(Перед запуском приложения почитать файл «Запуск БД и сайта»)

Данный проект представляет интернет-магазин, имеющий следующие особенности :

**Каждый товар в магазине содержит:**

· Название

· Описание

· Цена

· Дата создания

· Дата последнего изменения

· Категория

- Название категории

· Контакты автора

- Имя

- Номер телефона

- Email

· Местонахождение товара

· Состояние

**Система предоставляет возможность изменять/добавлять/удалять любую информацию о товарах.**

**Система предоставляет навигацию:**

· По категориям

· По полному или частичному имени товара

**Система обладает аутентификацией и авторизацией. Система может работать в 4-х режимах:**

· Режим гостя. Данный режим разрешает:

- Пользоваться навигацией

- Пользоваться поиском

- Просматривать товары

**· Режим пользователя. Данный режим разрешает все, что может гость, а также:**

- Просмотр списка своих товаров

- Добавлять/изменять/удалять свои товары

- Редактировать свой профиль

- Покупать товары других пользователей

**· Режим редактора. Данный режим разрешает все, что может пользователь, а также:**

- Создавать/удалять/изменять любые товары

**· Режим администратора. Данный режим разрешает все, что может редактор, а также:**

- Создавать/изменять/удалять пользователей

- Добавлять новые категории

Пароли и логины:

**Вот перечень пользователей(при желании новых можно зарегистрировать, и выдать любую роль с помощью аккаунта администратора)**

|  |  |  |
| --- | --- | --- |
| **Login** | **Password** | **Role** |
| **VlastelinValentin** | **павук** | **Administrator** |
| **Anatolya** | **123** | **Moderator** |
| **Vadimka12** | **343221** | **User** |
| **Valentinka48** | **admin** | **Moderator** |
| **Genka15** | **1234** | **User** |
| **Padinhton** | **padinhton** | **User** |

**Части кода программы**

**Класс ProductContext**

Работает с базой данных через технологию ADO.NET. Таких классов несколько, почти на каждый Entity из пространства имен Shop.Shared.Entities (User, Product, Category, Role, Location, Purchase, State)

using Shop.Data.DataContext.Interfaces;

using Shop.Shared.Entities;

using Shop.Shared.Helpers;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using SqlConst = Shop.Data.Constants.SqlQueryConstants;

using Typography = Shop.Shared.Constants.TypographyConstants;

namespace Shop.Data.DataContext.Realization.MsSql

{

public class ProductContext : IProductContext

{

UserContext \_userContext = new UserContext();

CategoryContext \_categoryContext = new CategoryContext();

StateContext \_stateContext = new StateContext();

LocationContext \_locationContext = new LocationContext();

public IReadOnlyCollection<Product> GetAllByName(string searchParameter, string searchQuery)

{

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

List<Product> products = new List<Product>();

var command = new SqlCommand(SqlConst.SelectAllProductInDbString, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

if (reader[searchParameter].ToString().Contains(searchQuery))

{

products.Add(GetProduct(reader));

}

}

return products;

}

}

public void Edit(Product editedProduct)

{

using (SqlConnection connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

var command = new SqlCommand($"UPDATE [Product]{Typography.NewLine}SET [Name] = @name, [Description] = @description, [CategoryId] = @categoryId, [LastModifiedDate] = @lastModifiedDate, [LocationId] = @locationId, [Price] = @price, [StateId] = @stateId, [UserId] = @userId {Typography.NewLine}WHERE Id = {editedProduct.Id}", connection);

command.Parameters.AddWithValue("@name", editedProduct.Name);

command.Parameters.AddWithValue("@description", editedProduct.Description);

command.Parameters.AddWithValue("@categoryId", editedProduct.Category.Id);

command.Parameters.AddWithValue("@lastModifiedDate", editedProduct.LastModifiedDate);

command.Parameters.AddWithValue("@locationId", \_locationContext.GetIdByName(editedProduct.LocationOfProduct));

command.Parameters.AddWithValue("@price", editedProduct.Price);

command.Parameters.AddWithValue("@stateId", editedProduct.State.Id);

command.Parameters.AddWithValue("@userId", editedProduct.Author.Id);

command.ExecuteNonQuery();

}

}

public IReadOnlyCollection<Product> GetByCategoryId(int categoryId)

{

List<Product> products = new List<Product>();

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

using (var command = new SqlCommand(SqlConst.SelectAllProductInDbString, connection))

{

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

if ((int)reader["CategoryId"] == categoryId)

{

products.Add(GetProduct(reader));

}

}

}

}

return products;

}

}

public IReadOnlyCollection<Product> GetByUserId(int userId)

{

List<Product> products = new List<Product>();

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

using (var command = new SqlCommand(SqlConst.SelectAllProductInDbString, connection))

{

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

if ((int)reader["UserId"] == userId)

{

products.Add(GetProduct(reader));

}

}

}

}

return products;

}

}

public Product GetProduct(SqlDataReader reader)

{

return new Product

{

Id = (int)reader["Id"],

Name = (string)reader["Name"],

Description = (string)reader["Description"],

Price = (decimal)reader["Price"],

CreationDate = (DateTime)reader["CreationDate"],

LastModifiedDate = (DateTime)reader["LastModifiedDate"],

Category = new Category

{

Id = (int)reader["CategoryId"],

Name = (string)reader["Category"]

},

Author = new User

{

Id = (int)reader["UserId"],

Login = (string)reader["Login"],

Email = (string)reader["Email"],

PhoneNumber = (string)reader["PhoneNumber"],

Role = RoleHelper.ConvertToRoleType((int)reader["RoleId"])

},

LocationOfProduct = (string)reader["Location"],

State = new State

{

Id = (int)reader["StateId"],

Name = (string)reader["State"]

}

};

}

public Product GetById(int id)

{

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

List<Product> products = new List<Product>();

string query = SqlConst.SelectAllProductInDbString + Typography.NewLine + $"WHERE [Product].[Id] = @id";

var command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@id", id);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

return GetProduct(reader);

}

}

public IReadOnlyCollection<Product> GetAll()

{

List<Product> returnProducts = new List<Product>();

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

List<Product> products = new List<Product>();

var command = new SqlCommand(SqlConst.SelectAllProductInDbString, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

returnProducts.Add(GetProduct(reader));

}

return returnProducts;

}

}

public void DeleteById(int id)

{

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

var command = new SqlCommand($"DELETE [Product] WHERE [Id] = {id}", connection);

command.ExecuteNonQuery();

}

}

public void Save(Product product)

{

using (var connection = new SqlConnection(SqlConst.ConnectionToConsoleShopString))

{

connection.Open();

var command = new SqlCommand($"INSERT INTO [dbo].[Product]([CategoryId],[LocationId],[StateId],[UserId],[Name],[Description],[Price],[CreationDate],[LastModifiedDate]) VALUES(@categoryId, @locationId, @stateId, @authorId, @productName, @description, @price, @creationDate, @lastModifiedDate)", connection);

command.Parameters.AddWithValue("@categoryId", product.Category.Id);

command.Parameters.AddWithValue("@locationId", \_locationContext.GetIdByName(product.LocationOfProduct)); //изменить после того, как сделаю наконец эту систему сложной локации с составными ключами

command.Parameters.AddWithValue("@stateId", product.State.Id);

command.Parameters.AddWithValue("@authorId", product.Author.Id);

command.Parameters.AddWithValue("@productName", product.Name);

command.Parameters.AddWithValue("@description", product.Description);

command.Parameters.AddWithValue("@price", product.Price);

command.Parameters.AddWithValue("@creationDate", product.CreationDate);

command.Parameters.AddWithValue("@lastModifiedDate", product.LastModifiedDate);

command.ExecuteNonQuery();

}

}

}

}