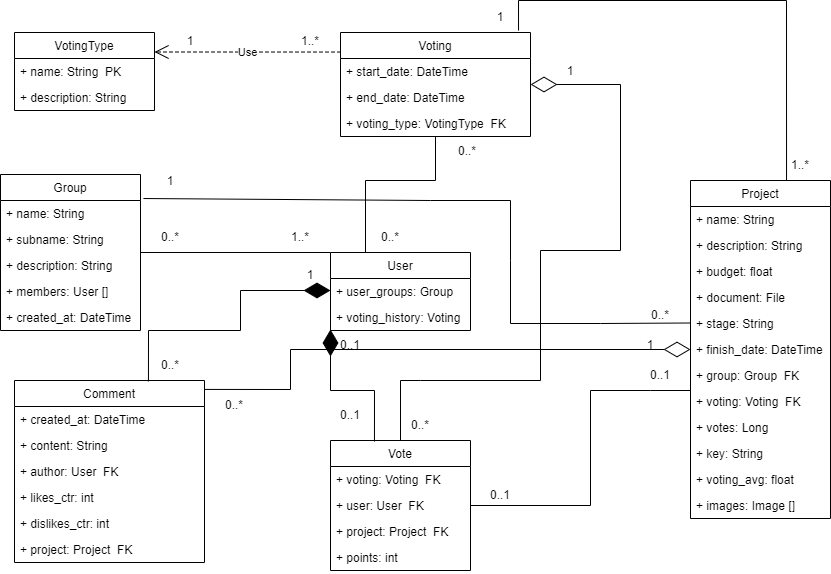
6. Architektura systemu

6.1 Schemat klas (diagram UML)

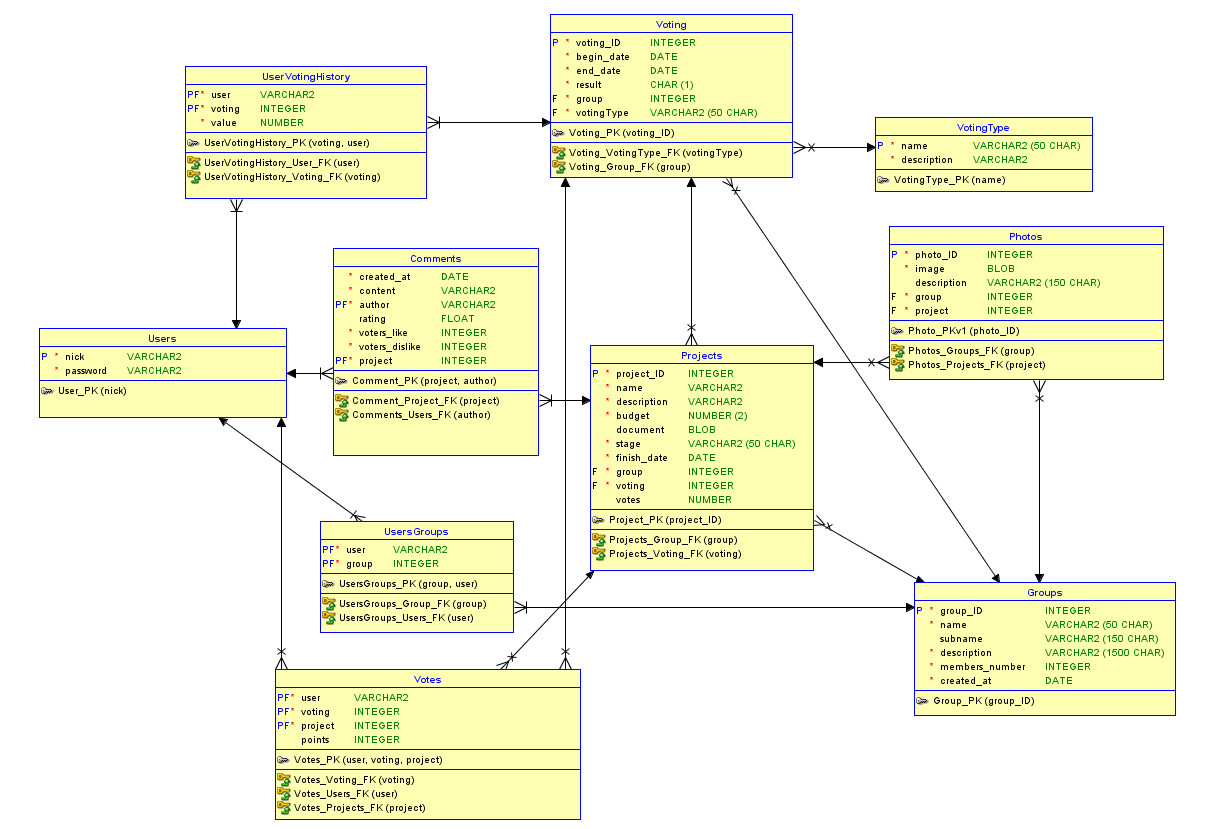


Przedstawiony powyżej schemat klas pokazuje charakterystykę połączeń i zależności pomiędzy konkretnymi klasami. Kluczowe związki na schemacie to:

* relacja User – Vote (Użytkownik – Głos): w tym przypadku typ relacji, który został użyty to agregacja całkowita, czyli kompozycja. Jest to efektem podrzędności głosu względem użytkownika. Innymi słowy: Każdy głos ma swojego właściciela, a ten może nim zarządzać   
  (usunąć, zmienić);
* relacja User – Comment (Użytkownik – Komentarz): w tym związku również użyto kompozycji z tych samych względów co powyżej. Użytkownik jest właścicielem i twórcą komentarza, ma nad nim „władzę” oraz może podjąć działania edycji lub wycofania, gdy uzna to za stosowne;
* relacja Voting – Vote (Głosowanie – Głos): użytym typem relacji jest agregacja częściowa, czyli „poziom” niżej od kompozycji jeśli chodzi o siłę związku. W tym przypadku właściciel czyli klasa Voting nie zarządza, a jedynie korzysta z klasy Vote. Głosy są składowymi, ale nie obowiązkowymi elementami głosowania (dopuszczalny, choć mało prawdopodobny jest przypadek, że w głosowaniu nie weźmie udziału żaden użytkownik). Całkowitym właścicielem Głosu jest wyżej wymieniony Użytkownik;
* relacja Project – Comment (Projekt – Komentarz): podobnie jak w powyższym przykładzie, klasy powiązane są poprzez agregację częściową. W zasadzie jest to bliźniaczy przypadek do relacji Głosowanie – Głos . Wynika to z faktu, iż projekt nie musi zawierać komentarzy, ale gdy to robi, jest nad nimi nadrzędny. Natomiast nie jest ich twórcą, ani zarządcą. Taką rolę pełni klasa Użytkownik, mający pełnię praw do rozporządzania komentarzem;
* relacja Voting – VotingType (Głosowanie – Typ Głosowania): połączenie pomiędzy klasami o najsłabszej mocy, zależność. Głosowanie musi mieć swój typ, ale nie tworzy nowej instancji tego typu. Oznacza to, że nie może nim zarządzać, ani go zmieniać.

Pozostałe połączenia pomiędzy klasami są przykładami wdrożenia asocjacji – obiekty danej klasy wykorzystują przez pewien dłuższy czas obiekty innej klasy, tak jak to widać na schemacie.

6.2 Schemat bazy danych



Baza danych składa się z jedenastu tabel. Zawiera informacje między innymi na temat użytkowników, projektów czy grup użytkowników. Dzięki odpowiednim relacjom, baza przechowuje historię głosowania użytkowników, co pozwala każdemu z nich podejrzeć, które z projektów najbardziej przypadły im do gustu. Poniżej, zaprezentowany jest opis tabel:

* Users – najważniejsza tabela w bazie danych. Przedstawia konta użytkowników składające się z loginu (nicku), będącego jednocześnie kluczem prywatnym (zatem nie mogą istnieć dwa konta o takim samym pseudonimie) oraz hasła;
* Groups – zawiera informacje na temat grupy użytkowników, czyli przestrzeni reprezentującej daną społeczność lokalną. Oprócz nazwy, zawiera opis, liczbę członków czy datę utworzenia;
* Projects – tabela przechowująca informacje na temat projektu, na przykład wymagany budżet, opis, galerię. Połączona jest z tabelami Groups i Voting, ponieważ każdy projekt jest wybierany w obrębie danej grupy oraz danego głosowania;
* Voting – magazyn informacji na temat głosowań w bazie danych. Kluczem podstawowym jest identyfikator, łączy się z Grupą, do której głosowanie, jest przypisane oraz Typem Głosowania, które jest nieodłączną jego częścią. Oprócz kluczy, zawiera jeszcze daty rozpoczęcia i zakończenia a także wynik;
* VotingType – tabela przedstawiająca różne typy, formy głosowań (preferencyjne, większościowe itp.). Zawiera tylko dwa pola: nazwę i opis. Typy rozróżniane są na podstawie ich nazw, są one unikalne;
* Votes – tabela reprezentująca głosy użytkowników. Nie mogą zostać pomylone, nie są również dwuznaczne. Kluczem podstawowym każdego głosu jest nazwa użytkownika, identyfikator głosowania oraz identyfikator projektu;
* Comments – w tej tabeli przechowywane są komentarze użytkowników na temat projektów. Rozróżniane są na podstawie klucza (autor, identyfikator projektu), gdzie poprzez autora rozumie się nazwę użytkownika. Komentarz zawiera również takie pola jak content czyli treść, created\_at, czyli data utworzenia, rating tj. ocena w skali od 0 do 5 z krokiem 0.5 oraz dwa pola przechowujące informację dotyczącą liczby ocen pozytywnych i negatywnych danego komentarza uzyskanych od innych udzielających się osób;
* Photos – tabela przechowująca galerię zdjęć. Każdy rekord zawiera ma swój identyfikator, jest przypisany do konkretnego projektu i grupy, a także oczywiście instancję załadowanego pliku ze zdjęciem. Dodatkowo, jest jeszcze pole z opisem, które nie jest obowiązkowe do prawidłowego wypełnienia tabeli;
* UserVotingHistory – tabela tworząca historię głosowań każdego użytkownika. Dzięki niej użytkownik może sprawdzić na swoim profilu, na które projekty oddał głosy. Rekordy rozróżnia klucz (nazwa użytkownika, identyfikator głosowania). Oprócz klucza, tabel zawiera pole value, reprezentujące identyfikator projektu, na który dany użytkownik zagłosował;
* UsersGroup – tabela złożona z rekordów w postaci użytkownik – grupa, mająca na celu w krótkim czasie uzyskać informację o tym, do których grup należy dany użytkownik. Rekordy nie mogą się powtórzyć, ponieważ oba pola tworzą klucz podstawowy. Wynika to oczywiście z faktu, że każda osoba do tej samej grupy może dołączyć tylko raz.

6.3 Komunikacja frontendu i backendu (REST API)

Źródło: <https://zaprogramujzycie.pl/czym-jest-api/>

Źródło: https://zaprogramujzycie.pl/czym-jest-rest/

Zanim przejdziemy do wytłumaczenia tego, w jaki sposób nasza aplikacja frontendowa łączy się z backendową, należy wyjaśnić kilka rzeczy.

REST, czyli Representational State Transfer jest formatem przekazywania danych wykorzystywanym w pośrednictwie pomiędzy klientem a serwerem. Format skonstruowany jest tak, aby obie strony potrafiły zrozumieć i przetworzyć dane. REST stał się bardzo popularnym rozwiązaniem architektonicznym dzięki swojej prostocie, szybkości i uniwersalności, a jego implementacja jest możliwa w wielu językach.

API oznacza Application Programming Interface. Jest to zwyczajny pośrednik, którzy otrzymuje dane od klienta i przekazuje je do serwera (i w drugą stronę), wykorzystując wybrany format prezentowania danych.

System SYLKOS w komunikacji pomiędzy swoimi aplikacjami wykorzystuje REST API.