Dokumentacja projektu

"Implementacja modułu bezpieczeństwa dostępu do danych (na poziomie encji) w zależności od przypisanej roli użytkownika. Moduł powinien wykorzystywać programowanie aspektowe, pozwalające na zastosowanie w dowolnym systemie, gdzie wykorzystano mapowanie O-R w technologii JEE lub Spring. Role systemu posiadają strukturę drzewiastą"

Design Patterns, Informatyka rok III, semestr 5

Kamil Bienek, Paweł Hanzlik, Piotr Paruch, Norbert Zagożdżon

Spis treści

1. Cel i założenia projektu

Celem projektu jest zaprojektowanie i implementacja modułu bezpieczeństwa do danych na poziomie encji. W założeniu ma to być biblioteka zapewniająca nakładkę na zapytania bazodanowe SQL.

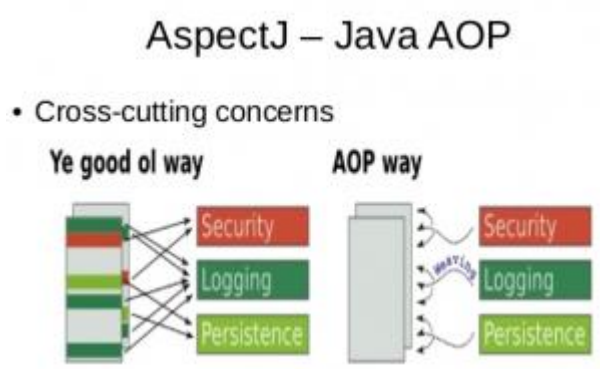
2. Opis komponentów

- **Bezpieczeństwo w bazach danych na poziomie encji**

Każda szanująca się firma powinna zapewnić, że jej baza danych jest zabezpieczona. Nie można dopuścić do sytuacji, gdy nowo zatrudniony pracownik na niskim stanowisku będzie miał dostęp do podglądu finansów firmy. Naszym zadaniem jest zapobieganie właśnie takim sytuacjom poprzez ograniczanie dostępu do bazy w zależności od sprawowanego stanowiska.

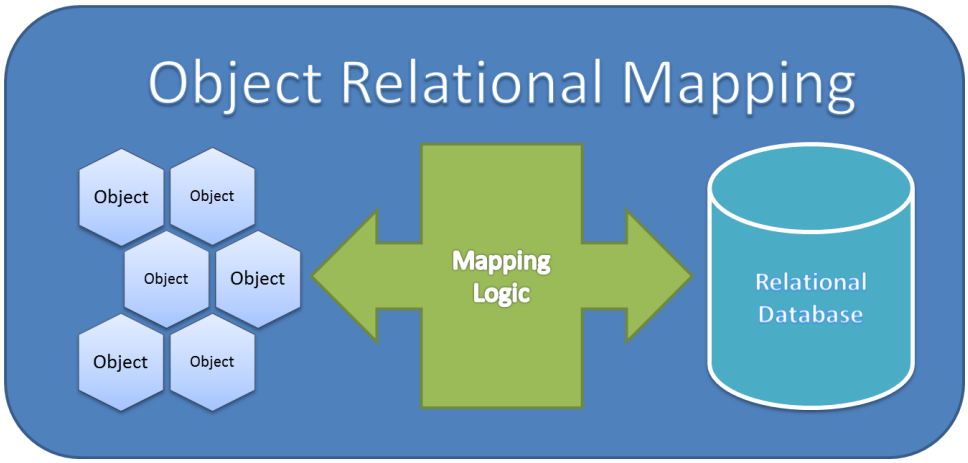
- **Programowanie aspektowe**

Programowanie aspektowe to paradygmat, który popiera separację zagadnień na jak najmniejsze części. Taki sposób implementowania sprawia, że moduły są łatwiej adaptowane w nowym projekcie i bardziej kompatybilne, tj. Mniej rzeczy należy zmienić. Dzięki zastosowaniu aspektowości kod staje się bardziej czytelny i mniej podatny na wystąpienie błędów, gdyż zamiast fizycznego przeplatania w programie kilku zagadnień, oddzielamy je w poszczególnych aspektach, zaś połączenie staje się logiczne.



- **Mapowanie O-R**

Mapowanie Obiektowo-Relacyjne jest to sposób odwzorowywania relacji w bazie danych na obiekty danej klasy w języku programowania i na odwrót. Poza samą zamianą możemy dzięki niemu również wykonywać operacje na danym z relacyjnej bazy danych jak na najzwyklej zaimplementowanych w kodzie obiektów klas wybranego języka programowania. Takie rozwiązanie jest niezwykle wygodne i potrafi zaoszczędzić cennego czasu i środków potrzebnych do obsługi tego typu problemów ręcznie bez mapowania.



Powyższy rysunek odzwierciedla użycie mapowanie obiektowo- relacyjnego

3. Wykorzystane narzędzia i technologie

- **Java**

  Będziemy używać języka Java w wersji 14

- **Spring**

  Użyjemy SpringBoot'a w wersji 2.3.

- **PostgreSQL**

  Do zarządzania naszą bazą danych. Wersja 13.

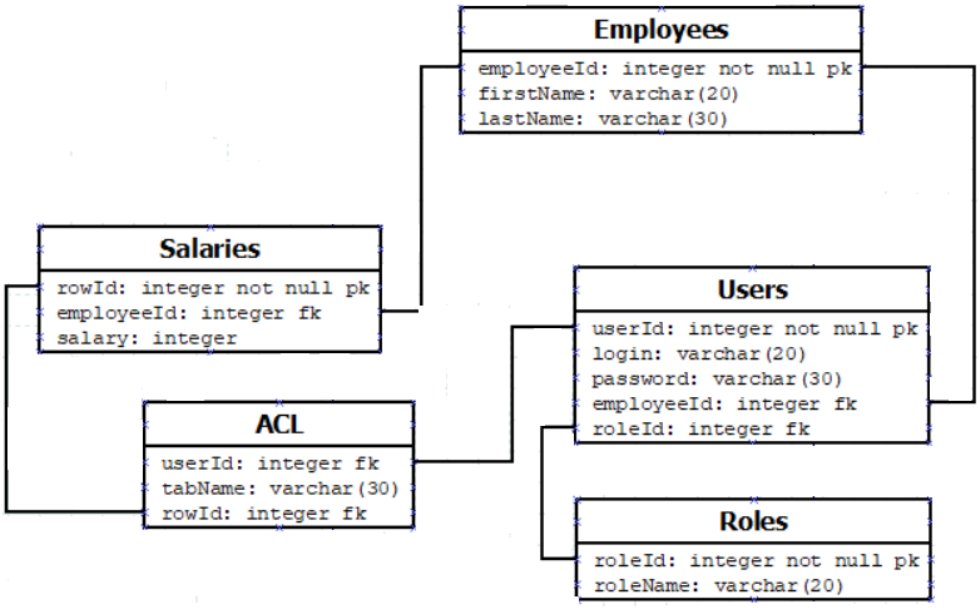
- **Hibernate**

Framework realizujący dostęp do danych w bazie. Podstawową rzeczą jaką będziemy wykorzystywać to mapowanie O-R, które w tym przypadku jest realizowane z wykorzystaniem języka XML.

- **Lombok**

Biblioteka Javy udostępniają adnotacje to tworzenia metod, głównie getterów i setterów.

4. Schemat bazy danych



5. Przykładowe zawartości tabel

Table: Employees

employeeId	firstName	lastName
1	Jan	Kowalski
2	Krzysztof	Nowak
3	Maria	Kowalska
4	Wojciech	Adamiec
5	Katarzyna	Nowakowska

Table: Salaries

rowId	employeeId	salary
1	1	10000
2	2	5000
3	3	3000
4	4	7500
5	5	2000

Table: Roles

roleId	roleName
1	administrator
2	szef
3	księgowa
4	menadżer
5	pracownik

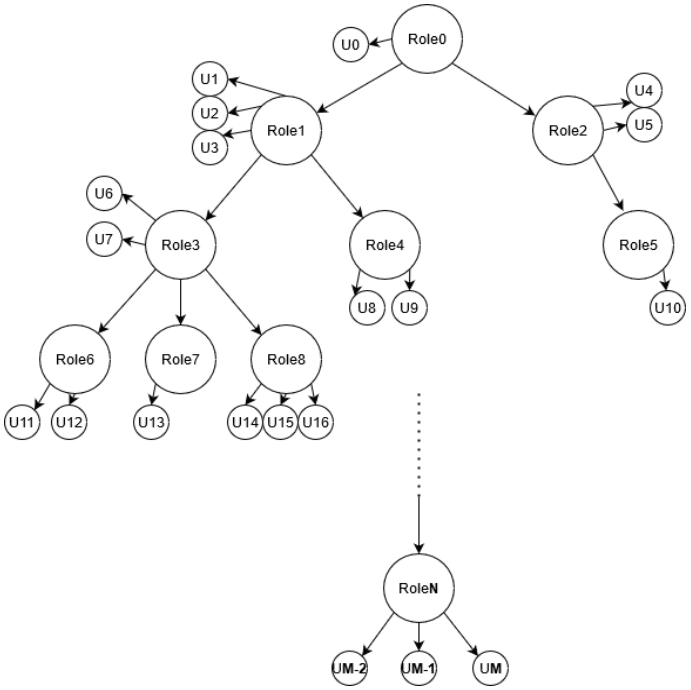
Table: Users

userId	login	password	employeeId	roleId
1	mkowalska	********	3	1
2	jkowalski	*******	1	2
3	wadamiec	**********	4	3
4	knowakowska	********	5	4

Table: ACL

userId	tabName	rowId
1	pensje	1
1	pensje	3
1	pensje	4
2	pensje	1
2	pensje	2
2	pensje	3
2	pensje	4
3	pensje	3
3	pensje	4
4	pensje	4

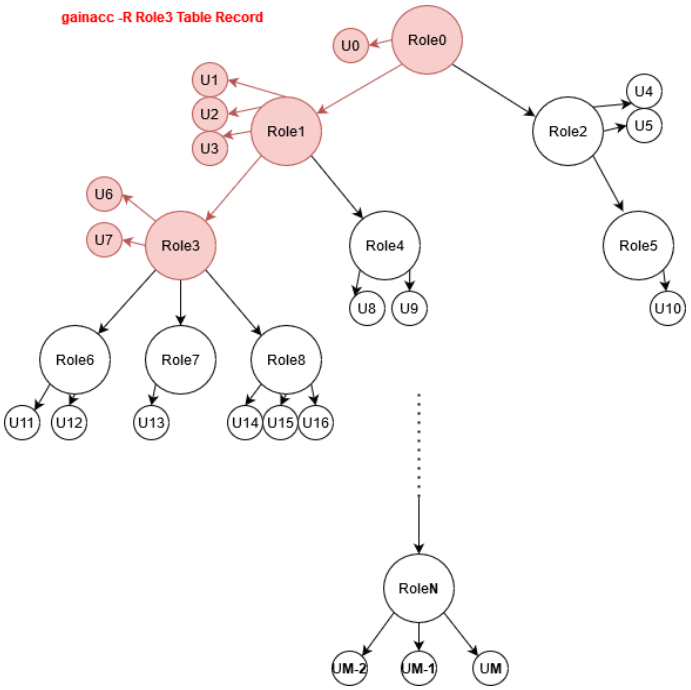
6. Struktura drzewa ról w systemie



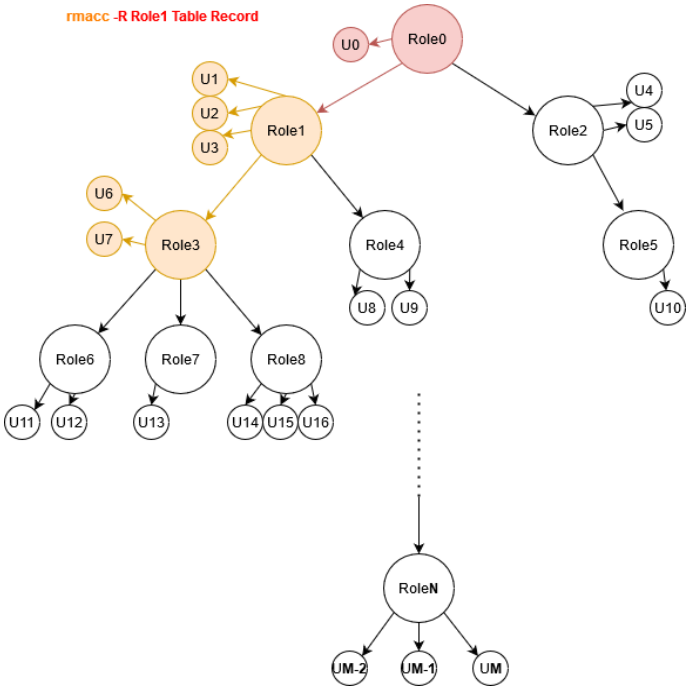
7. Drzewo ról z przykładowymi poleceniami nadającymi uprawnienia

Domyślnie dodawanie/odbieranie uprawnień dla ról/użytkowników będzie miało charakter hierarchiczny, istnieje też możliwość zarządzania dostępem dla pojedynczych ról/użytkowników.

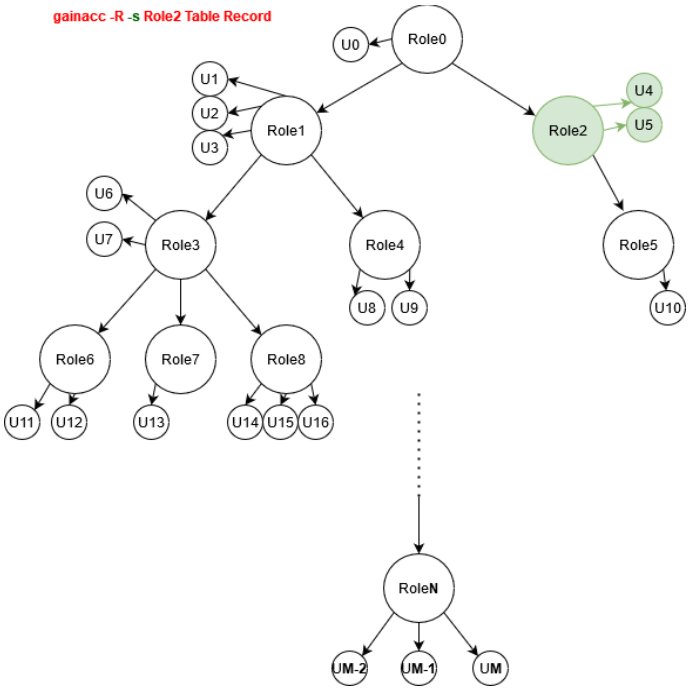
a. Dodawanie uprawnień roli do drzewa:



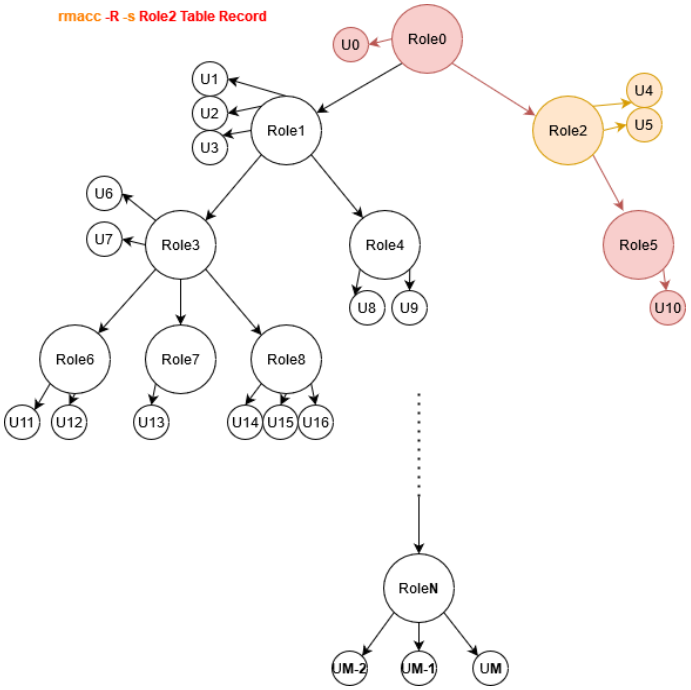
b. Odbieranie uprawnień roli w drzewie:



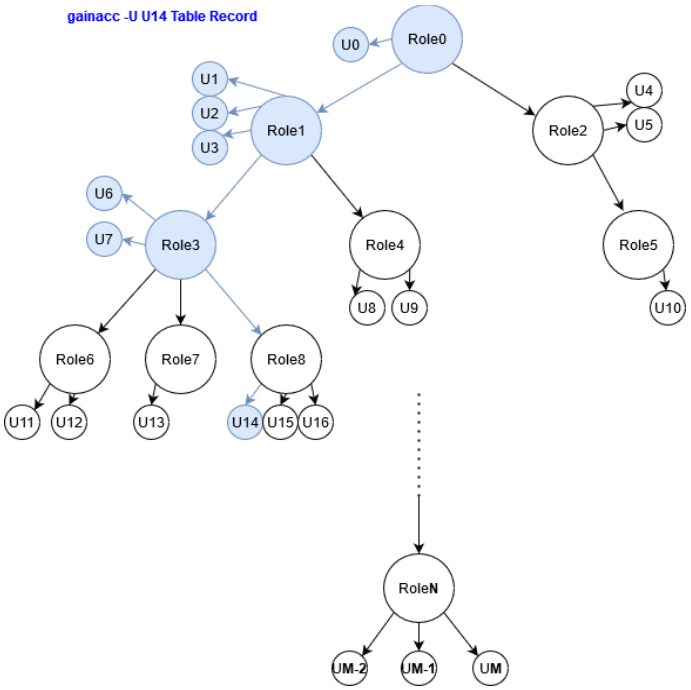
c. Dodawanie uprawnień pojedynczej roli:



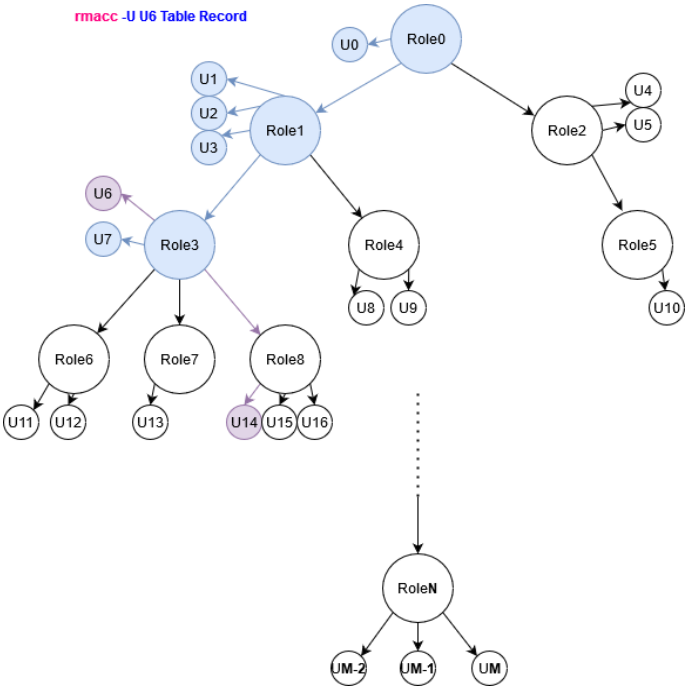
d. Odbieranie uprawnień pojedynczej roli:



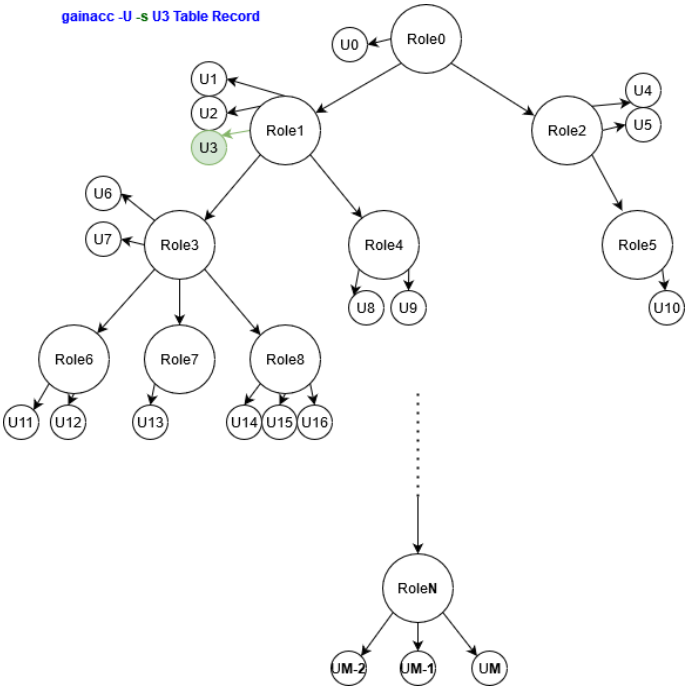
e. Dodawanie uprawnień użytkownikowi do drzewa:



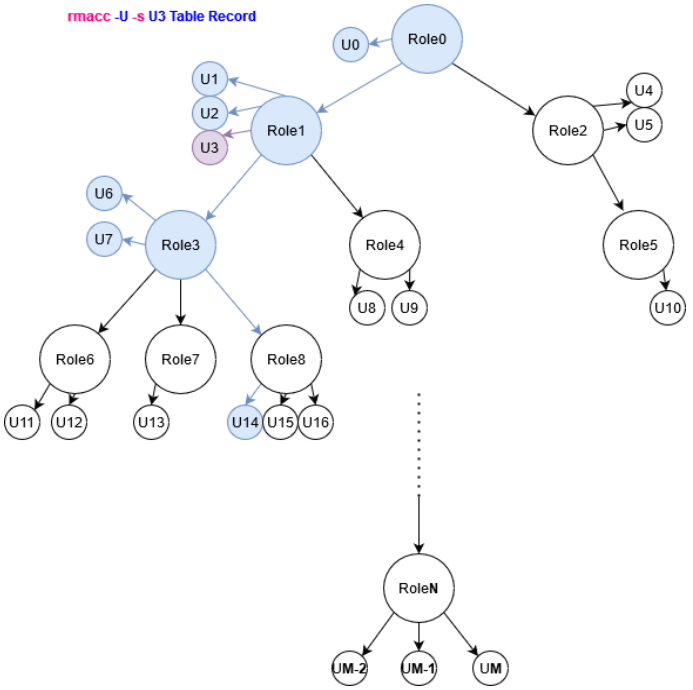
f. Odbieranie uprawnień użytkownikowi w drzewie:



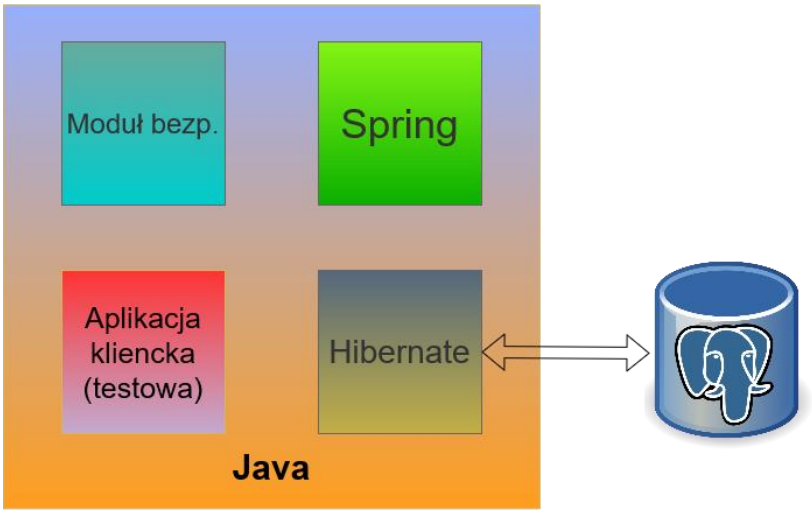
g. Dodawanie uprawnień pojedynczemu użytkownikowi:



h. Odbieranie uprawnień pojedynczemu użytkownikowi:

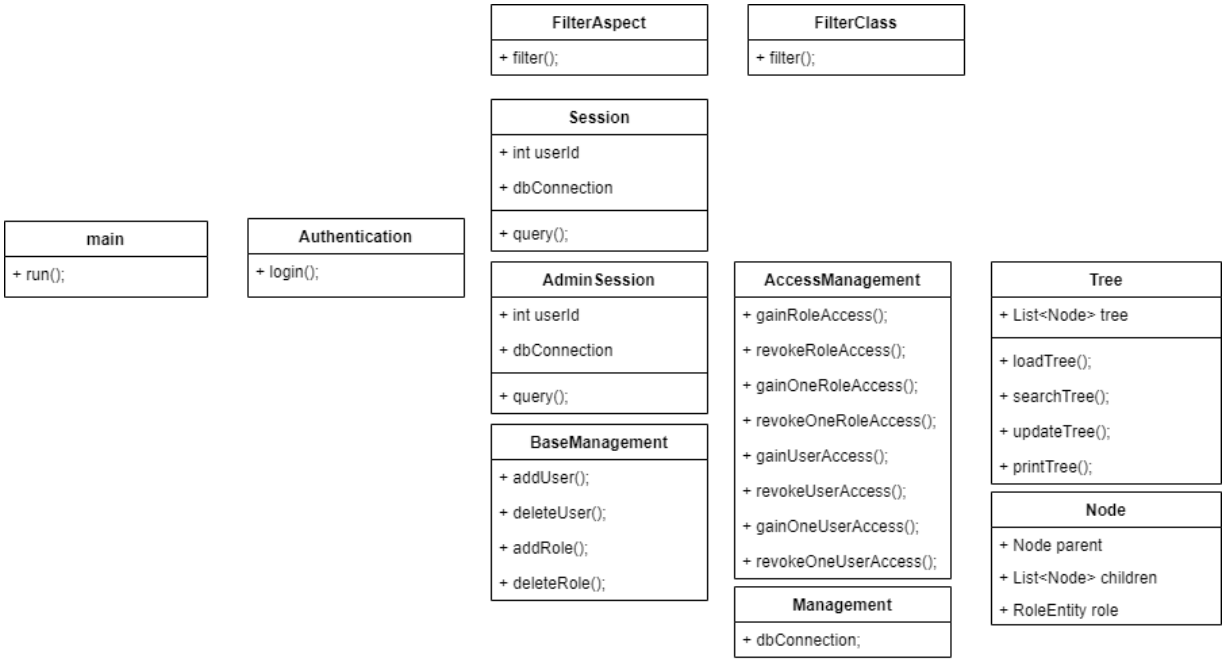


8. Architektura systemu



Jest to architektura całości naszej aplikacji, którą będziemy tworzyć w celu sprawdzenia poprawności działania naszego modułu, który jest sednem projektu. Zapytania bazodanowe będziemy wpisać przez konsolę, następnie przejdą one przez nasz moduł, zostaną odpowiednio zmodyfikowane tak, aby wynik był adekwatny do uprawnień posiadanych przez użytkownika, który utworzył to zapytanie.

9. Główne moduły



10. Wzorce projektowe

Kompozyt(Tree) - strukturalny wzorzec projektowy pozwalający komponować obiekty w struktury drzewiaste, a następnie traktować te struktury jakby były osobnymi obiektami.

Strategia(AccessManagement) - behawioralny wzorzec projektowy pozwalający zdefiniować rodzinę algorytmów, umieścić je w osobnych klasach i uczynić obiekty tych klas wymienialnymi.

Builder(Filter) - jest kreacyjnym wzorcem projektowym, który daje możliwość tworzenia złożonych obiektów etapami, krok po kroku. Wzorzec ten pozwala produkować różne typy oraz reprezentacje obiektu używając tego samego kodu konstrukcyjnego.

DAO(Data Access Object) – (Access Management) - obiekt dostępu do danych. Jest to klasa, która pozwala na pobieranie, dodawanie, usuwanie i modyfikowanie danych. Obiekt ten nie jest ograniczony do przechowywania konkretnego typu danych. Z jego wykorzystaniem można operować na wielu typach danych.