**ПРИЛОЖЕНИЕ M**

**Листинг кода**

using Microsoft.EntityFrameworkCore;

namespace DatabaseManager

{

public class DatabaseContext : DbContext

{

public DbSet<Unit> Units { get; set; } = null!;

public DbSet<Post> Posts { get; set; } = null!;

public DbSet<Sale> Sales { get; set; } = null!;

public DbSet<Product> Products { get; set; } = null!;

public DbSet<Receipt> Receipts { get; set; } = null!;

public DbSet<Contract> Contracts { get; set; } = null!;

public DbSet<Division> Divisions { get; set; } = null!;

public DbSet<Employee> Employees { get; set; } = null!;

public DbSet<Location> Locations { get; set; } = null!;

public DbSet<Material> Materials { get; set; } = null!;

public DbSet<Counterpartie> Counterparties { get; set; } = null!;

public DbSet<Product\_Material> Product\_Materials { get; set; } = null!;

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlite("Data Source=VacDB.db");

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

// Настройка связи Employee - Division (многие к одному)

modelBuilder.Entity<Employee>()

.HasOne(e => e.Division)

.WithMany(d => d.Employees)

.HasForeignKey(e => e.DivisionId)

.OnDelete(DeleteBehavior.NoAction); // Устанавливаем "ON DELETE NO ACTION"

// Настройка связи Employee - Post (многие к одному)

modelBuilder.Entity<Employee>()

.HasOne(e => e.Post)

.WithMany(p => p.Employees)

.HasForeignKey(e => e.PostId)

.OnDelete(DeleteBehavior.NoAction); // Устанавливаем "ON DELETE NO ACTION"

modelBuilder.Entity<Counterpartie>(entity =>

{

entity.Property(c => c.Type)

.HasConversion<string>() // Хранение значения перечисления как текста

.HasDefaultValue(CounterpartieType.Fiz) // Значение по умолчанию

.HasColumnType("TEXT");

// Используем ToTable для добавления CHECK-ограничения

entity.ToTable(t => t.HasCheckConstraint("CK\_Customer\_Type", "Type IN ('Fiz', 'Ur')"));

});

// Employee - Location (один ко многим)

modelBuilder.Entity<Location>()

.HasOne(l => l.Employee)

.WithMany()

.HasForeignKey(l => l.EmployeeId)

.OnDelete(DeleteBehavior.NoAction);

// Unit - Product (один ко многим)

modelBuilder.Entity<Product>()

.HasOne(p => p.Unit)

.WithMany(u => u.Products)

.HasForeignKey(p => p.UnitId)

.OnDelete(DeleteBehavior.NoAction);

// Unit - Material (один ко многим)

modelBuilder.Entity<Material>()

.HasOne(m => m.Unit)

.WithMany(u => u.Materials)

.HasForeignKey(m => m.UnitId)

.OnDelete(DeleteBehavior.NoAction);

// Location - Product (один ко многим)

modelBuilder.Entity<Product>()

.HasOne(p => p.Location)

.WithMany(l => l.Products)

.HasForeignKey(p => p.LocationId)

.OnDelete(DeleteBehavior.NoAction);

// Location - Material (один ко многим)

modelBuilder.Entity<Material>()

.HasOne(m => m.Location)

.WithMany(l => l.Materials)

.HasForeignKey(m => m.LocationId)

.OnDelete(DeleteBehavior.NoAction);

// Product - Material (многие ко многим)

modelBuilder.Entity<Product\_Material>()

.HasKey(pm => new { pm.Id });

modelBuilder.Entity<Product\_Material>()

.HasOne(pm => pm.Product)

.WithMany(p => p.ProductMaterials)

.HasForeignKey(pm => pm.ProductId);

modelBuilder.Entity<Product\_Material>()

.HasOne(pm => pm.Material)

.WithMany(m => m.ProductMaterials)

.HasForeignKey(pm => pm.MaterialId);

// Counterparty - Contract (один ко многим)

modelBuilder.Entity<Contract>()

.HasOne(c => c.Counterpartie)

.WithMany(cp => cp.Contracts)

.HasForeignKey(c => c.CounterpartyId)

.OnDelete(DeleteBehavior.NoAction);

// Counterparty - Receipt (один ко многим)

modelBuilder.Entity<Receipt>()

.HasOne(r => r.Counterpartie)

.WithMany(cp => cp.Receipts)

.HasForeignKey(r => r.CounterpartyId)

.OnDelete(DeleteBehavior.NoAction);

// Product - Contract (один ко многим)

modelBuilder.Entity<Contract>()

.HasOne(c => c.Product)

.WithMany(p => p.Contracts)

.HasForeignKey(c => c.ProductId)

.OnDelete(DeleteBehavior.NoAction);

// Contract - Sale (один ко многим)

modelBuilder.Entity<Sale>()

.HasOne(s => s.Contract)

.WithMany(c => c.Sales)

.HasForeignKey(s => s.ContractId)

.OnDelete(DeleteBehavior.NoAction);

}

}

}

using System.ComponentModel;

using System.ComponentModel.DataAnnotations.Schema;

using System.Runtime.CompilerServices;

namespace DatabaseManager

{

public class BaseModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler? PropertyChanged;

protected void OnPropertyChanged([CallerMemberName] string? name = null)

=> PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));

protected virtual bool SetProperty<T>(ref T member, T value, [CallerMemberName] string? propertyName = null)

{

if (EqualityComparer<T>.Default.Equals(member, value))

{

return false;

}

member = value;

OnPropertyChanged(propertyName);

return true;

}

[NotMapped]

private bool \_isSelected;

[NotMapped]

public bool IsSelected

{

get => \_isSelected;

set => SetProperty(ref \_isSelected, value);

}

}

}

namespace DatabaseManager

{

public class Contract : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private string? \_name;

public required string Name

{

get => \_name ?? throw new InvalidOperationException("Name must be initialized.");

set

{

\_name = value;

OnPropertyChanged(nameof(Name));

}

}

private int \_counterpartyId;

public int CounterpartyId

{

get => \_counterpartyId;

set

{

\_counterpartyId = value;

OnPropertyChanged(nameof(CounterpartyId));

}

}

private Counterpartie? \_counterpartie;

public Counterpartie? Counterpartie

{

get => \_counterpartie;

set

{

\_counterpartie = value;

OnPropertyChanged(nameof(Counterpartie));

}

}

private DateTime \_date;

public DateTime Date

{

get => \_date;

set

{

\_date = value;

OnPropertyChanged(nameof(Date));

}

}

private double \_summ;

public double Summ

{

get => \_summ;

set

{

\_summ = value;

OnPropertyChanged(nameof(Summ));

}

}

private int \_productId;

public int ProductId

{

get => \_productId;

set

{

\_productId = value;

OnPropertyChanged(nameof(ProductId));

}

}

private Product? \_product;

public Product? Product

{

get => \_product;

set

{

\_product = value;

OnPropertyChanged(nameof(Product));

}

}

private int \_count;

public int Count

{

get => \_count;

set

{

\_count = value > 0 ? value : 0;

OnPropertyChanged(nameof(Count));

}

}

private List<Sale> \_sales = [];

public List<Sale> Sales

{

get => \_sales;

set

{

\_sales = value;

OnPropertyChanged(nameof(Sales));

}

}

}

}

namespace DatabaseManager

{

public enum CounterpartieType { Fiz, Ur }

public class Counterpartie : BaseModel

{

private int \_Id;

public int Id

{

get => \_Id;

set => SetProperty(ref \_Id, value);

}

private string \_Name;

public required string Name

{

get => \_Name;

set => SetProperty(ref \_Name, value);

}

private string \_LegalAddress;

public required string LegalAddress

{

get => \_LegalAddress;

set => SetProperty(ref \_LegalAddress, value);

}

private string \_PhoneNomber;

public required string PhoneNomber

{

get => \_PhoneNomber;

set => SetProperty(ref \_PhoneNomber, value);

}

private string \_PostalAddress;

public required string PostalAddress

{

get => \_PostalAddress;

set => SetProperty(ref \_PostalAddress, value);

}

private string \_Unp;

public required string Unp

{

get => \_Unp;

set => SetProperty(ref \_Unp, value);

}

private string \_bankAccount;

public required string BankAccount

{

get => \_bankAccount;

set => SetProperty(ref \_bankAccount, value);

}

private CounterpartieType \_Type = CounterpartieType.Fiz;

public CounterpartieType Type

{

get => \_Type;

set => SetProperty(ref \_Type, value);

}

private string? \_Okulp;

public string? Okulp

{

get => \_Okulp;

set => SetProperty(ref \_Okulp, value);

}

private string? \_Okpo;

public string? Okpo

{

get => \_Okpo;

set => SetProperty(ref \_Okpo, value);

}

private string? \_Oked;

public string? Oked

{

get => \_Oked;

set => SetProperty(ref \_Oked, value);

}

private List<Contract> \_Contracts = [];

public List<Contract> Contracts

{

get => \_Contracts;

set => SetProperty(ref \_Contracts, value);

}

private List<Receipt> \_Receipts = [];

public List<Receipt> Receipts

{

get => \_Receipts;

set => SetProperty(ref \_Receipts, value);

}

}

}

namespace DatabaseManager

{

public class Division : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private string? \_name;

public required string Name

{

get => \_name ?? throw new InvalidOperationException("Name must be initialized.");

set

{

\_name = value;

OnPropertyChanged(nameof(Name));

}

}

private List<Employee> \_employees = [];

public List<Employee> Employees

{

get => \_employees;

set

{

\_employees = value;

OnPropertyChanged(nameof(Employees));

}

}

}

}

namespace DatabaseManager

{

public class Employee : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private string? \_fio;

public required string Fio

{

get => \_fio ?? throw new InvalidOperationException("Fio must be initialized.");

set

{

\_fio = value;

OnPropertyChanged(nameof(Fio));

}

}

private int \_divisionId;

public int DivisionId

{

get => \_divisionId;

set

{

\_divisionId = value;

OnPropertyChanged(nameof(DivisionId));

}

}

private Division? \_division;

public Division? Division

{

get => \_division;

set

{

\_division = value;

OnPropertyChanged(nameof(Division));

}

}

private int \_postId;

public int PostId

{

get => \_postId;

set

{

\_postId = value;

OnPropertyChanged(nameof(PostId));

}

}

private Post? \_post;

public Post? Post

{

get => \_post;

set

{

\_post = value;

OnPropertyChanged(nameof(Post));

}

}

private DateTime \_dateHire;

public DateTime DateHire

{

get => \_dateHire;

set

{

\_dateHire = value;

OnPropertyChanged(nameof(DateHire));

}

}

private DateTime? \_dateDismissal;

public DateTime? DateDismissal

{

get => \_dateDismissal;

set

{

\_dateDismissal = value;

OnPropertyChanged(nameof(DateDismissal));

}

}

}

}

namespace DatabaseManager

{

public class Location : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private string? \_name;

public required string Name

{

get => \_name ?? throw new InvalidOperationException("Name must be initialized.");

set

{

\_name = value;

OnPropertyChanged(nameof(Name));

}

}

private int \_employeeId;

public int EmployeeId

{

get => \_employeeId;

set

{

\_employeeId = value;

OnPropertyChanged(nameof(EmployeeId));

}

}

private Employee? \_employee;

public Employee? Employee

{

get => \_employee;

set

{

\_employee = value;

OnPropertyChanged(nameof(Employee));

}

}

private List<Product> \_products = new List<Product>();

public List<Product> Products

{

get => \_products;

set

{

\_products = value;

OnPropertyChanged(nameof(Products));

}

}

private List<Material> \_materials = new List<Material>();

public List<Material> Materials

{

get => \_materials;

set

{

\_materials = value;

OnPropertyChanged(nameof(Materials));

}

}

}

}

namespace DatabaseManager

{

public class Material : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private string? \_name;

public required string Name

{

get => \_name ?? throw new InvalidOperationException("Name must be initialized.");

set

{

\_name = value;

OnPropertyChanged(nameof(Name));

}

}

private int \_unitId;

public int UnitId

{

get => \_unitId;

set

{

\_unitId = value;

OnPropertyChanged(nameof(UnitId));

}

}

private Unit? \_unit;

public Unit? Unit

{

get => \_unit;

set

{

\_unit = value;

OnPropertyChanged(nameof(Unit));

}

}

private int \_locationId;

public int LocationId

{

get => \_locationId;

set

{

\_locationId = value;

OnPropertyChanged(nameof(LocationId));

}

}

private Location? \_location;

public Location? Location

{

get => \_location;

set

{

\_location = value;

OnPropertyChanged(nameof(Location));

}

}

private double \_count;

public double Count

{

get => \_count;

set

{

\_count = value > 0 ? value : 0;

OnPropertyChanged(nameof(Count));

OnPropertyChanged(nameof(GetSum));

}

}

private double \_price;

public double Price

{

get => \_price;

set

{

\_price = value > 0 ? value : 0;

OnPropertyChanged(nameof(Price));

OnPropertyChanged(nameof(GetSum));

}

}

private List<Product\_Material> \_productMaterials = [];

public List<Product\_Material> ProductMaterials

{

get => \_productMaterials;

set

{

\_productMaterials = value;

OnPropertyChanged(nameof(ProductMaterials));

}

}

private List<Receipt> \_receipts = [];

public List<Receipt> Receipts

{

get => \_receipts;

set

{

\_receipts = value;

OnPropertyChanged(nameof(Receipts));

}

}

[System.ComponentModel.DataAnnotations.Schema.NotMapped]

public double GetSum { get => Math.Round(Count \* Price, 3); }

}

}

namespace DatabaseManager

{

public class Post : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private string? \_name;

public required string Name

{

get => \_name ?? throw new InvalidOperationException("Name must be initialized.");

set

{

\_name = value;

OnPropertyChanged(nameof(Name));

}

}

private List<Employee> \_employees = new List<Employee>();

public List<Employee> Employees

{

get => \_employees;

set

{

\_employees = value;

OnPropertyChanged(nameof(Employees));

}

}

}

}

namespace DatabaseManager

{

public class Product : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private string? \_name;

public required string Name

{

get => \_name ?? throw new InvalidOperationException("Name must be initialized.");

set

{

\_name = value;

OnPropertyChanged(nameof(Name));

}

}

private string? \_serialNo;

public required string SerialNo

{

get => \_serialNo ?? throw new InvalidOperationException("SerialNo must be initialized.");

set

{

\_serialNo = value;

OnPropertyChanged(nameof(SerialNo));

}

}

private int \_unitId;

public int UnitId

{

get => \_unitId;

set

{

\_unitId = value;

OnPropertyChanged(nameof(UnitId));

}

}

private Unit? \_unit;

public Unit? Unit

{

get => \_unit;

set

{

\_unit = value;

OnPropertyChanged(nameof(Unit));

}

}

private int \_locationId;

public int LocationId

{

get => \_locationId;

set

{

\_locationId = value;

OnPropertyChanged(nameof(LocationId));

}

}

private Location? \_location;

public Location? Location

{

get => \_location;

set

{

\_location = value;

OnPropertyChanged(nameof(Location));

}

}

private List<Product\_Material> \_productMaterials = [];

public List<Product\_Material> ProductMaterials

{

get => \_productMaterials;

set

{

\_productMaterials = value;

OnPropertyChanged(nameof(ProductMaterials));

}

}

private List<Contract> \_contracts = [];

public List<Contract> Contracts

{

get => \_contracts;

set

{

\_contracts = value;

OnPropertyChanged(nameof(Contracts));

}

}

private int \_count;

public int Count

{

get => \_count;

set

{

\_count = value > 0 ? value : 0;

OnPropertyChanged(nameof(Count));

}

}

}

}

using System.Diagnostics;

namespace DatabaseManager

{

public class Product\_Material : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private int \_productId;

public int ProductId

{

get => \_productId;

set

{

\_productId = value;

OnPropertyChanged(nameof(ProductId));

}

}

private Product? \_product;

public Product? Product

{

get => \_product;

set

{

\_product = value;

OnPropertyChanged(nameof(Product));

}

}

private int \_materialId;

public int MaterialId

{

get => \_materialId;

set

{

\_materialId = value;

OnPropertyChanged(nameof(MaterialId));

}

}

private Material? \_material;

public Material? Material

{

get => \_material;

set

{

\_material = value;

OnPropertyChanged(nameof(Material));

}

}

private int \_quantity;

public int Quantity

{

get => \_quantity;

set

{

\_quantity = value > 0 ? value : 0;

OnPropertyChanged(nameof(Quantity));

}

}

[System.ComponentModel.DataAnnotations.Schema.NotMapped]

public double GetSum { get => Material == null ? 0 : Quantity \* Material.Price; }

}

}

namespace DatabaseManager

{

public class Receipt : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private int \_materialId;

public int MaterialId

{

get => \_materialId;

set

{

\_materialId = value;

OnPropertyChanged(nameof(MaterialId));

}

}

private Material? \_material;

public Material? Material

{

get => \_material;

set

{

\_material = value;

OnPropertyChanged(nameof(Material));

Summ = Count \* (Material != null ? Material.Price : 0);

}

}

private double \_summ;

public double Summ

{

get => \_summ;

set

{

\_summ = value;

OnPropertyChanged(nameof(Summ));

}

}

private DateTime \_date;

public DateTime Date

{

get => \_date;

set

{

\_date = value;

OnPropertyChanged(nameof(Date));

}

}

private int \_count;

public int Count

{

get => \_count;

set

{

\_count = value;

OnPropertyChanged(nameof(Count));

Summ = Count \* (Material != null ? Material.Price : 0);

}

}

private int \_counterpartyId;

public int CounterpartyId

{

get => \_counterpartyId;

set

{

\_counterpartyId = value;

OnPropertyChanged(nameof(CounterpartyId));

}

}

private Counterpartie? \_counterpartie;

public Counterpartie? Counterpartie

{

get => \_counterpartie;

set

{

\_counterpartie = value;

OnPropertyChanged(nameof(Counterpartie));

}

}

}

}

namespace DatabaseManager

{

public class Sale : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private int \_contractId;

public int ContractId

{

get => \_contractId;

set

{

\_contractId = value;

OnPropertyChanged(nameof(ContractId));

}

}

private Contract? \_contract;

public Contract? Contract

{

get => \_contract;

set

{

\_contract = value;

OnPropertyChanged(nameof(Contract));

}

}

private double \_summ;

public double Summ

{

get => \_summ;

set

{

\_summ = value;

OnPropertyChanged(nameof(Summ));

}

}

private DateTime \_date;

public DateTime Date

{

get => \_date;

set

{

\_date = value;

OnPropertyChanged(nameof(Date));

}

}

private int \_count;

public int Count

{

get => \_count;

set

{

\_count = value > 0 ? value : 0;

OnPropertyChanged(nameof(Count));

}

}

}

}

namespace DatabaseManager

{

public class Unit : BaseModel

{

private int \_id;

public int Id

{

get => \_id;

set

{

\_id = value;

OnPropertyChanged(nameof(Id));

}

}

private string? \_name;

public required string Name

{

get => \_name ?? throw new InvalidOperationException("Name must be initialized.");

set

{

\_name = value;

OnPropertyChanged(nameof(Name));

}

}

private List<Product> \_products = [];

public List<Product> Products

{

get => \_products;

set

{

\_products = value;

OnPropertyChanged(nameof(Products));

}

}

private List<Material> \_materials = [];

public List<Material> Materials

{

get => \_materials;

set

{

\_materials = value;

OnPropertyChanged(nameof(Materials));

}

}

private string? \_Description;

public string? Description

{

get => \_Description;

set

{

\_Description = value;

OnPropertyChanged(nameof(Description));

}

}

}

}

using DatabaseManager;

using System.Globalization;

using System.Windows.Data;

namespace VacTrack.Converter

{

class CounterpartieTypeConverter : IValueConverter

{

const string UrName = "Юр. Лицо";

const string FizName = "Физ. Лицо";

private static IEnumerable<string> UrFiz => [UrName, FizName];

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is CounterpartieType type)

return type switch

{

CounterpartieType.Ur => UrName,

CounterpartieType.Fiz => FizName,

\_ => value

};

if (value is List<CounterpartieType>)

return UrFiz;

return value;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is string type)

return type switch

{

UrName => CounterpartieType.Ur,

FizName => CounterpartieType.Fiz,

\_ => throw new ArgumentException($"Неизвестный тип контрагента: \"{type}\". Проверьте корректность данных.")

};

throw new ArgumentException($"Ожидалась строка, но получен объект типа {value?.GetType().Name ?? "null"}. Проверьте входные данные.");

}

}

}

using System.Globalization;

using System.Windows.Data;

namespace VacTrack.Converter

{

public class DockPanelWidthConverter : IValueConverter

{

public int Length { get; set; }

public DockPanelWidthConverter()

{

Length = 620;

}

public DockPanelWidthConverter(int length)

{

Length = length;

}

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is double totalWidth)

{

// Вычитаем ширину кнопок и других фиксированных элементов

double reservedWidth = Length; // Задайте ширину кнопок/отступов

return Math.Max(0, totalWidth - reservedWidth);

}

return 0;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

using System.Globalization;

using System.Windows.Controls;

namespace VacTrack.Validators

{

public class LengthRangeValidationRule : ValidationRule

{

public int MinLength { get; set; } = 0; // Минимальная длина (по умолчанию — без ограничения)

public int MaxLength { get; set; } = int.MaxValue; // Максимальная длина (по умолчанию — без ограничения)

public int Length { get; set; } = 0;

public override ValidationResult Validate(object value, CultureInfo cultureInfo)

{

var input = (value ?? "").ToString();

if (Length == 0)

{

if (input.Length < MinLength)

return new ValidationResult(false, $"Не может быть менее {MinLength} символов.");

if (input.Length > MaxLength)

return new ValidationResult(false, $"Не может превышать {MaxLength} символов.");

}

else

{

if (input.Length != Length)

return new ValidationResult(false, $"Должно быть ровно {Length} символов.");

}

return ValidationResult.ValidResult;

}

}

}

using System.Globalization;

using System.Windows.Controls;

namespace VacTrack.Validators

{

public class NotEmptyValidationRule : ValidationRule

{

public override ValidationResult Validate(object value, CultureInfo cultureInfo)

{

return string.IsNullOrWhiteSpace((value ?? "").ToString())

? new ValidationResult(false, "Не может быть пустым.")

: ValidationResult.ValidResult;

}

}

}

using System.Globalization;

using System.Text.RegularExpressions;

using System.Windows.Controls;

namespace VacTrack.Validators

{

public class PhoneNumberValidationRule : ValidationRule

{

// Список регулярных выражений для проверки номера телефона

private readonly string[] \_patterns =

{

@"^\+?[1-9]\d{1,14}$", // Международный формат (E.164)

@"^(?:\+7|8)\d{10}$", // Российский номер (+7 или 8)

@"^\+?\d{1,4}[-\s]?\(?\d{1,3}\)?[-\s]?\d{1,4}[-\s]?\d{1,4}$" // С пробелами и дефисами

};

public override ValidationResult Validate(object value, CultureInfo cultureInfo)

{

var input = (value ?? "").ToString();

if (string.IsNullOrWhiteSpace(input))

{

return new ValidationResult(false, "Номер телефона не может быть пустым.");

}

foreach (var pattern in \_patterns)

{

if (Regex.IsMatch(input, pattern))

{

return ValidationResult.ValidResult;

}

}

return new ValidationResult(false, "Введите корректный номер телефона.");

}

}

}

using System.Globalization;

using System.Windows.Controls;

namespace VacTrack.Validators

{

public class PositiveNumberValidationRule : ValidationRule

{

public override ValidationResult Validate(object value, CultureInfo cultureInfo)

{

// Преобразуем входное значение в строку и проверяем, не пустое ли оно

string? input = (value ?? "").ToString();

if (string.IsNullOrWhiteSpace(input))

return new ValidationResult(false, "Значение не может быть пустым.");

// Пытаемся преобразовать в число

if (double.TryParse(input, NumberStyles.Float, cultureInfo, out double result))

{

// Проверяем, что число строго положительное

if (result > 0)

return ValidationResult.ValidResult;

else

return new ValidationResult(false, "Число должно быть строго положительным.");

}

// Если преобразование не удалось

return new ValidationResult(false, "Некорректное числовое значение.");

}

}

}

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Timers;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

namespace VacTrack.ViewReport

{

public abstract class BaseReportViewModel<T> : INotifyPropertyChanged where T : BaseModel

{

#region interface implemented

public event PropertyChangedEventHandler? PropertyChanged;

protected void SetProperty<Ts>(ref Ts field, Ts value, [CallerMemberName] string? propertyName = null)

{

if (!EqualityComparer<Ts>.Default.Equals(field, value))

{

field = value;

OnPropertyChanged(propertyName);

}

}

protected void OnPropertyChanged([CallerMemberName] string? name = null)

=> PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));

#endregion

#region properties

protected DatabaseContext Db;

protected DbSet<T> DbSet;

private System.Timers.Timer? \_resetTimer;

private ObservableCollection<T> \_Items;

public ObservableCollection<T> Items

{

get => \_Items;

set

{

if (\_Items != value)

{

\_Items = value;

Refresh(null);

SetProperty(ref \_Items, value);

}

}

}

private FlowDocument? \_Document;

public FlowDocument? Document

{

get => \_Document;

set => SetProperty(ref \_Document, value);

}

private string? \_Message;

public string? Message

{

get => \_Message;

set

{

\_Message = value;

OnPropertyChanged();

StartResetTimer();

}

}

private Brush? \_MessageBrush;

public Brush? MessageBrush

{

get => \_MessageBrush;

set

{

\_MessageBrush = value;

OnPropertyChanged();

}

}

#endregion

public ICommand RefreshCommand { get; }

public ICommand PrintCommand { get; }

public ICommand RotateDocCommand { get; }

public BaseReportViewModel()

{

Db = new DatabaseContext();

Db.Database.EnsureCreated();

LoadData();

if (DbSet == null || \_Items == null) throw new Exception("Data loading error");

RefreshCommand = new RelayCommand(Refresh);

PrintCommand = new RelayCommand(PrintReport);

RotateDocCommand = new RelayCommand(SwapPageDimensions);

}

public abstract FlowDocument CreateReport();

protected void Refresh(object? obj)

{

Document = CreateReport();

// Настройка размеров документа

Document.PageWidth = 793.7; // A4 ширина в пикселях (96 DPI)

Document.PageHeight = 1122.52; // A4 высота в пикселях (96 DPI)

Document.PagePadding = new Thickness(50); // Поля страницы

Document.ColumnWidth = double.PositiveInfinity; // Убрать колонки

}

protected virtual void LoadData()

{

DbSet = Db.Set<T>();

DbSet.Load();

Items = DbSet.Local.ToObservableCollection();

}

private void PrintReport(object obj)

{

if (Document == null)

{

Message = "Документ пуст";

MessageBrush = Brushes.Orange;

return;

}

PrintDialog printDialog = new();

if (printDialog.ShowDialog() == true)

{

// Установка размера печатной области

Document.PageWidth = printDialog.PrintableAreaWidth;

Document.PageHeight = printDialog.PrintableAreaHeight;

// Печать документа

printDialog.PrintDocument(((IDocumentPaginatorSource)Document).DocumentPaginator, "Печать отчёта");

}

}

public void OpenFromCache() => LoadData();

private void StartResetTimer()

{

// Останавливаем предыдущий таймер, если он существует

\_resetTimer?.Stop();

// Создаем новый таймер, который сработает через 10 секунд

\_resetTimer = new System.Timers.Timer(10000); // 10 секунд

\_resetTimer.Elapsed += ResetMessage;

\_resetTimer.Start();

}

private void ResetMessage(object? sender, ElapsedEventArgs e)

{

App.Current.Dispatcher.Invoke(() =>

{

Message = string.Empty;

});

\_resetTimer?.Stop();

}

private void SwapPageDimensions(object obj)

{

if (Document == null)

{

Message = "Документ пуст";

MessageBrush = Brushes.Orange;

return;

}

// Сохраняем текущие значения

double tempWidth = Document.PageWidth;

double tempHeight = Document.PageHeight;

// Меняем местами ширину и высоту

Document.PageWidth = tempHeight;

Document.PageHeight = tempWidth;

}

public void CreateGroupedRows<TKey>(

ref TableRowGroup dataGroup,

Func<T, TKey> keySelector, // Функция для определения ключа группировки

Func<TKey, IEnumerable<string>> groupHeaderSelector, // Формат заголовка группы

Func<T, IEnumerable<string>> rowSelector // Формат строки данных

)

{

var groupedItems = Items.GroupBy(keySelector);

foreach (var group in groupedItems)

{

dataGroup.Rows.Add(CreateRow(groupHeaderSelector(group.Key))); // Добавляем заголовок группы

foreach (var item in group)

{

dataGroup.Rows.Add(CreateRow(rowSelector(item))); // Добавляем строку данных

}

}

}

public void CreateGroupedRows<TKey>(

ref TableRowGroup dataGroup,

Func<T, TKey> keySelector, // Функция для определения ключа группировки

Func<T, double> getSum, // Функция для определения суммы

Func<TKey, IEnumerable<string>> groupHeaderSelector, // Формат заголовка группы

Func<double, IEnumerable<string>> groupTotalSelector, // Формат Итога группы

Func<T, IEnumerable<string>> rowSelector, // Формат строки данных

ref double totalSum,

bool IsGroupTotalEnabled

)

{

var groupedItems = Items.GroupBy(keySelector);

foreach (var group in groupedItems)

{

dataGroup.Rows.Add(CreateRow(groupHeaderSelector(group.Key))); // Добавляем заголовок группы

double summ = 0;

foreach (var item in group)

{

dataGroup.Rows.Add(CreateRow(rowSelector(item))); // Добавляем строку данных

summ += getSum(item);

}

if (IsGroupTotalEnabled) dataGroup.Rows.Add(CreateRow(groupTotalSelector(summ)));

totalSum += summ;

}

}

public void CreateGroupedRows<TKey>(

ref TableRowGroup dataGroup,

Func<T, TKey> keySelector, // Функция для определения ключа группировки

Func<T, double> getSum, // Функция для определения суммы

Func<T, double> getCount,

Func<TKey, IEnumerable<string>> groupHeaderSelector, // Формат заголовка группы

Func<(double, double), IEnumerable<string>> groupTotalSelector, // Формат Итога группы

Func<T, IEnumerable<string>> rowSelector, // Формат строки данных

ref double totalSum,

bool IsGroupTotalEnabled

)

{

var groupedItems = Items.GroupBy(keySelector);

foreach (var group in groupedItems)

{

dataGroup.Rows.Add(CreateRow(groupHeaderSelector(group.Key))); // Добавляем заголовок группы

double summ = 0;

double count = 0;

foreach (var item in group)

{

dataGroup.Rows.Add(CreateRow(rowSelector(item))); // Добавляем строку данных

summ += getSum(item);

count += getCount(item);

}

if (IsGroupTotalEnabled) dataGroup.Rows.Add(CreateRow(groupTotalSelector((count, summ))));

totalSum += summ;

}

}

static public TableRow CreateRow(IEnumerable<string> columns)

{

TableRow row = new();

foreach (var column in columns)

row.Cells.Add(new TableCell(new Paragraph(new Run(column ?? string.Empty))));

return row;

}

}

}

<Page x:Class="VacTrack.ViewReport.ContractorContractsReport"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewReport"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:converter="clr-namespace:VacTrack.Converter"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ContractorContractsReport">

<Page.DataContext>

<local:ContractorContractsReportViewModel/>

</Page.DataContext>

<Page.InputBindings>

<KeyBinding Key="P" Modifiers="Control" Command="{Binding PrintCommand}" />

</Page.InputBindings>

<Grid>

<DockPanel VerticalAlignment="Top" Height="40">

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RefreshCommand}"

ToolTip="Обновить">

<materialDesign:PackIcon Kind="Refresh" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding PrintCommand}"

ToolTip="Печать отчёта">

<materialDesign:PackIcon Kind="Printer" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RotateDocCommand}"

ToolTip="Смена ориентации страницы">

<materialDesign:PackIcon Kind="FileRotateRightOutline" />

</Button>

<materialDesign:PopupBox StaysOpen="True" Margin="20,0,0,0" ToolTip="Выбор типа групировки"

Style="{StaticResource MaterialDesignMultiFloatingActionPopupBox}"

Width="30" Height="30" PlacementMode="BottomAndAlignLeftEdges">

<materialDesign:PopupBox.ToggleContent>

<materialDesign:PackIcon Kind="FormatListGroupPlus" Width="30" Foreground="{DynamicResource MaterialDesignBody}"/>

</materialDesign:PopupBox.ToggleContent>

<StackPanel Orientation="Vertical" Margin="0,5,0,0">

<RadioButton

Content="Групировка по контрагенту"

GroupName="GroupType"

IsChecked="{Binding IsGroupedTypeGroupedByCounterpartie}"

Style="{StaticResource MaterialDesignChoiceChipPrimaryRadioButton}"

/>

<RadioButton

Content="Групировка по изделию"

GroupName="GroupType"

IsChecked="{Binding IsGroupedTypeGroupedByProduct}"

Style="{StaticResource MaterialDesignChoiceChipPrimaryRadioButton}"

/>

<RadioButton

Content="Без групировки"

GroupName="GroupType"

IsChecked="{Binding IsGroupedTypeNoGrouped}"

Style="{StaticResource MaterialDesignChoiceChipPrimaryRadioButton}"

/>

<Border Margin="0,15,0,0" Background="{DynamicResource MaterialDesign.Brush.Background}" CornerRadius="15">

<StackPanel>

<Label Content="Настройка итогов"/>

<ListBox Style="{StaticResource MaterialDesignFilterChipPrimaryListBox}" Height="46">

<ListBoxItem Content="Итого группы" IsSelected="{Binding IsGroupTotalEnabled}" IsEnabled="{Binding IsGroupedTypeNoGrouped, Converter={StaticResource InvertBooleanConverter}, UpdateSourceTrigger=PropertyChanged}"/>

<ListBoxItem Content="Общие итоги" IsSelected="{Binding AreOverallTotalsEnabled}"/>

</ListBox>

</StackPanel>

</Border>

</StackPanel>

</materialDesign:PopupBox>

<materialDesign:PopupBox StaysOpen="True" Margin="10,0,0,0" ToolTip="Настиройки фильтрации"

Style="{StaticResource MaterialDesignMultiFloatingActionPopupBox}"

Width="30" Height="30" PlacementMode="BottomAndAlignLeftEdges"

>

<materialDesign:PopupBox.ToggleContent>

<materialDesign:PackIcon Kind="FilterSettingsOutline" Width="30" Foreground="{DynamicResource MaterialDesignBody}"/>

</materialDesign:PopupBox.ToggleContent>

<StackPanel Orientation="Vertical" Width="200" Background="{DynamicResource MaterialDesign.Brush.Background}">

<ComboBox

Margin="0,10,0,0"

Style="{StaticResource MaterialDesignOutlinedComboBox}"

ItemsSource="{Binding ContractCounterpartie}"

SelectedItem="{Binding FilterByCounterpartie, UpdateSourceTrigger=PropertyChanged}"

DisplayMemberPath="Name"

materialDesign:ComboBoxAssist.ShowSelectedItem="True"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="Фильтр по котрагенту"

materialDesign:TextFieldAssist.HasClearButton="True"

/>

<ComboBox

Margin="0,10,0,0"

Style="{StaticResource MaterialDesignOutlinedComboBox}"

ItemsSource="{Binding ContractProduct}"

SelectedItem="{Binding FilterByProduct, UpdateSourceTrigger=PropertyChanged}"

DisplayMemberPath="Name"

materialDesign:ComboBoxAssist.ShowSelectedItem="True"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="Фильтр по изделию"

materialDesign:TextFieldAssist.HasClearButton="True"/>

<Label Content="За период" Margin="0,15,0,0"/>

<DatePicker

SelectedDate="{Binding FilterStartDate, UpdateSourceTrigger=PropertyChanged}"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="С"

Style="{StaticResource MaterialDesignOutlinedDatePicker}"

/>

<DatePicker

SelectedDate="{Binding FilterEndDate, UpdateSourceTrigger=PropertyChanged}"

Margin="0,5,0,0"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="По"

Style="{StaticResource MaterialDesignOutlinedDatePicker}"

/>

<Button

Content="Очистить фильтр"

Margin="0,15,0,0"

Style="{StaticResource MaterialDesignOutlinedButton}"

Command="{Binding ClearFilterCommand}"

/>

</StackPanel>

</materialDesign:PopupBox>

<TextBlock DockPanel.Dock="Right" Height="25" Text="{Binding Message}" ToolTip="{Binding Message}" Foreground="{Binding MessageBrush}" Margin="10,0,0,0"/>

</DockPanel>

<FlowDocumentReader Document="{Binding Document}" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" ViewingMode="Scroll" Foreground="{DynamicResource MaterialDesignBody}" Margin="0,40,0,0" BorderThickness="5" BorderBrush="#19737171"/>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Media;

namespace VacTrack.ViewReport

{

/// <summary>

/// Логика взаимодействия для ContractorContractsReport.xaml

/// </summary>

public partial class ContractorContractsReport : Page

{

private ContractorContractsReportViewModel ThisViewModel => (ContractorContractsReportViewModel)DataContext;

public ContractorContractsReport() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public enum ContractGroupedType { NoGrouped, GroupedByCounterpartie, GroupedByProduct }

public class ContractorContractsReportViewModel : BaseReportViewModel<Contract>

{

public ObservableCollection<Counterpartie>? ContractCounterpartie { get; set; }

public ObservableCollection<Product>? ContractProduct { get; set; }

#region properties

private ContractGroupedType GroupedType = ContractGroupedType.NoGrouped;

public bool IsGroupedTypeNoGrouped

{

get => GroupedType == ContractGroupedType.NoGrouped;

set

{

if (value)

{

GroupedType = ContractGroupedType.NoGrouped;

Refresh(null);

OnPropertyChanged(nameof(IsGroupedTypeNoGrouped));

}

}

}

public bool IsGroupedTypeGroupedByCounterpartie

{

get => GroupedType == ContractGroupedType.GroupedByCounterpartie;

set

{

if (value)

{

GroupedType = ContractGroupedType.GroupedByCounterpartie;

Refresh(null);

OnPropertyChanged(nameof(IsGroupedTypeGroupedByCounterpartie));

OnPropertyChanged(nameof(IsGroupedTypeNoGrouped));

}

}

}

public bool IsGroupedTypeGroupedByProduct

{

get => GroupedType == ContractGroupedType.GroupedByProduct;

set

{

if (value)

{

GroupedType = ContractGroupedType.GroupedByProduct;

Refresh(null);

OnPropertyChanged(nameof(IsGroupedTypeGroupedByProduct));

OnPropertyChanged(nameof(IsGroupedTypeNoGrouped));

}

}

}

private bool \_IsGroupTotalEnabled = true;

public bool IsGroupTotalEnabled

{

get => \_IsGroupTotalEnabled;

set

{

SetProperty(ref \_IsGroupTotalEnabled, value);

Refresh(null);

}

}

private bool \_AreOverallTotalsEnabled = true;

public bool AreOverallTotalsEnabled

{

get => \_AreOverallTotalsEnabled;

set

{

SetProperty(ref \_AreOverallTotalsEnabled, value);

Refresh(null);

}

}

private Counterpartie? \_FilterByCounterpartie;

public Counterpartie? FilterByCounterpartie

{

get => \_FilterByCounterpartie;

set

{

SetProperty(ref \_FilterByCounterpartie, value);

LoadData();

}

}

private Product? \_FilterByProduct;

public Product? FilterByProduct

{

get => \_FilterByProduct;

set

{

SetProperty(ref \_FilterByProduct, value);

LoadData();

}

}

private DateTime? \_FilterStartDate;

public DateTime? FilterStartDate

{

get => \_FilterStartDate;

set

{

SetProperty(ref \_FilterStartDate, value);

LoadData();

}

}

private DateTime? \_FilterEndDate;

public DateTime? FilterEndDate

{

get => \_FilterEndDate;

set

{

SetProperty(ref \_FilterEndDate, value);

LoadData();

}

}

#endregion

public System.Windows.Input.ICommand ClearFilterCommand { get; }

public ContractorContractsReportViewModel()

{

ClearFilterCommand = new RelayCommand(ClearFilter);

}

protected override void LoadData()

{

ContractCounterpartie = new ObservableCollection<Counterpartie>([.. Db.Counterparties]);

ContractProduct = new ObservableCollection<Product>([.. Db.Products]);

DbSet = Db.Set<Contract>();

DbSet.Include(e => e.Counterpartie)

.Include(e => e.Product)

.ThenInclude(e => e != null ? e.Unit : null)

.Load();

Items = new ObservableCollection<Contract>(

DbSet.Local.Where(item =>

(FilterByCounterpartie == null || item.Counterpartie?.Id == FilterByCounterpartie.Id) &&

(FilterByProduct == null || item.Product?.Id == FilterByProduct.Id) &&

((FilterStartDate == null && FilterEndDate == null) ||

(FilterStartDate != null && FilterEndDate != null &&

item.Date >= FilterStartDate && item.Date <= FilterEndDate) ||

(FilterStartDate != null && FilterEndDate == null && item.Date == FilterStartDate) ||

(FilterStartDate == null && FilterEndDate != null && item.Date == FilterEndDate))

).ToList());

}

public override FlowDocument CreateReport()

{

FlowDocument doc = new() { FontFamily = new FontFamily("Times New Roman"), FontSize = 12, PagePadding = new Thickness(50) };

AddHeader(ref doc); // Добавляем заголовок Отчёта

Table table = new() { CellSpacing = 3, BorderBrush = Brushes.Gray, BorderThickness = new Thickness(1) };

AddTableHeader(ref table); // Добавляем заголовок таблицы

TableRowGroup dataGroup = new();

double totalSum = 0;

switch (GroupedType)

{

case ContractGroupedType.GroupedByCounterpartie: CreateGroupedByCounterpartie(ref dataGroup, ref totalSum); break;

case ContractGroupedType.GroupedByProduct: CreateGroupedByProduct(ref dataGroup, ref totalSum); break;

case ContractGroupedType.NoGrouped: CreateNoGroupedRows(ref dataGroup, ref totalSum); break;

}

if (AreOverallTotalsEnabled)

dataGroup.Rows.Add(CreateRow(["Итого", "", "", "", "", "", $"{totalSum}"]));

table.RowGroups.Add(dataGroup);

doc.Blocks.Add(table);

return doc;

}

private void CreateGroupedByProduct(ref TableRowGroup dataGroup, ref double totalSum)

{

CreateGroupedRows(

ref dataGroup,

item => item.Product?.Name,

item => item.Summ,

item => item.Count,

key => ["", "", "", $"{key}", "", "", ""],

cntTotal => ["Итого", "", "", "", $"{cntTotal.Item1}", "", $"{cntTotal.Item2}"],

item => [

$"{item.Name}",

$"{item.Counterpartie?.Name}",

$"{item.Date:dd.MM.yyyy}",

string.Empty,

$"{item.Count}",

$"{item.Product?.Unit?.Name}",

$"{item.Summ}"

],

ref totalSum,

IsGroupTotalEnabled);

}

private void CreateGroupedByCounterpartie(ref TableRowGroup dataGroup, ref double totalSum)

{

CreateGroupedRows(

ref dataGroup,

item => item.Counterpartie?.Name,

item => item.Summ,

key => ["", $"{key}", "", "", "", "", ""],

total => ["Итого", "", "", "", "", "", $"{total}"],

item => [

$"{item.Name}",

String.Empty,

$"{item.Date:dd.MM.yyyy}",

$"{item.Product?.Name}",

$"{item.Count}",

$"{item.Product?.Unit?.Name}",

$"{item.Summ}"

],

ref totalSum,

IsGroupTotalEnabled);

}

private void CreateNoGroupedRows(ref TableRowGroup dataGroup, ref double totalSum)

{

foreach (var item in Items)

{

dataGroup.Rows.Add(CreateRow([

$"{item.Name}",

$"{item.Counterpartie?.Name}",

$"{item.Date:dd.MM.yyyy}",

$"{item.Product?.Name}",

$"{item.Count}",

$"{item.Product?.Unit?.Name}",

$"{item.Summ}"]));

totalSum += item.Summ;

}

}

private static void AddTableHeader(ref Table table)

{

TableRowGroup headerGroup = new();

TableRow headerRow = new();

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Название договора"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Контрагент"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Дата заключения"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Изделие"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Количество"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Единица измерения"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Сумма договора"))) { FontWeight = FontWeights.Bold });

headerGroup.Rows.Add(headerRow);

table.RowGroups.Add(headerGroup);

}

private void AddHeader(ref FlowDocument doc)

{

// Добавляем заголовок отчета

Paragraph header = new(new Run("ООО \"ВакТайм\""))

{

FontSize = 16,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(header);

Paragraph title = new(new Run("Отчет по договорам с контрагентами"))

{

FontSize = 14,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(title);

string filterByPost = FilterByCounterpartie == null ? string.Empty : $"\nПо контрагенту: {FilterByCounterpartie.Name}";

string filterByDivision = FilterByProduct == null ? string.Empty : $"\nПо изделию: {FilterByProduct.Name}";

string periodFilter = (FilterStartDate, FilterEndDate) switch

{

(not null, not null) => $"\nЗа период с {FilterStartDate:dd.MM.yyyy} по {FilterEndDate:dd.MM.yyyy}",

(not null, null) => $"\nНа день {FilterStartDate:dd.MM.yyyy}",

(null, not null) => $"\nНа день {FilterEndDate:dd.MM.yyyy}",

\_ => string.Empty

};

string filteredText = filterByPost + filterByDivision + periodFilter;

string headerText = $"Дата формирования: {DateTime.Now:dd.MM.yyyy}" + filteredText;

Paragraph headerParagraph = new(new Run(headerText))

{

FontSize = 12,

TextAlignment = TextAlignment.Left,

LineHeight = 1.5 // Можно добавить межстрочный интервал

};

doc.Blocks.Add(headerParagraph);

}

private void ClearFilter(object obj)

{

FilterByCounterpartie = null;

FilterByProduct = null;

FilterStartDate = null;

FilterEndDate = null;

}

}

} <Page x:Class="VacTrack.ViewReport.EmployeeDivisionReport"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewReport"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="EmployeeDivisionReport">

<Page.DataContext>

<local:EmployeeDivisionReportViewModel/>

</Page.DataContext>

<Page.InputBindings>

<KeyBinding Key="P" Modifiers="Control" Command="{Binding PrintCommand}" />

</Page.InputBindings>

<Grid>

<DockPanel VerticalAlignment="Top" Height="40">

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RefreshCommand}"

ToolTip="Обновить">

<materialDesign:PackIcon Kind="Refresh" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding PrintCommand}"

ToolTip="Печать отчёта">

<materialDesign:PackIcon Kind="Printer" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RotateDocCommand}"

ToolTip="Смена ориентации страницы">

<materialDesign:PackIcon Kind="FileRotateRightOutline" />

</Button>

<materialDesign:PopupBox StaysOpen="True" Margin="20,0,0,0" ToolTip="Выбор типа групировки"

Style="{StaticResource MaterialDesignMultiFloatingActionPopupBox}"

Width="30" Height="30" PlacementMode="BottomAndAlignLeftEdges">

<materialDesign:PopupBox.ToggleContent>

<materialDesign:PackIcon Kind="FormatListGroupPlus" Width="30" Foreground="{DynamicResource MaterialDesignBody}"/>

</materialDesign:PopupBox.ToggleContent>

<StackPanel Orientation="Vertical" Margin="0,5,0,0">

<RadioButton

Content="Групировка по должности"

GroupName="GroupType"

IsChecked="{Binding IsGroupedTypeGroupedByPost}"

Style="{StaticResource MaterialDesignChoiceChipPrimaryRadioButton}"

/>

<RadioButton

Content="Групировка по подразделению"

GroupName="GroupType"

IsChecked="{Binding IsGroupedTypeGroupedByDivision}"

Style="{StaticResource MaterialDesignChoiceChipPrimaryRadioButton}"

/>

<RadioButton

Content="Без групировки"

GroupName="GroupType"

IsChecked="{Binding IsGroupedTypeNoGrouped}"

Style="{StaticResource MaterialDesignChoiceChipPrimaryRadioButton}"

/>

</StackPanel>

</materialDesign:PopupBox>

<materialDesign:PopupBox StaysOpen="True" Margin="10,0,0,0" ToolTip="Настиройки фильтрации"

Style="{StaticResource MaterialDesignMultiFloatingActionPopupBox}"

Width="30" Height="30" PlacementMode="BottomAndAlignLeftEdges"

>

<materialDesign:PopupBox.ToggleContent>

<materialDesign:PackIcon Kind="FilterSettingsOutline" Width="30" Foreground="{DynamicResource MaterialDesignBody}"/>

</materialDesign:PopupBox.ToggleContent>

<StackPanel Orientation="Vertical" Width="200" Background="{DynamicResource MaterialDesign.Brush.Background}">

<ComboBox

Margin="0,10,0,0"

Style="{StaticResource MaterialDesignOutlinedComboBox}"

ItemsSource="{Binding EmployPosts}"

SelectedItem="{Binding FilterByPost, UpdateSourceTrigger=PropertyChanged}"

DisplayMemberPath="Name"

materialDesign:ComboBoxAssist.ShowSelectedItem="True"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="Фильтр по должности"

materialDesign:TextFieldAssist.HasClearButton="True"

/>

<ComboBox

Margin="0,10,0,0"

Style="{StaticResource MaterialDesignOutlinedComboBox}"

ItemsSource="{Binding EmployDivisions}"

SelectedItem="{Binding FilterByDivision, UpdateSourceTrigger=PropertyChanged}"

DisplayMemberPath="Name"

materialDesign:ComboBoxAssist.ShowSelectedItem="True"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="Фильтр по подразделению"

materialDesign:TextFieldAssist.HasClearButton="True"/>

<Label Content="За период" Margin="0,15,0,0"/>

<DatePicker

SelectedDate="{Binding FilterStartDate, UpdateSourceTrigger=PropertyChanged}"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="С"

Style="{StaticResource MaterialDesignOutlinedDatePicker}"

/>

<DatePicker

SelectedDate="{Binding FilterEndDate, UpdateSourceTrigger=PropertyChanged}"

Margin="0,5,0,0"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="По"

Style="{StaticResource MaterialDesignOutlinedDatePicker}"

/>

<Button

Content="Очистить фильтр"

Margin="0,15,0,0"

Style="{StaticResource MaterialDesignOutlinedButton}"

Command="{Binding ClearFilterCommand}"

/>

</StackPanel>

</materialDesign:PopupBox>

<TextBlock DockPanel.Dock="Right" Height="25" Text="{Binding Message}" ToolTip="{Binding Message}" Foreground="{Binding MessageBrush}" Margin="10,0,0,0"/>

</DockPanel>

<FlowDocumentReader Document="{Binding Document}" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" ViewingMode="Scroll" Foreground="{DynamicResource MaterialDesignBody}" Margin="0,40,0,0" BorderThickness="5" BorderBrush="#19737171"/>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

namespace VacTrack.ViewReport

{

/// <summary>

/// Логика взаимодействия для EmployeeDivisionReport.xaml

/// </summary>

public partial class EmployeeDivisionReport : Page, ICachedPage

{

private EmployeeDivisionReportViewModel ThisViewModel => (EmployeeDivisionReportViewModel)DataContext;

public EmployeeDivisionReport() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public enum GroupedType { NoGrouped, GroupedByDivision, GroupedByPost }

public class EmployeeDivisionReportViewModel : BaseReportViewModel<Employee>

{

public ObservableCollection<Division>? EmployDivisions { get; set; }

public ObservableCollection<Post>? EmployPosts { get; set; }

#region properties

private GroupedType GroupedType = GroupedType.NoGrouped;

public bool IsGroupedTypeNoGrouped

{

get => GroupedType == GroupedType.NoGrouped;

set

{

if (value)

{

GroupedType = GroupedType.NoGrouped;

Refresh(null);

OnPropertyChanged(nameof(IsGroupedTypeNoGrouped));

}

}

}

public bool IsGroupedTypeGroupedByDivision

{

get => GroupedType == GroupedType.GroupedByDivision;

set

{

if (value)

{

GroupedType = GroupedType.GroupedByDivision;

Refresh(null);

OnPropertyChanged(nameof(IsGroupedTypeGroupedByDivision));

}

}

}

public bool IsGroupedTypeGroupedByPost

{

get => GroupedType == GroupedType.GroupedByPost;

set

{

if (value)

{

GroupedType = GroupedType.GroupedByPost;

Refresh(null);

OnPropertyChanged(nameof(IsGroupedTypeGroupedByPost));

}

}

}

private Post? \_FilterByPost;

public Post? FilterByPost

{

get => \_FilterByPost;

set

{

SetProperty(ref \_FilterByPost, value);

LoadData();

}

}

private Division? \_FilterByDivision;

public Division? FilterByDivision

{

get => \_FilterByDivision;

set

{

SetProperty(ref \_FilterByDivision, value);

LoadData();

}

}

private DateTime? \_FilterStartDate;

public DateTime? FilterStartDate

{

get => \_FilterStartDate;

set

{

SetProperty(ref \_FilterStartDate, value);

LoadData();

}

}

private DateTime? \_FilterEndDate;

public DateTime? FilterEndDate

{

get => \_FilterEndDate;

set

{

SetProperty(ref \_FilterEndDate, value);

LoadData();

}

}

#endregion

public ICommand ClearFilterCommand { get; }

public EmployeeDivisionReportViewModel()

{

ClearFilterCommand = new RelayCommand(ClearFilter);

}

protected override void LoadData()

{

EmployDivisions = new ObservableCollection<Division>([.. Db.Divisions]);

EmployPosts = new ObservableCollection<Post>([.. Db.Posts]);

DbSet = Db.Set<Employee>();

DbSet.Include(e => e.Division).Include(e => e.Post).Load();

Items = new ObservableCollection<Employee>(

DbSet.Local.Where(item =>

// Фильтрация по должности

(FilterByPost == null || item.Post?.Id == FilterByPost.Id) &&

// Фильтрация по подразделению

(FilterByDivision == null || item.Division?.Id == FilterByDivision.Id) &&

// Проверка на наличие фильтров по датам

((FilterStartDate == null && FilterEndDate == null) ||

// Если заданы оба фильтра (FilterStartDate и FilterEndDate)

(FilterStartDate != null && FilterEndDate != null &&

// Проверяем, что DateHire или DateDismissal попадают в указанный диапазон

((item.DateHire >= FilterStartDate && item.DateHire <= FilterEndDate) ||

(item.DateDismissal >= FilterStartDate && item.DateDismissal <= FilterEndDate))) ||

// Если задан только FilterStartDate, фильтруем по точному совпадению с этой датой

(FilterStartDate != null && FilterEndDate == null &&

(item.DateHire == FilterStartDate || item.DateDismissal == FilterStartDate)) ||

// Если задан только FilterEndDate, фильтруем по точному совпадению с этой датой

(FilterStartDate == null && FilterEndDate != null &&

(item.DateHire == FilterEndDate || item.DateDismissal == FilterEndDate)))

).ToList());

}

public override FlowDocument CreateReport()

{

FlowDocument doc = new() { FontFamily = new FontFamily("Times New Roman"), FontSize = 12, PagePadding = new Thickness(50) };

AddHeader(ref doc); // Добавляем заголовок Отчёта

Table table = new() { CellSpacing = 3, BorderBrush = Brushes.Gray, BorderThickness = new Thickness(1) };

AddTableHeader(ref table); // Добавляем заголовок таблицы

TableRowGroup dataGroup = new();

switch (GroupedType)

{

case GroupedType.GroupedByDivision: CreateGroupedByDivisionRows(ref dataGroup); break;

case GroupedType.GroupedByPost: CreateGroupedByPostRows(ref dataGroup); break;

case GroupedType.NoGrouped: CreateNoGroupedRows(ref dataGroup); break;

}

table.RowGroups.Add(dataGroup);

doc.Blocks.Add(table);

return doc;

}

private void CreateGroupedByDivisionRows(ref TableRowGroup dataGroup)

{

CreateGroupedRows(

ref dataGroup,

item => item.Division?.Name,

key => ["", $"{key}", "", "", ""], // Заголовок группы

item => [

$"{item.Fio}",

string.Empty,

$"{item.Post?.Name}",

$"{item.DateHire:dd.MM.yyyy}",

$"{item.DateDismissal:dd.MM.yyyy}"]

);

}

private void CreateGroupedByPostRows(ref TableRowGroup dataGroup)

{

CreateGroupedRows(

ref dataGroup,

item => item.Post?.Name,

key => ["", "", $"{key}", "", ""], // Заголовок группы

item => [

$"{item.Fio}",

$"{item.Division?.Name}",

string.Empty,

$"{item.DateHire:dd.MM.yyyy}",

$"{item.DateDismissal:dd.MM.yyyy}"]

);

}

private void CreateNoGroupedRows(ref TableRowGroup dataGroup)

{

foreach (var item in Items)

{

dataGroup.Rows.Add(CreateRow([

$"{item.Fio}",

$"{item.Division?.Name}",

$"{item.Post?.Name}",

$"{item.DateHire:dd.MM.yyyy}",

$"{item.DateDismissal:dd.MM.yyyy}"]));

}

}

private static void AddTableHeader(ref Table table)

{

TableRowGroup headerGroup = new();

TableRow headerRow = new();

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("ФИО"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Подразделение"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Должность"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Дата приема"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Дата увольнения"))) { FontWeight = FontWeights.Bold });

headerGroup.Rows.Add(headerRow);

table.RowGroups.Add(headerGroup);

}

private void AddHeader(ref FlowDocument doc)

{

// Добавляем заголовок отчета

Paragraph header = new(new Run("ООО \"ВакТайм\""))

{

FontSize = 16,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(header);

Paragraph title = new(new Run("Отчет по сотрудникам и их подразделениям"))

{

FontSize = 14,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(title);

string filterByPost = FilterByPost == null ? string.Empty : $"\nПо должности: {FilterByPost.Name}";

string filterByDivision = FilterByDivision == null ? string.Empty : $"\nПо подразделению: {FilterByDivision.Name}";

//string periodFilter =

// FilterStartDate != null && FilterEndDate != null ? $"\nЗа период с {FilterStartDate:dd.MM.yyyy} по {FilterEndDate:dd.MM.yyyy}" :

// FilterStartDate != null && FilterEndDate == null ? $"\nНа день {FilterStartDate:dd.MM.yyyy}" :

// FilterStartDate == null && FilterEndDate != null ? $"\nНа день {FilterEndDate:dd.MM.yyyy}" : string.Empty;

string periodFilter = (FilterStartDate, FilterEndDate) switch

{

(not null, not null) => $"\nЗа период с {FilterStartDate:dd.MM.yyyy} по {FilterEndDate:dd.MM.yyyy}",

(not null, null) => $"\nНа день {FilterStartDate:dd.MM.yyyy}",

(null, not null) => $"\nНа день {FilterEndDate:dd.MM.yyyy}",

\_ => string.Empty

};

string filteredText = filterByPost + filterByDivision + periodFilter;

string headerText = $"Дата формирования: {DateTime.Now:dd.MM.yyyy}" + filteredText;

Paragraph headerParagraph = new(new Run(headerText))

{

FontSize = 12,

TextAlignment = TextAlignment.Left,

LineHeight = 1.5 // Можно добавить межстрочный интервал

};

doc.Blocks.Add(headerParagraph);

}

private void ClearFilter(object obj)

{

FilterByDivision = null;

FilterByPost = null;

FilterStartDate = null;

FilterEndDate = null;

}

}

} <Page x:Class="VacTrack.ViewReport.MaterialUsageReport"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewReport"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="MaterialUsageReport">

<Page.DataContext>

<local:MaterialUsageReportViewModel/>

</Page.DataContext>

<Page.InputBindings>

<KeyBinding Key="P" Modifiers="Control" Command="{Binding PrintCommand}" />

</Page.InputBindings>

<Grid>

<DockPanel VerticalAlignment="Top" Height="40">

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RefreshCommand}"

ToolTip="Обновить">

<materialDesign:PackIcon Kind="Refresh" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding PrintCommand}"

ToolTip="Печать отчёта">

<materialDesign:PackIcon Kind="Printer" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RotateDocCommand}"

ToolTip="Смена ориентации страницы">

<materialDesign:PackIcon Kind="FileRotateRightOutline" />

</Button>

<ListBox Style="{StaticResource MaterialDesignFilterChipPrimaryListBox}" Height="46" Margin="20,0,0,0">

<ListBoxItem Content="Группировка" IsSelected="{Binding IsGroupingEnabled}"/>

<ListBoxItem Content="Итого группы" IsSelected="{Binding IsGroupTotalEnabled}" IsEnabled="{Binding IsGroupingEnabled}"/>

<ListBoxItem Content="Общие итоги" IsSelected="{Binding AreOverallTotalsEnabled}"/>

</ListBox>

<ComboBox

ItemsSource="{Binding Products}"

Style="{StaticResource MaterialDesignFloatingHintComboBox}"

Width="130"

DisplayMemberPath="Name"

materialDesign:ComboBoxAssist.ShowSelectedItem="True"

SelectedItem="{Binding SelectedProduct}"

Margin="20,0,0,0"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="Фильтр по изделию"

materialDesign:TextFieldAssist.HasClearButton="True"/>

<TextBlock DockPanel.Dock="Right" Height="25" Text="{Binding Message}" ToolTip="{Binding Message}" Foreground="{Binding MessageBrush}" Margin="10,0,0,0"/>

</DockPanel>

<FlowDocumentReader Document="{Binding Document}" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" ViewingMode="Scroll" Foreground="{DynamicResource MaterialDesignBody}" Margin="0,40,0,0" BorderThickness="5" BorderBrush="#19737171"/>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Media;

namespace VacTrack.ViewReport

{

/// <summary>

/// Логика взаимодействия для MaterialUsageReport.xaml

/// </summary>

public partial class MaterialUsageReport : Page

{

private MaterialUsageReportViewModel ThisViewModel => (MaterialUsageReportViewModel)DataContext;

public MaterialUsageReport() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class MaterialUsageReportViewModel : BaseReportViewModel<Product\_Material>

{

public List<Product>? Products { get; set; }

#region properties

private bool \_IsGroupingEnabled = true;

public bool IsGroupingEnabled

{

get => \_IsGroupingEnabled;

set

{

SetProperty(ref \_IsGroupingEnabled, value);

Refresh(null);

}

}

private bool \_IsGroupTotalEnabled = true;

public bool IsGroupTotalEnabled

{

get => \_IsGroupTotalEnabled;

set

{

SetProperty(ref \_IsGroupTotalEnabled, value);

Refresh(null);

}

}

private bool \_AreOverallTotalsEnabled = true;

public bool AreOverallTotalsEnabled

{

get => \_AreOverallTotalsEnabled;

set

{

SetProperty(ref \_AreOverallTotalsEnabled, value);

Refresh(null);

}

}

private Product? \_SelectedProduct;

public Product? SelectedProduct

{

get => \_SelectedProduct;

set

{

SetProperty(ref \_SelectedProduct, value);

LoadData();

}

}

#endregion

protected override void LoadData()

{

Products = new List<Product>([.. Db.Products]);

DbSet = Db.Set<Product\_Material>();

DbSet.Include(e => e.Material).ThenInclude(c => c != null ? c.Unit : null).Include(e => e.Product).Load();

if (SelectedProduct == null)

Items = DbSet.Local.ToObservableCollection();

else

Items = new ObservableCollection<Product\_Material>(DbSet.Local.Where(

item => item.Product?.Id == SelectedProduct.Id).ToList());

}

public override FlowDocument CreateReport()

{

FlowDocument doc = new() { FontFamily = new FontFamily("Times New Roman"), FontSize = 12, PagePadding = new Thickness(50) };

AddHeader(ref doc, SelectedProduct); // Добавляем заголовок Отчёта

Table table = new() { CellSpacing = 3, BorderBrush = Brushes.Gray, BorderThickness = new Thickness(1) };

table.Columns.Add(new TableColumn { Width = new GridLength(40) });

AddTableHeader(ref table); // Добавляем заголовок таблицы

// Данные таблицы

TableRowGroup dataGroup = new();

double totalSumm = 0;

if (IsGroupingEnabled)

CreateGroupedRows(ref dataGroup, ref totalSumm);

else

CreateUngroupedRows(ref dataGroup, ref totalSumm);

if (AreOverallTotalsEnabled) dataGroup.Rows.Add(CreateRow(["Итого", "", "", "", "", "", $"{totalSumm}"]));

table.RowGroups.Add(dataGroup);

doc.Blocks.Add(table);

return doc;

}

private void CreateGroupedRows(ref TableRowGroup dataGroup, ref double totalSum)

{

CreateGroupedRows(

ref dataGroup,

item => item.Product?.Name,

item => item.GetSum,

key => ["", $"{key}", "", "", "", "", ""],

total => ["Итого", "", "", "", "", "", $"{total}"],

item => [

$"{item.Id}",

String.Empty,

$"{item.Material?.Name}",

$"{item.Quantity}",

$"{item.Material?.Unit?.Name}",

$"{item.Material?.Price}",

$"{item.GetSum}"

],

ref totalSum,

IsGroupTotalEnabled);

}

private void CreateUngroupedRows(ref TableRowGroup dataGroup, ref double totalSumm)

{

foreach (var item in Items)

{

double price = item.GetSum;

totalSumm += price;

dataGroup.Rows.Add(CreateRow([

$"{item.Id}",

$"{item.Product?.Name}",

$"{item.Material?.Name}",

$"{item.Quantity}",

$"{item.Material?.Unit?.Name}",

$"{item.Material?.Price}",

$"{price}"]));

}

}

private static void AddTableHeader(ref Table table)

{

TableRowGroup headerGroup = new();

TableRow headerRow = new();

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Код"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Наименование изделия"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Наименование материала"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Количество"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Единица измерения"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Стоимость за единицу"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Стоимость"))) { FontWeight = FontWeights.Bold });

headerGroup.Rows.Add(headerRow);

table.RowGroups.Add(headerGroup);

}

private static void AddHeader(ref FlowDocument doc, Product? SelectedProduct)

{

// Добавляем заголовок отчета

Paragraph header = new(new Run("ООО \"ВакТайм\""))

{

FontSize = 16,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(header);

Paragraph title = new(new Run("Отчет по использованию материалов на изделия"))

{

FontSize = 14,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(title);

string filteredText = SelectedProduct == null ? "" : $"\nДля изделия: {SelectedProduct.Name}";

string headerText = $"Дата формирования: {DateTime.Now:dd.MM.yyyy}" + filteredText;

Paragraph headerParagraph = new(new Run(headerText))

{

FontSize = 12,

TextAlignment = TextAlignment.Left,

LineHeight = 1.5 // Можно добавить межстрочный интервал

};

doc.Blocks.Add(headerParagraph);

}

}

}

<Page x:Class="VacTrack.ViewReport.ProductSalesReport"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewReport"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ProductSalesReport">

<Page.DataContext>

<local:ProductSalesReportViewModel/>

</Page.DataContext>

<Page.InputBindings>

<KeyBinding Key="P" Modifiers="Control" Command="{Binding PrintCommand}" />

</Page.InputBindings>

<Grid>

<DockPanel VerticalAlignment="Top" Height="40">

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RefreshCommand}"

ToolTip="Обновить">

<materialDesign:PackIcon Kind="Refresh" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding PrintCommand}"

ToolTip="Печать отчёта">

<materialDesign:PackIcon Kind="Printer" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RotateDocCommand}"

ToolTip="Смена ориентации страницы">

<materialDesign:PackIcon Kind="FileRotateRightOutline" />

</Button>

<materialDesign:PopupBox StaysOpen="True" Margin="20,0,0,0" ToolTip="Выбор типа групировки"

Style="{StaticResource MaterialDesignMultiFloatingActionPopupBox}"

Width="30" Height="30" PlacementMode="BottomAndAlignLeftEdges">

<materialDesign:PopupBox.ToggleContent>

<materialDesign:PackIcon Kind="FormatListGroupPlus" Width="30" Foreground="{DynamicResource MaterialDesignBody}"/>

</materialDesign:PopupBox.ToggleContent>

<StackPanel Orientation="Vertical" Margin="0,5,0,0">

<RadioButton

Content="Групировка по контрагенту"

GroupName="GroupType"

IsChecked="{Binding IsGroupedTypeGroupedByCounterpartie}"

Style="{StaticResource MaterialDesignChoiceChipPrimaryRadioButton}"

/>

<RadioButton

Content="Групировка по изделию"

GroupName="GroupType"

IsChecked="{Binding IsGroupedTypeGroupedByProduct}"

Style="{StaticResource MaterialDesignChoiceChipPrimaryRadioButton}"

/>

<RadioButton

Content="Без групировки"

GroupName="GroupType"

IsChecked="{Binding IsGroupedTypeNoGrouped}"

Style="{StaticResource MaterialDesignChoiceChipPrimaryRadioButton}"

/>

<Border Margin="0,15,0,0" Background="{DynamicResource MaterialDesign.Brush.Background}" CornerRadius="15">

<StackPanel>

<Label Content="Настройка итогов"/>

<ListBox Style="{StaticResource MaterialDesignFilterChipPrimaryListBox}" Height="46">

<ListBoxItem Content="Итого группы" IsSelected="{Binding IsGroupTotalEnabled}" IsEnabled="{Binding IsGroupedTypeNoGrouped, Converter={StaticResource InvertBooleanConverter}, UpdateSourceTrigger=PropertyChanged}"/>

<ListBoxItem Content="Общие итоги" IsSelected="{Binding AreOverallTotalsEnabled}"/>

</ListBox>

</StackPanel>

</Border>

</StackPanel>

</materialDesign:PopupBox>

<materialDesign:PopupBox StaysOpen="True" Margin="10,0,0,0" ToolTip="Настиройки фильтрации"

Style="{StaticResource MaterialDesignMultiFloatingActionPopupBox}"

Width="30" Height="30" PlacementMode="BottomAndAlignLeftEdges">

<materialDesign:PopupBox.ToggleContent>

<materialDesign:PackIcon Kind="FilterSettingsOutline" Width="30" Foreground="{DynamicResource MaterialDesignBody}"/>

</materialDesign:PopupBox.ToggleContent>

<StackPanel Orientation="Vertical" Width="200" Background="{DynamicResource MaterialDesign.Brush.Background}">

<ComboBox

Margin="0,10,0,0"

Style="{StaticResource MaterialDesignOutlinedComboBox}"

ItemsSource="{Binding ContractCounterpartie}"

SelectedItem="{Binding FilterByCounterpartie, UpdateSourceTrigger=PropertyChanged}"

DisplayMemberPath="Name"

materialDesign:ComboBoxAssist.ShowSelectedItem="True"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="Фильтр по котрагенту"

materialDesign:TextFieldAssist.HasClearButton="True"

/>

<ComboBox

Margin="0,10,0,0"

Style="{StaticResource MaterialDesignOutlinedComboBox}"

ItemsSource="{Binding ContractProduct}"

SelectedItem="{Binding FilterByProduct, UpdateSourceTrigger=PropertyChanged}"

DisplayMemberPath="Name"

materialDesign:ComboBoxAssist.ShowSelectedItem="True"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="Фильтр по изделию"

materialDesign:TextFieldAssist.HasClearButton="True"/>

<Label Content="За период" Margin="0,15,0,0"/>

<DatePicker

SelectedDate="{Binding FilterStartDate, UpdateSourceTrigger=PropertyChanged}"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="С"

Style="{StaticResource MaterialDesignOutlinedDatePicker}"

/>

<DatePicker

SelectedDate="{Binding FilterEndDate, UpdateSourceTrigger=PropertyChanged}"

Margin="0,5,0,0"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="По"

Style="{StaticResource MaterialDesignOutlinedDatePicker}"

/>

<Button

Content="Очистить фильтр"

Margin="0,15,0,0"

Style="{StaticResource MaterialDesignOutlinedButton}"

Command="{Binding ClearFilterCommand}"

/>

</StackPanel>

</materialDesign:PopupBox>

<TextBlock DockPanel.Dock="Right" Height="25" Text="{Binding Message}" ToolTip="{Binding Message}" Foreground="{Binding MessageBrush}" Margin="10,0,0,0"/>

</DockPanel>

<FlowDocumentReader Document="{Binding Document}" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" ViewingMode="Scroll" Foreground="{DynamicResource MaterialDesignBody}" Margin="0,40,0,0" BorderThickness="5" BorderBrush="#19737171"/>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Media;

namespace VacTrack.ViewReport

{

/// <summary>

/// Логика взаимодействия для ProductSalesReport.xaml

/// </summary>

public partial class ProductSalesReport : Page

{

private ProductSalesReportViewModel ThisViewModel => (ProductSalesReportViewModel)DataContext;

public ProductSalesReport() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public enum SaleGroupedType { NoGrouped, GroupedByCounterpartie, GroupedByProduct }

public class ProductSalesReportViewModel : BaseReportViewModel<Sale>

{

public ObservableCollection<Counterpartie>? ContractCounterpartie { get; set; }

public ObservableCollection<Product>? ContractProduct { get; set; }

#region properties

private SaleGroupedType GroupedType = SaleGroupedType.NoGrouped;

public bool IsGroupedTypeNoGrouped

{

get => GroupedType == SaleGroupedType.NoGrouped;

set

{

if (value)

{

GroupedType = SaleGroupedType.NoGrouped;

Refresh(null);

OnPropertyChanged(nameof(IsGroupedTypeNoGrouped));

}

}

}

public bool IsGroupedTypeGroupedByCounterpartie

{

get => GroupedType == SaleGroupedType.GroupedByCounterpartie;

set

{

if (value)

{

GroupedType = SaleGroupedType.GroupedByCounterpartie;

Refresh(null);

OnPropertyChanged(nameof(IsGroupedTypeGroupedByCounterpartie));

OnPropertyChanged(nameof(IsGroupedTypeNoGrouped));

}

}

}

public bool IsGroupedTypeGroupedByProduct

{

get => GroupedType == SaleGroupedType.GroupedByProduct;

set

{

if (value)

{

GroupedType = SaleGroupedType.GroupedByProduct;

Refresh(null);

OnPropertyChanged(nameof(IsGroupedTypeGroupedByProduct));

OnPropertyChanged(nameof(IsGroupedTypeNoGrouped));

}

}

}

private bool \_IsGroupTotalEnabled = true;

public bool IsGroupTotalEnabled

{

get => \_IsGroupTotalEnabled;

set

{

SetProperty(ref \_IsGroupTotalEnabled, value);

Refresh(null);

}

}

private bool \_AreOverallTotalsEnabled = true;

public bool AreOverallTotalsEnabled

{

get => \_AreOverallTotalsEnabled;

set

{

SetProperty(ref \_AreOverallTotalsEnabled, value);

Refresh(null);

}

}

private Counterpartie? \_FilterByCounterpartie;

public Counterpartie? FilterByCounterpartie

{

get => \_FilterByCounterpartie;

set

{

SetProperty(ref \_FilterByCounterpartie, value);

LoadData();

}

}

private Product? \_FilterByProduct;

public Product? FilterByProduct

{

get => \_FilterByProduct;

set

{

SetProperty(ref \_FilterByProduct, value);

LoadData();

}

}

private DateTime? \_FilterStartDate;

public DateTime? FilterStartDate

{

get => \_FilterStartDate;

set

{

SetProperty(ref \_FilterStartDate, value);

LoadData();

}

}

private DateTime? \_FilterEndDate;

public DateTime? FilterEndDate

{

get => \_FilterEndDate;

set

{

SetProperty(ref \_FilterEndDate, value);

LoadData();

}

}

#endregion

public System.Windows.Input.ICommand ClearFilterCommand { get; }

public ProductSalesReportViewModel()

{

ClearFilterCommand = new RelayCommand(ClearFilter);

}

protected override void LoadData()

{

ContractCounterpartie = new ObservableCollection<Counterpartie>([.. Db.Counterparties]);

ContractProduct = new ObservableCollection<Product>([.. Db.Products]);

DbSet = Db.Set<Sale>();

DbSet.Include(e => e.Contract)

.Include(c => c.Contract != null ? c.Contract.Product : null)

.ThenInclude(e => e != null ? e.Unit : null)

.Include(c => c.Contract != null ? c.Contract.Counterpartie : null)

.Load();

Items = new ObservableCollection<Sale>(

DbSet.Local.Where(item =>

(FilterByCounterpartie == null || item.Contract?.Counterpartie?.Id == FilterByCounterpartie.Id) &&

(FilterByProduct == null || item.Contract?.Product?.Id == FilterByProduct.Id) &&

((FilterStartDate == null && FilterEndDate == null) ||

(FilterStartDate != null && FilterEndDate != null &&

item.Date >= FilterStartDate && item.Date <= FilterEndDate) ||

(FilterStartDate != null && FilterEndDate == null && item.Date == FilterStartDate) ||

(FilterStartDate == null && FilterEndDate != null && item.Date == FilterEndDate))

).ToList());

}

public override FlowDocument CreateReport()

{

FlowDocument doc = new() { FontFamily = new FontFamily("Times New Roman"), FontSize = 12, PagePadding = new Thickness(50) };

AddHeader(ref doc); // Добавляем заголовок Отчёта

Table table = new() { CellSpacing = 3, BorderBrush = Brushes.Gray, BorderThickness = new Thickness(1) };

AddTableHeader(ref table); // Добавляем заголовок таблицы

TableRowGroup dataGroup = new();

double totalSum = 0;

switch (GroupedType)

{

case SaleGroupedType.GroupedByCounterpartie: CreateGroupedByCounterpartie(ref dataGroup, ref totalSum); break;

case SaleGroupedType.GroupedByProduct: CreateGroupedByProduct(ref dataGroup, ref totalSum); break;

case SaleGroupedType.NoGrouped: CreateNoGroupedRows(ref dataGroup, ref totalSum); break;

}

if (AreOverallTotalsEnabled)

dataGroup.Rows.Add(CreateRow(["Итого", "", "", "", "", $"{totalSum}"]));

table.RowGroups.Add(dataGroup);

doc.Blocks.Add(table);

return doc;

}

private void CreateGroupedByCounterpartie(ref TableRowGroup dataGroup, ref double totalSum)

{

CreateGroupedRows(

ref dataGroup,

item => item.Contract?.Counterpartie?.Name,

item => item.Summ,

key => ["", $"{key}", "", "", "", ""],

total => ["Итого", "", "", "", "", $"{total}"],

item => [

$"{item.Date:dd.MM.yyyy}",

String.Empty,

$"{item.Contract?.Product?.Name}",

$"{item.Count}",

$"{item.Contract?.Product?.Unit?.Name}",

$"{item.Summ}"

],

ref totalSum,

IsGroupTotalEnabled);

}

private void CreateGroupedByProduct(ref TableRowGroup dataGroup, ref double totalSum)

{

CreateGroupedRows(

ref dataGroup,

item => item.Contract?.Product?.Name,

item => item.Summ,

item => item.Count,

key => ["", "", $"{key}", "", "", ""],

cntTotal => ["Итого", "", "", $"{cntTotal.Item1}", "", $"{cntTotal.Item2}"],

item => [

$"{item.Date:dd.MM.yyyy}",

$"{item.Contract?.Counterpartie?.Name}",

String.Empty,

$"{item.Count}",

$"{item.Contract?.Product?.Unit?.Name}",

$"{item.Summ}"

],

ref totalSum,

IsGroupTotalEnabled);

}

private void CreateNoGroupedRows(ref TableRowGroup dataGroup, ref double totalSum)

{

foreach (var item in Items)

{

dataGroup.Rows.Add(CreateRow([

$"{item.Date:dd.MM.yyyy}",

$"{item.Contract?.Counterpartie?.Name}",

$"{item.Contract?.Product?.Name}",

$"{item.Count}",

$"{item.Contract?.Product?.Unit?.Name}",

$"{item.Summ}"]));

totalSum += item.Summ;

}

}

private static void AddTableHeader(ref Table table)

{

TableRowGroup headerGroup = new();

TableRow headerRow = new();

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Дата реализации"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Контрагент"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Наименование изделия"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Количество"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Единица измерения"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Сумма реализации"))) { FontWeight = FontWeights.Bold });

headerGroup.Rows.Add(headerRow);

table.RowGroups.Add(headerGroup);

}

private void AddHeader(ref FlowDocument doc)

{

// Добавляем заголовок отчета

Paragraph header = new(new Run("ООО \"ВакТайм\""))

{

FontSize = 16,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(header);

Paragraph title = new(new Run("Отчет по реализации продукции"))

{

FontSize = 14,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(title);

string filterByPost = FilterByCounterpartie == null ? string.Empty : $"\nПо контрагенту: {FilterByCounterpartie.Name}";

string filterByDivision = FilterByProduct == null ? string.Empty : $"\nПо изделию: {FilterByProduct.Name}";

string periodFilter = (FilterStartDate, FilterEndDate) switch

{

(not null, not null) => $"\nЗа период с {FilterStartDate:dd.MM.yyyy} по {FilterEndDate:dd.MM.yyyy}",

(not null, null) => $"\nНа день {FilterStartDate:dd.MM.yyyy}",

(null, not null) => $"\nНа день {FilterEndDate:dd.MM.yyyy}",

\_ => string.Empty

};

string filteredText = filterByPost + filterByDivision + periodFilter;

string headerText = $"Дата формирования: {DateTime.Now:dd.MM.yyyy}" + filteredText;

Paragraph headerParagraph = new(new Run(headerText))

{

FontSize = 12,

TextAlignment = TextAlignment.Left,

LineHeight = 1.5 // Можно добавить межстрочный интервал

};

doc.Blocks.Add(headerParagraph);

}

private void ClearFilter(object obj)

{

FilterByCounterpartie = null;

FilterByProduct = null;

FilterStartDate = null;

FilterEndDate = null;

}

}

} <Window x:Class="VacTrack.ViewReport.ReceiptReport"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:VacTrack.ViewReport"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

Style="{StaticResource CustomWindowStyle}"

mc:Ignorable="d"

Title="Печать накладной" Height="450" Width="800">

<Window.DataContext>

<local:ReceiptReportViewModel/>

</Window.DataContext>

<Window.InputBindings>

<KeyBinding Key="P" Modifiers="Control" Command="{Binding PrintCommand}" />

</Window.InputBindings>

<Grid Background="{DynamicResource MaterialDesign.Brush.Background}">

<DockPanel VerticalAlignment="Top" Height="40">

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RefreshCommand}"

ToolTip="Обновить">

<materialDesign:PackIcon Kind="Refresh" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding PrintCommand}"

ToolTip="Печать отчёта">

<materialDesign:PackIcon Kind="Printer" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RotateDocCommand}"

ToolTip="Смена ориентации страницы">

<materialDesign:PackIcon Kind="FileRotateRightOutline" />

</Button>

<TextBlock DockPanel.Dock="Right" Height="25" Text="{Binding Message}" ToolTip="{Binding Message}" Foreground="{Binding MessageBrush}" Margin="10,0,0,0"/>

</DockPanel>

<FlowDocumentReader Document="{Binding Document}" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" ViewingMode="Scroll" Foreground="{DynamicResource MaterialDesignBody}" Margin="0,40,0,0" BorderThickness="5" BorderBrush="#19737171"/>

</Grid>

</Window>

using DatabaseManager;

using System.Globalization;

using System.Windows;

using System.Windows.Documents;

using System.Windows.Media;

namespace VacTrack.ViewReport

{

/// <summary>

/// Логика взаимодействия для ReceiptReport.xaml

/// </summary>

public partial class ReceiptReport : Window

{

private ReceiptReportViewModel ThisViewModel => (ReceiptReportViewModel)DataContext;

public ReceiptReport(Receipt receipt)

{

InitializeComponent();

ThisViewModel.SelectedReceipt = receipt;

}

}

public class ReceiptReportViewModel : BaseReportViewModel<Receipt>

{

private Receipt? \_SelectedReceipt;

public Receipt? SelectedReceipt

{

get => \_SelectedReceipt;

set

{

SetProperty(ref \_SelectedReceipt, value);

Refresh(null);

}

}

public override FlowDocument CreateReport()

{

FlowDocument doc = new() { FontFamily = new FontFamily("Times New Roman"), FontSize = 12, PagePadding = new Thickness(50) };

AddHeader(ref doc); // Добавляем заголовок Отчёта

Table table = new() { CellSpacing = 3, BorderBrush = Brushes.Gray, BorderThickness = new Thickness(1) };

AddTableHeader(ref table); // Добавляем заголовок таблицы

TableRowGroup dataGroup = new();

double totalSum = 0;

CreateNoGroupedRows(ref dataGroup, ref totalSum);

dataGroup.Rows.Add(CreateRow(["Итого", "", "", "", $"{totalSum}"]));

table.RowGroups.Add(dataGroup);

doc.Blocks.Add(table);

Paragraph headerParagraph = new(new Run($" \_\_\_\_\_\_\_\_\_\_\_\_\_ {SelectedReceipt?.Material?.Location?.Employee?.Fio}"))

{

FontSize = 12,

TextAlignment = TextAlignment.Left,

LineHeight = 1.5 // Можно добавить межстрочный интервал

};

doc.Blocks.Add(headerParagraph);

return doc;

}

private void CreateNoGroupedRows(ref TableRowGroup dataGroup, ref double totalSum)

{

dataGroup.Rows.Add(CreateRow([

$"{SelectedReceipt?.Material?.Name}",

$"{SelectedReceipt?.Count}",

$"{SelectedReceipt?.Material?.Unit?.Name}",

$"{SelectedReceipt?.Material?.Price}",

$"{SelectedReceipt?.Summ}"]));

totalSum += SelectedReceipt != null ? SelectedReceipt.Summ : 0;

}

private static void AddTableHeader(ref Table table)

{

TableRowGroup headerGroup = new();

TableRow headerRow = new();

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("ТМЦ"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Количество"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Единица измерения"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("цена"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Сумма"))) { FontWeight = FontWeights.Bold });

headerGroup.Rows.Add(headerRow);

table.RowGroups.Add(headerGroup);

}

private void AddHeader(ref FlowDocument doc)

{

Paragraph title = new(new Run($"Приходная накладная № {SelectedReceipt?.Id}\n" +

$"От {SelectedReceipt?.Date.ToString("d MMMM yyyy 'г.'", new CultureInfo("ru-RU"))}"))

{

FontSize = 14,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(title);

string headerText = $"\nПоставщик {SelectedReceipt?.Counterpartie?.Name}" +

$"\nСклад получатель {SelectedReceipt?.Material?.Location?.Name}, {SelectedReceipt?.Material?.Location?.Employee?.Fio}";

Paragraph headerParagraph = new(new Run(headerText))

{

FontSize = 12,

TextAlignment = TextAlignment.Left,

LineHeight = 1.5 // Можно добавить межстрочный интервал

};

doc.Blocks.Add(headerParagraph);

}

}

}

<Page x:Class="VacTrack.ViewReport.StockBalanceReport"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewReport"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="StockBalanceReport">

<Page.DataContext>

<local:StockBalanceReportViewModel/>

</Page.DataContext>

<Page.InputBindings>

<KeyBinding Key="P" Modifiers="Control" Command="{Binding PrintCommand}" />

</Page.InputBindings>

<Grid>

<DockPanel VerticalAlignment="Top" Height="40">

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RefreshCommand}"

ToolTip="Обновить">

<materialDesign:PackIcon Kind="Refresh" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding PrintCommand}"

ToolTip="Печать отчёта">

<materialDesign:PackIcon Kind="Printer" />

</Button>

<Button Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Style="{StaticResource MaterialDesignIconButton}" Width="30" Height="30" Margin="10,0,0,0"

Command="{Binding RotateDocCommand}"

ToolTip="Смена ориентации страницы">

<materialDesign:PackIcon Kind="FileRotateRightOutline" />

</Button>

<ListBox Style="{StaticResource MaterialDesignFilterChipPrimaryListBox}" Height="46" Margin="20,0,0,0">

<ListBoxItem Content="Группировка" IsSelected="{Binding IsGroupingEnabled}"/>

<ListBoxItem Content="Итого группы" IsSelected="{Binding IsGroupTotalEnabled}" IsEnabled="{Binding IsGroupingEnabled}"/>

<ListBoxItem Content="Общие итоги" IsSelected="{Binding AreOverallTotalsEnabled}"/>

</ListBox>

<ComboBox

ItemsSource="{Binding MaterialLocation}"

Style="{StaticResource MaterialDesignFloatingHintComboBox}"

DisplayMemberPath="Name"

materialDesign:ComboBoxAssist.ShowSelectedItem="True"

SelectedItem="{Binding FilterByLocation}"

Margin="20,0,0,0"

Foreground="{DynamicResource MaterialDesignBody}"

materialDesign:HintAssist.Hint="Фильтр по месту хранения"

materialDesign:TextFieldAssist.HasClearButton="True"/>

<TextBlock DockPanel.Dock="Right" Height="25" Text="{Binding Message}" ToolTip="{Binding Message}" Foreground="{Binding MessageBrush}" Margin="10,0,0,0"/>

</DockPanel>

<FlowDocumentReader Document="{Binding Document}" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" ViewingMode="Scroll" Foreground="{DynamicResource MaterialDesignBody}" Margin="0,40,0,0" BorderThickness="5" BorderBrush="#19737171"/>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Media;

namespace VacTrack.ViewReport

{

/// <summary>

/// Логика взаимодействия для StockBalanceReport.xaml

/// </summary>

public partial class StockBalanceReport : Page, ICachedPage

{

private StockBalanceReportViewModel ThisViewModel => (StockBalanceReportViewModel)DataContext;

public StockBalanceReport() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class StockBalanceReportViewModel : BaseReportViewModel<Material>

{

public ObservableCollection<Location>? MaterialLocation { get; set; }

#region properties

private bool \_IsGroupingEnabled = true;

public bool IsGroupingEnabled

{

get => \_IsGroupingEnabled;

set

{

SetProperty(ref \_IsGroupingEnabled, value);

Refresh(null);

}

}

private bool \_IsGroupTotalEnabled = true;

public bool IsGroupTotalEnabled

{

get => \_IsGroupTotalEnabled;

set

{

SetProperty(ref \_IsGroupTotalEnabled, value);

Refresh(null);

}

}

private bool \_AreOverallTotalsEnabled = true;

public bool AreOverallTotalsEnabled

{

get => \_AreOverallTotalsEnabled;

set

{

SetProperty(ref \_AreOverallTotalsEnabled, value);

Refresh(null);

}

}

private Location? \_FilterByLocation;

public Location? FilterByLocation

{

get => \_FilterByLocation;

set

{

SetProperty(ref \_FilterByLocation, value);

LoadData();

}

}

#endregion

protected override void LoadData()

{

MaterialLocation = new ObservableCollection<Location>([.. Db.Locations]);

DbSet = Db.Set<Material>();

DbSet.Include(e => e.Unit)

.Include(e => e.Location)

.ThenInclude(e => e != null ? e.Employee : null)

.Load();

Items = new ObservableCollection<Material>(DbSet.Local.Where(item =>

FilterByLocation == null || item.Location?.Id == FilterByLocation.Id

).ToList());

}

public override FlowDocument CreateReport()

{

FlowDocument doc = new() { FontFamily = new FontFamily("Times New Roman"), FontSize = 12, PagePadding = new Thickness(50) };

AddHeader(ref doc); // Добавляем заголовок Отчёта

Table table = new() { CellSpacing = 3, BorderBrush = Brushes.Gray, BorderThickness = new Thickness(1) };

AddTableHeader(ref table); // Добавляем заголовок таблицы

TableRowGroup dataGroup = new();

double totalSum = 0.0;

if (IsGroupingEnabled)

CreateGroupedRows(ref dataGroup, ref totalSum);

else

CreateNoGroupedRows(ref dataGroup, ref totalSum);

if (AreOverallTotalsEnabled)

dataGroup.Rows.Add(CreateRow(["Итого", "", "", "", "", $"{totalSum}"]));

table.RowGroups.Add(dataGroup);

doc.Blocks.Add(table);

return doc;

}

private void CreateGroupedRows(ref TableRowGroup dataGroup, ref double totalSum)

{

CreateGroupedRows(

ref dataGroup,

item => item.Location?.Name,

item => item.GetSum,

key => ["", $"{key}", "", "", "", ""],

total => ["Итого", "", "", "", "", $"{total}"],

item => [

$"{item.Name}",

String.Empty,

$"{item.Count}",

$"{item.Unit?.Name}",

$"{item.Price}",

$"{item.GetSum}"

],

ref totalSum,

IsGroupTotalEnabled);

}

private void CreateNoGroupedRows(ref TableRowGroup dataGroup, ref double totalSum)

{

foreach (var item in Items)

{

double summ = item.GetSum;

totalSum += summ;

dataGroup.Rows.Add(CreateRow([

$"{item.Name}",

$"{item.Location?.Name}",

$"{item.Count}",

$"{item.Unit?.Name}",

$"{item.Price}",

$"{summ}"]));

}

}

private static void AddTableHeader(ref Table table)

{

TableRowGroup headerGroup = new();

TableRow headerRow = new();

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Наименование"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Место хранения"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Количество"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Единица измерения"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Стоимость за единицу"))) { FontWeight = FontWeights.Bold });

headerRow.Cells.Add(new TableCell(new Paragraph(new Run("Стоимость"))) { FontWeight = FontWeights.Bold });

headerGroup.Rows.Add(headerRow);

table.RowGroups.Add(headerGroup);

}

private void AddHeader(ref FlowDocument doc)

{

// Добавляем заголовок отчета

Paragraph header = new(new Run("ООО \"ВакТайм\""))

{

FontSize = 16,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(header);

Paragraph title = new(new Run("Отчет по остаткам материалов и комплектующих"))

{

FontSize = 14,

FontWeight = FontWeights.Bold,

TextAlignment = TextAlignment.Center

};

doc.Blocks.Add(title);

string filteredText = FilterByLocation != null ?

$"\nПо месту хранения: {FilterByLocation.Name}, Ответственный: {FilterByLocation.Employee?.Fio}"

: string.Empty;

string headerText = $"Дата формирования: {DateTime.Now:dd.MM.yyyy}" + filteredText;

Paragraph headerParagraph = new(new Run(headerText))

{

FontSize = 12,

TextAlignment = TextAlignment.Left,

LineHeight = 1.5 // Можно добавить межстрочный интервал

};

doc.Blocks.Add(headerParagraph);

}

}

}

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Timers;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media;

namespace VacTrack.ViewTables

{

public abstract class BaseViewModel<T> : INotifyPropertyChanged where T : BaseModel

{

#region interface implemented

public event PropertyChangedEventHandler? PropertyChanged;

protected void OnPropertyChanged([CallerMemberName] string? name = null)

=> PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));

#endregion

#region properties

protected DatabaseContext Db;

protected DbSet<T> DbSet;

private System.Timers.Timer? \_resetTimer;

private ObservableCollection<T> \_Items;

public ObservableCollection<T> Items

{

get => \_Items;

set

{

if (\_Items != value)

{

\_Items = value;

OnPropertyChanged(nameof(Items));

}

}

}

private T? \_SelectedItem;

public T? SelectedItem

{

get => \_SelectedItem;

set

{

if (!EqualityComparer<T?>.Default.Equals(\_SelectedItem, value)) // Проверка на изменение значения

{

\_SelectedItem = value;

OnPropertyChanged(nameof(SelectedItem));

}

}

}

private string? \_Message;

public string? Message

{

get => \_Message;

set

{

\_Message = value;

OnPropertyChanged();

StartResetTimer();

}

}

private Brush? \_MessageBrush;

public Brush? MessageBrush

{

get => \_MessageBrush;

set

{

\_MessageBrush = value;

OnPropertyChanged();

}

}

private string? \_SearchText;

public string? SearchText

{

get => \_SearchText;

set

{

\_SearchText = value;

OnPropertyChanged();

Search();

}

}

private string? \_TableName;

public string? TableName

{

get => \_TableName;

set

{

\_TableName = value;

OnPropertyChanged();

}

}

public bool? IsAllSelected

{

get

{

var selected = Items.Select(item => item.IsSelected).Distinct().ToList();

return selected.Count == 1 ? selected.Single() : null;

}

set

{

if (value.HasValue)

{

SelectAll(value.Value, Items);

OnPropertyChanged();

}

}

}

#endregion

public ICommand AddCommand { get; }

public ICommand DeleteCommand { get; }

public ICommand SaveCommand { get; }

public ICommand CancelCommand { get; }

public ICommand MessageToClipboardCommand { get; }

public BaseViewModel(DatabaseContext db)

{

Db = db;

Db.Database.EnsureCreated();

LoadData();

if (DbSet == null || \_Items == null) throw new Exception("Data loading error");

AddCommand = new RelayCommand(AddItem);

DeleteCommand = new RelayCommand(DeleteItem);

SaveCommand = new RelayCommand(SaveChanges);

CancelCommand = new RelayCommand(CancelChanges);

MessageToClipboardCommand = new RelayCommand(MessageToClipboard);

foreach (var model in Items)

{

model.PropertyChanged += (sender, args) =>

{

if (args.PropertyName == nameof(Employee.IsSelected))

OnPropertyChanged(nameof(IsAllSelected));

};

}

}

protected abstract T CreateNewItem();

protected abstract bool FilterItem(T item, string? searchText);

protected virtual void LoadData()

{

DbSet = Db.Set<T>();

DbSet.Load();

Items = DbSet.Local.ToObservableCollection();

}

private void AddItem(object obj)

{

try

{

var newItem = CreateNewItem();

Items.Add(newItem); //DbSet.Add(newItem);

Message = "Новая запись добавлена";

MessageBrush = Brushes.Green;

}

catch (Exception ex)

{

Message = $"Ошибка при добавлении: {ex.Message}";

MessageBrush = Brushes.Red;

}

}

private void DeleteItem(object obj)

{

try

{

if (SelectedItem != null)

{

Items.Remove(SelectedItem); //DbSet.Remove(SelectedItem);

Message = "Запись удалена";

MessageBrush = Brushes.Green;

}

else

{

Message = "Выберите строку для удаления.";

MessageBrush = Brushes.Orange;

}

}

catch (Exception ex)

{

System.Windows.MessageBox.Show($"Ошибка при удалении: {ex.Message}");

MessageBrush = Brushes.Red;

}

}

private void SaveChanges(object obj)

{

try

{

// Получаем все измененные, добавленные или удаленные записи для таблицы T

var changedEntries = Db.ChangeTracker.Entries()

.Where(e => e.State == EntityState.Modified || e.State == EntityState.Added || e.State == EntityState.Deleted)

.ToList();

// Если изменения для таблицы T существуют, сохраняем их

if (changedEntries.Count != 0)

{

Db.SaveChanges(); // Сохраняем только изменения для текущего контекста

Message = $"Изменения для таблицы \"{TableName}\" успешно сохранены.";

MessageBrush = Brushes.Green; // Устанавливаем зеленый цвет для сообщения об успехе

}

else

{

Message = $"Нет изменений для сохранения в таблице \"{TableName}\".";

MessageBrush = Brushes.Orange; // Устанавливаем оранжевый цвет для сообщения, если изменений нет

}

}

catch (DbUpdateException dbEx)

{

// Обрабатываем исключения, связанные с обновлением базы данных

Message = $"Ошибка обновления базы данных: {dbEx.Message}";

MessageBrush = Brushes.Red; // Устанавливаем красный цвет для сообщения об ошибке

}

catch (Exception ex)

{

// Обрабатываем все остальные исключения

Message = $"Ошибка сохранения изменений: {ex.Message}";

MessageBrush = Brushes.Red; // Устанавливаем красный цвет для сообщения об ошибке

}

}

private void CancelChanges(object obj)

{

try

{

var changedEntries = Db.ChangeTracker.Entries()

.Where(e => e.State == EntityState.Modified || e.State == EntityState.Added || e.State == EntityState.Deleted);

if (changedEntries.ToList().Count == 0)

{

Message = "Нечего отменять";

MessageBrush = Brushes.Orange;

return;

}

Db.ChangeTracker.Clear();

LoadData(); // Перезагрузка данных, чтобы обновить коллекцию

Message = "Изменения отменены.";

MessageBrush = Brushes.Green;

}

catch (Exception ex)

{

Message = $"Ошибка при отмене изменений: {ex.Message}";

MessageBrush = Brushes.Red;

}

}

private void Search()

{

try

{

if (string.IsNullOrWhiteSpace(SearchText))

{

LoadData(); // Перезагрузка данных, если строка поиска пустая

Message = string.Empty;

return;

}

var filteredItems = DbSet.Local.Where(item => FilterItem(item, SearchText)).ToList();

Items = new ObservableCollection<T>(filteredItems);

Message = $"Найдено записей: {filteredItems.Count}";

MessageBrush = filteredItems.Count != 0 ? Brushes.Green : Brushes.Orange;

}

catch (Exception ex)

{

Message = $"Ошибка поиска: {ex.Message}";

MessageBrush = Brushes.Red;

}

}

private void StartResetTimer()

{

// Останавливаем предыдущий таймер, если он существует

\_resetTimer?.Stop();

// Создаем новый таймер, который сработает через 10 секунд

\_resetTimer = new System.Timers.Timer(10000); // 10 секунд

\_resetTimer.Elapsed += ResetMessage;

\_resetTimer.Start();

}

private void ResetMessage(object? sender, ElapsedEventArgs e)

{

App.Current.Dispatcher.Invoke(() =>

{

Message = string.Empty;

});

\_resetTimer?.Stop();

}

public void OpenFromCache() => LoadData();

private static void SelectAll(bool select, IEnumerable<T> models)

{

foreach (var model in models)

{

model.IsSelected = select;

}

}

private void MessageToClipboard(object obj) => Clipboard.SetText(Message);

}

} <Page x:Class="VacTrack.ViewTables.ContractViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:sys="clr-namespace:System;assembly=System.Runtime"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ContractViewTable">

<Page.DataContext>

<local:ContractViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter" />

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid Margin="0,20,0,35"

ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Название">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Name">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridComboBoxColumn

Header="Контрагент"

IsEditable="True"

ItemsSourceBinding="{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}, Path=DataContext.ContractCounterpartie, UpdateSourceTrigger=PropertyChanged}"

SelectedValueBinding="{Binding Counterpartie}"

DisplayMemberPath="Name">

</materialDesign:DataGridComboBoxColumn>

<DataGridTemplateColumn Header="Дата заключения">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Date, StringFormat={}{0:dd.MM.yyyy}}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<DatePicker Width="90"

SelectedDate="{Binding Date, UpdateSourceTrigger=PropertyChanged}"

Style="{StaticResource MaterialDesignFloatingHintDatePicker}"

materialDesign:HintAssist.Hint="Выбор даты"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Cумма">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Summ">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridComboBoxColumn

Header="Изделие"

IsEditable="True"

ItemsSourceBinding="{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}, Path=DataContext.ContractProduct, UpdateSourceTrigger=PropertyChanged}"

SelectedValueBinding="{Binding Product}"

DisplayMemberPath="Name">

</materialDesign:DataGridComboBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Количество">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Count">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows.Controls;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для ContractViewTable.xaml

/// </summary>

public partial class ContractViewTable : Page, ICachedPage

{

private ContractViewModel ThisViewModel => (ContractViewModel)DataContext;

public ContractViewTable() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class ContractViewModel : BaseViewModel<Contract>

{

public ObservableCollection<Counterpartie>? ContractCounterpartie { get; set; }

public ObservableCollection<Product>? ContractProduct { get; set; }

public ContractViewModel() : base(new DatabaseContext()) { }

protected override void LoadData()

{

TableName = "Договора";

ContractCounterpartie = new ObservableCollection<Counterpartie>([.. Db.Counterparties]);

ContractProduct = new ObservableCollection<Product>([.. Db.Products]);

DbSet = Db.Set<Contract>();

DbSet.Include(e => e.Counterpartie).Include(e => e.Product).Load();

Items = DbSet.Local.ToObservableCollection();

}

protected override Contract CreateNewItem() => new() { Name="Новый договор", Date = DateTime.Now };

protected override bool FilterItem(Contract item, string? searchText) =>

string.IsNullOrWhiteSpace(searchText) ||

item.Counterpartie?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Product?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Summ.ToString().Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true;

}

}

<Page x:Class="VacTrack.ViewTables.CounterpartieViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="1080"

Title="CounterpartieViewTable">

<Page.DataContext>

<local:CounterpartieViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter"/>

<converter:CounterpartieTypeConverter x:Key="CounterpartieTypeConverter"/>

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid Margin="0,20,0,35"

ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="ФИО/Наименование">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Name">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Юридический адрес">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="LegalAddress">

<Binding.ValidationRules>

<validators:LengthRangeValidationRule MinLength="5" MaxLength="128"/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Номер телефона">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="PhoneNomber">

<Binding.ValidationRules>

<validators:PhoneNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Почтовый адресс">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="PostalAddress">

<Binding.ValidationRules>

<validators:LengthRangeValidationRule MinLength="5" MaxLength="128"/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="УНП">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Unp">

<Binding.ValidationRules>

<validators:LengthRangeValidationRule Length="9"/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<DataGridTemplateColumn Header="Расчетный счет" Width="130">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding BankAccount}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<TextBox Text="{Binding BankAccount}" AcceptsReturn="True" VerticalScrollBarVisibility="Auto"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<materialDesign:DataGridComboBoxColumn

Header="Тип"

ItemsSourceBinding="{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}, Path=DataContext.Types, Converter={StaticResource CounterpartieTypeConverter}}"

SelectedValueBinding="{Binding Type, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource CounterpartieTypeConverter}}"

IsEditable="False"/>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="ОКЮЛП">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Okulp">

<Binding.ValidationRules>

<validators:LengthRangeValidationRule Length="10"/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="ОКПО">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Okpo">

<Binding.ValidationRules>

<validators:LengthRangeValidationRule Length="10"/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="ОКЕД">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Oked">

<Binding.ValidationRules>

<validators:LengthRangeValidationRule Length="5"/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page> using DatabaseManager;

using System.Windows.Controls;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для CounterpartieViewTable.xaml

/// </summary>

public partial class CounterpartieViewTable : Page, ICachedPage

{

private CounterpartieViewModel ThisViewModel => (CounterpartieViewModel)DataContext;

public CounterpartieViewTable() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class CounterpartieViewModel : BaseViewModel<Counterpartie>

{

public List<CounterpartieType> Types { get; } = [CounterpartieType.Fiz, CounterpartieType.Ur];

public CounterpartieViewModel() : base(new DatabaseContext()) { TableName = "Контрагенты"; }

protected override Counterpartie CreateNewItem() => new() {

Name = "Новый контрагент",

LegalAddress = "Юридический адрес",

PhoneNomber = "Номер телефона",

PostalAddress = "Почтовый адрес",

Unp = "УНП",

BankAccount = "Счёт"

};

protected override bool FilterItem(Counterpartie item, string? searchText) =>

string.IsNullOrWhiteSpace(searchText) ||

item.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true;

}

}

<Page x:Class="VacTrack.ViewTables.DivisionViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="DivisionViewTable">

<Page.DataContext>

<local:DivisionViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter" />

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

Margin="0,20,0,35"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}" Header="Наименование">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Name">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page>

using DatabaseManager;

using System.Windows.Controls;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для DivisionViewTable.xaml

/// </summary>

public partial class DivisionViewTable : Page

{

public DivisionViewTable()

{

InitializeComponent();

}

}

public class DivisionViewModel : BaseViewModel<Division>

{

public DivisionViewModel() : base(new DatabaseContext()) { TableName = "Подразделения"; }

protected override Division CreateNewItem() => new() { Name = "Новое подразделение" };

protected override bool FilterItem(Division item, string filter) =>

item.Name != null && item.Name.Contains(filter, StringComparison.CurrentCultureIgnoreCase);

}

}

<Page x:Class="VacTrack.ViewTables.EmployeeViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:sys="clr-namespace:System;assembly=System.Runtime"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="EmployeeViewTable">

<Page.DataContext>

<local:EmployeeViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter" />

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid Margin="0,20,0,35"

ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}" Header="ФИО">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Fio">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<DataGridTemplateColumn Header="Подразделения">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Division.Name, UpdateSourceTrigger=PropertyChanged}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<ComboBox

ItemsSource="{Binding DataContext.EmployDivisions, RelativeSource={RelativeSource AncestorType=Page}}"

DisplayMemberPath="Name"

SelectedValue="{Binding Division, UpdateSourceTrigger=PropertyChanged}"

IsEditable="True"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Должность">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Post.Name, UpdateSourceTrigger=PropertyChanged}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<ComboBox

ItemsSource="{Binding DataContext.EmployPosts, RelativeSource={RelativeSource AncestorType=Page}}"

DisplayMemberPath="Name"

SelectedValue="{Binding Post, UpdateSourceTrigger=PropertyChanged}"

IsEditable="True"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Дата найма">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding DateHire, StringFormat={}{0:dd.MM.yyyy}}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<DatePicker Width="90"

SelectedDate="{Binding DateHire, UpdateSourceTrigger=PropertyChanged}"

Style="{StaticResource MaterialDesignFloatingHintDatePicker}"

materialDesign:HintAssist.Hint="Выбор даты"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Дата увольнения">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding DateDismissal, StringFormat={}{0:dd.MM.yyyy}}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<DatePicker Width="90"

SelectedDate="{Binding DateDismissal, UpdateSourceTrigger=PropertyChanged, FallbackValue={x:Static sys:DateTime.Now}}"

Style="{StaticResource MaterialDesignFloatingHintDatePicker}"

materialDesign:HintAssist.Hint="Выбор даты"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows.Controls;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для EmployeeViewTable.xaml

/// </summary>

public partial class EmployeeViewTable : Page, ICachedPage

{

private EmployeeViewModel ThisViewModel => (EmployeeViewModel)DataContext;

public EmployeeViewTable() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class EmployeeViewModel : BaseViewModel<Employee>

{

public ObservableCollection<Division>? EmployDivisions { get; set; }

public ObservableCollection<Post>? EmployPosts { get; set; }

public EmployeeViewModel() : base(new DatabaseContext()) { }

protected override void LoadData()

{

TableName = "Сотрудники";

EmployDivisions = new ObservableCollection<Division>([.. Db.Divisions]);

EmployPosts = new ObservableCollection<Post>([.. Db.Posts]);

DbSet = Db.Set<Employee>();

DbSet.Include(e => e.Division).Include(e => e.Post).Load();

Items = DbSet.Local.ToObservableCollection();

}

protected override Employee CreateNewItem() => new() { Fio = "Новый сотрудник", DateHire = DateTime.Now };

protected override bool FilterItem(Employee item, string? searchText) =>

string.IsNullOrWhiteSpace(searchText) ||

item.Fio?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Division?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Post?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true;

}

}

<Page x:Class="VacTrack.ViewTables.LocationViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="LocationViewTable">

<Page.DataContext>

<local:LocationViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter" />

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid Margin="0,20,0,35"

ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}" Header="Наименование">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Name">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridComboBoxColumn

Header="Кладовщик"

IsEditable="True"

ItemsSourceBinding="{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}, Path=DataContext.LocalEmployee, UpdateSourceTrigger=PropertyChanged}"

SelectedValueBinding="{Binding Employee}"

DisplayMemberPath="Fio">

</materialDesign:DataGridComboBoxColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows.Controls;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для LocationViewTable.xaml

/// </summary>

public partial class LocationViewTable : Page, ICachedPage

{

private LocationViewModel ThisViewModel => (LocationViewModel)DataContext;

public LocationViewTable() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class LocationViewModel : BaseViewModel<Location>

{

public ObservableCollection<Employee>? LocalEmployee { get; set; }

public LocationViewModel() : base(new DatabaseContext()) { }

protected override void LoadData()

{

TableName = "Места хранения";

LocalEmployee = new ObservableCollection<Employee>([.. Db.Employees]);

DbSet = Db.Set<Location>();

DbSet.Include(e => e.Employee).Load();

Items = DbSet.Local.ToObservableCollection();

}

protected override Location CreateNewItem() => new() { Name = "Новый склад" };

protected override bool FilterItem(Location item, string? searchText) =>

string.IsNullOrWhiteSpace(searchText) ||

item.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Employee?.Fio?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true;

}

}

<Page x:Class="VacTrack.ViewTables.MaterialViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="850"

Title="MaterialViewTable">

<Page.DataContext>

<local:MaterialViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter"/>

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid Margin="0,20,0,35"

ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Наименование">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Name">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<DataGridTemplateColumn Header="Единица измерения">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Unit.Name, UpdateSourceTrigger=PropertyChanged}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<ComboBox

ItemsSource="{Binding DataContext.MaterialUnit, RelativeSource={RelativeSource AncestorType=Page}}"

DisplayMemberPath="Name"

SelectedValue="{Binding Unit, UpdateSourceTrigger=PropertyChanged}"

IsEditable="True"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Место хранения">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Location.Name, UpdateSourceTrigger=PropertyChanged}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<ComboBox

ItemsSource="{Binding DataContext.MaterialLocation, RelativeSource={RelativeSource AncestorType=Page}}"

DisplayMemberPath="Name"

SelectedValue="{Binding Location, UpdateSourceTrigger=PropertyChanged}"

IsEditable="True"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<materialDesign:DataGridTextColumn Header="Количество">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Count">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn Header="Цена за единицу">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Price">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn Header="Стоимость" IsReadOnly="True">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="GetSum">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows.Controls;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для MaterialViewTable.xaml

/// </summary>

public partial class MaterialViewTable : Page, ICachedPage

{

private MaterialViewModel ThisViewModel => (MaterialViewModel)DataContext;

public MaterialViewTable() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class MaterialViewModel : BaseViewModel<Material>

{

public ObservableCollection<Unit>? MaterialUnit { get; set; }

public ObservableCollection<Location>? MaterialLocation { get; set; }

public MaterialViewModel() : base(new DatabaseContext()) { }

protected override void LoadData()

{

TableName = "Материалы\\Комплектующие";

MaterialUnit = new ObservableCollection<Unit>([.. Db.Units]);

MaterialLocation = new ObservableCollection<Location>([.. Db.Locations]);

DbSet = Db.Set<Material>();

DbSet.Include(e => e.Unit).Include(e => e.Location).Load();

Items = DbSet.Local.ToObservableCollection();

}

protected override Material CreateNewItem() => new() { Name = "Новый материал" };

protected override bool FilterItem(Material item, string? searchText) =>

string.IsNullOrWhiteSpace(searchText) ||

item.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Unit?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Location?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true;

}

}

<Page x:Class="VacTrack.ViewTables.PostViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="PostViewTable">

<Page.DataContext>

<local:PostViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter" />

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

Margin="0,20,0,35"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}" Header="Наименование">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Name">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page>

using DatabaseManager;

using System.Windows.Controls;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для Post.xaml

/// </summary>

public partial class PostViewTable : Page

{

public PostViewTable()

{

InitializeComponent();

}

}

public class PostViewModel : BaseViewModel<Post>

{

public PostViewModel() : base(new DatabaseContext()) { TableName = "Должности"; }

protected override Post CreateNewItem() => new() { Name = "Новая должность" };

protected override bool FilterItem(Post item, string filter) =>

item.Name != null && item.Name.Contains(filter, StringComparison.CurrentCultureIgnoreCase);

}

} <Page x:Class="VacTrack.ViewTables.ProductViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="850"

Title="ProductViewTable">

<Page.DataContext>

<local:ProductViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter"/>

</Page.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="12\*"/>

<ColumnDefinition Width="7\*"/>

</Grid.ColumnDefinitions>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center" Grid.ColumnSpan="2"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" Grid.ColumnSpan="2" />

<DataGrid Margin="0,20,0,35"

ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Наименование">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Name">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Серийный номер">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="SerialNo">

<Binding.ValidationRules>

<validators:LengthRangeValidationRule MinLength="5" MaxLength="10"/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<DataGridTemplateColumn Header="Единица измерения">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Unit.Name, UpdateSourceTrigger=PropertyChanged}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<ComboBox

ItemsSource="{Binding DataContext.ProductUnit, RelativeSource={RelativeSource AncestorType=Page}}"

DisplayMemberPath="Name"

SelectedValue="{Binding Unit, UpdateSourceTrigger=PropertyChanged}"

IsEditable="True"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Место хранения">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Location.Name, UpdateSourceTrigger=PropertyChanged}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<ComboBox

ItemsSource="{Binding DataContext.ProductLocation, RelativeSource={RelativeSource AncestorType=Page}}"

DisplayMemberPath="Name"

SelectedValue="{Binding Location, UpdateSourceTrigger=PropertyChanged}"

IsEditable="True"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Количество">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Count">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<Label Content="Материалы и Комлектуюшие для" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center" Grid.Column="1"/>

<Label Content="{Binding SelectedItem.Name}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center" Grid.Column="1" Margin="0,15,0,0"/>

<DataGrid Grid.Column="1"

Margin="5,40,0,35"

SelectedItem="{Binding SelectedMaterial}"

ItemsSource="{Binding ProdMater}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<materialDesign:DataGridComboBoxColumn

Header="Наименование"

IsEditable="True"

ItemsSourceBinding="{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}, Path=DataContext.ProductMaterials, UpdateSourceTrigger=PropertyChanged}"

SelectedValueBinding="{Binding Material}"

DisplayMemberPath="Name">

</materialDesign:DataGridComboBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Количество">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Quantity">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<materialDesign:PopupBox

ToolTip="Добавить или удалить материал"

Grid.Column="1"

VerticalAlignment="Top"

HorizontalAlignment="Right"

Margin="0,46,7,0" Width="30" Height="30"

PlacementMode="BottomAndAlignCentres"

Style="{StaticResource MaterialDesignMultiFloatingActionPopupBox}"

Background="Transparent">

<materialDesign:PopupBox.ToggleContent>

<materialDesign:PackIcon Kind="Pencil" Foreground="{DynamicResource MaterialDesignBody}"/>

</materialDesign:PopupBox.ToggleContent>

<StackPanel>

<Button Style="{StaticResource MaterialDesignIconButton}"

ToolTip="Добавить материал к изделию"

Command="{Binding AddMaterialCommand}"

Margin="0,10,0,0" Width="30" Height="30"

Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Foreground="{DynamicResource MaterialDesignBody}">

<materialDesign:PackIcon Kind="Plus" Width="25" Height="25"/>

</Button>

<Button Style="{StaticResource MaterialDesignIconButton}"

ToolTip="Удалить выбраный материал"

Command="{Binding DeleteMaterialCommand}"

Margin="0,10,0,0" Width="30" Height="30"

Background="{DynamicResource MaterialDesignTextFieldBoxBackground}"

Foreground="{DynamicResource MaterialDesignBody}">

<materialDesign:PackIcon Kind="DeleteForeverOutline" Width="25" Height="25"/>

</Button>

</StackPanel>

</materialDesign:PopupBox>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel" Grid.ColumnSpan="2">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page> using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для ProductViewTable.xaml

/// </summary>

public partial class ProductViewTable : Page, ICachedPage

{

private ProductViewModel ThisViewModel => (ProductViewModel)DataContext;

public ProductViewTable() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class ProductViewModel : BaseViewModel<Product>

{

public ObservableCollection<Unit>? ProductUnit { get; set; }

public ObservableCollection<Location>? ProductLocation { get; set; }

public ObservableCollection<Material>? ProductMaterials { get; set; }

public ObservableCollection<Product\_Material>? ProdMater { get; set; }

private Product\_Material? \_SelectedMaterial;

public Product\_Material? SelectedMaterial

{

get => \_SelectedMaterial;

set

{

if (!EqualityComparer<Product\_Material?>.Default.Equals(\_SelectedMaterial, value))

{

\_SelectedMaterial = value;

OnPropertyChanged(nameof(SelectedItem));

}

}

}

private Product? LastSelectedMaterial;

public ICommand AddMaterialCommand { get; }

public ICommand DeleteMaterialCommand { get; }

public ProductViewModel() : base(new DatabaseContext())

{

AddMaterialCommand = new RelayCommand(AddMaterial);

DeleteMaterialCommand = new RelayCommand(DeleteMaterial);

PropertyChanged += OnPropertyChangedHandler;

}

private void OnPropertyChangedHandler(object? sender, PropertyChangedEventArgs e)

{

if (e.PropertyName == nameof(SelectedItem) && sender is ProductViewModel pvm)

LoadProdMater(pvm.SelectedItem);

}

private void LoadProdMater(Product? product)

{

if (product == null)

ProdMater?.Clear();

else

{

if (LastSelectedMaterial == product) return;

// Выполняем один запрос к базе данных с учетом всех связанных данных

var productMaterials = Db.Product\_Materials

.Include(e => e.Material) // Подгружаем связанные материалы

.Where(e => e.ProductId == product.Id) // Фильтруем по нужному продукту

.ToList(); // Выполняем запрос и загружаем данные в память

// Создаем ObservableCollection на основе загруженных данных

ProdMater = new ObservableCollection<Product\_Material>(productMaterials);

LastSelectedMaterial = product;

// Уведомляем об изменении свойства

OnPropertyChanged(nameof(ProdMater));

}

}

private void AddMaterial(object obj)

{

try

{

if (ProdMater != null || ProdMater?.Count == 0)

{

if (SelectedItem == null) throw new Exception();

var newPM = new Product\_Material() { ProductId = SelectedItem.Id, Product = SelectedItem };

ProdMater.Add(newPM);

Db.Product\_Materials.Add(newPM);

Message = "Материал добавлен";

MessageBrush = Brushes.Green;

}

else

{

Message = "Выберите изделие";

MessageBrush = Brushes.Orange;

}

}

catch (Exception ex)

{

Message = $"Ошибка при добавлении материала: {ex.Message}";

MessageBrush = Brushes.Red;

}

}

private void DeleteMaterial(object obj)

{

try

{

if (SelectedMaterial != null && ProdMater != null)

{

Db.Product\_Materials.Remove(SelectedMaterial);

ProdMater.Remove(SelectedMaterial);

Message = "Материал удален";

MessageBrush = Brushes.Green;

}

else

{

Message = "Выберите материал для удаления.";

MessageBrush = Brushes.Orange;

}

}

catch (Exception ex)

{

System.Windows.MessageBox.Show($"Ошибка при удалении материала: {ex.Message}");

MessageBrush = Brushes.Red;

}

}

protected override void LoadData()

{

TableName = "Изделия";

ProductUnit = new ObservableCollection<Unit>([.. Db.Units]);

ProductLocation = new ObservableCollection<Location>([.. Db.Locations]);

ProductMaterials = new ObservableCollection<Material>([.. Db.Materials]);

DbSet = Db.Set<Product>();

DbSet.Include(e => e.Unit).Include(e => e.Location).Load();

Items = DbSet.Local.ToObservableCollection();

}

protected override Product CreateNewItem() => new() { Name = "Новое изделие", SerialNo = "Серийный номер" };

protected override bool FilterItem(Product item, string? searchText) =>

string.IsNullOrWhiteSpace(searchText) ||

item.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Unit?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Location?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true;

}

}

<Page x:Class="VacTrack.ViewTables.ReceiptViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:sys="clr-namespace:System;assembly=System.Runtime"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ReceiptViewTable">

<Page.DataContext>

<local:ReceiptViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter" Length="715"/>

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid Margin="0,20,0,35"

ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridComboBoxColumn

Header="Контрагент"

IsEditable="True"

ItemsSourceBinding="{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}, Path=DataContext.ReceiptCounterpartie, UpdateSourceTrigger=PropertyChanged}"

SelectedValueBinding="{Binding Counterpartie}"

DisplayMemberPath="Name">

</materialDesign:DataGridComboBoxColumn>

<DataGridTemplateColumn Header="Дата поступления">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Date, StringFormat={}{0:dd.MM.yyyy}}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<DatePicker Width="90"

SelectedDate="{Binding Date, UpdateSourceTrigger=PropertyChanged}"

Style="{StaticResource MaterialDesignFloatingHintDatePicker}"

materialDesign:HintAssist.Hint="Выбор даты"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<materialDesign:DataGridComboBoxColumn

Header="Материал"

IsEditable="True"

ItemsSourceBinding="{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}, Path=DataContext.ReceiptMaterial, UpdateSourceTrigger=PropertyChanged}"

SelectedValueBinding="{Binding Material}"

DisplayMemberPath="Name">

</materialDesign:DataGridComboBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Количество">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Count">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<DataGridTextColumn IsReadOnly="True" Header="Ед.Изм." Binding="{Binding Material.Unit.Name, UpdateSourceTrigger=PropertyChanged}"/>

<DataGridTextColumn IsReadOnly="True" Header="Цена" Binding="{Binding Material.Price, UpdateSourceTrigger=PropertyChanged}"/>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Cумма">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Summ" UpdateSourceTrigger="PropertyChanged">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

<Button Content="Печать" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding PrintCommand}" Width="95" ToolTip="Печать выбраного документа"/>

</DockPanel>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для ReceiptViewTable.xaml

/// </summary>

public partial class ReceiptViewTable : Page, ICachedPage

{

private ReceiptViewModel ThisViewModel => (ReceiptViewModel)DataContext;

public ReceiptViewTable() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class ReceiptViewModel : BaseViewModel<Receipt>

{

public ObservableCollection<Counterpartie>? ReceiptCounterpartie { get; set; }

public ObservableCollection<Material>? ReceiptMaterial { get; set; }

public ICommand PrintCommand { get; }

public ReceiptViewModel() : base(new DatabaseContext())

{

PrintCommand = new RelayCommand(PrintDoc);

}

protected override void LoadData()

{

TableName = "Поступления";

ReceiptCounterpartie = new ObservableCollection<Counterpartie>([.. Db.Counterparties]);

var MaterDbSet = Db.Set<Material>();

MaterDbSet.Include(m => m.Unit).Load();

ReceiptMaterial = MaterDbSet.Local.ToObservableCollection();

DbSet = Db.Set<Receipt>();

DbSet.Include(e => e.Counterpartie)

.Include(e => e.Material)

.ThenInclude(e => e != null ? e.Unit : null)

.Include(e => e.Material != null ? e.Material.Location : null)

.ThenInclude(e => e != null ? e.Employee : null)

.Load();

Items = DbSet.Local.ToObservableCollection();

}

protected override Receipt CreateNewItem() => new() { Date = DateTime.Now };

protected override bool FilterItem(Receipt item, string? searchText) =>

string.IsNullOrWhiteSpace(searchText) ||

item.Counterpartie?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Material?.Name?.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Summ.ToString().Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true;

private void PrintDoc(object obj)

{

if (SelectedItem != null)

new ViewReport.ReceiptReport(SelectedItem).ShowDialog();

else

{

Message = "Выберете поступление";

MessageBrush = Brushes.Orange;

}

}

}

}

<Page x:Class="VacTrack.ViewTables.SaleViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:sys="clr-namespace:System;assembly=System.Runtime"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="SaleViewTable">

<Page.DataContext>

<local:SaleViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter" />

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid Margin="0,20,0,35"

ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn Header="Изделие" IsReadOnly="True" Binding="{Binding Contract.Product.Name, UpdateSourceTrigger=PropertyChanged}"/>

<materialDesign:DataGridComboBoxColumn

Header="Договор"

IsEditable="True"

ItemsSourceBinding="{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}, Path=DataContext.SaleContract, UpdateSourceTrigger=PropertyChanged}"

SelectedValueBinding="{Binding Contract}"

DisplayMemberPath="Name">

</materialDesign:DataGridComboBoxColumn>

<DataGridTemplateColumn Header="Дата">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Date, StringFormat={}{0:dd.MM.yyyy}}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<DatePicker Width="90"

SelectedDate="{Binding Date, UpdateSourceTrigger=PropertyChanged}"

Style="{StaticResource MaterialDesignFloatingHintDatePicker}"

materialDesign:HintAssist.Hint="Выбор даты"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Cумма">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Summ">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}"

Header="Количество">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Count">

<Binding.ValidationRules>

<validators:PositiveNumberValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page>

using DatabaseManager;

using Microsoft.EntityFrameworkCore;

using System.Collections.ObjectModel;

using System.Windows.Controls;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для SaleViewTable.xaml

/// </summary>

public partial class SaleViewTable : Page, ICachedPage

{

private SaleViewModel ThisViewModel => (SaleViewModel)DataContext;

public SaleViewTable() => InitializeComponent();

public void OnNavigatedFromCache() => ThisViewModel.OpenFromCache();

}

public class SaleViewModel : BaseViewModel<Sale>

{

public ObservableCollection<Contract>? SaleContract { get; set; }

public SaleViewModel() : base(new DatabaseContext()) { }

protected override void LoadData()

{

TableName = "Реализация";

SaleContract = new ObservableCollection<Contract>([.. Db.Contracts]);

DbSet = Db.Set<Sale>();

DbSet.Include(e => e.Contract).ThenInclude(c => c != null ? c.Product : null).Load();

Items = DbSet.Local.ToObservableCollection();

}

protected override Sale CreateNewItem() => new() { Date = DateTime.Now };

protected override bool FilterItem(Sale item, string? searchText) =>

string.IsNullOrWhiteSpace(searchText) ||

item.Contract?.Product?.Name.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Contract?.Name.Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true ||

item.Summ.ToString().Contains(searchText, StringComparison.CurrentCultureIgnoreCase) == true;

}

}

<Page x:Class="VacTrack.ViewTables.UnitViewTable"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack.ViewTables"

xmlns:validators="clr-namespace:VacTrack.Validators"

xmlns:converter="clr-namespace:VacTrack.Converter"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="UnitViewTable">

<Page.DataContext>

<local:UnitViewModel/>

</Page.DataContext>

<Page.Resources>

<converter:DockPanelWidthConverter x:Key="DockPanelWidthConverter" />

</Page.Resources>

<Grid>

<Label Content="{Binding TableName}" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" HorizontalAlignment="Center"/>

<FrameworkElement x:Name="DataContextProxy" DataContext="{Binding}" />

<DataGrid ItemsSource="{Binding Items}"

SelectedItem="{Binding SelectedItem}"

AutoGenerateColumns="False"

CanUserAddRows="False"

HeadersVisibility="All"

Margin="0,20,0,35"

BorderThickness="3"

BorderBrush="#19737171">

<DataGrid.Columns>

<DataGridCheckBoxColumn Binding="{Binding IsSelected, UpdateSourceTrigger=PropertyChanged}"

EditingElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnEditingStyle}"

ElementStyle="{StaticResource MaterialDesignDataGridCheckBoxColumnStyle}">

<DataGridCheckBoxColumn.Header>

<Border Background="Transparent">

<CheckBox IsChecked="{Binding DataContext.IsAllSelected, Source={x:Reference DataContextProxy}}" />

</Border>

</DataGridCheckBoxColumn.Header>

<DataGridCheckBoxColumn.HeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}" BasedOn="{StaticResource MaterialDesignDataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</DataGridCheckBoxColumn.HeaderStyle>

</DataGridCheckBoxColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}" Header="Наименование">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Name">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

<materialDesign:DataGridTextColumn EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnPopupEditingStyle}" ElementStyle="{StaticResource MaterialDesignDataGridTextColumnStyle}" Header="Описание">

<materialDesign:DataGridTextColumn.Binding>

<Binding Path="Description">

<Binding.ValidationRules>

<validators:NotEmptyValidationRule/>

</Binding.ValidationRules>

</Binding>

</materialDesign:DataGridTextColumn.Binding>

</materialDesign:DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel LastChildFill="False" VerticalAlignment="Bottom" Height="35" x:Name="MainDockPanel">

<Button Content="Добавить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding AddCommand}" Width="95" ToolTip="добавление новой записи"/>

<Button Content="Удалить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding DeleteCommand}" Width="95" ToolTip="удаление выбраной записи"/>

<Button Content="Сохранить" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" Command="{Binding SaveCommand}" Width="95" ToolTip="сохраняет внесеные изменения"/>

<TextBox materialDesign:TextFieldAssist.HasClearButton="True"

Style="{StaticResource MaterialDesignFloatingHintTextBox}"

Width="200"

materialDesign:HintAssist.Hint="Поиск"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{Binding SearchText, UpdateSourceTrigger=PropertyChanged}"

Margin="5,0,3,0"/>

<TextBlock HorizontalAlignment="Center"

VerticalAlignment="Center"

ToolTip="{Binding Message}"

Text="{Binding Message}"

Foreground="{Binding MessageBrush}"

TextTrimming="CharacterEllipsis"

Margin="5,0,3,0"

MaxWidth="{Binding ActualWidth, ElementName=MainDockPanel, Converter={StaticResource DockPanelWidthConverter}}"/>

<Button Content="Отмена" Style="{StaticResource MaterialDesignPaperButton}" Height="25" FontSize="12" Margin="0,0,5,0" DockPanel.Dock="Right" Cursor="Hand" Command="{Binding CancelCommand}" Width="95" ToolTip="отменяет не сохраненые изменения"/>

</DockPanel>

</Grid>

</Page>

using DatabaseManager;

using System.Windows.Controls;

namespace VacTrack.ViewTables

{

/// <summary>

/// Логика взаимодействия для Unit.xaml

/// </summary>

public partial class UnitViewTable : Page

{

public UnitViewTable()

{

InitializeComponent();

}

}

public class UnitViewModel : BaseViewModel<Unit>

{

public UnitViewModel() : base(new DatabaseContext()) { TableName = "Единицы измерения"; }

protected override Unit CreateNewItem() => new() { Name = "Новая единица" };

protected override bool FilterItem(Unit item, string filter) =>

item.Name != null && item.Name.Contains(filter, StringComparison.CurrentCultureIgnoreCase);

}

}

<Window x:Class="VacTrack.AboutProgram"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:VacTrack"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

Style="{StaticResource CustomWindowStyle}"

mc:Ignorable="d"

Title="О программе" Height="300" Width="500" ResizeMode="NoResize">

<Window.Resources>

<DrawingImage x:Key="VacTimeLogoDrawingImage">

<DrawingImage.Drawing>

<DrawingGroup ClipGeometry="M0,0 V1171 H1171 V0 H0 Z">

<DrawingGroup Opacity="1">

<DrawingGroup.ClipGeometry>

<RectangleGeometry RadiusX="0" RadiusY="0" Rect="0,0,1171,1171" />

</DrawingGroup.ClipGeometry>

<GeometryDrawing Brush="#66FFFFFF">

<GeometryDrawing.Pen>

<Pen Brush="#FF000000" Thickness="15" StartLineCap="Flat" EndLineCap="Flat" LineJoin="Miter" />

</GeometryDrawing.Pen>

<GeometryDrawing.Geometry>

<EllipseGeometry RadiusX="578" RadiusY="578" Center="585.5,585.5" />

</GeometryDrawing.Geometry>

</GeometryDrawing>

<GeometryDrawing Brush="#FF6EBFAC" Geometry="F0 M1171,1171z M0,0z M406.648,491L342.008,359.494C346.429,351.451,354.982,346,364.808,346L994.24,346C1002.71,346,1010.24,350.055,1014.99,356.331L951.135,491 406.648,491z" />

<GeometryDrawing Brush="#FF6EBFAC">

<GeometryDrawing.Geometry>

<RectangleGeometry RadiusX="33" RadiusY="33" Rect="613,467,147,537" />

</GeometryDrawing.Geometry>

</GeometryDrawing>

<GeometryDrawing Brush="#FF475653">

<GeometryDrawing.Geometry>

<RectangleGeometry RadiusX="21" RadiusY="21" Rect="436,531,127,472" />

</GeometryDrawing.Geometry>

</GeometryDrawing>

<GeometryDrawing Brush="#FF475653" Geometry="F0 M1171,1171z M0,0z M122.646,346C116.225,362.487,116.411,381.495,124.716,398.683L397.327,962.822C412.586,994.398 450.554,1007.63 482.13,992.368 513.707,977.109 526.935,939.141 511.676,907.565L240.309,346 122.646,346z" />

<GeometryDrawing Brush="#FF475653" Geometry="F1 M1171,1171z M0,0z M558.501,911.722C558.501,961.98 551,1002.72 467.001,1002.72 414.5,1002.72 417.411,1001 375.5,916 340,844 416.467,820.722 467.001,820.722 517.535,820.722 558.501,861.464 558.501,911.722z" />

</DrawingGroup>

</DrawingGroup>

</DrawingImage.Drawing>

</DrawingImage>

</Window.Resources>

<Grid Background="{DynamicResource MaterialDesign.Brush.Background}">

<Image Source="{StaticResource VacTimeLogoDrawingImage}" HorizontalAlignment="Left" Width="160" Stretch="UniformToFill"/>

<TextBlock Text="ПС «Учет выпуска и реализации продукции ООО«ВакТайм»" TextWrapping="WrapWithOverflow" Margin="165,5,5,0" Foreground="{DynamicResource MaterialDesignBody}" VerticalAlignment="Top" FontSize="14" Width="230" HorizontalAlignment="Left"/>

<Label Content="Разработал: Грачёв Денис Сергеевич" Margin="165,45,5,0" VerticalAlignment="Top"/>

<ScrollViewer Foreground="{DynamicResource MaterialDesignBody}" Width="300" Margin="165,100,0,50">

<TextBlock Text="Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat." TextWrapping="WrapWithOverflow"/>

</ScrollViewer>

<Button Style="{StaticResource MaterialDesignIconForegroundButton}" Foreground="{StaticResource MaterialDesign.Brush.Secondary}" Width="30" Height="30" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="0,0,70,10" Click="GitHubOpen">

<materialDesign:PackIcon Kind="Github" Width="30" Height="30"/>

</Button>

<Button Content="OK" Style="{StaticResource MaterialDesignOutlinedSecondaryButton}" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="0,0,10,8" Height="32" Click="OK\_BtnClick"/>

</Grid>

</Window>

using System.Diagnostics;

using System.Windows;

namespace VacTrack

{

/// <summary>

/// Логика взаимодействия для AboutProgram.xaml

/// </summary>

public partial class AboutProgram : Window

{

public AboutProgram()

{

InitializeComponent();

}

private void GitHubOpen(object sender, RoutedEventArgs e)

{

try

{

Process.Start(new ProcessStartInfo("https://github.com/maldavan5916/VacTimeTP") { UseShellExecute = true });

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void OK\_BtnClick(object sender, RoutedEventArgs e) => Close();

}

}

<Application x:Class="VacTrack.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:VacTrack"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

StartupUri="MainWindow.xaml">

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<materialDesign:CustomColorTheme BaseTheme="Inherit" PrimaryColor="#FFE64A1C" SecondaryColor="#03DAC6" />

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesign3.Defaults.xaml"/>

<ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.PopupBox.xaml" />

</ResourceDictionary.MergedDictionaries>

<Style x:Key="CustomWindowStyle" TargetType="{x:Type Window}" BasedOn="{StaticResource MaterialDesignWindow}">

<Setter Property="WindowChrome.WindowChrome">

<Setter.Value>

<WindowChrome CaptionHeight="30"

CornerRadius="15"

GlassFrameThickness="1"

NonClientFrameEdges="None"

ResizeBorderThickness="10"

UseAeroCaptionButtons="True" />

</Setter.Value>

</Setter>

<Setter Property="BorderBrush" Value="{DynamicResource MaterialDesignPaper}" />

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Window}">

<Grid>

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="3,30,3,3">

<AdornerDecorator>

<ContentPresenter />

</AdornerDecorator>

</Border>

<DockPanel Height="30"

VerticalAlignment="Top"

LastChildFill="False">

<TextBlock Margin="10,5,0,0"

VerticalAlignment="Center"

DockPanel.Dock="Left"

FontSize="16"

Foreground="{DynamicResource MaterialDesignBody}"

Text="{TemplateBinding Title}"/>

<Button x:Name="btnClose"

Style="{StaticResource MaterialDesignFlatDarkButton}"

Foreground="Red"

Click="CloseClick"

DockPanel.Dock="Right"

WindowChrome.IsHitTestVisibleInChrome="True">

<materialDesign:PackIcon Kind="WindowClose" />

</Button>

<Button x:Name="btnRestore"

Style="{StaticResource MaterialDesignFlatDarkButton}"

Foreground="{DynamicResource MaterialDesignBody}"

Click="MaximizeRestoreClick"

DockPanel.Dock="Right"

WindowChrome.IsHitTestVisibleInChrome="True">

<materialDesign:PackIcon Kind="WindowMaximize" />

</Button>

<Button x:Name="btnMinimize"

Style="{StaticResource MaterialDesignFlatDarkButton}"

Foreground="{DynamicResource MaterialDesignBody}"

Click="MinimizeClick"

DockPanel.Dock="Right"

WindowChrome.IsHitTestVisibleInChrome="True">

<materialDesign:PackIcon Kind="WindowMinimize"/>

</Button>

</DockPanel>

</Grid>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</ResourceDictionary>

</Application.Resources>

</Application>

using System.Windows;

namespace VacTrack

{

/// <summary>

/// Interaction logic for App.xaml

/// </summary>

public partial class App : Application

{

public App()

{

InitializeComponent();

}

private void CloseClick(object sender, RoutedEventArgs e)

{

var window = (Window)((FrameworkElement)sender).TemplatedParent;

window.Close();

}

private void MaximizeRestoreClick(object sender, RoutedEventArgs e)

{

var window = (Window)((FrameworkElement)sender).TemplatedParent;

if (window.WindowState == WindowState.Normal)

{

window.WindowState = WindowState.Maximized;

}

else

{

window.WindowState = WindowState.Normal;

}

}

private void MinimizeClick(object sender, RoutedEventArgs e)

{

var window = (Window)((FrameworkElement)sender).TemplatedParent;

window.WindowState = WindowState.Minimized;

}

}

}

<Page x:Class="VacTrack.HomePage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="HomePage">

<Page.Resources>

<DrawingImage x:Key="VacTimeLogoDrawingImage">

<DrawingImage.Drawing>

<DrawingGroup ClipGeometry="M0,0 V1171 H1171 V0 H0 Z">

<DrawingGroup Opacity="1">

<DrawingGroup.ClipGeometry>

<RectangleGeometry RadiusX="0" RadiusY="0" Rect="0,0,1171,1171" />

</DrawingGroup.ClipGeometry>

<GeometryDrawing Brush="#66FFFFFF">

<GeometryDrawing.Pen>

<Pen Brush="#FF000000" Thickness="15" StartLineCap="Flat" EndLineCap="Flat" LineJoin="Miter" />

</GeometryDrawing.Pen>

<GeometryDrawing.Geometry>

<EllipseGeometry RadiusX="578" RadiusY="578" Center="585.5,585.5" />

</GeometryDrawing.Geometry>

</GeometryDrawing>

<GeometryDrawing Brush="#FF6EBFAC" Geometry="F0 M1171,1171z M0,0z M406.648,491L342.008,359.494C346.429,351.451,354.982,346,364.808,346L994.24,346C1002.71,346,1010.24,350.055,1014.99,356.331L951.135,491 406.648,491z" />

<GeometryDrawing Brush="#FF6EBFAC">

<GeometryDrawing.Geometry>

<RectangleGeometry RadiusX="33" RadiusY="33" Rect="613,467,147,537" />

</GeometryDrawing.Geometry>

</GeometryDrawing>

<GeometryDrawing Brush="#FF475653">

<GeometryDrawing.Geometry>

<RectangleGeometry RadiusX="21" RadiusY="21" Rect="436,531,127,472" />

</GeometryDrawing.Geometry>

</GeometryDrawing>

<GeometryDrawing Brush="#FF475653" Geometry="F0 M1171,1171z M0,0z M122.646,346C116.225,362.487,116.411,381.495,124.716,398.683L397.327,962.822C412.586,994.398 450.554,1007.63 482.13,992.368 513.707,977.109 526.935,939.141 511.676,907.565L240.309,346 122.646,346z" />

<GeometryDrawing Brush="#FF475653" Geometry="F1 M1171,1171z M0,0z M558.501,911.722C558.501,961.98 551,1002.72 467.001,1002.72 414.5,1002.72 417.411,1001 375.5,916 340,844 416.467,820.722 467.001,820.722 517.535,820.722 558.501,861.464 558.501,911.722z" />

</DrawingGroup>

</DrawingGroup>

</DrawingImage.Drawing>

</DrawingImage>

</Page.Resources>

<Grid>

<Image Source="{StaticResource VacTimeLogoDrawingImage}" MaxHeight="400" MaxWidth="400"/>

</Grid>

</Page>

using System.Windows.Controls;

namespace VacTrack

{

/// <summary>

/// Логика взаимодействия для HomePage.xaml

/// </summary>

public partial class HomePage : Page

{

public HomePage()

{

InitializeComponent();

}

}

}

namespace VacTrack

{

public interface ICachedPage

{

void OnNavigatedFromCache();

}

}

<Window x:Class="VacTrack.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:VacTrack"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

Style="{StaticResource CustomWindowStyle}"

mc:Ignorable="d"

Title="Учёт выпуска изделий ООО''ВакТайм''" Height="500" Width="900" MinHeight="450" MinWidth="800">

<Grid Background="{DynamicResource MaterialDesign.Brush.Background}">

<Menu IsMainMenu="True" VerticalAlignment="Top" materialDesign:MenuAssist.TopLevelMenuItemHeight="20">

<MenuItem Header="Файл">

<MenuItem Header="Настройки">

<MenuItem x:Name="MenuThemeToggle" Header="Тёмная тема" IsCheckable="True" IsChecked="True" Click="MenuThemeToggle\_Click"/>

</MenuItem>

<MenuItem Header="Создать новое окно" Click="CreateNewWindow"/>

<MenuItem Header="Начальний экран" Click="NavigateToPage" Tag="Home"/>

<MenuItem Header="Выход" Click="Close"/>

</MenuItem>

<MenuItem Header="Справочники">

<MenuItem Header="Сотрудники" Click="NavigateToPage" Tag="Employee"/>

<MenuItem Header="Должности" Click="NavigateToPage" Tag="Post"/>

<MenuItem Header="Изделия" Click="NavigateToPage" Tag="Product"/>

<MenuItem Header="Единицы измерения" Click="NavigateToPage" Tag="Unit"/>

<MenuItem Header="Места хранения" Click="NavigateToPage" Tag="Location"/>

<MenuItem Header="Подразделения" Click="NavigateToPage" Tag="Division"/>

<MenuItem Header="Материалы\Коплектуюшие" Click="NavigateToPage" Tag="Material"/>

<MenuItem Header="Контрагенты" Click="NavigateToPage" Tag="Counterpartie"/>

</MenuItem>

<MenuItem Header="Документы">

<MenuItem Header="Договора" Click="NavigateToPage" Tag="Contract"/>

<MenuItem Header="Поступления" Click="NavigateToPage" Tag="Receipt"/>

<MenuItem Header="Реализация" Click="NavigateToPage" Tag="Sale"/>

</MenuItem>

<MenuItem Header="Отчёты">

<MenuItem Header="Отчет по использованию материалов на изделия" Click="NavigateToPage" Tag="MaterialUsageReport" ToolTip="Показывает, какие материалы использовались для производства определенных изделий, их количество и стоимость."/>

<MenuItem Header="Отчет по сотрудникам и их подразделениям" Click="NavigateToPage" Tag="EmployeeDivisionReport" ToolTip="Список всех сотрудников с указанием их подразделений, должностей, даты приема и даты увольнения (если есть)."/>

<MenuItem Header="Отчет по остаткам материалов и комплектуюших" Click="NavigateToPage" Tag="StockBalanceReport" ToolTip="Информация о текущих остатках материалов и комплектуюших с указанием мест хранения."/>

<MenuItem Header="Отчет по договорам с контрагентами" Click="NavigateToPage" Tag="ContractorContractsReport" ToolTip="Информация о заключенных договорах с контрагентами, их суммах и изделиях."/>

<MenuItem Header="Отчет по реализации продукции" Click="NavigateToPage" Tag="ProductSalesReport" ToolTip="Детализация продаж продукции: что было продано, кому, за какую сумму и в каком количестве."/>

</MenuItem>

<MenuItem Header="Справка">

<MenuItem Header="О программе" Click="OpenAboutProgram"/>

<MenuItem Header="Помощь" Click="OpenHelp"/>

</MenuItem>

</Menu>

<Frame x:Name="MainFrame" NavigationUIVisibility="Hidden" Margin="5,25,5,5" />

</Grid>

</Window>

using DatabaseManager;

using MaterialDesignThemes.Wpf;

using System.Diagnostics;

using System.Windows;

using System.Windows.Controls;

namespace VacTrack

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

readonly DatabaseContext Db = new();

private readonly Dictionary<string, Page> \_pagesCache = [];

private readonly PaletteHelper \_paletteHelper = new();

public MainWindow()

{

Db.Database.EnsureCreated();

InitializeComponent();

MainFrame.Navigate(new HomePage());

}

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

{

var theme = \_paletteHelper.GetTheme();

if (MenuThemeToggle.IsChecked)

theme.SetDarkTheme();

else

theme.SetLightTheme();

\_paletteHelper.SetTheme(theme);

}

private void Close(object sender, RoutedEventArgs e)

{

Close();

}

private void OpenAboutProgram(object sender, RoutedEventArgs e) => new AboutProgram().ShowDialog();

private void CreateNewWindow(object sender, RoutedEventArgs e) => new MainWindow().Show();

private void OpenHelp(object sender, RoutedEventArgs e)

{

throw new NotImplementedException();

}

private void NavigateToPage(object sender, RoutedEventArgs e)

{

try

{

var menuItem = sender as MenuItem;

string? pageKey = menuItem?.Tag.ToString();

if (string.IsNullOrEmpty(pageKey))

return;

// Проверяем, есть ли страница в кэше

if (!\_pagesCache.TryGetValue(pageKey, out Page? targetPage))

{

// Создаем новую страницу, если её нет в кэше

targetPage = pageKey switch

{

"Contract" => new ViewTables.ContractViewTable(),

"Counterpartie" => new ViewTables.CounterpartieViewTable(),

"Division" => new ViewTables.DivisionViewTable(),

"Employee" => new ViewTables.EmployeeViewTable(),

"Location" => new ViewTables.LocationViewTable(),

"Material" => new ViewTables.MaterialViewTable(),

"Post" => new ViewTables.PostViewTable(),

"Product" => new ViewTables.ProductViewTable(),

"Receipt" => new ViewTables.ReceiptViewTable(),

"Sale" => new ViewTables.SaleViewTable(),

"Unit" => new ViewTables.UnitViewTable(),

"Home" => new HomePage(),

"MaterialUsageReport" => new ViewReport.MaterialUsageReport(),

"EmployeeDivisionReport" => new ViewReport.EmployeeDivisionReport(),

"StockBalanceReport" => new ViewReport.StockBalanceReport(),

"ContractorContractsReport" => new ViewReport.ContractorContractsReport(),

"ProductSalesReport" => new ViewReport.ProductSalesReport(),

\_ => new NotFoundPage("Запрашиваемая страница не найдена"),

};

// Добавляем новую страницу в кэш

\_pagesCache[pageKey] = targetPage;

}

else

{

// Если страница из кэша, уведомляем её

if (targetPage is ICachedPage cachedPage)

cachedPage.OnNavigatedFromCache();

}

// Навигация на найденную или созданную страницу

MainFrame.Navigate(targetPage);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

} <Page x:Class="VacTrack.NotFoundPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:VacTrack"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="NotFoundPage">

<Page.Resources>

<DrawingImage x:Key="status\_notfound\_borderless\_svgrepo\_comDrawingImage">

<DrawingImage.Drawing>

<DrawingGroup ClipGeometry="M0,0 V22 H22 V0 H0 Z">

<GeometryDrawing Brush="{DynamicResource MaterialDesignBody}" Geometry="F1 M22,22z M0,0z M12.822,11.29C13.638,10.709 14.243,9.942 14.505,8.968 15.108,6.725 13.532,4.415 10.975,4.415 9.825,4.415 8.89,4.825 8.2,5.504 7.78,5.917 7.528,6.339 7.4,6.671A1.179,1.179,0,0,0,9.6,7.518C9.616,7.475 9.7,7.334 9.852,7.184 10.116,6.925 10.465,6.772 10.975,6.772 11.913,6.772 12.445,7.552 12.229,8.356 12.124,8.746 11.859,9.082 11.456,9.368A3.25,3.25,0,0,1,10.511,9.838A1.179,1.179,0,0,0,9.637,10.976L9.637,13.21A1.179,1.179,0,1,0,11.995,13.21L11.995,11.78A5.9,5.9,0,0,0,12.822,11.288z" />

<GeometryDrawing Brush="{DynamicResource MaterialDesignBody}">

<GeometryDrawing.Geometry>

<EllipseGeometry RadiusX="1.275" RadiusY="1.322" Center="10.825,16.711" />

</GeometryDrawing.Geometry>

</GeometryDrawing>

</DrawingGroup>

</DrawingImage.Drawing>

</DrawingImage>

</Page.Resources>

<Grid>

<StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">

<Image Source="{StaticResource status\_notfound\_borderless\_svgrepo\_comDrawingImage}" Width="200" Height="200"/>

<TextBlock x:Name="textBlock" Text="base text" HorizontalAlignment="Center" FontSize="20" Foreground="{DynamicResource MaterialDesignBody}"/>

</StackPanel>

</Grid>

</Page>

using System.Windows.Controls;

namespace VacTrack

{

/// <summary>

/// Логика взаимодействия для NotFoundPage.xaml

/// </summary>

public partial class NotFoundPage : Page

{

public NotFoundPage(string message)

{

InitializeComponent();

textBlock.Text = message;

}

}

}

using System.Windows.Input;

public class RelayCommand : ICommand

{

private readonly Action<object> \_execute;

private readonly Predicate<object> \_canExecute;

public RelayCommand(Action<object> execute, Predicate<object> canExecute = null)

{

\_execute = execute;

\_canExecute = canExecute;

}

public bool CanExecute(object parameter) => \_canExecute?.Invoke(parameter) ?? true;

public void Execute(object parameter) => \_execute(parameter);

public event EventHandler CanExecuteChanged

{

add => CommandManager.RequerySuggested += value;

remove => CommandManager.RequerySuggested -= value;

}

}