# RESTful API Project

# Custom API Specification Document

# Hector Avalos

# INFO 762 Interoperability

# Professor: Dr. Grover Walters

# Date: 4/27/2022

## Custom API Specification

My custom API accepts API data from the Driver Application displaying information from each of the three APIs, Fiscal API, Gamer Power API, and Cheap Shark API.

### The Python Custom API code

# Import Dependencies

from flask import Flask

from flask\_restful import Api, Resource, reqparse, abort, fields, marshal\_with

from flask\_sqlalchemy import SQLAlchemy

# define application and database variables

app = Flask(\_\_name\_\_)

api = Api(app)

app.config["SQLALCHEMY\_DATABASE\_URI"] = "sqlite:///database.db"

db = SQLAlchemy(app)

app\_version = "v1/"

# create the data definition

class DataModel(db.Model):

id = db.Column(db.Integer, primary\_key=True)

country = db.Column(db.String(100), nullable=False)

currency = db.Column(db.String(100), nullable=False)

exchange\_rate = db.Column(db.Integer, nullable=False)

title = db.Column(db.String(100), nullable=False)

worth = db.Column(db.Integer, nullable=False)

gameID = db.Column(db.Integer, nullable=False)

salePrice = db.Column(db.Integer, nullable=False)

# outputs to log/screen to verify data visually

def \_\_repr\_\_(self):

return f"DataModel(country = {country}, currency = {currency}, exchange\_rate = {exchange\_rate}, title = {title}, worth = {worth}, gameID = {gameID}, salePrice = {salePrice})"

# run this statement the first thme to create the database structure

# db.create\_all()

# handle the incoming data request with a parser

# arguments for a put request

data\_put\_args = reqparse.RequestParser()

data\_put\_args.add\_argument(

"country", type=str, help="Name of the country is required", required=True

)

data\_put\_args.add\_argument("currency", type=str, help="Type of currency", required=True)

data\_put\_args.add\_argument(

"exchange\_rate", type=int, help="Rate of exchange", required=True

)

data\_put\_args.add\_argument(

"title", type=str, help="Name of the title is required", required=True

)

data\_put\_args.add\_argument(

"worth", type=int, help="How much it is worth", required=True

)

data\_put\_args.add\_argument("gameID", type=int, help="GameID is required", required=True)

data\_put\_args.add\_argument("salePrice", type=int, help="Price of game", required=True)

# arguments for an update request

data\_update\_args = reqparse.RequestParser()

data\_update\_args.add\_argument(

"country", type=str, help="Name of the country is required"

)

data\_update\_args.add\_argument("currency", type=str, help="Type of currency")

data\_update\_args.add\_argument("exchage\_rate", type=int, help="Rate of exchange")

data\_update\_args.add\_argument("title", type=str, help="Name of the title is required")

data\_update\_args.add\_argument("worth", type=int, help="How much it is worth")

data\_update\_args.add\_argument("gameID", type=int, help="gameID is required")

data\_update\_args.add\_argument("salePrice", type=int, help="Price of game")

# Map the types to columns extracted from the database object

resource\_fields = {

"id": fields.Integer,

"country": fields.String,

"currency": fields.String,

"exchange\_rate": fields.Integer,

"title": fields.String,

"worth": fields.Integer,

"gameID": fields.Integer,

"salePrice": fields.Integer,

}

# Set up the Resource Functions for CRUD

class Data(Resource):

# GET (READ in CRUD)

# @marshal\_with serializes output from the DB as a dictionary (json object) so we can work with it in python

@marshal\_with(resource\_fields)

def get(self, data\_id):

result = DataModel.query.filter\_by(id=data\_id).first()

if not result:

abort(404, message="Could not find data with that id")

return result, 201

# POST (CREATE in CRUD)

@marshal\_with(resource\_fields)

def put(self, data\_id):

args = data\_put\_args.parse\_args()

result = DataModel.query.filter\_by(id=data\_id).first()

if result:

abort(409, message="Data id taken...")

data = DataModel(

id=data\_id,

country=args["country"],

currency=args["currency"],

exchange\_rate=args["exchange\_rate"],

title=args["title"],

worth=args["worth"],

gameID=args["gameID"],

salePrice=args["salePrice"],

)

db.session.add(data)

db.session.commit()

return data, 201

# PUT (UPDATE in CRUD)

@marshal\_with(resource\_fields)

def patch(self, data\_id):

args = data\_update\_args.parse\_args()

result = DataModel.query.filter\_by(id=data\_id).first()

if not result:

abort(404, message="Data doesn't exist, cannot update")

if args["country"]:

result.country = args["country"]

if args["currency"]:

result.currency = args["currency"]

if args["exchange\_rate"]:

result.exchange\_rate = args["exchange\_rate"]

if args["title"]:

result.title = args["title"]

if args["worth"]:

result.worth = args["worth"]

if args["gameID"]:

result.gameID = args["gameID"]

if args["salePrice"]:

result.salePrice = args["salePrice"]

db.session.commit()

return result, 200

# DELETE (DELETE in CRUD)

def delete(self, data\_id):

abort\_if\_data\_id\_doesnt\_exist(data\_id)

del Data[data\_id]

return "", 204

# Register the Resource called data to the API (remember to change versions when making changes for submission)

api.add\_resource(Data, "/" + app\_version + "data/<int:data\_id>")

# Run the API body

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)