Doc. Arquitetura e design técnico

OP-03080 Desenvolvimento Plataforma GIS

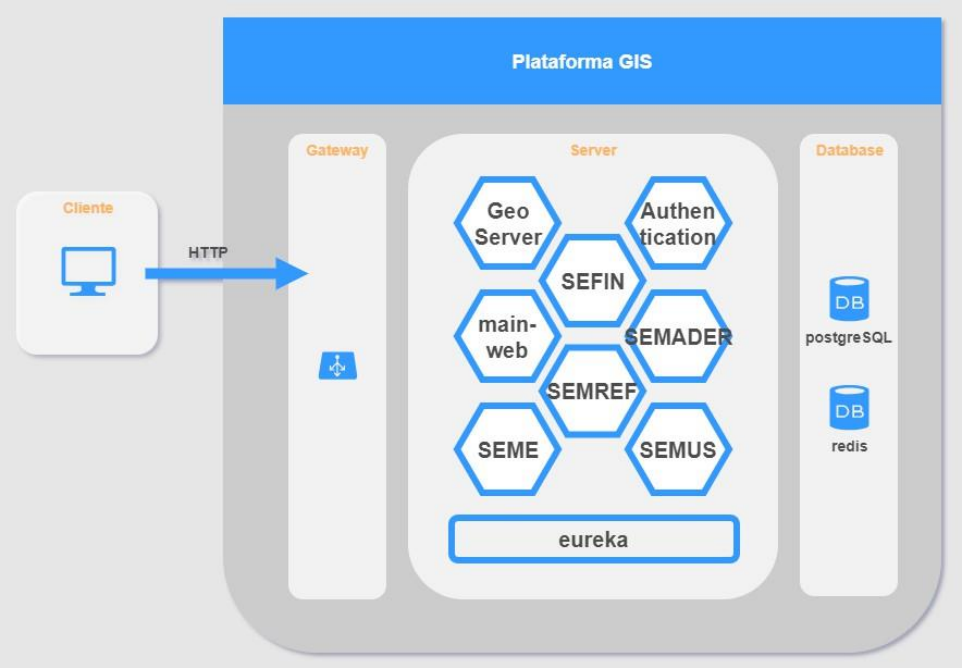
Sumário

1. Introdução

O projeto consiste em um Sistema de Informações Geográficas para o gerenciamento de informações que possam ser de interesse dos Municípios do Brasil.

2. Arquitetura e desenho técnico

2.1 Modelo de componentes



A aplicação consiste em um portal da web (web principal) que contém a interface do usuário.

No backoffice, temos uma arquitetura baseada nos microsserviços REST onde cada um desses microsserviços contém a implementação de cada um dos módulos funcionais.

Como é esperado que as instalações do aplicativo possam ter diferentes módulos funcionais ativos, essa arquitetura permite ter instalado e executando apenas os microsserviços necessários.

Assim, em sua publicação, podemos ter uma única URL base. Temos um gateway implementado com o spring. Portanto, teremos um único ponto de entrada para solicitações http.

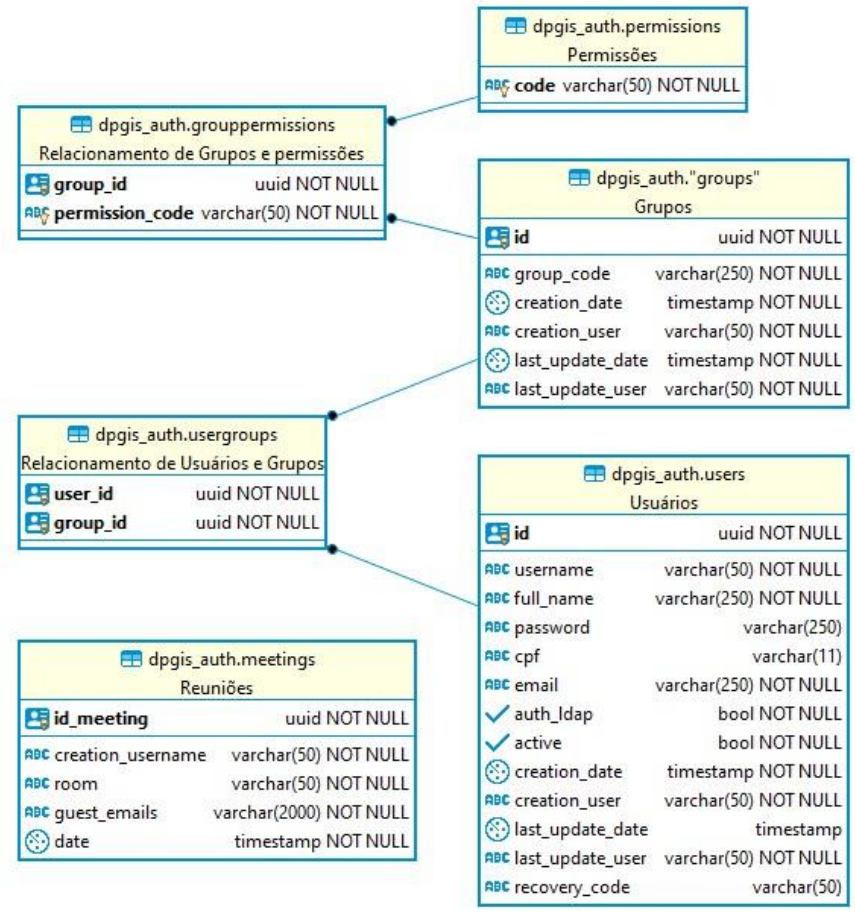
2.2 Modelo Físico

2.2.1. Base de dados

A estrutura do banco de dados é dividida em diferentes diagramas que correspondem aos módulos (e microsserviços) do aplicativo. Como a idéia é que, dependendo do ambiente em que está instalado, apenas os esquemas usados e configurados no referido ambiente são criados, cada um dos esquemas existentes até agora e seu diagrama de dados são descritos abaixo.

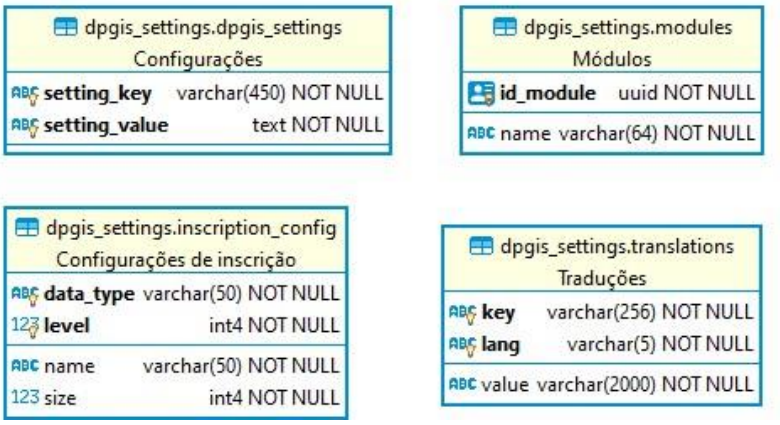
2.2.1.1. Dpgis_auth (Autenticação)

Neste esquema estão as tabelas para o gerenciamento de segurança no aplicativo. O modelo de dados é o seguinte:



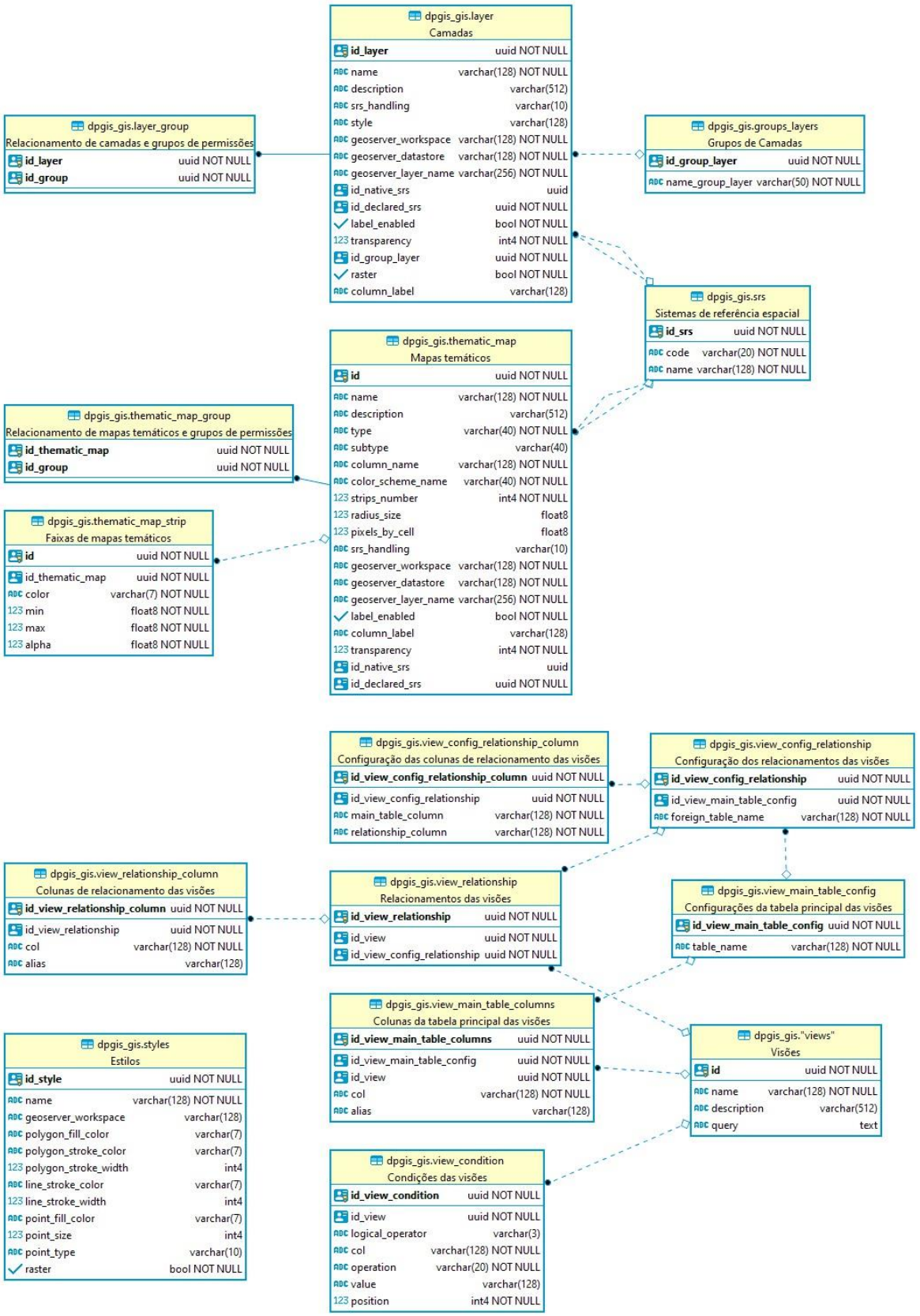
2.2.1.2. Dpgis_settings (Configurações)

Neste esquema, você encontrará as tabelas de configuração do aplicativo. O modelo de dados é o seguinte:



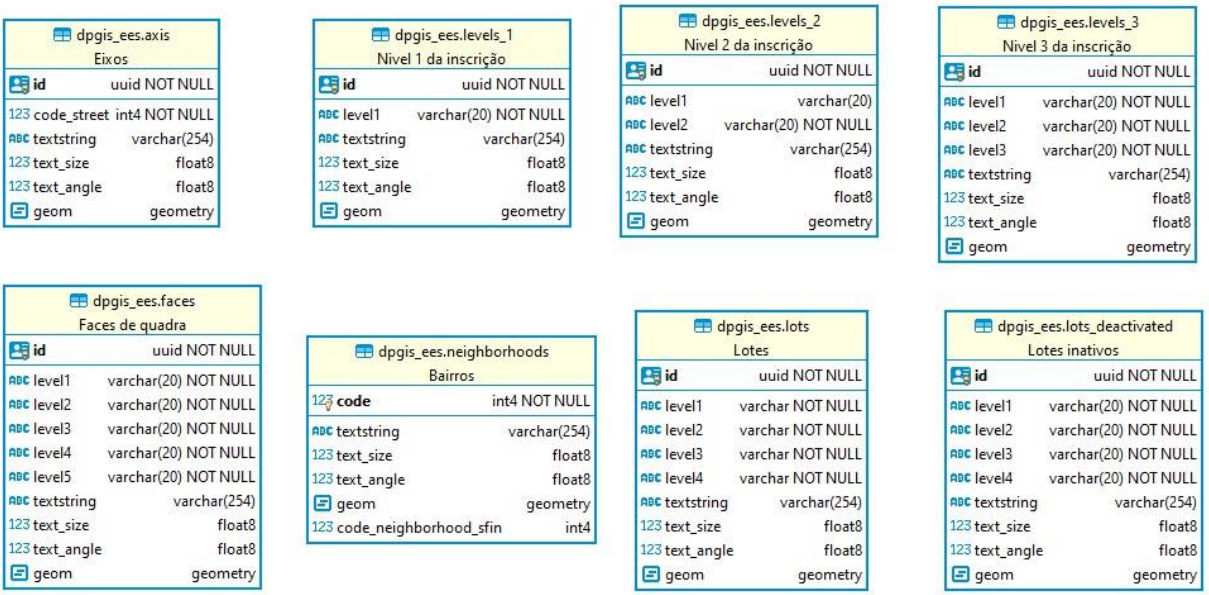
2.2.1.3. Dpgis_gis (informações Geográficas)

Nesse esquema, estão as tabelas para o gerenciamento de informações geográficas, como o gerenciamento de camadas, estilos etc. O modelo de dados é o seguinte:



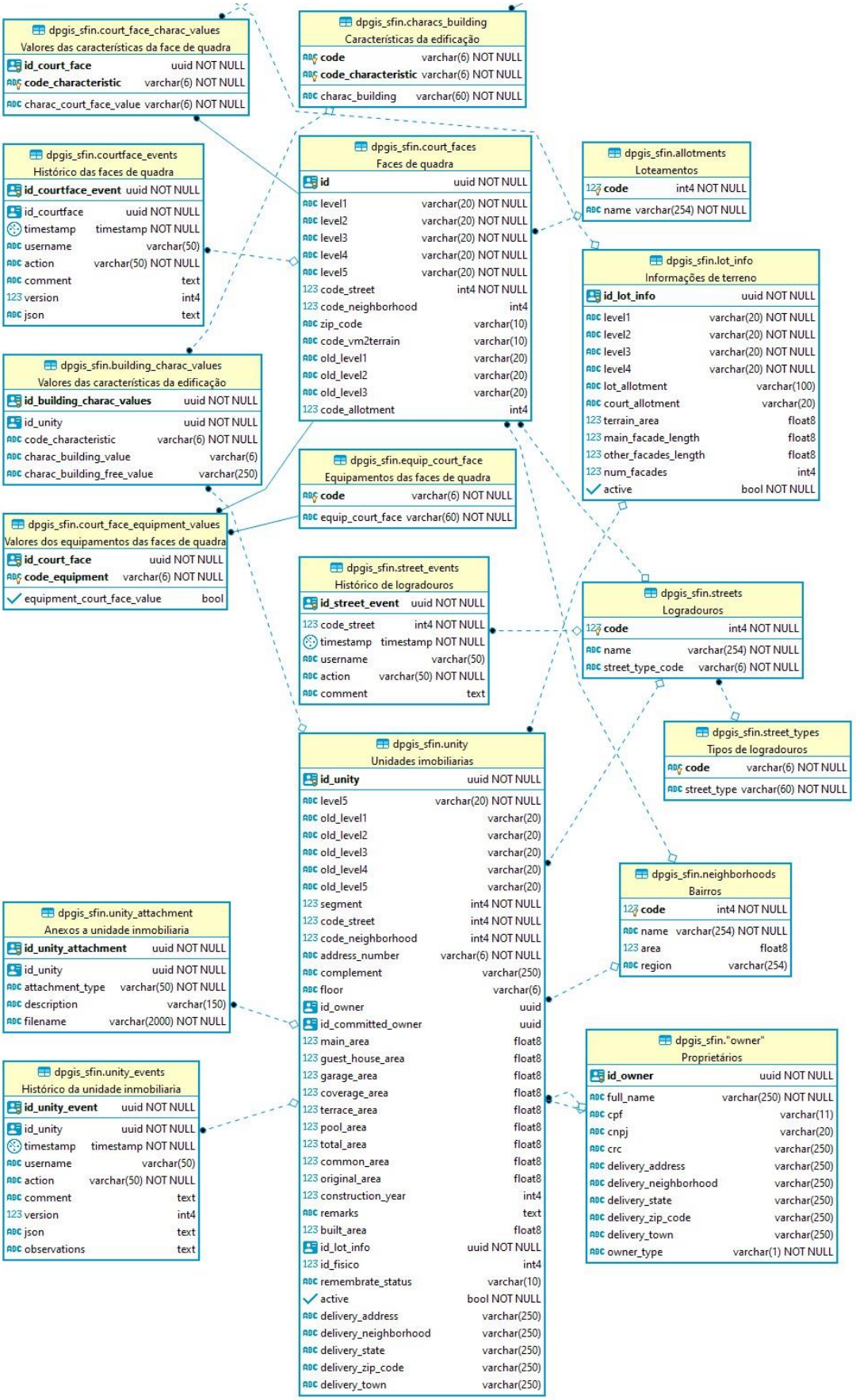
2.2.1.4. Dpgis_ees (Elementos Espaciais)

Neste esquema estão as tabelas que contêm os principais elementos espaciais do aplicativo. O modelo de dados é o seguinte:



2.2.1.5. Dpgis_sfin (Imobiliário)

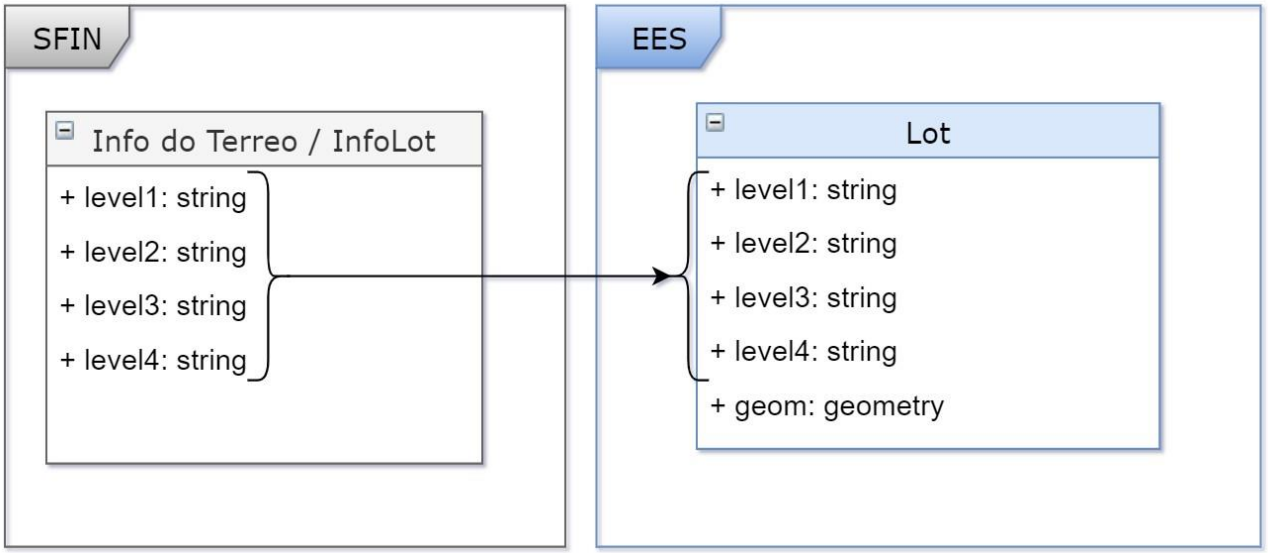
Neste esquema estão as tabelas que contêm os dados do módulo imobiliário. O modelo de dados é o seguinte:



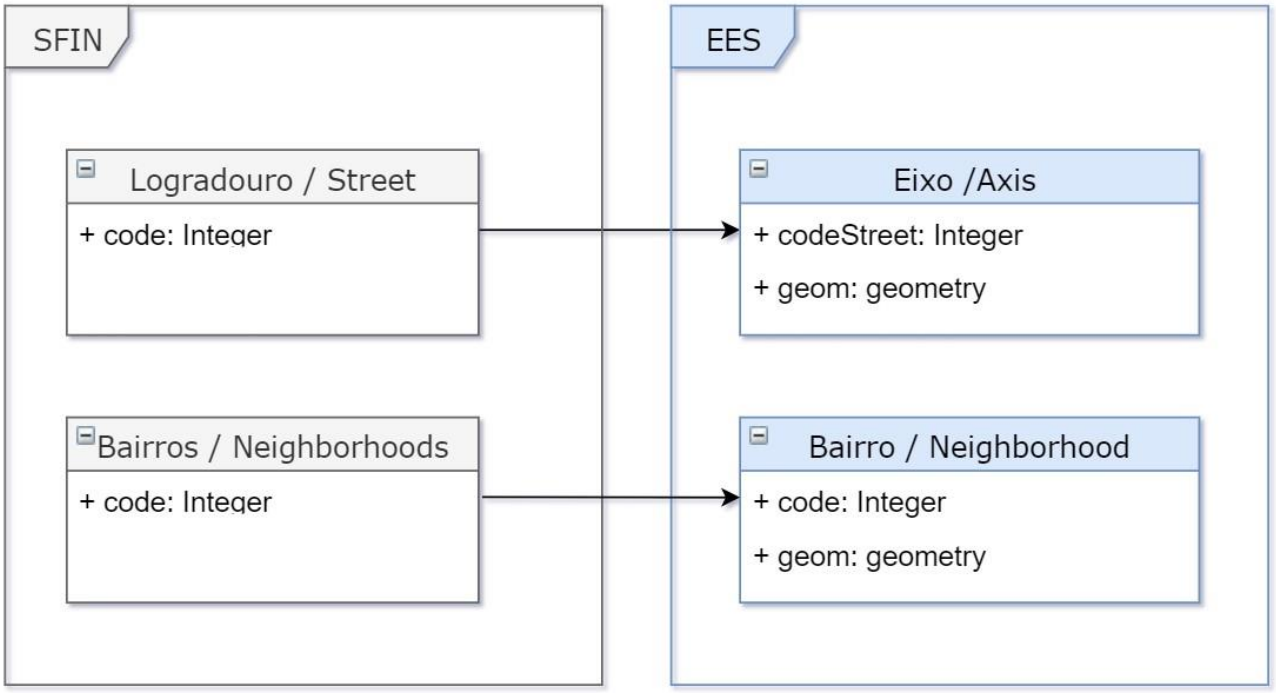
2.2.2. Relações Entidades – Módulos EES-SFIN

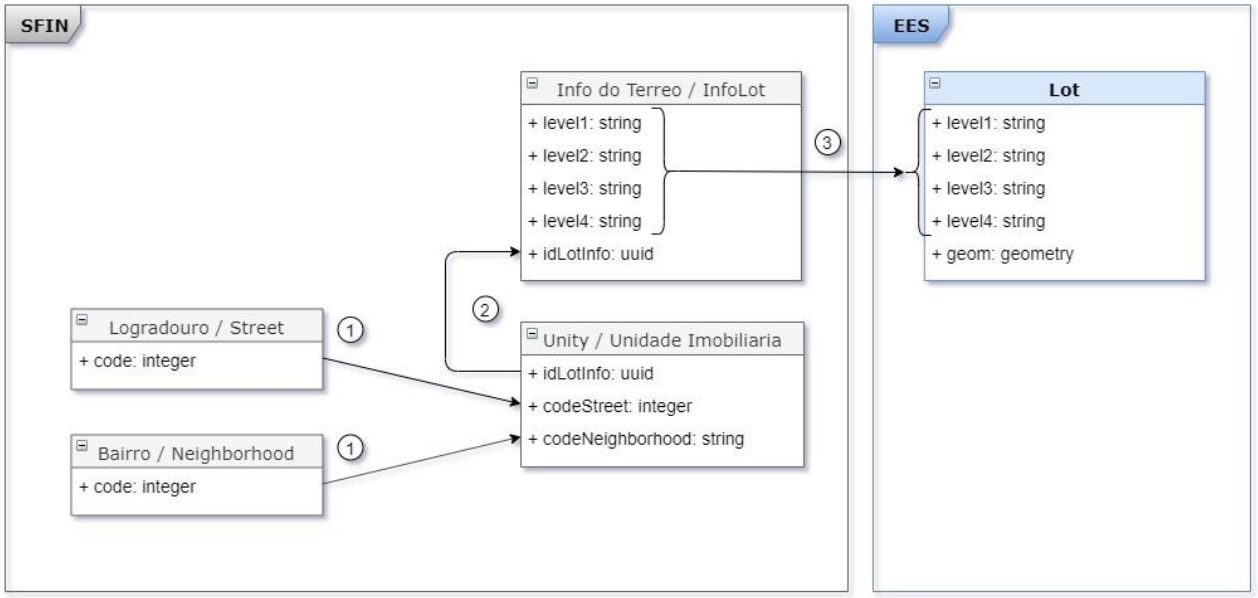
Para acessar os elementos espaciais (contendo o geom) associados a cada entidade do módulo Financeiro (SFIN), as entidades são relacionadas da seguinte maneira:

Info de Terreno:

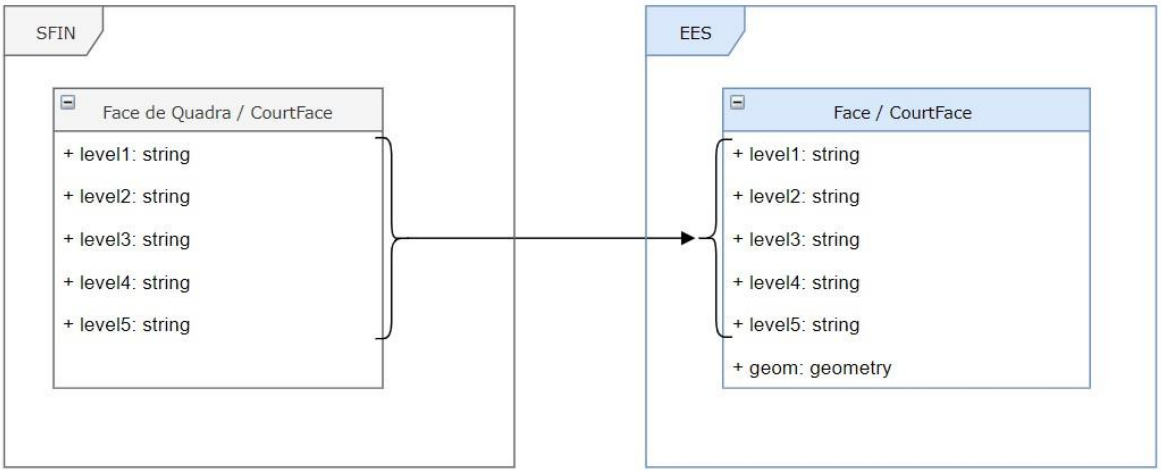


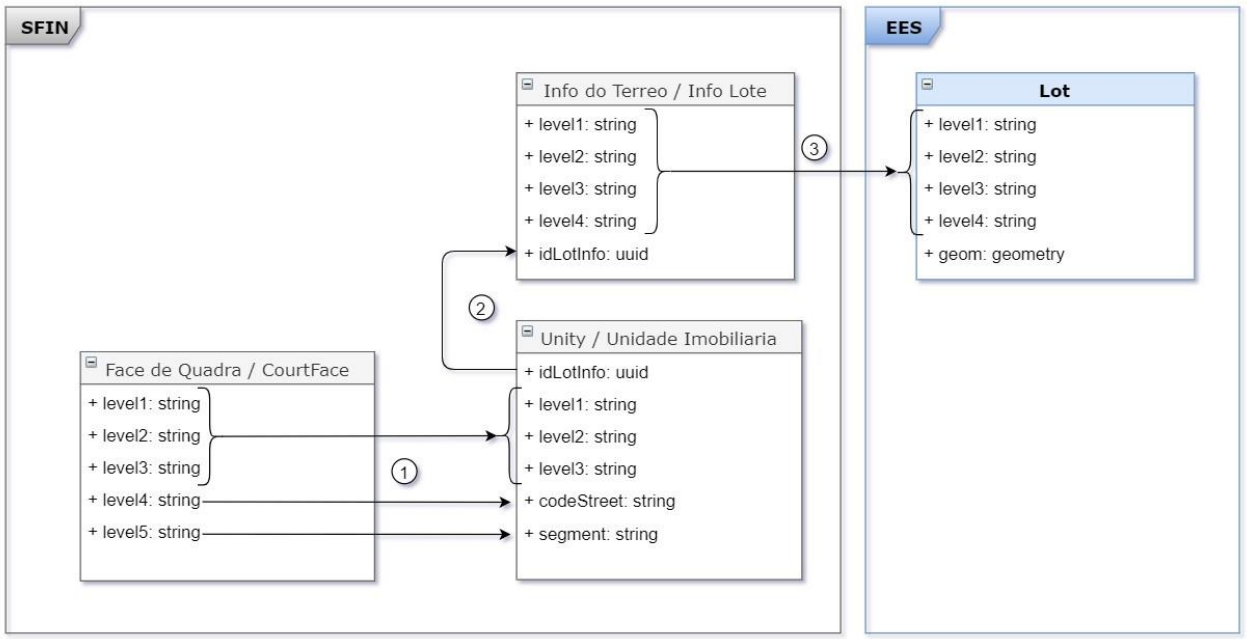
Bairros e Logradouro:



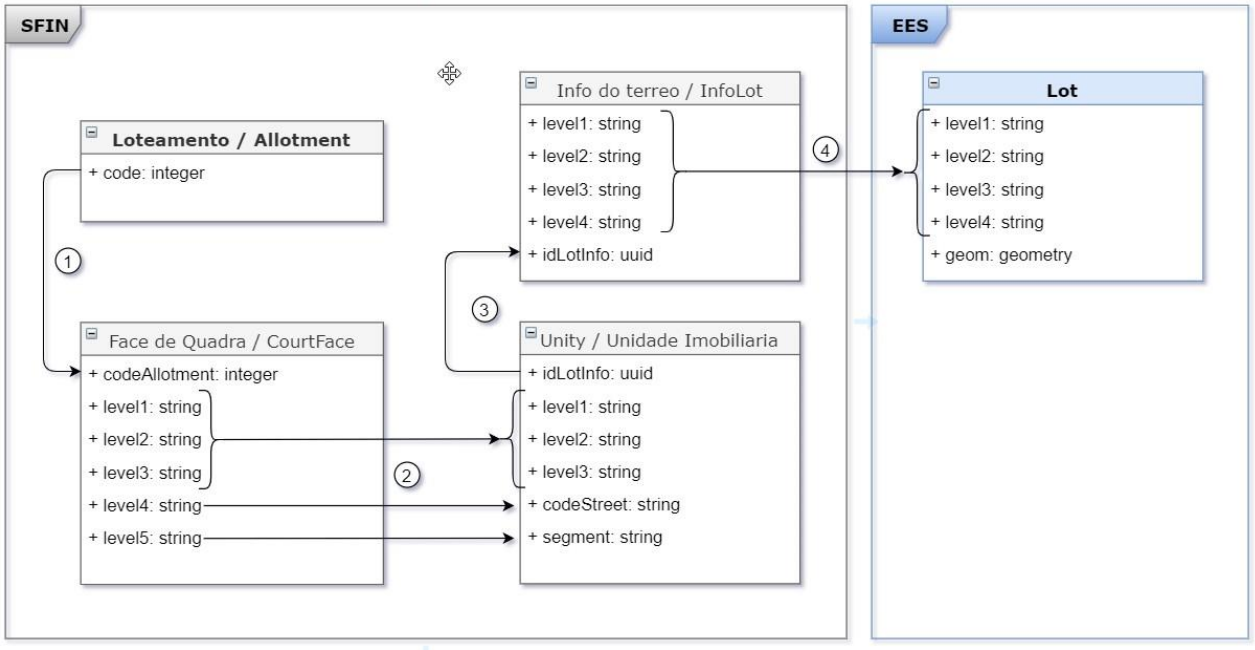


Face de Quadra:

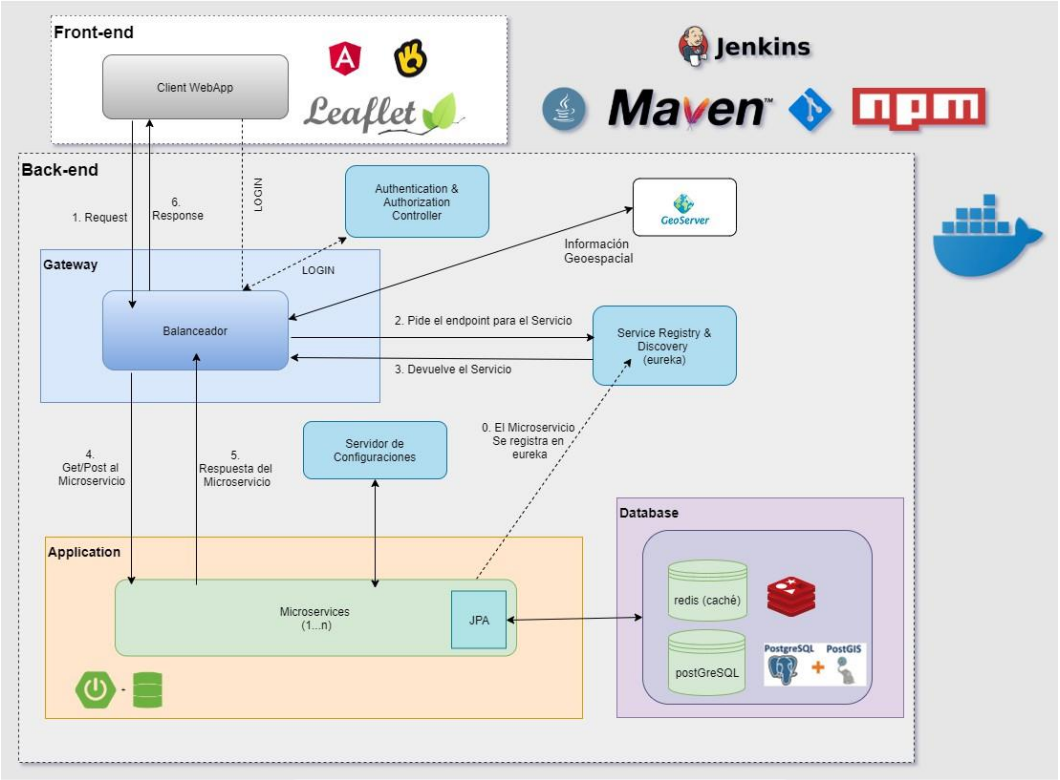



SFIN
EES
Info do Terreo / Info Lote
+ level1: string
+ level2: string
+ level3: string
+ level4: string
+ idLotInfo: uuid
Lot
+ level1: string
+ level2: string
+ level3: string
+ level4: string
+ geom: geometry
Unity / Unidade Imobiliaria
+ idLotInfo: uuid
+ level1: string
+ level2: string
+ level3: string
+ codeStreet: string
+ segment: string
Face de Quadra / CourtFace
+ level1: string
+ level2: string
+ level3: string
+ level4: string
+ level5: string
1
2
3

Loteamento:


SFIN
EES
Info do terreo / InfoLot
+ level1: string
+ level2: string
+ level3: string
+ level4: string
+ idLotInfo: uuid
Lot
+ level1: string
+ level2: string
+ level3: string
+ level4: string
+ geom: geometry
Loteamento / Allotment
+ code: integer
Face de Quadra / CourtFace
+ codeAllotment: integer
+ level1: string
+ level2: string
+ level3: string
+ level4: string
+ level5: string
Unity / Unidade Imobiliaria
+ idLotInfo: uuid
+ level1: string
+ level2: string
+ level3: string
+ codeStreet: string
+ segment: string
1
2
3
4

2.3 Entorno do Sistema



- A plataforma recebe HttpRequest do navegador da Web do Cliente E envia HttpResponse com o resultado do processamento da solicitação de entrada.
- **WebApp Cliente**:
  - **OntimizeWeb:** Plataforma de desenvolvimento rápido de aplicações web *(info: https://www.ontimize.com/web/)*
  - **Angular6 y typescript:** framework javascript para desenvolvimento. *(info: https://angular.io/ e https://www.typescriptlang.org/)*
  - **Leaflet** para gerenciar o módulo de mapas *(info: https://live.osgeo.org/es/overview/leaflet_overview.html)*
- **Spring Cloud Gateway:** API Spring MVC que fornece o caminho para rotear solicitações recebidas para a plataforma para os serviços de descanso implantados no servidor. *(info: http://spring.io/projects/spring-cloud- gateway)*
- **Server:**
  - **GeoServer**: Gerenciador de dados Espaciais (info: http://geoserver.org/)
  - **Microservico Autenticacão e Autorizacão**. Módulo que autoriza e autentica os usuários. (info: https://spring.io/projects/spring-security)
  - **Microservicos RESTful**. Usaremos Spring Boot e Spring Data JPA para implementação.(info: http://spring.io/projects/spring-boot e https://docs.spring.io/spring-data/jpa/docs/current/reference/html/)
  - **Database**: PostgreSQL + Postgis.
  - **Redis**: Base de dados em memória, para armazenar cache;
  - **Docker Swarm**: Gerenciador de serviços para sistemas multi plataformas. (info: https://docs.docker.com/ e https://docs.docker.com/engine/reference/commandline/swarm)

- **Database**: O banco de dados da plataforma é postgreSQL. Um sistema de banco de dados relacional orientado a objetos.
    - **dpgis**: Base de dados postgreSQl e postGIS com o esquema que conterá todas as informações da plataforma e seus módulos funcionais.
    - **Redis**: Banco de dados em memória, para armazenar em cache as informações do dpgis e disponibilizá-las muito mais rapidamente.
- **Desenvolvimento**
    - **Jenkins**: servidor de integração continua (info: https://jenkins.io/) **OPCIONAL**
    - **Java**: openjdk11 (info: https://openjdk.java.net/).
    - **Maven**: Gestor de construção de projetos (info: https://maven.apache.org/)
    - **GitBucket**: Repositório de código (info: https://gitbucket.github.io/). (Pode ser qualquer repositório)
    - **NPM**: Gestor de pacotes para node.js (más info: https://www.npmjs.com/)

2.4 Desenho Técnico do Sistema

2.4.1.3 Projeto gateway

O repositório dpgis-gateway abriga o código do projeto do tipo Maven para integrar o Registrador de Serviços de Gateway.

2.4.1.4 Projeto geoserver

O repositório dpgis-geoserver abriga o código do projeto do tipo Maven para integrar o GeoServer.

2.4.1.5 Projetos microservicios (ms)

2.4.1.5.1 Projetos Base

O código do projeto do tipo Maven para integrar o dpgis-base está alojado no repositório dpgis-base. É o projeto pai a partir do qual todos os projetos de microsserviços são posteriormente estendidos.

O código do projeto do tipo Maven está hospedado no repositório dpgis-base-ontimize-ee para integrar o dpgis-base-ontimize-ee. É o projeto pai que é posteriormente estendido por todos os projetos Ontimize EE Microservices.

2.4.1.5.2 Projetos archetypes

No repositório dpgis-archetype, o código do projeto do tipo Maven é hospedado para integrar o dpgis-archetype. É o projeto de arquétipo com o qual gerar projetos de microsserviços.
O código do projeto do tipo Maven está hospedado no repositório dpgis-archetype-ontimize-ee para integrar o dpgis-archetype-ontimize-ee. É o projeto de arquétipo com o qual gerar os projetos Ontimize EE Microservices.
As instruções de uso estão no arquivo README.md.

2.4.1.5.3 Projetos Microservicios y MS Ontimize EE:

- ***dpgis_auth***: É responsável por configurar a segurança, autenticação e autorização de todo o aplicativo. Inclui no módulo de administração o gerenciamento de usuários, grupos e permissões.
- ***dpgis_auth-ontimize-ee***: Lista de Usuarios e Grupos.
- ***dpgis_gis***: Módulo **CAMADAS (CA)**. Ele é responsável por gerenciar: Camadas, Vistas e Estilos.

  Módulo **ELEMENTOS ESPACIAIS (EES)**. Ele é responsável pela gestão de lotes, Eixos, Bairros, Faces, Nível1, Nível2 e Nível3. Além disso, é responsável pelas chamadas para o plug-in Geoserver para criar os diferentes elementos das camadas (Camadas, Áreas de trabalho, Estilos, Recursos, ...).
- ***dpgis_gis-ontimize-ee***: Camadas, vistas e listas de estilos.
- ***dpgis_settings***: Inclui no módulo de **Administração** a gestão das informações das prefeituras e as plainlhas. Além das traduções e configurações das inscrições das entidades do módulo SFIN.
- ***dpgis_sfin***: Módulo **SEFIN (SF)**. Ele é responsável pela administração de: Imóveis, Bairros, Loteamentos, Faces de Quadra, Logradouros.
- ***dpgis_sfin-oee***: Lista de imobiliários, Bairros, Loteamentos, Faces de Quadra, Logradouros.

2.4.1.6 Projeto main-web

O código do projeto do nó para integrar o webapp dpgis está alojado no repositório dpgis-main-web.

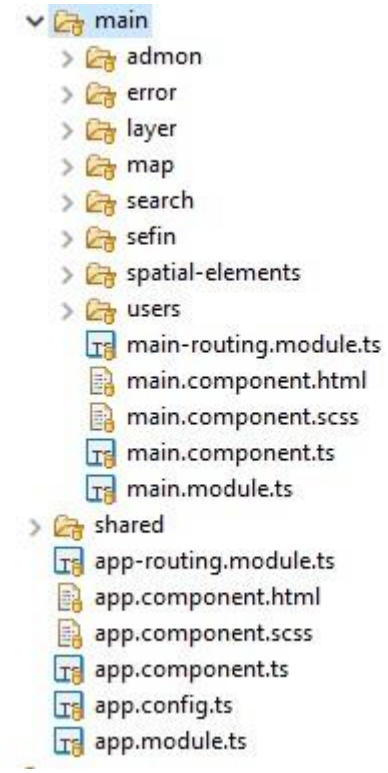
2.4.2 Estructura de una transacción común

A estrutura típica para cada módulo de transação / aplicativo é descrita abaixo:

2.4.2.1 Parte Front

- Na pasta **src/assets** estão os css, imagens e javascripts;
- Na pasta **src/app/shared**, temos os serviços comuns. Por exemplo, o serviço de guarda, o serviço da barra lateral do menu, o serviço de carregamento, o serviço de lanchonete, os validadores.
- Na pasta **src/app/dpgis-listrenderer** está o componente com o qual você constroem as listas de ações a serem executadas com cada registro nas listagens.
- Na pasta **src/app/dpgis-loafing** está o componente que exibe um controle giratório de progresso quando uma solicitação é enviada pela frente e aguarda uma resposta pela parte traseira.
- Na pasta **src/app/dpgis-tablecellrendereraction** está o componente para poder executar ações das células da tabela.

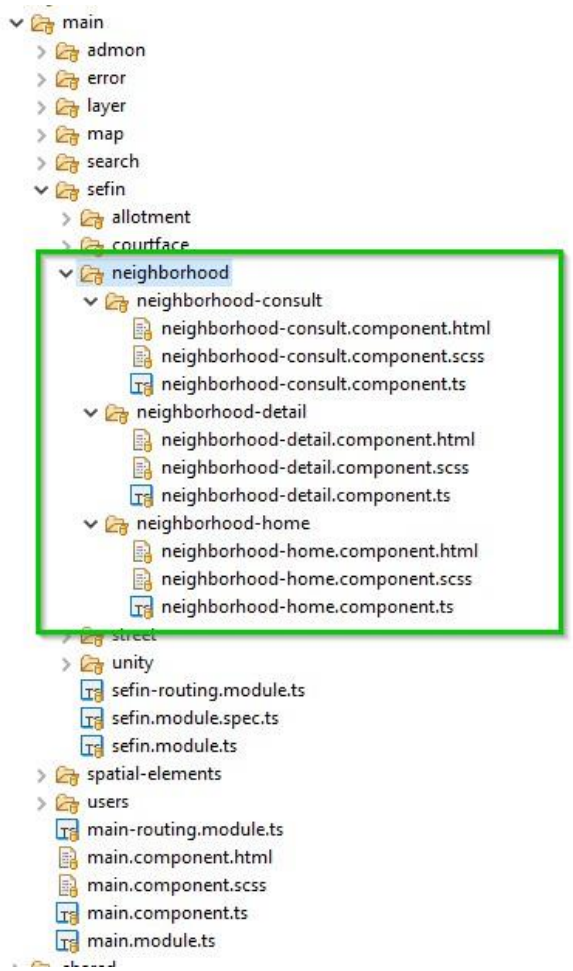
- Na pasta **src/app/login** está o componente de login
- Na pasta src / app / main, você encontrará todos os módulos principais e todas as suas operações:
  - admon (Administração);
  - camada (Camadas, Vistas, Estilos)
  - mapa (mapas)
  - pesquisa (unidades de pesquisa rápida)
  - sefin (Imóveis, Bairros, Loteamentos, Faces Quadradas, Logradouros)
  - elementos espaciais (EES): lotes, eixos, vizinhanças, faces, nível 1, nível 2 e nivel3;



A transação típica em cada módulo contém:

- Um HomeComponent - para mostrar a lista de entradas no módulo. A listagem faz solicitações ao microsserviço ontimize ee associado à entidade. Você deve ter criado uma exibição no banco de dados, com todas as colunas da entidade ou com as relacionadas que queremos incluir para mostrar, classificar e filtrar mais tarde.
- Um ConsultComponent - Depois que um elemento é selecionado na lista, com permissões de consulta, um formulário é exibido com todos os campos da entidade sem poder modificá-los. Após o acesso, é feita uma solicitação ao microsserviço java que cria as informações completas da entidade e as envia.
- Um DetailComponent - Ao acessar com permissões de gravação, você pode criar novas entidades ou modificar as existentes. A partir da lista, um novo é criado ou um existente é acessado. E depois que as informações são validadas, elas são enviadas ao microservice java para atualizar as tabelas.

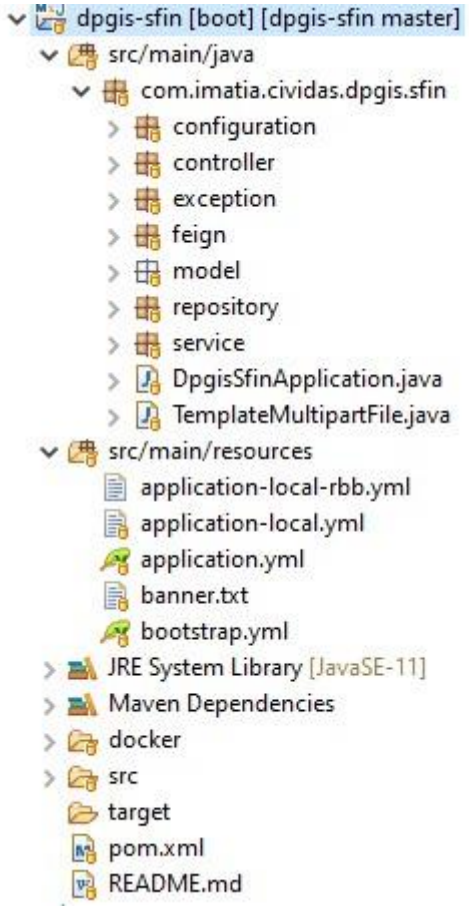
Por exemplo, a opção Bairro (Bairro) do módulo SEFIN está estruturada da seguinte maneira:



2.4.2.2 Parte Back

2.4.2.2.1 Microservicio Java Spring Boot

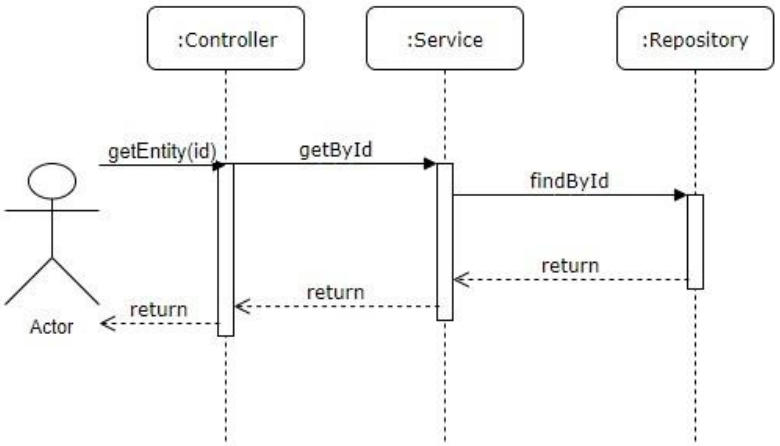
Tomando como exemplo a estrutura de dpgis-sfin:



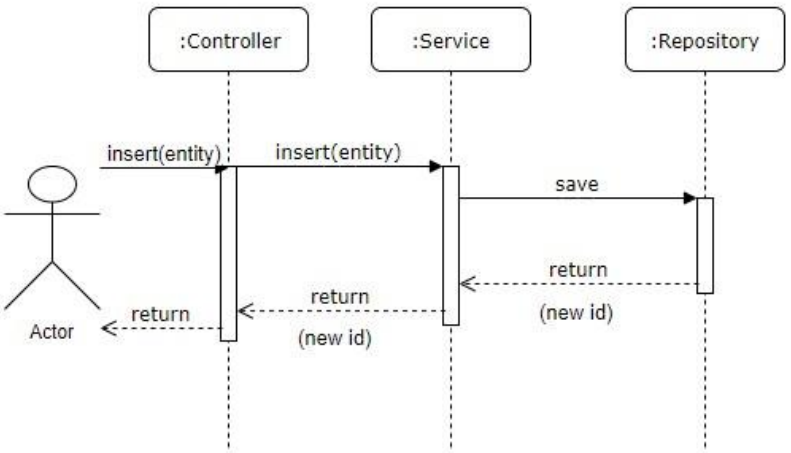
- Resources: Nesta pasta estão os arquivos yml de configuração. O application-local - *. Yml são configurações locais.
- Java
  - Configuração: as classes @Configuration. Todos os microsserviços têm o SwaggerConfiguration para criar acesso aos métodos disponíveis nas classes @Controller.
- Controller: as classes @Controller. São as classes java que implementam a recepção, tratamento e resposta das solicitações da parte frontal.
- Feign: as classes @FeignClient que permitem chamadas entre microsserviços.
- Model: os grãos usados no microsserviço. Tanto o @Entity, como os serializáveis
- Repositório: classes Java anotadas com @Repository. Eles se estendem da classe PagingAndSortingRepository ou CrudRepository.
- Service: as interfaces e a implementação de cada uma delas. Cada serviço tem um atributo privado injetado no Repositório; e é através dessa classe com a qual ela acessa a tabela correspondente no banco de dados.

Diagramas de sequência de operações comuns:

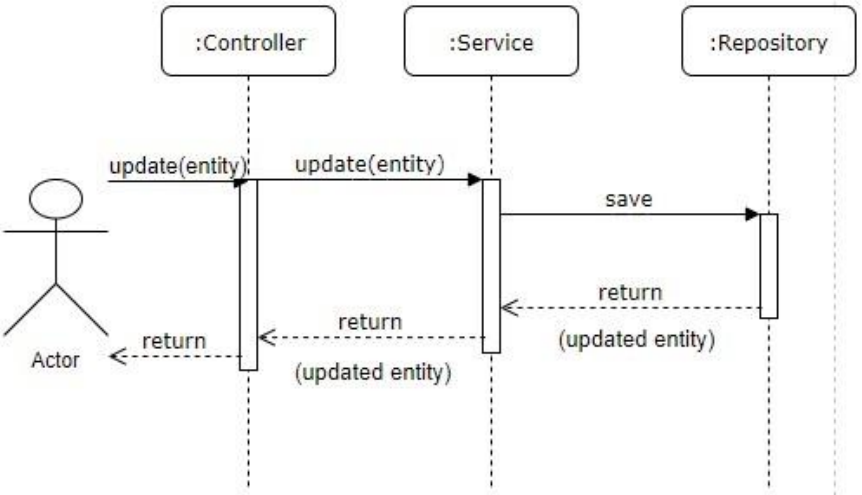
- findById: Request de tipo GET. Ele recebe o identificador da entidade.
  Retorna a entidade encontrada. Se não conseguir encontrá-lo, retornará 400 (não encontrado).



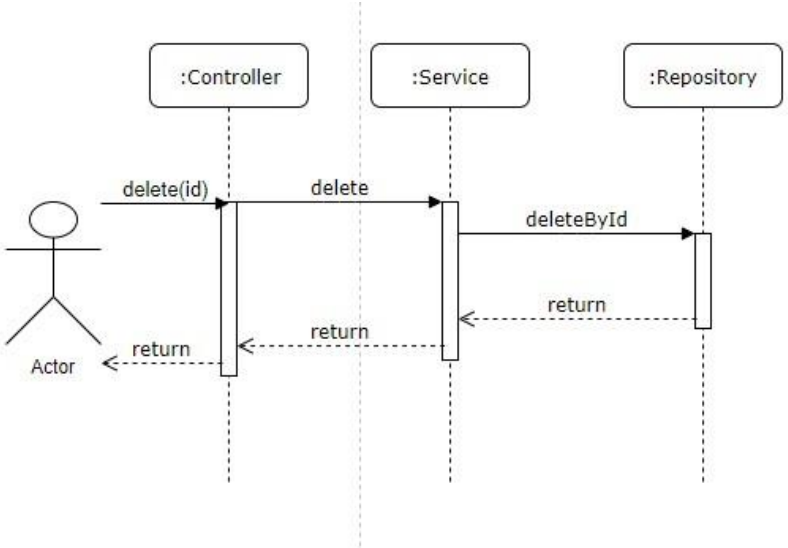
- insert: Request de tipo POST. Ele recebe no RequestBody a instância da entidade a ser inserida.



- update: Request de tipo PUT. Ele recebe no RequestBody a instância da entidade a ser modificada.

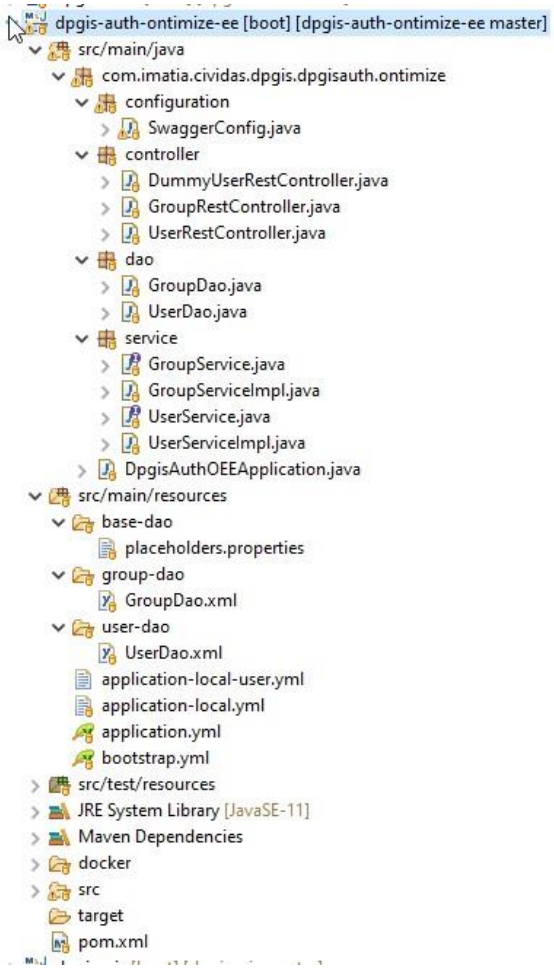


- delete: Request de tipo DELETE. Ele recebe a identificação da entidade a ser removida.



2.4.2.2.2 Microservicio Java Ontimize EE

Tomando como exemplo a estrutura do dpgis-auth-ontimize-ee:



- Java:
  - Controller: classes Java com a anotação @RestController, responsável por receber, processar e responder a solicitações da frente.
  - Dao: classes Java com a anotação @Repository configurada com xmls hospedados na pasta Resources. Eles estendem a classe OntimizeJdbcratelySupport e manipulam o acesso às tabelas do banco de dados.
  - Service: classes Java anotadas com @Service e que implementam a interface correspondente à entidade. Eles têm a classe DefaultOntimizejiangHelper injetada; e eles são responsáveis por iniciar a consulta correspondente da classe dao na tabela relacionada. O DPGIS permite apenas chamadas para o método allotmentPaginationQuery AdvancedEntityResult.
- Resources:
  - o placeholder.properties com o qual o Dao é configurado
  - o XXXX-dao: o arquivo de configuração com as informações de dao configuradas na classe dao específica.

2.5 Integrações com outros sistemas o serviços

2.5.1 GeoServer

2.5.1.1 Integração

Foi criado o controlador (GeoserverController), que atua como um proxy entre a visualização e o servidor geográfico, permitindo gerenciar a segurança e fornecer métodos de conveniência para obter dados do servidor geográfico.
Para fazer solicitações à API Geoserver Rest, a biblioteca Geoserver foi usada Gerente. Há casos em que a biblioteca não possui as chamadas implementadas; nesse caso, elas fazem um RestTemplateExchange.
Para acessar as APIs WFS e WMS, foram criados métodos que simplesmente fazem proxy do próprio Geoserver por meio de um RestTemplate Exchange.

2.5.1.2 Configuração Features

Ao atender às características de uma camada, o GeoServer executa um arredondamento de
casas decimais de geometria. Para evitar isso, ele deve ser configurado em "Configurações> Global > Número de casas decimais ", uma quantidade suficiente para que as geometrias estejam em conformidade com as processadas por telhas.

2.5.1.3 Projeções

O GeoServer é capaz de detectar a projeção (SRS) de uma camada se a fonte de dados tiver esses metadados. Nos arquivos carregados pela SHP no banco de dados PostGis, esses dados não aparecem. É desejável que esses dados estejam acessíveis e, se possível, usem uma projeção recomendada para uso com mapas da web (EPSG: 4326).
Tomando como exemplo uma camada com projeção 32723, para reprojetar, você deve executar a consulta:

```
UPDATE test.layer SET geom =
st_transform(st_geomfromtext(st_astext(geom),32723), 4326);
```

2.5.1.4 Publicação de camadas

Para a publicação de camadas, é necessário que a tabela exista em um banco de dados em um dos repositórios de dados configurados no Geoserver. Ao criar a camada na transação da camada DPGIS, é feita uma solicitação usando o método it.geosolutions.geoserver.rest.GeoServerRESTPublisher.publishDBLayer (String, String, GSFeatureTypeEncoder, GSLayerEncoder) para publicar a camada. Além disso, se a camada tiver o rótulo ativado. Um estilo específico é criado para exibir essa tag, na área de trabalho denominada "dpgis-hidden", usando o método it.geosolutions.geoserver.rest.GeoServerRESTPublisher.publishStyleInWorkspace (String, String, String).

2.5.1.5 Criação de estilos

Para gerar estilos a partir da transação de estilo DPGIS. Um arquivo SLD é montado a partir dos campos preenchidos pelo usuário. Para isso, contamos com a biblioteca Geotools. Depois que o SLD é gerado, publicamos usando o método it.geosolutions.geoserver.rest.GeoServerRESTPublisher.publishStyleInWorkspace (String, String, String).

2.5.2 Gotenberg

Na pilha de aplicativos, um serviço com a imagem da janela de encaixe Gotenberg é iniciado para transformar os documentos ODT gerados a partir de modelos em formato PDF. URL nos microsserviços, ele é configurado na propriedade "dpgis.pdfConverterUrl".

2.6 Segurança e controle de acesso

2.6.1 Segurança no Backend: Spring Security

Para proteger o acesso às solicitações de retorno, usamos o Spring Security (https://docs.spring.io/spring-security/site/docs/3.1.x/reference/springsecurity-single.html) e o Spring LDAP (https://spring.io/projects/spring-ldap)

2.6.1.1 Configuração de microservicio dpgis_auth

Em auth.yml (pasta *.../dpgis/config-server/*) se pode configurar :

1 - O cliente de acesso oauth
2 - O URL de acesso para redefinir a senha e o aplicativo de reuniões.
3 - A conta, o servidor e o protocolo para envio de e-mails de dpgis.
4 - O servidor, a porta e o domínio do acesso LDAP.
5 - A apiKey para acessar o googlemaps.

```
security:
  oauth2:
    client:
      client-id: [redacted]
      client-secret: [redacted]
      authorized-grant-types: password,authorization_code,refresh_code,implicit
      scope: openid

dpgis:
  oauth:
    enabled: true
  auth.meeting:
    providerUrl: https://meet.jit.si/
    reminderMinutesBeforeMeeting: 15
  auth:
    password-reset-url: http://[redacted]/login/password-reset

spring:
  mail:
    host: smtp.gmail.com
    port: 587
    username: [redacted]@gmail.com
    password: [redacted]
    protocol: smtp
    properties:
      mail.smtp.auth: true
      mail.smtp.starttls.enable: true

ldap:
  active: true
  host: [redacted]
  port: 3268
  domain: [redacted]
  base.dn: DC=[redacted],DC=[redacted]
  search.filtre: (&(objectClass=person)(sAMAccountName=%s))
```

```
google.maps:
  apiKey: [redacted]
  libraries: places
```

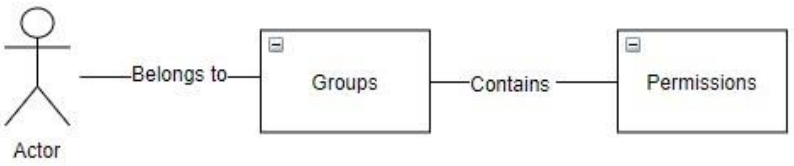
2.6.1.2 Configuração dos RestController dos microservicos.

Em cada classe marcada com a anotação @RestController, os métodos públicos têm a anotação @PreAuthorize associada a eles. Nesta anotação, definimos as permissões autorizadas para acessar a transação.

Por exemplo, no UnityController do microsserviço dpgis-sfin, a solicitação POST (associada à inserção da unidade imobiliária) permite apenas o acesso às permissões ADMIN e SEFIN_CRUD desta maneira:

```
@PreAuthorize("hasAnyAuthority('ADMIN', 'SEFIN_CRUD')")
public ... OAuth2Authentication principal) throws IOException {
    return unityService.insertUnity(unity, principal.getName());
}
```

2.6.1.3 Criação de usuarios, permisões e grupos



No DPGIS, podemos criar os usuários que terão acesso às transações da parte traseira e às opções de menu, mapas e transações também da parte frontal.
Podemos associar um ou mais grupos a cada usuário. Esses grupos também são eles criarão e configurarão a partir do próprio aplicativo. Os grupos são aqueles que contêm uma lista das Permissões que finalmente serão associadas aos usuários conectados ao aplicativo.

2.6.1.3.1 Autenticação de usuarios por LDAP

Quando um usuário é criado, ele pode ser configurado para autenticar em um servidor LDAP (que será configurado no arquivo yml de dpgis_auth). Nesse caso, quando o formulário de login for enviado, será verificado que o usuário existe nas tabelas DPGIS, mas sua senha será validada no servidor em vez da senha armazenada no banco de dados.

2.6.2 Segurança no Front

2.6.2.1 Segurança no Front: Chamadas com token

Cada solicitação enviada de frente contra os microsserviços deve ter o token gerado inserido ao fazer a chamada de autorização contra o ms dpgis_auth (dpgis_auth: porta / auth / oauth / token).

No auth-token.interceptor.ts, implementamos o método HttpInterceptor interceptar (solicitação: HttpRequest <any>, próximo: HttpHandler): Observável <...> para inserir no cabeçalho de cada solicitação na parte traseira a Autorização: 'Portador' e o token recebido da chamada de microsserviço dpgis-auth .

2.6.2.2 Segurança no Front: Menu DPGIS

O serviço dpgis-menu-sidebar.service.ts cria a barra lateral exibida quando o usuário é autenticado e autorizado. Cada módulo DPGIS tem permissões associadas. Além disso, dentro de cada módulo, existem operações que também podem ser executadas apenas se o usuário tiver outras permissões específicas definidas para essa operação. Por exemplo, um usuário com permissões SFIN_READ tem permissão para acessar o módulo SFIN. Mas depois de acessá-lo, você não tem permissão para criar entidades, modificá-las ou excluí-las.

Você também pode definir permissões de acesso e modificação associadas às camadas criadas no módulo CAPAS.

2.6.2.3 Segurança no Front: CanActivate

No dpgis-guard.service.ts, implementamos o método CanActivate canActivate (route: ActivatedRouteSnapshot): booleano para permitir o acesso ou não à lista de permissões que extraímos do usuário autenticado.

2.6.2.3.1 CanActivate para definir Routes

Ao definir o roteamento de cada módulo, usamos o componente DpgisGuardService para criar as rotas, injetando a matriz de permissões que têm permissão para acessar a rota.
Por exemplo: Em sefin-routing.module.ts para configurar o acesso ao Bairros / Bairros é implementado assim:

```
20
21  const routes: Routes = [
22    {
23        path: 'neighborhoods',
24        component: NeighborhoodHomeComponent,
25        canActivate: [DpgisGuardService],
26        data: {
27            expectedPermissions: ['ADMIN', 'SEFIN_CRUD', 'SEFIN_READ']
28        }
29    },
30    {
31       path: 'neighborhoods/neighborhood-detail/:id',
32       component: NeighborhoodDetailComponent,
33       canActivate: [DpgisGuardService],
34       data: {
35           expectedPermissions: ['ADMIN', 'SEFIN_CRUD']
36       }
37    },
38    {
39        path: 'neighborhoods/neighborhood-consult/:id',
40        component: NeighborhoodConsultComponent,
41        canActivate: [DpgisGuardService],
42        data: {
43            expectedPermissions: ['SEFIN_READ']
44        }
45     },
46     {
47         path: 'allotments',
```

Ou seja: Atribuímos todas as permissões (ADMIN, CRUD e READ) ao caminho da lista (HomeComponent). Mas para o caminho para adicionar e editar (DetailComponent), atribuímos apenas permissões ADMIN e CRUD. Além disso, para a rota para consulta (ConsultComponent), atribuímos apenas a permissão READ.

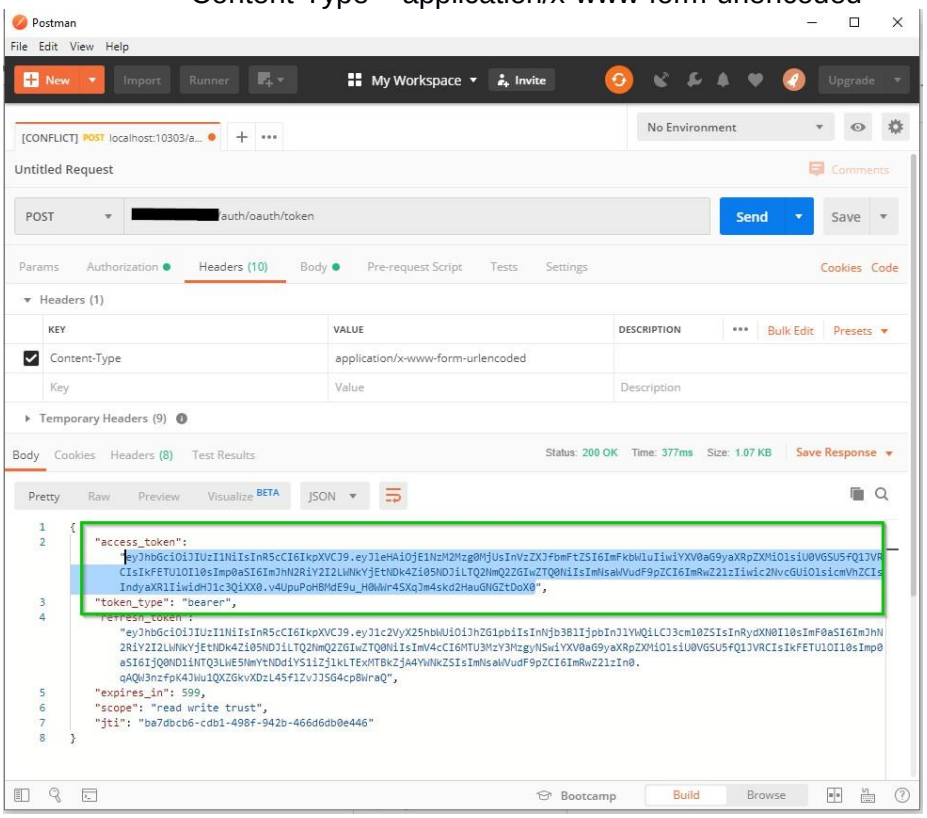
2.6.2.3.2 CanActivate para securizar en components typescript o html:

- No typescript: usaremos o DPGISService e seu método userAuthenticatedHasPermission (string []): booleano. Este método retornará true se o usuário autenticado tiver alguma das permissões recebidas na matriz de cadeias. E falso se não tiver nenhum.
- Em html, a política é acessada com uma chamada* userHasPermissions = "['XXXX']", que é adicionado ao próprio componente html. Por exemplo, o menu para gerenciar usuários, grupos e permissões é mostrado apenas para usuários com permissão 'ADMIN' e é implementado da seguinte forma no botão html:

```
<div class="admon-menu-container">
 <button mat-button [matMenuTriggerFor]="menu">{{'ADMON_MENU_OPTION_ADMON' | oTranslate}}<mat-icon>account_box</mat-icon></button>
    <mat-menu #menu="matMenu">
        <button mat-menu-item routerLink="admon/profile"><mat-icon>account_circle</mat-icon>{{'ADMON_MENU_OPTION_PERFIL' | oTranslate}}</button>
        <button mat-menu-item routerLink="admon" "userHasPermissions="['ADMIN']"><mat-icon>desktop_windows</mat-icon>{{'ADMON_MENU_OPTION_ADMON' | oTranslate}}</button>
        <button mat-menu-item routerLink="admon/meetings" userHasPermissions="['ADMIN']"><mat-icon>duo</mat-icon>{{'ADMON_MENU_OPTION_MEETINGS' | oTranslate}}</button>
        <button mat-menu-item (click)="logout()"><mat-icon>https</mat-icon>{{'ADMON_MENU_OPTION_LOGOUT' | oTranslate}}</button>
    </mat-menu>
</div>
```

2.6.3 Anexo I: Obter um token com uma Chamada externa.

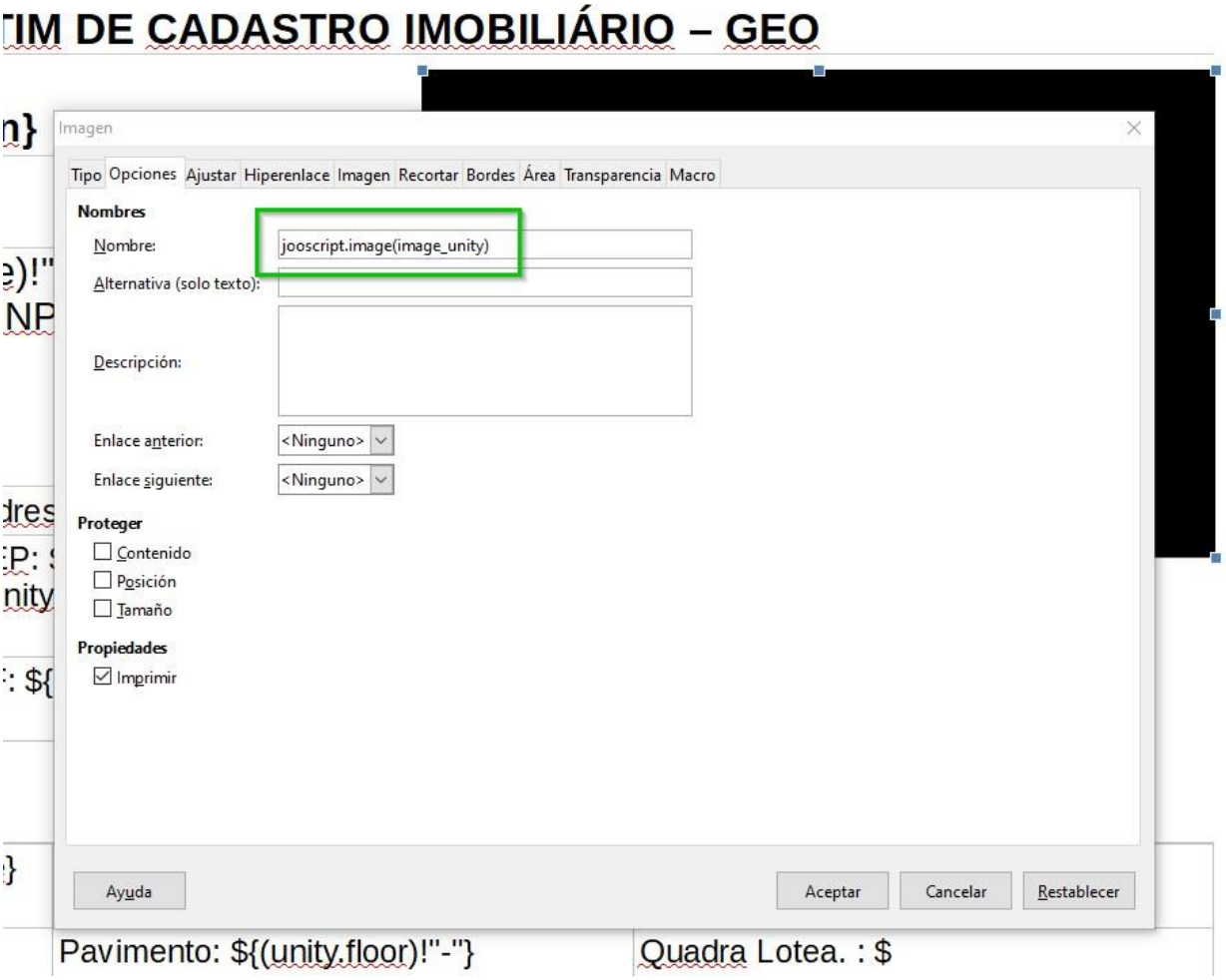
- Usando Postman (https://www.getpostman.com/)
- Criar um Request de tipo POST
- URL= IP_SERVER:PORT_SERVER/auth/oauth/token
- Authorization:
  - Type = Basic Auth
  - Username and password (o cliente de acceso configurado em 3.6.1.1)
- Headers:
  - Content-Type = application/x-www-form-urlencoded



3 Planilhas.

Para a geração de documentos, existem modelos que serão diferentes dependendo do ambiente. Esses modelos estão no formato ODT, embora no futuro possam estar em qualquer formato do LibreOffice. Para inserir dados do aplicativo nesses modelos, a biblioteca JODReports foi usada, que por sua vez usa o Freemarker. Com o que poderia ser usado diretivas, expressões, etc. O Freemarker fornece. Nós descrevemos abaixo os campos disponíveis para cada um dos modelos existentes.

(*) Para inserir as imagens no documento .odt do modelo, você deve atribuí-lo, acessando as Propriedades, o Nome: jooscript.image (xxxxxxxxxx)

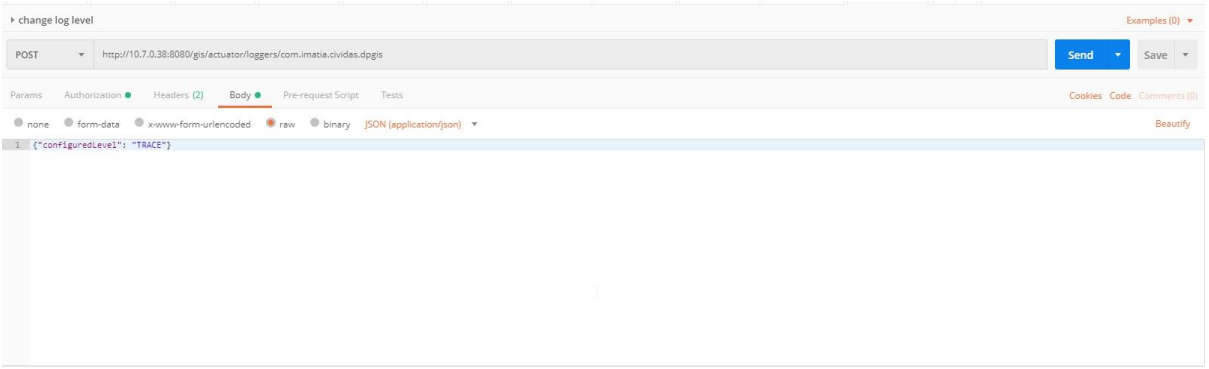


4. Outros

4.1 Monitorización

Os microsserviços ativaram o atuador api que permite o monitoramento e a modificação a quente de certos parâmetros dos microsserviços.
Por exemplo, para gerenciar o nível de log, temos o atuador do terminal / loggers /. Se quisermos modificar o nível de log para um pacote específico. Devemos seguir estas etapas:

1. A obtenção de um token de administrador está descrita no Apêndice III do documento de configuração da localidade.
2. Usando o Postman (ou alguma outra ferramenta que permita fazer solicitações do tipo POST), faça uma solicitação para o URL, passando o token do Bearer obtido no ponto anterior e os parâmetros necessários, conforme mostrado na captura de tela.



4.2 Carga de imagens de imobiliario

Para fazer upload das imagens da propriedade, existe o URL "sfin / upload-data / copyImages", Este url recebe os seguintes parâmetros:

- ImagesSource: A fonte dessas imagens é indicada aqui, RECADASTRAMENTO (Imagens obtidas pela equipe da SQLTecnologia)
- SourcePath: caminho onde essas imagens estão no servidor. Lembre-se de que, para o serviço ver essa pasta, ele deve estar visível no contêiner do Docker que executa o microsserviço SFIN.

Esse URL é protegido para usuários com permissões de administrador e um token de autenticação deve ser passado a ele; portanto, é recomendável fazer a chamada usando swagger.
O URL do serviço no swagger é "sfin / swagger-ui.html? Urls.primaryName = upload-data # / upload-data-controller / copyImagesUsingPUT ”

4.3 Carga de adjuntos de inmobiliario

Da mesma forma que, para carregar as imagens da propriedade, você pode fazer o upload dos arquivos anexados, para isso, temos o URL "sfin / upload-data / copyAttachments". Este URL recebe os seguintes parâmetros:

- attachmentsSource: Aqui é indicada a origem desses arquivos anexados que podem ser RECADASTRAMENTO (Esboço gerado pela equipe da SQLTecnologia)
- SourcePath: caminho em que esses anexos estão no servidor. Lembre-se de que, para o serviço ver essa pasta, ele deve estar visível no contêiner do Docker que executa o microsserviço SFIN.

Esse URL é protegido para usuários com permissões de administrador e um token de autenticação deve ser passado a ele; portanto, é recomendável fazer a chamada usando Swagger.
O URL do serviço no swagger é "sfin / swagger-ui.html? Urls.primaryName = upload-data # / upload-data-controller / copyAttachmentsUsingPUT"

4.6 token y refresh token

No swagger de microserviço de autenticação, temos acesso a 2 operações públicas: **getToken** e **refreshToken**. Primeiro, precisamos iniciar o getToken para obter um token de autorização para dpgis. Este token tem uma duração padrão de 10 minutos. Quando esse primeiro token expirar, podemos obter um novo token ou atualizar o primeiro. A atualização tem uma duração padrão de 30 minutos.

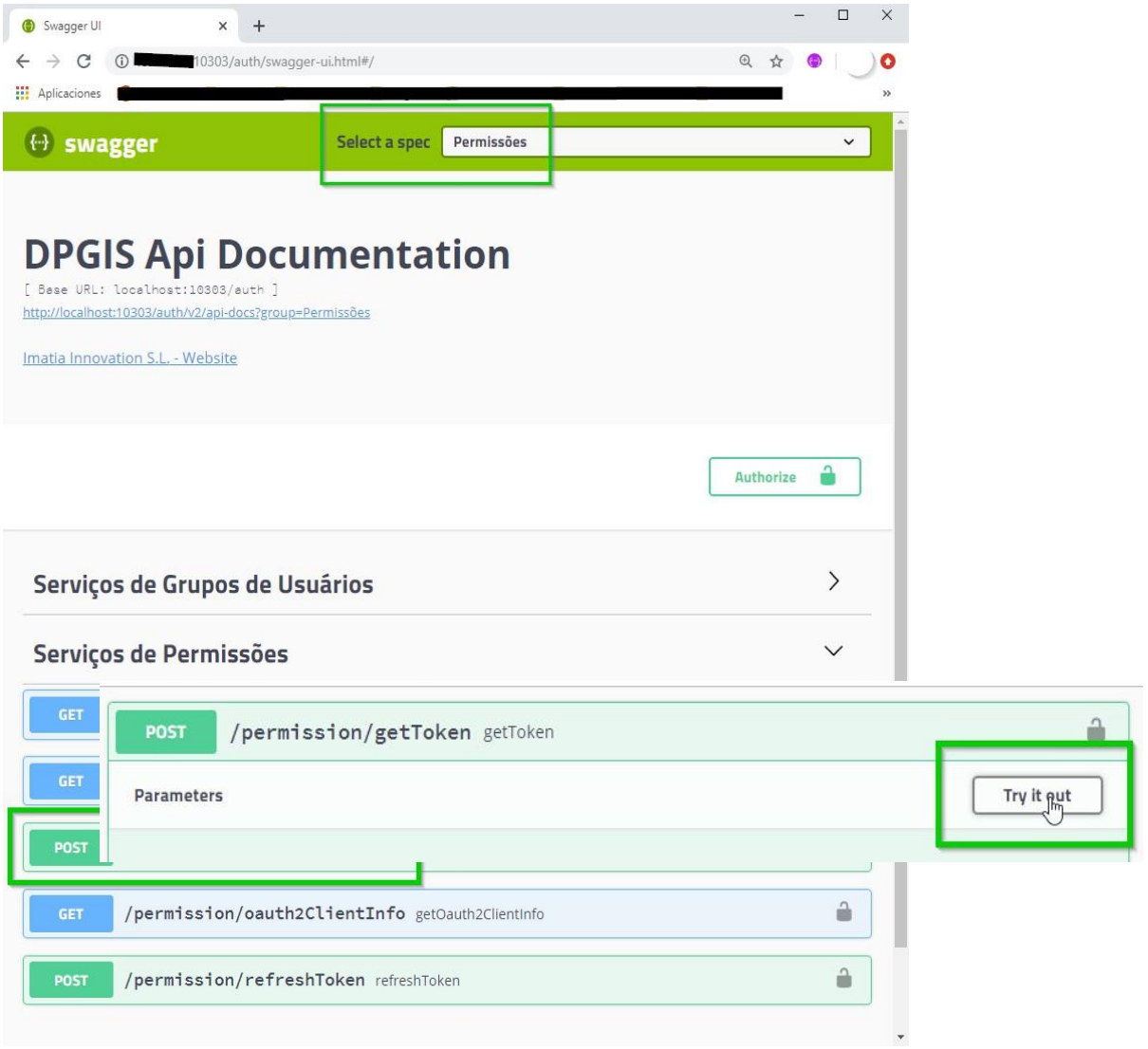
NOTA:

Os tempos de duração podem ser configurados em milissegundos no yml de autenticação:

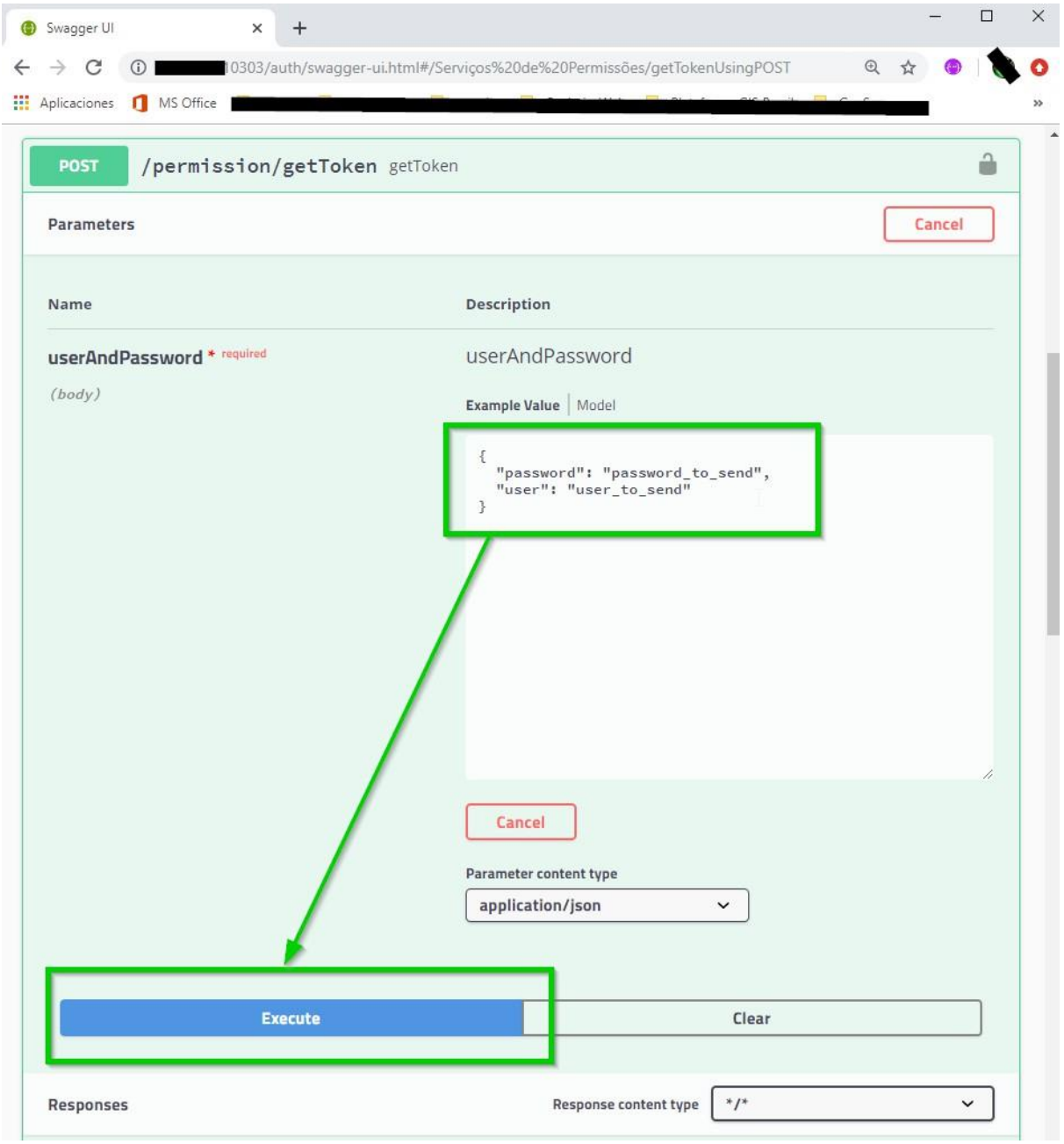
```
dpgis.oauth.accessTokenValiditySeconds=600
dpgis.oauth.refreshTokenValiditySeconds=1800
```

1) getToken:

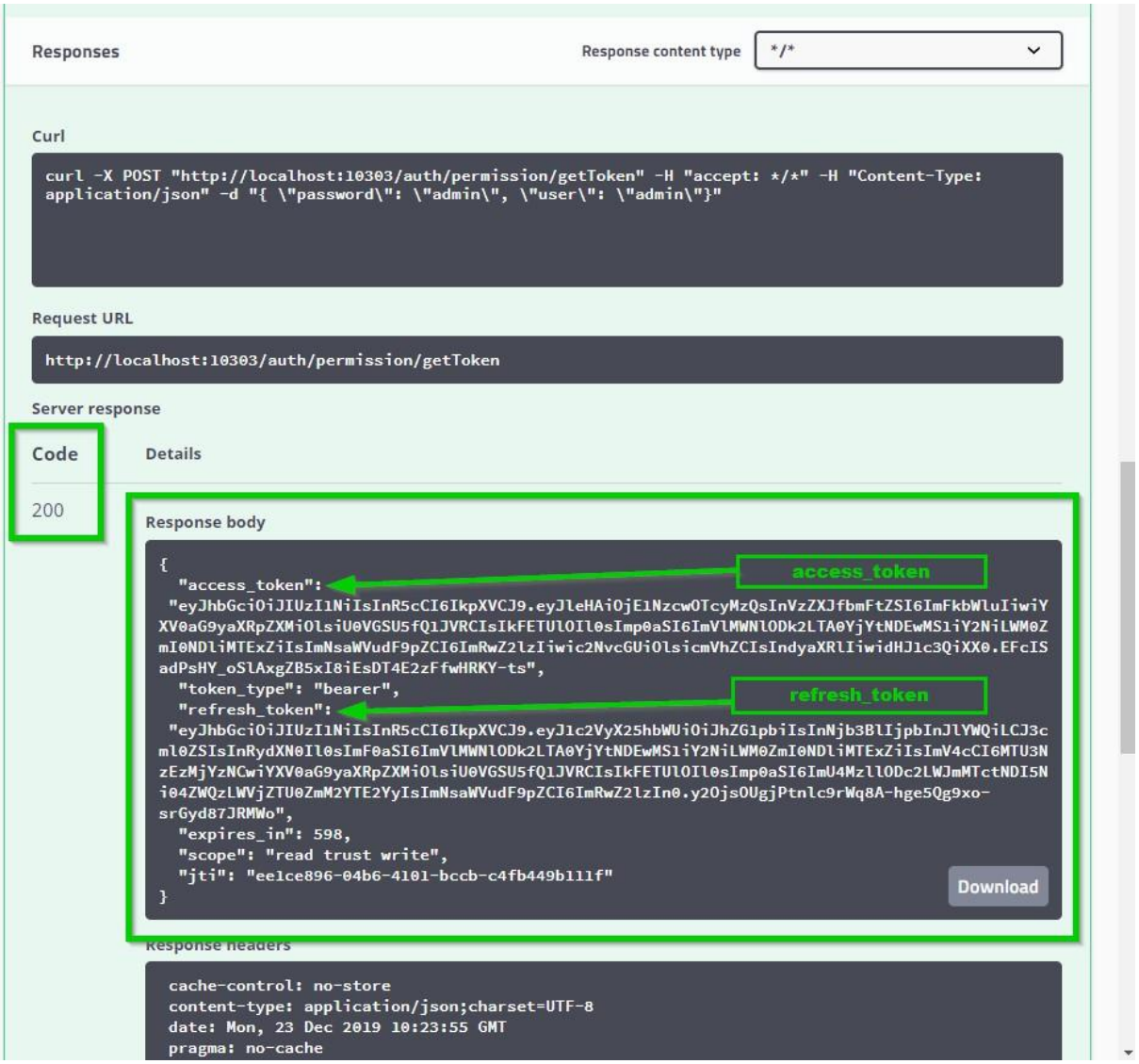
- Acessamos o swagger do microserviço de autenticação
- No grupo Permissões, acessamos a operação getToken



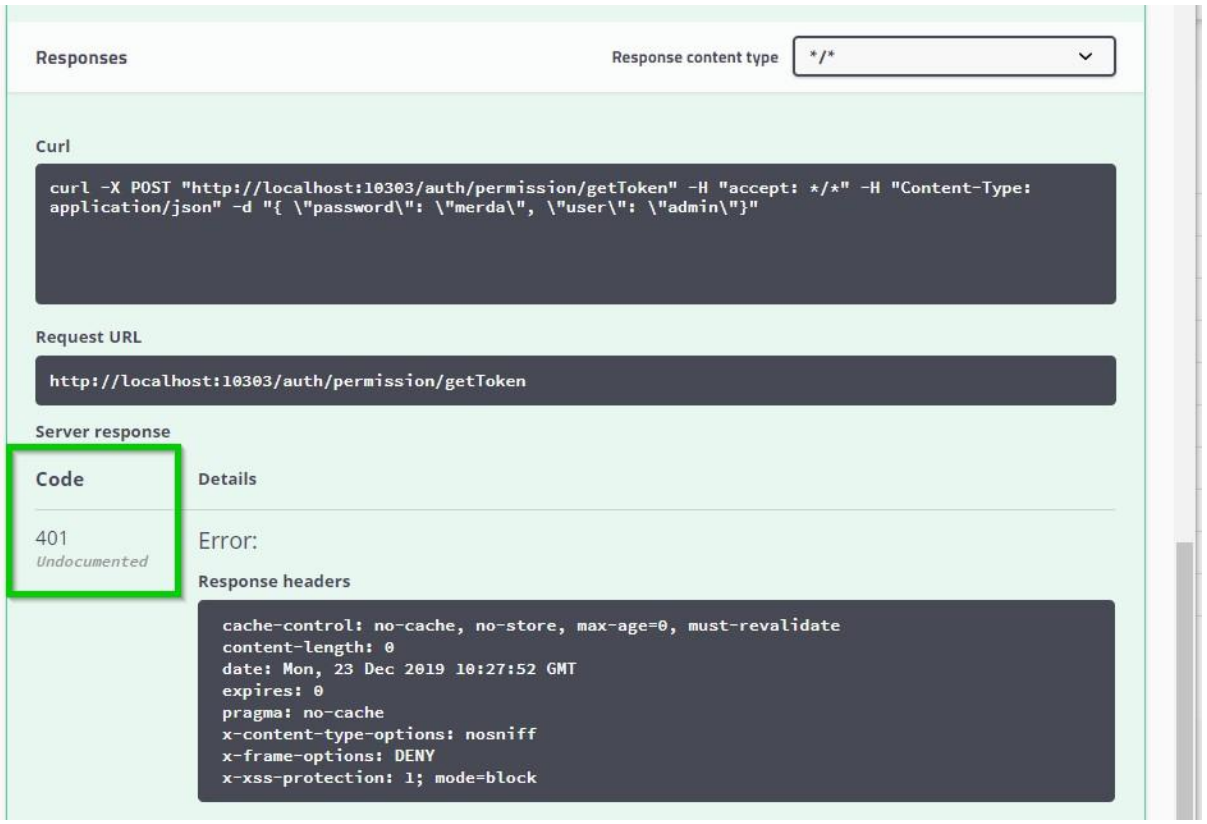
- Na seção Parâmetros, insira o usuário e a senha com os quais queremos ser autenticados no dpgis:



- Na seção Respostas, obteremos a resposta ao envio da solicitação:
  - Se a solicitação foi aprovada e o nome de usuário / senha estão corretos:

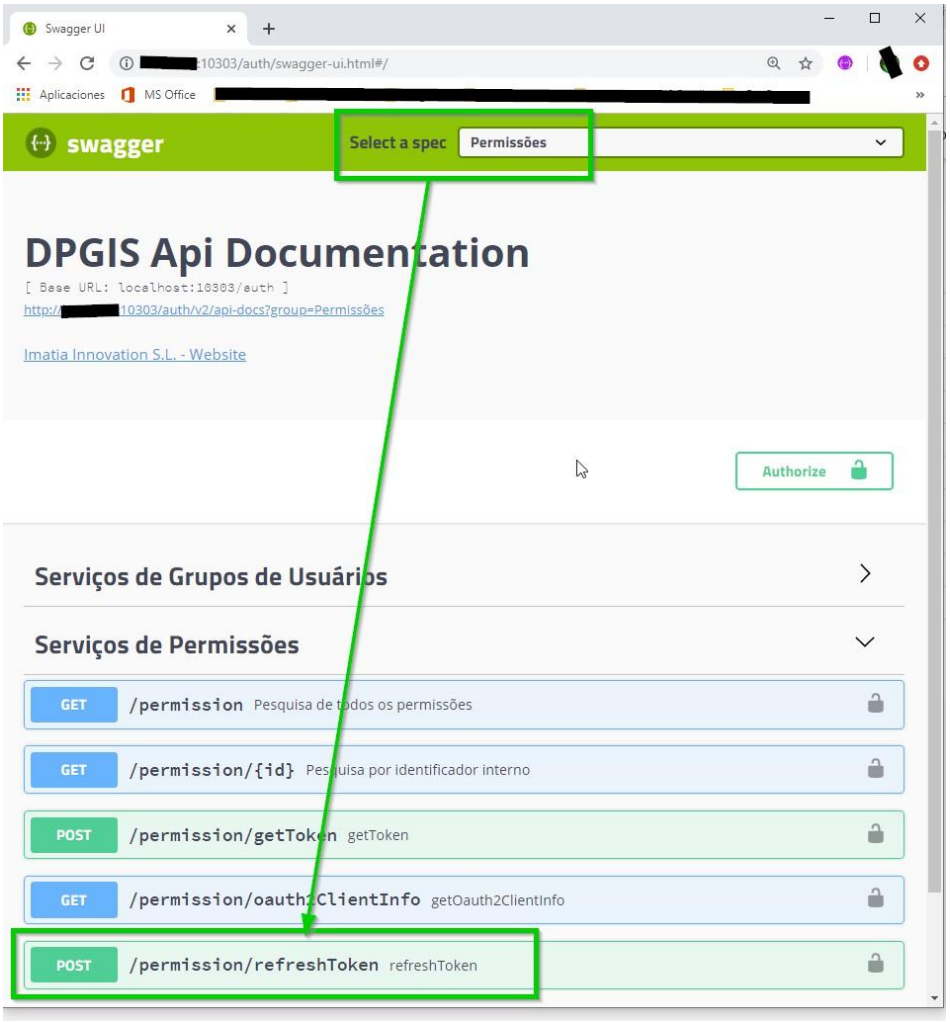


- Se você não foi autenticado:



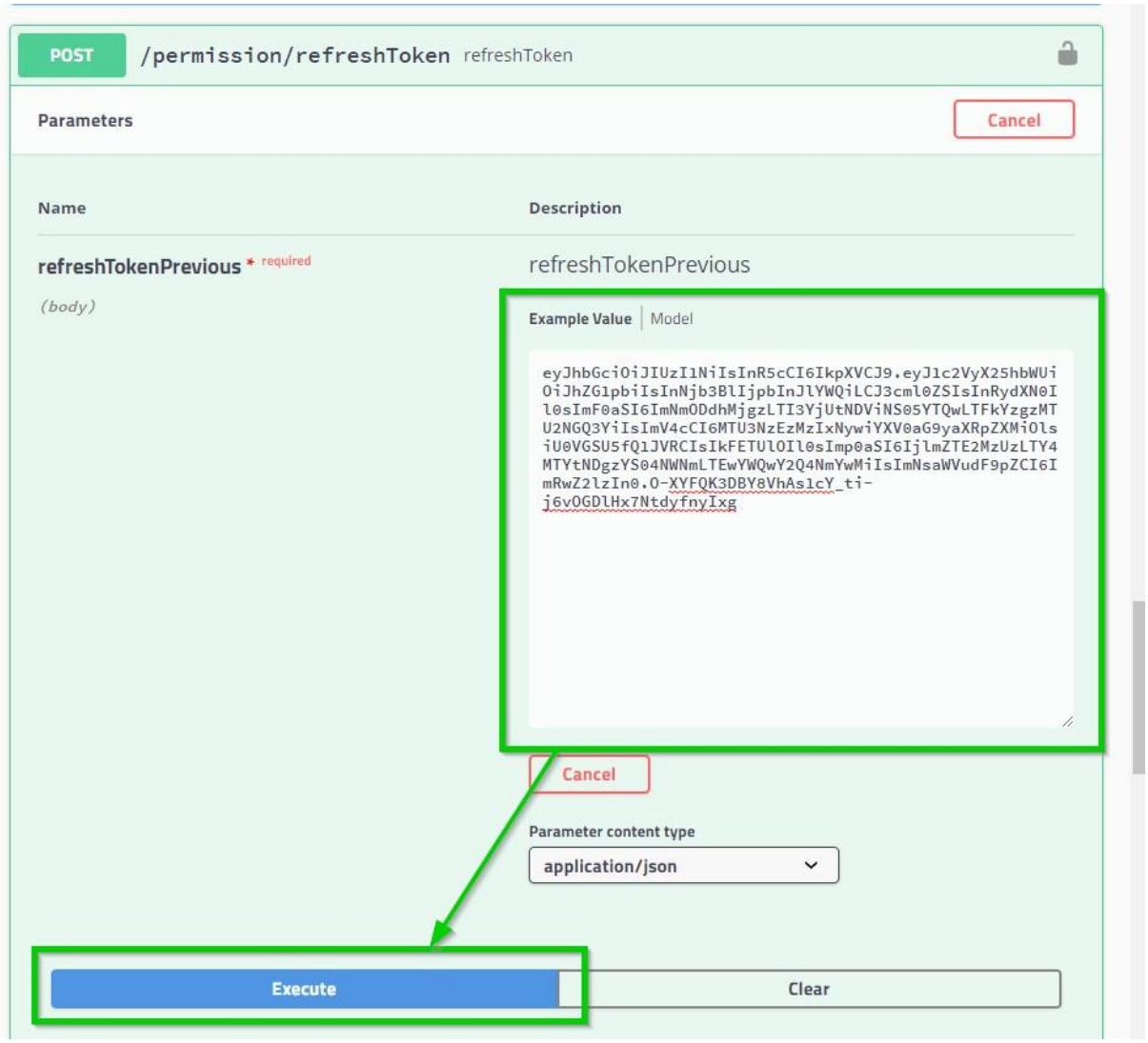
2) refreshToken:

- Acessamos o swagger do microserviço de autenticação;
- No grupo Permissões, acessamos a operação refreshToken;

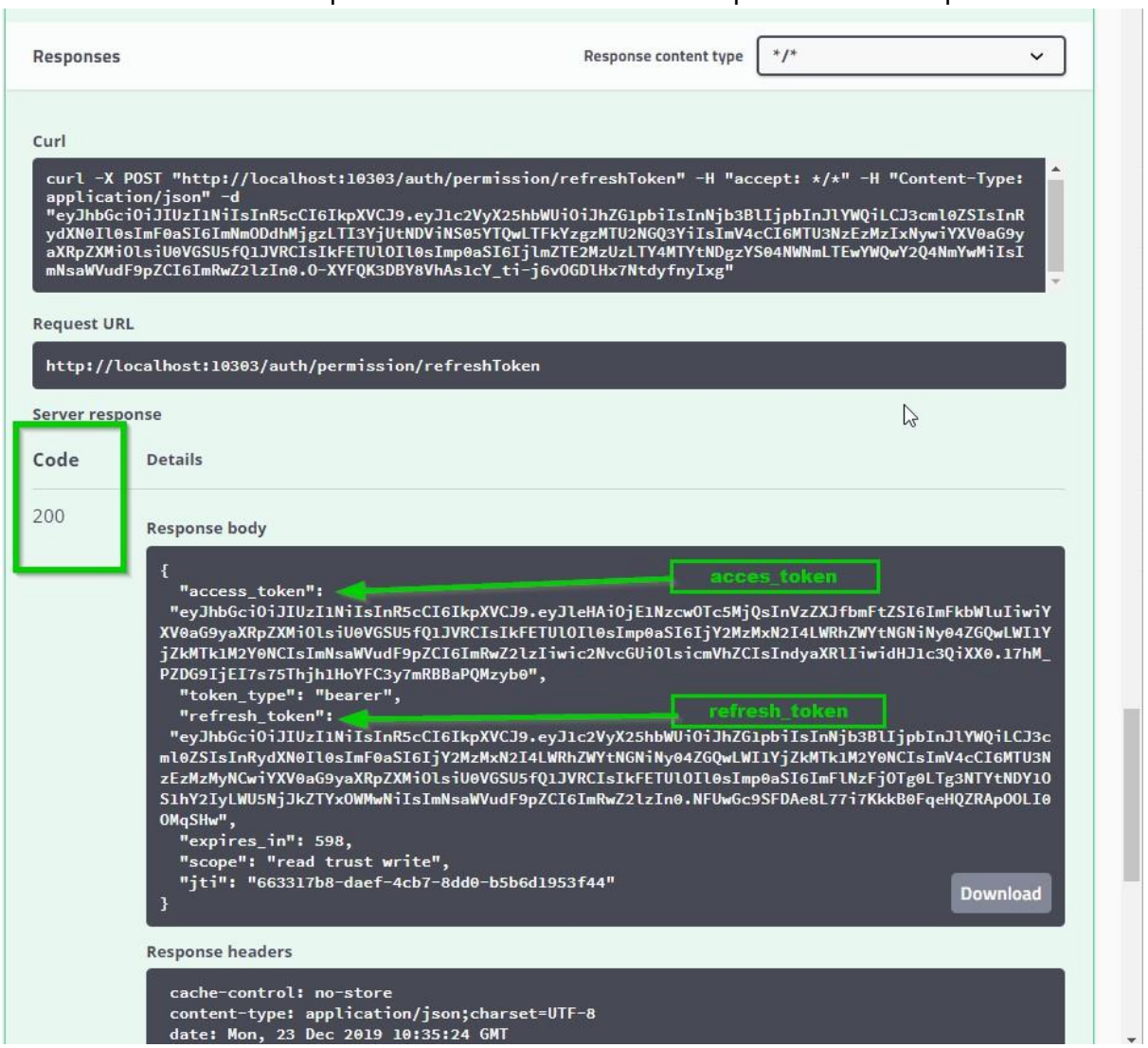


- click em Try it out

- Na seção Parâmetros, insira refresh_token que obtivemos da resposta na etapa anterior (getToken):



- Na seção Respostas, obteremos a resposta ao envio da solicitação:
  - Se a solicitação foi aprovada e o refresh_token corresponde à anterior, obteremos o novo access_token e o refresh_token que devemos enviar novamente quando o atual expirar:



- Se o refresh_token não corresponder a nenhum anterior: