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

Class AuthController

Namespace: SeaBattle.Controllers

Assembly: SeaBattle.dll

Контроллер для управления аутентификацией и регистрацией пользователей

```
[ApiController]
[Route("api/[controller]")]
public class AuthController : ControllerBase
```

Inheritance

object ← ControllerBase ← AuthController

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) ,
ControllerBase.Content(string) , ControllerBase.Content(string, string) ,
ControllerBase.Content(string, string, Encoding) ,
ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() ,
ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) ,
ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) ,
ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) ,
ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) ,
ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() ,
ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) ,
ControllerBase.RedirectToAction(string, string) ,
ControllerBase.RedirectToAction(string, string, object) ,
ControllerBase.RedirectToAction(string, string, string) ,
ControllerBase.RedirectToAction(string, string, object, string) ,
ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToActionPermanent(string) ,
ControllerBase.RedirectToActionPermanent(string, object) ,
ControllerBase.RedirectToActionPermanent(string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, object) ,
ControllerBase.RedirectToActionPermanent(string, string, object, string) ,
ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,
ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,

ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,

ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,
ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) ,
ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) ,
ControllerBase.AcceptedAtAction(string, object) ,
ControllerBase.AcceptedAtAction(string, string, object) ,
ControllerBase.AcceptedAtAction(string, object, object) ,
ControllerBase.AcceptedAtAction(string, string, object, object) ,
ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) ,
ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) ,
ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() ,
ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) ,
ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() ,
ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) ,
ControllerBase.Forbid(AuthenticationProperties, params string[]) ,
ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() ,
ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) ,
ControllerBase.SignOut(AuthenticationProperties, params string[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel) ,

ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync(object, Type, string) ,
ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) ,
ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) ,
ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response ,
ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext ,
ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url ,
ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User ,
ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) ,
object.GetHashCode() , object.GetType() , object.MemberwiseClone() ,
object.ReferenceEquals(object, object) , object.ToString()

Constructors

AuthController(IUserService)

Инициализирует новый экземпляр контроллера аутентификации

```
public AuthController(IUserService userService)
```

Parameters

`userService` IUserService

Сервис для работы с пользователями

Methods

Login(LoginRequest)

Выполняет вход пользователя в систему

```
[HttpPost("login")]
public Task<IActionResult> Login(LoginRequest request)
```

Parameters

`request` LoginRequest

Данные для входа (имя пользователя и пароль)

Returns

Task<IActionResult>

Информация о пользователе при успешном входе

Register(RegisterRequest)

Регистрирует нового пользователя в системе

```
[HttpPost("register")]
public Task<IActionResult> Register(RegisterRequest request)
```

Parameters

`request` RegisterRequest

Данные для регистрации (имя пользователя и пароль)

Returns

Task<IActionResult>

Информация о созданном пользователе

Class CreateGameRequest

Namespace: SeaBattle.Controllers

Assembly: SeaBattle.dll

Модель запроса для создания новой игры

```
public class CreateGameRequest
```

Inheritance

object ← CreateGameRequest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

CreatorName

Имя создателя игры

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

Property Value

string

IsOpenLobby

Флаг, указывающий является ли лобби открытым для присоединения

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

Property Value

bool

Class GameController

Namespace: SeaBattle.Controllers

Assembly: SeaBattle.dll

Контроллер для управления игровым процессом морского боя

```
[ApiController]
[Route("api/[controller]")]
public class GameController : ControllerBase
```

Inheritance

object ← ControllerBase ← GameController

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) ,
ControllerBase.Content(string) , ControllerBase.Content(string, string) ,
ControllerBase.Content(string, string, Encoding) ,
ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() ,
ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) ,
ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) ,
ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) ,
ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) ,
ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() ,
ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) ,
ControllerBase.RedirectToAction(string, string) ,
ControllerBase.RedirectToAction(string, string, object) ,
ControllerBase.RedirectToAction(string, string, string) ,
ControllerBase.RedirectToAction(string, string, object, string) ,
ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToActionPermanent(string) ,
ControllerBase.RedirectToActionPermanent(string, object) ,
ControllerBase.RedirectToActionPermanent(string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, string) ,
ControllerBase.RedirectToActionPermanent(string, string, object) ,
ControllerBase.RedirectToActionPermanent(string, string, object, string) ,
ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,
ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,

ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,

ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,
ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) ,
ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) ,
ControllerBase.AcceptedAtAction(string, object) ,
ControllerBase.AcceptedAtAction(string, string, object) ,
ControllerBase.AcceptedAtAction(string, object, object) ,
ControllerBase.AcceptedAtAction(string, string, object, object) ,
ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) ,
ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) ,
ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() ,
ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) ,
ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() ,
ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) ,
ControllerBase.Forbid(AuthenticationProperties, params string[]) ,
ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() ,
ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) ,
ControllerBase.SignOut(AuthenticationProperties, params string[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel) ,

ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync(object, Type, string) ,
ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) ,
ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) ,
ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response ,
ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext ,
ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url ,
ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User ,
ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) ,
object.GetHashCode() , object.GetType() , object.MemberwiseClone() ,
object.ReferenceEquals(object, object) , object.ToString()

Constructors

GameController(IGameService, IHubContext<LobbyHub>)

Инициализирует новый экземпляр контроллера игры

```
public GameController(IGameService gameService, IHubContext<LobbyHub> lobbyHub)
```

Parameters

`gameService` IGameService

Сервис для управления игровым процессом

`lobbyHub` IHubContext<LobbyHub>

Хаб для работы с лобби

Methods

CreateGame(CreateGameRequest)

Создает новую игру

```
[HttpPost("create")]
public Task<ActionResult<Game>> CreateGame(CreateGameRequest request)
```

Parameters

`request` CreateGameRequest

Данные для создания игры

Returns

Task<ActionResult<Game>>

Информация о созданной игре

GetLeaderboard(int)

Получает таблицу лидеров

```
[HttpGet("leaderboard")]
public Task<ActionResult<IEnumerable<PlayerRanking>>> GetLeaderboard(int count = 10)
```

Parameters

`count` int

Количество записей для отображения

Returns

Task<ActionResult<IEnumerable<PlayerRanking>>>

Список лучших игроков

GetOpenLobbies()

Получает список открытых лобби

```
[HttpGet("lobbies")]
public Task<ActionResult<IEnumerable<LobbyInfo>>> GetOpenLobbies()
```

Returns

Task<ActionResult<IEnumerable<LobbyInfo>>>

Список доступных игровых лобби

JoinGame(JoinGameRequest)

Присоединяет игрока к существующей игре

```
[HttpPost("join")]
public Task<ActionResult<Game>> JoinGame(JoinGameRequest request)
```

Parameters

`request` JoinGameRequest

Данные для присоединения к игре

Returns

Task<ActionResult<Game>>

Информация об игре

MakeShot(ShotRequest)

Выполняет выстрел по указанной позиции

```
[HttpPost("shot")]
public Task<ActionResult<ShotResult>> MakeShot(ShotRequest request)
```

Parameters

`request` ShotRequest

Данные о выстреле

Returns

Task<ActionResult<ShotResult>>

Результат выстрела

SetReady(ReadyRequest)

Устанавливает готовность игрока к началу игры

```
[HttpPost("ready")]
public Task<ActionResult<Game>> SetReady(ReadyRequest request)
```

Parameters

`request` ReadyRequest

Данные о готовности игрока

Returns

Task<ActionResult<Game>>

Обновленная информация об игре

Class JoinGameRequest

Namespace: SeaBattle.Controllers

Assembly: SeaBattle.dll

Модель запроса для присоединения к игре

```
public class JoinGameRequest
```

Inheritance

object ← JoinGameRequest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

GameId

Идентификатор игры

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

Property Value

string

OpponentName

Имя присоединяющегося игрока

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

Property Value

string

Class LoginRequest

Namespace: SeaBattle.Controllers

Assembly: SeaBattle.dll

Модель запроса для входа в систему

```
public class LoginRequest
```

Inheritance

object ← LoginRequest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Password

Пароль пользователя

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

Property Value

string

Username

Имя пользователя

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

Property Value

string

Class ReadyRequest

Namespace: SeaBattle.Controllers

Assembly: SeaBattle.dll

Модель запроса для установки готовности игрока

```
public class ReadyRequest
```

Inheritance

object ← ReadyRequest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

GameId

Идентификатор игры

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

Property Value

string

PlayerName

Имя игрока

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

Property Value

string

Class RegisterRequest

Namespace: SeaBattle.Controllers

Assembly: SeaBattle.dll

Модель запроса для регистрации нового пользователя

```
public class RegisterRequest
```

Inheritance

object ← RegisterRequest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Password

Пароль пользователя

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

Property Value

string

Username

Имя пользователя

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

Property Value

string

Class ShotRequest

Namespace: SeaBattle.Controllers

Assembly: SeaBattle.dll

Модель запроса для выполнения выстрела

```
public class ShotRequest
```

Inheritance

object ← ShotRequest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

GameId

Идентификатор игры

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

Property Value

string

PlayerName

Имя игрока, выполняющего выстрел

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

Property Value

string

Position

Позиция выстрела на игровом поле

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

Property Value

Position

Namespace SeaBattle.Data

Classes

ApplicationDbContext

DesignTimeDbContextFactory

Class ApplicationDbContext

Namespace: SeaBattle.Data

Assembly: SeaBattle.dll

```
public class ApplicationDbContext : DbContext, IInfrastructure<IServiceProvider>,
IDbContextDependencies, IDbSetCache, IDbContextPoolable, IResettableService,
IDisposable, IAsyncDisposable
```

Inheritance

object ← DbContext ← ApplicationDbContext

Implements

IInfrastructure<IServiceProvider>, IDbContextDependencies, IDbSetCache, IDbContextPoolable, IResettableService, IDisposable, IAsyncDisposable

Inherited Members

DbContext.Set<TEntity>() , DbContext.Set<TEntity>(string) ,
DbContext.OnConfiguring(DbContextOptionsBuilder) ,
DbContext.ConfigureConventions(ModelConfigurationBuilder) , DbContext.SaveChanges() ,
DbContext.SaveChanges(bool) , DbContext.SaveChangesAsync(CancellationToken) ,
DbContext.SaveChangesAsync(bool, CancellationToken) , DbContext.Dispose() ,
DbContext.DisposeAsync() , DbContext.Entry<TEntity>(TEntity) , DbContext.Entry(object) ,
DbContext.Add<TEntity>(TEntity) , DbContext.AddAsync<TEntity>(TEntity, CancellationToken) ,
DbContext.Attach<TEntity>(TEntity) , DbContext.Update<TEntity>(TEntity) ,
DbContext.Remove<TEntity>(TEntity) , DbContext.Add(object) ,
DbContext.AddAsync(object, CancellationToken) , DbContext.Attach(object) ,
DbContext.Update(object) , DbContext.Remove(object) , DbContext.AddRange(params object[]) ,
DbContext.AddRangeAsync(params object[]) , DbContext.AttachRange(params object[]) ,
DbContext.UpdateRange(params object[]) , DbContext.RemoveRange(params object[]) ,
DbContext.AddRange(IEnumerable<object>) ,
DbContext.AddRangeAsync(IEnumerable<object>, CancellationToken) ,
DbContext.AttachRange(IEnumerable<object>) , DbContext.UpdateRange(IEnumerable<object>) ,
DbContext.RemoveRange(IEnumerable<object>) , DbContext.Find(Type, params object[]) ,
DbContext.FindAsync(Type, params object[]) ,
DbContext.FindAsync(Type, object[], CancellationToken) , DbContext.Find<TEntity>(params object[]) ,
DbContext.FindAsync<TEntity>(params object[]) ,
DbContext.FindAsync<TEntity>(object[], CancellationToken) ,
DbContext.FromExpression<TResult>(Expression<Func<IQueryable<TResult>>>) ,

DbContext.Database , DbContext.ChangeTracker , DbContext.Model , DbContext.ContextId , DbContext.SavingChanges , DbContext.SavedChanges , DbContext.SaveChangesFailed , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ApplicationDbContext(DbContextOptions<ApplicationDbContext>)

```
public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
```

Parameters

`options` DbContextOptions<ApplicationDbContext>

Properties

GameHistories

```
public virtual DbSet<GameHistory> GameHistories { get; set; }
```

Property Value

DbSet<GameHistory>

PlayerRankings

```
public virtual DbSet<PlayerRanking> PlayerRankings { get; set; }
```

Property Value

DbSet<PlayerRanking>

Users

```
public virtual DbSet<User> Users { get; set; }
```

Property Value

DbSet <User>

Methods

OnModelCreating(ModelBuilder)

Override this method to further configure the model that was discovered by convention from the entity types exposed in DbSet<TEntity> properties on your derived context. The resulting model may be cached and re-used for subsequent instances of your derived context.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
```

Parameters

`modelBuilder` ModelBuilder

> The builder being used to construct the model for this context. Databases (and other extensions) typically define extension methods on this object that allow you to configure aspects of the model that are specific to a given database.

Remarks

If a model is explicitly set on the options for this context (via UseModel(IModel)) then this method will not be run. However, it will still run when creating a compiled model.

See Modeling entity types and relationships for more information and examples.

Class DesignTimeDbContextFactory

Namespace: SeaBattle.Data

Assembly: SeaBattle.dll

```
public class DesignTimeDbContextFactory : IDesignTimeDbContextFactory<ApplicationDbContext>
```

Inheritance

object ← DesignTimeDbContextFactory

Implements

IDesignTimeDbContextFactory<ApplicationDbContext>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

CreateDbContext(string[])

Creates a new instance of a derived context.

```
public ApplicationDbContext CreateDbContext(string[] args)
```

Parameters

`args` string[]

Arguments provided by the design-time service.

Returns

ApplicationDbContext

An instance of ApplicationDbContext.

Namespace SeaBattle.Hubs

Classes

GameHub

SignalR хаб для обработки real-time взаимодействия в игре морской бой

LobbyHub

SignalR хаб для управления игровыми лобби

Class GameHub

Namespace: SeaBattle.Hubs

Assembly: SeaBattle.dll

SignalR хаб для обработки real-time взаимодействия в игре морской бой

```
public class GameHub : Hub, IDisposable
```

Inheritance

object ← Hub ← GameHub

Implements

IDisposable

Inherited Members

Hub.Dispose(bool) , Hub.Dispose() , Hub.Clients , Hub.Context , Hub.Groups , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

GameHub(IGameService, ILogger<GameHub>)

Инициализирует новый экземпляр хаба игры

```
public GameHub(IGameService gameService, ILogger<GameHub> logger)
```

Parameters

`gameService` IGameService

Сервис для управления игровым процессом

`logger` ILogger<GameHub>

Логгер для записи событий

Methods

CreateGame(string, bool)

Создает новую игру

```
public Task<string> CreateGame(string creatorName, bool isOpenLobby = false)
```

Parameters

`creatorName` string

Имя создателя игры

`isOpenLobby` bool

Флаг, указывающий является ли лобби открытым

Returns

Task <string>

Идентификатор созданной игры

GetGameState(string, string)

Получает текущее состояние игры

```
public Task GetGameState(string gameId, string playerName)
```

Parameters

`gameId` string

Идентификатор игры

`playerName` string

Имя игрока

Returns

Task

GetMyGameHistory(string, int)

Получает историю игр пользователя

```
public Task GetMyGameHistory(string playerName, int count = 10)
```

Parameters

`playerName` string

Имя игрока

`count` int

Количество записей для отображения

Returns

Task

GetOpenLobbies()

Получает список открытых лобби

```
public Task<List<Game>> GetOpenLobbies()
```

Returns

Task<List<Game>>

Список доступных игр

JoinGame(string, string)

Присоединяет игрока к существующей игре

```
public Task<Game?> JoinGame(string gameId, string joinerName)
```

Parameters

`gameId` string

Идентификатор игры

`joinerName` string

Имя присоединяющегося игрока

Returns

Task<Game>

Информация об игре или null, если присоединиться не удалось

MakeShot(string, string, Position)

Выполняет выстрел по указанной позиции

```
public Task<ShotResultResponse> MakeShot(string gameId, string playerName,
Position position)
```

Parameters

`gameId` string

Идентификатор игры

`playerName` string

Имя игрока, выполняющего выстрел

`position` Position

Позиция выстрела

Returns

Task<ShotResultResponse>

Результат выстрела

OnConnectedAsync()

Обрабатывает подключение клиента к хабу

```
public override Task OnConnectedAsync()
```

Returns

Task

OnDisconnectedAsync(Exception?)

Обрабатывает отключение клиента от хаба

```
public override Task OnDisconnectedAsync(Exception? exception)
```

Parameters

`exception` Exception

Исключение, если оно возникло при отключении

Returns

Task

SetReady(string, string)

Устанавливает готовность игрока к началу игры

```
public Task<Game?> SetReady(string gameId, string playerName)
```

Parameters

`gameId` string

Идентификатор игры

`playerName` string

Имя игрока

Returns

Task<Game>

Обновленная информация об игре или null в случае ошибки

SubmitBoardPlacement(string, string, int[][])

Отправляет расстановку кораблей на игровом поле

```
public Task<Game?> SubmitBoardPlacement(string gameId, string playerName, int[]
[] clientBoard)
```

Parameters

`gameId` string

Идентификатор игры

`playerName` string

Имя игрока

`clientBoard` int[][]

Массив с расстановкой кораблей

Returns

Task<Game>

Обновленная информация об игре или null в случае ошибки

Class LobbyHub

Namespace: SeaBattle.Hubs

Assembly: SeaBattle.dll

SignalR хаб для управления игровыми лобби

```
public class LobbyHub : Hub, IDisposable
```

Inheritance

object ← Hub ← LobbyHub

Implements

IDisposable

Inherited Members

Hub.Dispose(bool) , Hub.Dispose() , Hub.Clients , Hub.Context , Hub.Groups , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

LobbyHub(IGameService)

Инициализирует новый экземпляр хаба лобби

```
public LobbyHub(IGameService gameService)
```

Parameters

`gameService` IGameService

Сервис для управления игровым процессом

Methods

CreateLobby(string, bool)

Создает новое лобби для игры

```
public Task CreateLobby(string creatorName, bool isOpenLobby)
```

Parameters

`creatorName` string

Имя создателя лобби

`isOpenLobby` bool

Флаг, указывающий является ли лобби открытым

Returns

Task

JoinLobby(string, string)

Присоединяет игрока к существующему лобби

```
public Task JoinLobby(string gameId, string opponentName)
```

Parameters

`gameId` string

Идентификатор игры

`opponentName` string

Имя присоединяющегося игрока

Returns

Task

OnConnectedAsync()

Обрабатывает подключение клиента к хабу и отправляет список открытых лобби

```
public override Task OnConnectedAsync()
```

Returns

Task

OnDisconnectedAsync(Exception?)

Обрабатывает отключение клиента от хаба и обновляет список открытых лобби

```
public override Task OnDisconnectedAsync(Exception? exception)
```

Parameters

`exception` Exception

Исключение, если оно возникло при отключении

Returns

Task

Namespace SeaBattle.Migrations

Classes

GameHistory

A base class inherited by each EF Core migration.

Initial

A base class inherited by each EF Core migration.

New

A base class inherited by each EF Core migration.

PlayerRanking

A base class inherited by each EF Core migration.

Class GameHistory

Namespace: SeaBattle.Migrations

Assembly: SeaBattle.dll

A base class inherited by each EF Core migration.

```
[DbContext(typeof(ApplicationDbContext))]
[Migration("20250515082549_GameHistory")]
public class GameHistory : Migration
```

Inheritance

object ← Migration ← GameHistory

Inherited Members

Migration.InitialDatabase , Migration.TargetModel , Migration.UpOperations , Migration.DownOperations , Migration.ActiveProvider , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

See Database migrations for more information and examples.

Methods

BuildTargetModel(ModelBuilder)

Implemented to build the TargetModel.

```
protected override void BuildTargetModel(ModelBuilder modelBuilder)
```

Parameters

`modelBuilder` ModelBuilder

The ModelBuilder to use to build the model.

Remarks

See Database migrations for more information and examples.

Down(MigrationBuilder)

Builds the operations that will migrate the database 'down'.

```
protected override void Down(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by this migration so that it returns to the state that it was in before this migration was applied.

This method must be overridden in each class that inherits from Migration if both 'up' and 'down' migrations are to be supported. If it is not overridden, then calling it will throw and it will not be possible to migrate in the 'down' direction.

See Database migrations for more information and examples.

Up(MigrationBuilder)

Builds the operations that will migrate the database 'up'.

```
protected override void Up(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by the previous migration so that it is up-to-date with regard to this migration.

This method must be overridden in each class that inherits from Migration.

See Database migrations for more information and examples.

Class Initial

Namespace: SeaBattle.Migrations

Assembly: SeaBattle.dll

A base class inherited by each EF Core migration.

```
[DbContext(typeof(ApplicationDbContext))]
[Migration("20250511101602_Initial")]
public class Initial : Migration
```

Inheritance

object ← Migration ← Initial

Inherited Members

Migration.InitialDatabase , Migration.TargetModel , Migration.UpOperations , Migration.DownOperations , Migration.ActiveProvider , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

See Database migrations for more information and examples.

Methods

BuildTargetModel(ModelBuilder)

Implemented to build the TargetModel.

```
protected override void BuildTargetModel(ModelBuilder modelBuilder)
```

Parameters

`modelBuilder` ModelBuilder

The ModelBuilder to use to build the model.

Remarks

See Database migrations for more information and examples.

Down(MigrationBuilder)

Builds the operations that will migrate the database 'down'.

```
protected override void Down(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by this migration so that it returns to the state that it was in before this migration was applied.

This method must be overridden in each class that inherits from Migration if both 'up' and 'down' migrations are to be supported. If it is not overridden, then calling it will throw and it will not be possible to migrate in the 'down' direction.

See Database migrations for more information and examples.

Up(MigrationBuilder)

Builds the operations that will migrate the database 'up'.

```
protected override void Up(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by the previous migration so that it is up-to-date with regard to this migration.

This method must be overridden in each class that inherits from Migration.

See Database migrations for more information and examples.

Class New

Namespace: SeaBattle.Migrations

Assembly: SeaBattle.dll

A base class inherited by each EF Core migration.

```
[DbContext(typeof(ApplicationDbContext))]
[Migration("20250518154546_New")]
public class New : Migration
```

Inheritance

object ← Migration ← New

Inherited Members

Migration.InitialDatabase , Migration.TargetModel , Migration.UpOperations , Migration.DownOperations , Migration.ActiveProvider , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

See Database migrations for more information and examples.

Methods

BuildTargetModel(ModelBuilder)

Implemented to build the TargetModel.

```
protected override void BuildTargetModel(ModelBuilder modelBuilder)
```

Parameters

`modelBuilder` ModelBuilder

The ModelBuilder to use to build the model.

Remarks

See Database migrations for more information and examples.

Down(MigrationBuilder)

Builds the operations that will migrate the database 'down'.

```
protected override void Down(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by this migration so that it returns to the state that it was in before this migration was applied.

This method must be overridden in each class that inherits from Migration if both 'up' and 'down' migrations are to be supported. If it is not overridden, then calling it will throw and it will not be possible to migrate in the 'down' direction.

See Database migrations for more information and examples.

Up(MigrationBuilder)

Builds the operations that will migrate the database 'up'.

```
protected override void Up(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by the previous migration so that it is up-to-date with regard to this migration.

This method must be overridden in each class that inherits from Migration.

See Database migrations for more information and examples.

Class PlayerRanking

Namespace: SeaBattle.Migrations

Assembly: SeaBattle.dll

A base class inherited by each EF Core migration.

```
[DbContext(typeof(ApplicationDbContext))]
[Migration("20250515090022_PlayerRanking")]
public class PlayerRanking : Migration
```

Inheritance

object ← Migration ← PlayerRanking

Inherited Members

Migration.InitialDatabase , Migration.TargetModel , Migration.UpOperations , Migration.DownOperations , Migration.ActiveProvider , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

See Database migrations for more information and examples.

Methods

BuildTargetModel(ModelBuilder)

Implemented to build the TargetModel.

```
protected override void BuildTargetModel(ModelBuilder modelBuilder)
```

Parameters

`modelBuilder` ModelBuilder

The ModelBuilder to use to build the model.

Remarks

See Database migrations for more information and examples.

Down(MigrationBuilder)

Builds the operations that will migrate the database 'down'.

```
protected override void Down(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by this migration so that it returns to the state that it was in before this migration was applied.

This method must be overridden in each class that inherits from Migration if both 'up' and 'down' migrations are to be supported. If it is not overridden, then calling it will throw and it will not be possible to migrate in the 'down' direction.

See Database migrations for more information and examples.

Up(MigrationBuilder)

Builds the operations that will migrate the database 'up'.

```
protected override void Up(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by the previous migration so that it is up-to-date with regard to this migration.

This method must be overridden in each class that inherits from Migration.

See Database migrations for more information and examples.

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

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

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

string

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 GameStateInfo

Namespace: SeaBattle.Models

Assembly: SeaBattle.dll

```
public class GameStateInfo
```

Inheritance

object ← GameStateInfo

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

CreatorBoard

```
public required CellState[,] CreatorBoard { get; set; }
```

Property Value

CellState[,]

CreatorName

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

Property Value

string

CreatorShots

```
public required Position[] CreatorShots { get; set; }
```

Property Value

Position[]

GameId

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

Property Value

string

IsCreatorReady

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

Property Value

bool

IsCreatorTurn

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

Property Value

bool

IsGameEnded

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

Property Value

bool

IsGameStarted

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

Property Value

bool

IsOpenLobby

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

Property Value

bool

IsOpponentReady

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

Property Value

bool

OpponentBoard

```
public required CellState[,] OpponentBoard { get; set; }
```

Property Value

CellState[,]

OpponentName

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

Property Value

string

OpponentShots

```
public required Position[] OpponentShots { get; set; }
```

Property Value

Position[]

Winner

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

Property Value

string

Class GameStateManager

Namespace: SeaBattle.Models

Assembly: SeaBattle.dll

```
public static class GameStateManager
```

Inheritance

object ← GameStateManager

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

CreateGame(string, bool)

```
public static GameStateInfo CreateGame(string creatorName, bool isOpenLobby = false)
```

Parameters

`creatorName` string

`isOpenLobby` bool

Returns

GameStateInfo

GetGame(string)

```
public static GameStateInfo? GetGame(string gameId)
```

Parameters

`gameId` string

Returns

GameStateInfo

GetOpenLobbies()

```
public static IEnumerable<GameStateInfo> GetOpenLobbies()
```

Returns

IEnumerable<GameStateInfo>

RemoveGame(string)

```
public static void RemoveGame(string gameId)
```

Parameters

`gameId` string

Enum GameStateManager.GameState

Namespace: SeaBattle.Models

Assembly: SeaBattle.dll

```
public enum GameStateManager.GameState
```

Fields

`Finished = 3`

`InProgress = 2`

`WaitingForOpponent = 0`

`WaitingForReady = 1`

Enum GameStatus

Namespace: SeaBattle.Models

Assembly: SeaBattle.dll

```
public enum GameStatus
```

Fields

`Finished = 3`

`InProgress = 2`

`WaitingForOpponent = 0`

`WaitingForReady = 1`

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

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 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`

Промах

`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

Namespace SeaBattle.Models.Converters

Classes

BoardConverter

Class BoardConverter

Namespace: SeaBattle.Models.Converters

Assembly: SeaBattle.dll

```
public class BoardConverter : JsonConverter<CellState[,]>
```

Inheritance

object ← JsonConverter ← JsonConverter<CellState[,]> ← BoardConverter

Inherited Members

JsonConverter<CellState[,]>.CanConvert(Type) ,
JsonConverter<CellState[,]>.Read(ref Utf8JsonReader, Type, JsonSerializerOptions) ,
JsonConverter<CellState[,]>.ReadAsPropertyName(ref Utf8JsonReader, Type, JsonSerializerOptions) ,
JsonConverter<CellState[,]>.Write(Utf8JsonWriter, CellState[,], JsonSerializerOptions) ,
JsonConverter<CellState[,]>.WriteAsPropertyName(Utf8JsonWriter, CellState[,], JsonSerializerOptions) ,
JsonConverter<CellState[,]>.HandleNull , JsonConverter<CellState[,]>.Type ,
JsonConverter.CanConvert(Type) , JsonConverter.Type , object.Equals(object) ,
object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Read(ref Utf8JsonReader, Type, JsonSerializerOptions)

Reads and converts the JSON to type `T`.

```
public override CellState[,] Read(ref Utf8JsonReader reader, Type typeToConvert,
JsonSerializerOptions options)
```

Parameters

`reader` Utf8JsonReader

The reader.

`typeToConvert` Type

The type to convert.

`options` JsonSerializerOptions

An object that specifies serialization options to use.

Returns

CellState[,]

The converted value.

Write(Utf8JsonWriter, CellState[,], JsonSerializerOptions)

Writes a specified value as JSON.

```
public override void Write(Utf8JsonWriter writer, CellState[,] value,
JsonSerializerOptions options)
```

Parameters

`writer` Utf8JsonWriter

The writer to write to.

`value` CellState[,]

The value to convert to JSON.

`options` JsonSerializerOptions

An object that specifies serialization options to use.

Namespace SeaBattle.Services

Classes

GameService

UserService

Interfaces

IGameService

IUserService

Class GameService

Namespace: SeaBattle.Services

Assembly: SeaBattle.dll

```
public class GameService : IGameService
```

Inheritance

object ← GameService

Implements

IGameService

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

GameService(ILogger<GameService>, ApplicationDbContext)

```
public GameService(ILogger<GameService> logger, ApplicationDbContext dbContext)
```

Parameters

`logger` ILogger<GameService>

`dbContext` ApplicationDbContext

Methods

AddGameToHistory(Game, string, string)

```
public Task AddGameToHistory(Game game, string winnerUsername, string loserUsername)
```

Parameters

`game` Game

`winnerUsername` string

`loserUsername` string

Returns

Task

CreateGame(string, bool)

```
public Task<Game> CreateGame(string creatorName, bool isOpenLobby)
```

Parameters

`creatorName` string

`isOpenLobby` bool

Returns

Task<Game>

GetGame(string)

```
public Task<Game?> GetGame(string gameId)
```

Parameters

`gameId` string

Returns

Task<Game>

GetLeaderboardAsync(int)

```
public Task<List<PlayerRanking>> GetLeaderboardAsync(int topN)
```

Parameters

`topN` int

Returns

Task<List<PlayerRanking>>

GetOpenLobbies()

```
public Task<List<Game>> GetOpenLobbies()
```

Returns

Task<List<Game>>

GetPlayerGameHistory(string, int)

```
public Task<List<GameHistory>> GetPlayerGameHistory(string playerName, int count = 10)
```

Parameters

`playerName` string

`count` int

Returns

Task<List<GameHistory>>

JoinGame(string, string)

```
public Task<Game?> JoinGame(string gameId, string joinerName)
```

Parameters

`gameId` string

`joinerName` string

Returns

Task<Game>

MakeShot(string, string, Position)

```
public Task<(Game? game, ShotResult result)> MakeShot(string gameId, string playerName,
Position position)
```

Parameters

`gameId` string

`playerName` string

`position` Position

Returns

Task<(Game game, ShotResult result)>

PlaceShipsAsync(string, string, CellState[,])

```
public Task<Game?> PlaceShipsAsync(string gameId, string playerName,
CellState[,] clientBoard)
```

Parameters

`gameId` string

`playerName` string

`clientBoard` CellState[,]

Returns

Task<Game>

SetReady(string, string)

```
public Task<Game?> SetReady(string gameId, string playerName)
```

Parameters

`gameId` string

`playerName` string

Returns

Task<Game>

Interface IGameService

Namespace: SeaBattle.Services

Assembly: SeaBattle.dll

```
public interface IGameService
```

Methods

CreateGame(string, bool)

```
Task<Game> CreateGame(string creatorName, bool isOpenLobby)
```

Parameters

`creatorName` string

`isOpenLobby` bool

Returns

Task<Game>

GetGame(string)

```
Task<Game?> GetGame(string gameId)
```

Parameters

`gameId` string

Returns

Task<Game>

GetLeaderboardAsync(int)

```
Task<List<PlayerRanking>> GetLeaderboardAsync(int topN)
```

Parameters

`topN` int

Returns

Task<List<PlayerRanking>>

GetOpenLobbies()

```
Task<List<Game>> GetOpenLobbies()
```

Returns

Task<List<Game>>

GetPlayerGameHistory(string, int)

```
Task<List<GameHistory>> GetPlayerGameHistory(string playerName, int count = 10)
```

Parameters

`playerName` string

`count` int

Returns

Task<List<GameHistory>>

JoinGame(string, string)

```
Task<Game?> JoinGame(string gameId, string joinerName)
```

Parameters

`gameId` string

`joinerName` string

Returns

Task<Game>

MakeShot(string, string, Position)

```
Task<(Game? game, ShotResult result)> MakeShot(string gameId, string playerName,
Position position)
```

Parameters

`gameId` string

`playerName` string

`position` Position

Returns

Task<(Game game, ShotResult result)>

PlaceShipsAsync(string, string, CellState[,])

```
Task<Game?> PlaceShipsAsync(string gameId, string playerName, CellState[,] clientBoard)
```

Parameters

`gameId` string

`playerName` string

`clientBoard` CellState[,]

Returns

Task<Game>

SetReady(string, string)

```
Task<Game?> SetReady(string gameId, string playerName)
```

Parameters

`gameId` string

`playerName` string

Returns

Task<Game>

Interface IUserService

Namespace: SeaBattle.Services

Assembly: SeaBattle.dll

```
public interface IUserService
```

Methods

CreateUser(string, string)

```
Task<User> CreateUser(string username, string password)
```

Parameters

`username` string

`password` string

Returns

Task<User>

GetUserByUsername(string)

```
Task<User?> GetUserByUsername(string username)
```

Parameters

`username` string

Returns

Task<User>

ValidateUser(string, string)

```
Task<bool> ValidateUser(string username, string password)
```

Parameters

`username` string

`password` string

Returns

Task<bool>

Class UserService

Namespace: SeaBattle.Services

Assembly: SeaBattle.dll

```
public class UserService : IUserService
```

Inheritance

object ← UserService

Implements

IUserService

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

UserService(ApplicationDbContext)

```
public UserService(ApplicationDbContext context)
```

Parameters

`context` ApplicationDbContext

Methods

CreateUser(string, string)

```
public Task<User> CreateUser(string username, string password)
```

Parameters

`username` string

`password` string

Returns

Task<User>

GetUserByUsername(string)

```
public Task<User?> GetUserByUsername(string username)
```

Parameters

`username` string

Returns

Task<User>

ValidateUser(string, string)

```
public Task<bool> ValidateUser(string username, string password)
```

Parameters

`username` string

`password` string

Returns

Task<bool>