Inhalt

Link to github: https://github.com/DenesTak/PetShelter

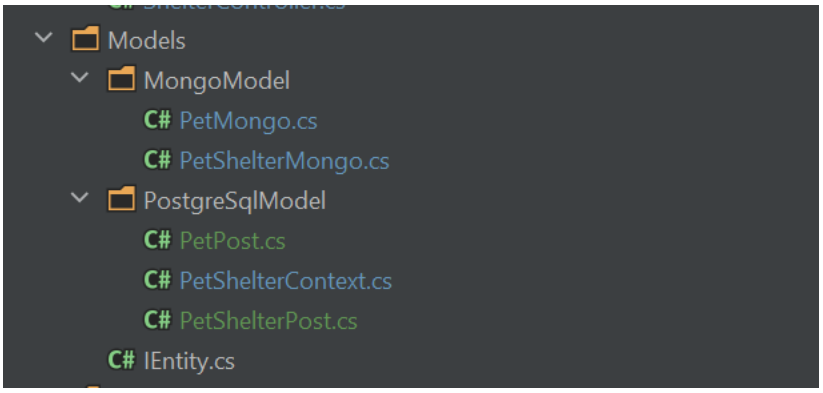
Step 1

This simple solo project is about Pets and Pet shelters. A Pet shelter can have many pets, but a pet can only be in one shelter at a time and a shelter can't have more animals than its capacity.

Step 2

Models (1 Point)

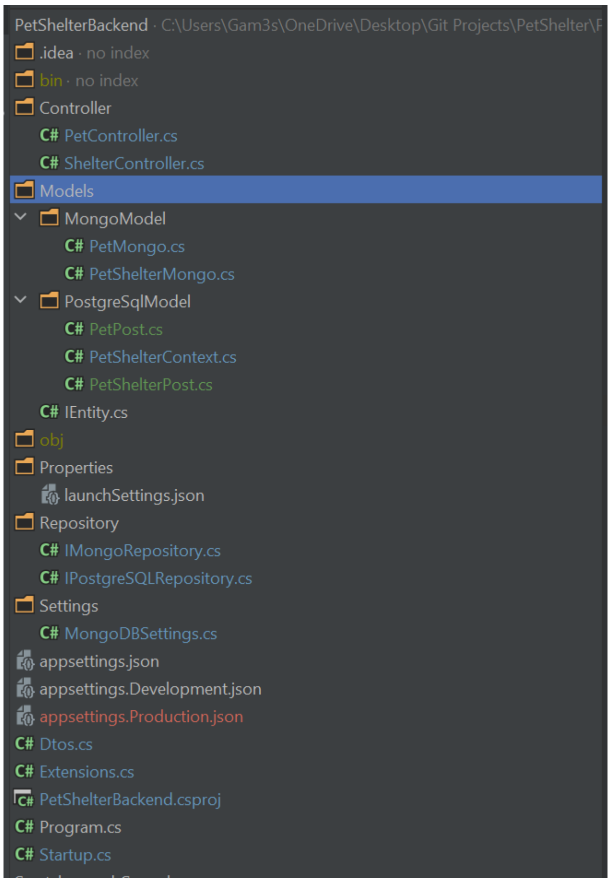
The models can be found under the "Models" Folder



```
Models
  MongoModel
    C# PetMongo.cs
    C# PetShelterMongo.cs
  PostgreSqlModel
    C# PetPost.cs
    C# PetShelterContext.cs
    C# PetShelterPost.cs
  C# IEntity.cs
```

Implementation (1 Point)

The models are in the "Models" folder and there are separate Repositories for Mongo and PostgreSQL, as well as controllers for pet and shelter where the API are located.

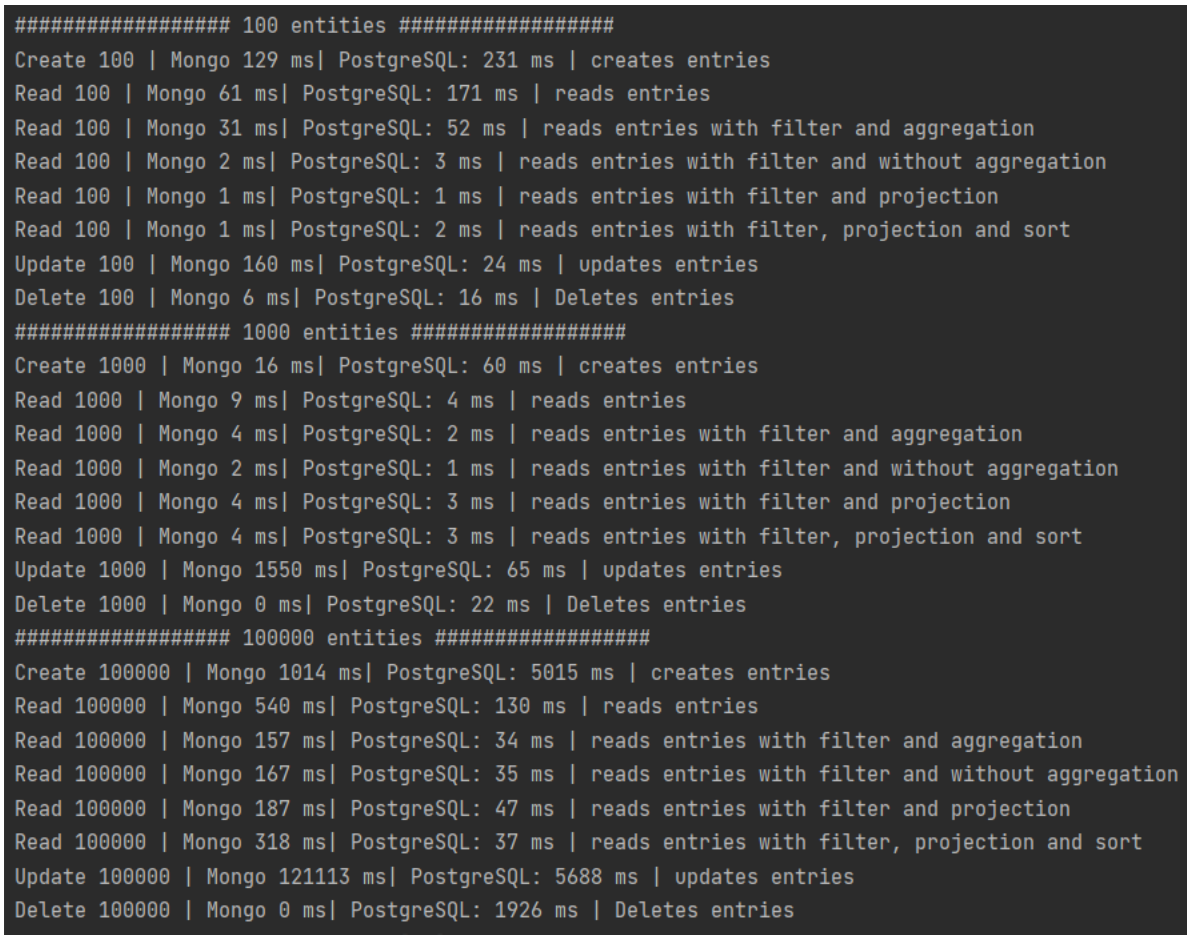


```
PetShelterBackend · C:\Users\Gam3s\OneDrive\Desktop\Git Projects\PetShelter\
.idea · no index
bin · no index
Controller
  C# PetController.cs
  C# ShelterController.cs
Models
  MongoModel
    C# PetMongo.cs
    C# PetShelterMongo.cs
  PostgreSqlModel
    C# PetPost.cs
    C# PetShelterContext.cs
    C# PetShelterPost.cs
  C# IEntity.cs
obj
Properties
  launchSettings.json
Repository
  C# IMongoRepository.cs
  C# IPostgreSQLRepository.cs
Settings
  C# MongoDBSettings.cs
appsettings.json
appsettings.Development.json
appsettings.Production.json
C# Dtos.cs
C# Extensions.cs
PetShelterBackend.csproj
C# Program.cs
C# Startup.cs
```

Step 3

CRUD performance testing (2 Points)

Testing CRUD performance (100 – 1.000 – 100.000)

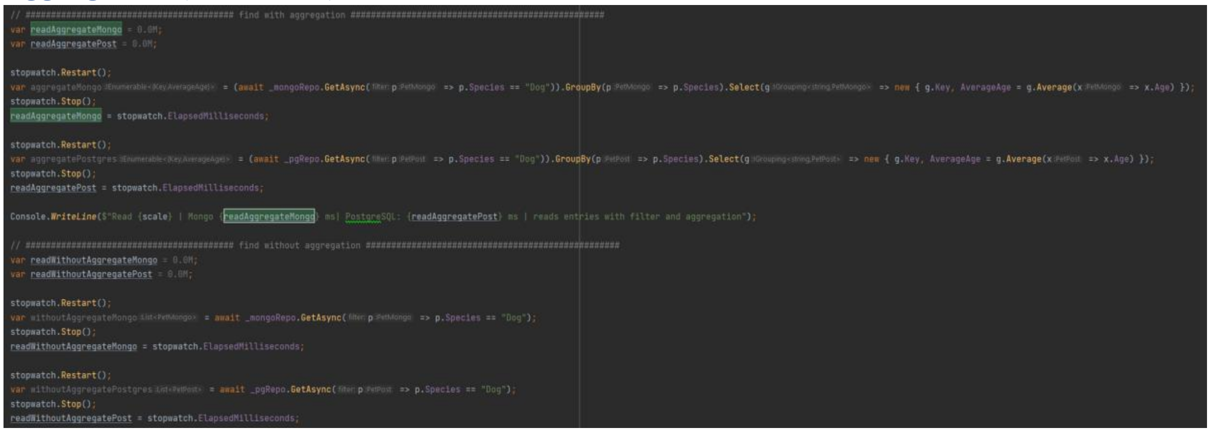


```
################## 100 entities ##################
Create 100 | Mongo 129 ms| PostgreSQL: 231 ms | creates entries
Read 100 | Mongo 61 ms| PostgreSQL: 171 ms | reads entries
Read 100 | Mongo 31 ms| PostgreSQL: 52 ms | reads entries with filter and aggregation
Read 100 | Mongo 2 ms| PostgreSQL: 3 ms | reads entries with filter and without aggregation
Read 100 | Mongo 1 ms| PostgreSQL: 1 ms | reads entries with filter and projection
Read 100 | Mongo 1 ms| PostgreSQL: 2 ms | reads entries with filter, projection and sort
Update 100 | Mongo 160 ms| PostgreSQL: 24 ms | updates entries
Delete 100 | Mongo 6 ms| PostgreSQL: 16 ms | Deletes entries
################## 1000 entities ##################
Create 1000 | Mongo 16 ms| PostgreSQL: 60 ms | creates entries
Read 1000 | Mongo 9 ms| PostgreSQL: 4 ms | reads entries
Read 1000 | Mongo 4 ms| PostgreSQL: 2 ms | reads entries with filter and aggregation
Read 1000 | Mongo 2 ms| PostgreSQL: 1 ms | reads entries with filter and without aggregation
Read 1000 | Mongo 4 ms| PostgreSQL: 3 ms | reads entries with filter and projection
Read 1000 | Mongo 4 ms| PostgreSQL: 3 ms | reads entries with filter, projection and sort
Update 1000 | Mongo 1550 ms| PostgreSQL: 65 ms | updates entries
Delete 1000 | Mongo 0 ms| PostgreSQL: 22 ms | Deletes entries
################## 100000 entities ##################
Create 100000 | Mongo 1014 ms| PostgreSQL: 5015 ms | creates entries
Read 100000 | Mongo 540 ms| PostgreSQL: 130 ms | reads entries
Read 100000 | Mongo 157 ms| PostgreSQL: 34 ms | reads entries with filter and aggregation
Read 100000 | Mongo 167 ms| PostgreSQL: 35 ms | reads entries with filter and without aggregation
Read 100000 | Mongo 187 ms| PostgreSQL: 47 ms | reads entries with filter and projection
Read 100000 | Mongo 318 ms| PostgreSQL: 37 ms | reads entries with filter, projection and sort
Update 100000 | Mongo 121113 ms| PostgreSQL: 5688 ms | updates entries
Delete 100000 | Mongo 0 ms| PostgreSQL: 1926 ms | Deletes entries
```

Biggest difference was at inserting 100 000 entries when mongo took 1/5th of the time of PostgreSQL. Interesting is also the delete 100 000 entries that took 0 ms?

Additional Tasks

Aggregation (0.5 Points)

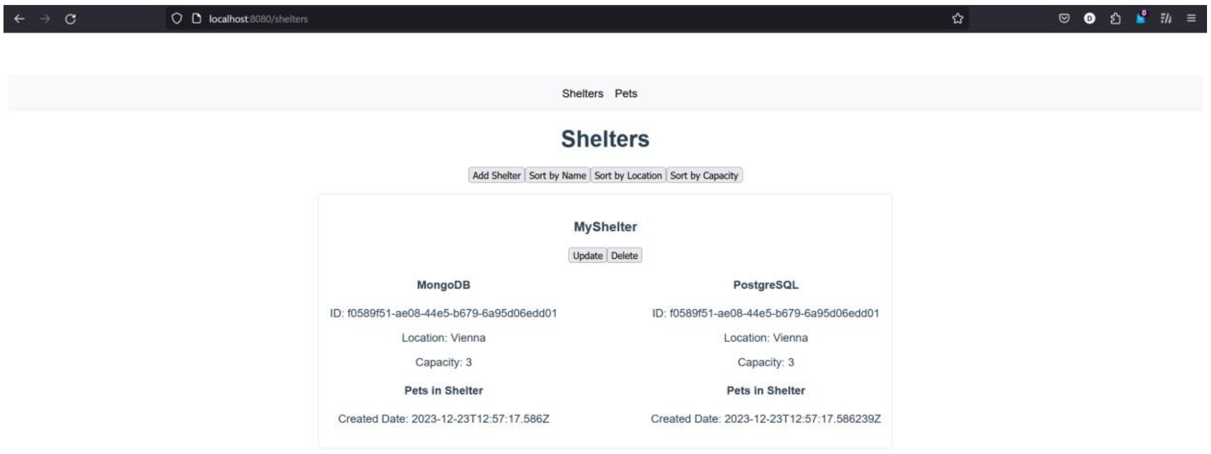


```
// ######################################## find with aggregation ################################################
var readAggregateMongo = 0.0M;
var readAggregatePost = 0.0M;

stopwatch.Restart();
var aggregateMongo = (await _mongoRepo.GetAsync(p => p.Species == "Dog")).GroupBy(p => p.Species).Select(g => new { g.Key, AverageAge = g.Average(x => x.Age) });
stopwatch.Stop();
readAggregateMongo = stopwatch.ElapsedMilliseconds;

stopwatch.Restart();
var aggregatePostgres = (await _pgRepo.GetAsync(p => p.Species == "Dog")).GroupBy(p => p.Species).Select(g => new { g.Key, AverageAge = g.Average(x => x.Age) });
stopwatch.Stop();
readAggregatePost = stopwatch.ElapsedMilliseconds;

Console.WriteLine($"Read {scale} | Mongo {readAggregateMongo} ms| PostgreSQL: {readAggregatePost} ms | reads entries with filter and aggregation");

// ######################################## find without aggregation ################################################
var readWithoutAggregateMongo = 0.0M;
var readWithoutAggregatePost = 0.0M;

stopwatch.Restart();
var withoutAggregateMongo = await _mongoRepo.GetAsync(p => p.Species == "Dog");
stopwatch.Stop();
readWithoutAggregateMongo = stopwatch.ElapsedMilliseconds;

stopwatch.Restart();
var withoutAggregatePostgres = await _pgRepo.GetAsync(p => p.Species == "Dog");
stopwatch.Stop();
readWithoutAggregatePost = stopwatch.ElapsedMilliseconds;
```

(See speed comparison task 3)

Frontend (1.5 Points)

Basic frontend for application with two pages (one for shelter, one for pet) that send request to the API and allows the data to be filtered by species and sorted by certain properties.


localhost:8080/shelters

Shelters Pets

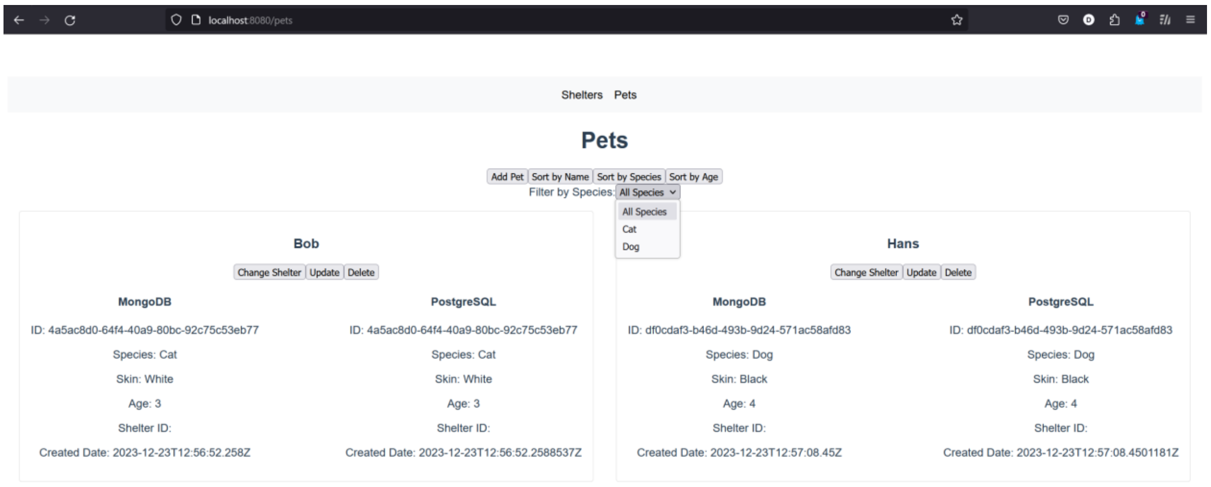
Shelters

Add Shelter | Sort by Name | Sort by Location | Sort by Capacity

MyShelter

Update | Delete

MongoDB	PostgreSQL
ID: f0589f51-ae08-44e5-b679-6a95d06edd01	ID: f0589f51-ae08-44e5-b679-6a95d06edd01
Location: Vienna	Location: Vienna
Capacity: 3	Capacity: 3
Pets in Shelter	Pets in Shelter
Created Date: 2023-12-23T12:57:17.586Z	Created Date: 2023-12-23T12:57:17.586239Z


localhost:8080/pets

Shelters Pets

Pets

Add Pet | Sort by Name | Sort by Species | Sort by Age

Filter by Species: All Species (All Species, Cat, Dog)

Bob

Change Shelter | Update | Delete

MongoDB	PostgreSQL
ID: 4a5ac8d0-64f4-40a9-80bc-92c75c53eb77	ID: 4a5ac8d0-64f4-40a9-80bc-92c75c53eb77
Species: Cat	Species: Cat
Skin: White	Skin: White
Age: 3	Age: 3
Shelter ID:	Shelter ID:
Created Date: 2023-12-23T12:56:52.258Z	Created Date: 2023-12-23T12:56:52.2588537Z

Hans

Change Shelter | Update | Delete

MongoDB	PostgreSQL
ID: df0cdaf3-b46d-493b-9d24-571ac58afd83	ID: df0cdaf3-b46d-493b-9d24-571ac58afd83
Species: Dog	Species: Dog
Skin: Black	Skin: Black
Age: 4	Age: 4
Shelter ID:	Shelter ID:
Created Date: 2023-12-23T12:57:08.45Z	Created Date: 2023-12-23T12:57:08.4501181Z

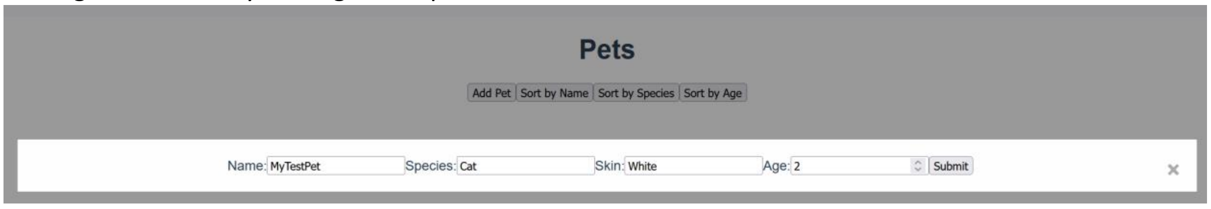
Interactive Frontend (0.75 Points)

Via the buttons shown at frontend, entries can be added/modified and deleted.

Cloud (1 Point)

Successfully connected my PostgresSQL database to azure cloud and my mongo to atlas.

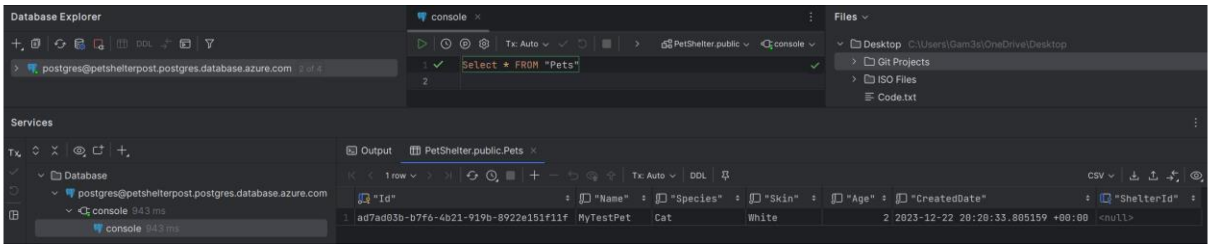
Testing connective by adding a new pet via frontend:


Pets

Add Pet | Sort by Name | Sort by Species | Sort by Age

Name: MyTestPet Species: Cat Skin: White Age: 2 Submit

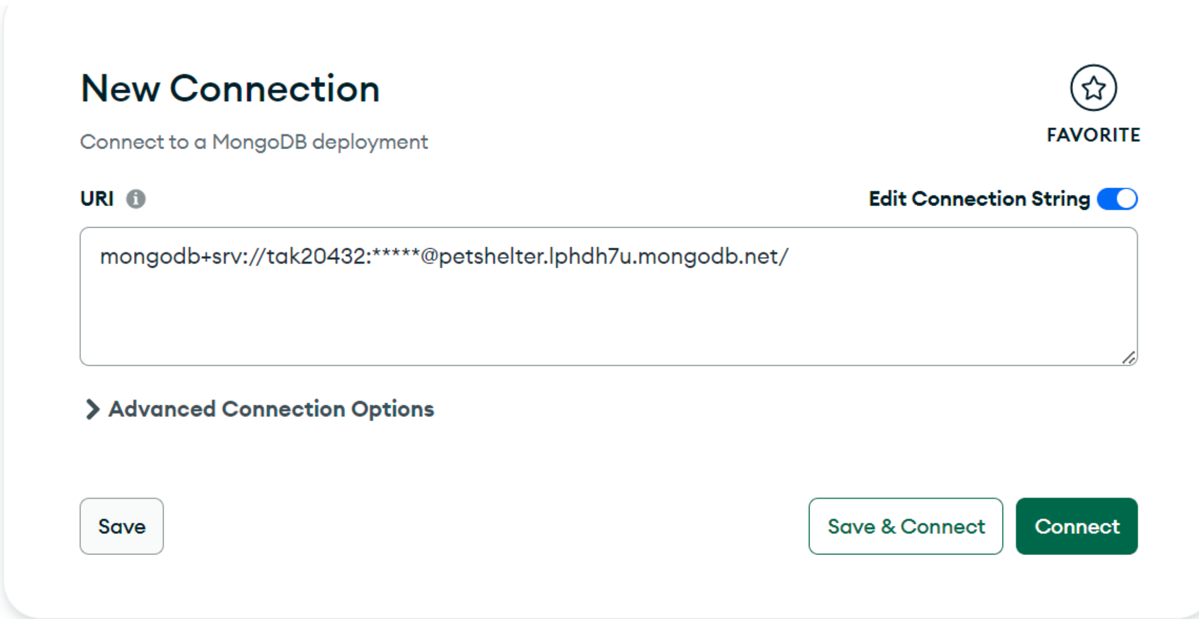
Successfully added to azure cloud:



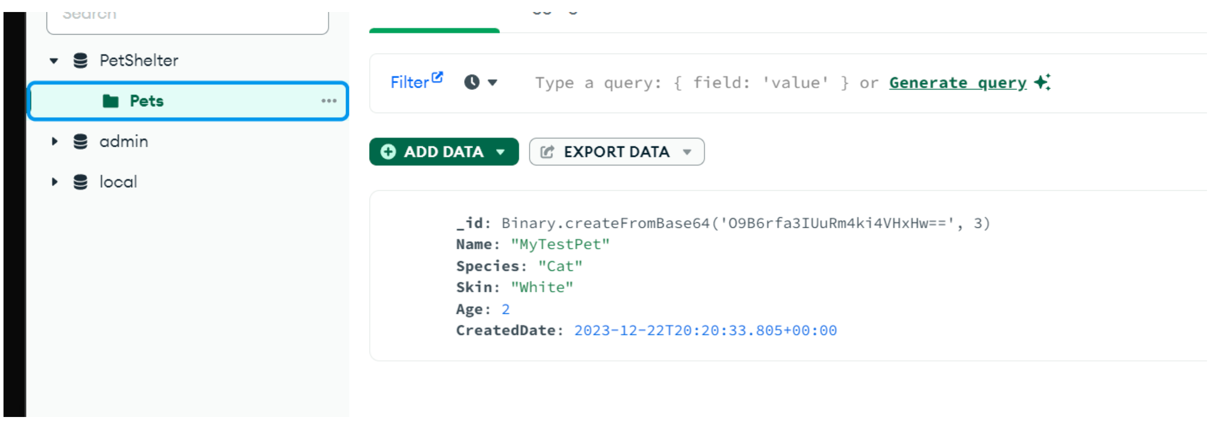
```
Select * FROM "Pets"
```

	"Id"	"Name"	"Species"	"Skin"	"Age"	"CreatedDate"	"ShelterId"
1	ad7ad03b-b7f6-4b21-919b-8922e151f11f	MyTestPet	Cat	White	2	2023-12-22 20:20:33.805159 +00:00	<null>

Successfully added to atlas cloud:



URI: `mongodb+srv://tak20432:*****@petshelter.lphdh7u.mongodb.net/`

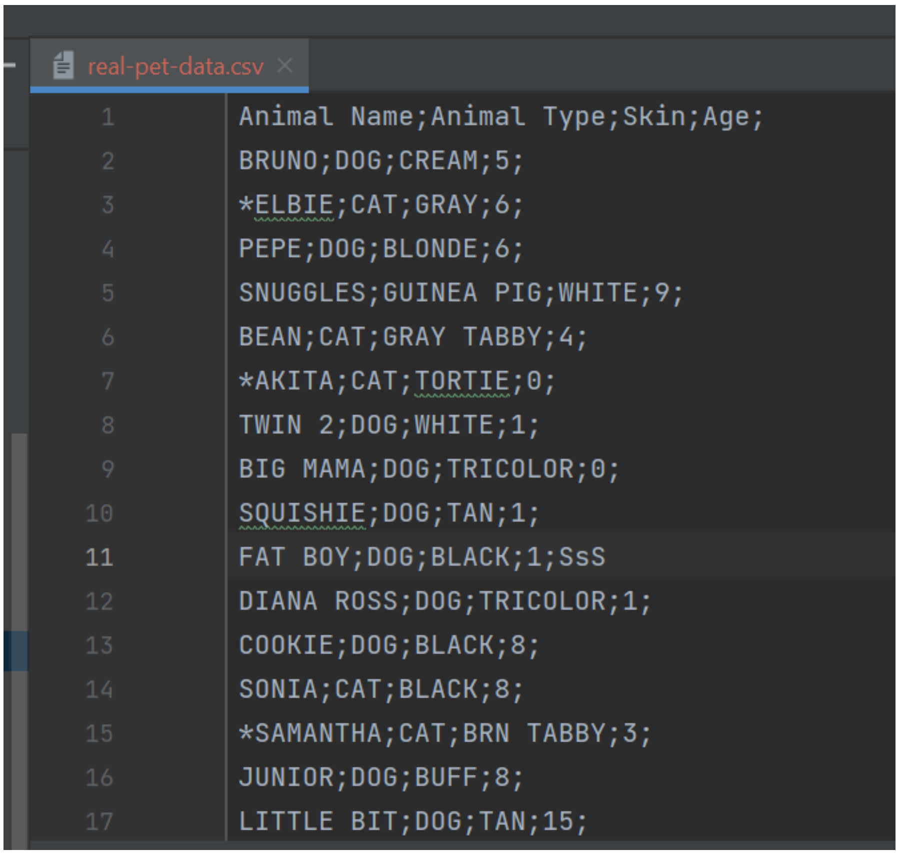


```
_id: Binary.createFromBase64('O9B6rfa3IUuRm4ki4VHxHw==', 3)
Name: "MyTestPet"
Species: "Cat"
Skin: "White"
Age: 2
CreatedDate: 2023-12-22T20:20:33.805+00:00
```

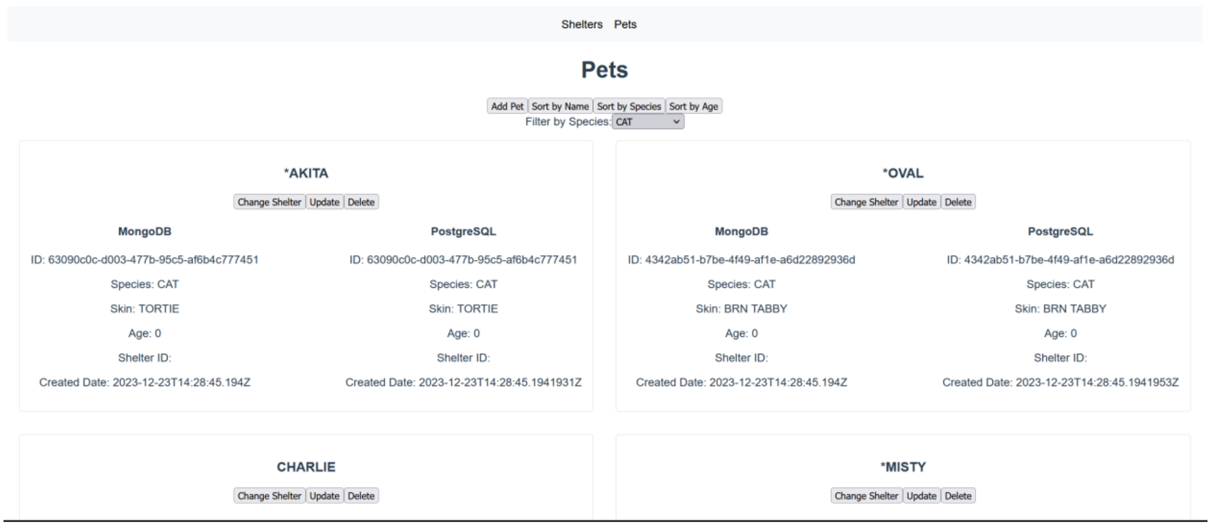
Special/Real Data (1 Point)

Pets uses real data from https://data.longbeach.gov/explore/dataset/animal-shelter-intakes-and-outcomes/table/?disjunctive.animal_type&disjunctive.primary_color&disjunctive.sex&disjunctive.intake_cond&disjunctive.intake_type&disjunctive.reason&disjunctive.outcome_type&disjunctive.outcome_subtype&disjunctive.intake_is_dead&disjunctive.outcome_is_dead

The data from there has been sorted and reformat to fit my model.



```
Animal Name;Animal Type;Skin;Age;
BRUNO;DOG;CREAM;5;
*ELBIE;CAT;GRAY;6;
PEPE;DOG;BLONDE;6;
SNUGGLES;GUINEA PIG;WHITE;9;
BEAN;CAT;GRAY TABBY;4;
*AKITA;CAT;TORTIE;0;
TWIN 2;DOG;WHITE;1;
BIG MAMA;DOG;TRICOLOR;0;
SQUISHIE;DOG;TAN;1;
FAT BOY;DOG;BLACK;1;SsS
DIANA ROSS;DOG;TRICOLOR;1;
COOKIE;DOG;BLACK;8;
SONIA;CAT;BLACK;8;
*SAMANTHA;CAT;BRN TABBY;3;
JUNIOR;DOG;BUFF;8;
LITTLE BIT;DOG;TAN;15;
```



Pets

Filter by Species: CAT

*AKITA

MongoDB
ID: 63090c0c-d003-477b-95c5-af6b4c777451
Species: CAT
Skin: TORTIE
Age: 0
Shelter ID:
Created Date: 2023-12-23T14:28:45.194Z

PostgreSQL
ID: 63090c0c-d003-477b-95c5-af6b4c777451
Species: CAT
Skin: TORTIE
Age: 0
Shelter ID:
Created Date: 2023-12-23T14:28:45.1941931Z

*OVAL

MongoDB
ID: 4342ab51-b7be-4f49-af1e-a6d22892936d
Species: CAT
Skin: BRN TABBY
Age: 0
Shelter ID:
Created Date: 2023-12-23T14:28:45.194Z

PostgreSQL
ID: 4342ab51-b7be-4f49-af1e-a6d22892936d
Species: CAT
Skin: BRN TABBY
Age: 0
Shelter ID:
Created Date: 2023-12-23T14:28:45.1941953Z

CHARLIE

*MISTY