# **Dokumentacja**

**Opis aplikacji**

Nie wyszło, przyznaję się, nie wyrobiłam się, więc opisuje to co by było jakby działało z tymi danymi i komponentami co mam.  
Aplikacja jest przeznaczona dla graczy gier z serii pokemon, chcących sprawdzić różne statystyki, w jakich lokalizacjach są dostępne pokemony, oraz dowiedzieć się więcej o strategii walki w grach.

Ze względu na to, że ta seria istnieje już od prawie 30 lat (w Japonii pierwsze gry wyszły w ‘96), a gier jest ponad 100, i danych mnóstwo, aplikacja jest w pełni uzupełniona dla pierwszej serii/generacji, w dużej mierze uzupełniona dla drugiej, i częściowo od 3 do 6 generacji, dla następnych uzupełnione są tylko ruchy, nowy typ “fairy”, oraz pokemony które ewoluują z tych z poprzednich generacji. Część mechanik niestety nie jest utworzona.

Był też plan zrobić dodatkowo interaktywne mapki dla gier kilku pierwszych generacji, jest to coś czego jeszcze nigdzie nie ma na żadnej z podobnych stron, jest tylko dla nowszych gier. W mojej aplikacji jednak mapek nie ma, ale są pod nie wyświetlane gotowe dane lokalizacje i ceny w sklepach, lokalizacje pokemonów i dokładne o nich dane jak przedział poziomów, szansa, i gdzie dokładnie są (trawa/woda piętra budynków i tak dalej). Niezależnie od projektu (jest 30.05 i wiem że się nie wyrobię) zrobię kiedyś taką interaktywną mapkę.

Aplikacja nie działa, żeby zademonstrować że coś jednak jest, przywróciłam testowe hardcoded dane które wyświetlają się poprawnie, komponentów gotowych do użycia też jest całkiem sporo. Faktycznie chciałam żeby to działało, widać po ilości TODO, że powoli pracowałam nad tym ale wyszło jak wyszło.

W helpers/displayImagesTest.js jest działająca część aplikacji do wyświetlania obrazków pokemonów.

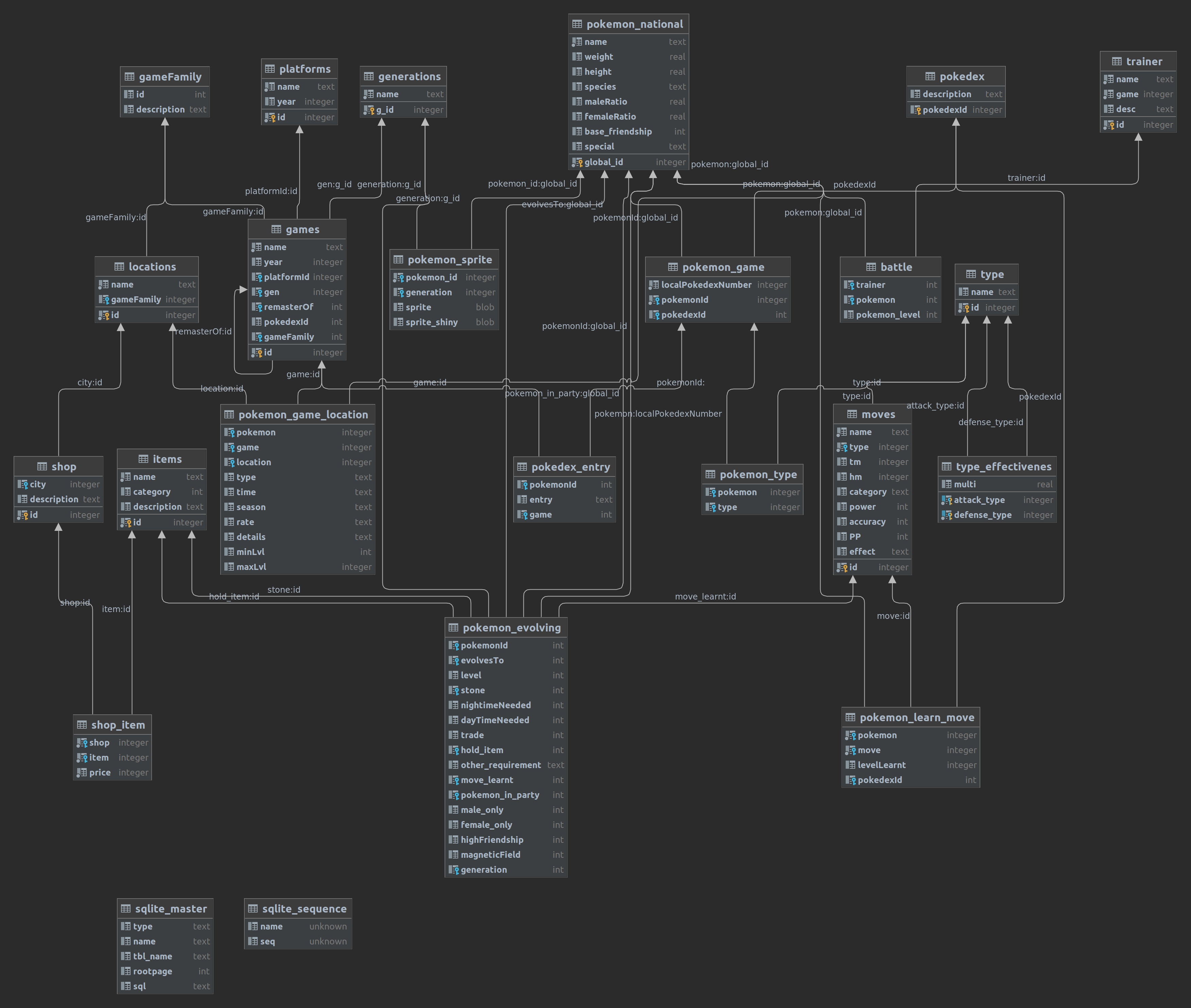
Baza: sqlite

z bardziej imponujących i mniej oczywistych: pokemon\_game\_location ma wszystkie możliwe dane o pokemonach w danej lokalizacji, tabela moves jest uzupełniona aż do 9 generacji, pokemon\_evolving (uzupełniona 1 i 2 gen.) ma wpisane wszystkie metody, i ich kombinacje poziom, dzień, noc, wszystko. W pokemon\_sprite są miniaturki wczytane jako blob, jest ich dużo, dlatego baza tyle waży.  
ze względu na małe różnice pomiędzy niektórymi grami, przykład 3 generacji:  
pokemon ruby, sapphire, emerald, firered,leafgreen  
ruby i sapphire się wzajemnie uzupelniają jeśli chodzi o dostępne pokemony i lokazlizację, emerald ma wszystko, firered i leafgreen to remastery gier z 1 generacji  
traktuje to tak: samo games to każda osobna gra, gamefamily to te najbardziej podobne do siebie np “reb/blue” koło siebie ale yellow osobno, tabela pokedex to też gry ale już całe np. ruby/sapphire/emerald jako jedno, ponieważ mają takie same pokedexy, generations to są tak naprawdę regiony w których rozgrywa się akcja, “kanto”, “johto” itp.  
Tabele “trainer” i “battle” są puste, z załozenia miałyby mieć dane o wrogich trenerach i ich pokemonach w każdym ze spotkań z nimi. Pokemon\_national to najbardziej podstawowe dane, nazwa, i tak dalej, pokemon\_game zawiera dane o numerach pokemona w różnych pokedexach, pokedex\_entry o wpisach o danym pokemonie w każdej grze, pokemon\_type zawiera dane jaki typ lub dwa ma dany pokemon, a type\_effectiveness o tym nad jakimi typami ma przewagę dany typ pokemona. Dodatkowe np. “6789” oznacza dla których generacji jest to prawdziwe, ponieważ przewagi typów częściowo się zmieniały.  
Pokemon\_learn\_move przechowuje dane na jakim poziomie pokemon uczy się danego ruchu w danej grze.

**WAŻNE** dane oczywiście nie były wpisywane ręcznie, tylko za pomocą programów w helpers/ i innych tego typu.

Style: jest dużo więcej media query niż bym chciała, ale strona wyświetla się ok, chyba od szerokości 300px, realistycznie nikt nie będzie miał węższego telefonu. Baner przy węższych ekranach się zmienia na mniejszy. Pobawiłam się trochę ngrok i posprawdzałam wyświetlanie na telefonach, jest ok, **jeśli coś wyświetla się dziwnie to raczej kwestia devtoolsa bo na samych urządzeniach działa**. Jedyny problem przy wyświetlaniu długiej tabelki z mnożnikami typów.

**Dokładniejszy opis tego co jest: na stronie głównej aplikacji.**



**Schemat jest w osobnym pliku png w /docs/schema1.2.png**

Wiele zapytań jest bardzo długich, są w osobnym pliku a raczej plikach, w backend/queries.js są w formie krótkich funkcji zwracających zapytanie ze wstawionym argumentem najczęściej id, wiem o istnieniu formatowania z “?”, ale wiele z tych zapytań jest bardzo długich, nawet i po 50 linijek w sql wychodziło, więc nie chce tego wstawiać ich do głównego backendu, w queries.js są one tak sformatowane że wszystko jest w jedej linijce i za wiele nie widać, w backend/queries.sql są w bardziej czytelnej formie dokładnie te same (w webstormie w zwykłym sqlu działa używanie {id} jako parametru do wpisania, bez odpalania js).

Najbardziej skomplikowane które nie były wykorzystane: zapytanie zwracajace cały łańcuch ewolucji.

Query\_getFamily zwraca czy pokemon jest ewolucją, oraz czy ma ewolucję, to jest po to żeby potem tworzyć całe łańcuchy ewolucji i je wyświetlać, jeśli jest ewolucją, to w pokemon\_evolution (z założenia bo nie mam niestety tej mechaniki) szuka tego pokemona z którego ewoluuje, i dalej drzewko ewolucji jest robione dla tego pierwszego.

Query\_getMoveset, byłoby do wyświetlania ruchów na stronie, według generacji/rodzin gier bo się różnią, tutaj ignoruję drobne różnice pomiędzy np. red/blue a yellow ponieważ zdecydowana większość danych jest dokładnie taka sama.

**Testy**

**uruchamiać za pomocą mocha, nie jest**

teststatus.js, są celowo zostawione błędy dla dundunsparce i kilku innych, ponieważ w tabeli pokemon\_types są dane tylko (od 1) do 6 generacji, (do id 697), dlatego jest “TypeError: Cannot read property 'type1' of undefined” dla: Iron Treads, Type: Null (tak, to taki pokemon), Mr. Rime oraz Dudunsparce, i byłby ten sam błąd dla wszystkich innych z id większym niż 697.

Nie działało dla pokemonów które mają 2 typy, problem był banalny, nazwy kolumn w zapytaniu dla pokemonów z 2 typami były inne. (muszą być 2 zapytania, bo to drugie uzywa cte i row number partition by żeby rekordy nie były podwójnie).

Nadal nie działa dla Mime Jr., błąd jest w tym samym miejscu, uruchamiane osobno zapytanie działa poprawnie.

w basic functions jest błąd   
Expected :Farfetch'd  
Actual :Farfetch''d  
tak ma być, bo dodatkowy apostrof jest potrzebny żeby zapytanie działało

testy frontendu wyglądają na to że nie działają przez błędy frameworka, jest coś nie tak z konfiguracją i jest “unexcpected token <” ale i tak napisałam jaieś testy.

