Design Doc

Started from:

https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.0&tabs=visual-studio

Info for configuring Data: https://developer.okta.com/blog/2019/04/24/crud-app-aspnet-core-sql-server

**HTTP Core API (CRUD) (Admin Access) (Postman)**

GET /api/getAnimal

GET /api/getUser

POST /api/postUser

POST /api/postAnimal

POST /api/modifyAnimalOwnership

PUT /api/putUser

PUT /api/putAnimal

PUT /api/modifyAnimalOwnership

POST /api/modifyAnimalOwnership

DELETE /api/deleteUser (Override, soft delete)

DELETE /api/deleteAnimal (Override, soft delete)

DELETE /api/deleteAnimalOwnership (Override, soft delete)

**HTTP Functional API (General / Gameplay Access)**

/service/getAnimalsForUser (all animals a user owns)

/service/getAnimalForUser (A specific animal)

* ***Update “offline” state here?***

/service/feedAnimal

* ***Update hunger***

/service/strokeAnimal

* ***Update happiness***

**API Service**

* Mechanically needs to calculate updated State of Animal against Time Delta / Tick to change hunger / happiness. On “login” / pull of state.

User 1-\* Animals

User 1-\* AnimalOwnership \* - 1 Animal

User 1-\* AnimalOwnership \* - 1 Animal 1 – 1 AnimalState (Configurable, Change Log)

**Class Encapsulation**

Animal DTO

* Name
* Type
* Rate of Happiness decrease
* Rate of Hunger increase
* Happiness
  + Stroke
* Hunger
  + Feed

**Persistance**

User (ID (long (Primary Key)), Username (String), isDeleted bool DEFAULT FALSE)

AnimalOwnership (ID (long (Primary Key), UserId (long (Foreign Key)), AnimalId (long (Foreign Key), Owned Animal Name (String), isDeleted (bool DEFAULT FALSE)),

Animal (ID (long (Primary key))), Animal Type (String), isDeleted (bool DEFAULT FALSE))

*AnimalState(ID (long (Primary Key), AnimalId(long (Foreign Key), HappinessDefault (int), HappinessDecrease (int), HungerDefault(int), HungerIncrease(int), Updated (Timestamp))*

**Tasks**

**Min**

Setup (Default API, DB / Persistence connecting / Structure) (**Complete)**

Implement User Model (with default data) (**Complete)**

Implement Animal Model (with default data) (**Complete)**

Implement Basic CRUD (**Complete)**

**MVP**

Implement feeding functionality (**Complete)**

Implement stroking functionality (**Complete)**

Implement Core Service functionality (degrading mechanics against time) (**Complete)**

**Stretch**

Advanced CRUD (modifying defaults / degradation values)

Audit trail (Who’s modifying Core state changes).

**Discussion points**

Routing

Storage of Defaults / State values (Database? Can update while service is still live, could ingest into the class).

Feeding (could use objects to further decrease hunger).

Discussion point : https://stackoverflow.com/questions/57912012/net-core-3-upgrade-cors-and-jsoncycle-xmlhttprequest-error