

Projeto rasplus-api

O objetivo deste projeto é construir a aplicação do rasplus-api, desenvolvido nos cursos de spring rest e spring security para uma arquitetura de MS.

Sobre o Rasmoo Plus

O **Rasmoo Plus** é uma plataforma de cursos de programação baseada em assinatura, projetada para atender às necessidades de quem busca aprimorar suas habilidades na área. Os alunos podem escolher entre três tipos de planos: **mensal**, **anual** e **vitalício**, garantindo flexibilidade para se adaptar ao seu objetivo e orçamento.

Após fazer login, o usuário tem acesso à vasta biblioteca de cursos disponíveis e pode começar imediatamente suas aulas. Além disso, a plataforma permite a personalização do perfil, possibilitando a atualização de informações cadastrais de forma simples e prática.

Link das telas que serão desenvolvidas nos cursos de front

[Figma](#)

Repositório [ws-rasmoo-plus](#)

Repositório [rasplus-api-keycloak](#)

Para atender as telas do fluxo de login, criação e recuperação de conta, área logada com a listagem dos cursos e edição de dados cadastrados, inicialmente, foi projeto o seguinte design

Módulo 2 - [rasplus-customer-data-api](#)

Rotas:

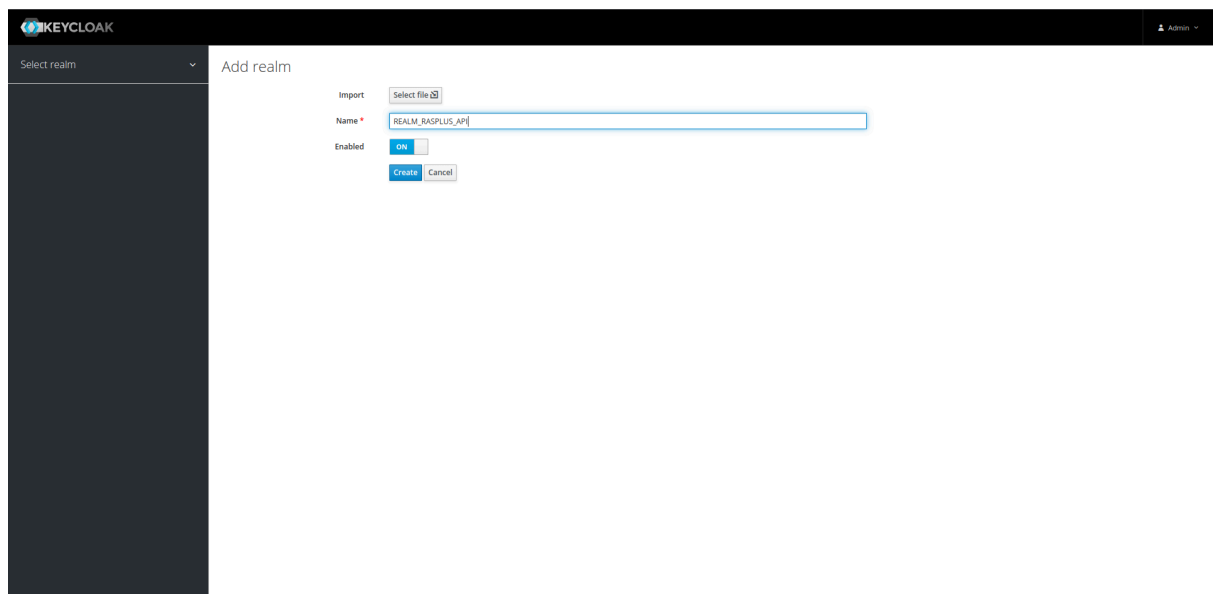
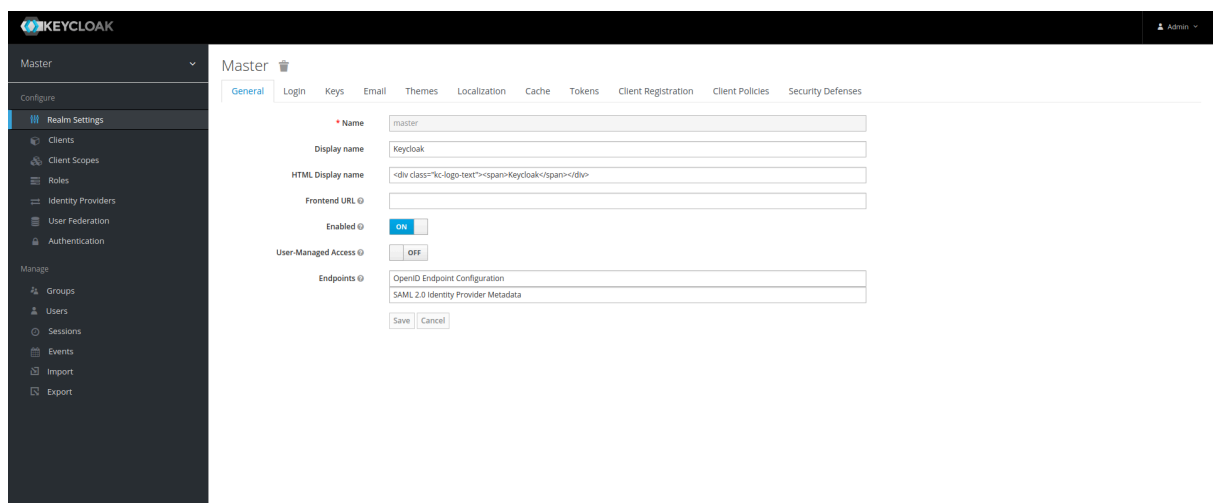
- (POST)/api/customer/v1/auth/: Realiza integração com o keycloak para autenticação
- (POST)/api/customer/v1/auth/recovery-code/send: Envia código de recuperação para atualizar credenciais
- (GET)/api/customer/v1/auth/recovery-code: Verifica se código de recuperação é válido
- (PATCH)/api/customer/v1/auth/recovery-code/password: Atualiza senha
- (POST)/api/customer/v1/auth/representations/credentials
- (PUT)/api/customer/v1/auth/representations/credentials/{email}

Github estrutura de branchs

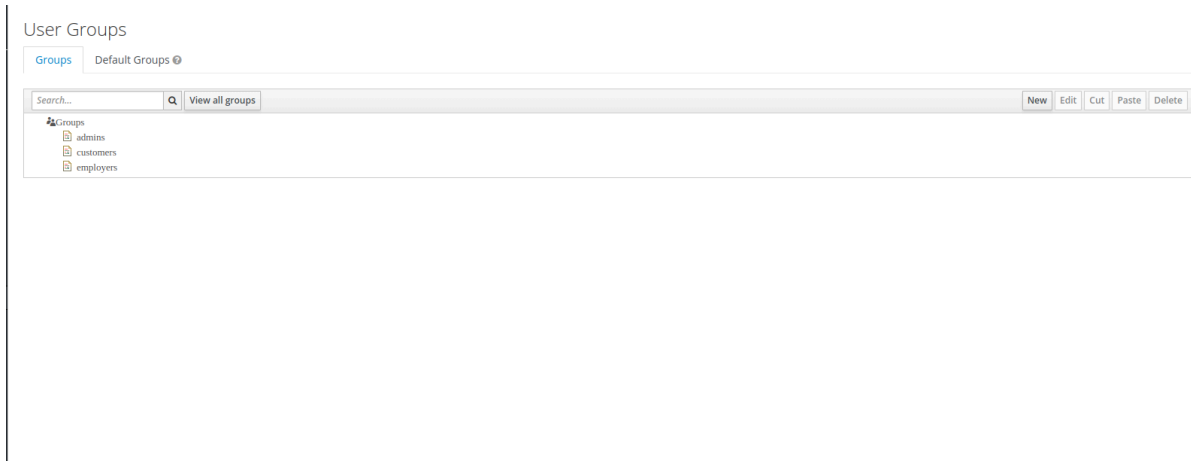
“class_{class_numer}-{class_description}”

Configuração do Keycloak

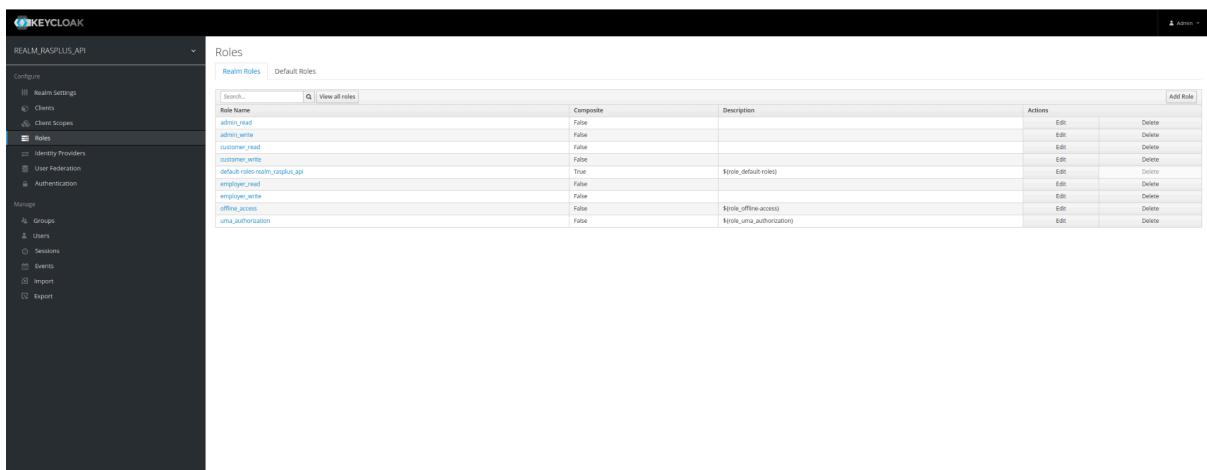
1 - Crie o Realm REALM_RASPLUS_API - Clique em Master > add realm > REALM_RASPLUS_API



2 - Crie os grupos “admins”, “employers” e “customers”. Vá até groups>new group> nome do grupo. Faça isso 3 vezes.



3 - Cadastre as roles admin_write, admin_read, customer_write, customer_read, employer_write, employer_read. Roles > add Role



4 - Crie o Client RASPLUS_API

The screenshot shows the Keycloak administration interface. The left sidebar is dark grey with the 'KEYCLOAK' logo at the top. Below the logo, the 'REALM_RASPLUS_API' is selected. The sidebar is divided into 'Configure' and 'Manage' sections. The 'Configure' section includes 'Realm Settings', 'Clients' (which is highlighted), 'Client Scopes', 'Roles', 'Identity Providers', 'User Federation', and 'Authentication'. The 'Manage' section includes 'Groups', 'Users', 'Sessions', 'Events', 'Import', and 'Export'. The main content area is white and titled 'Add Client'. It contains an 'Import' button with a 'Select file' link. Below this, there are three input fields: 'Client ID *' with the value 'RASPLUS_API', 'Client Protocol' with a dropdown menu showing 'openid-connect', and 'Root URL'. At the bottom of these fields are 'Save' and 'Cancel' buttons.

KEYCLOAK

REALM_RASPLUS_API

Configure

- Realm Settings
- Clients
- Client Scopes
- Roles
- Identity Providers
- User Federation
- Authentication

Manage

- Groups
- Users
- Sessions
- Events
- Import
- Export

Clients > Add Client

Add Client

Import [Select file](#)

Client ID *

Client Protocol

Root URL

[Save](#) [Cancel](#)

Configure o Client

Clients > RASPLUS_API

RASPLUS_API

Settings

Credentials

Keys

Roles

Client Scopes

Mappers

Scope

Revocation

Sessions

Offline Access

Clustering

Installation

Client ID

RASPLUS_API

Name

\$(client_account)

Description

Enabled

ON

Always Display in Console

OFF

Consent Required

OFF

Login Theme

Client Protocol

openid-connect

Access Type

confidential

Standard Flow Enabled

ON

Implicit Flow Enabled

OFF

Direct Access Grants Enabled

ON

Service Accounts Enabled

OFF

OAuth 2.0 Device Authorization Grant Enabled

ON

OIDC CIBA Grant Enabled

OFF

Authorization Enabled

OFF

Front Channel Logout

OFF

Root URL

\$(authBaseUrl)

Valid Redirect URIs

/realms/REALM_RASPLUS_API/account/*

Base URL

/realms/REALM_RASPLUS_API/account/

Admin URL

Logo URL

Policy URL

Terms of service URL

Web Origins

/*

Backchannel Logout URL

Backchannel Logout Session Required

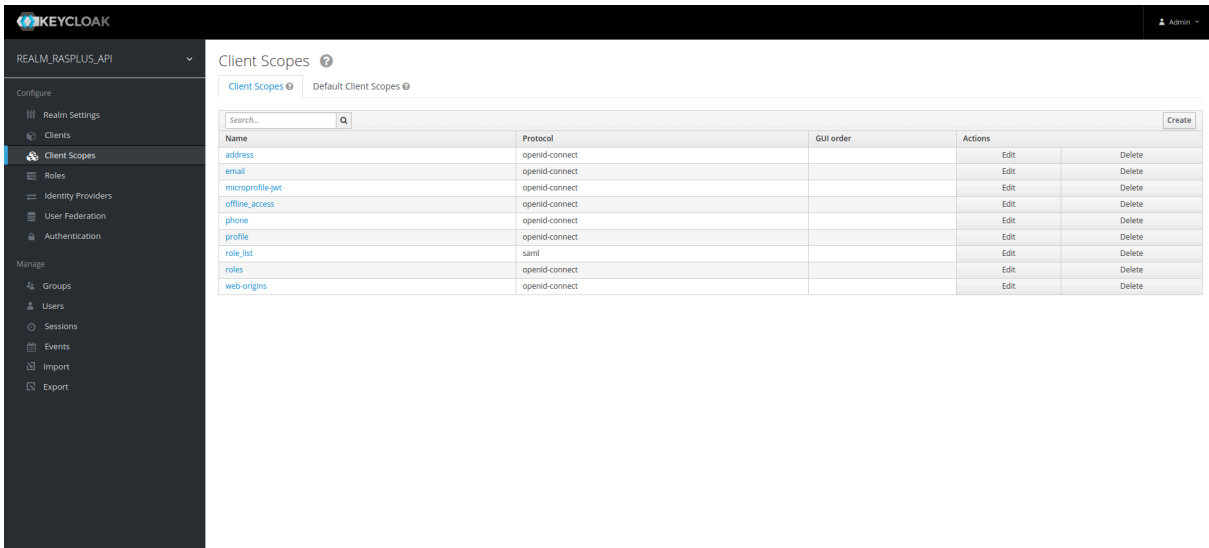
ON

Backchannel Logout Revoke Offline Sessions

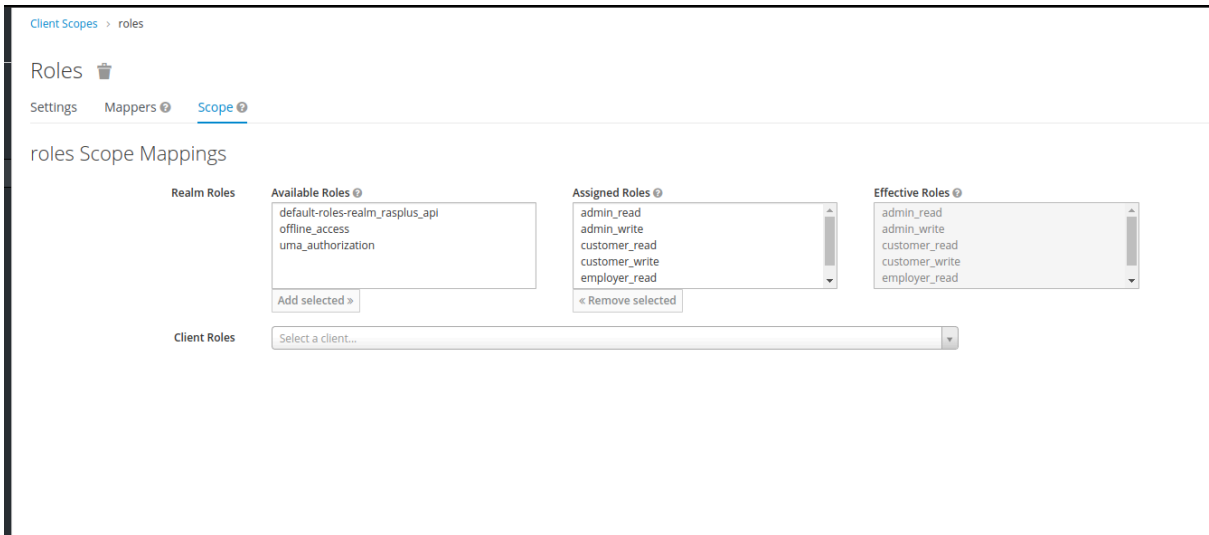
OFF

> Fine Grain OpenID Connect Configuration

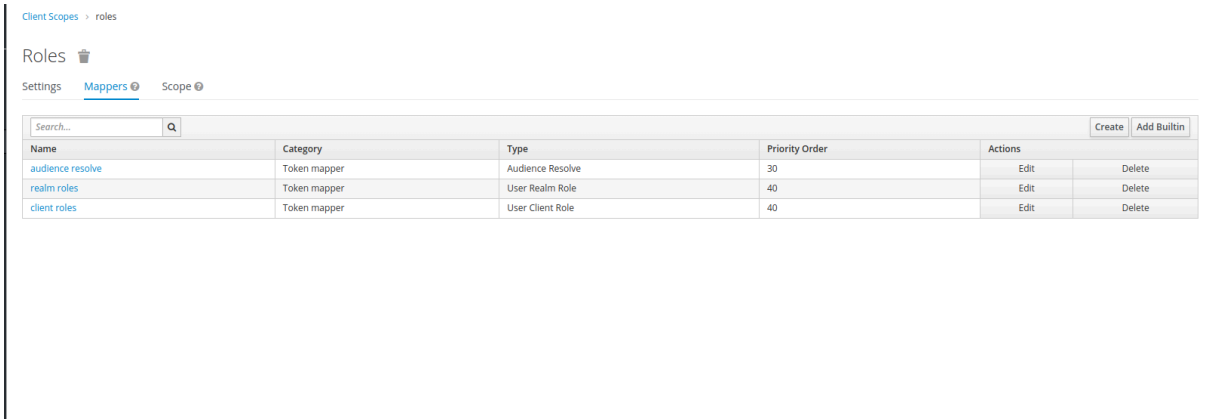
5 - Habilite o client scopes em “client scopes > roles”



Adicione os scopes criados



Vá em “mappers > clients roles”



Realize as configurações que serão enviadas.

Client Roles

Protocol ID: openid-connect

ID: ecbbd129-45dc-48f4-827b-2f7f66697bb6

Name: client roles

Mapper Type: User Client Role

Client ID: RASPLUS_API

Client Role prefix:

Multivalued: ☒

Token Claim Name: resource_access.\${client_id}.roles

Claim JSON Type: String

Add to ID token: ☒

Add to access token: ☒

Add to userinfo: ☒

Save Cancel

6 - Crie um usuário administrativo em “Users > add user”

KEYCLOAK

REALM_RASPLUS_API

Users > Add user

Add user

ID

Created At

Username *

Email

First Name

Last Name

User Enabled ☒

Email Verified ☒

Groups

Path	Actions
/admins	<button>Remove</button>

Required User Actions

Save Cancel

7 - Adicione as políticas de senha em “Authentication > Password Policy”

KEYCLOAK

REALM_RASPLUS_API

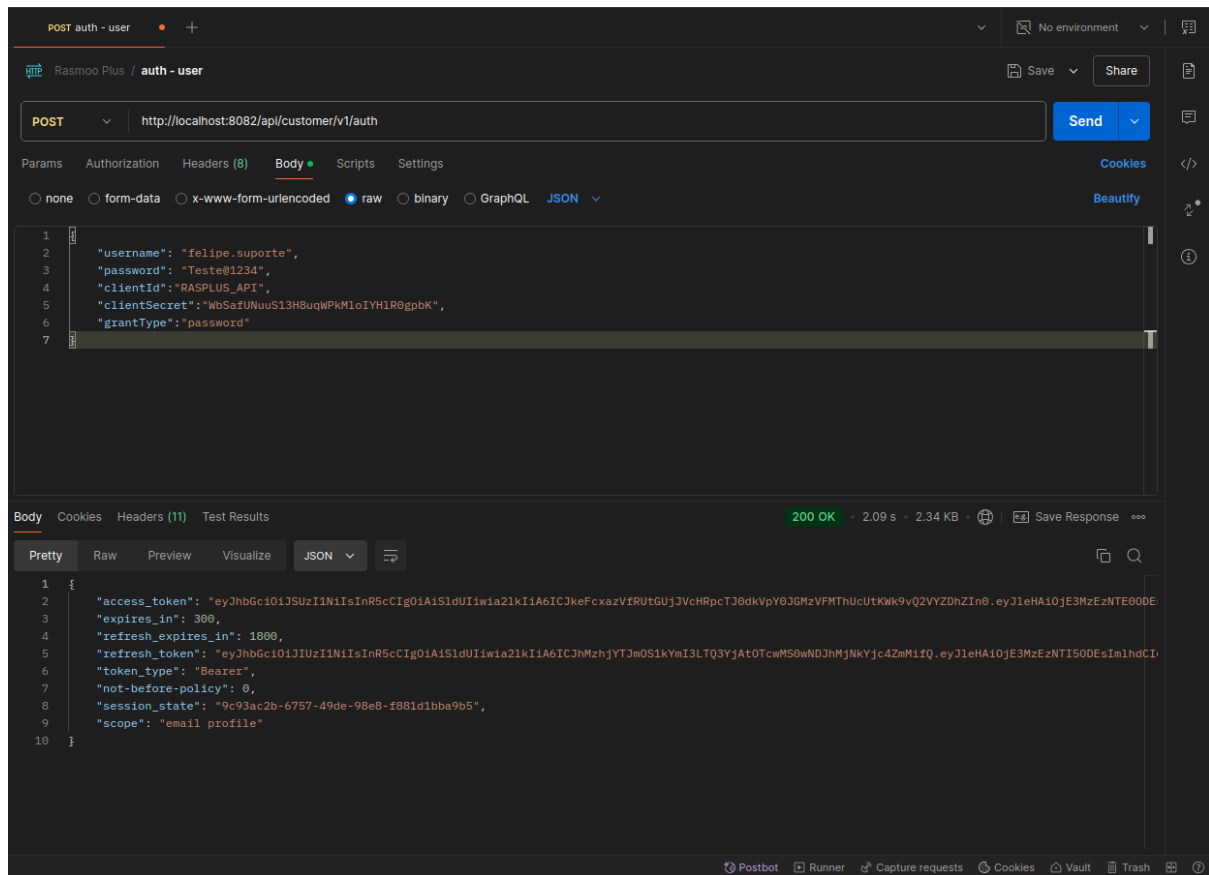
Authentication

Flows Bindings Required Actions **Password Policy** OTP Policy WebAuthn Policy WebAuthn Passwordless Policy CIBA Policy

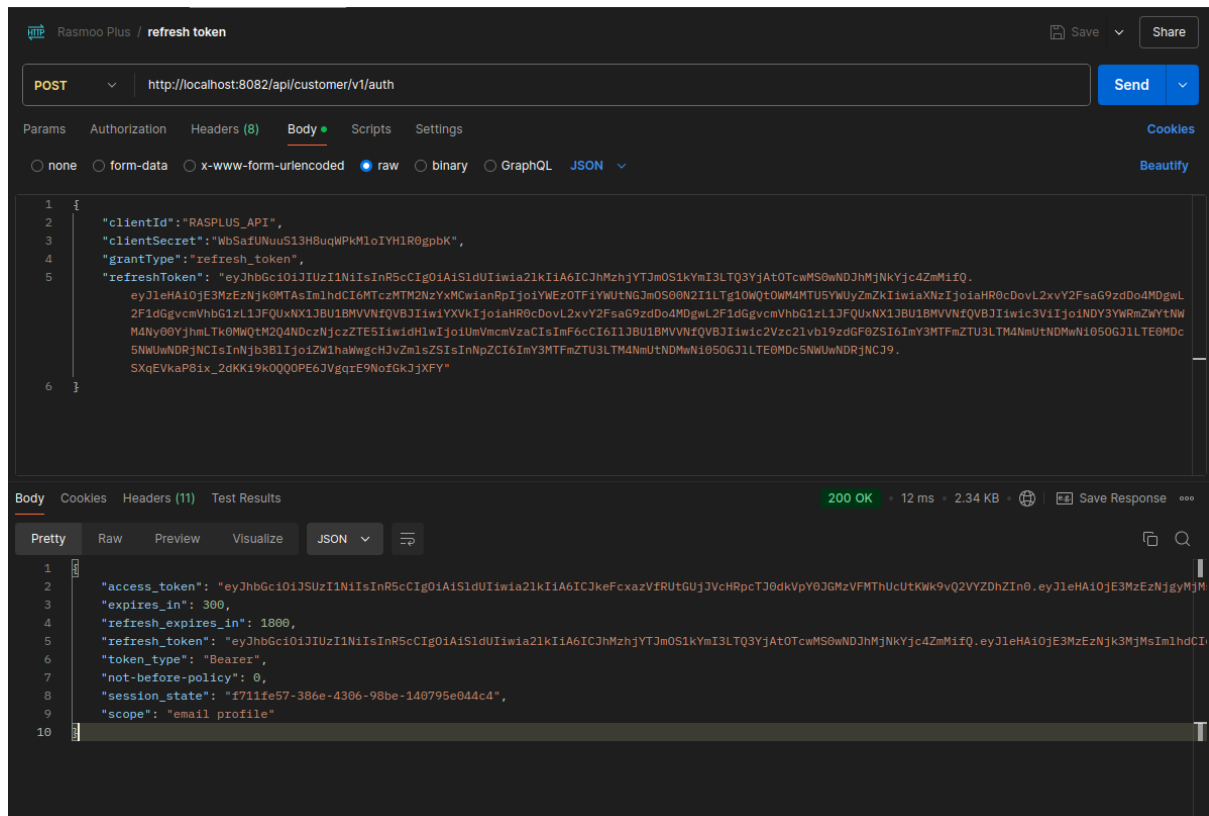
Policy Type	Policy Value	Actions
Expire Password	<input type="text" value="365"/>	<button>Delete</button>
Not Email	<input type="text"/>	<button>Delete</button>
Not Username	<input type="text"/>	<button>Delete</button>
Maximum Length	<input type="text" value="16"/>	<button>Delete</button>
Minimum Length	<input type="text" value="8"/>	<button>Delete</button>
Not Recently Used	<input type="text" value="3"/>	<button>Delete</button>
Digits	<input type="text" value="1"/>	<button>Delete</button>
Lowercase Characters	<input type="text" value="1"/>	<button>Delete</button>
Uppercase Characters	<input type="text" value="1"/>	<button>Delete</button>
Special Characters	<input type="text" value="1"/>	<button>Delete</button>

Save Cancel

8 - Teste a requisição de login



9 - Teste do refresh_token



10 - Teste o fluxo de client_credentials

Habilite o “Service account Enabled” em “clients > RASPLUS_API”

CLIENTS

RASPLUS_API

Settings

Credentials

Keys

Roles

Client Scopes

Mappers

Scope

Revocation

Sessions

Offline Access

Clustering

Installation

Service Account Roles

Client ID

RASPLUS_API

Name

\$(client_account)

Description

Enabled

ON

Always Display in Console

OFF

Consent Required

OFF

Login Theme

Client Protocol

openid-connect

Access Type

confidential

Standard Flow Enabled

ON

Implicit Flow Enabled

OFF

Direct Access Grants Enabled

ON

Service Accounts Enabled

ON

OAuth 2.0 Device Authorization Grant Enabled

ON

OIDC CIBA Grant Enabled

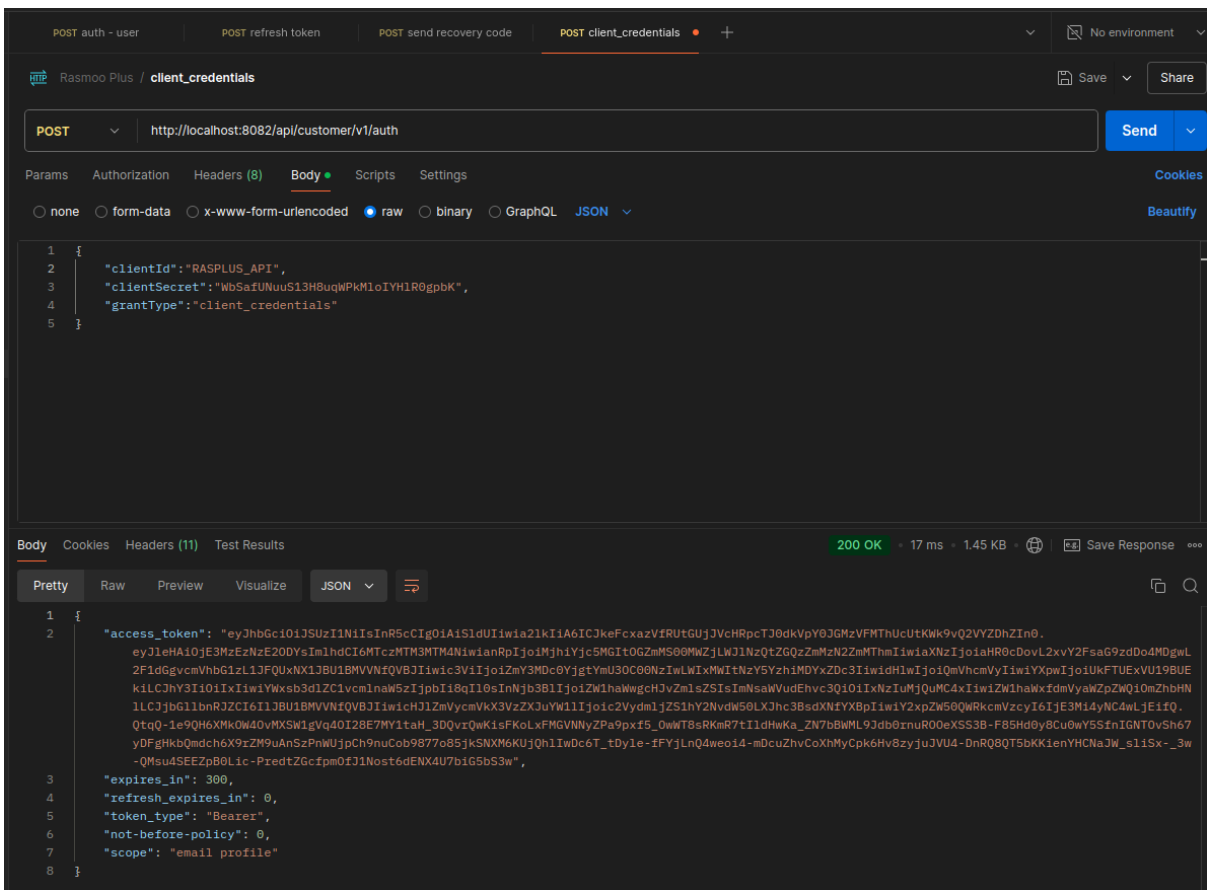
OFF

Authorization Enabled

OFF

Front Channel Logout

OFF



Crie a role client_read_write

*Não esqueça de habilitar o novo scope ao client scope.

Adicione o client_read_write ao service account

Clients > RASPLUS_API

RASPLUS_API

Settings Credentials Keys Roles Client Scopes Mappers Scope Revocation Sessions Offline Access Clustering Installation Service Account Roles

Service Account

Service Account User service-account-rasplus_api

Service Account Roles

Realm Roles

Available Roles

- admin_read
- admin_write
- customer_read
- customer_write
- employer_read

Add selected >

Assigned Roles

- client_read_write
- default-roles-realm_rasplus_api

<< Remove selected

Effective Roles

- client_read_write
- default-roles-realm_rasplus_api
- offline_access
- uma_authorization

Client Roles

Select a client...

11 - Habilite o admin-cli para criar usuários

Em “clients>admin-cli”

Clients > admin-cli

Admin-cli

Settings Credentials Keys Roles Client Scopes Mappers Scope Revocation Sessions Offline Access Clustering Installation Service Account Roles

Client ID admin-cli

Name \${client_admin-cli}

Description

Enabled ON

Always Display in Console OFF

Consent Required OFF

Login Theme

Client Protocol openid-connect

Access Type confidential

Standard Flow Enabled OFF

Implicit Flow Enabled OFF

Direct Access Grants Enabled ON

Service Accounts Enabled ON

OAuth 2.0 Device Authorization Grant Enabled ON

OIDC CIBA Grant Enabled OFF

Authorization Enabled OFF

Front Channel Logout OFF

Root URL

Base URL

Admin URL

Logo URL

Policy URL

Terms of service URL

Web Origins

Backchannel Logout URL

Backchannel Logout Session Required OFF

Backchannel Logout Revoke Offline Sessions OFF

> Fine Grain OpenID Connect Configuration

Copie o token JWT e tente a autenticação

Configuração do Swagger

Crie uma classe “configuration > SwaggerConfig” e adicione o seguinte código para habitar a autenticação por Bearer Authorization

Java

```
package com.client.api.cutomer.configuration;

import io.swagger.v3.oas.annotations.OpenAPIDefinition;
import io.swagger.v3.oas.annotations.enums.SecuritySchemeType;
import io.swagger.v3.oas.annotations.info.Contact;
import io.swagger.v3.oas.annotations.info.Info;
import io.swagger.v3.oas.annotations.info.License;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.security.SecurityScheme;
import org.springframework.context.annotation.Configuration;

@Configuration
@OpenAPIDefinition(
    info = @Info(
        title = "User API",
        version = "${api.version}",
        contact = @Contact(
            name = "Rasmoo", email = "midias@rasmoo.com", url =
"https://www.rasmoo.com"
        ),
        license = @License(
            name = "Apache 2.0", url =
"https://www.apache.org/licenses/LICENSE-2.0"
        ),
        description = "API para para atender ao frontend do RASM00
PLUS"
    ),
    security = {@SecurityRequirement(name = "Bearer Authentication")}
)
@SecurityScheme(
    name = "Bearer Authentication",
    type = SecuritySchemeType.HTTP,
    bearerFormat = "JWT",
    scheme = "bearer"
)
public class SwaggerConfig {
}
```