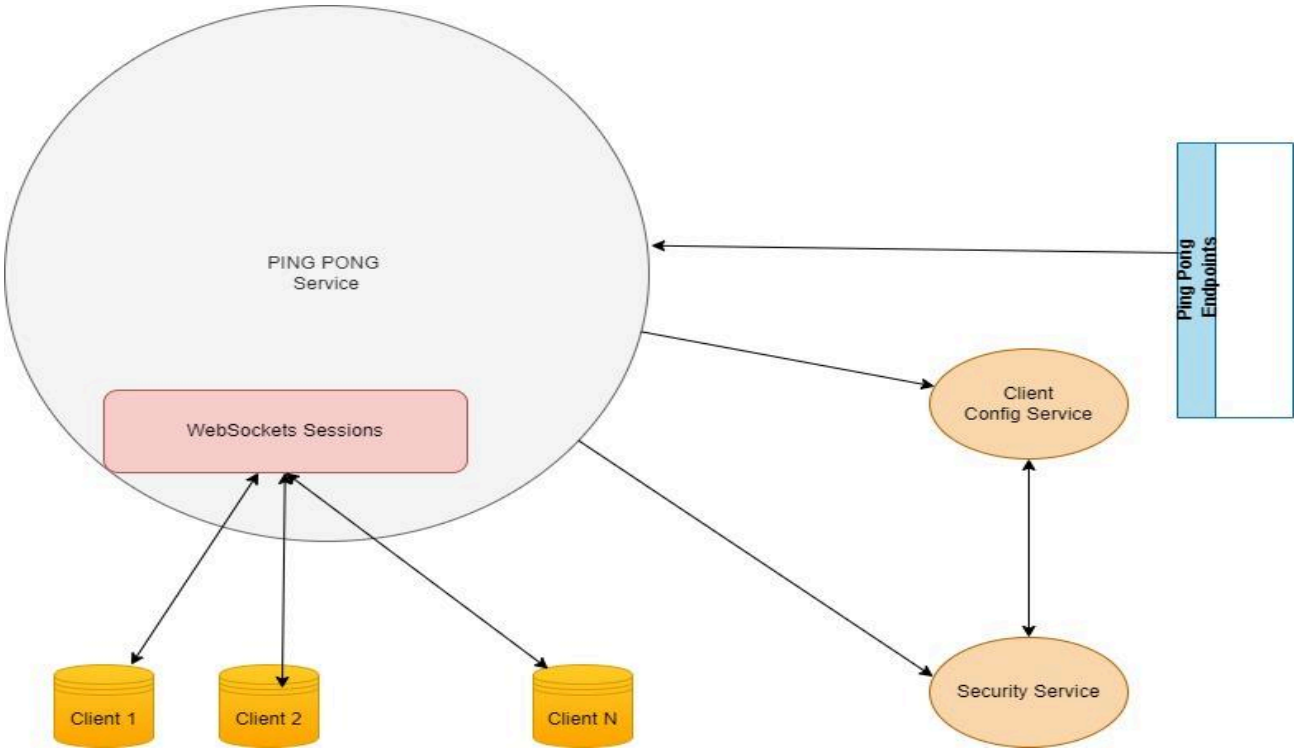
FoodTec Ping-Pong Service

You are requested to create the Ping Pong service echo-system presented in the following diagram.



Requirements

- All services/clients are SpringBoot applications running on Java 17+
- Ping Pong service is running on localhost:8080
- Client Config service is running on localhost:8090
- Security service is running on localhost:8070
- Security service exposes the following endpoints
    - Get /security/{application-id}/token . It returns a String (length 10) to be used as authorization token. You can hardcode the application-id/token values into your code (assign an application id to each service).
    - Get /security/{application-id}/valid/{token}. It returns 200 if this token is valid for this application-id (from the hardcoded values). An appropriate response code if not the token is not valid for the application
- Ping Pong service exposes the following web socket (STOMP) URL `"ws://localhost:8080/socket/ping-pong"`
- Client applications are connected into Ping-Pong service with a WebSocket and maintain the connection open. If disconnected they must attempt to reconnect.
- Clients application provide the following headers during their connection
    - “Auth-id” (String with a standard length of 10)

    - “Client-id” (int)
- Before Ping-Pong service accepts the ws connection is must consult Client Config service if a client is allowed to connect (based on header values)
- Ping-Pong service must acquire a token from the Security Service. It will use this token when communicating to Client Config Service
- After successful connection to Ping-Pong application clients subscribe to the following topic ‘/acceptance/ping-pong’
- Client config service maintains the following table(Hardcoded) of information
  That tell us if a client is allowed to connect/ allowed to ping/allowed to pong

Client ID	Can Ping	Can Pong	Can Connect	Auth-id
123	yes	no	yes	abcd
122	yes	yes	yes	abbc
121	yes	yes	no	aabb

- Client config service must expose this information in an endpoint so Ping-Pong service can query about the rights of each clients (You will need to define the endpoint and request/response between the 2 services)
- Ping-Pong Service exposes the following endpoints (no security on any of them)
    - /acceptance/{client-id}/ping POST
    - /acceptance/{client-id}/pong POST
    - /connection/{client-id}/drop PUT
- When ping and pong endpoints are called Ping-Pong service
    - Must check if the client is connected. If not return appropriate response
    - If connected, consult Client-Config-Service to see if it can ping or pong. If not, return an appropriate response.
    - If allowed to ping/pong the service must send a PING/PONG message through the web socket.
    - When a client receives the message through the web socket it must log it.
- When drop end point is called
    - If the client matched to the client-id is connected the service must drop the connection and return an appropriate response.
    - If the client matched to the client-id is not connected the service return an appropriate response

Suggestions

- Use best practices
- Include comments
- Logging is required
- If you need to make assumptions please document them

You will need to provide your implementation as a GRADLE project with a README file and it should be importable to Eclipse or Intellij.