EXERCISE 3: SOA

ENACHE IULIA, OGOKE FELICITY

15-4-2022

—

DISTRIBUTED SYSTEMS

WEB SERVICE/ FRAMEWORK

WHY WE CHOSE REST?

- Compared to gRPC and SOAP we are more familiar with REST which we worked with during our second semester.

- Rest is more flexible in that it supports a variety of data formats, rather than requiring XML. JSON is easier for us to read and write than XML.

- Rest API's can also offer better performance than SOAP because they can cache information.

- SOAP messages utilize a lot of bandwidth. So, in case if there might be a problem of resources and bandwidth, REST is a better option.

- The coding of REST APIs and services is far more seamless than SOAP.

AUTHENTICATION/AUTHORIZATION SERVICE

CLASSES:

- class Registration (Resource) – class containing method for registering new users into the system.
- class Login (Resource)- class containing method for logging existing users into the system and providing them with a token.

FUNCTIONS:

- def validate_registration(data)- function that ensures both password and role of users are provided during registration.

- def validate_login(data) - function which ensures the validation of users during login. It checks if password and username are provided, and that the password provided is correct.

- def emit_token(data) - function which creates a token by concatenating a user's role with a random string.

- def verify_token (username, token) - verifies if token is correct for the user

- def post (self, username) - method of the Registration class that creates a user into the system using their username.

- def put(self) - method of the Login class that logs validated users into the system and emits a token.

MASTER DATA SERVICE

CLASSES:

class Jobs (Resource)
class Results (Resource)

FUNCTIONS:

- def create_response(message)- function which returns a requests source, destination, and message body.

- def validate_post_job(data)- a function which validates and ensures all meta data information are provided when posting a job.

- def validate_get_delete_job_result(data)- a function which validates and ensures all necessary meta data information are provided when getting or deleting a job or a result.

- def validate_update_job(data)- a function which validates and ensures all necessary meta data information are provided when updating a job.

- def validate_user_permission_master_data (username, token)- validates that only users of roles “managers” and “administrators” are allowed to use the service. It also checks if their tokens are valid.

- def create_job(data) – Creates a new job and saves it to database.

- def fetch_job(job_id)- function returning a saved job given its job ID

- def update_job(data) - updates whatever changes are made by the client in the database

- def delete_job(data) – deletes a job given by its ID in the database.
- def post(self)- method of Job class that adds a job to the database after verification.
- def get(self)- method of Job class that returns a job by ID
- def put(self) - method of Job class that updates a job
- def delete(self) - method of Job class that deletes a job
- def validate_post_update_result(data)- a function which validates and ensures all necessary meta data information are provided when updating a result of a job.
- def create_result(data) - function that creates a new result and stores it in a database
- def fetch_result(job_id) - function returning the result of a job by its job ID.
- def update_result(data) – updates changes made by users on a result.
- def delete_result(job_id) – deletes the result of a job from database
- def post(self) - method of class Result that adds the result of a job to database if all metadata is provided and validation is complete.
- def get(self) - method of class Result thatreturns the result of a job
- def put(self) - method of class Result that updates a result of a job
- def delete(self) – method of class Result that deletes a result of a job.

MODELS.py FILE

We used the Model as a declarative class of the SQLAlchemy object, db, to create the two models.

class Job(db.Model) – builds the Job model for the database, containing 6 columns: id, username, timestamp, status, date_range and assets.

class Result(db.Model) - build result model for the database, containing the 3 columns Job ID, timestamp, assets.

MODULES

- Flask [Requests, Jsonify]
- Flask_restful [Resource]
- uuid
- Flask_sqlalchemy