**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ЯРОСЛАВСКОЙ ОБЛАСТИ**

Государственное профессиональное образовательное учреждение   
Ярославской области

Рыбинский полиграфический колледж

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Дипломный проект

|  |  |
| --- | --- |
| Разработка автоматизированной системы складского учета в магазине | |
| музыкальных инструментов | |
| специальность | 09.02.07 – Информационные системы и программирование |
|  | |

Пояснительная записка

|  |
| --- |
| ДП.0902.07.000000.00 ПЗ |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы | 4-ИС-2 |  |  |  | В.С. Киселев |
|  | *(Код)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Руководитель | преподаватель |  |  |  | С.В. Ермолычева |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Консультант по экономической части | преподаватель |  |  |  | М.Г. Кудряшова |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Консультант по охране труда | преподаватель |  |  |  | С.В. Ермолычева |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Нормоконтроль | преподаватель |  |  |  | Т.М. Моисеева |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Рецензент | преподаватель |  |  |  | С.О. Лаврентьев |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| К защите допустить |  |  |  |  |  |
| Зам. директора по учебной работе | |  |  |  | И.Н. Осокина |
|  |  |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |

г. Рыбинск

2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc137826767)

[1 Исследовательский раздел 4](#_Toc137826768)

[2 Конструкторский раздел 6](#_Toc137826769)

[2.1 Информационная модель 6](#_Toc137826770)

[2.2 Проектирование клиентской части системы 9](#_Toc137826771)

[2.3 Проектирование серверной части системы 14](#_Toc137826772)

[2.4 Разработка графического интерфейса 17](#_Toc137826773)

[3 Технологическая часть 18](#_Toc137826774)

[3.1 Тестирование и отладка. Составление тестового набора 18](#_Toc137826775)

[3.2 Инструкция по сопровождению и эксплуатации клиент-серверной системы 19](#_Toc137826776)

[4 Технико-экономический раздел 23](#_Toc137826777)

[5 Раздел охраны труда 34](#_Toc137826778)

[Заключение 36](#_Toc137826779)

[Список используемых источников 37](#_Toc137826780)

[Приложение А 38](#_Toc137826781)

Введение

В современном мире становится все меньше компаний, которые предпочитают хранить свои данные в бумажном виде, поскольку такие данные могут потеряться. Поэтому появляется спрос на ведение учета данных в электронном виде.

В рамках данного дипломного проекта планируется разработка автоматизированной системы складского учета в магазине музыкальных инструментов, которая поможет вести учет данных и будет максимально проста и понятна для неопытных пользователей, а также упростит поиск товара продавцам.

1 Исследовательский раздел

В ходе подготовки к разработке проекта были рассмотрены две среды программирования:

* Microsoft Visual Studio 2019;
* Borland Delphi 2007.

Рассмотрим функционал данных сред.

Microsoft Visual Studio 2019 – это условно-бесплатный продукт, разработанный компанией Microsoft. Данная среда легко подойдет для начинающих программистов, поскольку сами разработчики Microsoft ведут обучающий блог о языке программирования C#. Эта среда позволяет разрабатывать пользовательский интерфейс и устанавливать сторонние библиотеки и дает подсказки при написании кода. Также она позволяет отслеживать найденные ошибки и диагностировать поведение созданной программы. Минусами данного продукта можно считать зависимость от операционной системы и комплектующих компьютера[1].

Borland Delphi 2007 – это платный продукт, разработанный компанией Borland. Достоинствами данной среды является скорость разработки компонентов, принцип открытости и универсальность. Минусами можно считать отсутствие перегрузки операторов и объектной модели, малое число параметров для оптимизации кода, большие размеры exe-файлов, а также отсутствие подсказок при написании кода. Но при этом он подойдет разработчикам, поскольку синтаксис написания кода довольно прост и не зависит от комплектующих компьютера[2].

После тщательного выявления достоинств и недостатков данных сред программирования, была выбрана среда программирования Microsoft Visual Studio 2019, которая позволит реализовать клиент-серверную часть проекта.

Чтобы удостовериться в актуальности реализации проекта был организован поиск аналоговых работ. В конечном итоге было найдено множество программ, лидирующее место в которых занимал 1С. Но минусом данной программы была плата за установку на одного пользователя и обширный функционал, в котором трудно разобраться.

В конечном итоге, при анализе найденных аналогов, разрабатываемый программный продукт будет обладать главным достоинством – он будет включать в себя минимальный набор требуемых функций, а также в нем не будет ограничений на количество установок.

Для разработки проекта было принято решение использовать технологии WPF и языка C#, а в качестве системы управления базой данных была выбрана СУБД MSSQL.

2 Конструкторский раздел

2.1 Информационная модель

Чёрная сфера - используется для обозначения работы системы, внутреннее устройство которой неизвестно. В данную систему поступают входные данные, а выходят выходные данные. Черная сфера представлена на рисунке 2.1.



Рисунок 2.1 – Модель «черная сфера»

Представление разрабатываемого продукта в виде модели «черная сфера» показан на рисунке 2.2.



Рисунок 2.2 – Модель «черная сфера» с параметрами

Проведем первый этап декомпозиции для выявления структуры программного продукта. Декомпозиция позволит рассмотреть более детально все процессы в проекте, а также поможет уточнить ее структуру. Результат первого этапа декомпозиции представлен на рисунке 2.3.

Рисунок 2.3 – Результат первого этапа декомпозиции

На первом этапе декомпозиции были получены основные структуры. На схеме в виде прямоугольников входные и выходные параметры, методы показаны в виде овалов, а взаимодействие между процессами и данными в виде стрелок. Для получения более точных структур данных и процессов проведем второй этап декомпозиции. Результат второго этапа декомпозиции представлен на рисунке 2.4.



Рисунок 2.4 – Результат второго этапа декомпозиции

После третьего этапа декомпозиции получена конечная структура программного продукта. Конечный результат декомпозиции представлен на рисунке 2.5.



Рисунок 2.5 – Конечный этап декомпозиции

2.2 Проектирование клиентской части системы

2.2.1 Разработка модулей клиентской части системы

Программа будет иметь пару вспомогательных классов, которые помогут облегчить хранение информации в базе данных.

Один из классов – класс, отвечающий за перевод изображений в двоичный формат записи. В таблице 2.1 приведено описание полей класса ActionsWithPictures.

Таблица 2.1 – Описание полей класса ActionsWithPictures.

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| pathImages | string | Путь до изображения |

В таблице 2.2 приведено описание методов класса ActionsWithPictures.

Таблица 2.2 – Описание методов класса ActionsWithPictures.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Принимает | Возвращает | Описание |
| ConvertImageToBinary | Строку изображения | Двоичные данные | Переводит изображение в двоичный код |
| GetBase64ImageFromDb | Ключ изображения |  | Получает данные изображения |

2.2.2 Разработка функционально-модульной схемы клиентской части системы

Одним из главных этапов проектирования клиентской части системы является разработка функционально-модульной схемы, которая помогает определить вид модулей, их состав, а также назначение и характер использования системы.

В ходе проектирования была разработана функционально-модульная схема клиентской части системы, представленная на рисунке 2.6.



Рисунок 2.6 – Функционально-модульная схема клиентской части системы

Главный модуль системы будет являться основной формой.

Модуль просмотра продуктов будет служить для просмотра товаров без возможности вносить изменения данных о продуктах.

Модуль авторизации будет содержать в себе форму авторизации для доступа к базе данных.

Модуль редактирования продуктов будет содержать в себе список продуктов. Также данный модуль будет позволять вносить изменения в базе данных.

2.2.3 Разработка алгоритмов клиентской части системы

Исходя из описания модулей, можно понять, какие методы потребуются для создания продукта. Ниже приведен наиболее значимый метод.

Метод AddCommit вызывается при добавлении продукта и выполняет следующие действия:

* проверка корректности введенных данных;
* сохранение корректной информации в базе данных;
* добавление нового продукта в список;
* обновление списка.

Блок-схема его алгоритма приведена на рисунке 2.7.



Рисунок 2.7 – Блок-схема метода AddCommit

Метод RecordCopy вызывается при копировании продукта и выполняет следующие действия:

* проверка на выделенный продукт;
* сохранение корректной информации в базе данных;
* добавление нового продукта в список;
* обновление списка.

Блок-схема его алгоритма приведена на рисунке 2.8.



Рисунок 2.8 – Блок-схема метода RecordCopy

Метод RecordDelete вызывается при удалении продукта и выполняет следующие действия:

* проверка на выделенный продукт;
* изменение информации в базе данных;
* обновление списка.

Блок-схема его алгоритма приведена на рисунке 2.9.



Рисунок 2.9 – Блок-схема метода RecordDelete

2.3 Проектирование серверной части системы

Одним из главных этапов разработки клиент-серверной системы является изучение предметной области и ее сущностей.

В результате изучения предметной области были выявлены сущности и ее атрибуты, которые представлены в таблице.

В таблице 2.12 приведено описание сущности «Страны».

Таблица 2.12 – Описание атрибутов сущности «Страны»

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип столбца | Описание |
| id | int | Первичный ключ |
| name | nvarchar | Название страны |

В таблице 2.13 приведено описание сущности «Производители».

Таблица 2.13 – Описание атрибутов сущности «Производители»

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип столбца | Описание |
| id | int | Первичный ключ |
| name | nvarchar | Название производителя |
| countryId | int | Внешний ключ страны |

В таблице 2.14 приведено описание сущности «Модели».

Таблица 2.14 – Описание атрибутов сущности «Модели»

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип столбца | Описание |
| id | int | Первичный ключ |
| name | nvarchar | Название модели |
| year | int | Год модели |

В таблице 2.15 приведено описание сущности «Типы».

Таблица 2.15 – Описание атрибутов сущности «Типы»

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип столбца | Описание |
| id | int | Первичный ключ |
| name | nvarchar | Название типа |

В таблице 2.16 приведено описание сущности «Места».

Таблица 2.16 – Описание атрибутов сущности «Места»

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип столбца | Описание |
| id | int | Первичный ключ |
| name | nvarchar | Название места |

В таблице 2.17 приведено описание сущности «Пользователи».

Таблица 2.17 – Описание атрибутов сущности «Пользователи»

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип столбца | Описание |
| id | int | Первичный ключ |
| login | nvarchar | Логин |
| password | nvarchar | Пароль |

В таблице 2.18 приведено описание сущности «Продукты».

Таблица 2.18 – Описание атрибутов сущности «Продукты»

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип столбца | Описание |
| id | int | Первичный ключ |
| typeId | int | Внешний ключ типа |
| modelId | int | Внешний ключ модели |
| name | nvarchar | Название продукта |
| description | nvarchar | Описание продукта |
| manufacturerId | int | Внешний ключ производителя |
| cost | money | Цена продукта |
| count | int | Количество продукта |
| weight | float | Вес продукта |
| height | float | Высота продукта |
| width | float | Ширина продукта |
| length | float | Длина продукта |
| image | image | Изображение продукта |
| placeId | int | Место хранения |

В результате разработанной структуры сущностей предметной области была разработана схема базы данных, изображенная на рисунке 2.11.



Рисунок 2.11 – Реляционная схема БД

В результате анализа и нормализации реляционной схемы БД была разработана нормализованная схема базы данных, изображенная на рисунке 2.12.

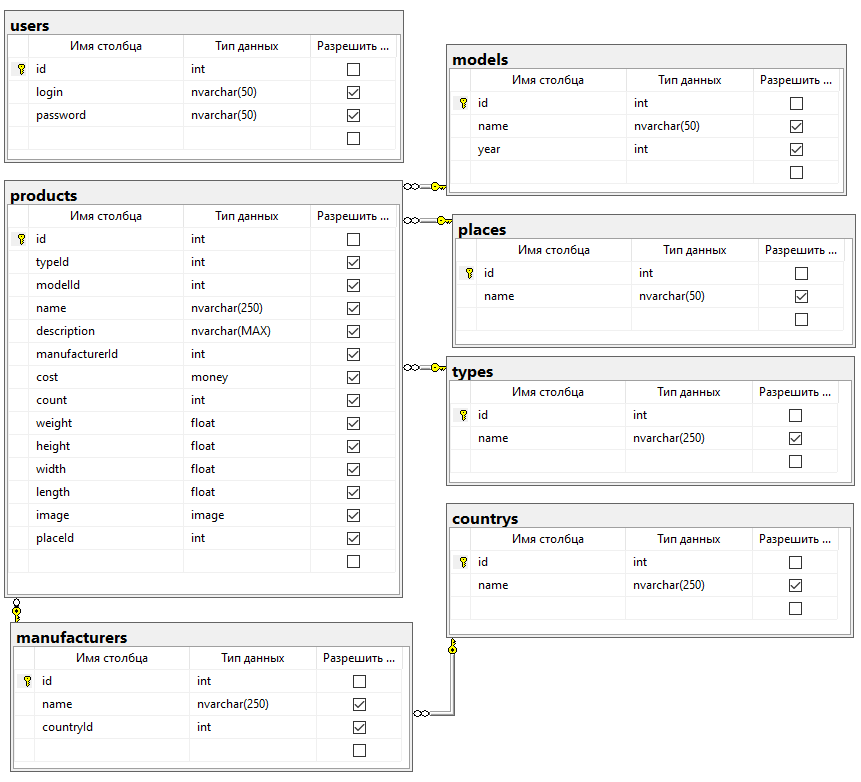


Рисунок 2.12 – Нормализованная схема базы данных

2.4 Разработка графического интерфейса

Для разрабатываемой программы будет продуман понятный и простой интерфейс, который позволит неопытному пользователю понять функциональность данной системы при помощи всплывающих подсказок, а также ошибок.

3 Технологическая часть

3.1 Тестирование и отладка. Составление тестового набора

Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки [3]. В связи с тем, что почти невозможно составить реальную программу без ошибок, и почти невозможно для достаточно сложной программы быстро найти и устранить все имеющиеся в ней ошибки, разумно уже при разработке программы на этапах алгоритмизации и программирования готовиться к обнаружению ошибок на стадии отладки принимать профилактические меры по их предупреждению [4].

Тестирование будет происходить через тест кейсы. Тест кейс – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. При передаче тестировщику тест-кейсов, он должен пройтись по всем его пунктам и выполнить описанные действия, которые должны привести к определенным результатам [5].

Тестовый набор данных отображен в таблице 3.1.

Таблица 3.1 – Набор тестов для программного продукта

|  |  |
| --- | --- |
| Ситуация | Результат воздействия |
| Подключение к базе данных, указывая неверные данные | Вывод на экране сообщения с ошибкой ввода данных |
| Попытка просмотра информации о продукте | Отображение данных продукта |
| Попытка добавления нового продукта, без заполнения данных | Вывод сообщения с ошибкой |
| Попытка удаления продукта | Вывод сообщения об удалении продукта |
| Вход в приложение без учетных данных | Отсутствие доступа к редактированию данных |
| Попытка редактирования продукта, при заполненных данных | Успешное создание нового продукта |
| Поиск продукта | Поиск работает корректно |

3.2 Инструкция по сопровождению и эксплуатации клиент-серверной системы

Для корректной работы программного продукта, предъявляются следующие требования к аппаратным ресурсам.

Рекомендуемые требования:

* ОЗУ 2Гб и более;
* CPU с тактовой частотой не ниже 2Ггц.

Требования к программным ресурсам:

* Microsoft Windows;
* .NetFramework 5 и выше.

Программа поставляется в виде zip-архива. Для ее установки необходимо разархивировать архив в нужную папку. Если все прошло удачно, то можно приступать к эксплуатации.

При запуске программы на экране появляется главное окно, как показано на рисунке 3.1.

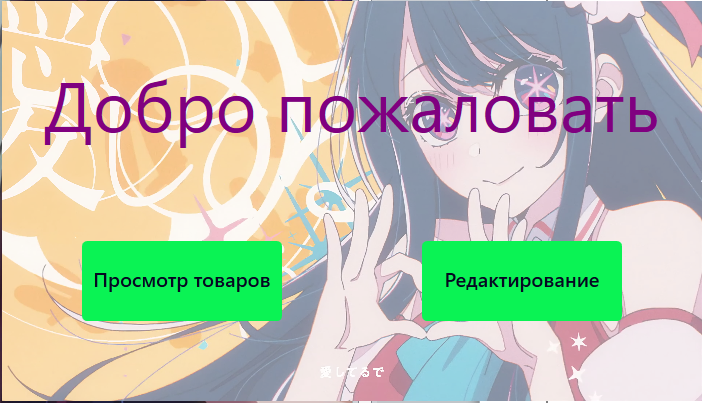


Рисунок 3.1 – Главное окно

При выборе кнопки «Редактирование» на экране появляется окно авторизации, оно показано на рисунке 3.2.

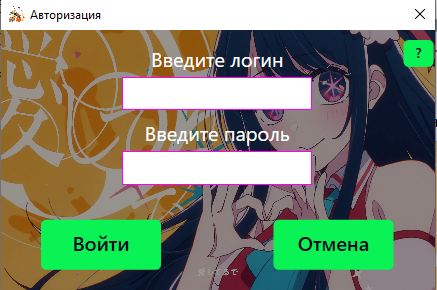


Рисунок 3.2 – Окно авторизации

При вводе корректных данных на экране появится окно редактирования продуктов, как на рисунке 3.3.

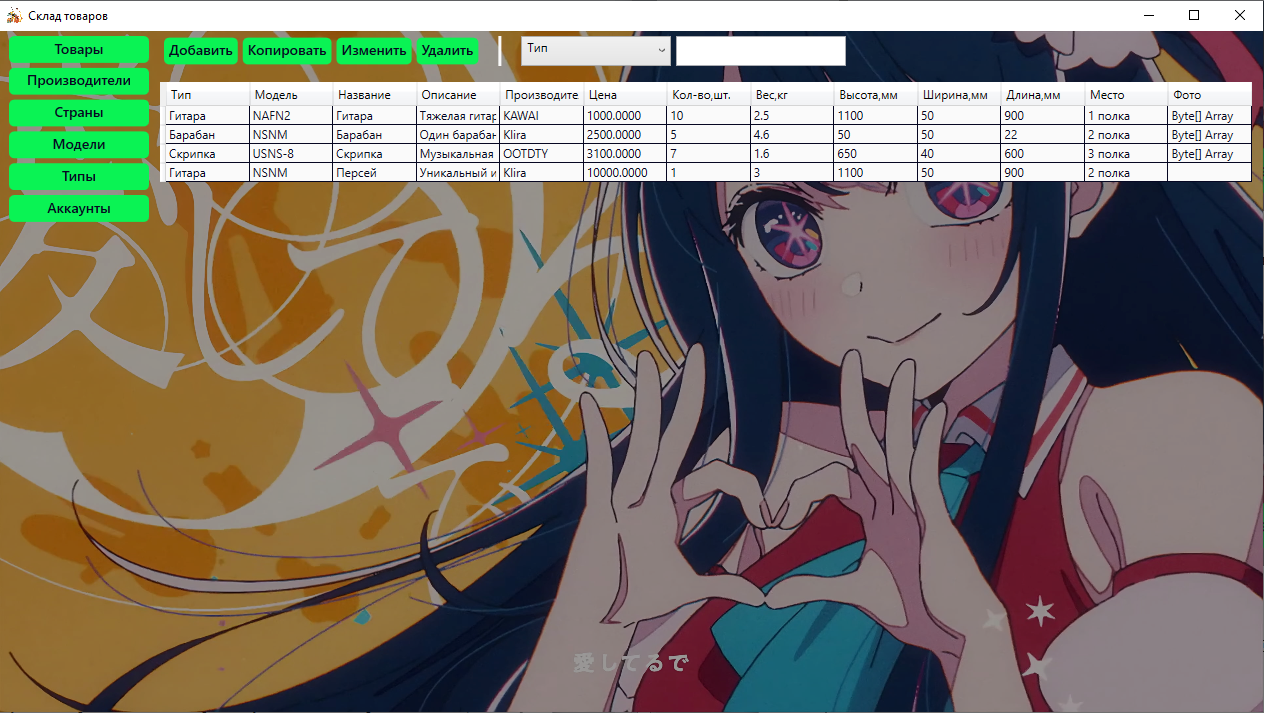


Рисунок 3.3 – Окно редактирования продуктов

На экране отображаются все данные о продуктах, при желании эти данные можно изменить, как на рисунке 3.4.

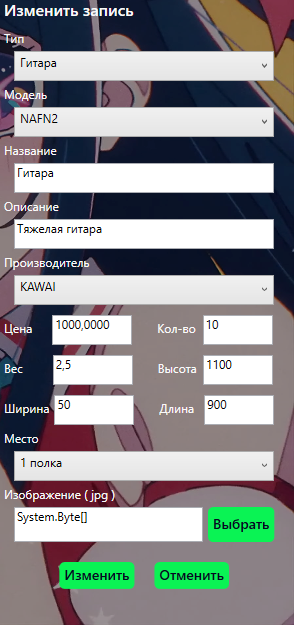


Рисунок 3.4 – Изменение данных продукта

Также на окне редактирования присутствуют вспомогательные справочники, они изображены на рисунке 3.5.

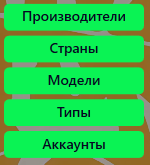


Рисунок 3.5 – Справочники окна редактирования

Справочник представляет собой страницу с расположенной в нём таблицей, заполненной данными, и меню с операциями управления данными и фильтрацией данных, как на рисунке 3.6.

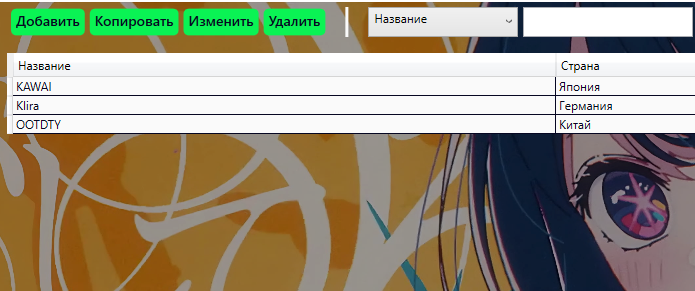


Рисунок 3.6 – Справочник производителей

Подобным образом выглядят справочники с типами, моделями, аккаунтами, странами.

4 Технико-экономический раздел

Основой расчета затрат на любой производственный процесс обычно является смета затрат. Смета затрат представляет собой сводный план всех расходов предприятия или организации на рассматриваемый период деятельности. Она определяет общую сумму издержек производства по видам используемых ресурсов, стадиям производственной деятельности, уровням управления предприятием и другим направлениям расходов. В смету включаются затраты основного и вспомогательного производства, связанные с изготовлением и продажей рассматриваемого продукта, а также на содержание административно-управленческого персонала, выполнение различных работ и услуг, в том числе и не входящих в основную производственную деятельность предприятия или организации.

Первым компонентом, входящим в сметный расчет, является материалы. В состав этих затрат принято включать стоимость материалов, которые будут проданы заказчику вместе с программным продуктом.

Поскольку кроме программного продукта никакие материалы продаваться не будут, то стоимость материалов будет рваным нулю.

Вторым компонентом, включаемым в сметный расчет, являются затраты на вспомогательные материалы. В состав этих затрат принято включать стоимость расходуемых за период работ покупных инструментов и малоценного хозяйственного инвентаря.

Совокупные затраты на вспомогательные материалы и малоценный инвентарь ВМ*,* руб. рассчитываются по формуле 4.1:

, (4.1)

где *ВМi* – стоимость, затраченная на каждый вспомогательный материал, руб.

Все вспомогательные материалы, используемые в рамках работы непосредственно над проектом, их количество и стоимость включены в таблицу 4.1.

Таблица 4.1 – Вспомогательные материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу, руб., коп. | Количество, ед. изм. | Стоимость,  руб., коп. |
| Бумажный лист формата А4. | 2,00 | 63 | 126,00 |
| Ватман формата А0, шт. | 150,00 | 3 | 450,00 |
| Картридж для принтера, шт. | 1490,00 | 0,2 | 298,00 |
| Итого |  |  | 874,00 |

Стоимость отдельного вспомогательного материала *ВМi,* руб. рассчитывается по формуле 4.2:

, (4.2)

где Ni – стоимость отдельного вспомогательного материала за штуку, руб.;

Ki – количество отдельного вспомогательного материала, руб.

Подстановкой указанных выше значений в формулу 4.1 получено:

Третьим компонентом затрат на разработку являются затраты на энергетические ресурсы ЭР, руб, состоящие из затрат на электроэнергию и затрат на топливо, и рассчитываемые по формуле 4.3:

, (4.3)

где Э – затраты на электроэнергию, руб.;

Т – затраты на топливо, руб.

Так как разработка программного продукта не требует затрат на топливо, то стоимость топлива принимается равным нулю.

Затраты на электроэнергию вычисляются по формуле 4.4:

, (4.4)

где W – совокупная потребленная мощность, КВт×ч;

ЦЭ – стоимость одного КВт×ч электроэнергии, руб.; составляет 3.62руб.

Совокупная потребленная мощность *W* рассчитывается по формуле 4.5:

, (4.5)

где Wi – мощность, потребляемая отдельным устройством, КВт.

В таблице 4.2 представлены все возможные устройства, используемые в разработке продукта, потребляющие электроэнергию.

Таблица 4.2 – Используемые устройства

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Потребляемая мощность, КВт | Продолжительность эксплуатации в смену, час. | Количество смен в цикле производства (разработки), шт. | Потребленная мощность, КВт×ч |
| Компьютер | 0,33 | 8 | 10 | 26,4 |
| Принтер | 0,34 | 0,5 | 10 | 1,7 |
| Итого |  |  |  | 28,1 |

Потребленная мощность *Wi*, кВт×ч для каждого из устройств рассчитывается по формуле 4.6:

, (4.6)

где Pi – мощность устройства, КВт;

ti – продолжительность эксплуатации устройства за одну рабочую смену. час.;

КСi – количество смен, в течение которых использовалось устройство, шт.

Подстановкой значений из таблицы 4.2 в формулу 4.5 получено значение потребленной мощности:

Подстановкой указанных выше значений в формулу 4.4, получаем затраты на электроэнергию:

Подстановкой указанных выше значений в формулу 4.3, получаем затраты на энергетические ресурсы:

Четвертым этапом сметного расчета является расчет амортизации использованного в проекте оборудования.

Амортизация – это процесс постепенного переноса стоимости средств, по мере их износа, на стоимость продукции, которая производится. Амортизация распространяется на основные средства, основные фонды и нематериальные активы. Она исчисляется линейным, нелинейным методом, способом уменьшения остаточной стоимости, способом списания стоимости по сумме лет использования.

Амортизационные отчисления – отчисления части стоимости основных средств для возмещения их износа. Амортизационные отчисления включаются в издержки производства и производятся коммерческими организациями на основе установленных норм и балансовой стоимости основных средств.

Общий объем амортизационных отчислений *АМ*, руб., включаемый в расчет по проекту для всех устройств можно вычислить по формуле 4.7:

, (4.7)

где *Аi* – амортизационное отчисление для отдельного основного средства, руб.

Для разработки программного продукта потребовалось оборудование и нематериальные активы, перечисленные в таблице 4.3.

Таблица 4.3 – Информация по расчету амортизационных отчислений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование основного средства | Изначальная стоимость, руб., коп. | Срок службы, лет | Сумма ежегодного амортизационного отчисления,  руб., коп. | Период использования в проекте, кален. дн. | Сумма амортизационного отчисления, руб., коп. |
| Стол | 500,00 | 4 | 125,00 | 10 | 0,0 |
| Стул | 500,00 | 5 | 125,00 | 10 | 0,0 |
| Ноутбук | 46 999,00 | 5 | 9399,00 | 10 | 0,0 |
| Visual Studio | 0,00 | 1 | 0,00 | 10 | 0,0 |
| Итого |  |  |  |  | 0,0 |

Амортизация не начисляется на оборудование ниже 100 000 руб., так как считается малоценным.

Следующим этапом сметного расчета является затраты на оплату труда.

Фонд оплаты труда ЗП, руб., включаемый в затраты по проекту, можно рассчитать по формуле 4.11:

, (4.11)

где ОЗП – основная заработная плата всех работников, участвовавших в проекте, руб.;

ДЗП – дополнительная заработная плата всех работников, участвовавших в проекте, руб.;

Основная заработная плата ОЗП, руб., включаемая в затраты проекта, будет складываться из двух частей, для расчёта можно воспользоваться формулой 4.12:

, (4.12)

где ОкЗП – сумма заработных плат всех рабочих, участвовавших в проекте, оплачиваемых по окладной системе оплаты труда, руб.;

СдЗП – сумма заработных плат всех рабочих, участвовавших в проекте, оплачиваемых по сдельной системе оплаты труда;

Заработная плата всех рабочих ОкЗП, руб., участвовавших в проекте, оплачиваемых по окладной системе оплаты труда, вычисляется по формуле 4.13:

, (4.13)

где ОкЗПi – заработная плата отдельного рабочего, участвовавшего в проекте, оплачиваемого по окладной системе оплаты труда, руб.

В таблице 4.4 сведены данные по работникам, принимавшим участие в проекте, оплачиваемым по окладной системе оплаты труда.

Таблица 4.4 – Заработная плата

|  |  |  |  |
| --- | --- | --- | --- |
| Должность, профессия работника | Месячный оклад, руб., коп. | Отработано смен, шт. | Заработная плата, руб., коп. |
| Инженер-программист | 50 000,00 | 10 | 24509,8 |
| Итого |  |  | 24509,8 |

В рамках проекта заработная плата ОкЗПi, руб., заработанная каждым из работников, оплачиваемых по окладной системе оплаты труда, рассчитывается по формуле 4.14:

, (4.14)

где Оi – оклад работника, руб.,

КСi – количество смен (рабочих дней), которое он работал по проекту, шт., составляет 15 дней;

20,4 – это среднемесячное количество смен (рабочих дней), шт.

Подстановкой указанных выше значений в формулу 4.13 получено:

Заработная плата всех рабочих, оплачиваемых по сдельной системе оплаты труда, , руб., рассчитывается по формуле 4.15:

, (4.15)

где СдЗПi – сумма заработных плат всех рабочих, участвовавших в проекте, оплачиваемых по сдельной системе оплаты труда, руб.

Так как в разработке проекта не участвовали рабочие, оплачиваемые по сдельной системе оплаты труда, сдельная заработная плата равна нулю.

Подстановкой указанных выше значений в формулу 4.12 получено:

Предполагается, что в оплату труда по проекту должна быть заложена в том числе, и дополнительная заработная плата ДЗП, которую необходимо будет выплачивать тем же самым работникам, например, в связи с очередным отпуском, то ее расчет будем осуществлять исходя из того, что ее размер должен составлять какую-то разумную обоснованную долю от основной заработной платы.

Известно, что большинство работников находятся в очередных плановых отпусках в среднем около одного месяца в году, то логично предположить, что доля дополнительной заработной платы ДЗП будет около 1/12 от основной заработной платы, что составляет приблизительно 8-9%. Логично допустить, что кроме отпусков работникам в соответствии с действующим законодательством предстоит делать еще какие-либо выплаты в виде дополнительной заработной платы. По этой причине полученную долю можно несколько увеличить и выбирать ее в размере 9-12%. Таким образом, сумму затрат на дополнительную заработную плату ДЗП, руб., можно рассчитать по формуле 4.16:

, (4.16)

где ОЗП – основная заработная плата всех работников, участвовавших в проекте, руб.;

КДЗП – коэффициент, %.

В рамках проекта КДЗП выбран равным 10%.

Подстановкой указанных выше значений в формулу 4.16 получено:

Подстановкой указанных выше значений в формулу 4.11 получено:

Шестым компонентом, включаемым в сметный расчет, являются страховые взносы.

Социальное страхование – это система социальной защиты, задача которой – обеспечивать реализацию конституционного права экономически активных граждан на материальное обеспечение в старости, в случае болезни, полной или частичной утраты трудоспособности, потери кормильца, безработицы.

Уплата страховых взносов является прямой обязанностью организации-работодателя, при этом часть выплат носит персональный характер. Так, например, выплат на лицевой счет каждого из работников в пенсионном фонде зависит от размера его персональной заработной платы.

Общий размер страховых взносов СВ, руб., необходимых для перечисления во все страховые фонды и включаемый в состав затрат при производстве, можно вычислить как некоторый процент от фонда заработной платы, для расчёта можно воспользоваться формулой 4.17:

, (4.17)

где ЗП – фонд оплаты труда по проекту, рассчитанный ранее (руб.), составляет руб.;

ССВ – ставка (размер налога) страховых взносов, %.

Ставка страховых взносов ССВ на 2023 год составляет 30%.

Подстановкой указанных выше значений в формулу 4.17 получено:

Следующим этапом сметного расчета является накладные расходы.

В зависимости от размеров предприятия или организации накладные расходы НР, руб., могут варьироваться в достаточно широком диапазоне. Поэтому в обычной практике в сметных расчетах их принято отражать в процентном отношении к заработной плате основных работников, для расчёта можно воспользоваться формулой 4.18:

, (4.18)

где ЗП – фонд оплаты труда по проекту, руб.;

СНР – размер накладных расходов, %.

Размер накладных расходов СНР составляет 10%.

Подстановкой указанных выше значений в формулу 4.18 получено:

Восьмым этапом сметного расчета является расчет себестоимости.

Себестоимость – это текущие затраты на производство товара или услуги, запуск их в обращение и реализацию. Расчет и анализ себестоимости продукции является важнейшей задачей любого предприятия и входит в систему управленческого учета, так как именно себестоимость лежит в основе большинства управленческих решений.

Себестоимость Сб, руб. программного продукта определяется по формуле 4.19.

*Сб* = *М* + ВМ + ЭР + АМ + *ЗП* + *СВ* +*НР,* (4.19)

где *М* – затраты на материалы руб.;

*ВМ* – затраты на вспомогательные материалы, руб.;

*ЭР* – энергетические затраты, руб.;

*АМ* – затраты на амортизацию, руб.;

*ЗП* – затраты на заработную плату, руб.;

*СВ* – страховые взносы, уплаченные работодателем с фонда оплаты труда, руб.;

*НР* – накладные расходы, возникающие как непредусмотренные всеми предшествующими статьями, руб.;

Подстановкой указанных выше значений в формулу 4.19 получено:

После расчета себестоимости (всех возможных затрат на разработку и производство), к ней добавляется необходимый объем прибыли, он и будет являться девятым продуктом сметного расчета.

Объем прибыли обычно определяется как процент от рассчитанной себестоимости. Объем прибыли П, руб. можно рассчитать по формуле 4.20:

, (4.20)

где Сб – себестоимость производства или разработки, руб.;

СП – процент прибыли, на которую рассчитывает организатор коммерческого предприятия, %.

Процент прибыли СП, на которую рассчитывает организатор коммерческого предприятия составляет 30%.

Подстановкой указанных выше значений в формулу 4.20 получено:

Следующим этапом сметного расчета является налог на добавленную стоимость.

Налог на добавленную стоимость (НДС) – косвенный налог, форма изъятия в бюджет государства части стоимости товара, работы или услуги, которая создаётся на всех стадиях процесса производства товаров, работ и услуг и вносится в бюджет по мере реализации.

Для расчета налога на добавленную стоимость НДС, руб. можно воспользоваться формулой 4.21:

, (4.21)

где Сб – себестоимость разработки, руб.;

П – прибыль, руб.;

СНДС – ставка налога на добавленную стоимость, %.

Ставка налога СНДС на добавленную стоимость составляет 20%.

Подстановкой указанных выше значений в формулу 4.21 получено:

Конечная стоимость разработки *С*, руб. рассчитывается как сумма себестоимости, прибыли и НДС, для расчёта можно воспользоваться формулой 4.22:

, (4.22)

где Сб – себестоимость разработки, руб.

П – прибыль. руб.;

НДС – сумма налога на добавленную стоимость, руб.;

Подстановкой указанных выше значений в формулу 4.22 получено:

Результаты расчетов показали, что конечная стоимость продукта составит руб., при этом продажа 10 копий ПП позволит снизить стоимость такой системы до руб.

5 Раздел охраны труда

Охрана труда – это целая система законодательных и нормативно-правовых актов, технических, гигиенических, лечебно-профилактических мероприятий и средств, которые обеспечивают безопасность, сохранение здоровья и работоспособности человека в процессе труда. В наши дни труд стал более интенсивным и требует огромных затрат умственной, эмоциональной и физической нагрузок.

На рабочем месте программист осуществляет трудовую деятельность и проводит большую часть рабочего времени. Правильная организация рабочего места программиста повышает производительность труда от 8 до 20%. Следуя рекомендациям ГОСТ 12.2.032-78, необходимо организовать рабочее место таким образом, чтобы взаимное расположение всех его элементов соответствовало физическим и психологическим требованиям. Главные элементы рабочего места программиста – это письменный стол и кресло. Рабочее место организуется в соответствии с ГОСТ 12.2.032-78, информация из работы [6].

Площадь рабочего места с компьютером с жидкокристаллическим или плазменным экраном должна быть не менее 4,5 кв. м, а расстояние между столами с мониторами (от тыла одного монитора до экрана другого) не менее 2 м. Монитор должен располагаться на расстоянии 50-70 см от глаз программиста. Параметры рабочего стола сотрудника: возможность регулировки высоты рабочего стола, или точная высота – 72,5 см, ширина – 80, 100, 120 или 140 см, глубина рабочего стола 80 или 100 см, высота и ширина пространства под столешницей (для ног) – не менее 50 см, глубина на уровне колен не менее 45 см, а на уровне вытянутых ног не менее 65 см.

Правильное освещение рабочего места – это очень важный момент в трудовой деятельности человека, влияющий на эффективность труда, при этом такой момент предупреждает травматизм и профессиональные заболевания. При недостаточном освещении приходится напрягать зрение, при этом ослабляется внимание и это приводит к наступлению преждевременной утомленности. Слишком яркое освещение тоже плохо, так как оно вызывает ослепление, раздражение и резь в глазах. При искусственном освещении, источниками света служат два вида ламп: лампы накаливания и люминесцентные.

Известно, что шум ухудшает условия труда и оказывает вредное воздействие на организм человека. Согласно ГОСТ 12.1.003-88 «Шум для помещений расчетчиков и программистов, уровни шума не должны превышать соответственно: 71, 61, 54, 49, 45, 42, 40, 38 дБ», информация из работы [7].

При работе компьютерной техники выделяется много тепла, что может привести к пожароопасной ситуации. Источниками зажигания так же могут служить приборы, применяемые для технического обслуживания, устройства электропитания, кондиционеры воздуха. Серьёзную опасность представляют различные электроизоляционные материалы, используемые для защиты от механических воздействий отдельных радиодеталей. В связи с этим, участки, на которых используется компьютерная техника, по пожарной опасности относятся к категории пожароопасных «В». При пожаре люди должны покинуть помещение в течение минимального времени. В помещениях с компьютерной техникой, недопустимо применение воды и пены ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования. Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях, информация из работы [8].

Заключение

По результатам дипломного проекта была разработана автоматизированная система складского учета в магазине музыкальных инструментов.

Главным достоинством можно выделить простой и понятный интерфейс, а также отзывчивость со стороны программы.

Список используемых источников

1. <https://ruprogi.ru/software/visual-studio>
2. <https://blog.skillfactory.ru/glossary/delphi/>
3. <https://studopedia.ru/22_29871_neobhodimost-otladki-programmnogo-produkta.html>
4. <https://infopedia.su/4x1ec5.html>
5. <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-4-ch3.html>
6. <https://xn--d1aux.xn--p1ai/opisanie-rabochego-mesta-programmista-na-predpriyatii/>
7. <https://xn--d1aux.xn--p1ai/opisanie-rabochego-mesta-programmista-na-predpriyatii/>
8. <https://studopedia.ru/8_107307_osveshchenie-pomeshcheniy-vichislitelnih-tsentrov.html>

Приложение А

Программный код класса ActionsWithPictures

internal class ActionsWithPictures

{

public static string pathImages = @"";

public static byte[] ConvertImageToBinary(string iFile)

{

FileInfo fInfo = new FileInfo(iFile);

long numBytes = fInfo.Length;

FileStream fStream = new FileStream(iFile, FileMode.Open, FileAccess.Read);

BinaryReader br = new BinaryReader(fStream);

byte[] imageData = br.ReadBytes((int)numBytes);

return imageData;

}

public static void GetBase64ImageFromDb(int id)

{

if (File.Exists($"{pathImages}MTWImage\_{id}.jpg")) return;

List<byte[]> iScreen = new List<byte[]>();

using (SqlConnection sqlConnection = new SqlConnection(@"data source=LAPTOP-GRBD40RP;initial catalog=MTW;integrated security=True"))

{

sqlConnection.Open();

SqlCommand sqlCommand = new SqlCommand();

sqlCommand.Connection = sqlConnection;

sqlCommand.CommandText = $"SELECT image FROM products WHERE id = {id}";

SqlDataReader sqlReader = sqlCommand.ExecuteReader();

byte[] iTrimByte = null;

while (sqlReader.Read())

{

iTrimByte = (byte[])sqlReader["image"];

iScreen.Add(iTrimByte);

}

sqlConnection.Close();

}

// конвертируем бинарные данные в изображение

byte[] imageData = iScreen[0];

MemoryStream ms = new MemoryStream(imageData);

System.Drawing.Image newImage = System.Drawing.Image.FromStream(ms);

// сохраняем изоражение на диск

string imageName = @"" + pathImages + "MTWImage\_" + id + ".jpg";

newImage.Save(imageName, ImageFormat.Jpeg);

}

}

Программный код класса WelcomeWindow

public partial class WelcomeWindow : Window

{

public WelcomeWindow()

{

InitializeComponent();

}

private void AuthBtn\_Click(object sender, RoutedEventArgs e)

{

AuthorizationWindow window = new AuthorizationWindow();

Close();

window.Show();

}

private void SeeBtn\_Click(object sender, RoutedEventArgs e)

{

ProductsWindow window = new ProductsWindow();

Close();

window.Show();

}

}

Программный код класса AuthorizationWindow

public partial class AuthorizationWindow : Window

{

public AuthorizationWindow()

{

InitializeComponent();

}

private void AuthCommit(object sender, RoutedEventArgs e)

{

Base.users user = SourceCore.db.users.SingleOrDefault(U => U.login == LoginText.Text && U.password == PasswordText.Text);

if (user != null)

{

Warehouse.WarehouseWindow wh = new Warehouse.WarehouseWindow();

Close();

wh.Show();

}

else

{

MessageBox.Show("Неверно указан логин и/или пароль!", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

};

}

private void AuthRollBack(object sender, RoutedEventArgs e)

{

WelcomeWindow ww = new WelcomeWindow();

Close();

ww.Show();

}

}

Программный код класса ProductsWindow

public partial class ProductsWindow : Window

{

private int DlgMode;

private Base.products SelectedProduct;

private ObservableCollection<Base.products> Products;

public ProductsWindow()

{

InitializeComponent();

DataContext = this;

DlgLoad(false, "");

UpdateGrid(null);

}

private void LoadingProd(object sender, RoutedEventArgs e)

{

List<string> Columns = new List<string>();

for (int i = 0; i < 12; i++)

{

Columns.Add(PageGrid.Columns[i].Header.ToString());

}

FilterComboBox.ItemsSource = Columns;

FilterComboBox.SelectedIndex = 0;

foreach (DataGridColumn Column in PageGrid.Columns)

{

Column.CanUserSort = false;

}

}

private void UpdateGrid(Base.products product)

{

if ((product == null) && (PageGrid.ItemsSource != null))

{

product = (Base.products)PageGrid.SelectedItem;

}

Products = new ObservableCollection<Base.products>(SourceCore.db.products);

PageGrid.ItemsSource = Products;

PageGrid.ItemsSource = SourceCore.db.products.ToList();

PageGrid.SelectedItem = product;

}

private void DlgLoad(bool b, string DlgModeContent)

{

if (b == true)

{

ColumnChange.Width = new GridLength(300);

PageGrid.IsHitTestVisible = false;

}

else

{

ColumnChange.Width = new GridLength(0);

PageGrid.IsHitTestVisible = true;

DlgMode = -1;

}

}

private void FilterTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var textbox = sender as TextBox;

switch (FilterComboBox.SelectedIndex)

{

case 0:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.types.name.ToString().Contains(textbox.Text)).ToList();

break;

case 1:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.models.name.ToString().Contains(textbox.Text)).ToList();

break;

case 2:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.name.Contains(textbox.Text)).ToList();

break;

case 3:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.description.Contains(textbox.Text)).ToList();

break;

case 4:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.manufacturers.name.ToString().Contains(textbox.Text)).ToList();

break;

case 5:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.cost.ToString().Contains(textbox.Text)).ToList();

break;

case 6:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.count.ToString().Contains(textbox.Text)).ToList();

break;

case 7:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.weight.ToString().Contains(textbox.Text)).ToList();

break;

case 8:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.height.ToString().Contains(textbox.Text)).ToList();

break;

case 9:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.width.ToString().Contains(textbox.Text)).ToList();

break;

case 10:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.length.ToString().Contains(textbox.Text)).ToList();

break;

case 11:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.places.name.ToString().Contains(textbox.Text)).ToList();

break;

}

}

}

Программный код класса WarehouseWindow

public partial class WarehouseWindow : Window

{

public WarehouseWindow()

{

InitializeComponent();

Frame.Navigate(new Pages.WarehouseProducts());

}

private void ProductsBtn(object sender, RoutedEventArgs e)

{

Frame.Navigate(new Pages.WarehouseProducts());

}

private void ManufacturesBtn(object sender, RoutedEventArgs e)

{

Frame.Navigate(new Pages.WarehouseManufactures());

}

private void CountrysBtn(object sender, RoutedEventArgs e)

{

Frame.Navigate(new Pages.WarehouseCountrys());

}

private void ModelsBtn(object sender, RoutedEventArgs e)

{

Frame.Navigate(new Pages.WarehouseModels());

}

private void TypesBtn(object sender, RoutedEventArgs e)

{

Frame.Navigate(new Pages.WarehouseTypes());

}

private void UsersBtn(object sender, RoutedEventArgs e)

{

Frame.Navigate(new Pages.WarehouseUsers());

}

}

Программный код класса WarehouseCountrys

public partial class WarehouseCountrys : Page

{

private int DlgMode;

private Base.countrys SelectedCountrys;

private ObservableCollection<Base.countrys> Countrys;

public WarehouseCountrys()

{

InitializeComponent();

DataContext = this;

DlgLoad(false, "");

UpdateGrid(null);

}

private void LoadingCountrys(object sender, RoutedEventArgs e)

{

List<string> Columns = new List<string>();

for (int i = 0; i < 1; i++)

{

Columns.Add(PageGrid.Columns[i].Header.ToString());

}

FilterComboBox.ItemsSource = Columns;

FilterComboBox.SelectedIndex = 0;

foreach (DataGridColumn Column in PageGrid.Columns)

{

Column.CanUserSort = false;

}

}

private void UpdateGrid(Base.countrys country)

{

if ((country == null) && (PageGrid.ItemsSource != null))

{

country = (Base.countrys)PageGrid.SelectedItem;

}

Countrys = new ObservableCollection<Base.countrys>(SourceCore.db.countrys);

PageGrid.ItemsSource = Countrys;

PageGrid.ItemsSource = SourceCore.db.countrys.ToList();

PageGrid.SelectedItem = country;

}

private void DlgLoad(bool b, string DlgModeContent)

{

if (b == true)

{

ColumnChange.Width = new GridLength(300);

PageGrid.IsHitTestVisible = false;

RecordLabel.Content = DlgModeContent + " запись";

AddCommit.Content = DlgModeContent;

RecordAdd.IsEnabled = false;

RecordCopy.IsEnabled = false;

RecordEdit.IsEnabled = false;

RecordDellete.IsEnabled = false;

}

else

{

ColumnChange.Width = new GridLength(0);

PageGrid.IsHitTestVisible = true;

RecordAdd.IsEnabled = true;

RecordCopy.IsEnabled = true;

RecordEdit.IsEnabled = true;

RecordDellete.IsEnabled = true;

DlgMode = -1;

}

}

private void FillTextBox()

{

NameTextBox.Text = SelectedCountrys.name;

}

private void RecordAdd\_Click(object sender, RoutedEventArgs e)

{

DlgLoad(true, "Добавить");

DataContext = null;

DlgMode = 0;

}

private void RecordkCopy\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Копировать");

SelectedCountrys = (Base.countrys)PageGrid.SelectedItem;

FillTextBox();

DlgMode = 0;

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordEdit\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Изменить");

SelectedCountrys = (Base.countrys)PageGrid.SelectedItem;

FillTextBox();

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordDelete\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

try

{

// Ссылка на удаляемую книгу

Base.countrys DeletingAccessory = (Base.countrys)PageGrid.SelectedItem;

// Определение ссылки, на которую должен перейти указатель после удаления

if (PageGrid.SelectedIndex < PageGrid.Items.Count - 1)

{

PageGrid.SelectedIndex++;

}

else

{

if (PageGrid.SelectedIndex > 0)

{

PageGrid.SelectedIndex--;

}

}

Base.countrys SelectingAccessory = (Base.countrys)PageGrid.SelectedItem;

SourceCore.db.countrys.Remove(DeletingAccessory);

SourceCore.db.SaveChanges();

UpdateGrid(SelectingAccessory);

}

catch

{

MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);

}

}

}

private void FilterTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var textbox = sender as TextBox;

switch (FilterComboBox.SelectedIndex)

{

case 0:

PageGrid.ItemsSource = SourceCore.db.countrys.Where(q => q.name.ToString().Contains(textbox.Text)).ToList();

break;

}

}

private void AddCommit\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrEmpty(NameTextBox.Text)) errors.AppendLine("Укажите название");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (DlgMode == 0)

{

try

{

var NewBase = new Base.countrys();

NewBase.name = NameTextBox.Text.Trim();

SourceCore.db.countrys.Add(NewBase);

SelectedCountrys = NewBase;

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

else

{

try

{

var EditBase = new Base.countrys();

EditBase = SourceCore.db.countrys.First(p => p.id == SelectedCountrys.id);

EditBase.name = NameTextBox.Text.Trim();

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

try

{

SourceCore.db.SaveChanges();

UpdateGrid(SelectedCountrys);

DlgLoad(false, "");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

private void AddRollback\_Click(object sender, RoutedEventArgs e)

{

UpdateGrid(SelectedCountrys);

DlgLoad(false, "");

}

}

Программный код класса WarehouseManufactures

public partial class WarehouseManufactures : Page

{

private int DlgMode;

private Base.manufacturers SelectedManufactures;

private ObservableCollection<Base.manufacturers> Manufacturers;

public WarehouseManufactures()

{

InitializeComponent();

DataContext = this;

DlgLoad(false, "");

CountryComboBox.ItemsSource = SourceCore.db.countrys.ToList();

UpdateGrid(null);

}

private void LoadingManufactures(object sender, RoutedEventArgs e)

{

List<string> Columns = new List<string>();

for (int i = 0; i < 2; i++)

{

Columns.Add(PageGrid.Columns[i].Header.ToString());

}

FilterComboBox.ItemsSource = Columns;

FilterComboBox.SelectedIndex = 0;

foreach (DataGridColumn Column in PageGrid.Columns)

{

Column.CanUserSort = false;

}

}

private void UpdateGrid(Base.manufacturers manufacturer)

{

if ((manufacturer == null) && (PageGrid.ItemsSource != null))

{

manufacturer = (Base.manufacturers)PageGrid.SelectedItem;

}

Manufacturers = new ObservableCollection<Base.manufacturers>(SourceCore.db.manufacturers);

PageGrid.ItemsSource = Manufacturers;

PageGrid.ItemsSource = SourceCore.db.manufacturers.ToList();

CountryComboBox.ItemsSource = SourceCore.db.countrys.ToString();

PageGrid.SelectedItem = manufacturer;

}

private void DlgLoad(bool b, string DlgModeContent)

{

if (b == true)

{

ColumnChange.Width = new GridLength(300);

PageGrid.IsHitTestVisible = false;

RecordLabel.Content = DlgModeContent + " запись";

AddCommit.Content = DlgModeContent;

RecordAdd.IsEnabled = false;

RecordCopy.IsEnabled = false;

RecordEdit.IsEnabled = false;

RecordDellete.IsEnabled = false;

}

else

{

ColumnChange.Width = new GridLength(0);

PageGrid.IsHitTestVisible = true;

RecordAdd.IsEnabled = true;

RecordCopy.IsEnabled = true;

RecordEdit.IsEnabled = true;

RecordDellete.IsEnabled = true;

DlgMode = -1;

}

}

private void FillTextBox()

{

NameTextBox.Text = SelectedManufactures.name;

CountryComboBox.Text = SelectedManufactures.countrys.name.ToString();

}

private void RecordAdd\_Click(object sender, RoutedEventArgs e)

{

DlgLoad(true, "Добавить");

DataContext = null;

DlgMode = 0;

}

private void RecordkCopy\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Копировать");

SelectedManufactures = (Base.manufacturers)PageGrid.SelectedItem;

FillTextBox();

DlgMode = 0;

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordEdit\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Изменить");

SelectedManufactures = (Base.manufacturers)PageGrid.SelectedItem;

FillTextBox();

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordDelete\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

try

{

// Ссылка на удаляемую книгу

Base.manufacturers DeletingAccessory = (Base.manufacturers)PageGrid.SelectedItem;

// Определение ссылки, на которую должен перейти указатель после удаления

if (PageGrid.SelectedIndex < PageGrid.Items.Count - 1)

{

PageGrid.SelectedIndex++;

}

else

{

if (PageGrid.SelectedIndex > 0)

{

PageGrid.SelectedIndex--;

}

}

Base.manufacturers SelectingAccessory = (Base.manufacturers)PageGrid.SelectedItem;

SourceCore.db.manufacturers.Remove(DeletingAccessory);

SourceCore.db.SaveChanges();

UpdateGrid(SelectingAccessory);

}

catch

{

MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);

}

}

}

private void FilterTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var textbox = sender as TextBox;

switch (FilterComboBox.SelectedIndex)

{

case 0:

PageGrid.ItemsSource = SourceCore.db.manufacturers.Where(q => q.name.ToString().Contains(textbox.Text)).ToList();

break;

case 1:

PageGrid.ItemsSource = SourceCore.db.manufacturers.Where(q => q.countrys.name.ToString().Contains(textbox.Text)).ToList();

break;

}

}

private void AddCommit\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrEmpty(NameTextBox.Text)) errors.AppendLine("Укажите название");

if (((Base.countrys)CountryComboBox.SelectedItem == null) || (CountryComboBox.Text == " ...")) errors.AppendLine("Укажите страну");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (DlgMode == 0)

{

try

{

var NewBase = new Base.manufacturers();

NewBase.name = NameTextBox.Text.Trim();

NewBase.countrys = (Base.countrys)CountryComboBox.SelectedItem;

SourceCore.db.manufacturers.Add(NewBase);

SelectedManufactures = NewBase;

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

else

{

try

{

var EditBase = new Base.manufacturers();

EditBase = SourceCore.db.manufacturers.First(p => p.id == SelectedManufactures.id);

EditBase.name = NameTextBox.Text.Trim();

EditBase.countrys = (Base.countrys)CountryComboBox.SelectedItem;

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

try

{

SourceCore.db.SaveChanges();

UpdateGrid(SelectedManufactures);

DlgLoad(false, "");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

private void AddRollback\_Click(object sender, RoutedEventArgs e)

{

UpdateGrid(SelectedManufactures);

DlgLoad(false, "");

}

}

Программный код класса WarehouseModels

public partial class WarehouseModels : Page

{

private int DlgMode;

private Base.models SelectedModels;

private ObservableCollection<Base.models> Models;

public WarehouseModels()

{

InitializeComponent();

DataContext = this;

DlgLoad(false, "");

UpdateGrid(null);

}

private void LoadingModels(object sender, RoutedEventArgs e)

{

List<string> Columns = new List<string>();

for (int i = 0; i < 2; i++)

{

Columns.Add(PageGrid.Columns[i].Header.ToString());

}

FilterComboBox.ItemsSource = Columns;

FilterComboBox.SelectedIndex = 0;

foreach (DataGridColumn Column in PageGrid.Columns)

{

Column.CanUserSort = false;

}

}

private void UpdateGrid(Base.models model)

{

if ((model == null) && (PageGrid.ItemsSource != null))

{

model = (Base.models)PageGrid.SelectedItem;

}

Models = new ObservableCollection<Base.models>(SourceCore.db.models);

PageGrid.ItemsSource = Models;

PageGrid.ItemsSource = SourceCore.db.models.ToList();

PageGrid.SelectedItem = model;

}

private void DlgLoad(bool b, string DlgModeContent)

{

if (b == true)

{

ColumnChange.Width = new GridLength(300);

PageGrid.IsHitTestVisible = false;

RecordLabel.Content = DlgModeContent + " запись";

AddCommit.Content = DlgModeContent;

RecordAdd.IsEnabled = false;

RecordCopy.IsEnabled = false;

RecordEdit.IsEnabled = false;

RecordDellete.IsEnabled = false;

}

else

{

ColumnChange.Width = new GridLength(0);

PageGrid.IsHitTestVisible = true;

RecordAdd.IsEnabled = true;

RecordCopy.IsEnabled = true;

RecordEdit.IsEnabled = true;

RecordDellete.IsEnabled = true;

DlgMode = -1;

}

}

private void FillTextBox()

{

NameTextBox.Text = SelectedModels.name;

YearTextBox.Text = SelectedModels.year.ToString();

}

private void RecordAdd\_Click(object sender, RoutedEventArgs e)

{

DlgLoad(true, "Добавить");

DataContext = null;

DlgMode = 0;

}

private void RecordkCopy\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Копировать");

SelectedModels = (Base.models)PageGrid.SelectedItem;

FillTextBox();

DlgMode = 0;

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordEdit\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Изменить");

SelectedModels = (Base.models)PageGrid.SelectedItem;

FillTextBox();

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordDelete\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

try

{

// Ссылка на удаляемую книгу

Base.models DeletingAccessory = (Base.models)PageGrid.SelectedItem;

// Определение ссылки, на которую должен перейти указатель после удаления

if (PageGrid.SelectedIndex < PageGrid.Items.Count - 1)

{

PageGrid.SelectedIndex++;

}

else

{

if (PageGrid.SelectedIndex > 0)

{

PageGrid.SelectedIndex--;

}

}

Base.models SelectingAccessory = (Base.models)PageGrid.SelectedItem;

SourceCore.db.models.Remove(DeletingAccessory);

SourceCore.db.SaveChanges();

UpdateGrid(SelectingAccessory);

}

catch

{

MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);

}

}

}

private void FilterTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var textbox = sender as TextBox;

switch (FilterComboBox.SelectedIndex)

{

case 0:

PageGrid.ItemsSource = SourceCore.db.models.Where(q => q.name.ToString().Contains(textbox.Text)).ToList();

break;

case 1:

PageGrid.ItemsSource = SourceCore.db.models.Where(q => q.year.ToString().Contains(textbox.Text)).ToList();

break;

}

}

private void AddCommit\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrEmpty(NameTextBox.Text)) errors.AppendLine("Укажите название");

if (string.IsNullOrEmpty(YearTextBox.Text)) errors.AppendLine("Укажите год");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (DlgMode == 0)

{

try

{

var NewBase = new Base.models();

NewBase.name = NameTextBox.Text.Trim();

NewBase.year = int.Parse(YearTextBox.Text.Trim());

SourceCore.db.models.Add(NewBase);

SelectedModels = NewBase;

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

else

{

try

{

var EditBase = new Base.models();

EditBase = SourceCore.db.models.First(p => p.id == SelectedModels.id);

EditBase.name = NameTextBox.Text.Trim();

EditBase.year = int.Parse(YearTextBox.Text.Trim());

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

try

{

SourceCore.db.SaveChanges();

UpdateGrid(SelectedModels);

DlgLoad(false, "");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

private void AddRollback\_Click(object sender, RoutedEventArgs e)

{

UpdateGrid(SelectedModels);

DlgLoad(false, "");

}

}

Программный код класса WarehouseProducts

public partial class WarehouseProducts : Page

{

private int DlgMode;

private Base.products SelectedProduct;

private ObservableCollection<Base.products> Products;

public WarehouseProducts()

{

InitializeComponent();

DataContext = this;

DlgLoad(false, "");

TypeComboBox.ItemsSource = SourceCore.db.types.ToList();

ModelComboBox.ItemsSource = SourceCore.db.models.ToList();

ManufacturerComboBox.ItemsSource = SourceCore.db.manufacturers.ToList();

PlaceComboBox.ItemsSource = SourceCore.db.places.ToList();

UpdateGrid(null);

}

private void LoadingProd(object sender, RoutedEventArgs e)

{

List<string> Columns = new List<string>();

for (int i = 0; i < 12; i++)

{

Columns.Add(PageGrid.Columns[i].Header.ToString());

}

FilterComboBox.ItemsSource = Columns;

FilterComboBox.SelectedIndex = 0;

foreach (DataGridColumn Column in PageGrid.Columns)

{

Column.CanUserSort = false;

}

}

private void UpdateGrid(Base.products product)

{

if ((product == null) && (PageGrid.ItemsSource != null))

{

product = (Base.products)PageGrid.SelectedItem;

}

Products = new ObservableCollection<Base.products>(SourceCore.db.products);

PageGrid.ItemsSource = Products;

PageGrid.ItemsSource = SourceCore.db.products.ToList();

PageGrid.SelectedItem = product;

TypeComboBox.Text = SourceCore.db.types.ToString();

ModelComboBox.Text = SourceCore.db.models.ToString();

ManufacturerComboBox.Text = SourceCore.db.manufacturers.ToString();

PlaceComboBox.Text = SourceCore.db.places.ToString();

}

private void DlgLoad(bool b, string DlgModeContent)

{

if (b == true)

{

ColumnChange.Width = new GridLength(300);

PageGrid.IsHitTestVisible = false;

RecordLabel.Content = DlgModeContent + " запись";

AddCommit.Content = DlgModeContent;

RecordAdd.IsEnabled = false;

RecordCopy.IsEnabled = false;

RecordEdit.IsEnabled = false;

RecordDellete.IsEnabled = false;

}

else

{

ColumnChange.Width = new GridLength(0);

PageGrid.IsHitTestVisible = true;

RecordAdd.IsEnabled = true;

RecordCopy.IsEnabled = true;

RecordEdit.IsEnabled = true;

RecordDellete.IsEnabled = true;

DlgMode = -1;

}

}

private void FillTextBox()

{

TypeComboBox.Text = SelectedProduct.types.name.ToString();

ModelComboBox.Text = SelectedProduct.models.name.ToString();

NameTextBox.Text = SelectedProduct.name;

DescriptionTextBox.Text = SelectedProduct.description.ToString();

ManufacturerComboBox.Text = SelectedProduct.manufacturers.name.ToString();

CostTextBox.Text = SelectedProduct.cost.ToString();

CountTextBox.Text = SelectedProduct.count.ToString();

WeightTextBox.Text = SelectedProduct.weight.ToString();

HeightTextBox.Text = SelectedProduct.height.ToString();

WidthTextBox.Text = SelectedProduct.width.ToString();

LengthTextBox.Text = SelectedProduct.length.ToString();

PlaceComboBox.Text = SelectedProduct.places.name.ToString();

RecordTextImage.Text = SelectedProduct.image.ToString();

}

private void RecordAdd\_Click(object sender, RoutedEventArgs e)

{

DlgLoad(true, "Добавить");

DataContext = null;

DlgMode = 0;

}

private void RecordkCopy\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Копировать");

SelectedProduct = (Base.products)PageGrid.SelectedItem;

FillTextBox();

DlgMode = 0;

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordEdit\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Изменить");

SelectedProduct = (Base.products)PageGrid.SelectedItem;

FillTextBox();

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordDelete\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

try

{

// Ссылка на удаляемую книгу

Base.products DeletingAccessory = (Base.products)PageGrid.SelectedItem;

// Определение ссылки, на которую должен перейти указатель после удаления

if (PageGrid.SelectedIndex < PageGrid.Items.Count - 1)

{

PageGrid.SelectedIndex++;

}

else

{

if (PageGrid.SelectedIndex > 0)

{

PageGrid.SelectedIndex--;

}

}

Base.products SelectingAccessory = (Base.products)PageGrid.SelectedItem;

SourceCore.db.products.Remove(DeletingAccessory);

SourceCore.db.SaveChanges();

UpdateGrid(SelectingAccessory);

}

catch

{

MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);

}

}

}

private void FilterTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var textbox = sender as TextBox;

switch (FilterComboBox.SelectedIndex)

{

case 0:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.types.name.ToString().Contains(textbox.Text)).ToList();

break;

case 1:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.models.name.ToString().Contains(textbox.Text)).ToList();

break;

case 2:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.name.Contains(textbox.Text)).ToList();

break;

case 3:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.description.Contains(textbox.Text)).ToList();

break;

case 4:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.manufacturers.name.ToString().Contains(textbox.Text)).ToList();

break;

case 5:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.cost.ToString().Contains(textbox.Text)).ToList();

break;

case 6:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.count.ToString().Contains(textbox.Text)).ToList();

break;

case 7:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.weight.ToString().Contains(textbox.Text)).ToList();

break;

case 8:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.height.ToString().Contains(textbox.Text)).ToList();

break;

case 9:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.width.ToString().Contains(textbox.Text)).ToList();

break;

case 10:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.length.ToString().Contains(textbox.Text)).ToList();

break;

case 11:

PageGrid.ItemsSource = SourceCore.db.products.Where(q => q.places.name.ToString().Contains(textbox.Text)).ToList();

break;

}

}

private void AddCommit\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (((Base.types)TypeComboBox.SelectedItem == null) || (TypeComboBox.Text == " ...")) errors.AppendLine("Укажите тип");

if (((Base.models)ModelComboBox.SelectedItem == null) || (ModelComboBox.Text == " ...")) errors.AppendLine("Укажите модель");

if (string.IsNullOrEmpty(NameTextBox.Text)) errors.AppendLine("Укажите название");

if (string.IsNullOrEmpty(DescriptionTextBox.Text)) errors.AppendLine("Укажите описание");

if (((Base.manufacturers)ManufacturerComboBox.SelectedItem == null) || (ManufacturerComboBox.Text == " ...")) errors.AppendLine("Укажите производителя");

if (string.IsNullOrEmpty(CostTextBox.Text)) errors.AppendLine("Укажите цену");

if (string.IsNullOrEmpty(CountTextBox.Text)) errors.AppendLine("Укажите количество");

if (string.IsNullOrEmpty(WeightTextBox.Text)) errors.AppendLine("Укажите вес");

if (string.IsNullOrEmpty(HeightTextBox.Text)) errors.AppendLine("Укажите высоту");

if (string.IsNullOrEmpty(WidthTextBox.Text)) errors.AppendLine("Укажите ширину");

if (string.IsNullOrEmpty(LengthTextBox.Text)) errors.AppendLine("Укажите длину");

if (((Base.places)PlaceComboBox.SelectedItem == null) || (PlaceComboBox.Text == " ...")) errors.AppendLine("Укажите место");

string[] buf = RecordTextImage.Text.Split('.');

if (buf[buf.Length - 1] != "jpg") errors.AppendLine("Укажите название картинки");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (DlgMode == 0)

{

try

{

var NewBase = new Base.products();

NewBase.types = (Base.types)TypeComboBox.SelectedItem;

NewBase.models = (Base.models)ModelComboBox.SelectedItem;

NewBase.name = NameTextBox.Text.Trim();

NewBase.description = DescriptionTextBox.Text.Trim();

NewBase.manufacturers = (Base.manufacturers)ManufacturerComboBox.SelectedItem;

NewBase.cost = decimal.Parse(CostTextBox.Text.Trim());

NewBase.count = int.Parse(CountTextBox.Text.Trim());

NewBase.weight = float.Parse(WeightTextBox.Text.Trim());

NewBase.height = float.Parse(HeightTextBox.Text.Trim());

NewBase.width = float.Parse(WidthTextBox.Text.Trim());

NewBase.length = float.Parse(LengthTextBox.Text.Trim());

NewBase.places = (Base.places)PlaceComboBox.SelectedItem;

NewBase.image = ActionsWithPictures.ConvertImageToBinary(RecordTextImage.Text);

SourceCore.db.products.Add(NewBase);

SelectedProduct = NewBase;

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

else

{

try

{

var EditBase = new Base.products();

EditBase = SourceCore.db.products.First(p => p.id == SelectedProduct.id);

EditBase.types = (Base.types)TypeComboBox.SelectedItem;

EditBase.models = (Base.models)ModelComboBox.SelectedItem;

EditBase.name = NameTextBox.Text.Trim();

EditBase.description = DescriptionTextBox.Text.Trim();

EditBase.manufacturers = (Base.manufacturers)ManufacturerComboBox.SelectedItem;

EditBase.cost = decimal.Parse(CostTextBox.Text.Trim());

EditBase.count = int.Parse(CountTextBox.Text.Trim());

EditBase.weight = float.Parse(WeightTextBox.Text.Trim());

EditBase.height = float.Parse(HeightTextBox.Text.Trim());

EditBase.width = float.Parse(WidthTextBox.Text.Trim());

EditBase.length = float.Parse(LengthTextBox.Text.Trim());

EditBase.places = (Base.places)PlaceComboBox.SelectedItem;

EditBase.image = ActionsWithPictures.ConvertImageToBinary(RecordTextImage.Text);

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

try

{

SourceCore.db.SaveChanges();

UpdateGrid(SelectedProduct);

DlgLoad(false, "");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

private void AddRollback\_Click(object sender, RoutedEventArgs e)

{

UpdateGrid(SelectedProduct);

DlgLoad(false, "");

}

private void SelectFileButton\_Click(object sender, RoutedEventArgs e)

{

Microsoft.Win32.OpenFileDialog openFileDialog = new Microsoft.Win32.OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

RecordTextImage.Text = openFileDialog.FileName;

}

}

}

Программный код класса WarehouseTypes

public partial class WarehouseTypes : Page

{

private int DlgMode;

private Base.types SelectedTypes;

private ObservableCollection<Base.types> Types;

public WarehouseTypes()

{

InitializeComponent();

DataContext = this;

DlgLoad(false, "");

UpdateGrid(null);

}

private void LoadingTypes(object sender, RoutedEventArgs e)

{

List<string> Columns = new List<string>();

for (int i = 0; i < 1; i++)

{

Columns.Add(PageGrid.Columns[i].Header.ToString());

}

FilterComboBox.ItemsSource = Columns;

FilterComboBox.SelectedIndex = 0;

foreach (DataGridColumn Column in PageGrid.Columns)

{

Column.CanUserSort = false;

}

}

private void UpdateGrid(Base.types type)

{

if ((type == null) && (PageGrid.ItemsSource != null))

{

type = (Base.types)PageGrid.SelectedItem;

}

Types = new ObservableCollection<Base.types>(SourceCore.db.types);

PageGrid.ItemsSource = Types;

PageGrid.ItemsSource = SourceCore.db.types.ToList();

PageGrid.SelectedItem = type;

}

private void DlgLoad(bool b, string DlgModeContent)

{

if (b == true)

{

ColumnChange.Width = new GridLength(300);

PageGrid.IsHitTestVisible = false;

RecordLabel.Content = DlgModeContent + " запись";

AddCommit.Content = DlgModeContent;

RecordAdd.IsEnabled = false;

RecordCopy.IsEnabled = false;

RecordEdit.IsEnabled = false;

RecordDellete.IsEnabled = false;

}

else

{

ColumnChange.Width = new GridLength(0);

PageGrid.IsHitTestVisible = true;

RecordAdd.IsEnabled = true;

RecordCopy.IsEnabled = true;

RecordEdit.IsEnabled = true;

RecordDellete.IsEnabled = true;

DlgMode = -1;

}

}

private void FillTextBox()

{

NameTextBox.Text = SelectedTypes.name;

}

private void RecordAdd\_Click(object sender, RoutedEventArgs e)

{

DlgLoad(true, "Добавить");

DataContext = null;

DlgMode = 0;

}

private void RecordkCopy\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Копировать");

SelectedTypes = (Base.types)PageGrid.SelectedItem;

FillTextBox();

DlgMode = 0;

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordEdit\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Изменить");

SelectedTypes = (Base.types)PageGrid.SelectedItem;

FillTextBox();

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordDelete\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

try

{

// Ссылка на удаляемую книгу

Base.types DeletingAccessory = (Base.types)PageGrid.SelectedItem;

// Определение ссылки, на которую должен перейти указатель после удаления

if (PageGrid.SelectedIndex < PageGrid.Items.Count - 1)

{

PageGrid.SelectedIndex++;

}

else

{

if (PageGrid.SelectedIndex > 0)

{

PageGrid.SelectedIndex--;

}

}

Base.types SelectingAccessory = (Base.types)PageGrid.SelectedItem;

SourceCore.db.types.Remove(DeletingAccessory);

SourceCore.db.SaveChanges();

UpdateGrid(SelectingAccessory);

}

catch

{

MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);

}

}

}

private void FilterTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var textbox = sender as TextBox;

switch (FilterComboBox.SelectedIndex)

{

case 0:

PageGrid.ItemsSource = SourceCore.db.types.Where(q => q.name.ToString().Contains(textbox.Text)).ToList();

break;

}

}

private void AddCommit\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrEmpty(NameTextBox.Text)) errors.AppendLine("Укажите название");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (DlgMode == 0)

{

try

{

var NewBase = new Base.types();

NewBase.name = NameTextBox.Text.Trim();

SourceCore.db.types.Add(NewBase);

SelectedTypes = NewBase;

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

else

{

try

{

var EditBase = new Base.types();

EditBase = SourceCore.db.types.First(p => p.id == SelectedTypes.id);

EditBase.name = NameTextBox.Text.Trim();

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

try

{

SourceCore.db.SaveChanges();

UpdateGrid(SelectedTypes);

DlgLoad(false, "");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

private void AddRollback\_Click(object sender, RoutedEventArgs e)

{

UpdateGrid(SelectedTypes);

DlgLoad(false, "");

}

}

Программный код класса WarehouseUsers

public partial class WarehouseUsers : Page

{

private int DlgMode;

private Base.users SelectedUsers;

private ObservableCollection<Base.users> Users;

public WarehouseUsers()

{

InitializeComponent();

DataContext = this;

DlgLoad(false, "");

UpdateGrid(null);

}

private void LoadingUsers(object sender, RoutedEventArgs e)

{

List<string> Columns = new List<string>();

for (int i = 0; i < 2; i++)

{

Columns.Add(PageGrid.Columns[i].Header.ToString());

}

FilterComboBox.ItemsSource = Columns;

FilterComboBox.SelectedIndex = 0;

foreach (DataGridColumn Column in PageGrid.Columns)

{

Column.CanUserSort = false;

}

}

private void UpdateGrid(Base.users user)

{

if ((user == null) && (PageGrid.ItemsSource != null))

{

user = (Base.users)PageGrid.SelectedItem;

}

Users = new ObservableCollection<Base.users>(SourceCore.db.users);

PageGrid.ItemsSource = Users;

PageGrid.ItemsSource = SourceCore.db.users.ToList();

PageGrid.SelectedItem = user;

}

private void DlgLoad(bool b, string DlgModeContent)

{

if (b == true)

{

ColumnChange.Width = new GridLength(300);

PageGrid.IsHitTestVisible = false;

RecordLabel.Content = DlgModeContent + " запись";

AddCommit.Content = DlgModeContent;

RecordAdd.IsEnabled = false;

RecordCopy.IsEnabled = false;

RecordEdit.IsEnabled = false;

RecordDellete.IsEnabled = false;

}

else

{

ColumnChange.Width = new GridLength(0);

PageGrid.IsHitTestVisible = true;

RecordAdd.IsEnabled = true;

RecordCopy.IsEnabled = true;

RecordEdit.IsEnabled = true;

RecordDellete.IsEnabled = true;

DlgMode = -1;

}

}

private void FillTextBox()

{

LoginTextBox.Text = SelectedUsers.login;

PasswordTextBox.Text = SelectedUsers.password;

}

private void RecordAdd\_Click(object sender, RoutedEventArgs e)

{

DlgLoad(true, "Добавить");

DataContext = null;

DlgMode = 0;

}

private void RecordkCopy\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Копировать");

SelectedUsers = (Base.users)PageGrid.SelectedItem;

FillTextBox();

DlgMode = 0;

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordEdit\_Click(object sender, RoutedEventArgs e)

{

if (PageGrid.SelectedItem != null)

{

DlgLoad(true, "Изменить");

SelectedUsers = (Base.users)PageGrid.SelectedItem;

FillTextBox();

}

else

{

MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);

}

}

private void RecordDelete\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

try

{

// Ссылка на удаляемую книгу

Base.users DeletingAccessory = (Base.users)PageGrid.SelectedItem;

// Определение ссылки, на которую должен перейти указатель после удаления

if (PageGrid.SelectedIndex < PageGrid.Items.Count - 1)

{

PageGrid.SelectedIndex++;

}

else

{

if (PageGrid.SelectedIndex > 0)

{

PageGrid.SelectedIndex--;

}

}

Base.users SelectingAccessory = (Base.users)PageGrid.SelectedItem;

SourceCore.db.users.Remove(DeletingAccessory);

SourceCore.db.SaveChanges();

UpdateGrid(SelectingAccessory);

}

catch

{

MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);

}

}

}

private void FilterTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var textbox = sender as TextBox;

switch (FilterComboBox.SelectedIndex)

{

case 0:

PageGrid.ItemsSource = SourceCore.db.users.Where(q => q.login.ToString().Contains(textbox.Text)).ToList();

break;

case 1:

PageGrid.ItemsSource = SourceCore.db.users.Where(q => q.password.ToString().Contains(textbox.Text)).ToList();

break;

}

}

private void AddCommit\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrEmpty(LoginTextBox.Text)) errors.AppendLine("Укажите логин");

if (string.IsNullOrEmpty(PasswordTextBox.Text)) errors.AppendLine("Укажите пароль");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (DlgMode == 0)

{

try

{

var NewBase = new Base.users();

NewBase.login = LoginTextBox.Text.Trim();

NewBase.password = PasswordTextBox.Text.Trim();

SourceCore.db.users.Add(NewBase);

SelectedUsers = NewBase;

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

else

{

try

{

var EditBase = new Base.users();

EditBase = SourceCore.db.users.First(p => p.id == SelectedUsers.id);

EditBase.login = LoginTextBox.Text.Trim();

EditBase.password = PasswordTextBox.Text.Trim();

}

catch (Exception)

{

MessageBox.Show("Введены некоректные данные");

}

}

try

{

SourceCore.db.SaveChanges();

UpdateGrid(SelectedUsers);

DlgLoad(false, "");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

private void AddRollback\_Click(object sender, RoutedEventArgs e)

{

UpdateGrid(SelectedUsers);

DlgLoad(false, "");

}

}