Cloud Computing - coursework report

Anthony Sébert - 1705851

Statement of compliance

A statement of compliance listing all the implemented and un-implemented features. (i.e. a check-list of achieved functionalities)

Subscriptions, Requests, and Locations

You need to design Java classes to represent subscriptions, requests, and locations of users. Java objects created from these classes will be mapped into DynamoDB data items and stored.

Web Services

You need to design web services to support the required operations of the web app. Here are some operations potentially needed:

- To send/approve/deny a subscription request
- To check-in a user's current location
- To retrieve a list of subscriptions
- To retrieve a specification user's location
- To retrieve all friends' locations

The User Interface

The project provide a landing page with a login form to log in or register, created in pure HTML/CSS.

Implementation

A description of your Java classes modelling the subscriptions, requests, and user locations.

`User` class

Class representing a user as an object, with the following attributes and methods

attributes

- longitude (double) : The longitude of the user
- latitude (double) : The latitude of the user
- id (String) : The unique user id
- incoming_requests (List) : List all requests that have not yet been accepted/rejected by the user
- subscriptions (List) : List all users that have accepted the user request
- subscribers (List) : List all users whose requests have been accepted by the user
- outgoing_requests (List) : List all request from the user that have not yet been accepted/rejected by other users

methods

- User() : Need this default constructor for AWS SDK. Create a User object
- User(String _id) : Create a User object
- longitude() (double) : Get a user's longitude
- latitude() (double) : Get a user's latitude

  /** * @return the user's id * @see #id */ @DynamoDBHashKey(attributeName = "id") @DynamoDBAutoGeneratedKey public String id() { return id; } /** * @return the user's incoming requests * @see #incoming_requests */ @DynamoDBAttribute(attributeName = "incoming_requests") public List incoming_requests() { return incoming_requests; } /** * @return the user's subscriptions * @see #subscriptions */ @DynamoDBAttribute(attributeName = "subscriptions") public List subscriptions() { return subscriptions; } /** * @return the user's subscribers * @see #subscribers */ @DynamoDBAttribute(attributeName = "subscribers") public List subscribers() { return subscribers; } /** * @return the user's outgoing requests * @see #outgoing_requests */ @DynamoDBAttribute(attributeName = "outgoing_requests") public List outgoing_requests() { return outgoing_requests; } // setters /** * Set the user's longitude attribute * @param _longitude the new longitude value * @see #longitude */ public void set_longitude(double _longitude) { longitude = _longitude; } /** * Set the user's latitude attribute * @param _longitude the new latitude value * @see #latitude */ public void set_latitude(double _latitude) { latitude = _latitude; } // incrementers /** * Add a user id to the incoming requests list * @param id the user id to add * @return true if the operation is a success, false otherwise * @see #incoming_requests */ public boolean add_incoming_request(String id) { return incoming_requests.add(id); } /** * Add a user id to the outgoing requests list * @param id the user id to add * @return true if the operation is a success, false otherwise * @see #outgoing_requests */ public boolean add_outgoing_request(String id) { return outgoing_requests.add(id); } // decrementers /** * Remove a user id from the incoming requests list * @param id the user id to remove * @return true if the operation is a success, false otherwise * @see #incoming_requests */ public boolean reject_incoming_request(String id) { return incoming_requests.remove(id); } /** * Remove a user id from the outgoing requests list * @param id the user id to remove * @return true if the operation is a success, false otherwise * @see #outgoing_requests */ public boolean outgoing_request_rejected(String id) { return outgoing_requests.remove(id); } // movers /** * Move a user id from the incoming requests list to the subscribers list * @param id the user id to move * @return true if the operation is a success, false otherwise * @see #incoming_requests * @see #subscribers */ public boolean accept_incoming_request(String id) { return subscribers.add(incoming_requests.remove(incoming_requests.indexOf(id))); } /** * Move a user id from the outgoing requests list to the subscriptions list * @param id the user id to move * @return true if the operation is a success, false otherwise * @see #outgoing_requests * @see #subscriptions */ public boolean outgoing_request_accepted(String id) { return subscriptions.add(outgoing_requests.remove(outgoing_requests.indexOf(id))); } // other /** * Converts a {@code User} to a {@code String} * @return A {@code String} representing the {@code User} */ public String to_string() { return "{" + id + "}(" + longitude + ":" + latitude + ")\n" + incoming_requests.toString() + "\n" + subscribers.toString() + "\n" + outgoing_requests.toString() + "\n" + subscriptions.toString(); }

API specification

Register

This service is responsible for the registration of new users. It is called by a **POST** request at */api/register*. The parameter `user_id`, extracted from the HTML form, corresponds to the user to create. The service returns:

- **409** if the user is already present in the database
- **200** if the operation have been performed successfully
- **500** in case of a server error (very likely during the communicatino with the database) On success, the service returns a JSON object containing a message that describes the result of the operation.

Login

This service is responsible for the login of existing users. It is called by a **POST** request at */api/login*. The parameter `user_id`, extracted from the HTML form, corresponds to the user to retrieve. The service returns:

- **200** if the operation have been performed successfully
- **404** if the user cannot be found On success, the service returns a JSON object containing a message that describes the result of the operation.

Logout

This service is responsible for logout an user. It is called by a **GET** request at */api/logout*. No parameter is required. The service returns **200** since the operation cannot fail On success, the service returns a JSON object containing a message that describes the result of the operation.

User

This service is responsible for the creation of users and the retrieval of user information. A **POST** request at */api/user* can be performed to create an user. The parameter `user_id`, extracted from the HTML form, corresponds to the user to create. The service returns:

- **201** if the operation have been performed successfully
- **400** if a problem occured On success, the service returns a JSON object containing a message that describes the result of the operation.

{id}

This service is responsible for retrieve user information. It is called by a **GET** request at */api/user/{id}*. The parameter `user_id`, extracted from the HTML form, corresponds to the user which information is requested.