**Dokumentacja: Backend Brawl Stars**

**1. Plik app.js**

Plik app.js zawiera kod backendu napisany przy użyciu Express.js oraz Prisma do obsługi bazy danych.

**Główne Elementy:**

* Konfiguracja aplikacji Express oraz podstawowych middleware'ów.
* Ustanowienie połączenia z bazą danych Prisma.
* Definicja endpointów API oraz ich obsługa.

**Dependencies:**

* **express**: Biblioteka do tworzenia serwerów HTTP w Node.js.
* **cors**: Middleware umożliwiający CORS (Cross-Origin Resource Sharing) dla zapytań z frontendu.
* **@prisma/client**: Klient Prisma służący do komunikacji z bazą danych przy użyciu ORM (Object-Relational Mapping).
* **body-parser**: Middleware służący do parsowania ciała zapytań HTTP.

**Inicjalizacja aplikacji i połączenie z bazą danych:**

* Tworzenie instancji aplikacji Express (app).
* Inicjalizacja klienta Prisma (prisma) do obsługi operacji na bazie danych.

**Konfiguracja middleware'ów:**

* body-parser używany jest do parsowania ciała zapytań HTTP w formacie URL-kodowanym oraz JSON.
* express.static() służy do udostępniania zasobów statycznych (np. plików HTML, CSS, obrazków) z folderu public.
* cors jest używany do umożliwienia CORS tylko dla zapytań pochodzących z <http://localhost:3000>.

**Routing:**

* Utworzenie routera router dla ścieżki /api.
* Definicja endpointów HTTP (GET, POST) obsługujących różne żądania klienta.

**Funkcje pomocnicze:**

* getRandomBrawler(): Funkcja asynchroniczna do pobierania losowego Brawlera z bazy danych.

**Uruchomienie serwera:**

* Serwer nasłuchuje na określonym porcie (zmienna środowiskowa PORT lub domyślnie 4000).
* Plik app.js eksportuje aplikację Express, co pozwala na jej łatwe importowanie i testowanie w innych modułach.

**2. Testy Jednostkowe**

Plik tests.js zawiera testy jednostkowe dla endpointów API. Testy sprawdzają poprawność działania endpointów poprzez wykonywanie żądań HTTP i asercje wyników.

**Endpointy Testowane:**

1. **GET /api/brawlers-list**:
   * Sprawdza czy zwracana jest lista wszystkich postaci (Brawlerów).
   * Oczekiwany status: 200, Oczekiwany format odpowiedzi: JSON array.
2. **GET /api/brawlers**:
   * Sprawdza czy zwracane są nazwy wszystkich Brawlerów.
   * Oczekiwany status: 200, Oczekiwany format odpowiedzi: JSON array z nazwami.
3. **GET /api/start**:
   * Sprawdza czy gra rozpoczyna się poprawnie, wybierając losowego Brawlera.
   * Oczekiwany status: 200, Oczekiwana wiadomość: "Gra rozpoczęta! Spróbuj odgadnąć Brawlera."
4. **POST /api/guess**:
   * Sprawdza czy odgadywanie postaci działa poprawnie.
   * Oczekiwany status: 200, Oczekiwany format odpowiedzi: JSON z wynikami odgadywania.
5. **POST /api/help**:
   * Sprawdza czy zgłoszenie pomocy jest poprawnie obsługiwane.
   * Oczekiwany status: 200, Oczekiwana odpowiedź: Potwierdzenie utworzenia zgłoszenia.

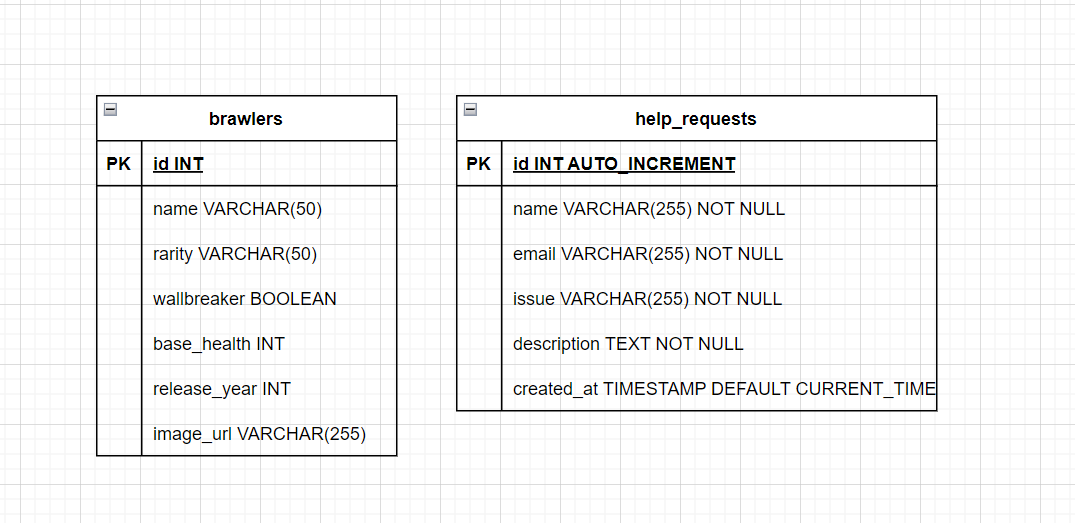
**3. Struktura Bazy Danych**

Plik SQL definiuje strukturę bazy danych oraz zawiera przykładowe dane dla tabeli brawlers:

**Tabela brawlers:**

* Kolumny: id, name, rarity, wallbreaker, base\_health, release\_year, image\_url.

**Tabela help\_requests:**

* Kolumny: id, name, email, issue, description, created\_at.

**Dokumentacja: Frontend Brawl Stars**

1. **Struktura projektu**

Frontend aplikacji Brawl Stars został zbudowany przy użyciu frameworka React.js, co umożliwia szybkie tworzenie dynamicznych interfejsów użytkownika.

1. **Pliki główne**
   * **index.js**: Plik główny, w którym renderowany jest główny komponent aplikacji.
   * **App.js**: Główny komponent aplikacji, odpowiedzialny za renderowanie innych komponentów oraz zarządzanie stanem aplikacji.
2. **Komponenty**
   * **Navbar.js**: Komponent odpowiedzialny za nawigację po aplikacji. Zawiera linki do różnych widoków.
   * **Home.js**: Komponent strony głównej, wyświetlający informacje o grze i przycisk rozpoczęcia.
   * **BrawlerList.js**: Komponent wyświetlający listę postaci (Brawlerów) w grze.
   * **GuessBrawler.js**: Komponent umożliwiający użytkownikowi zgadywanie postaci (Brawlerów) na podstawie podpowiedzi.
   * **HelpForm.js**: Komponent formularza zgłaszania pomocy, który umożliwia użytkownikowi wysłanie zgłoszenia o potrzebie pomocy.
3. **Biblioteki i narzędzia**
   * **React Router**: Wykorzystano React Router do nawigacji między różnymi widokami aplikacji.
   * **Axios**: Do wykonywania żądań HTTP do backendu wykorzystano bibliotekę Axios.
   * **React Context API**: Zastosowano API kontekstu React do zarządzania stanem aplikacji i przekazywania danych między komponentami.
   * **SweetAlert2**: Biblioteka SweetAlert2 użyta do wyświetlania interaktywnych powiadomień i komunikatów dla użytkownika.
   * **React Select**: Do stworzenia interaktywnego pola wyboru w formularzu zgadywania postaci użyto biblioteki React Select.
   * **JS-Confetti**: Wykorzystano JS-Confetti do wyświetlenia efektu konfetti po trafnym zgadnięciu postaci.
4. **Stylowanie**
   * Do stylowania interfejsu użytkownika wykorzystano arkusze stylów CSS oraz pliki SCSS dla większej modularności i łatwiejszego zarządzania stylami.
   * Stylowanie zorganizowane zostało w sposób modułowy, aby każdy komponent miał swoje własne style i uniknąć konfliktów w nazwach klas CSS.

Ta dokumentacja podsumowuje strukturę i główne elementy frontendu aplikacji Brawl Stars, obejmując użyte technologie, komponenty oraz narzędzia wykorzystane w jego tworzeniu.