Micheal Willard
932480626
CS 496
Assignment 3 Part 2: API

Explanation of the URL Structure

The URL structure of the API is pretty straightforward. URL routes a well mapped to specific interaction functions.

The base URL is: https://hw3api.appspot.com/

To access player related functions, these routes can be called:
https://hw3api.appspot.com/player

This URL will call the GET or POST function of the Player class. If an id is included after the /player/ route, it will look up that player. If no id is included it will present the user with all the existing player ids.

To access Game related functions, /game can be added. This will Create a game, when data is included, else return a list of the game ids. When an id is present, it will obtain the JSON data for that game.

To delete a Game, the route /delete/game/id is used. Similarly, to access the Attending and Not Attending functions, /a/game/id and /na/game/id route is used, respectively.

Description of RESTfulness

To qualify as RESTful, an API must meet a number of requirements.
Requirements:

- Client-Server: where the client handles the UI and the server handles the data
- Stateless: where no sessions are maintained
- Cacheable
- Layered
- Uniform Interface

The API created for this project is not RESTful, as it is not cacheable and not a layered system. To make this service cacheable, a simple marking could be added to note that it is non-cacheable. Alternately, code could be added to notify the client how long the data can be cached before it must be re-requested. To make this a properly uniform interface, the JSON would need to return URLs for accessing the layers and giving the client proper visibility of the interface.

This API is built with the server handling the data and leaving any UI duties to the client. All the API provides is data. There are no states maintained by this API, so no client state data is stored or created. The system is layered. The URL structure provides the client with their only interaction, they have no visibility of what goes on behind the scenes.

Changes Made to Schema

There were no changes made to the schema from last week. The database design was perfectly acceptable for working with the implementation. There was only a minor modification, where storing the date and time as a date time object was causing output issues within the the code. So date and time were separated out into a date and a time string.

Possible Changes

The system itself is pretty barebones at the moment. The changes that could be made would be basically for expanding the amount of data being stored in the database. These changes could also help expand the ability of the API to be used for multiple teams as opposed to just one. Removing the opp_name attribute from the Game objects and replacing it with away_team and home_team would be a start. The Player objects would also need to be updated to include a team attribute. At that point the API would be setup to handle unlimited teams.

Tests

Please see hw3CurlTest.txt for the actual text used to test the API.

Test #	Description	Passed	Error Thrown for Invalid Input
1	Create a game	Yes	400 for missing required values
2	Get all Game IDs	Yes	N/A
3	Get a game by ID	Yes	404 delivered
4	Delete a game by ID	Yes	Yes
4a	Receive Error for no game ID	Yes	Resource could not be found
5	Add an attendee	Yes	404 delivered for incorrect game id
6	Add a non-attendee	Yes	404 delivered for incorrect game id
7	Create a Player	Yes	Yes
7a	With just name	Yes	Yes
7b	With all attributes	Yes	N/A
8	Get Player by ID	Yes	Yes