Web Applications II 2023-24

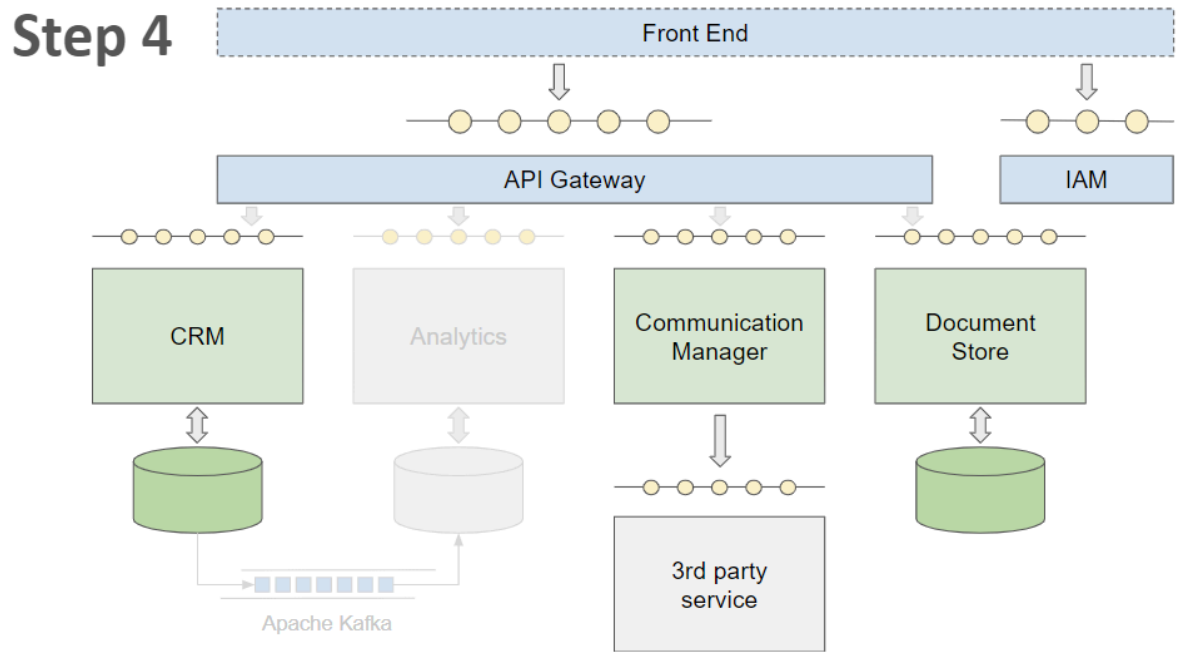
A system for temporary job placement services

Lab5 – Securing the system

Learning objectives

- Using Spring Cloud Gateway to create an API gateway for the system
- Using Spring Security framework to implement role-based access control
- Using Spring Security framework and Keycloak IAM to implement OAuth2.0 Authorization Code Flow authentication
- Enabling sign-in for users
- Creating the first front-end interface to interact with the system, based on a SPA

Description



In this lab, you will be tasked with implementing an API Gateway as part of the Customer Relationship Management microservice system. The API Gateway will not only manage the routing of requests to the various microservices but also serve as an OAuth client, interfacing with an Identity and Access Management (IAM) system for secure authentication and authorization. Ensuring the security of the system is fundamental, as it protects sensitive customer data - providing compliance with data protection regulations - and maintains the integrity and trustworthiness of the CRM platform.

You will use Keycloak as the IAM solution, which will handle token issuance and store sensitive information useful to authenticate users of the system (e.g., *passwords*). Additionally, you will implement role-based authorization on the various services and ensure JWT (JSON Web Token) verification. By the end of this lab, you will have a foundational API Gateway with integrated security features and a basic front-end for user authentication.

Steps

1. To accept the assignment, use GitHub Classroom as you did in previous labs: https://classroom.github.com/a/58TiBSsz
2. Set Up **Keycloak** as *IAM* Microservice
    a. Add it as a service of your docker compose file https://hub.docker.com/r/bitnami/keycloak/
        i. keycloak:
            image: keycloak/keycloak
            ports:
                - '9090:8080'
            environment :
                KEYCLOAK_ADMIN: admin
                KEYCLOAK_ADMIN_PASSWORD: password
            command: start-dev
    b. Through the admin console: https://www.keycloak.org/docs/24.0.4/server_admin/
        i. Create a new realm for your CRM system
        ii. Create *user* accounts (with credentials) and assign them appropriate roles (e.g., *guest*, *operator*, *manager*)
        iii. Set up a *client* in Keycloak to represent your *API Gateway*
        iv. Configure the client to use the OAuth 2.0 protocol, enabling only the *standard flow*
            1. *Valid redirect URIs: http://localhost:8080/login/oauth2/code/<client_name>*
            2. *Valid post logout redirect URIs: http://localhost:8080/*
        v. Take note of the client's secret from the Credentials section
3. Implement **API Gateway**

    a. Create a new microservice based on Spring Cloud Gateway.
       Implement basic routing functionality to forward requests to the appropriate microservices
    b. Integrate the API Gateway with Keycloak to act as an *OAuth client*
        i. Add dependencies, specify configuration through application.yml, and correctly configure the *SecurityFIlterChain*, to implement OAuth2.0 code exchange to obtain and verify tokens from Keycloak (using the *OidcAuthorizationCodeAuthenticationProvider*)
    c. Add a test endpoint, whose path is accessible only with authentication to verify your configuration
    d. Implement relying party-initiated logout
4. Implement **RBAC** in microservices
    a. Each microservice plays the role of an OAuth2.0 resource server. It has to enforce role-based access by checking the roles present in the JWT, so using a *JwtAuthenticationProvider*.
    b. Ensure each microservice verifies the JWT with the Keycloak public key before processing requests. Thanks to Spring Security framework each microservices can also easily check JWT validity and expiration.
    c. Configure the endpoints to allow only certain roles to perform specific actions. For example, the GET endpoints to retrieve contacts should be accessible to both *guest* and *operator* roles, while the PUT and POST endpoints to modify and create customers should be restricted to the *operator* role only.
5. Develop Initial **Front-End** as a *single-page application*
    a. Create a basic front-end interface allowing users to log in and log out (working with CSRF protection enabled).
        i. Use a framework of your choice (e.g., React + Vite + TS, Angular, Vue...)
    b. Display user information and roles upon successful login.
6. *(optional in this phase)* Add previous features
    a. Integrate the functionalities developed in the previous labs into the new system
    b. Ensure compatibility and secure communication between the new and existing components

Submission rules

- The work must be submitted by June, 13 23:59
- The last commit before the deadline will be evaluated. Alternatively, create a release and label it as “completed”.