Blossom api

If you want to use this service, you can relax because this is pretty easy. As you know, we only work with 8 entities (attribute details at project documentation) and in this api we provide all the means to have a full client application working using the business logic.

All the allowed services are shown below, if no optional arguments are stipulated it means all arguments are mandatory.

- User methods
    - GET
        - List<User> getAllUsers()
            - /users
        - User getUser(id)
            - /users/{id}
    - POST
        - User createUser({name, email, password, phoneNumber})
            - /users
            - optional arguments: phoneNumber
    - PUT
        - User updateUser({id, name, email, password, phoneNumber})
            - /users
            - optional arguments: name, email, password, phoneNumber
    - DELETE
        - void deleteUser(id)
            - /users/{id}
- Plant methods
    - GET
        - List<Plant> getAllPlants()
            - /plants

        - Plant getPlant(id)
            - /plants/{id}
    - POST
        - Plant createPlant({cientificName, englishName, phMax, phMin, humMin, humMax})
            - /plants
    - PUT
        - Plant updatePlant({id, cientificName, englishName, phMax, phMin, humMin, humMax})
            - /plants
            - optional arguments: cientificName, englishName, phMax, phMin, humMin, humMax
    - DELETE
        - void deletePlant(id)
            - /plants/{id}
- Parcel methods
    - GET
        - List<Parcel> getAllParcels()
            - /parcels
        - Parcel getParcel(id)
            - /parcels/{id}
    - POST
        - Parcel createParcel({owner, location, plantId})
            - /parcels
            - optional arguments: location, plantId (this means there is no plant in this parcel)
    - PUT
        - Parcel updateParcel({id, owner, location, plantId})
            - /parcels
            - optional arguments: owner, location, plantId
            - if plantId == -1, this means that there is no plant in this parcel
    - DELETE
        - void deleteParcel(id)

                - /parcels/{id}
- Ph sensor methods
    - GET
        - List<PhSensor> getAllPhSensors()
            - /phsensors
        - PhSensor getPhSensor(id)
            - /phsensors/{id}
        - Set<PhSensor> getPhSensorsByParcel(id)
            - /phsensors/parcel/{id}
        - List<PhSensor> getPhSensorByOwner(id)
            - /phsensors/owner/{id}
    - POST
        - PhSensor createPhSensor({parcelId})
            - /phsensors
    - PUT
        - PhSensor updatePhSensor({id, parcelId, ownerId})
            - ownerId is needed to verify if the client owns the parcel
    - DELETE
        - void deletePhSensor(id)
            - /phsensors/{id}
- Ph measure methods
    - GET
        - List<PhMeasure> getAllPhMeasures()
            - /phmeasures
        - List<PhMeasure> getAllPhMeasureBySensor(id)
            - /phmeasures/sensor/{id}
    - POST
        - PhMeasure addPhMeasure({sensorId, value})
- Humidity sensor methods
    - GET
        - List<HumSensor> getAllHumSensors()
            - /humsensors
        - HumSensor getHumSensor(id)
            - /humsensors/{id}

        - Set<HumSensor> getHumSensorsByParcel(id)
            - /humsensors/parcel/{id}
        - List<HumSensor> getHumSensorByOwner(id)
            - /humsensors/owner/{id}
    - POST
        - HumSensor createHumSensor({parcelId})
            - /humsensors
    - PUT
        - HumSensor updateHumSensor({id, parcelId, ownerId})
            - ownerId is needed to verify if the client owns the parcel
    - DELETE
        - void deleteHumSensor(id)
            - /phsensors/{id}
- Humidity measure methods
    - GET
        - List<HumMeasure> getAllHumMeasures()
            - /hummeasures
        - List<HumMeasure> getAllHumMeasureBySensor(id)
            - /hummeasures/sensor/{id}
    - POST
        - HumMeasure addHumMeasure({sensorId, value})
- Avaliation methods
    - GET
        - List<Avaliation> getAllAvaliations()
            - /avaliations
        - Avaliation getAvaliation(id)
            - /avaliations/{id}
        - Set<Avaliation> getAllAvaliationsByUser(id)
            - /avaliations/user/{id}
    - POST
        - Avaliation addAvaliation({userId, stars, comment})
            - /avaliations

- PUT
  - Avaliation updateAvaliation({id, stars, comment})
    - /avaliations
    - optional arguments: stars, comment
- DELETE
  - void deleteAvaliation(id)
    - /avaliations/{id}