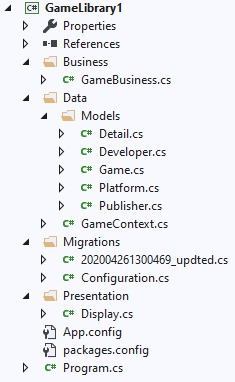
**G&Y‘s Game Store**

Изготвили: Димитър Генов и Йоан Трендафилов

ПМГ „Акад. Боян Петканчин“ град Хасково

Проект по 7-ми модул от НП „Обучение за ИТ кариера“

Overview



* Folder:**Data**

**# В тази папка създаваме моделите за таблиците, които след това реализираме в базата данни създадена в GameContext.cs**

* Folder:*Models*

Detail.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameLibrary1.Data.Models

{

public class Detail

{

public int ID { get; set; }

public string Genre { get; set; }

public string Rating { get; set; }

}

}

Developer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

namespace GameLibrary1.Data.Models

{

public class Developer

{

[Key]

public int ID { get; set; }

public string DevName { get; set; }

public string DevAdress { get; set; }

//public virtual ICollection<Game> Games { get; set; }

}

}

Platform.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameLibrary1.Data.Models

{

public class Platform

{

public int ID { get; set; }

public string Console { get; set; }

public decimal ConsolePrice { get; set; }

}

}

Publisher.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameLibrary1.Data.Models

{

public class Publisher

{

public int ID { get; set; }

public string PubName { get; set; }

public int Year { get; set; }

}

}

Game.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameLibrary1.Data.Models

{

public class Game

{

public int ID { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public int Stock { get; set; }

public int Developer { get; set; }

public int Genre { get; set; }

public int Console { get; set; }

public int Publisher { get; set; }

}

}

* *GameContext.cs*

using GameLibrary1.Data.Models;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameLibrary1.Data

{

public class GameContext:DbContext

{

public GameContext() : base("name=GameContext") { }

public DbSet<Game> Games { get; set; }

public DbSet<Developer> Developers { get; set; }

public DbSet<Detail> Details { get; set; }

public DbSet<Platform> Platforms { get; set; }

public DbSet<Publisher> Publishers { get; set; }

}

}

* Folder:**Business**

# В тази папка сме поставили функциите за работа с базата данни

* *GameBusiness.cs*

using GameLibrary1.Data;

using GameLibrary1.Presentation;

using GameLibrary1.Data.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameLibrary1.Business

{

public class GameBusiness

{

private GameContext gameContext;

//Game

public List<Game>GetGames()

{

using (gameContext=new GameContext())

{

return gameContext.Games.ToList();

}

}

public Game Get(int id)

{

using (gameContext = new GameContext())

{

return gameContext.Games.Find(id);

}

}

public void Add(Game game)

{

using (gameContext = new GameContext())

{

gameContext.Games.Add(game);

gameContext.SaveChanges();

}

}

public void Update(Game game)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Games.Find(game.ID);

if(prod!=null)

{

gameContext.Entry(prod).CurrentValues.SetValues(game);

gameContext.SaveChanges();

}

}

}

public void Delete(int id)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Games.Find(id);

if (prod != null)

{

gameContext.Games.Remove(prod);

gameContext.SaveChanges();

}

}

}

//Developer

public List<Developer> GetDevelopers()

{

using (gameContext = new GameContext())

{

return gameContext.Developers.ToList();

}

}

public void AddDev(Developer developer)

{

using (gameContext = new GameContext())

{

gameContext.Developers.Add(developer);

gameContext.SaveChanges();

}

}

public Developer GetDeveloper(int id)

{

using (gameContext = new GameContext())

{

return gameContext.Developers.Find(id);

}

}

public void UpdateDev(Developer developer)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Developers.Find(developer.ID);

if (prod != null)

{

gameContext.Entry(prod).CurrentValues.SetValues(developer);

gameContext.SaveChanges();

}

}

}

public void DeleteDev(int id)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Developers.Find(id);

if (prod != null)

{

gameContext.Developers.Remove(prod);

gameContext.SaveChanges();

}

}

}

//Details

public List<Detail> GetDetails()

{

using (gameContext = new GameContext())

{

return gameContext.Details.ToList();

}

}

public void AddDetail(Detail detail)

{

using (gameContext = new GameContext())

{

gameContext.Details.Add(detail);

gameContext.SaveChanges();

}

}

public Detail GetDetail(int id)

{

using (gameContext = new GameContext())

{

return gameContext.Details.Find(id);

}

}

public void UpdateDet(Detail detail)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Details.Find(detail.ID);

if (prod != null)

{

gameContext.Entry(prod).CurrentValues.SetValues(detail);

gameContext.SaveChanges();

}

}

}

public void DeleteDetail(int id)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Details.Find(id);

if (prod != null)

{

gameContext.Details.Remove(prod);

gameContext.SaveChanges();

}

}

}

//Platform

public List<Platform> GetPlatforms()

{

using (gameContext = new GameContext())

{

return gameContext.Platforms.ToList();

}

}

public void AddPlatform(Platform platform)

{

using (gameContext = new GameContext())

{

gameContext.Platforms.Add(platform);

gameContext.SaveChanges();

}

}

public Platform GetPlatform(int id)

{

using (gameContext = new GameContext())

{

return gameContext.Platforms.Find(id);

}

}

public void UpdatePlat(Platform platform)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Platforms.Find(platform.ID);

if (prod != null)

{

gameContext.Entry(prod).CurrentValues.SetValues(platform);

gameContext.SaveChanges();

}

}

}

public void DeletePlatform(int id)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Platforms.Find(id);

if (prod != null)

{

gameContext.Platforms.Remove(prod);

gameContext.SaveChanges();

}

}

}

//Publisher

public List<Publisher> GetPublishers()

{

using (gameContext = new GameContext())

{

return gameContext.Publishers.ToList();

}

}

public void AddPublisher(Publisher publisher)

{

using (gameContext = new GameContext())

{

gameContext.Publishers.Add(publisher);

gameContext.SaveChanges();

}

}

public Publisher GetPublisher(int id)

{

using (gameContext = new GameContext())

{

return gameContext.Publishers.Find(id);

}

}

public void UpdatePub(Publisher publisher)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Publishers.Find(publisher.ID);

if (prod != null)

{

gameContext.Entry(prod).CurrentValues.SetValues(publisher);

gameContext.SaveChanges();

}

}

}

public void DeletePublisher(int id)

{

using (gameContext = new GameContext())

{

var prod = gameContext.Publishers.Find(id);

if (prod != null)

{

gameContext.Publishers.Remove(prod);

gameContext.SaveChanges();

}

}

}

}

}

* Folder:**Presentation**

# В тази папка се случва кореспонденцията с usera. След избиране на желаната от него опция, програмата извиква нужната функция от Business папката и извършва необходимите промени или изважда необходимата информация от базата данни.

* *Display.cs*

using GameLibrary1.Business;

using GameLibrary1.Data;

using GameLibrary1.Data.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameLibrary1.Presentation

{

public class Display

{

GameBusiness gameBusiness= new GameBusiness();

private int closeOP = 6;

//GameBusiness gameBusiness;

public Display() { Input(); }

private void ShowOpt()

{

Console.WriteLine(new string('-',40));

Console.WriteLine(new string(' ', 16)+"Options"+new string(' ',16));

Console.WriteLine(new string('-', 40));

Console.WriteLine("1.Go to game menu");

Console.WriteLine("2.Go to developer menu");

Console.WriteLine("3.Go to detail menu");

Console.WriteLine("4.Go to platform menu");

Console.WriteLine("5.Go to publisher menu");

Console.WriteLine("6.Exit");

}

private void Input()

{

int lol = -1;

do

{

ShowOpt();

Console.Write("Your choice: ");

lol = int.Parse(Console.ReadLine());

switch (lol)

{

case 1:

GameMenu();

break;

case 2:

DevMenu();

break;

case 3:

DetMenu();

break;

case 4:

PlatMenu();

break;

case 5:

PubMenu();

break;

default:

break;

}

} while (lol != closeOP);

}

private void ListAll()

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 18) + "Games" + new string(' ', 18));

Console.WriteLine(new string('-', 40));

var games = gameBusiness.GetGames();

foreach(var item in games )

{

Console.WriteLine("{0} {1} {2} {3} {4} {5} {6} {7}", item.ID,item.Name,item.Price,item.Stock,item.Developer,item.Genre,item.Console,item.Publisher);

}

}

private void Add()

{

Publisher publisher = new Publisher();

Platform platform = new Platform();

Developer developer = new Developer();

Detail detail = new Detail();

Game game = new Game();

Console.WriteLine("Enter name:");

game.Name = Console.ReadLine();

Console.WriteLine("Enter price:");

game.Price = decimal.Parse(Console.ReadLine());

Console.WriteLine("Enter stock:");

game.Stock = int.Parse(Console.ReadLine());

Console.WriteLine("Ënter developer id:");

var id = int.Parse(Console.ReadLine());

developer = gameBusiness.GetDeveloper(id);

game.Developer = developer.ID;

Console.WriteLine("Enter genre id");

var id1 = int.Parse(Console.ReadLine());

detail = gameBusiness.GetDetail(id1);

game.Genre = detail.ID;

Console.WriteLine("Enter console id");

var id2 = int.Parse(Console.ReadLine());

platform = gameBusiness.GetPlatform(id2);

game.Console = platform.ID;

Console.WriteLine("Enter publisher id");

var id3 = int.Parse(Console.ReadLine());

publisher = gameBusiness.GetPublisher(id3);

game.Publisher = publisher.ID;

gameBusiness.Add(game);

}

private void Update()

{

Developer developer = new Developer();

Detail detail = new Detail();

Platform platform = new Platform();

Publisher publisher = new Publisher();

Console.WriteLine("Enter ID of game to update:");

int id = int.Parse(Console.ReadLine());

Game game = gameBusiness.Get(id);

if (game != null)

{

Console.WriteLine("Enter new name:");

game.Name = Console.ReadLine();

Console.WriteLine("Enter new price:");

game.Price = decimal.Parse(Console.ReadLine());

Console.WriteLine("Enter stock:");

game.Stock = int.Parse(Console.ReadLine());

Console.WriteLine("Ënter developer id:");

var id4 = int.Parse(Console.ReadLine());

developer = gameBusiness.GetDeveloper(id4);

game.Developer = developer.ID;

Console.WriteLine("Enter genre id");

var id1 = int.Parse(Console.ReadLine());

detail = gameBusiness.GetDetail(id1);

game.Genre = detail.ID;

Console.WriteLine("Enter console id");

var id2 = int.Parse(Console.ReadLine());

platform = gameBusiness.GetPlatform(id2);

game.Console = platform.ID;

Console.WriteLine("Enter publisher id");

var id3 = int.Parse(Console.ReadLine());

publisher = gameBusiness.GetPublisher(id3);

game.Publisher = publisher.ID;

gameBusiness.Update(game);

}

else { Console.WriteLine("Can not update what does not exist!"); }

}

private void Search()

{

Console.WriteLine("Enter id to search:");

int id = int.Parse(Console.ReadLine());

Game game = gameBusiness.Get(id);

if (game != null)

{

Console.WriteLine(new string('-', 40));

Console.WriteLine("ID: " + game.ID);

Console.WriteLine("Name: " + game.Name);

Console.WriteLine("Price: " + game.Price);

Console.WriteLine("Stock: " + game.Stock);

Developer developer = new Developer();

developer = gameBusiness.GetDeveloper(game.Developer);

Console.WriteLine("Developer: " + developer.DevName);

Detail detail = new Detail();

detail = gameBusiness.GetDetail(game.Genre);

Console.WriteLine("Genre: " + detail.Genre);

Platform platform = new Platform();

platform = gameBusiness.GetPlatform(game.Console);

Console.WriteLine("Console: " + platform.Console);

Publisher publisher = new Publisher();

publisher = gameBusiness.GetPublisher(game.Publisher);

Console.WriteLine("Publisher: "+ publisher.PubName);

Console.WriteLine("Year: "+ publisher.Year);

Console.WriteLine(new string('-', 40));

}

else { Console.WriteLine("There is no such game!"); }

}

private void Remove()

{

Console.WriteLine("Enter ID of game you wish to remove: ");

int id=int.Parse(Console.ReadLine());

gameBusiness.Delete(id);

Console.WriteLine("Done.");

}

//Developer

private void AddDev()

{

Developer developer = new Developer();

Console.WriteLine("Developer name:");

developer.DevName = Console.ReadLine();

Console.WriteLine("Developer adress:");

developer.DevAdress = Console.ReadLine();

gameBusiness.AddDev(developer);

}

private void ListDev()

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 18) + "Developers" + new string(' ', 18));

Console.WriteLine(new string('-', 40));

var developers = gameBusiness.GetDevelopers();

foreach(var item in developers)

{

Console.WriteLine("{0} {1} {2}", item.ID,item.DevName,item.DevAdress);

}

}

private void UpdateDev()

{

Console.WriteLine("Enter ID of developer you wish to update:");

var id = int.Parse(Console.ReadLine());

Developer developer = gameBusiness.GetDeveloper(id);

if (developer != null)

{

Console.WriteLine("Enter new developer name:");

developer.DevName = Console.ReadLine();

Console.WriteLine("Enter new developer adress:");

developer.DevAdress = Console.ReadLine();

gameBusiness.UpdateDev(developer);

}

else { Console.WriteLine("Can not update what does not exist!"); }

}

private void RemoveDev()

{

Console.WriteLine("Enter ID of developer you wish to remove: ");

int id = int.Parse(Console.ReadLine());

gameBusiness.DeleteDev(id);

Console.WriteLine("Done.");

}

//Detail

private void AddDet()

{

Detail detail = new Detail();

Console.WriteLine("Genre: ");

detail.Genre = Console.ReadLine();

Console.WriteLine("Rating: ");

detail.Rating = Console.ReadLine();

gameBusiness.AddDetail(detail);

}

private void ListDetails()

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 18) + "Details" + new string(' ', 18));

Console.WriteLine(new string('-', 40));

var details = gameBusiness.GetDetails();

foreach (var item in details)

{

Console.WriteLine("{0} {1} {2}", item.ID, item.Genre, item.Rating);

}

}

private void UpdateDet()

{

Console.WriteLine("Enter id of detail you wish to update:");

var id = int.Parse(Console.ReadLine());

Detail detail = gameBusiness.GetDetail(id);

if (detail != null)

{

Console.WriteLine("Enter new genre:");

detail.Genre = Console.ReadLine();

Console.WriteLine("Enter new rating:");

detail.Rating = Console.ReadLine();

gameBusiness.UpdateDet(detail);

}

else { Console.WriteLine("Can not update what does not exist!"); }

}

private void RemoveDet()

{

Console.WriteLine("Enter ID of detail you wish to remove: ");

int id = int.Parse(Console.ReadLine());

gameBusiness.DeleteDetail(id);

Console.WriteLine("Done.");

}

//Platform

private void AddPlat()

{

Platform platform = new Platform();

Console.WriteLine("Console name: ");

platform.Console = Console.ReadLine();

Console.WriteLine("Price: ");

platform.ConsolePrice = decimal.Parse(Console.ReadLine());

gameBusiness.AddPlatform(platform);

}

private void ListPlat()

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 18) + "Platforms" + new string(' ', 18));

Console.WriteLine(new string('-', 40));

var platforms = gameBusiness.GetPlatforms();

foreach (var item in platforms)

{

Console.WriteLine("{0} {1} {2}", item.ID, item.Console, item.ConsolePrice);

}

}

private void UpdatePlat()

{

Console.WriteLine("Enter id of platform you wish to update:");

var id = int.Parse(Console.ReadLine());

Platform platform = gameBusiness.GetPlatform(id);

if (platform != null)

{

Console.WriteLine("Enter new console name:");

platform.Console = Console.ReadLine();

Console.WriteLine("Enter new console price:");

platform.ConsolePrice = decimal.Parse(Console.ReadLine());

gameBusiness.UpdatePlat(platform);

}

else { Console.WriteLine("Can not update what does not exist!"); }

}

private void RemovePlat()

{

Console.WriteLine("Enter ID of platform you wish to remove: ");

int id = int.Parse(Console.ReadLine());

gameBusiness.DeletePlatform(id);

Console.WriteLine("Done.");

}

//Publisher

private void AddPub()

{

Publisher publisher = new Publisher();

Console.WriteLine("Publisher name: ");

publisher.PubName = Console.ReadLine();

Console.WriteLine("Year: ");

publisher.Year = int.Parse(Console.ReadLine());

gameBusiness.AddPublisher(publisher);

}

private void ListPub()

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 18) + "Publishers" + new string(' ', 18));

Console.WriteLine(new string('-', 40));

var publishers = gameBusiness.GetPublishers();

foreach (var item in publishers)

{

Console.WriteLine("{0} {1} {2}", item.ID, item.PubName, item.Year);

}

}

private void UpdatePub()

{

Console.WriteLine("Enter id of publisher you wish to update:");

var id = int.Parse(Console.ReadLine());

Publisher publisher = gameBusiness.GetPublisher(id);

if (publisher != null)

{

Console.WriteLine("Enter new publisher name:");

publisher.PubName = Console.ReadLine();

Console.WriteLine("Enter new year of publication:");

publisher.Year = int.Parse(Console.ReadLine());

gameBusiness.UpdatePub(publisher);

}

else { Console.WriteLine("Can not update what does not exist!"); }

}

private void RemovePub()

{

Console.WriteLine("Enter ID of publisher you wish to remove: ");

int id = int.Parse(Console.ReadLine());

gameBusiness.DeletePublisher(id);

Console.WriteLine("Done.");

}

private void GameMenu()

{

int lol1 = -1;

do

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 10) + "Game Menu Options" + new string(' ', 16));

Console.WriteLine(new string('-', 40));

Console.WriteLine("1.Display all games");

Console.WriteLine("2.Add a game");

Console.WriteLine("3.Update game info");

Console.WriteLine("4.Find game by id");

Console.WriteLine("5.Remove game");

Console.WriteLine("6.Back");

Console.Write("Your choice: ");

lol1 = int.Parse(Console.ReadLine());

switch (lol1)

{

case 1:

ListAll();

break;

case 2:

Add();

break;

case 3:

Update();

break;

case 4:

Search();

break;

case 5:

Remove();

break;

default:

break;

}

} while (lol1 != 6);

}

private void DevMenu()

{

int lol2 = -1;

do

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 10) + "Developer Menu Options" + new string(' ', 16));

Console.WriteLine(new string('-', 40));

Console.WriteLine("1.List all developers");

Console.WriteLine("2.Add developer");

Console.WriteLine("3.Remove developer");

Console.WriteLine("4.Update developer");

Console.WriteLine("5.Back");

Console.Write("Your choice: ");

lol2 = int.Parse(Console.ReadLine());

switch(lol2)

{

case 1:

ListDev();

break;

case 2:

AddDev();

break;

case 3:

RemoveDev();

break;

case 4:

UpdateDev();

break;

default:

break;

}

} while (lol2 != 5);

}

private void DetMenu()

{

int lol3 = -1;

do

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 10) + "Details Menu Options" + new string(' ', 16));

Console.WriteLine(new string('-', 40));

Console.WriteLine("1.List all details");

Console.WriteLine("2.Add details");

Console.WriteLine("3.Remove detail");

Console.WriteLine("4.Update detail");

Console.WriteLine("5.Back");

Console.Write("Your choice: ");

lol3 = int.Parse(Console.ReadLine());

switch (lol3)

{

case 1:

ListDetails();

break;

case 2:

AddDet();

break;

case 3:

RemoveDet();

break;

case 4:

UpdateDet();

break;

default:

break;

}

} while (lol3 != 5);

}

private void PlatMenu()

{

int lol4 = -1;

do

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 10) + "Platform Menu Options" + new string(' ', 16));

Console.WriteLine(new string('-', 40));

Console.WriteLine("1.List all platforms");

Console.WriteLine("2.Add platform");

Console.WriteLine("3.Remove platform");

Console.WriteLine("4.Update platform");

Console.WriteLine("5.Back");

Console.Write("Your choice: ");

lol4 = int.Parse(Console.ReadLine());

switch (lol4)

{

case 1:

ListPlat();

break;

case 2:

AddPlat();

break;

case 3:

RemovePlat();

break;

case 4:

UpdatePlat();

break;

default:

break;

}

} while (lol4 != 5);

}

private void PubMenu()

{

int lol5 = -1;

do

{

Console.WriteLine(new string('-', 40));

Console.WriteLine(new string(' ', 10) + "Publisher Menu Options" + new string(' ', 16));

Console.WriteLine(new string('-', 40));

Console.WriteLine("1.List all publishers");

Console.WriteLine("2.Add publisher");

Console.WriteLine("3.Remove publisher");

Console.WriteLine("4.Update publisher");

Console.WriteLine("5.Back");

Console.Write("Your choice: ");

lol5 = int.Parse(Console.ReadLine());

switch (lol5)

{

case 1:

ListPub();

break;

case 2:

AddPub();

break;

case 3:

RemovePub();

break;

case 4:

UpdatePub();

break;

default:

break;

}

} while (lol5 != 5);

}

}

}

* Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using GameLibrary1.Business;

using GameLibrary1.Data;

using GameLibrary1.Presentation;

using GameLibrary1.Data.Models;

namespace GameLibrary1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine(new string('\*', 40));

Console.WriteLine(new string(' ', 10) + "G&Y's Game Store" + new string(' ', 10));

Console.WriteLine(new string('\*', 40));

Display display = new Display();

}

}

}