# Garbage Collectors 🗑

Systém by měl sloužit firmě, která se zabývá svozem odpadu. Měl by napomáhat při organizaci provozu firmy. Systém bude poskytovat přehled o aktuálních svozech, svozových nádobách a stavu skládek na jehož základě bude možné svozy optimalizovat z hlediska trasy nebo typu potřebného vozu (objem, konstrukce apod.).

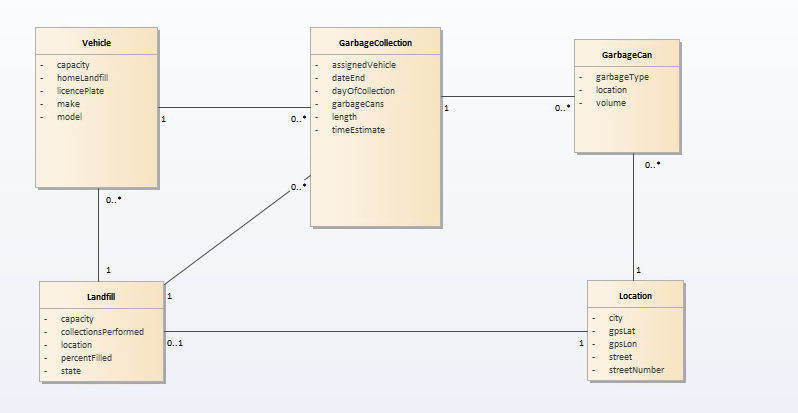
# Spuštění

Po vytvoření kontejnerů pomocí docker compose je třeba spustit startRepl.js pro funkční replikaci.  
Kolekce a jejich validace se založí pomocí createDB.js, následně mohou být naplněny pomocí initDB.js

Upozornění: Data vytvořená pomocí initDB.js jsou náhodná a nebudou konzistentní při opakovaném generování. Výstupy některých skriptů se tedy mohou lišit.

Tato dokumentace je dostupná ve formátech docx, pdf a html.

# E-R Diagram



# Slovní definice validací

Location:

* city je nepovinný atribut typu string.
* gpsLat (zeměpisná šířka) je povinný atribut typu double a dosahuje hodnot -90 až 90.
* gpsLon (zeměpisná výška) je povinný atribut typu double a dosahuje hodnot -180 až 180.
* street je nepovinný atribut typu string.
* streetNumber je nepovinný atribut typu string.

GarbageCan

* garbageType je povinný atribut a typu GType (enum string hodnot dle projektu z PPRO).
* location je povinný ObjectID odkazující na kolekci Location.
* volume je povinný atribut tybu double a může dosahovat hodnot 10 až 1000.

Landfill

* capacity je povinný atribut typu int a dosahuje hodnot větších než 10000l.
* collectionsPerfomed je seznam všech svozů této skládky, odkazy pomocí ObjectID
* location je povinný ObjectID odkazující na kolekci Location.
* percentFilled je volitelný atribut typu integer a dosahuje hodnot 0 a vyšších.
* operational je povinný atribut typu boolean.

Vehicle

* capacity je povinný atribut typu double a dosahuje hodnot v rozmezí 1000 až 10000l.
* homeLandfill je povinný ObjectID odkazující na Landfill.
* licencePlate je povinný atribut typu string.
* make je volitelný atribut typu string.
* model je volitelný atribut typu string.

GarbageCollection

* assignedVehicle je povinný ObjectID odkazující na Vehicle.
* dateEnd je volitelný atribut typu Date.
* dayOfCollection je povinný atribut typu enum nabývající hodnot: Monday, Tuesday, Wednesday, Thursday, Friday.
* garbageCans je seznam popelnic ve svozu
* length je délka svozu v kilometrech, datový typ Double
* timeEstimate je odhad času potřebného na svoz v minutách, datový typ Int

# Popis ukázky API

APItest.js

**1) Přidání nové nádoby**

Očekávané vstupy:

* GPS souřadnice svozové nádoby
* Typ odpadu
* Objem svozové nádoby
* Volitelně adresa stanoviště

Je vytvořena nová lokace na základě GPS souřadnic, která je následně použita pro vytvoření nové svozové nádoby v databázi. API by mohlo výsledný objekt vracet v odpovědi.

**2) Vytvoření svozu pro malou obec**

Očekávané vstupy:

* Název obce
* Den svozu
* Poznávací značka vozidla provádějící svoz

Na základě jména obce jsou nalezeny všechny svozové nádoby, které se v ní nachází. Následně jsou přidány do nového svozu, ke kterému je přiřazeno zvolené vozidlo. Tento je zamýšlen pouze jako zjednodušení pro malé obce a vsi a nebyl by využíván pro větší města.

# Zálohovací skript

backup.ps1

backup.sh

backup\_vystup.txt

**Popis:**

PowerShell script slouží k vložení backup.sh do primárního kontejneru a jeho spuštění. Tento Bash script vytvoří pomocí mongodum zálohu v home/dump kontejneru, která je pak vykopírována PowerShell scriptem do složky, ze které byl spuštěn.

# Vyčištění databáze

clearDB.js

deleteDB.js

**Popis:**

clearDB.js slouží k promazání dat všech kolekcí, zanechává validace a prázdné kolekce. deleteDB.js smaže data i kolekce.

# Dotazy nad schématem

Dotaz 1 (agregace):

Celkový počet svozových nádob, které mají typ Nebezpečné.

Dotaz 2 (agregace):

Seznam všech skládek které jsou naplněny z více než 10%.

Dotaz 3 (agregace):

Seznam všech aut, které mají značku Škoda a jejich domovská skládka sídlí v Praze.

Dotaz 4 (agregace):

Četnost jednotlivých druhů popelnic nacházející se v Praze.

Dotaz 5 (agregace):

Nalezení svozu, který má největší souhrnný objem popelnic.

# Skripty s navrženými dotazy nad schématem

* Dotaz 1 (agregace)

Verze1:

dotaz\_1a.js

dotaz\_1a\_vystup.txt

Verze2:

dotaz\_1b.js

dotaz\_1b\_vystup.txt

* Dotaz 2 (agregace)

Verze1:

dotaz\_2a.js

dotaz\_2a\_vystup.txt

Verze2:

dotaz\_2b.js

dotaz\_2b\_vystup.txt

* Dotaz 3 (agregace)

dotaz\_3.js

dotaz\_3\_vystup.txt

* Dotaz 4 (agregace)

dotaz\_4.js

dotaz\_4\_vystup.txt

* Dotaz 5 (agregace)

dotaz\_5.js

dotaz\_5\_vystup.txt

Porovnání výsledků z explain():

Dotaz číslo 1a byl rychlejší o 1ms než 1b, přičemž totalDocsExamined měly oba dotazy shodné s hodnotou 1038.

Dotaz číslo 2a byl rychlejší o 5ms než dotaz 2b. Rozdíl v totalDocsExamined též není zanedbatelný, u 1a dosahuje hodnoty 20 a dotaz 2b hodnoty 1038 dokumentů.

Porovnání efektivity složených indexů:

Nejvyšší nárůst rychlosti díky složených indexů byl u dotazu číslo 2b. Z původních 5ms jsme dosáhli zrychlení zpracování na 3ms. U dotazu číslo 1b bylo zlepšení zanedbatelné.