

**Dokumentacja projektu**  
**‘Znajdowanie optymalnego miejsca parkingowego**  
**w oparciu w ważony MaxSat Solver’**

**Studio Projektowe I**

Twórcy : Bartosz Biegun , Paweł Hanzlik

Opiekun: prof. Radosław Klimek



26.04.2021

## Spis treści

1.	Cel projektu .....	3
2.	Zasada działania .....	3
3.	Architektura systemu .....	4
4.	Schemat bazy danych .....	5
5.	Przykłady rekordów w tabelach .....	6
6.	Diagram klas .....	7
7.	Wykorzystane technologie .....	8

## 1. Cel projektu

Celem projektu jest zaimplementowanie solvera typ MaxSat do znajdowania najbardziej przystępnego miejsca parkingowego.

Aplikacja będzie symulowała obszar danego miasta podzielonego na wyznaczone strefy, wewnątrz której ulokowane są parkingi. Każdy parking może posiadać kilka typów miejsc parkingowych z odpowiednimi parametrami. Użytkownik będzie posiadał możliwość utworzenia zapytania odpytującego serwer o miejsce parkingowe z podanymi wartościami lub uruchomić prostą symulację generującą wiele podobnych zapytań oraz modyfikującą stan bazy w zależności od pory dnia.

## 2. Zasada działania

Problem spełnialności to koncept związany z logiką matematyczną, zostanie on wykorzystany do rozwiązania problemu znalezienia przystępnego miejsca parkingowego w Krakowie.

Z problemem SAT mamy do czynienia gdy mając formułę zdaniową chce się określić, czy istnieje podstawienie wartościami '0' i '1' pod zmienne zdaniowe, by formuła była spełniona.

Problemy Max-Sat składają się z ważonych klauzul połączonych koniunkcjami :

$$(\neg p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_3) \wedge (\neg p_1 \vee \neg p_3)$$

Ponieważ nie zawsze da się spełnić wszystkie klauzule, solver Max Sat znajduje rozwiązanie w którym największa liczba klauzul jest spełniona

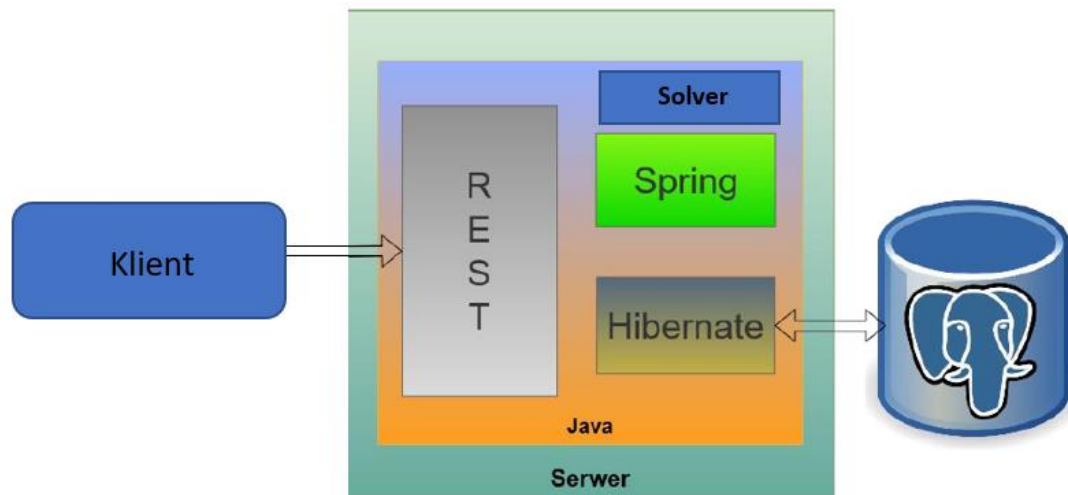
Z problemem Max-SAT to rozszerzenie problemu SAT w taki sposób, aby w przypadku gdy nie da się dobrać wartości tak, aby spełniona była cała formuła dobiera się je tak, by zmaksymalizować ilość spełnionych formuł.

Ważony Max-SAT to kolejne rozszerzenie, dodające tym razem odpowiadające wagi każdej z klauzul i uwzględnienie ich a procesie rozwiązywania w taki sposób aby suma wag niespełnionych klauzul była jak najmniejsza.

Aby reprezentować proces parkowania system będzie generować formułę w której zdania będą odpowiadać przyporządkowaniu samochodu do miejsca parkingowego, przy czym waga klauzuli będzie odpowiadać przystępności (odległości) miejsca dla klienta.

Dodatkowo zostanie dodana klauzula o nieskończonej wadze uniemożliwiająca przyporządkowanie wielu samochodów do jednego miejsca, i wymuszająca przyporządkowanie samochodowi dokładnie jednego miejsca

### 3. Architektura systemu



Główny moduł aplikacji zostanie napisany w języku Java z użyciem Spring Framework. Baza danych działać będzie na serwerze PostgreSQL, natomiast łączenie jej z projektem realizowane będzie przy użyciu Hibernate. Z zewnątrz klient wysyłać będzie zapytanie obsługiwane przez REST API, które na podstawie danych z bazy oraz obliczeń Solvera zwróci wynik.

## 4. Schemat bazy danych

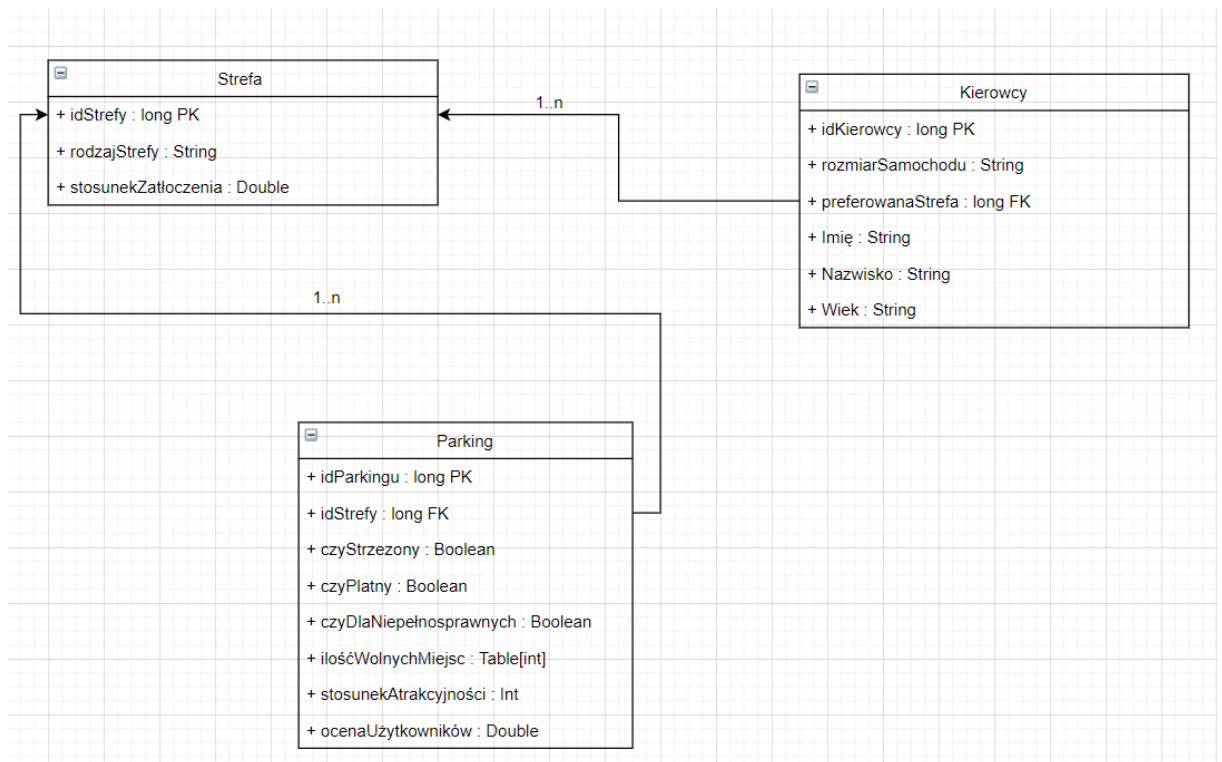


Tabela Strefa oznacza podział miasta na obszary posiadające wartość zatłoczenia w danym momencie oraz rodzaj Strefy tj. obrzeża, centrum, poza miastem, przemysłowa. Zawiera wiele rekordów typu Parking.

Tabela Kierowcy zawiera dane klienta, informacje o rozmiarach ich samochodów oraz preferencjach dotyczących miejsca parkingowego.

Tabela Parking zawiera informacje dotyczące parkingu w danej strefie miasta.

## 5. Przykłady rekordów w tabelach

Tabela: strefy

idStrefy	rodzajStrefy	stosunekZatloczenia
1	centrum	0.7
2	centrum	0.8
3	obrzeża	0.5
4	obrzeża	0.4
5	poza miastem	0.1

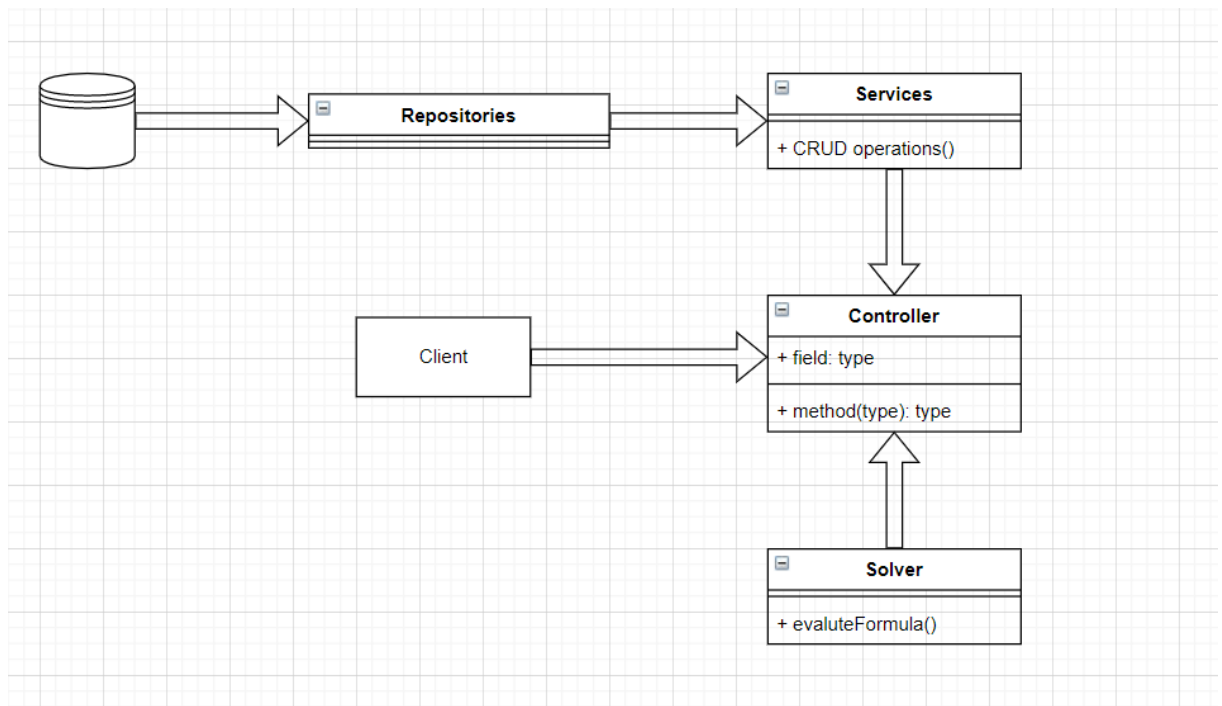
Tabela: kierowcy

idKierowcy	rozmiarSamochodu	preferowanaStrefa	Imię	Nazwisko	Wiek
1	mały		2 Anna	Nowak	34
2	średni		4 Krzysztof	Kowalczyk	23
3	duży		3 Michał	Kowalski	40
4	niepełnosprawność		5 Jan	Nowakowski	56
5	duży		2 Wojciech	Kwiatkowski	42

Tabela: parkingi

idParkingu	idStrefy	czyStrzeżony	czyPłatny	czyDlaNiepełnosprawnych	ilośćWolnychMiejsc	stosunekAtrakcyjności	ocenyUżytkowników
1	2	Tak	Tak	Tak	[50,20]	0.9	5,00
2	2	Tak	Nie	Nie	[100,null]	0.6	4,50
3	1	Nie	Nie	Tak	[150,30]	0.7	4,00
4	4	Tak	Tak	Nie	[30,null]	0.5	4,20
5	3	Tak	Nie	Nie	[60,null]	0.3	4,50
6	5	Nie	Tak	Tak	[40,10]	0.3	3,70

## 6. Diagram klas



Sekcja **Repositories** to interfejsy dziedziczące po **JpaRepository**, realizujące połączenie z bazą. Serwisy będą przechowywały metody umożliwiające pobieranie, dodawanie czy też modyfikowanie oraz usuwanie danych z bazy. **Solver** będzie to zbiór klas, które implementują algorytm **Max-Sat solver** i na podstawie zapytania klienta zwróci najlepsze miejsce parkingowe. Kontroler będzie przetwarzał zapytanie, przekazywał do solvera i zwracał odpowiedź klientowi.

## 7. Wykorzystane technologie

- **Java**

Będziemy używać języka Java w wersji 15

- **Spring**

Użyjemy SpringBoot'a w wersji 2.4.

- **PostgreSQL**

Do zarządzania bazą danych. Wersja 13.

- **Hibernate**

Czyli framework realizujący dostęp do danych w bazie. Podstawową rzeczą jaką będziemy wykorzystywać to mapowanie O-R.

- **Lombok**

Biblioteka Javy udostępniająca adnotacje do tworzenia metod, głównie getterów i setterów.