ALLIANZ ARCHITECTURE TECHNICAL TEST

Challenge to apply Software Architecture position

Tasks

###Task 1:

- Create an API to maintain the Entities (CRUD).
  - Decide on your own how the methods should look like.

    > I have created endpoints suitable to application functionality taking care of business methods

  - Add example data to the resources folder.

    > In the Swagger are the DTO models that indicate how to interact with endpoints. The application use a database h2 that it's uploaded through script.sql (that you can find in the Resource folder) when application starts. You can find data examples to testing in the section called "Casos de uso". ###Task 2:

- On claim creation must negotiate an economical compensation agreement between parts.
  - If both policies are from the same insurance the agreement is automatic.

    > Is implemented.

  - If policies are from distinct insurances has a 33% of probability to accept the proposal.

    > Own assumption: the guilty insurances has an amount that aloud them to accept an automatic agreement.

- Create an endpoint to insurance API to receive the claim compensation.

  > Given that the system manages a list of negotiations proposals and reasons to reject/reach for every claim, it has this issue covered. Once the claim reaches the amount to collect, it gets order as last in the list. There is an endpoint to consult all the claims reach per company, to know the total amount in a fixed time.

- The compensation negotiation must reach an agreement.

  > Own assumption: The claims could have three states. Proposal, Reject and Reach.

- When agreement is reached can not be negotiated again.

  > Given that the system uses security and rolers, only a certain type and level of user can reach.

- Each compensation proposal must be less than the previous one.
- The details are up to you.

###Task 3 (Optional):

- This task is voluntary if you can't get enough of hacking tech challenges, implement security.
- Secure the API so that authentication is needed to access it . The details are up to you.
- Please include instructions how to authenticate/login, so that we can test the endpoints you implemented!

  > This task is implemented with jwt + role profiling.

  - All endpoints validate that the Authorization arrives to the Header with the jwt.
  - There is a published endpoint that allows login and validates against the database.
  - The jwt payload contains the user, the creation and expiration dates. This is the way the user its authenticated. After this the application obtains the role of the user to know if he is authorized to consume that endpoint.
  - The user is located in the Service layer to filter data according to his accessibility (for example, the logged-in user can only see the policies of the company it belongs to, and depending on his Role will be able to modify it)

  > End explication

Approach

> The concept from which the application was developed is based on the fact that insurance companies have employees/producers (users) who handle negotiations and customer claims. A client who has an accident, contacts his producer to initiate the claim. The client who is at fault for that accident also acts through his own producer.

Other assumptions

> The application may not be consumed by REST alone in the future. That's why from the service layer and downwards it only handles objects from the business model, leaving the DTO models only for the controller layer. This way, in the future it could generate a service wrapper and there would be no need to modify the application

Lines to improve:

> Start

- Increase code-coverage above 80%, especially in services.
- Develop global and customized exception management from the service layer down and for ResponseEntity.
- Extend automatic reach functionality by amount to apply to different insurance and policy types.
- All messages obtained from properties should have their language with Internationalization I18N.
- PUT methods

> End explication

Use Case

Normal Flow:

1. The client communicates with his insurance producer indicating what he had an accident by providing the policy number of guilty.
2. The producer initiates the claim by entering the policy data of both and the proposed amount of compensation.

    iii. If the policies are from the same insurer, it is automatically accepted and closed. 4. If the policies are from different insurers and the amount is less than or equal to the acceptance amount, the company automatically accepted and closed. 5. The System updates the negotiation record with the data. 6. The system modifies the status of the claim to CLOSED. 7. End of use case.

    iv. The System updates the negotiation record with the data.

    v. The system modifies the status of the claim to CLOSED.

    vi. End of use case.
3. The System updates the trading record with the data received.
4. The system initiates a claim in the OFFERED state.
5. The producer of the guilty can reject or accept the offer of negotiation.

If PRODUCER ACCEPT THE OFFER: 6. The System updates the negotiation record with the data. 7. The system modifies the status of the claim to CLOSED. 8. End of use case.

If PRODUCER REJECT THE OFFER: 6. The System updates the negotiation record with the data. 7. The system modifies the status of the claim to REJECTED. 8. The victim's producer makes a new offer 9. The process returns to point 3 of **NORMAL FLOW**

Frameworks implemented on this solution

1. SpringBoot
    i. Security
    ii. JPA
    iii. Web
    iv. Test
2. JUnit
3. Mokito
4. Lombok
5. H2 BD In memory
6. Swagger

Services

Services are exposed in Swagger.

DataBase Access

The app Allow access to Database for Web Console. This configuration is in security class. If you need access to database, run the application go to this Link. User is "sa" and passwors is "sa"

Users

There are nine users. three users for each of the companies. Each of them has a different profile, with the first on the list being the lowest, then the intermediate and finally the highest.

All passwords are "12345" to simplify.

The users are:

- Sallia
  - john (user)
  - albert (pm)
  - nikola (supervisor)
- Rotular
  - richard(user)
  - gilfoyle(pm)
  - dinesh(supervisor)
- Xenix
  - steve(user)
  - stevej(pm)
  - ronald(supervisor)

For more information you can go to h2 console to view all tables and data.

Compile

```
mvn clean install
```

Run

```
CD into folder app
mvn spring-boot:run or java -jar assessment-1.0.0-SNAPSHOT.jar
```