# Dokumentacja Aplikacji GYMPROGress

## Spis treści

1. [Wprowadzenie](#wprowadzenie)

2. [Wymagania systemowe](#wymagania-systemowe)

3. [Struktura projektu](#struktura-projektu)

4. [Instalacja i konfiguracja](#instalacja-i-konfiguracja)

5. [Architektura systemu](#architektura-systemu)

6. [Baza danych](#baza-danych)

7. [API Endpoints](#api-endpoints)

8. [Przewodnik użytkownika](#przewodnik-użytkownika)

9. [Rozwiązywanie problemów](#rozwiązywanie-problemów)

10. [Bezpieczeństwo](#bezpieczeństwo)

## Wprowadzenie

GYMPROGress to nowoczesna aplikacja webowa do śledzenia postępów treningowych, oferująca zaawansowane funkcje monitorowania ćwiczeń, wizualizacji danych oraz synchronizacji między urządzeniami.

### Główne funkcje:

- Tworzenie elastycznych planów treningowych

- Śledzenie progresu z systemem kolorowej wizualizacji

- System ocen trudności z emoji

- Timer przerw między seriami

- Zaawansowane wykresy progresu

- Eksport/import danych przez link

- Leaderboardy i funkcje społecznościowe

- PWA (Progressive Web App)

## Wymagania systemowe

### Po stronie klienta:

- Przeglądarka internetowa: Chrome 80+, Firefox 75+, Safari 13+, Edge 80+

- Włączona obsługa JavaScript

- Wymagana rozdzielczość: min. 320px × 480px

- Zalecane urządzenia: komputer, tablet, smartphone

### Po stronie serwera:

- Node.js 16.0 lub nowszy

- SQLite3

- 512MB RAM (minimalnie)

- 1GB wolnego miejsca na dysku

## Struktura projektu

```

gymprogress/

├── client/ # Frontend React

│ ├── src/

│ │ ├── components/ # Komponenty React

│ │ ├── hooks/ # Własne hooki

│ │ ├── utils/ # Narzędzia

│ │ └── styles/ # Style CSS Modules

│ └── public/ # Ikony PWA, manifest

├── server/ # Backend Node.js

│ ├── models/ # Model bazy danych

│ ├── routes/ # Endpointy API

│ ├── middleware/ # Middleware (auth, validation)

│ └── database/ # Konfiguracja SQLite

└── shared/ # Wspólne narzędzia

```

## Instalacja i konfiguracja

### Środowisko developerskie:

1. \*\*Klonowanie repozytorium\*\*

```bash

git clone <repository-url>

cd gymprogress

```

2. \*\*Instalacja zależności\*\*

```bash

# Instalacja zależności serwera

cd server

npm install

# Instalacja zależności klienta

cd ../client

npm install

```

3. \*\*Konfiguracja bazy danych\*\*

```bash

cd server

npm run init-db

```

4. \*\*Uruchomienie aplikacji\*\*

```bash

# Uruchomienie serwera (w terminalu 1)

cd server

npm run dev

# Uruchomienie klienta (w terminalu 2)

cd client

npm run dev

```

### Środowisko produkcyjne:

1. \*\*Budowanie aplikacji\*\*

```bash

# Budowanie klienta

cd client

npm run build

# Przygotowanie serwera

cd ../server

npm install --production

```

2. \*\*Konfiguracja zmiennych środowiskowych\*\*

Utwórz plik `.env` w katalogu server:

```

PORT=3001

JWT\_SECRET=twoj\_tajny\_klucz

NODE\_ENV=production

DATABASE\_PATH=./database/gymprogress.db

```

3. \*\*Uruchomienie produkcji\*\*

```bash

cd server

npm start

```

## Architektura systemu

### Frontend:

- \*\*React\*\* z \*\*Vite\*\* - nowoczesny bundler zapewniający szybkie budowanie

- \*\*CSS Modules\*\* - izolacja stylów komponentów

- \*\*Chart.js\*\* - wizualizacja danych treningowych

- \*\*PWA\*\* - możliwość instalacji na urządzenia mobilne

### Backend:

- \*\*Node.js\*\* + \*\*Express\*\* - serwer API

- \*\*SQLite\*\* - baza danych

- \*\*JWT\*\* - autoryzacja użytkowników

### Przepływ danych:

1. Klient wysyła żądanie do API

2. Middleware weryfikuje autoryzację

3. Kontroler przetwarza żądanie

4. Model wykonuje operacje na bazie danych

5. Dane są zwracane do klienta w formacie JSON

## Baza danych

### Schemat bazy danych:

```sql

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

email TEXT UNIQUE,

password\_hash TEXT,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE IF NOT EXISTS workout\_plans (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

name TEXT,

days\_count INTEGER,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users (id)

);

CREATE TABLE IF NOT EXISTS workout\_days (

id INTEGER PRIMARY KEY AUTOINCREMENT,

plan\_id INTEGER,

day\_number INTEGER,

name TEXT,

FOREIGN KEY (plan\_id) REFERENCES workout\_plans (id)

);

CREATE TABLE IF NOT EXISTS exercises (

id INTEGER PRIMARY KEY AUTOINCREMENT,

day\_id INTEGER,

name TEXT,

description TEXT,

sets INTEGER,

FOREIGN KEY (day\_id) REFERENCES workout\_days (id)

);

CREATE TABLE IF NOT EXISTS exercise\_logs (

id INTEGER PRIMARY KEY AUTOINCREMENT,

exercise\_id INTEGER,

reps INTEGER,

weight REAL,

duration INTEGER,

difficulty\_emoji TEXT,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (exercise\_id) REFERENCES exercises (id)

);

```

### Opis tabel:

- \*\*users\*\* - przechowuje dane użytkowników

- \*\*workout\_plans\*\* - plany treningowe użytkowników

- \*\*workout\_days\*\* - dni treningowe w planach

- \*\*exercises\*\* - ćwiczenia w dniach treningowych

- \*\*exercise\_logs\*\* - historia wykonanych ćwiczeń

## API Endpoints

### Podstawowe:

- `GET /api` - Informacja o API

### Plany treningowe:

- `GET /api/workout-plans` - Pobierz wszystkie plany treningowe

- `POST /api/workout-plans` - Utwórz nowy plan treningowy

- `GET /api/workout-plans/:id` - Pobierz szczegóły planu treningowego

### Dni treningowe:

- `POST /api/workout-days` - Dodaj dzień treningowy do planu

### Ćwiczenia:

- `POST /api/exercises` - Dodaj ćwiczenie do dnia treningowego

- `GET /api/exercises/:day\_id` - Pobierz ćwiczenia dla dnia treningowego

- `POST /api/exercise-logs` - Zapisz wyniki ćwiczenia

## Przewodnik użytkownika

### Tworzenie planu treningowego:

1. Zaloguj się do aplikacji

2. Kliknij "Nowy plan" na stronie głównej

3. Wprowadź nazwę planu i liczbę dni

4. Dodaj ćwiczenia do każdego dnia

5. Zapisz plan

### Śledzenie treningu:

1. Wybierz plan treningowy

2. Rozpocznij trening wybierając odpowiedni dzień

3. Dla każdego ćwiczenia wprowadź:

- Liczbę powtórzeń

- Ciężar

- Czas trwania (opcjonalnie)

- Ocenę trudności (emoji)

4. Używaj timera do mierzenia przerw

### Analiza postępów:

1. Przejdź do zakładki "Statystyki"

2. Przeglądaj wykresy progresu dla każdego ćwiczenia

3. Porównuj swoje wyniki z poprzednimi treningami

4. Sprawdzaj leaderboardy (dla użytkowników zalogowanych)

## Rozwiązywanie problemów

### Częste problemy i rozwiązania:

\*\*Problem: Aplikacja nie uruchamia się\*\*

- Rozwiązanie: Sprawdź, czy masz zainstalowany Node.js w wersji 16 lub nowszej

\*\*Problem: Błąd przy połączeniu z bazą danych\*\*

- Rozwiązanie: Upewnij się, że katalog `server/database` istnieje i ma uprawnienia do zapisu

\*\*Problem: Timer nie działa\*\*

- Rozwiązanie: Sprawdź, czy przeglądarka nie blokuje autoodtwarzania dźwięku

\*\*Problem: Nie zapisują się treningi\*\*

- Rozwiązanie: Sprawdź połączenie internetowe i czy serwer jest uruchomiony

### Kontakt do supportu:

- Email: support@gymprogress.com

- Dokumentacja online: https://docs.gymprogress.com

## Bezpieczeństwo

### Zabezpieczenia aplikacji:

- Autoryzacja JWT z ważnością tokenów

- Hasła przechowywane jako hash (bcrypt)

- Walidacja danych wejściowych

- Zabezpieczenie przed atakami XSS i SQL Injection

### Zalecenia bezpiecznego użytkowania:

- Używaj silnych, unikalnych haseł

- Nie udostępniaj linków do swoich danych treningowych

- Wyloguj się po zakończeniu sesji na publicznych urządzeniach

- Regularnie aktualizuj aplikację

---

\*Dokumentacja wygenerowana dla GYMPROGress. Ostatnia aktualizacja: 15/09/2025.\*