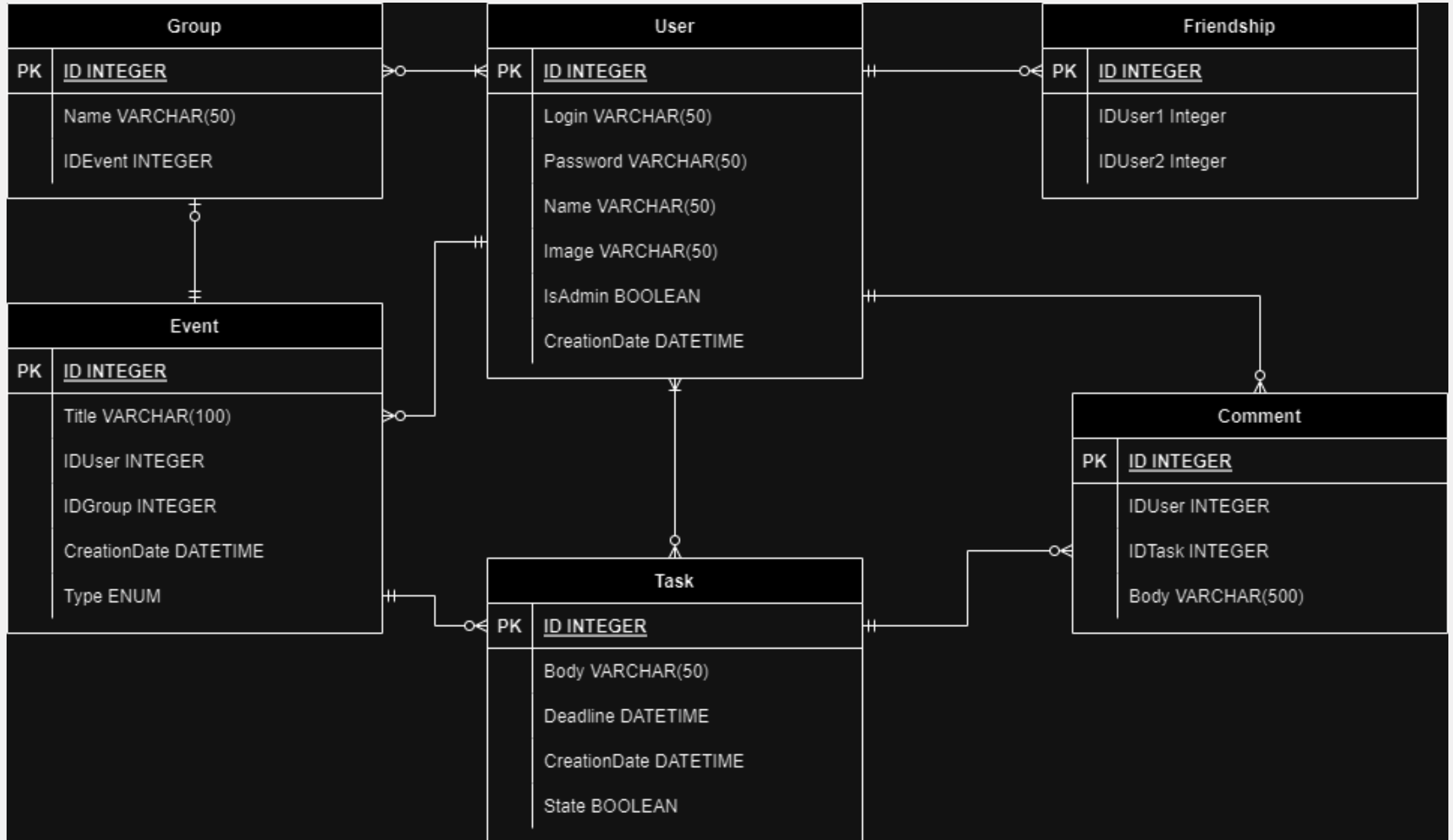# CheckChart

Projekt z Przedmiotu

Tworzenie aplikacji internetowych i bazodanowych

ETAP 2 - WebAPI

Skład grupy projektowej:

Karol Kołodziejczyk,
Jakub Woźniak,
Wiktor Kobryń IPpp30

Diagram ERD Aplikacji

Przykładowe klasy Modelu


```csharp
public enum EventType
{
    chart, task, activity
}

[Table("EVENTS")]
Odwołania: 10
public class Event : IEntityTypeConfiguration<Event>
{
    [Key, Column("ID")]
    Odwołania: 2
    public int Id { get; set; }

    [Required, MaxLength(100)]
    Odwołania: 4
    public string Title { get; set; }

    Odwołania: 5
    public int UserId { get; set; }
    [ForeignKey(nameof(UserId))]
    1 odwołanie
    public User User { get; set; }

    Odwołania: 8
    public int? GroupId { get; set; }
    [ForeignKey(nameof(GroupId))]
    Odwołania: 2
    public Group Group { get; set; }

    Odwołania: 3
    public DateTime CreationDate { get; set; }

    Odwołania: 3
    public EventType Type { get; set; }

    1 odwołanie
    public ICollection<EventTask> EventTasks { get; set; }

    Odwołania: 0
    public void Configure(EntityTypeBuilder<Event> builder)
    {
        builder.HasOne(x => x.User).WithMany(x => x.Events).OnDelete(DeleteBehavior.Cascade);
        builder.HasOne(x => x.Group).WithOne(x => x.Event).OnDelete(DeleteBehavior.Restrict);
    }
}
```

Event

```csharp
namespace Model
{
    [Table("EVENT_TASKS")]

    public class EventTask : IEntityTypeConfiguration<EventTask>
    {
        [Key, Column("ID")]
        public int Id { get; set; }

        [ForeignKey(nameof(EventId))]
        public int? EventId { get; set; }

        public Event Event { get; set; }

        [Required, MaxLength(200)]
        public string Body { get; set; }

        public DateTime Deadline { get; set; }

        public DateTime CreationDate { get; set; }

        public bool State { get; set; }

        1 odwołanie
        public ICollection<Comment> Comments { get; set; }
        Odwołania: 3
        public ICollection<User> Users { get; set; }

        Odwołania: 0
        public void Configure(EntityTypeBuilder<EventTask> builder)
        {
            builder.HasOne(u => u.Event).WithMany(x => x.EventTasks).OnDelete(DeleteBehavior.Restrict);
        }
    }
}
```

EventTask

3

Klasa kontekstu - CheckChartContext

```csharp
using Microsoft.EntityFrameworkCore;
using Model;

namespace DAL
{
    Odwołania: 17
    public class CheckChartContext : DbContext
    {
        Odwołania: 9
        public DbSet<Comment> Comments { get; set; }
        Odwołania: 11
        public DbSet<Event> Events { get; set; }
        Odwołania: 4
        public DbSet<Friendship> Friendships { get; set; }
        Odwołania: 10
        public DbSet<Group> Groups { get; set; }
        Odwołania: 12
        public DbSet<EventTask> EventTasks { get; set; }
        Odwołania: 13
        public DbSet<User> Users { get; set; }

        Odwołania: 0
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {

            optionsBuilder
                .UseSqlServer("Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\Kuba\\Desktop\\ProjektTaiib_KobrynKolodziejczykWozniakMOJE" +
                "\\ProjektTaiib_KobrynKolodziejczykWozniak-Wiktor\\CheckChartDB.mdf;Integrated Security=True;Connect Timeout=30");
        }
    }
}
```

4

klasy i interfejsy BLL – przykład implementacji

```csharp
using BLL.ResponseDTO;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BLL.BLLInterfaces
{

    public interface IEventTaskService
    {

        bool AddEventTask(int userId, EventTaskRequestDTO eventTaskRequest);

        bool AddUserToEventTask(int userId, int eventTaskId);

        bool ChangeEventTask(int eventTaskId, EventTaskRequestDTO eventTaskRequest);

        bool DeleteEventTask(int eventTaskId);

        bool FinishEventTask(int eventTaskId);

        IEnumerable<EventTaskResponseDTO> GetUserEventTasks(int userId);

        IEnumerable<EventTaskResponseDTO> GetEventEventTasks(int eventId);
    }
}
```

```csharp
}
Odwołania: 2
public bool ChangeEventTask(int eventTaskId, EventTaskRequestDTO eventTaskRequest)
{
    var e = db.EventTasks.Find(eventTaskId);
    if (e == null)
        return false;

    if (eventTaskRequest.Deadline < DateTime.Now)
        return false;

    if (eventTaskRequest.Body == null)
        return false;

    e.Deadline = eventTaskRequest.Deadline;
    e.Body = eventTaskRequest.Body;

    db.SaveChanges();
    return true;

}
```

Interfejs BLL - IEventTaskService   EventTaskService

5

Kontrolery – przykład implementacji

```csharp
[HttpDelete("DeleteEvent")]
Odwołania: 0
public void DeleteEvent([FromQuery] int eventId)
{
    eventService.DeleteEvent(eventId);
}

[HttpGet("GetUserEvents")]
Odwołania: 0
public IEnumerable<EventResponseDTO> GetUserEvents([FromQuery] int userId)
{
    return eventService.GetUserEvents(userId);
}

[HttpGet("GetGroupEvent")]
Odwołania: 0
public EventResponseDTO GetGroupEvent([FromQuery] int groupId)
{
    return eventService.GetGroupEvent(groupId);
}
```

```csharp
namespace API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]

    public class EventController : ControllerBase
    {

        readonly IEventService eventService;


        public EventController(IEventService eventService)
        {
            this.eventService = eventService;
        }

        [HttpPost("AddEvent")]

        public void AddEvent([FromQuery] EventRequestDTO eventRequestDTO)
        {
            eventService.AddEvent(eventRequestDTO);
        }


        [HttpPut("ChangeEventTitle")]
        Odwołania: 0
        public void ChangeEventTitle([FromQuery] int eventId, string title)
        {
            eventService.ChangeEventTitle(eventId, title);
        }
    }
```

EventTaskController

6

Metody dostępne w WebAPI - Swagger

API `1.0` `OAS3`

http://localhost:5171/swagger/v1/swagger.json

Comment

POST	/api/Comment/AddComment
PUT	/api/Comment/ChangeComment
DELETE	/api/Comment/DeleteComment
GET	/api/Comment/GetEventTaskComments

Event

POST	/api/Event/AddEvent
PUT	/api/Event/ChangeEventTitle
DELETE	/api/Event/DeleteEvent
GET	/api/Event/GetUserEvents
GET	/api/Event/GetGroupEvent

EventTask

POST	/api/EventTask/AddEventTask
PUT	/api/EventTask/AddUserToEventTask
PUT	/api/EventTask/ChangeEventTask
DELETE	/api/EventTask/DeleteEventTask
PUT	/api/EventTask/FinishEventTask
GET	/api/EventTask/GetUserEventTasks
GET	/api/EventTask/GetEventEventTasks

Group

POST	/api/Group/AddGroup
PUT	/api/Group/ChangeGroupName
DELETE	/api/Group/DeleteGroup
PUT	/api/Group/AddUserToGroup
GET	/api/Group/GetEventGroup
GET	/api/Group/GetUserGroups

User

POST	/api/User/AddFriendship
DELETE	/api/User/DeleteFriendship
GET	/api/User/GetUserFriends
GET	/api/User/GetAllUsers
GET	/api/User/GetUserByName

7

Swagger – test metody

Swagger – test metody

Swagger – test metody

Dziękujemy za uwagę

Skład grupy projektowej:

Karol Kołodziejczyk,
Jakub Woźniak,
Wiktor Kobryń IPpp30