WeSketch Team
Stevica Stojković, 15426
Stefan Stanković, 15382

WeSketch

Online alat za kolaborativno vektorsko crtanje

Sadržaj

Sadržaj 0

Uvod 0

Model podataka 0

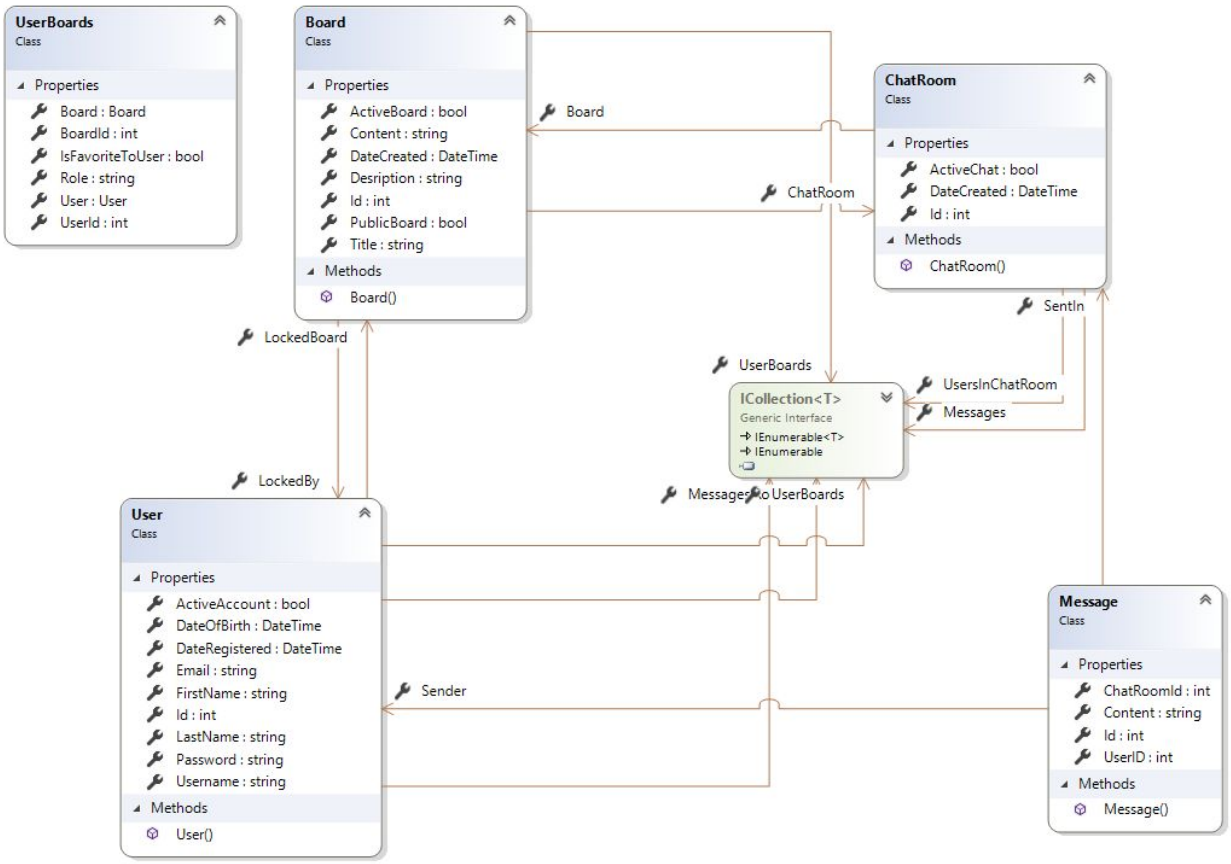
Data Mapper pattern 4

Uvod

Svrha ovog dokumenta je opis modela podataka i modela perzistencije koji se koriste u projektu *WeSketch* .

Model podataka

Na slici je dat dijagram *POCO* klasa klasa koje se pamte u bazi podataka:



Klase koje se pamte u bazi podataka su:

1. Klasa *Board* pamti podatke o tablama(projektima, crtežima):

```
public class Board
{
    public Board()
    {
        UserBoards = new HashSet<UserBoards>();
    }

    public int Id { get; set; }
    public DateTime DateCreated { get; set; }
    public bool ActiveBoard { get; set; }
    public bool PublicBoard { get; set; }
    public string Title { get; set; }
    public string Desription { get; set; }
    public string Content { get; set; }

    public virtual ChatRoom ChatRoom { get; set; }
    public virtual User LockedBy { get; set; }

    public virtual ICollection<UserBoards> UserBoards { get; set; }
}
```

2. Klasa *ChatRoom* pamti podatke o ćaskanjima:

```
public class ChatRoom
{
    public ChatRoom()
    {
        UsersInChatRoom = new HashSet<User>();
        Messages = new HashSet<Message>();
    }

    public int Id { get; set; }
    public bool ActiveChat { get; set; }
    public DateTime DateCreated { get; set; }

    public virtual Board Board { get; set; }

    public virtual ICollection<User> UsersInChatRoom { get; set; }
    public ICollection<Message> Messages { get; set; }
}
```

3. Klasa *Message* pamti podatke o poslatim porukama:

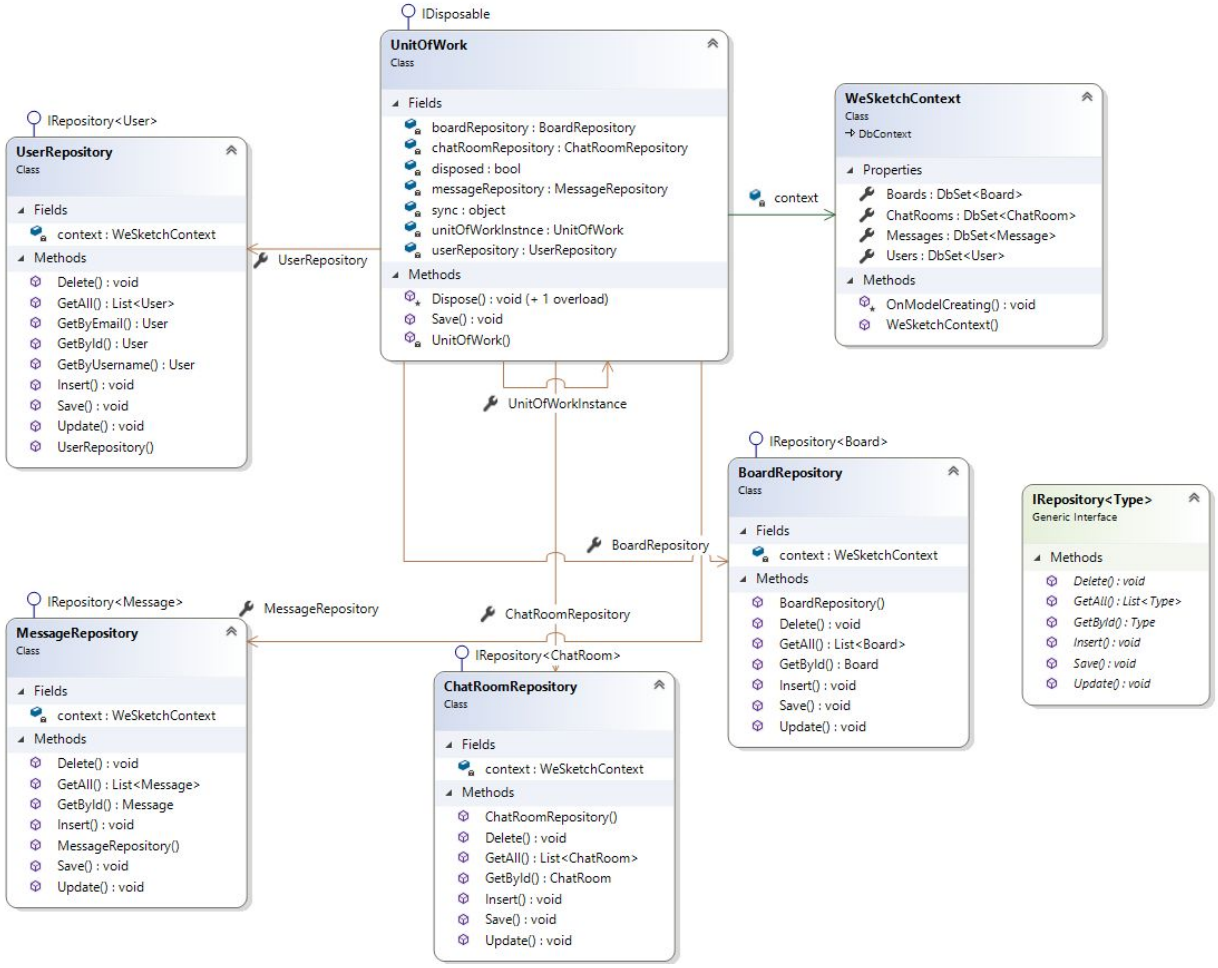
```
public class Message
{
    public Message()
    {

    }

    public int Id { get; set; }
    public string Content { get; set; }

    public int UserID { get; set; }
    public int ChatRoomId { get; set; }

    public virtual ChatRoom SentIn { get; set; }
    public virtual User Sender { get; set; }
}
```

4. Klasa *User* pamti podatke o korisnicima:

```
public class User
{
    public User()
    {
        ChatRooms = new HashSet<ChatRoom>();
        Messages = new HashSet<Message>();
        UserBoards = new HashSet<UserBoards>();
    }

    public int Id { get; set; }
    public string Username { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DateRegistered { get; set; }
    public bool ActiveAccount { get; set; }
    public DateTime DateOfBirth { get; set; }

    public virtual Board LockedBoard { get; set; }

    public virtual ICollection<ChatRoom> ChatRooms { get; set; }
    public ICollection<Message> Messages { get; set; }

    public virtual ICollection<UserBoards> UserBoards { get; set; }
}
```

Data Mapper pattern

Sloj podataka je modelovan po *Data Mapper* obrascu. Dijagram klasa sloja podataka dat je na slici:



Funkcije za rad sa bazom podataka date su u generičkom interfejsu *IRepository*:

```
public interface IRepository<Type>
{
    List<Type> GetAll();
    Type GetById(int id);
    void Insert(Type typeInstance);
    void Delete(int id);
    void Update(Type typeInstance);
    void Save();
}
```

Repository klase implementiraju ovaj interfejs za konkretnu *POCO* klasu. Primer klase koja implementira interfejs:

```
public class BoardRepository : IRepository<Board>
{
    private WeSketchContext context;

    public BoardRepository(WeSketchContext context)...
    public void Delete(int id)...
    public List<Board> GetAll()...
    public Board GetById(int id)...
    public void Insert(Board typeInstance)...
    public void Save()...
    public void Update(Board typeInstance)...
}
```

Primer klase koja implementira interfejs i ima dodatne funkcije koje komuniciraju sa bazom:

```
public class UserRepository : IRepository<User>
{
    private WeSketchContext context;

    public UserRepository(WeSketchContext context)...

    public void Delete(int id)...
    public List<User> GetAll()...
    public User GetById(int id)...
    public void Insert(User typeInstance)...
    public void Save()...
    public void Update(User typeInstance)...

    public User GetByUsername(string username)...
    public User GetByEmail(string email)...
}
```

Sve *Repository* klase se instanciraju u klasi *UnitOfWork*:

```
public class UnitOfWork : IDisposable
{
    Singleton implementation
    #region DB context
    private WeSketchContext context = new WeSketchContext();
    #endregion
    #region Repository variables
    private BoardRepository boardRepository;
    private ChatRoomRepository chatRoomRepository;
    private MessageRepository messageRepository;
    private UserRepository userRepository;
    #endregion
    #region Repository properties
    public BoardRepository BoardRepository
    {
        get
        {
            return this.boardRepository ?? new BoardRepository(context);
        }
    }
```

Ova klasa obezbeđuje da svi *Repository* dele isti Entity Framework context. Sve klase koje zahtevaju pristup bazi se prvo obraćaju ovoj klasa a onda se preko određenog *Repository*-a učitava potrebni *POCO*.

Mapiranja su obavljena u klasi *WeSketchContext*:

```
public WeSketchContext() ...

DbSets

protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    modelBuilder.HasDefaultSchema("WeSketch");

    #region PrimaryKeys
    modelBuilder.Entity<Board>().HasKey<int>(s => s.Id);
    modelBuilder.Entity<ChatRoom>().HasKey<int>(s => s.Id);
    modelBuilder.Entity<Message>().HasKey<int>(s => s.Id);
    modelBuilder.Entity<User>().HasKey<int>(s => s.Id);
    modelBuilder.Entity<UserBoards>().HasKey(s => new {s.UserId, s.BoardId});

    Auto increment
    #endregion
    Columns
    Foreign Keys
}
```