Vehicle API User Story ([service]/users/[username]/vehicles)

1. The API allows a user to register a vehicle.
   1. It should return an OK and the URI of the new vehicle when the registration is successful.
   2. It should return a NOT\_AUTHORIZED when the user is trying to register a vehicle to another user.
   3. It should return a BAD\_REQUEST when the vehicle is null or empty.
   4. It should return a NOT\_MODIFIED when the vehicle name was registered by the user before.
   5. It should return a BAD\_REQUEST when the mandatory fields weren’t set.
   6. It should return a BAD\_REQUEST when some of the fields weren’t valid:
      1. Name should contain at least a character, without spaces (one word), only alphanumeric characters.
      2. Make should contain only alphanumeric characters. It could be null.
      3. Model should contain only alphanumeric characters. It could be null.
      4. Initial mileage should be a natural number. It could be null.
      5. Year should be greater than 1850 (first motored vehicle?). It could be null.
      6. Color should contain only alphanumeric characters. It could be null.
2. The API allows a user to update a registered vehicle.
   1. It should return an OK when the updating was successful.
   2. It should return an OK when the new vehicle name is a duplicated/existing name.
   3. It should return a NOT\_AUTHORIZED when the user is trying to update a vehicle he doesn’t own.
   4. It should return a NOT\_FOUND when the vehicle name wasn’t registered by the user before (not idempotent).
   5. It should return a BAD\_REQUEST when the vehicle is null or empty.
   6. It should return a BAD\_REQUEST when the new vehicle name is null or empty.
   7. It should return a BAD\_REQUEST when some of the fields weren’t valid (see the validation checklist for the vehicle properties in the registration story).
3. The API allows a user to delete a registered car.
   1. It should return an OK when the deletion was successful.
   2. It should return a NOT\_FOUND when the vehicle name doesn’t exist.
   3. It should return a NOT\_AUTHORIZED when the vehicle the user is trying to delete belongs to another user.
   4. It should return a BAD\_REQUEST when the vehicle name is not valid (see validation checklist in the registration story).
4. The API allows a user to list its registered vehicles with filtering.
   1. It should return an OK and a list of all the registered vehicles of the users when the filters are empty or null.
   2. It should return an OK and an empty list when the user doesn’t have any registered vehicles despite of the filters sent.
   3. It should return an OK and an empty list when there weren’t vehicles that match the criteria of the filters sent.
   4. It should return a BAD\_REQUEST when any of the filters had an invalid value:
      1. The subject should be a property of the vehicle.
      2. It should have a valid criterion: equals to, greater than, greater than or equals to, less than, less than or equals to, like.
      3. The criterion should be related to the nature of the property to filter (e.g. for a text, we shouldn’t apply the greater than or less than).
      4. The predicate should have the same nature of the subject/property to filter.
   5. It should return an OK and a list that matches the filter list sent.