MediaTonic Animal Game API Guide

Notes:

* Built using .NET Core 2.1
* Since this was a small-scale solution, I only used 1 project solution. For a real-world application requiring DB storage via Entity Framework I would use a clean architecture structure of having infrastructure dealing with data, core project and then API/Web layer to deal with presentation. Domain Driven Design approach to separate the concerns of the application using things Interface Service Pattern for business logic.
  + There wasn’t a need to over engineer a solution.
* Dealing with the animals being. I would normally run background scheduler method to update the ratings on the animals. It’s not a requirement I’ve really had to program around before just because of the type of projects I’ve built.
* I use swagger now with all Web API projects, it’s good for passing off to front-end devs and using a tool like NSwag it can create the models required for which presentation/front-end is going to be interacting with the WebAPI. Swagger is set to root of the application.
* I use a single Animal class and then have different animal types with an Enum type which then could be used to switch through what animal needed to be created. This approach means different types of animals could then have different actions. E.g. A dog could have a bark action, while a rabbit wouldn’t. Could use abstract class and then have derived classes but would require more thought around the design within the timeframes.
* Used xUnit for testing logic and WebAPI Integration while developing the solution using Live Unit Testing (I have a copy of VS 2017 Enterprise).
* Neutral Rating is 50 with maximum of 100 and minimum of 0.
  + Happiness rating increases with petting and decreases over time.
    - The Happiness rating is updated whenever it’s read and when the animal
  + Hungriness rating decreases with feeding and increases over time.
* The hungrinessratings are calculated by the rate set to them and then multiplied by mins since they were last fed.
  + Ditto for Happiness except they will be less happy as time goes on.
* One issue I have is .NET Core doesn’t have the Moles framework, so I can’t change the SystemTime without changing how I’ve built solution around Factory Pattern for creating objects for injecting a system.datetime dependecy, this wasn’t apparent until I created the tests for the solution. So, for 2 of the tests, I’ve had to put a Sleep Timer in, if I were to progress this solution then this would be changed.
* Static Lists done for temporary of users and animals.
* ViewModel used for mapping class objects to those displayed via the API Get Requests.
* Guid ID’s generate which can then be passed in via the API for when wanting to interact with a pet.
* Lots of assumptions regarding logic of the app and how the API should work.
  + No user id required of user when getting pet from API (created some methods to show how it might work).
  + I would ask lots of questions if this were a production app about logic, business rules etc.
* No interfaces required for animal and user since the animal and users are currently the final class required, however if these were to be extended then use of interfaces and abstract classes would be used. Just thinking around SOLID principles.

Usage:

* CLI:
  + Go to route of project (src/ MediaTonic.AnimalGame).
  + dotnet run.
    - <https://localhost:5001>
    - <https://localhost:5000>
* VS:
  + Set MediaTonic.AnimalGame.API as startup project.
  + Run
  + <https://localhost:44353> (HTTPS)
  + <http://localhost:5937>

This will bring up the swagger menu which will detail the different API Controllers, Methods (with what parameters need passing and their datatypes) and also details of the viewmodels (these could be called APIModels as well) with their datatypes.

Test Data to Use:

* User API Controller
  + Method: /api/User/GetByUserName
  + Username: mn123

From getting the user will have a default animal when can be used. The happiness and hungriness will update for every minute passed when retrieved via the API via the animal controller or when a user is retrieved which also details their animals in a viewmodel.

The user and animal both have post methods to create new objects. The animal controller then also has two httpput methods for feeding and petting the animal, also some optional methods are made for retrieving the animal with some user parameters (id and username) to show if authentication were to be used.