Clinton Hawkes
CS493 Cloud Dev
HW3

Link to my API - http://hw3-hawkesc.wl.r.appspot.com

I will break my critique of the three endpoints into three paragraphs. One for each endpoint.

First endpoint is for “modify a boat”. This endpoint uses the PATCH method to edit the boat properties, and while this is okay to do, I don’t know if it is the best choice. We learned that PATCH is used to perform partial updates to resources and not alter unmentioned properties, while PUT is used to update all included attributes in the request and delete any current attributes that were not in the request. The API spec states that all attributes for the boat must be included in the request to modify the boat or the request will be denied. Since all the attributes are present and are being modified, I think that a PUT would be a better fit. The URL pattern does fit the REST paradigm though. The URL of /boat/<boat_id> fits the /collection/thing pattern perfectly.

The second endpoint is for “boat arrives at a slip”. This endpoint uses the PUT method to modify a slip and list the specified boat as being docked. Once again, we learned that PATCH is typically used to perform a partial update to the resource without affecting the attributes that were not included in the request. Since there is only one attribute being changed when this endpoint is called, I had originally concluded that a PATCH would have been better. Then I took another look at the URL /slips/<slip_id>/<boat_id>. This follows that pattern of /collection/thing/thing. This is not the typical REST pattern that we are taught. I looked at the “exceptions” category from week 2 lectures on URL, and it said that this is something that we may encounter if the URL is identifying a specific property of a “thing”. So if we think of the slip’s “current_boat” attribute as the thing we are modifying, then this URL pattern would follow the REST paradigm. Also, since the “current_boat” is the only attribute that can be changed with this URL/method, a PUT method seems to be best.

Third endpoint is for “boat departs from slip”. This endpoint uses the DELETE method to modify the “current_boat” attribute of the slip by removing the specified boat. This represents the boat departing the slip. A lot of the reasoning for this endpoint can be found in the second endpoint above, but I will review. For the URL, I think that this endpoint still adheres to the REST paradigm. We learned in the week 2 lectures on URL formation that there is an exception we may encounter that does not follow the /collection/thing/collection/thing pattern, but follows a /collection/thing/thing pattern. This is typically used when we are identifying a specific property of a thing. That specific property in this case is the “current_boat” attribute of the specified slip. So the URL pattern follows the RESTful principals. As for the method used to set the boat back out to sea, I think the DELETE method is appropriate. When a ship arrives at a slip, we “PUT” the ship’s information in the “current_boat” attribute of the slip. When a ship leaves the slip, we “DELETE” the ship’s information from the “current_boat” attribute of the slip.