```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using MusicStore.Logic.Context;

namespace MusicStore.Logic.Controllers
{
    internal abstract class GenericController<E, I> : ControllerObject, IController<I>
        where E : Entities.EntityObject, I, Contracts.ICopyable<I>, new()
        where I : Contracts.IIdentifiable
    {
        protected abstract List<E> Set { get; }

        protected GenericController(ContextObject contextObject)
            : base(contextObject)
        {

        }
        protected GenericController(ControllerObject controllerObject)
            : base(controllerObject)
        {

        }

        public virtual I Create()
        {
            return new E();
        }

        public virtual IEnumerable<I> GetAll()
        {
            return Set.Where(i => i.State != Entities.EntityState.Deleted)
                      .Select(i =>
                      {
                          var result = new E();

                          result.CopyProperties(i);
                          return result;
                      });
        }
        public virtual I GetById(int id)
        {
            var result = default(E);
            var item = Set.SingleOrDefault(i => i.State != Entities.EntityState.Deleted && i.Id == id);

            if (item != null)
            {
                result = new E();
                result.CopyProperties(item);
            }
            return result;
        }
        protected virtual void BeforeInserting(I entity)
        {
        }
        public virtual I Insert(I entity)
        {
            if (entity == null)
                throw new ArgumentNullException(nameof(entity));

            BeforeInserting(entity);
            var insertEntity = new E();

            insertEntity.CopyProperties(entity);
            insertEntity.Id = 0;
            insertEntity.State = Entities.EntityState.Added;
            Set.Add(insertEntity);
            AfterInserted(insertEntity);
            return insertEntity;
        }
        protected virtual void AfterInserted(E entity)
        {

        }

        protected virtual void BeforeUpdating(I entity)
        {

        }
        public virtual void Update(I entity)
        {
            if (entity == null)
                throw new ArgumentNullException(nameof(entity));

            BeforeUpdating(entity);
            var updateEntity = Set.SingleOrDefault(i => i.State !=
              Entities.EntityState.Deleted && i.Id == entity.Id);

            if (updateEntity != null)
            {
                updateEntity.CopyProperties(entity);
                AfterUpdated(updateEntity);
                updateEntity.State = Entities.EntityState.Modified;
            }
        }
        protected virtual void AfterUpdated(E entity)
        {

        }

        protected virtual void BeforeDeleting(I entity)
        {

        }
        public void Delete(int id)
        {
            var item = Set.SingleOrDefault(i => i.State !=
              Entities.EntityState.Deleted && i.Id == id);

            if (item != null)
            {
                BeforeDeleting(item);
                item.State = Entities.EntityState.Deleted;
                AfterDeleted(item);
            }
        }
        protected virtual void AfterDeleted(E entity)
        {

        }

        public void Save()
        {
            Context.Save();
        }
    }
}

```