Lambdas

## 1. common values

1. Inputs
   * language:
     + 2 characters
     + must match one in DB
     + empty defaults to ‘FI’
2. Returns
   * statusCode (always returned):
     + 200 on success
     + 400 on invalid parameter
     + 401 on user not logged in
     + 500 on DB or server error
     + 501 on not implemented

## 2. Lambda endpoints

## /reviews

* + Lambda: lambdaHandler
  + Inputs:
    - restaurantId
    - pageSize
    - pageNumber
  + Returns:
    - reviews
      * restaurant\_id
      * user\_id
      * posted
      * status
      * title
      * free\_text
      * rating\_overall
      * rating\_reliability
      * rating\_variety
      * rating\_service\_and\_quality
      * pricing
      * thums\_up
      * thumbs\_down

## /search

* + Lambda: lambdaHandler
  + Inputs:
    - pageSize
    - pageNumber
    - orderBy
    - maxDistance
    - minOverallRating
    - minReliabilityRating
    - minVarietyRating
    - minServiceAndQualityRating
    - maxPricing
    - cityName
    - globalDietId (array, must be set, but can be empty)
    - currentLatitude
    - currentLongitude
  + Returns:
    - restaurants
      * restaurant\_id
      * restaurant\_name
      * email
      * city\_name
      * website
      * street\_address
      * reviews
      * rating\_overall
      * rating\_reliability
      * rating\_variety
      * rating\_service\_and\_quality
      * pricing
      * trending
      * weighted\_reviews
      * weighted\_rating\_overall
      * weighted\_rating\_reliability
      * weighted\_rating\_variety
      * weighted\_rating\_service\_and\_quality
      * weighted\_pricing
      * weighted\_trending
      * votes\_rate
      * latitude
      * longitude
      * opens\_mon
      * closes\_mon
      * opens\_tue
      * closes\_tue
      * opens\_wed
      * closes\_wed
      * opens\_thu
      * closes\_thu
      * opens\_fri
      * closes\_fri
      * opens\_sat
      * closes\_sat
      * opens\_sun
      * closes\_sun

## /restaurant

* + Lambda: lambdaHandler
  + Inputs:
    - restaurantId
  + Returns:
    - restaurant
      * restaurant\_id
      * name
      * email
      * website
      * street\_address
      * geo\_location
      * rating\_overall
      * rating\_variety
      * rating\_service\_and\_quality

## /landing - deprecated

* + Lambda: lambdaHandler
  + Inputs: -
  + Returns:
    - restaurants
      * restaurant\_id
      * restaurant\_name
      * email
      * website
      * street\_address
      * geo\_location
      * rating\_overall
      * rating\_reliability
      * rating\_variety
      * rating\_service\_and\_quality

## /profile

* + Lambda: profileLambda
  + Inputs:
    - language
    - user\_id (optional if logged in)
  + Returns:
    - username
    - email
    - own\_profile boolean
    - diplay\_name
    - gender
    - image\_url
    - birth\_year
    - birth\_month
    - description
    - country\_id
    - city\_id
    - countries\_visited
    - cities\_visited
    - reviews: (count)
    - thumbs\_up
    - thumbs\_down
    - thumbs\_up\_given
    - thumbs\_down\_given
    - activity\_level
    - last\_active

## /profile/edit

* + Lambda: editProfileLambda
  + Inputs (any or all may be unset):
    - username:
      * 4 or more characters
      * alphanumeric and ‘-’, ‘\_’ -characters
      * throw 501
    - password:
      * whatever
      * throw 501
    - email:
      * whatever
      * throw 501
    - display\_name:
      * 4 or more characters
      * alphanumeric and ‘-’, ‘\_’ -characters
      * whitespace allowed
    - gender:
      * 1 char or null
    - image:
      * throw 501
    - birth\_year:
      * any int
    - birth\_month:
      * int between 1 and 12
    - description:
      * whatever
    - country\_id:
      * must match one in db
    - city\_id:
      * must match one in db
    - diet\_id:
      * throw 501
    - new\_diet
      * throw 501
  + Returns:
    - message : Operation completed successfully

## /user/create

* + Lambda: createUserLambda
  + Inputs:
    - username
      * not empty
    - cognito\_sub
      * not empty
    - email
      * not empty
    - language
  + Returns:
    - message : Operation completed successfully

## /user/delete

* + Lambda: deleteUserLambda
  + Inputs:
    - reviews
      * boolean
      * true deletes own reviews
    - permanent
      * boolean
      * true clears email, username and cognito\_sub from db
  + Returns:
    - message : User removed successfully

## /ownReviews

* + Lambda: getOwnReviewsLambda
  + Inputs:
    - offset
      * defaults to 0
    - limit
      * defaults to 10
    - status
      * unset, 0, 1 or 2
  + Returns:
    - review\_count
    - reviews
      * review\_id
      * status
      * posted (date)
      * title
      * restaurant\_id
      * name (restaurant)
      * image\_url (not done, probably array)
      * free\_text
      * rating\_overall
      * rating\_reliability
      * rating\_variety
      * rating\_service\_and\_quality
      * pricing
      * thumbs\_up
      * thumbs\_down
      * diets (glabal\_diet\_id array)
      * accepted (date) (if status = 1)
      * rejected (date) (if status = 2)
      * reason (if status = 2)

## /ownReviews/edit

* + Lambda: editReviewLambda
  + Inputs (may be unset if not said otherwise):
    - review\_id
      * must be set
    - title
      * whatever
    - images
      * throw 501
    - text
      * whatever
    - rating\_overall
      * float between 0 and 5
    - rating\_variety
      * float between 0 and 5
    - rating\_service\_and\_quality
      * float between 0 and 5
    - pricing
      * float between 0 and 3
    - diets
      * array of global\_diet\_id s
  + Returns:
    - message : Operation completed successfully

## /ownReviews/delete

* + Lambda: deleteReviewLambda
  + Inputs:
    - review\_id
  + Returns:
    - message : Operation completed successfully

## /ownDiets

* + Lambda: getOwnDietsLambda
  + Inputs: -
  + Returns:
    - selected\_diet\_id
    - own\_diets
      * diet\_id
      * global\_diet\_id
      * name
      * groups (food\_group\_id array)

## /ownDiets/edit

* + Lambda: editOwnDietLambda
  + Inputs:
    - diet\_id
    - groups (array) or global\_diet\_id
    - name
  + Returns:
    - global\_diet\_id

## /ownDiets/create

* + Lambda: createOwnDietLambda
  + Inputs:
    - groups (array) or global\_diet\_id
    - name
  + Returns:
    - global\_diet\_id

## /ownDiets/delete

* + Lambda: deleteDietLambda
  + Inputs:
    - diet\_id
  + Returns:
    - message: Diet removed succesfully

## /location/countries

* + Lambda: getCountriesLambda
  + Inputs:
    - language
  + Returns:
    - object
      * country\_id
      * name

## /location/cities

* + Lambda: getCitiesLambda
  + Inputs:
    - country\_id
      * must match one in db
    - language
  + Returns:
    - object
      * city\_id
      * name

## /diet/presets

* + Lambda: getPresetDietsLambda
  + Inputs:
    - language
  + Returns:
    - object
      * global\_diet\_id
      * name
      * groups (food\_group\_id array)

## /diet/all

* + Lambda: getAllDietsLambda
  + Inputs:
    - language
  + Returns:
    - object
      * global\_diet\_id
      * name
      * groups (food\_group\_id array)

## /diet/groups

* + Lambda: getFoodGroupsLambda
  + Inputs:
    - language
  + Returns:
    - object
      * food\_group\_id
      * name
      * groups (food\_group\_id array)

## /voteDiet

* + Lambda: voteDietLambda
  + Inputs:
    - restaurant\_id
    - diet\_id
    - up
    - down
      * both set: deletes vote
      * both unset: toggles vote
  + Returns:
    - message: Operation completed successfully

## /diet/createAdmin

* + Lambda: createDietAdminLambda
  + Inputs:
    - language
    - preset (boolean)
    - groups (array)
    - name
  + Returns:
    - global\_diet\_id

## /diet/create

* + Lambda: createDietLambda
  + Inputs:
    - language
    - groups (array)
    - name
  + Returns:
    - global\_diet\_id

## /diet/update

* + Lambda: updateDietLambda
  + Inputs: (any of edited parameters can be unset (name, groups, preset))
    - language
    - preset
    - groups (array)
    - global\_diet\_id
    - name
  + Returns:

## /postReview

* + Lambda: postReviewLambda
  + Inputs:
    - restaurant\_id
    - title
      * whatever
    - images
      * throw 501
    - text
      * whatever
    - rating\_overall
      * float between 0 and 5
    - rating\_variety
      * float between 0 and 5
    - rating\_service\_and\_quality
      * float between 0 and 5
    - pricing
      * float between 0 and 3
    - diets
      * array of global\_diet\_id s
  + Returns:
    - message: Operation completed successfully

## /dbUpdate – should be run on schedule

* + Lambda: aggregationLambda
  + Inputs:
  + Returns:
  + Lambda:
  + Inputs:
  + Returns: