POLITECHNIKA WROCŁAWSKA

WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA (INF)

SPECJALNOŚĆ: INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH (INS)

INTERNETOWE BAZY DANYCH

TYDZIEŃ DRUGI

AUTOR:

Krzysztof Dombek

PROWADZĄCY PRACĘ:

Dr inż. Paweł Głuchowski

Autorzy:

Krzysztof Dombek 226093

Łukasz Broll 225972

WROCŁAW, 2018

**Spis treści**

[1. Diagramy 3](#_Toc529464016)

[1.1. Diagram konceptualny 3](#_Toc529464017)

[1.2. Diagram logiczny 4](#_Toc529464018)

[1.3. Diagram fizyczny 4](#_Toc529464019)

[1.4. Diagram klas 5](#_Toc529464020)

[1.5. Diagram ERD 6](#_Toc529464021)

[2. Analiza ilości instancji 7](#_Toc529464022)

[2.1. Liczba instancji dla każdej encji 7](#_Toc529464023)

[2.2. Operacje CRUD 8](#_Toc529464024)

[3. Dostęp oraz poprawność integralności 9](#_Toc529464025)

[3.1. Dostęp do bazy danych 9](#_Toc529464026)

[3.2. Analiza i poprawa integralności bazy danych 10](#_Toc529464027)

[4. Analiza i poprawa wydajności bazy 10](#_Toc529464028)

[4.1. Procedury 10](#_Toc529464029)

[4.2. Indeksy 10](#_Toc529464030)

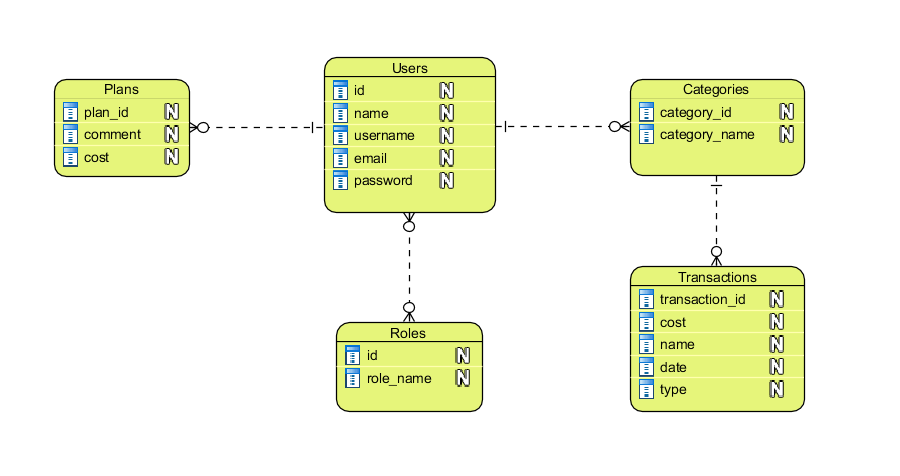
[4.3. Normalizacja 11](#_Toc529464031)

[4.4. Optymalizacja zapytań 11](#_Toc529464032)

# Diagramy

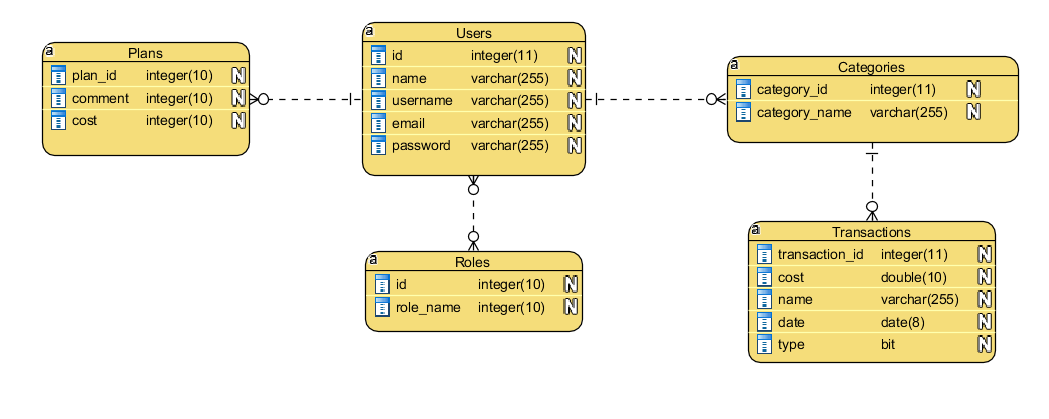
* 1. Diagram konceptualny

Pierwszy z modelów ERD. Powstał na podstawie zebranych wymagań biznesowych. Tabele wraz z relacjami zostały zaprojektowane zgodnie z potrzebami firmy. Na tym etapie, projekt bazy danych pod względem technicznym nie jest jeszcze rozważany.



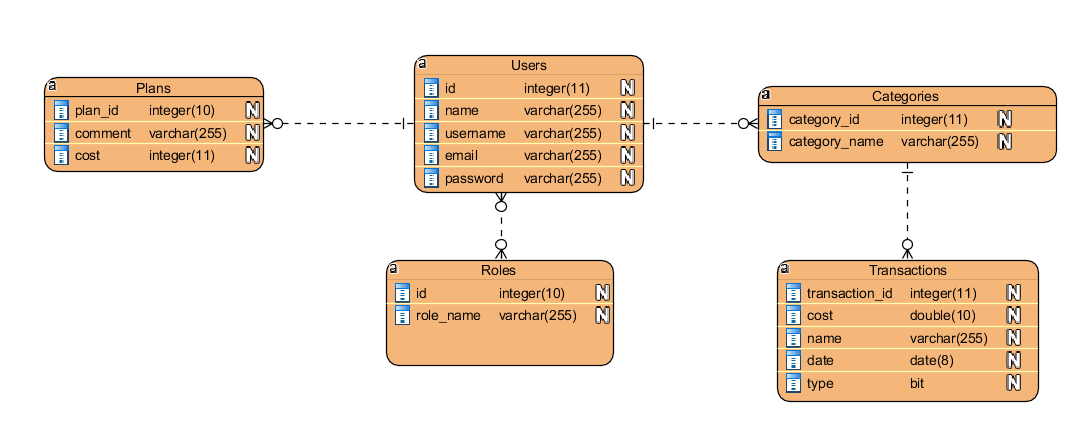
* 1. Diagram logiczny

Model logiczny ERD również odpowiada za modelowanie wymagań biznesowych, jednak jest on bardziej złożony niż model konceptualny, ze względu na określone typy kolumn w bazie danych.



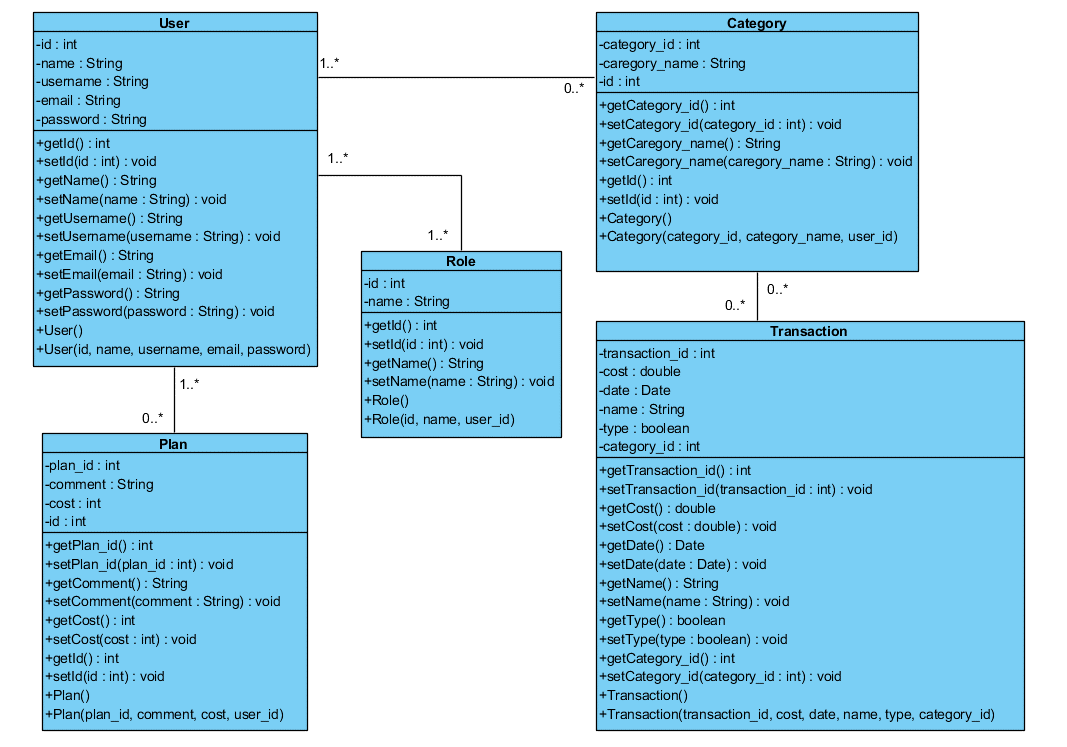
* 1. Diagram fizyczny

Model fizyczny ERD odwzorowuje faktyczny model bazy danych zgodnych ze standardami DBMS (*Database Management System*).



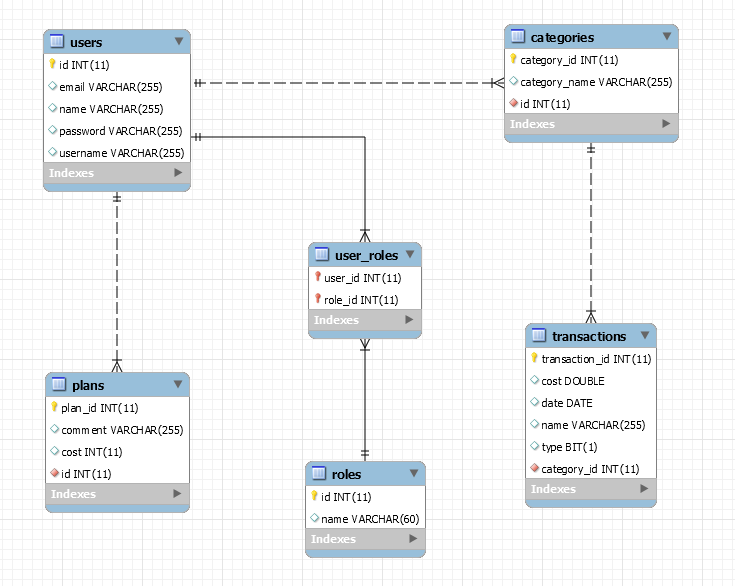
* 1. Diagram klas

Na podstawie powyższych diagramów oraz analiz, stworzony został model w postaci klas, z kodem źródłowym napisanym w języku Java. Wykorzystując framework Hibernate, który odpowiedzialny jest za realizacje warstwy dostępu do danych, klasy zostały zmapowane za pomocą adnotacji oraz definicji odpowiednich połączeń, jak zaprezentowanona diagramie poniżej.



* 1. Diagram ERD

Na poniższych schemacie znajduję się ostateczny model diagramu, po wygenerowaniu schematu za pomocą mapowania klas oraz adnotacji Hiberante.



W bazie danych zostały wykorzystane następujące typy relacji:

- jeden-do-wielu,

- wiele-do-wielu.

Relacja wiele-do-wielu została wykorzystana w przypadku nadawania roli każdemu z użytkowników. System będzie posiadał dwie role: ADMIN oraz USER. Domyślnie każdemu użytkownikowi po rejestracji nadawana będzie rola USER, Tylko użytkownik z prawami ADMIN będzie mógł nadać innemu użytkownikowi wyższe uprawnienia.

Relacja ta została zastosowana z myślą o rozwój systemu. W przypadku, gdy potencjalny klient chciałby wyszczególnić więcej ról dla poszczególnych użytkowników, będzie możliwe dodanie ich w prosty sposób.

Relacja jeden-do-wielu znajduję swoje zastosowanie w przypadku, gdy mowa o poszczególnym koncie użytkownika. Dany użytkownik może posiadać wiele kategorii, a jedna konkretna kategoria, może posiadać wiele transakcji. To samo dotyczy notatek/planów związanych z wydatkami użytkownika.

1. Analiza ilości instancji

Baza składa się z 5 tabel głównych oraz z jednej tabeli wygenerowanej jako pomocnicza tabela w przypadku relacji wiele-do-wielu. W przypadku tabeli Users, będzie ona wykorzystywana przy rejestracji oraz przy logowaniu. Istotnym faktem jest, iż każdy użytkownik będzie posiadał listę kategorii/transakcji/planów związanych z wydatkiem, które nie będą dostępne dla innych użytkowników systemu, ponieważ są to dane prywatne. Wiąże się to z faktem, iż dowolne zapytanie do bazy będzie wymagało id użytkownika, który aktualnie jest zalogowany w aplikacji. Problem ten rozwiązany zostanie za pomocą tokena JWT (Json Web Token), który zagwarantuje bezpieczeństwo REST’owych endpointów oraz umożliwiać będzie uzyskanie informacji o zalogowanym użytkowniku.

* 1. Liczba instancji dla każdej encji

Tabela USERS odpowiada za przychowywanie użytkowników. Liczba instancji w tej tabeli zależna jest od ilości osób chcących korzystać z aplikacji, w tym przypadku przyjmiemy, że będzie to liczba około 5 tysięcy.

Tabela ROLES odpowiada za przechowywanie ról zdefiniowanych w systemie oraz przyznawanie ich każdemu użytkownikowi zakładającemu konto. Według wymagań, powinna ona zawierać dwie role – ROLE\_USER oraz ROLE\_ADMIN. Jak wcześniej zostało wspomniane, relacja wiele-do-wielu umożliwia proste dodania ról w przyszłości. W tabeli głównej znajdować się będą 2 rekordy z rolami. W tabeli pośredniczącej (przy tworzeniu relacji wiele-do-wielu) dodane będzie dokładnie tyle samo rekordów co do tabeli USERS, gdyż przetrzymuje ona informacje o roli każdego z użytkowników.

Tabela CATEGORIES szacowana jest na kilkanaście/kilkadziesiąt tysięcy rekordów. Każdy z użytkowników będzie posiadać kategorie w ilości od 3-10. Podobnie tabela PLANS, w której użytkownik będzie gromadził swoje notatki/przypomnienia, tj. co powinien kupić oraz za jaką, oszacowaną wcześniej kwotę.

Tabela TRASNACTIONS będzie zawierała najwięcej rekordów. Uwzględniając przyjęte założenia – liczba użytkowników 5 tysięcy, liczba kategorii 20 tysięcy – liczba transakcji będzie wynosiła kilkaset tysięcy rekordów. Każda transakcja musi być przypisana do kategorii.

* 1. Operacje CRUD

Podstawowe operacje na bazie:

* Create
* Read
* Update
* Delete

W systemie główną funkcjonalnością będzie wprowadzanie danych, które umożliwiać będzie generowanie zestawień wynikowych z danych okresów czasów czy swobodne przeglądanie wpisów użytkownika uwzględniając datę ich powstawania. Na tej podstawie szacuje się, iż operacje usuwania będą zdecydowanie najrzadziej wykonywanymi operacjami. Na kolejnym miejscu znajdzie się operacja modyfikacji. Z racji faktu, iż wpisy do aplikacji dodajemy po wykonanej transakcji, operacja modyfikacji wykorzystywana będzie tylko w razie niepoprawnego wprowadzenia danych do systemu. Kolejną operacją będzie odczyt, który wykorzystywany będzie zdecydowanie częściej. Użytkownik będzie miał wgląd w swoje transakcje przypisane do odpowiednich kategorii. Zdecydowanie najczęściej wykonywaną operacją będzie dodawanie transakcji. Szacuje się, że każdy z użytkowników każdego dnia doda przynajmniej 5 transakcji.

Na podstawie powyższej analizy wynika, że tabela TRANSACTIONS będzie wykorzystywane najczęściej. Tabela CATEGORIES oraz PLANS mają podobne przeznaczenie, na podstawie którego oszacowano, że najczęściej wykonywane na nich operacje to read oraz create. Tabela ROLES wykorzystywana będzie równie często co tabela USERS, przy operacjach create/read/delete oraz ewentualnie update.

1. Dostęp oraz poprawność integralności
   1. Dostęp do bazy danych

Dostęp do bazy danych zaimplementowany zostanie za pomocą framework’a Spring,   
a konkretnie Spring Data Repository. Narzędzie to bazuje na JPA (*Java Persistence API*).

Dzięki odwzorowaniu tabel z postaci klas Javy, będzie można w prosty sposób wykonywać zapytania na bazie. Spring Data udostępnia szereg podstawowych zapytań typu findBy{Query}, gdzie Query to nazwy operacji i pól na których są wykonywne. Jest to bardzo prosty i dobry sposób, aby zapytania były efektywne oraz proste w implementacji.

W przypadku bardziej skomplikowanych zapytań, skorzystać można z adnotacji *@Query*, która w prosty sposób pozwala zaimplementować zapytanie, które dana funkcja ma wykonać.

Korzystanie z Spring Data Repository wymaga stworzenia interfejsu, który dziedziczyć będzie po CrudRepository<>, następnie implementacji serwisu, w którym znajdować będą się ciała poszczególnym funkcji, a na końcu implementacji kontrolera, który korzystając z funkcji serwisowych, będzie zwracał odpowiednie wyniki.

Na podstawie powyższych stwierdzeń, wywnioskować można następujące wymagania:

* Każdy użytkownik będzie posiadał dostęp jedynie do swoich (wprowadzonych przez siebie) danych,
* Użytkownik będzie musiał stworzyć konto, a potem zalogować się, aby uzyskać dostęp do funkcjonalności CRUD na bazie danych,
* Baza powinna być prosta w rozbudowie,
* Zapytania na bazie danych powinny zostać zoptymalizowane, w celu zmiejszenia czasu wykonywania każdego z nich,
* Osoby nieupoważnione nie mają dostępu do danych,
* W celu ochrony danych, baza danych powinna mieć backup raz na 6 godziny,
  1. Analiza i poprawa integralności bazy danych

Baza danych powinna spełniać każdą z integralności:

* Wartość danych spełniają wcześniej zdefiniowane i założone ograniczenia,
* Każda tabela posiada klucz główny, a jego wartości w tabelach są unikatowe,
* Każda wartość klucza obcego jest równa jakiejś wartości klucza głównego w powiązanej tabeli,

Każdy użytkownik sam zarządza swoimi zasobami w aplikacji, przez co zaimplementowany został system usuwania kaskadowego. Związane jest to z faktem, iż gdy użytkownik usunie kategorię, w której znajdują się transakcje ściśle z nią powiązane, wszystkie transakcje również zostaną usunięte bezpowrotnie.

1. Analiza i poprawa wydajności bazy
   1. Procedury

W przypadku naszego systemu, w pierwszej wersji nie przewiduje się implementacji żadnych procedur. Konieczne operacje, które wymagać będą wykonywania większej ilości operacji jednocześnie, zostaną zaimplementowane w języku Java.

Przykładem procedury, którą system będzie posiadał, jest wcześniej wspomniane usuwanie kaskadowe.

* 1. Indeksy

Indeksy zostaną nałożone na klucze główne w celach przyspieszenia wyszukiwania danego rekordu w bazie danych. Wspomagać one będą operacje INSERT, UPDATE, DELETE oraz istotne będą przy sortowaniu, np. według daty w przypadku analizy wydatków.

* W tabeli USERS indeks nałożony został na pole **username** z powod wykorzystania mechanizmu generującego token JWT, który przechowuje między innymi nazwę użytkownika, który loguje się do aplikacji,
* W tabelach CATEGORIES oraz TRANSACTIONS indeksy nałożone zostały na **category\_id** oraz **transaction\_id** ze względu na częste wyszukiwanie po tych polach.
  1. Normalizacja

Tabele w bazie danych powinny spełniać wymogi związane z normalizacją. Każda z tabel powinna opisywać jeden obiekt. Wartości poszczególnych pól powinny być elementarne. Tabela nie powinna zawierać kolekcji. Kolejność wierszy może być dowolna.

* 1. Optymalizacja zapytań

W projekcie nie jest planowane zastosowanie dodatkowych funkcji lub narzędzi do optymalizacji zapytań. Framework Spring oraz Hibernate oferują zoptymalizowane metody oraz zapytania, dzięki czemu nie ma konieczności ręcznego implementowania tych metod.

Wcześniej wspomiane zapytania typu findBy{Query} udostępnianie przez Spring Data są na tym etapie wykorzystywane.