МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ВТ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Программирование в среде .NET»

Тема: РЕАЛИЗАЦИЯ СЛОЯ ДОСТУПА К ДАННЫМ ПРИЛОЖЕНИЯ

Студент гр. 6306 Милакин А.Д.

Преподаватель Пешехонов К. А.

Санкт-Петербург
2020

Цель работы

Исследовать вопрос реализации слоя доступа к данным приложения средствами языка C#.

Задание

Написать слой доступа к данным: Entity Framework Code First + MS SQL Server Developer Edition

Код программы

Код программы целиком выложен на гитхаб. Приведем пример сущности, интерфейса и реализации класса доступа и соответствующий контекст.

Код сущности

```
namespace Media.DataAccess.Entities
{
  public class Podcast
  {
    [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    //идентификатор
    public int Id { get; set; }

    //Название фильма
    public string Title { get; set; }

    //Автор
    public string Author { get; set; }

    //Длительность
    public int Duration { get; set; }
    //public int? AlbumId => Album.Id;
    public virtual Album Album { get; set; }
    public int? AlbumId { get; set; }
  }
}
```

Код интерфейса

```
namespace Media.DataAccess.Contracts
{
  public interface IPodcastDataAccess
  {
    Task<Podcast> InsertAsync(PodcastUpdateModel movie);
    Task<IEnumerable<Podcast>> GetAsync();
    Task<Podcast> GetAsync(IPodcastIdentity movieId);
    Task<Podcast> UpdateAsync(PodcastUpdateModel movie);
    Task<Podcast> GetByAsync(IPodcastContainer movie);
  }
}
```

Код реализации

```
namespace Media.DataAccess.Implementations
{
  public class PodcastDataAccess : IPodcastDataAccess
  {
    private AlbumContext Context { get; }
    private IMapper Mapper { get; }

    public PodcastDataAccess(AlbumContext context, IMapper mapper)
    {
      this.Context = context;
      Mapper = mapper;
    }

    public async Task<Podcast> InsertAsync(PodcastUpdateModel podcast)
    {
      var result = await this.Context.AddAsync(this.Mapper.Map<DataAccess.Entities.Podcast>(podcast));
      await this.Context.SaveChangesAsync();
      return this.Mapper.Map<Podcast>(result.Entity);
    }

    public async Task<IEnumerable<Podcast>> GetAsync()
    {
      return this.Mapper.Map<IEnumerable<Podcast>>(
        await this.Context.Podcast.Include(x => x.Album).ToListAsync());

    }

    public async Task<Podcast> GetAsync(IPodcastIdentity podcastId)
    {
      var result = await this.Get(podcastId);
      return this.Mapper.Map<Podcast>(result);
    }

    public async Task<Podcast> UpdateAsync(PodcastUpdateModel podcast)
    {
      var existing = await this.Get(podcast);
```

```
        var result = this.Mapper.Map(podcast, existing);

        this.Context.Update(result);

        await this.Context.SaveChangesAsync();

        return this.Mapper.Map<Podcast>(result);
    }

    public async Task<Podcast> GetByAsync(IPodcastContainer podcast)
    {
        return podcast.PodcastId.HasValue
            ? this.Mapper.Map<Podcast>(await this.Context.Podcast.FirstOrDefaultAsync(x => x.Id ==
podcast.PodcastId))
            : null;
    }

    private async Task<Media.DataAccess.Entities.Podcast> Get(IPodcastIdentity podcastId)
    {
        if(podcastId == null)
            throw new ArgumentNullException(nameof(podcastId));
        return await this.Context.Podcast.Include(x => x.Album).FirstOrDefaultAsync(x => x.Id ==
podcastId.Id);
    }
  }
}
```

Код контекста

```
namespace Media.DataAccess.Context
{
  public partial class AlbumContext : DbContext
  {
    public AlbumContext()
    {
    }

    public AlbumContext(DbContextOptions<AlbumContext> options) : base(options)
    {
    }

    public virtual DbSet<Album> Album { get; set; }
    public virtual DbSet<Track> Track { get; set; }
    public virtual DbSet<Podcast> Podcast { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {

    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Album>(entity =>
```

```
            {
                entity.Property(a                                                                    =>
a.Id).UseIdentityColumn().Metadata.SetBeforeSaveBehavior(PropertySaveBehavior.Ignore);
                entity.Property(a => a.Name).IsRequired();
                entity.Property(a => a.Address).IsRequired();
            });

        modelBuilder.Entity<Track>(entity =>
            {
                entity.Property(t                                                                    =>
t.Id).UseIdentityColumn().Metadata.SetBeforeSaveBehavior(PropertySaveBehavior.Ignore);
                entity.Property(t => t.Title).IsRequired();
                entity.Property(t => t.Author).IsRequired();
                entity.Property((t => t.Duration)).IsRequired();
                entity.HasOne(t => t.Album)
                    .WithMany(a => a.Track)
                    .HasForeignKey(t => t.AlbumId)
                    .HasConstraintName("FK_Track_Album");
            });

        modelBuilder.Entity<Podcast>(entity =>
            {
                entity.Property(p                                                                    =>
p.Id).UseIdentityColumn().Metadata.SetBeforeSaveBehavior(PropertySaveBehavior.Ignore);
                entity.Property(p => p.Title).IsRequired();
                entity.Property(p => p.Author).IsRequired();
                entity.Property((p => p.Duration)).IsRequired();
                entity.HasOne(p => p.Album)
                    .WithMany(a => a.Podcast)
                    .HasForeignKey(p => p.AlbumId)
                    .HasConstraintName("FK_Podcast_Album");
            });
        this.OnModelCreatingPartial(modelBuilder);
    }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
  }
}
```

Вывод

В процессе выполнения лабораторной работы мы исследовали вопрос реализации слоя доступа к данным приложения средствами языка C#.