Workouter

Jan Baran

Link: <https://workouter.azurewebsites.net>

1. Opis projektu

Workouter to aplikacja webowa służąca do układania planów treningowych i zapisywania swoich wyników po wizycie na siłowni. Wykorzystuje ona bazę danych Neo4j dostępną w serwisie Neo4j AuraDB. Do jej stworzenia został użyty pythonowy framework Flask, który odpowiada zarówno za backend (zapytania do bazy danych), jak i frontend (renderowanie strony z uprzednio przygotowanych szablonów – plików .html).

1. Instrukcja obsługi

Przy pierwszym wejściu na stronę należy utworzyć konto w aplikacji:

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Widok 1. Rejestracja

a następnie się na nie zalogować:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Widok 2. Login

Po wykonaniu powyższych kroków wyświetli się następująca strona:

Obraz zawierający tekst, Czcionka, zrzut ekranu, biały

Opis wygenerowany automatycznie

Widok 3. Strona główna

Zakładka „Exercises” pozwala podejrzeć ćwiczenia, które są dodane na stałe do bazy danych oraz te, które zostały przez użytkownika utworzone. Ćwiczenie dodaje się za pomocą przycisku „Add Exercise”. Użytkownik może usunąć tylko stworzone przez siebie ćwiczenia.

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Widok 4. Ćwiczenia

Po wybraniu ćwiczenia użytkownik zostaje przeniesiony na jego stronę i tam wprowadza dane. Jeśli przykładowo zrobił on 3 serie po 10 powtórzeń ciężarem o wadze 20 kg, to musi 3 razy wcisnąć przycisk „Add set”, wpisując w „Reps” 10, a w „Weight (kg)” 20.

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Widok 5. Strona ćwiczenia - ilustracja przykładu

Gdy wprowadzone zostaną wszystkie dane, należy wcisnąć „Save sets” – sprawi to, że wynik od teraz będzie zapisany w bazie danych i wyświetlany na stronie w dziale „History”:

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Widok 6. Strona ćwiczenia - zapisane serie z 9 listopada

Wracając do zakładek, „My trainings” pozwala stworzyć użytkownikowy własny trening, czyli kontener, w którym może trzymać interesujące go ćwiczenia:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie

Widok 7. Moje treningi

Po wybraniu treningu należy dodać do niego ćwiczenia – robi się to poprzez wciśnięcie przycisku   
„Add Exercises”, wybraniu z listy interesujących użytkownika ćwiczeń (tych samych, które są w zakładce „Exercises”) i zatwierdzeniu operacji przyciskiem „Add”:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Widok 8. Strona treningu - wybór ćwiczeń do dodania

Ostatnią zakładką jest „Other trainings” – to w niej można zobaczyć wszystkie treningi, które stworzyli inni użytkownicy. Jeśli użytkownikowi spodoba się trening, może wcisnąć widniejący przy nim przycisk „follow” – sprawi to, że trening przestanie się tu wyświetlać, a zacznie w zakładce „My trainings”. Tak pozyskanych treningów nie da się edytować, ale można zapisywać w nich swoje wyniki siłowe.

Obraz zawierający tekst, linia, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

1. Diagramy

W Workouterze zastosowano następujący model danych:

1. Węzły:

* User – reprezentujący użytkownika
* Exercise – reprezentujący ćwiczenie
* Training – reprezentujący trening
* Completed – reprezentujący zapisane serie wykonanego ćwiczenia

1. Relacje:

* ADDED\_TRAINING – między użytkownikiem a treningiem (gdy użytkownik doda trening)
* FOLLOWS - między użytkownikiem a treningiem (gdy użytkownik „zafollowuje” trening innego użytkownika)
* ADDED\_EXERCISE – między użytkownikiem a ćwiczeniem (gdy użytkownik doda ćwiczenie)
* INCLUDES – między treningiem a ćwiczeniem (gdy trening zawiera ćwiczenie)
* INCLUDES\_SETS – między ćwiczeniem a zapisanymi seriami ćwiczenia

Diagram UML:

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznie

Diagram 1. Diagram UML

Przykładowy fragment grafu reprezentującego bazę danych:

Obraz zawierający krąg, Wielobarwność

Opis wygenerowany automatycznie

Diagram 2. Wizualizacja fragmentu bazy danych (Neo4j Workspace)

Kolory:

* Zielony – użytkownicy
* Różowy – treningi
* Niebieski – ćwiczenia
* Żółty – zapisane serie ćwiczeń

Zapisane serie ćwiczeń mają tę samą nazwę co użytkownicy, gdyż zapisują użytkownika, który wykonał ćwiczenie.

4. API

Endpointy (z pominięciem tych, których jedynym celem jest zwrócenie templatu ze stroną):

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Metoda HTTP** | **Opis** |
| /login | POST | Uwierzytelnia użytkownika (zwracany JWT token) |
| /register | POST | Tworzy konto nowemu użytkownikowi |
| /mytrainings\_protected | GET | Zwraca listę utworzonych i śledzonych treningów |
| /mytrainings | POST | Dodaje nowy trening przypisany do użytkownika |
| /mytrainings/delete/<int:id> | DELETE | Usuwa trening |
| /othertrainings\_protected | GET | Zwraca listę treningów, które są stworzone przez innych użytkowników i nieśledzone |
| /exercises | POST | Dodaje nowe ćwiczenie przypisane do użytkownika |
| /exercises\_protected | GET | Zwraca listę ćwiczeń, które są domyślnie w bazie lub użytkownik je dodał |
| /exercises/delete/<int:id> | DELETE | Usuwa ćwiczenie |
| /training/exercises/<int:id> | GET | Zwraca ćwiczenia zawierające się w treningu |
| /training/exercisesUnadded/<int:id> | GET | Zwraca ćwiczenia, które są dostępne w endpoincie „/exercises”, ale nie zawierają się w treningu |
| /training | POST | Dodaje ćwiczenie do treningu |
| /training/update | PUT | Usuwa ćwiczenie z treningu |
| /follow | POST | Dodaje trening jako śledzony dla konkretnego użytkownika |
| /unfollow | PUT | Usuwa trening ze śledzonych |
| /checkowner | POST | Zwraca informację czy użytkownik jest właścicielem treningu |
| /exercise/<int:id> | GET | Zwaca nazwę ćwiczenia, w którym można zapisać dane |
| /exercise | POST | Dodaje dane do ćwiczenia |
| /exercise/<int:id>/history | GET | Zwraca historię ćwiczenia |

5. Wdrożenie

Aby uruchomić projekt należy posiadać zainstalowany język Python w wersji 3.10 (na inne wersje twórca nie daje gwarancji) oraz zainstalować do niego biblioteki komendą:

*pip install -r requirements.txt*

Serwer uruchamia komenda:

*flask run*

Niestety, tak uruchomiona aplikacja nie będzie działać poprawnie, gdyż dane potrzebne do uruchomienia serwera (uri i hasło do AuraDB) są ukryte w zmiennych środowiskowych. Działająca aplikacja jest dostępna pod linkiem podanym na początku dokumentacji. Na platformę Azure jest wdrażana automatycznie dzięki Github Actions. Po założeniu konta użytkownik sam musi stworzyć swoje treningi, jedyne do czego ma dostęp to kilka stworzonych uprzednio ćwiczeń i pozostałe po testach aplikacji treningi innych użytkowników.