, ref isEditing);

}

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

{

commonMethods.FillCombobox<Group>(GroupName.Text, Groups);

}

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

{

commonMethods.FillCombobox<Provider>(ProviderName.Text, Providers);

}

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

{

commonMethods.FillCombobox<Unit>(UnitName.Text, Units);

}

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

{

commonMethods.FillCombobox<Manufacturer>(ManufacturerName.Text, Manufacturers);

}

}

}