**Dokumentacja projektu**

**‘ Znajdowanie optymalnego miejsca parkingowego**

**w oparciu w ważony MaxSat Solver’**

**Studio Projektowe I**

Twórcy : Bartosz Biegun , Paweł Hanzlik

Opiekun: prof. Radosław Klimek



05.05.2021

Spis treści

[1. Cel projektu 3](#_Toc71121144)

[2. Zasada działania 3](#_Toc71121145)

[3. Architektura systemu 4](#_Toc71121146)

[4. Schemat bazy danych 5](#_Toc71121147)

[5. Przykłady rekordów w tabelach 6](#_Toc71121148)

[6. Diagram klas 7](#_Toc71121149)

[7. Wykorzystane technologie 8](#_Toc71121150)

[8. Generowanie klauzul z dostępnych danych 9](#_Toc71121151)

[a. Określenie zmiennych zdaniowych 9](#_Toc71121152)

[b. Określenie klauzul 9](#_Toc71121153)

[c. Interpretacja wyników 9](#_Toc71121154)

# 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.

# 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 :

(¬p1∨¬p2)∧(¬p1∨p3)∧(¬p1∨¬p3)

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

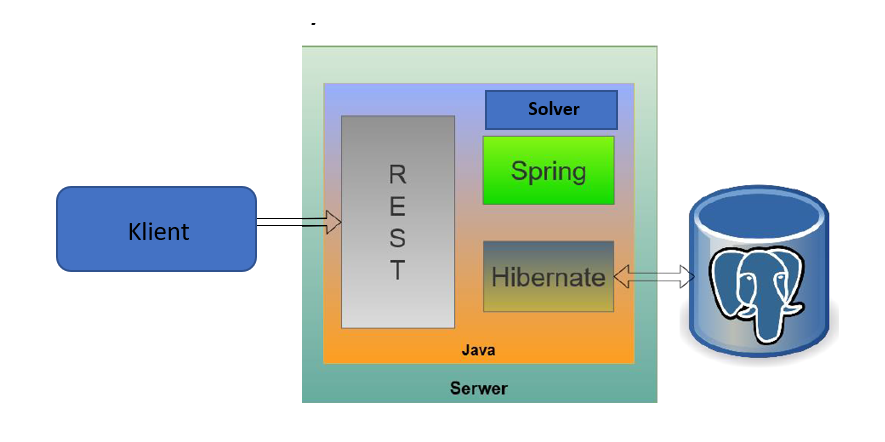
Z problemem Max-SAT to rozszerzenie problemu SAT w taki sposób, aby w przypadku gdy nie da się dobrać wartości tak, aby spełniona była cała formuła dobiera się je tak, by zmaksymalizować ilość spełnionych formuł.

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

# Architektura systemu

****

Główny moduł aplikacji zostanie napisany w języku Java z użyciem Spring Framework. Baza danych działać będzie na serwerze PostreSQL, 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.

# Schemat bazy danych

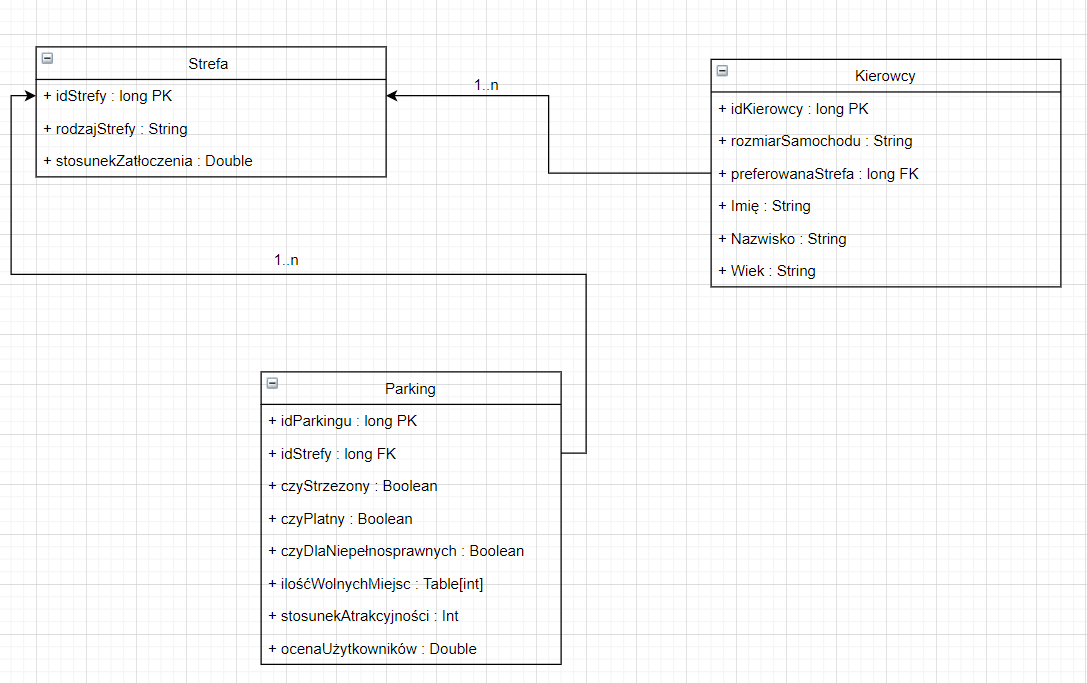
****

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.

# Przykłady rekordów w tabelach

Tabela: strefy

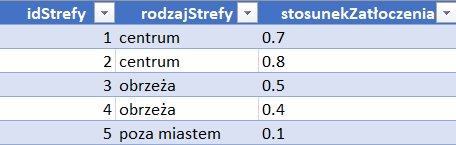
****

Tabela: kierowcy

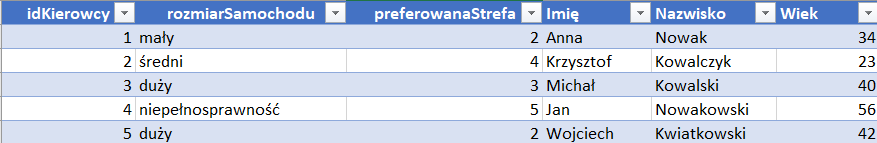
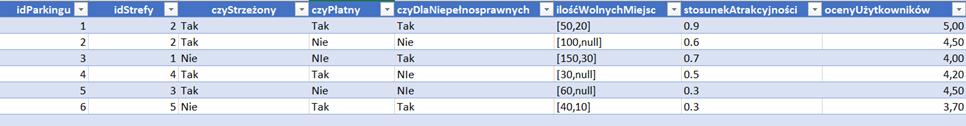
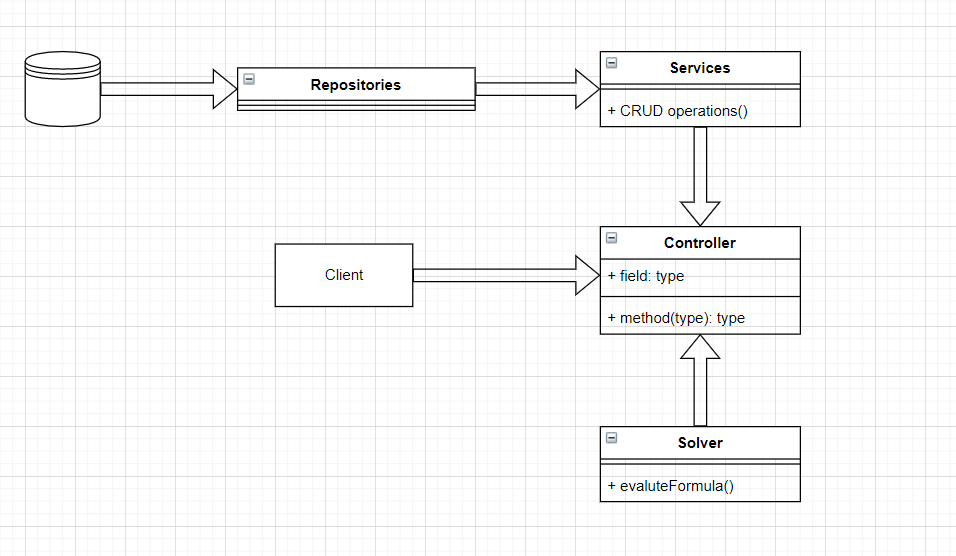
****

Tabela: parkingi

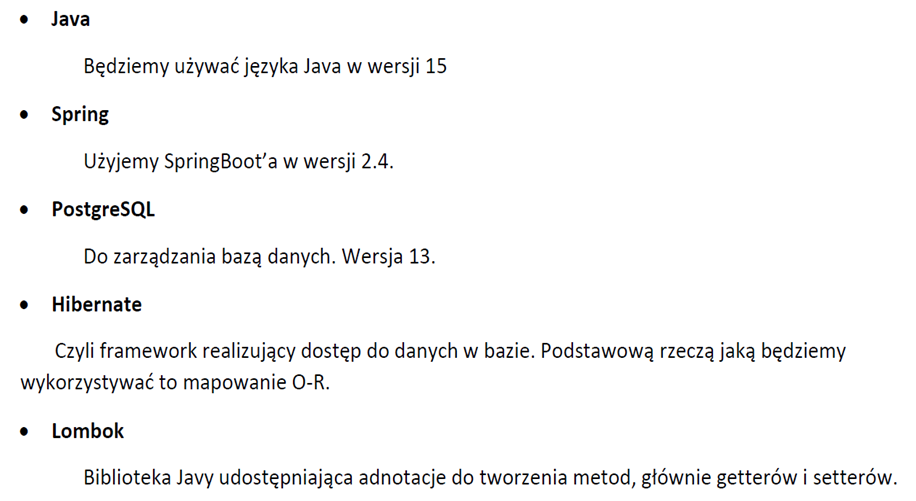


# Diagram klas

****

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.

# Wykorzystane technologie

****

# Generowanie klauzul z dostępnych danych

Do przetwarzania danych za pomocą algorytmu Weighted Max-Sat wymagane jest przekształcenie je na zmienne zdaniowe a następnie przyporządkowanie do klauzul.

Każdej klauzuli przypisywana jest waga określająca jak ważne jest spełnienie klauzuli.

Każde dostępne miejsce parkingowe zostaje porównane z klauzulą i przedstawione użytkownikowi. Lista dostępnych miejsc zostaje posortowana po ilości spełnionych klauzul tak

aby najbardziej dopasowane miejsca znalazły się na górze listy.

## Określenie zmiennych zdaniowych

Na podstawie bazy danych wyróżnimy 6 zmiennych zdaniowych:

* + - S1 - Parking jest strzeżony
    - S2 - Parking jest płatny
    - S3 - Parking Posiada miejsca dla niepełnosprawnych
    - S4 – Parking jest w preferowanej strefie klienta
    - S5 – Parking posiada ocenę wyższą niż 4 gwiazdki
    - S6 – Parking posiada co najmniej 20% wolnych miejsc
    - S7 - Atrakcyjność parkingu wynosi co najmniej 50%

## Określenie klauzul

Analizując dane o kliencie możemy oszacować jego preferencje i dopasować do nich zmienne zdaniowe

* + - U1 – Rozmiar Samochodu duży
    - U2 – Kierowca jest starszy niż 50 lat
    - U3 – Kierowca jest młodszy niż 25 lat
    - U4 – Kierowca pochodzi z Krakowa

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | S1 v S6 (90) |  |  | ¬S2 v S5 (20) |
| U1 |  | ¬ U1 |  |
|  | S5 v S4 (60) |  | S1 v S7 (50) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | S3 v S4 (90) |  |  | ¬S2 v S4 (40) |
| U2 |  | ¬ U2 |  |
|  | S1 v S6 (45) |  | S3 v S6 (50) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | ¬ S1 v S6 (70) |  |  | ¬S2 (40) |
| U3 |  | ¬ U3 |  |
|  | ¬ S2 v S1 (55) |  | S7 (50) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | S3 v S6 (20) |  |  | S7 v S5 (100) |
| U4 |  | ¬ U4 |  |
|  | S5 v S4 (60) |  | S1 v S4 (50) |

## Interpretacja wyników

Algorytm iteruje po wszystkich miejscach parkingowych i wyodrębnia te z największą ilością spełnionych klauzul, a następnie przedstawia je użytkownikowi jako listę.