

# Введение в SeaBattle

SeaBattle - это многопользовательская онлайн-игра "Морской бой", разработанная с использованием современных технологий.

## Особенности проекта

- Многопользовательский режим с использованием SignalR для real-time коммуникации
- Современный веб-интерфейс
- Система рейтинга игроков
- История игр и статистика
- REST API для интеграции

## Технологии

- **Backend:** ASP.NET Core 8.0
- **Real-time коммуникации:** SignalR
- **База данных:** PostgreSQL с Entity Framework Core
- **API документация:** Swagger/OpenAPI
- **Контейнеризация:** Docker

## Структура проекта

- **Controllers/** - REST API контроллеры
- **Models/** - Модели данных с XML-документацией
- **Services/** - Бизнес-логика
- **Hubs/** - SignalR хабы для real-time взаимодействия
- **Data/** - Контекст базы данных и миграции

## Документация моделей

В проекте используются следующие основные модели:

1. **Game** - основная модель игры с полями для состояния игры, игроков и игрового поля
2. **GameHistory** - модель для хранения истории игр
3. **GameState** - перечисление состояний игры
4. **Position** - модель позиции на игровом поле
5. **ShotResult** - перечисление результатов выстрела
6. **ShotResultResponse** - модель ответа на выстрел
7. **User** - модель пользователя
8. **PlayerRanking** - модель рейтинга игрока
9. **LobbyInfo** - модель информации о лобби
10. **CellState** - перечисление состояний клетки

Все модели содержат подробную XML-документацию на русском языке.

# Начало работы с SeaBattle

## Требования к системе

Для работы с проектом необходимо:

- .NET 8.0 SDK
- PostgreSQL 15
- Docker (опционально)
- Visual Studio 2022 или VS Code с C# расширением

## Установка и настройка

1. Клонировать репозиторий:

```
git clone https://github.com/Mug1vara97/SeaBattle
cd SeaBattle/Server
```

2. Восстановите зависимости:

```
dotnet restore
```

3. Настройте строку подключения к базе данных в `appsettings.json`:

```
{
  "ConnectionStrings": {
    "DefaultConnection":
      "Host=localhost;Database=seabattle;Username=your_username;Password=your_password"
  }
}
```

4. Примените миграции базы данных:

```
dotnet ef database update
```

5. Запустите приложение:

```
dotnet run
```

## Запуск через Docker

1. Соберите Docker образ:

```
docker build -t seabattle .
```

2. Запустите контейнер:

```
docker run -p 5000:80 seabattle
```

## Документация API

Документация API доступна через Swagger UI после запуска приложения по адресу:

<http://localhost:5000/swagger>

## Структура решения

- **SeaBattle/** - основной проект
  - **Controllers/** - REST API контроллеры
  - **Models/** - модели данных с XML-документацией
  - **Services/** - бизнес-логика
  - **Hubs/** - SignalR хабы
  - **Data/** - работа с базой данных
- **GameServiceTests/** - модульные тесты

## XML-документация

Все модели в проекте содержат подробную XML-документацию на русском языке. Документация включает:

- Описание назначения классов
- Описание свойств
- Описание методов и их параметров
- Примеры использования (где применимо)

# Namespace SeaBattle.Controllers

## Classes

### [AuthController](#)

Контроллер для управления аутентификацией и регистрацией пользователей

### [CreateGameRequest](#)

Модель запроса для создания новой игры

### [GameController](#)

Контроллер для управления игровым процессом морского боя

### [JoinGameRequest](#)

Модель запроса для присоединения к игре

### [LoginRequest](#)

Модель запроса для входа в систему

### [ReadyRequest](#)

Модель запроса для установки готовности игрока

### [RegisterRequest](#)

Модель запроса для регистрации нового пользователя

### [ShotRequest](#)

Модель запроса для выполнения выстрела

# Namespace SeaBattle.Models

## Classes

### [Game](#)

Представляет игровую сессию морского боя

### [GameHistory](#)

Представляет историю игры морского боя

### [GameStateInfo](#)

### [GameStateManager](#)

### [LobbyInfo](#)

Представляет информацию о игровом лобби

### [PlayerRanking](#)

Представляет рейтинг игрока в таблице лидеров

### [Position](#)

Представляет позицию на игровом поле

### [ShotResultResponse](#)

Представляет ответ на выполненный выстрел

### [User](#)

Представляет пользователя в системе

## Enums

### [CellState](#)

Перечисление возможных состояний клетки игрового поля

### [GameState](#)

Перечисление возможных состояний игры

### [GameStateManager.GameState](#)

### [GameStatus](#)

### [ShotResult](#)

Перечисление возможных результатов выстрела

# Class Game

Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll








Представляет игровую сессию морского боя

```
public class Game
```

## Inheritance

[object](#)  ← Game

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

# Properties

## CreatorBoard

Игровое поле создателя игры

```
[JsonConverter(typeof(BoardConverter))]  
public CellState[,]? CreatorBoard { get; set; }
```

## Property Value

[CellState](#)[,]

## CreatorBoardSet

Флаг, указывающий установлены ли корабли создателем

```
public bool CreatorBoardSet { get; set; }
```

## Property Value

[bool](#)

## CreatorName

Имя создателя игры

```
public string CreatorName { get; set; }
```

Property Value

[string](#)

## CreatorReady

Флаг готовности создателя игры

```
public bool CreatorReady { get; set; }
```

Property Value

[bool](#)

## CreatorShots

Список выстрелов создателя игры

```
public List<Position> CreatorShots { get; set; }
```

Property Value

[List](#) <[Position](#)>

## CurrentTurn

Имя игрока, чей ход сейчас



```
public string? CurrentTurn { get; set; }
```

Property Value

[string](#)↗

## Id

Уникальный идентификатор игры

```
public string Id { get; set; }
```

Property Value

[string](#)↗

## IsGameEnded

```
public bool IsGameEnded { get; }
```

Property Value

[bool](#)↗

## IsOpenLobby

Флаг, указывающий является ли лобби открытым для присоединения

```
public bool IsOpenLobby { get; set; }
```

Property Value

[bool](#)↗

# JoinerBoard

Игровое поле присоединившегося игрока

```
[JsonConverter(typeof(BoardConverter))]  
public CellState[,]? JoinerBoard { get; set; }
```

Property Value

[CellState\[,\]](#)

# JoinerBoardSet

Флаг, указывающий установлены ли корабли присоединившимся игроком

```
public bool JoinerBoardSet { get; set; }
```

Property Value

[bool](#)

# JoinerName

Имя присоединившегося игрока

```
public string? JoinerName { get; set; }
```

Property Value

[string](#)

# JoinerReady

Флаг готовности присоединившегося игрока

```
public bool JoinerReady { get; set; }
```

Property Value

[bool](#)

## JoinerShots

Список выстрелов присоединившегося игрока

```
public List<Position> JoinerShots { get; set; }
```

Property Value

[List](#) <[Position](#)>

## State

Текущее состояние игры

```
public GameState State { get; set; }
```

Property Value

[GameState](#)

## Winner

Имя победителя игры

```
public string? Winner { get; set; }
```

Property Value

[string](#)

# Class GameHistory

Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll








Представляет историю игры морского боя

```
public class GameHistory
```

## Inheritance

[object](#)  ← GameHistory

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

## Constructors

### GameHistory()

```
public GameHistory()
```

## Properties

### CreatorName

Имя создателя игры

```
public string CreatorName { get; set; }
```

### Property Value

[string](#) 

## DurationMinutes

Продолжительность игры в минутах

```
public double DurationMinutes { get; }
```

Property Value

[double](#)

## GameFinishedAt

Дата и время завершения игры

```
public DateTime GameFinishedAt { get; set; }
```

Property Value

[DateTime](#)

## GameId

Идентификатор игры

```
[Required]  
public string GameId { get; set; }
```

Property Value

[string](#)

## Id

Уникальный идентификатор записи истории

```
[Key]
```

```
public Guid Id { get; set; }
```

Property Value

[Guid](#)

## JoinerName

Имя присоединившегося игрока

```
public string JoinerName { get; set; }
```

Property Value

[string](#)

## OpponentUsername

```
public string? OpponentUsername { get; set; }
```

Property Value

[string](#)

## PlayerUsername

```
[Required]
```

```
public string PlayerUsername { get; set; }
```

Property Value

[string](#)

## Result

[Required]

```
public string Result { get; set; }
```

Property Value

[string](#)

## StartedAt

Дата и время начала игры

```
public DateTime StartedAt { get; set; }
```

Property Value

[DateTime](#)

## TotalMoves

Количество ходов в игре

```
public int TotalMoves { get; set; }
```

Property Value

[int](#)

## Winner

Имя победителя игры

```
public string Winner { get; set; }
```

Property Value





# Enum GameState

Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll

Перечисление возможных состояний игры

```
public enum GameState
```

## Fields

Cancelled = 6

Игра отменена

Finished = 5

Игра завершена

InProgress = 4

Игра в процессе

Unknown = 0

Неизвестное состояние

WaitingForOpponent = 1

Ожидание второго игрока

WaitingForReady = 3

Ожидание готовности игроков

WaitingForShips = 2

Ожидание расстановки кораблей

# Class Position

Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll








Представляет позицию на игровом поле

```
public class Position
```

## Inheritance

[object](#)  ← Position

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

# Properties

## Col

Координата столбца (0-9)

```
public int Col { get; set; }
```

Property Value

[int](#) 

## IsHit

```
public bool IsHit { get; set; }
```

Property Value

[bool](#) 

# Row

Координата строки (0-9)

```
public int Row { get; set; }
```

Property Value

[int](#)

# Enum ShotResult

Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll

Перечисление возможных результатов выстрела

```
public enum ShotResult
```

## Fields

**Destroyed = 3**

Корабль уничтожен

**Error = 0**

Пропих

**Hit = 2**

Корабль уничтожен

**Miss = 1**

Попадание

**Win = 4**

Победа (последний корабль уничтожен)

# Class ShotResultResponse

Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll








Представляет ответ на выполненный выстрел

```
public class ShotResultResponse
```

## Inheritance

[object](#)  ← ShotResultResponse

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

## Properties

### CurrentTurn

Имя игрока, чей следующий ход

```
public string? CurrentTurn { get; set; }
```

Property Value

[string](#) 

### ErrorMessage

Сообщение об ошибке, если она произошла

```
public string? ErrorMessage { get; set; }
```

Property Value

[string](#)

## GameState

Текущее состояние игры

```
public GameState GameState { get; set; }
```

Property Value

[GameState](#)

## IsHit

Флаг успешности попадания

```
public bool IsHit { get; set; }
```

Property Value

[bool](#)

## Position

Позиция выстрела

```
public Position Position { get; set; }
```

Property Value

[Position](#)

## Result

Результат выстрела

```
public ShotResult Result { get; set; }
```

Property Value

[ShotResult](#)

## Success

Флаг успешности выполнения операции

```
public bool Success { get; set; }
```

Property Value

[bool](#)

# Class User

Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll








Представляет пользователя в системе

```
public class User
```

## Inheritance

[object](#)  ← User

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

# Properties

## CreatedAt

```
public DateTime CreatedAt { get; set; }
```

Property Value

[DateTime](#) 

## Id

Уникальный идентификатор пользователя

```
public int Id { get; set; }
```

Property Value

[int](#) 



## Losses

Количество поражений

```
public int Losses { get; set; }
```

Property Value

[int](#)

## PasswordHash

Хэш пароля пользователя

```
[Required]  
public required string PasswordHash { get; set; }
```

Property Value

[string](#)

## TotalGames

Общее количество игр

```
public int TotalGames { get; }
```

Property Value

[int](#)

## Username

Имя пользователя

```
[Required]  
[StringLength(50)]
```

```
public required string Username { get; set; }
```

Property Value

[string](#)↗

## Wins

Количество побед

```
public int Wins { get; set; }
```

Property Value

[int](#)↗

# Class PlayerRanking


Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll








Представляет рейтинг игрока в таблице лидеров

```
public class PlayerRanking
```

## Inheritance

[object](#)  ← PlayerRanking

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

## Properties

### Losses

Количество поражений

```
public int Losses { get; set; }
```

Property Value

[int](#) 

### PlayerUsername

Имя игрока

```
[Key]  
public string PlayerUsername { get; set; }
```

Property Value

[string](#)

## Rating

```
public int Rating { get; set; }
```

Property Value

[int](#)

## TotalGames

Общее количество игр

```
public int TotalGames { get; }
```

Property Value

[int](#)

## WinRate

Процент побед

```
public double WinRate { get; }
```

Property Value

[double](#)

## Wins

Количество побед

```
public int Wins { get; set; }
```

Property Value

[int](#)

# Class LobbyInfo

Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll








Представляет информацию о игровом лобби

```
public class LobbyInfo
```

## Inheritance

[object](#)  ← LobbyInfo

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#) 

## Properties

### CreatedAt

Дата и время создания лобби

```
public DateTime CreatedAt { get; set; }
```

Property Value

[DateTime](#) 

### CreatorName

Имя создателя лобби

```
public string CreatorName { get; set; }
```

Property Value

[string](#)

## GameId

Идентификатор игры

```
public string GameId { get; set; }
```

Property Value

[string](#)

## IsOpenLobby

Флаг, указывающий является ли лобби открытым для присоединения

```
public bool IsOpenLobby { get; set; }
```

Property Value

[bool](#)

# Enum CellState

Namespace: [SeaBattle.Models](#)

Assembly: SeaBattle.dll

Перечисление возможных состояний клетки игрового поля

```
public enum CellState
```

## Fields

`Empty = 0`

Пустая клетка

`Hit = 2`

Клетка с попаданием

`Miss = 3`

Клетка с промахом

`Ship = 1`

Клетка с кораблем