МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ П. О. СУХОГО»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Избранные главы информатики»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 3

«Разработка веб-приложения для организации удобного пользовательского интерфейса»

Выполнил:
студент группы ИП-32
Суховенко Э. С.
Принял:
Преподаватель
Процкая М.А.

Гомель 2022

Цель работы. Ознакомиться с возможностями ASP.NET Core и других Web технологий для создания простых Web приложений.

Задание.

Разработать веб-приложение на базе ASP. NET Core MVC, предоставляющее пользовательский интерфейс для выбранной предметной области и позволяющее пользователю выполнять разные операции с данными. Для выполнения операций нужно использовать слой BLL (разработанный ранее).

Приложение должно обеспечивать набор базовых операций, таких как редактирование справочников, запрос данных, изменение, удаление, добавление новых. В зависимости от выбранной предметной области, набор операций/действий и данных может отличаться.

Это приложение должно содержать ссылку на BLL и **не должно содержать ссылку на DAL**.

В приложении должно быть минимум 3 представления. Все страницы должны быть оформлены в едином стиле, основанном на использовании одной или нескольких мастер-страниц, и иметь систему навигации (строка меню, гиперссылки, кнопки)

Для создания внешнего вида веб-страниц можно использовать общеизвестные и популярные фреймворки, например, Bootstrap.

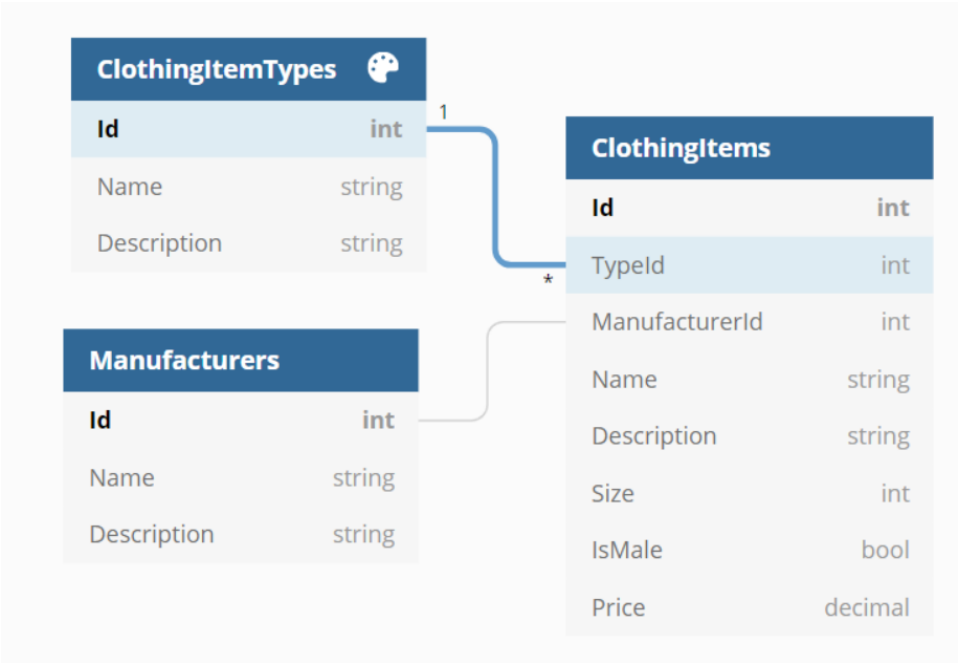

Рисунок 1 – диаграмма базы данных

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Text;

namespace ClothesShop.BLL.DTO
{
   public class ClothingItemDTO
   {
      public int Id { get; set; }
      [Required]
```

```
        public string Name { get; set; }
        [Required]
        public string Description { get; set; }
        [Required]
        [Range(30, 70, ErrorMessage = "Size should be in range from 30 to 70")]
        public int Size { get; set; }
        [DisplayName("Male")]
        public bool IsMale { get; set; }
        [Required]
        [Range(1, int.MaxValue, ErrorMessage = "Price should be positive")]
        public int Price { get; set; }

        [DisplayName("Type")]
        public int ClothingItemTypeId { get; set; }

        [DisplayName("Manufacturer")]
        public int ManufacturerId { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Text;

namespace ClothesShop.BLL.DTO
{
    public class ClothingItemTypeDTO
    {
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        [Required]
        public string Description { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Text;

namespace ClothesShop.BLL.DTO
{
    public class ManufacturerDTO
    {
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        [Required]
```

```
        public string Description { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ClothesShop.BLL.Interfaces
{
    public interface IEntityService<T> where T : class
    {
        void Create(T item);
        void Update(T item);
        void Delete(int id);
        IEnumerable<T> Get();
        T Get(int id);
    }
}

using AutoMapper;
using ClothesShop.BLL.DTO;
using ClothesShop.BLL.Exceptions;
using ClothesShop.BLL.Interfaces;
using ClothesShop.BLL.Interfaces.EntityServices;
using ClothesShop.DAL.Entities;
using ClothesShop.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ClothesShop.BLL.Services
{
    public class ClothingItemService : IClothingItemService
    {
        private IUnitOfWork _storage;

        public ClothingItemService(IUnitOfWork storage)
        {
            _storage = storage;
        }

        public void Create(ClothingItemDTO item)
        {
            try
```

```
        {
            Validate(item);
            ClothingItem clothItem = new MapperConfiguration(cfg =>
cfg.CreateMap<ClothingItemDTO, ClothingItem>())
                .CreateMapper()
                .Map<ClothingItem>(item);
            _storage.ClothingItems.Create(clothItem);
            _storage.Save();
        }
        catch (Exception exception)
        {
            throw new EntityServiceException($"Невозможно добавить одежду. {ex-
ception.Message}");
        }
    }

    public void Delete(int id)
    {
        try
        {
            _storage.ClothingItems.Delete(id);
            _storage.Save();
        }
        catch (Exception exception)
        {
            throw new EntityServiceException($"Невозможно удалить одежду. {ex-
ception.Message}");
        }
    }

    public IEnumerable<ClothingItemDTO> Get()
    {
        try
        {
            var clothingItems = _storage.ClothingItems.Get();
            return new MapperConfiguration(cfg => cfg.CreateMap<ClothingItem,
ClothingItemDTO>())
                .CreateMapper()
                .Map<List<ClothingItemDTO>>(clothingItems);
        }
        catch (Exception exception)
        {
            throw new EntityServiceException($"Невозможно получить одежду. {ex-
ception.Message}");
        }
    }

    public ClothingItemDTO Get(int id)
```

```
        {
            try
            {
                var clothingItem = _storage.ClothingItems.Get(id);
                return new MapperConfiguration(cfg => cfg.CreateMap<ClothingItem,
ClothingItemDTO>())
                    .CreateMapper()
                    .Map<ClothingItemDTO>(clothingItem);
            }
            catch (Exception exception)
            {
                throw new EntityServiceException($"Невозможно получить одежду. {ex-
ception.Message}");
            }
        }

        public void Update(ClothingItemDTO item)
        {
            try
            {
                Validate(item);
                var clothingItem = new MapperConfiguration(cfg =>
cfg.CreateMap<ClothingItemDTO, ClothingItem>())
                .CreateMapper()
                .Map<ClothingItem>(item);
                var clothItem = _storage.ClothingItems.Find(m => m.Id == cloth-
ingItem.Id).FirstOrDefault();

                clothItem.Name = clothingItem.Name;
                clothItem.Description = clothingItem.Description;
                clothItem.Size = clothingItem.Size;
                clothItem.IsMale = clothingItem.IsMale;
                clothItem.Price = clothingItem.Price;
                clothItem.ClothingItemTypeId = clothingItem.ClothingItemTypeId;
                clothItem.ManufacturerId = clothingItem.ManufacturerId;

                _storage.ClothingItems.Update(clothItem);
                _storage.Save();
            }
            catch (Exception exception)
            {
                throw new EntityServiceException($"Невозможно обновить одежду. {ex-
ception.Message}");
            }
        }

        public bool Exists(int id)
        {
```

```
            if(_storage.ClothingItems.Find(item => item.Id == id).Count() == 1)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public void Validate(ClothingItemDTO item)
        {
            if (item.Name == null || item.Name.Length < 4)
            {
                throw new ValidationException("Короткое название одежды. Должно
быть больше 3 символов", "Name");
            }

            if (item.Description == null || item.Description.Length < 4)
            {
                throw new ValidationException("Короткое описание одежды. Должно
быть больше 3 символов", "Description");
            }

            if (item.Size < 20 || item.Size > 70)
            {
                throw new ValidationException("Неверный размер. Размер одежды дол-
жен быть в диапазоне от 20 до 70.", "Size");
            }

            if (item.Price < 1)
            {
                throw new ValidationException("Неверная цена. Цена должна быть по-
ложительной", "Size");
            }

            try
            {
                _storage.ClothingItemTypes.Get(item.ClothingItemTypeId);
            }
            catch
            {
                throw new Exception("Нет такого типа одежды");
            }

            try
            {
                _storage.Manufacturers.Get(item.ManufacturerId);
```

```
            }
            catch
            {
                throw new Exception("Нет такого производителя");
            }
        }
    }
}
using AutoMapper;
using ClothesShop.BLL.DTO;
using ClothesShop.BLL.Exceptions;
using ClothesShop.BLL.Interfaces;
using ClothesShop.BLL.Interfaces.EntityServices;
using ClothesShop.DAL.Entities;
using ClothesShop.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ClothesShop.BLL.Services
{
    public class ClothingItemTypeService : IClothingItemTypeService
    {
        private IUnitOfWork _storage;

        public ClothingItemTypeService(IUnitOfWork storage)
        {
            _storage = storage;
        }

        public void Create(ClothingItemTypeDTO item)
        {
            try
            {
                Validate(item);
                ClothingItemType type = new MapperConfiguration(cfg =>
cfg.CreateMap<ClothingItemTypeDTO, ClothingItemType>())
                    .CreateMapper()
                    .Map<ClothingItemType>(item);
                _storage.ClothingItemTypes.Create(type);
                _storage.Save();
            }
            catch (Exception exception)
            {
                throw new EntityServiceException($"Невозможно добавить тип одежды.
{exception.Message}");
```

```
        }
    }

    public void Delete(int id)
    {
        try
        {
            _storage.ClothingItemTypes.Delete(id);
            _storage.Save();
        }
        catch (Exception exception)
        {
            throw new EntityServiceException($"Невозможно удалить тип одежды.
{exception.Message}");
        }
    }

    public IEnumerable<ClothingItemTypeDTO> Get()
    {
        try
        {
            var types = _storage.ClothingItemTypes.Get();
            return new MapperConfiguration(cfg =>
cfg.CreateMap<ClothingItemType, ClothingItemTypeDTO>())
                .CreateMapper()
                .Map<List<ClothingItemTypeDTO>>(types);
        }
        catch (Exception exception)
        {
            throw new EntityServiceException($"Невозможно получить типы одеж-
ды. {exception.Message}");
        }
    }

    public ClothingItemTypeDTO Get(int id)
    {
        try
        {
            var type = _storage.ClothingItemTypes.Get(id);
            return new MapperConfiguration(cfg =>
cfg.CreateMap<ClothingItemType, ClothingItemTypeDTO>())
                .CreateMapper()
                .Map<ClothingItemTypeDTO>(type);
        }
        catch (Exception exception)
        {
            throw new EntityServiceException($"Невозможно получить тип одежды.
{exception.Message}");
```

```
        }
    }

    public void Update(ClothingItemTypeDTO item)
    {
        try
        {
            Validate(item);
            var type = new MapperConfiguration(cfg =>
cfg.CreateMap<ClothingItemTypeDTO, ClothingItemType>())
            .CreateMapper()
            .Map<ClothingItemType>(item);
            var itemType = _storage.ClothingItemTypes.Find(m => m.Id ==
type.Id).FirstOrDefault();
            itemType.Name = type.Name;
            itemType.Description = type.Description;
            _storage.ClothingItemTypes.Update(itemType);
            _storage.Save();
        }
        catch (Exception exception)
        {
            throw new EntityServiceException($"Невозможно обновить тип одежды.
{exception.Message}");
        }
    }
    public bool Exists(int id)
    {
        if (_storage.ClothingItemTypes.Find(item => item.Id == id).Count() == 1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    private void Validate(ClothingItemTypeDTO item)
    {
        if (item.Name == null || item.Name.Length < 4)
        {
            throw new ValidationException("Короткое название типа одежды.
Должно быть больше 3 символов", "Name");
        }

        if (item.Description == null || item.Description.Length < 4)
        {
```

```
                throw new ValidationException("Короткое описание типа одежды.
Должно быть больше 3 символов", "Description");
            }
        }
    }
}

using AutoMapper;
using ClothesShop.BLL.DTO;
using ClothesShop.BLL.Exceptions;
using ClothesShop.BLL.Interfaces.EntityServices;
using ClothesShop.DAL.Entities;
using ClothesShop.DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ClothesShop.BLL.Services
{
    public class ManufacturerService : IManufacturerService
    {
        private IUnitOfWork _storage;

        public ManufacturerService(IUnitOfWork storage)
        {
            _storage = storage;
        }

        public void Create(ManufacturerDTO item)
        {
            try
            {
                Validate(item);
                Manufacturer man = new MapperConfiguration(cfg =>
cfg.CreateMap<ManufacturerDTO, Manufacturer>())
                    .CreateMapper()
                    .Map<Manufacturer>(item);
                _storage.Manufacturers.Create(man);
                _storage.Save();
            }
            catch (Exception exception)
            {
                throw new EntityServiceException($"Невозможно добавить производи-
теля. {exception.Message}");
            }
        }
```

```
        public void Delete(int id)
        {
            try
            {
                _storage.Manufacturers.Delete(id);
                _storage.Save();
            }
            catch (Exception exception)
            {
                throw new EntityServiceException($"Невозможно удалить производите-
ля. {exception.Message}");
            }
        }

        public IEnumerable<ManufacturerDTO> Get()
        {
            try
            {
                var manufacturers = _storage.Manufacturers.Get();
                return new MapperConfiguration(cfg => cfg.CreateMap<Manufacturer,
ManufacturerDTO>())
                    .CreateMapper()
                    .Map<List<ManufacturerDTO>>(manufacturers);
            }
            catch (Exception exception)
            {
                throw new EntityServiceException($"Невозможно получить производи-
телей. {exception.Message}");
            }
        }

        public ManufacturerDTO Get(int id)
        {
            try
            {
                var manufacturer = _storage.Manufacturers.Get(id);
                return new MapperConfiguration(cfg => cfg.CreateMap<Manufacturer,
ManufacturerDTO>())
                    .CreateMapper()
                    .Map<ManufacturerDTO>(manufacturer);
            }
            catch (Exception exception)
            {
                throw new EntityServiceException($"Невозможно получить производи-
теля. {exception.Message}");
            }
        }
```

```
        public void Update(ManufacturerDTO item)
        {
            try
            {
                Validate(item);
                var manufacturer = new MapperConfiguration(cfg =>
cfg.CreateMap<ManufacturerDTO, Manufacturer>())
                .CreateMapper()
                .Map<Manufacturer>(item);
                var man = _storage.Manufacturers.Find(m => m.Id == manufactur-
er.Id).FirstOrDefault();
                man.Name = manufacturer.Name;
                man.Description = manufacturer.Description;
                _storage.Manufacturers.Update(man);
                _storage.Save();
            }
            catch (Exception exception)
            {
                throw new EntityServiceException($"Невозможно обновить производи-
теля. {exception.Message}");
            }
        }

        private void Validate(ManufacturerDTO item)
        {
            if (item.Name == null || item.Name.Length < 4)
            {
                throw new ValidationException("Короткое имя производителя. Должно
быть больше 3 символов", "Name");
            }

            if (item.Description == null || item.Description.Length < 4)
            {
                throw new ValidationException("Короткое описание производителя.
Должно быть больше 3 символов", "Description");
            }
        }
        public bool Exists(int id)
        {
            if (_storage.Manufacturers.Find(item => item.Id == id).Count() == 1)
            {
                return true;
            }
            else
            {
                return false;
            }
```

```
        }

        public IEnumerable<ManufacturerDTO> FilterByNameContainedText(string
text)
        {
            var mans = _storage.Manufacturers.Find(man => man.Name.Contains(text));
            return new MapperConfiguration(cfg => cfg.CreateMap<Manufacturer, Man-
ufacturerDTO>())
                .CreateMapper()
                .Map<List<ManufacturerDTO>>(mans);
        }
    }
}
using ClothesShop.DAL.Entities;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Text;

namespace ClothesShop.DAL.EF
{
    public class ClothesShopContext : DbContext
    {
        public DbSet<ClothingItemType> ClothingItemTypes { get; set; }
        public DbSet<Manufacturer> Manufacturers { get; set; }
        public DbSet<ClothingItem> ClothingItems { get; set; }
        public ClothesShopContext(DbContextOptions options) : base(options)
        {
            Database.EnsureCreated();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Text;

namespace ClothesShop.DAL.Entities
{
    public class ClothingItem
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public int Size { get; set; }
        public bool IsMale { get; set; }
        public int Price { get; set; }

        public int ClothingItemTypeId { get; set; }
```

```
        public ClothingItemType ClothingItemType { get; set; }

        public int ManufacturerId { get; set; }
        public Manufacturer Manufacturer { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Text;

namespace ClothesShop.DAL.Entities
{
    public class ClothingItemType
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Text;

namespace ClothesShop.DAL.Entities
{
    public class Manufacturer
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
    }
}
using ClothesShop.DAL.EF;
using ClothesShop.DAL.Entities;
using ClothesShop.DAL.Interfaces;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ClothesShop.DAL.Repositories
{
    public class ClothingItemDbRepository : IRepository<ClothingItem>
    {
        private readonly ClothesShopContext _context;
        public ClothingItemDbRepository(ClothesShopContext context)
        {
```

```
        _context = context;
    }
    public void Create(ClothingItem item)
    {
        _context.ClothingItems.Add(item);
    }

    public void Delete(int id)
    {
        var item = _context.ClothingItems.Find(id);
        if(item != null)
        {
            _context.ClothingItems.Remove(item);
        }
        else
        {
            throw new Exception("Такая одежда не найдена");
        }
    }

    public IEnumerable<ClothingItem> Find(Func<ClothingItem, bool> predicate)
    {
        return _context.ClothingItems.Where(predicate).ToList();
    }

    public IEnumerable<ClothingItem> Get()
    {
        return _context.ClothingItems.ToList();
    }

    public ClothingItem Get(int id)
    {
        var item = _context.ClothingItems.Find(id);
        if (item != null)
        {
            return item;
        }
        else
        {
            throw new Exception("Такая одежда не найдена");
        }
    }

    public void Update(ClothingItem item)
    {
        _context.Entry(item).State = EntityState.Modified;
    }
}
```

```
}using ClothesShop.DAL.EF;
using ClothesShop.DAL.Entities;
using ClothesShop.DAL.Interfaces;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ClothesShop.DAL.Repositories
{
    public class ClothingItemTypeDbRepository : IRepository<ClothingItemType>
    {
        private readonly ClothesShopContext _context;
        public ClothingItemTypeDbRepository(ClothesShopContext context)
        {
            _context = context;
        }
        public void Create(ClothingItemType item)
        {
            _context.ClothingItemTypes.Add(item);
        }

        public void Delete(int id)
        {
            var type = _context.ClothingItemTypes.Find(id);
            if(type != null)
            {
                _context.ClothingItemTypes.Remove(type);
            }
            else
            {
                throw new Exception("Такой тип одежды не найден");
            }
        }

        public IEnumerable<ClothingItemType> Find(Func<ClothingItemType, bool>
predicate)
        {
            return _context.ClothingItemTypes.Where(predicate).ToList();
        }

        public IEnumerable<ClothingItemType> Get()
        {
            return _context.ClothingItemTypes.ToList();
        }

        public ClothingItemType Get(int id)
```

```
        {
            var type = _context.ClothingItemTypes.Find(id);
            if (type != null)
            {
                return type;
            }
            else
            {
                throw new Exception("Такой тип одежды не найден");
            }
        }

        public void Update(ClothingItemType item)
        {
            _context.Entry(item).State = EntityState.Modified;
        }
    }
}
using ClothesShop.DAL.EF;
using ClothesShop.DAL.Entities;
using ClothesShop.DAL.Interfaces;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ClothesShop.DAL.Repositories
{
    public class ManufacturerDbRepository : IRepository<Manufacturer>
    {
        private readonly ClothesShopContext _context;
        public ManufacturerDbRepository(ClothesShopContext context)
        {
            _context = context;
        }
        public void Create(Manufacturer item)
        {
            _context.Manufacturers.Add(item);
        }

        public void Delete(int id)
        {
            var man = _context.Manufacturers.Find(id);
            if (man != null)
            {
                _context.Manufacturers.Remove(man);
            }
```

```
            else
            {
                throw new Exception("Такой производитель не найден");
            }
        }

        public IEnumerable<Manufacturer> Find(Func<Manufacturer, bool> predicate)
        {
            return _context.Manufacturers.Where(predicate).ToList();
        }

        public IEnumerable<Manufacturer> Get()
        {
            return _context.Manufacturers.ToList();
        }

        public Manufacturer Get(int id)
        {
            var man = _context.Manufacturers.Find(id);
            if (man != null)
            {
                return man;
            }
            else
            {
                throw new Exception("Такой производитель не найден");
            }
        }

        public void Update(Manufacturer item)
        {
            _context.Entry(item).State = EntityState.Modified;
        }
    }
}
```

Вывод: Ознакомился с возможностями ASP.NET Core и других Web технологий для создания простых Web приложений.