Vanier College

Computer Science Department

Web Services 420-551-VA

Section 00001

Project Proposal

Emmy Ea

Jiamin Yuan

Konstantinos Nikopoulos

TABLE OF CONTENTS

PROPOSAL IDEA

Our RESTful Web service shows animal records at shelters. It will display animals that are lost, found or adoptable. Our web service will allow sorting animals by category.

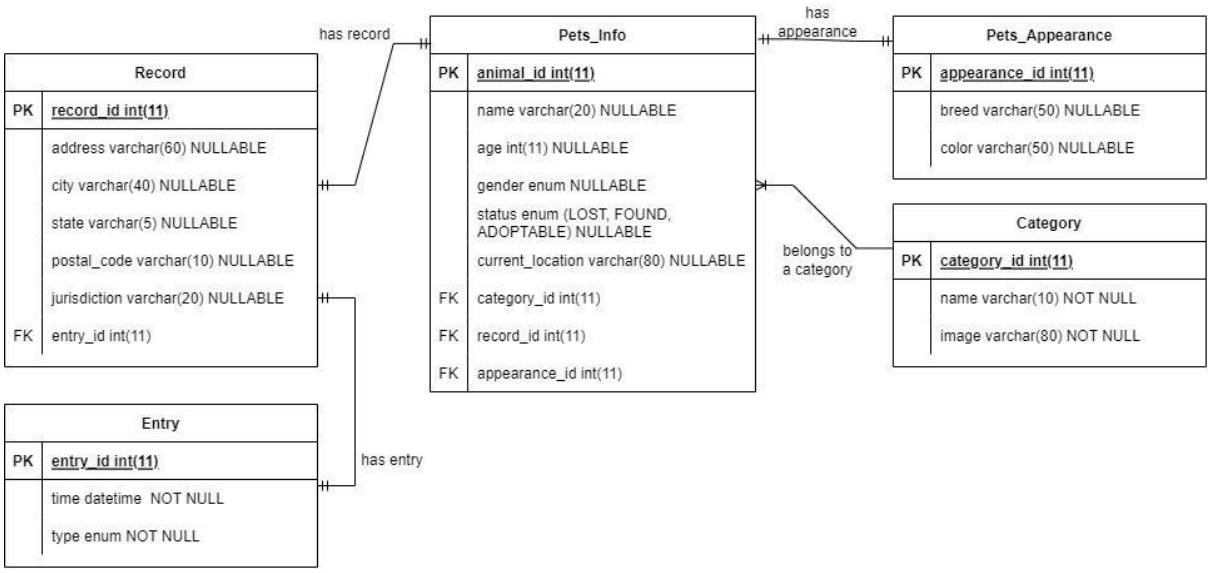
Our media type is of JSON format.

ERD

https://drive.google.com/file/d/13N6eAcmQmnF-kng8evjTvDArNpGCtJM4/view?usp=sharing

GITHUB REPOSITORY

https://github.com/EaEmmy/pets-api.git



ERD RELATIONSHIPS

Pets_info - Pets_Appearance (one to one):

Animals have one appearance. Each animal has a unique appearance.

Pets_info - Record (one to one):

Animals have one record. A record belongs to one animal.

Record - Entry (one to one):

A record has one entry and an entry belongs to a record.

Start with start comparison

Pets_info - Category (many to one):

An animal belongs to one category. A category has more than one animal.

RESOURCES

Name	URI	HTTP Methods	Description
records	/records	GET, POST, PATCH, DELETE	A list of pets' current information
entries	/entries	GET, POST, PATCH, DELETE	A list of pets' entry
pets	/pets	GET, POST, PATCH, DELETE	A list of registered pets
appearances	/appearance	GET, POST, PATCH, DELETE	A list of pets' appearances
categories	/categories	GET, POST, PATCH, DELETE	A list of filtered pets

RECORDS

Subtype of Records:

Name	Type	Description
record_id	int	The record id of a pet
address	string	The current address of a pet
city	string	The current city of a pet
state	string	The current state of a pet
postal_code	string	The current postal_code of a pet
jurisdiction	string	The current jurisdiction of a pet
entry_id	int	The entry id of a pet (FK)

Filtering Records:

Parameters	URI	Type	Result

city	/record/{city}	String	Pets residing within the given city are included in the response
state	/record/{state}	String	Pets residing within the given state are included in the response
postal_code	/record/{postal_code}	String	Pets residing within the given postal code are included in the response
jurisdiction	/record/{jurisdiction}	String	Pets residing within the given jurisdiction are included in the response

ENTRIES

Subtype of Entries:

Name	Type	Description
entry_id	int	The entry id
time	string	The entry time of a registered pet
type	string	The type of entry of a pet

Filtering Entries:

Parameters	URI	Type	Result
type	/entries/{type}	String	Entries of given type are included in the response

PETS

Subtype of Pets:

Name	Type	Description
animal_id	int	The animal id of a pet
name	string	The name of a pet
age	string	The age of a pet

status	string	The status of a pet (lost, found or adoptable)
current_location	string	The current location of a pet
category_id	int	The category id of a pet
record_id	int	The record id of a pet
appearance_id	int	The appearance id of a pet

Filtering Pets:

Parameters	URI	Type	Result
name	/pets/{name}	String	Pets with the given name are included in the response
age	/pets/{age}	String	Pets with the given age are included in the response
gender	/pets/{gender}	String	Pets with the given gender are included in the response
status	/pets/{status}	String	Pets with the given status are included in the response
category	/pets/{category}	String	Pets with the given category are included in the response

APPEARANCES

Subtypes of Appearances:

Name	Type	Description
appearance_id	int	The appearance id of a pet
breed	string	The breed of a pet
colour	string	The colour of a pet

Filtering Appearances:

Parameters	URI	Type	Result
breed	/appearances/{breed}	String	Pets with the given breed are included in the response
colour	/appearances/{colour}	String	Pets with the given colour are included in the response

CATEGORIES

Subtypes of Categories:

Name	Type	Description
category_id	int	The category id of a pet
name	string	The category name of a pet
image	string	The image of a pet

Filtering Categories:

Parameters	URI	Type	Result
name	/categories/{name}	String	Pets with the given category are included in the response

RELATIONSHIPS

URI	Description
/records/{entry_id}	A specific entry for a record
/pets/{category_id}	A specific category for a pet
/pets/{record_id}	A specific record for a pet

/pets/{appearance_id}	A specific appearance for a pet

HTTP RESPONSE STATUS CODES

GET:

- 200: Ok
- 404: Resource not found

POST//PUT:

- 201: Created
- 403: Forbidden
- 422: Validation Fail

PATCH:

- 200: Ok
- 404: Resource not found
- 422: Validation Fail

DELETE:

- 202: Accepted
- 204: No Content

PAGINATION

URI	Headers/Parameter
/records	?page={page_number}&pageSize={page_size}
/entries	?page={page_number}&pageSize={page_size}
/pets	?page={page_number}&pageSize={page_size}
/appearance	?page={page_number}&pageSize={page_size}
/categories	?page={page_number}&pageSize={page_size}

Composite Resources

Animal Facts:
https://api.api-ninjas.com/v1/animals?name=
It provides interesting scientific facts on various animal species. We can use it as a science popularization of animal knowledge.

Cat Facts:
https://cat-fact.herokuapp.com/facts/
https://alexwohlbruck.github.io/cat-facts/
Provides facts about cats.

Dog facts:
http://dog-api.kinduff.com
Provides facts about dogs.

Pet Status Code (Images):
https://http.dog/
https://http.cat/
Provides status code prompts with dog/cat images.

Project Plan (List of Tasks)

Create Database
Create a database for our api on PhpMyAdmin. Create all the tables shown in our ERD.
Responsible: Jiamin

Populate Database
Populate our database with data relevant to our api.
Responsible: Jiamin, Emmy, Kosta

Implementing Routing and Callback
Implement necessary files for resource routing and callback.
Responsible: Kosta

Implement Controllers
Implement controllers for all resources.
Responsible: Jiamin, Emmy, Kosta

- records: Jiamin
- entries: Emmy
- pets: Kosta
- appearances: Jiamin
- categories: Emmy

Implement Models
Implement models for all resources.
Responsible: Jiamin, Emmy, Kosta

- records: Kosta
- entries: Jiamin
- pets: Emmy
- appearances: Kosta
- categories: Emmy

Implement Error Catching/Status Codes
Implement error exceptions and catching. Add relevant code, message and description.
Responsible: Jiamin, Emmy, Kosta

GET: - Emmy

- 200: Ok
- 404: Resource not found

POST//PUT: - Jiamin

- 201: Created
- 403: Forbidden

- 422: Validation Fail

PATCH: - Kosta

- 200: Ok
- 404: Resource not found
- 422: Validation Fail

DELETE: - Emmy

- 202: Accepted
- 204: No Content

Implement Main resources filters
Implement main resources filters for the resources' controllers, models and index.php.
Responsible: Jiamin, Emmy, Kosta

- records: Jiamin
- entries: Emmy
- pets: Kosta
- appearances: Jiamin
- categories: Emmy

Implement Subtypes/relationship filtering
Implement subtypes filters for the resources' controllers, models and index.php.
Responsible: Jiamin, Emmy, Kosta

- records: Kosta
- entries: Jiamin
- pets: Emmy
- appearances: Kosta
- categories: Emmy

Implement Middleware
Implement middleware and content negotiation.
Responsible: Jiamin, Emmy, Kosta

Implement Pagination
Implement pagination for page number and page size.
Responsible: Jiamin, Emmy, Kosta

Implement Composite Resources
Implement at least two composite resources from our list into our api.
Responsible: Jiamin, Emmy, Kosta