

# Serwer Smart Dartboard

Produkcyjny serwer aplikacji Smart Dartboard z integracją Arduino.

## Funkcjonalności

- **Zarządzanie grą w czasie rzeczywistym** via Socket.IO
- **Integracja dartboardu Arduino** via port szeregowy
- **Uwierzytelnianie JWT** dla bezpiecznych sesji użytkownika
- **Wiele trybów gry**: 501, 301, Cricket, Killer
- **Śledzenie statystyk graczy**
- **System lobbyów** z zarządzaniem gospodarzem

## Stos techniczny

- **Runtime**: Node.js
- **Framework**: Express 5
- **Czas rzeczywisty**: Socket.IO
- **Uwierzytelnianie**: JWT + bcryptjs
- **Sprzęt**: SerialPort (Arduino)
- **Baza danych**: Magazyn pliku JSON

## Wymagania

- Node.js 18+
- Dartboard Arduino (opcjonalnie - serwer działa bez niego)

## Instalacja

```
npm install
```

## Konfiguracja

Zmienne środowiskowe (opcjonalnie):

Zmienna	Domyślnie	Opis
PORT	3001	Port serwera
JWT_SECRET	smart-dartboard-secret-key-2024	Klucz podpisania JWT
SERIAL_PORT_PATH	/dev/ttyACM0	Port szeregowy Arduino
SERIAL_BAUD_RATE	115200	Szybkość transmisji szeregowej

## Uruchomienie

## Produkcja

```
node server.js
```

Serwer będzie dostępny pod adresem:

- **Aplikacja:** <http://localhost:3001>
- **API:** <http://localhost:3001/api>
- **WebSocket:** <ws://localhost:3001>

Z PM2 (rekomendowane)

```
pm2 start server.js --name dartboard
```

## Punkty końcowe API

### Uwierzytelnianie

Metoda	Endpoint	Opis
POST	<a href="#">/api/auth/register</a>	Zarejestruj nowego użytkownika
POST	<a href="#">/api/auth/login</a>	Zaloguj użytkownika
GET	<a href="#">/api/auth/me</a>	Pobierz bieżącego użytkownika

### Profil

Metoda	Endpoint	Opis
PUT	<a href="#">/api/profile/avatar</a>	Aktualizuj awatar
GET	<a href="#">/api/profile/avatars</a>	Pobierz dostępne awatary
PUT	<a href="#">/api/profile/username</a>	Zmień nazwę użytkownika
PUT	<a href="#">/api/profile/password</a>	Zmień hasło

### Lobby

Metoda	Endpoint	Opis
GET	<a href="#">/api/lobbies</a>	Wyświetl wszystkie lobby
POST	<a href="#">/api/lobbies</a>	Utwórz lobby
GET	<a href="#">/api/lobbies/:id</a>	Pobierz szczegóły lobby
POST	<a href="#">/api/lobbies/:id/join</a>	Dołącz do lobby

Metoda	Endpoint	Opis
POST	<code>/api/lobbies/:id/leave</code>	Opuść lobby
PUT	<code>/api/lobbies/:id/mode</code>	Ustaw tryb gry
POST	<code>/api/lobbies/:id/start</code>	Uruchom grę
POST	<code>/api/lobbies/:id/end</code>	Zakończ grę
POST	<code>/api/lobbies/:id/abort</code>	Przerwij grę
POST	<code>/api/lobbies/:id/undo-throw</code>	Cofnij ostatni rzut

## Gra

Metoda	Endpoint	Opis
GET	<code>/api/game/can-start</code>	Sprawdź dostępność dartboardu
GET	<code>/api/game/dartboard/status</code>	Pobierz stan połączenia dartboardu

## Zdarzenia Socket.IO

### Klient → Serwer

Zdarzenie	Ładunek	Opis
<code>join_lobby</code>	<code>lobbyId: string</code>	Dołącz do pokoju lobby
<code>leave_lobby</code>	<code>lobbyId: string</code>	Opuść pokój lobby

### Serwer → Klient

Zdarzenie	Ładunek	Opis
<code>lobby_update</code>	<code>Lobby</code>	Stan lobby zmienił się
<code>game_update</code>	<code>GameState</code>	Stan gry zmienił się
<code>game_started</code>	<code>GameState</code>	Gra się zaczęła
<code>game_ended</code>	-	Gra zakończyła się normalnie
<code>game_aborted</code>	<code>{ abortedBy: string }</code>	Gra przerwana
<code>host_changed</code>	<code>{ newHostId, newHostName }</code>	Host zmienił się
<code>lobby_deleted</code>	-	Lobby zostało usunięte
<code>dartboard_status</code>	<code>{ connected: boolean }</code>	Stan połączenia dartboardu

## Protokół Arduino

Serwer oczekuje wiadomości JSON z Arduino poprzez port szeregowy:

```
{
  "event": "hit",
  "sector": 20,
  "multiplier": 3,
  "score": 60
}
```

## Pola

Pole	Typ	Opis
event	string	Typ zdarzenia (hit)
sector	number	Sektor dartboardu (1-20, 25 dla bull)
multiplier	number	1 = pojedyncze, 2 = podwójne, 3 = potrójne
score	number	Obliczony wynik (sektor × mnożnik)

## Baza danych

Dane przechowywane są w `database.json`:

```
{
  "users": [],
  "lobbies": [],
  "activeGame": null
}
```

Baza danych jest automatycznie tworzona przy pierwszym uruchomieniu.

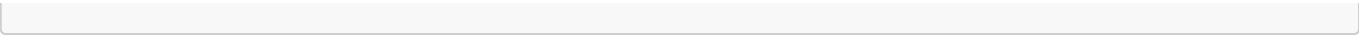
## Struktura projektu

```
backend-emulator/
├── server.js      # Główny plik serwera
├── database.json  # Magazyn danych
├── package.json   # Zależności
├── public/        # Kompilacja frontendu (z dart-app)
└── README.md     # Ten plik
```

## Programowanie

Symuluj rzut (bez Arduino)

```
curl -X POST http://localhost:3001/api/lobbies/{id}/simulate-throw \
-H "Authorization: Bearer {token}"
```



# Licencja

MIT