

Especialización en Back End II

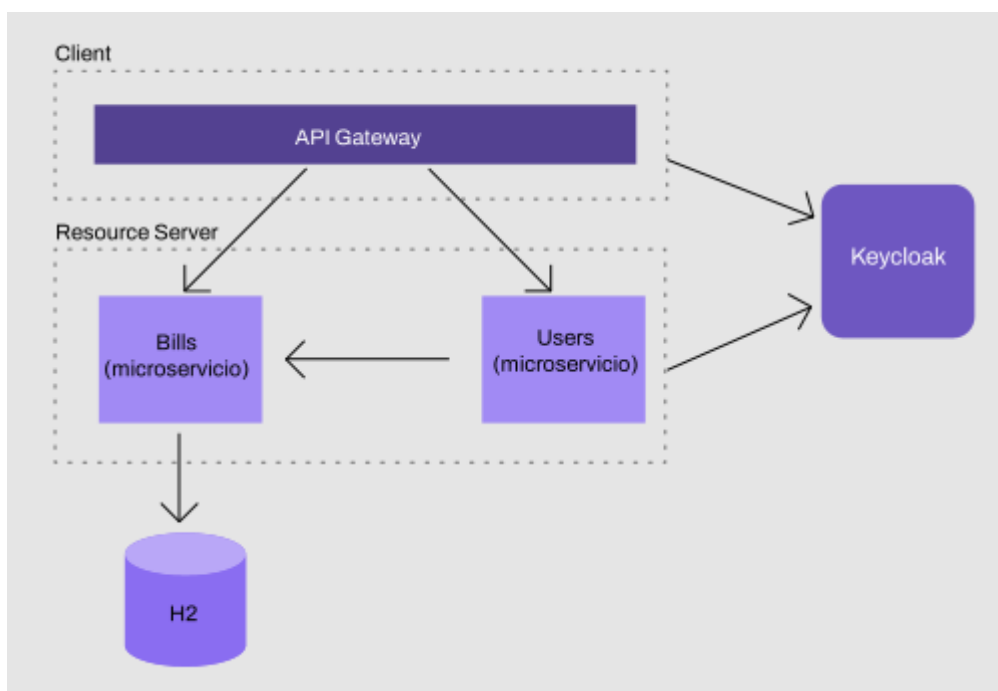
Trabajo práctico final

Llegamos al trabajo práctico final donde pondremos en práctica todos los conocimientos vistos a lo largo de la materia, tanto en clase como en Playground. Tal como pudiste observar en los criterios de evaluación, tendrás hasta el final de la próxima clase para realizar la consigna y recibirás la nota vía mail en el transcurso de la siguiente semana. ¡Éxitos!

Consigna

Continuando con la tarea propuesta en el trabajo práctico parcial, ahora trabajamos en el sistema de facturación. Para esta etapa vamos a trabajar en dos funcionalidades nuevas:

- Los diferentes proveedores de facturas podrán **dar de alta facturas**.
- Los **usuarios podrán buscar** sus facturas.





Tomando en cuenta el diagrama se nos pide:

En **Keycloak**:

- Crear un grupo llamado **PROVIDERS**.
- Asignar usuarios a este nuevo grupo.
- Crear un cliente que permita autenticarse utilizando las credenciales del cliente (client credentials).

En **bills-service**:

- Crear un endpoint que nos permita dar de alta facturas. Restricciones:
 - Solo los usuarios pertenecientes al grupo **PROVIDERS** podrán dar de alta facturas. Leer del JWT el listado de grupos para luego validar en el controller.
- Crear un endpoint que nos permita buscar facturas por ID de usuario. Restricciones:
 - Usuarios autenticados.

En **users-service**:

- Crear el microservicio y agregar un endpoint que nos permita buscar a un usuario y sus facturas.
 - Buscar el usuario por ID utilizando **Keycloak REST Admin Client**.
 - Para buscar las facturas, utilizar **Feign**. Configurar el cliente de Feign para autenticarnos y obtener el token de Keycloak cuando enviamos la petición.

En **API Gateway**:

- Agregar al ruteo el microservicio de usuarios.

TIP: Para obtener el access token y utilizarlo desde Postman podemos completar los siguientes campos en Postman y hacer clic “**Get New Access Token**”.

The screenshot shows the 'Configure New Token' dialog in Postman for an OAuth 2.0 authorization. The 'TYPE' is set to 'OAuth 2.0'. The 'Grant Type' is 'Authorization Code'. The 'Callback URL' is 'http://localhost:8090'. The 'Auth URL' is 'http://localhost:8082/realms/digital-house/protocol/openid-connect/auth'. The 'Access Token URL' is 'http://localhost:8082/realms/digital-house/protocol/openid-connect/token'. The 'Client ID' is 'gateway'. The 'Client Secret' is 'a5946L5jX0YKydqNxay68LKKU2BxOnof'. The 'Scope' is 'openid'. The 'State' is empty. The 'Client Authentication' is set to 'Send as Basic Auth header'. The 'Token Name' field is empty. The 'Authorize using browser' checkbox is unchecked.



Una vez que tenemos el token lo pegamos en:

POST http://localhost:8090/api/v1/bills Send Save

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies Code

TYPE: OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to:

Current Token: This access token is only available to you. Sync the token to let collaborators on this request use it.

Access Token: Available Tokens eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXZWQ6Ij09eyJndhYUT13VlgYnktSUENpNTd2... Bearer

Tests

Al buscar un usuario por ID, debemos obtener una respuesta similar a la siguiente:

```
{
  "id": "14def3b7-733d-4796-8ea2-29be94fb1988",
  "username": "tomas",
  "email": "pereyratomas18@gmail.com",
  "firstName": null,
  "bills": [
    {
      "idBill": "4b3c9a20-ebd6-4c0c-9537-c8469231996c",
      "customerBill": "14def3b7-733d-4796-8ea2-29be94fb1988",
      "productBill": "2",
      "totalPrice": 100.0
    }
  ]
}
```