UNIVERSIDADE D
COIMBRA

Luís Guilherme Marques Duarte

2020_N20 REPRESENTAÇÃO GEORREFERENCIADA DE DADOS DE DESEMPENHO E QUALIDADE DE SERVIÇO EM REDES DE TELECOMUNICAÇÕES

VOLUME 1

Relatório de Estágio no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas Inteligentes orientada pelo Professor Alberto Cardoso, pelo Engenheiro Miguel Marques e pela Engenheira Cristina Pires e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Outubro de 2021

Faculdade de Ciências e Tecnologia
Departamento de Engenharia Informática

2020_N20 Representação georreferenciada de dados de desempenho e qualidade de serviço em redes de telecomunicações

Luís Guilherme Marques Duarte

Relatório de Estágio no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas Inteligentes orientada pelo Professor Alberto Cardoso, pelo Engenheiro Miguel Marques e pela Engenheira Cristina Pires e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Outubro de 2021

UNIVERSIDADE D
COIMBRA

Resumo

Nas últimas décadas tem-se verificado um aumento significativo da quantidade de dados. Este aumento deve-se à crescente utilização de equipamentos eletrónicos. Todos estes equipamentos geram dados de forma contínua, sobrecarregando as ferramentas de armazenamento. Deste modo existem serviços como o Altaia, que recolhem, analisam e apresentam esses dados de forma organizada. Este trabalho prende-se com uma das vertentes da apresentação de dados, nomeadamente a georreferenciação, para a qual existem diversas ferramentas.

O objetivo deste trabalho começou por ser encontrar uma ferramenta de representação de dados georreferenciados que melhor sirva o propósito do serviço. Na segunda fase, pretendeu-se dinamizar o processo adjacente à representação, criação de *layers* e disponibilização das mesmas para consulta, sobre as quais se podem aplicar filtros temporais, optar por vários formatos de representação e fazer consultas detalhada dos dados.

O presente documento mostra as etapas de análise das ferramentas de representação geográfica assim como a avaliação feita sobre elas. Após a seleção das ferramentas a utilizar, são descritos os requisitos para a segunda fase e a solução encontrada para os satisfazer. Posteriormente são explicadas as várias funcionalidades desenvolvidas e explicada a relação entre os vários componentes utilizados. Por fim, são apresentados os testes feitos à solução apresentada. Em suma, os requisitos foram atingidos, ainda assim, existem diversas melhorias que podem ser abordados em trabalhos futuros.

Palavras-Chave

Serviços REST; GeoServer; OpenLayers;

Abstract

During the last decades there was a significant increase of generated data. This is due to the increasing use of electronic equipment's. Each of this equipment's generates data continuously flooding the storing databases. Having this in mind, there are services like Altaia, that collect, analyse, and show this data in an organized and treated manner. This project is related with one of the strands of data visualization, namely georeferencing. There are multiple possible frameworks for georeferenced data representation.

The purpose of this project, in the first phase, was to identify the frameworks for georeferenced data representation that satisfies the needs of the Altaia service. In the second phase, it was intended to develop more autonomous processes leading up to the representation, layer creation and making them available for consultation. During that consultation we can apply temporal filters, chose various representation forms, and even check detailed information of that data.

The present document shows the steps of the analysis of geographic representation tools as well as the criteria used for their selection. Upon decided wich of those to use, it's described the requirements for the second phase and the solution devised to meet them. Furthermore, it's explained the features developed and how the used component interacts with each other. In the end the tests for the presented solution are shown. In conclusion, despite the work done, and even though the requirements were met, there are plenty of improvements that can be addressed in the future.

Keywords

REST Services; GeoServer; OpenLayers;

Agradecimentos

Agradeço a todos os orientadores, que me apoiaram durante todo este processo, nomeadamente, o professor Alberto Cardoso, o Eng.º Miguel Marques, o Eng.º Manuel Araújo e a Eng.ª Cristina Pires, que sempre mostraram grande paciência para as minhas questões e disponibilidade para as esclarecer.

Agradeço também aos membros da equipa do Altaia que me acolheram e se mostraram sempre disponíveis a ajudar com qualquer dúvida que tivesse.

Índice

Acrónimos

GPS: Global Positioning System
REST: Representational State Transfer
API: Application Programming Interface
NQM: Network Quality Management
SQM: Service Quality Management
CQM: Customer Quality Management
SLO: Service Level Objective
QoE: Quality of Experience
HVA: High Value Accounts
GIS: Geographic Information System
WGS 84: World Geodetic System
ESRI: Environmental Systems Research Institute
OSGeo: Open Source Geospatial Foundation
OGC: Open Geospatial Consortium
WFS: Web Feature Service
WMS: Web Map Service
WCS: Web Coverage Service
WPS: Web Processing Service
JVM: Java Virtual Machine
ORM: Object Relational Mapper
CDI: Contexts and Dependency Injection
JPA: Java Persistence API
JAX-RS: Jakarta RESTful Web Services

Lista de Figuras

Lista de Tabelas

Capítulo 1
Introdução

O presente trabalho é realizado no âmbito da unidade curricular Dissertação/Estágio em Sistemas Inteligentes, do Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.
O estágio decorreu em parceria com a empresa Altice Labs. Esta é uma empresa do grupo Altice, com foco no desenvolvimento tecnológico quer de software quer de hardware. Desde a sua criação conta com alguns feitos a nível mundial. Desses consta Portugal como sendo o 1º país a ter rede telefónica totalmente automatizada, nas décadas de 50 e 60, o primeiro cartão de telemóvel pré-pago, na década de 90 e um dos mais recentes, em 2017, primeiro "GPON Gateway 802.11 ac 4x4" (router com fonte direta de fibra ótica). Altice Labs está sediada em Aveiro, mas também tem polos em outras zonas do mundo como Israel, França, República Dominicana e Estados Unidos [1].
O objetivo do trabalho é dinamizar o módulo de representação de dados georreferenciados do produto Altaia. Tem uma componente *frontend* onde é usada uma ferramenta de georreferenciação, e uma componente de *backend* que tem como objetivo dinamizar todo o processo.

1.1 Altaia

O Altaia é uma solução de *Assurance end-to-end* que calcula em tempo-real indicadores de desempenho de rede, indicadores de serviço e dados de utilizadores. Os indicadores produzidos fornecem aos CSP (*Comunication Service Providers*) informação essencial para a monitorização da rede e garantia da qualidade dos serviços fornecidos.
Tendo por base uma plataforma modular e flexível baseada em Java EE, usando APIs normalizadas de gestão do TMForum e adaptadas para Java pelo consórcio OSS/J, tem a capacidade de se integrar com facilidade com os sistemas de rede e de serviços. Suporta tanto transmissões como informações com referência temporal, garantindo assim a obtenção de todo o potencial da informação e desencadear ações proativas.
A aquisição de *data source*s, é garantida por subsistemas de mediação responsáveis pela sua coleta e normalização, sendo armazenados na base de dados Altaia DBN0.
O sistema Altaia é baseado numa arquitetura escalável, (Figura 1), capaz de recolher, enriquecer, calcular e apresentar os dados em diferentes perspetivas: NQM (*Network Quality Management*), SQM (*Service Quality Management*) e CQM (*Customer Quality Management*).
NQM compreende, perceber como funcionam as redes dos operadores e assegurar a melhor performance possível, otimizar adequadamente a capacidade da rede de modo a evitar desperdício de recursos, configurar alarmes para momentos de redução de performance, e análise profunda dos dados através de aplicações de análise.

SQM é composto por SLO's (*Service Level Objectives*) que podem ser ativados para deteção de degradação do serviço, fornece serviços de resolução de problemas e melhoria de qualidade adequadas, possibilita rápida deteção de degradação do serviço e possíveis impactos, e ainda relaciona os recursos da rede com os serviços e clientes.
CQM engloba monitorização em tempo real da QoE (*Quality of Experience*) para serviços móveis fixos e convergentes, gestão desempenho de HVA (*High Value Accounts*) e clientes VIP, disponibilização de informação relativa a contas de clientes específicos aos seus gestores em tempo real, e ainda permite a verificação do cliente.

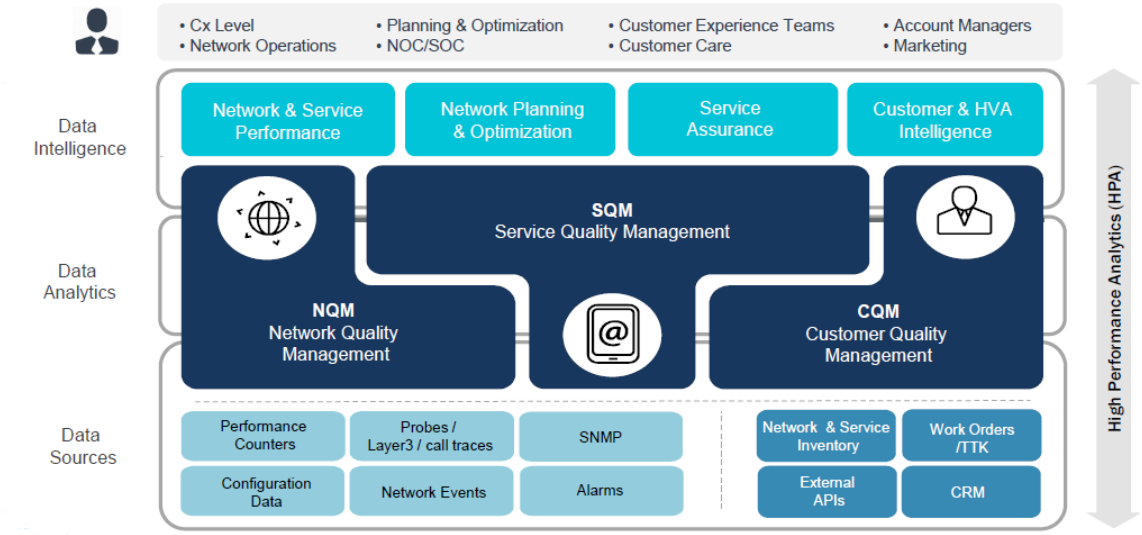


Figura 1 - Arquitetura do Altaia.

Data Inteligence consiste em um conjunto de aplicações com capacidade de consulta, análise e criação de alarmes. Essas aplicações usam os dados gerados pelo Data Analytics acedendo à informação em tempo real e histórico.
Dentro do módulo de consulta de dados, o sub-módulo, no qual este trabalho será integrado, consiste na representação geográfica de dados. O objetivo do módulo é fornecer aos clientes um mapa no qual eles podem analisar, numa perspetiva geográfica, os indicadores calculados pelo sistema. A informação apresentada varia consoante os indicadores selecionados pelo utilizador.

1.2 Problema

Como referido anteriormente, o produto Altaia trabalha com os dados de grandes empresas, conseguindo deste modo obter um volume de dados muito considerável. Quanto maior o volume de dados, maior a quantidade de informação a ser apresentada ao cliente. Ferramentas como o ArcGIS são capazes de lidar facilmente com esses dados, permitindo a sua representação geográfica, contudo não são as mais adequadas para o contexto do produto. Atualmente a representação geográfica usada pelo Altaia consulta *views* geradas manualmente e posteriormente carregadas para o contexto web, onde são apresentadas. Isto gera problemas como a necessidade de criar manualmente novas vistas cada vez que um cliente pede novas métricas, para além do facto de todo este processo, representado na Figura 2, ser manual.

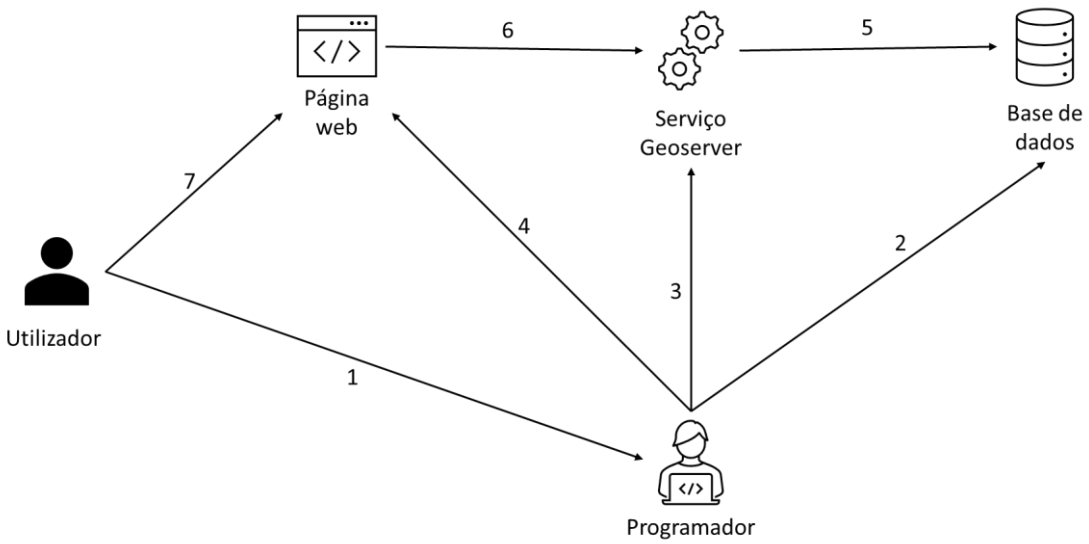


Figura 2- Diagrama descritivo do modelo lógico inicial. Cada número representado corresponde à seguinte tarefa: 1 – Pedir para criar a representação geográfica de um conjunto de dados (nova). 2- Consultar a base de dados para identificar os atributos necessários. 3- Criar uma layer que contenha a informação pretendida. 4- Disponibilizar a consulta da layer na interface web. 5- O GeoServer recolhe informação para layer segundo a query introduzida pelo programador. 6- A página web representa a layer gerada. 7- Consultar a informação pretendida georreferenciada.

O trabalho inicial consistiu num estudo de várias ferramentas de representação de dados georreferenciados. Dentro de um conjunto de hipóteses estas foram testadas segundo um conjunto de critérios que permitia diferenciar o comportamento das mesmas em circunstâncias semelhantes às condições pré-definidas.
A fase seguinte do trabalho foi o levantamento de requisitos para a automatização do processo. Após o seu levantamento, foi estabelecido o modelo lógico a utilizar assim como as interações necessárias entre sistemas para que tal seja possível. Esta fase só era possível apos a determinação de todos os componentes a utilizar.
A última fase de trabalhos consistiu em construir um sistema que acomodasse todas as tarefas que eram anteriormente manuais. Para tal, foram utilizados serviços REST, capazes de comunicar com os vários componentes envolventes e ao mesmo tempo agilizar o processo de consulta dos dados, permitindo ao utilizador criar e consultar informação em tempo real.

1.3 Metodologia

A equipa na qual fui integrado trabalha em SCRUM, uma metodologia de desenvolvimento ágil. Os sprints eram de 15 dias (2 semanas), ao fim dos quais se fazia a retrospetiva do *sprint*. Apesar de participar nas reuniões diárias ("*stand-ups*") associadas à metodologia de trabalho, a metodologia que utilizei aproxima-se mais de um modelo *waterfall* incremental.
Dadas algumas das especificações do projeto era necessário realizar as tarefas numa determinada sequência lógica devido às dependências entre ela. Outro fator importante foi ser um projeto individual, pelo que não era expectável a realização de várias tarefas

em simultâneo. Ainda assim por vezes foi necessário alterar, adicionar ou remover tarefas tendo em conta os resultados obtidos.

1.4 Estrutura do Documento

Deste documento consta uma introdução onde é abordado o tema do trabalho, o sistema onde o trabalho se integra, o problema a resolver e metodologia de trabalho.
Segue-se capítulo do planeamento, onde são descritas as tarefas realizadas ao longo do estágio assim como uma breve explicação.
Também contem um capítulo sobre o estado da arte onde se contextualiza o leitor com a história e algumas das atuais ferramentas de georreferenciação, bem como ferramentas de trabalho utilizadas.
Posteriormente é apresentado o capítulo de análise de ferramentas onde são evidenciados os requisitos iniciais, os critérios usados para comparar as ferramentas analisadas, os dados a utilizar nos testes assim como os resultados e conclusões sobre os mesmos.
De seguida é apresentada a solução para a segunda parte do problema, a parte de dinamização do processo de criação de *layers* onde são apresentados os requisitos, e proposta de solução.
Após apresentada a solução proposta, encontra-se o capítulo com as funcionalidades necessárias ao seu funcionamento, onde são evidenciados os seus principais detalhes e como elas se complementam.
O capítulo seguinte retrata todos os testes de funcionalidades realizados para comprovação dos resultados obtidos garantido que todos os requisitos acordados são atingidos.
Por fim são apresentadas as conclusões onde é feita uma retrospetiva do trabalho realizado, apresentadas propostas de desenvolvimento futuro e melhoria, bem como uma apreciação de todo o processo.

Capítulo 2
Planeamento

O estágio na Altice Labs começou a 1 de setembro de 2020, e estende-se até 31 de julho de 2021.

Nas primeiras semanas a tarefa era conhecer o produto, essencialmente modo de funcionamento e partes integrantes, foi feita ao longo de 3-4 semanas. Após essa fase foram abordados os problemas a resolver durante o estágio académico ao longo primeiro semestre, assim como feito um prognóstico das tarefas previstas para o segundo semestre.

Dentro das tarefas para o primeiro semestre comecei por explorar páginas e artigos sobre várias ferramentas de representação de dados. Após identificadas algumas dessas ferramentas e posteriormente à sua discussão com a equipa, foi decidido fazer um estudo de algumas dessas ferramentas. O estudo consistia numa implementação base, que pudesse funcionar como teste. Para cada ferramenta seria construída uma implementação base e seria posteriormente avaliada. De forma a garantir o máximo de transparência dessas avaliações, defini um conjunto de critérios, verificados pela chefia da equipa, que permitissem responder satisfatoriamente ao problema.

Com base nestas tarefas o restante tempo de estágio ao longo do primeiro semestre visou implementar e avaliar cada uma das ferramentas.

É de salientar que o esforço previsto para o primeiro semestre é o equivalente a 12 créditos, ou seja, aproximadamente 16-18 horas semanais. Tendo isto presente, o esforço para cada tarefa identificada pode ser verificado no mapa de Gant (Figura 3).

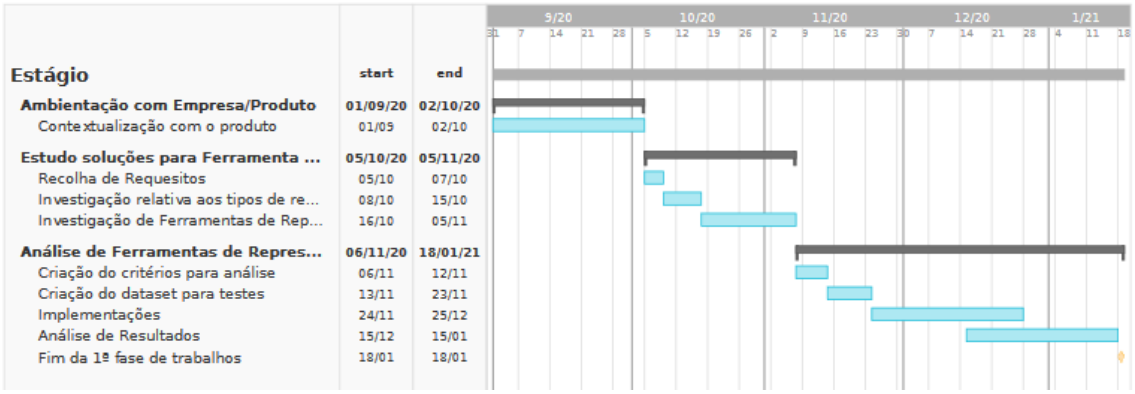
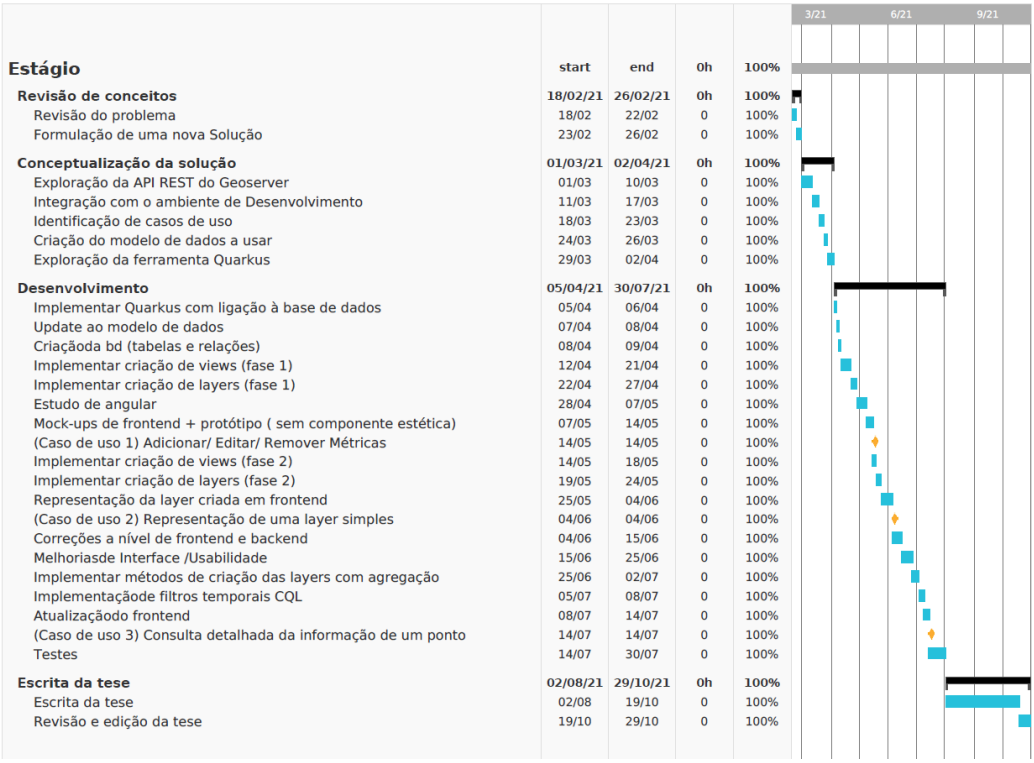


Figura 3 - Mapa de Gantt relativo ao trabalho realizado no 1º semestre.

Para a segunda fase de trabalhos, equivalente ao segundo semestre letivo, perante os resultados e observações feitas à primeira fase de trabalhos foi necessário alterar a abordagem ao problema pelo que as primeiras semanas foram utilizadas para esse efeito.

Depois de conceptualizada uma solução para o problema passei à exploração das ferramentas a utilizar e estruturar o modelo de dados que seria utilizado. A solução

identificada teria um processo de desenvolvimento incremental começando pelas implementações de um caso de uso mais simples até ao caso de uso mais complexo.
O primeiro caso de uso a implementar correspondia à interação mais básica do utilizado com o sistema, permitindo adicionar/ editar / remover métricas.
O segundo caso de uso contemplava já a representação geográfica das métricas de forma simples e genérica. Este método ainda não incluía agregação dos dados por ponto.
O terceiro caso de uso era mais complexo uma vez que introduzia agregação dos dados sobre os pontos, consulta da respetiva informação e localização temporal. Exigiu um esforço maior que o caso de uso 2 tanto pelos métodos utilizados como pela complexidade do processo.
Após tratados os casos de uso propostos iniciou-se a fase de testes que decorreu até ao fim do estágio que terminou a 31 de julho. As tarefas e a sua duração podem ser consultadas na Figura 4, e no apêndice C.



Estágio	start	end	0h	100%
Revisão de conceitos	**18/02/21**	**26/02/21**	**0h**	**100%**
Revisão do problema	18/02	22/02	0	100%
Formulação de uma nova Solução	23/02	26/02	0	100%
Conceptualização da solução	**01/03/21**	**02/04/21**	**0h**	**100%**
Exploração da API REST do Geoserver	01/03	10/03	0	100%
Integração com o ambiente de Desenvolvimento	11/03	17/03	0	100%
Identificação de casos de uso	18/03	23/03	0	100%
Criação do modelo de dados a usar	24/03	26/03	0	100%
Exploração da ferramenta Quarkus	29/03	02/04	0	100%
Desenvolvimento	**05/04/21**	**30/07/21**	**0h**	**100%**
Implementar Quarkus com ligação à base de dados	05/04	06/04	0	100%
Update ao modelo de dados	07/04	08/04	0	100%
Criaçãoda bd (tabelas e relações)	08/04	09/04	0	100%
Implementar criação de views (fase 1)	12/04	21/04	0	100%
Implementar criação de layers (fase 1)	22/04	27/04	0	100%
Estudo de angular	28/04	07/05	0	100%
Mock-ups de frontend + protótipo (sem componente estética)	07/05	14/05	0	100%
(Caso de uso 1) Adicionar/ Editar/ Remover Métricas	14/05	14/05	0	100%
Implementar criação de views (fase 2)	14/05	18/05	0	100%
Implementar criação de layers (fase 2)	19/05	24/05	0	100%
Representação da layer criada em frontend	25/05	04/06	0	100%
(Caso de uso 2) Representação de uma layer simples	04/06	04/06	0	100%
Correções a nível de frontend e backend	04/06	15/06	0	100%
Melhoriasde Interface /Usabilidade	15/06	25/06	0	100%
Implementar métodos de criação das layers com agregação	25/06	02/07	0	100%
Implementaçãode filtros temporais CQL	05/07	08/07	0	100%
Atualizaçãodo frontend	08/07	14/07	0	100%
(Caso de uso 3) Consulta detalhada da informação de um ponto	14/07	14/07	0	100%
Testes	14/07	30/07	0	100%
Escrita da tese	**02/08/21**	**29/10/21**	**0h**	**100%**
Escrita da tese	02/08	19/10	0	100%
Revisão e edição da tese	19/10	29/10	0	100%

Figura 4 - Mapa de Gantt relativo ao trabalho realizado no 2º semestre.

Capítulo 3
Estado da Arte

"Big Data" é um termo quase banal nos dias de hoje na área de informática e apesar de não se saber ao certo quando surgiu, é algo com que todas as empresas com um volume de negócios considerável precisam de saber tratar. Dentro do mundo que é "Big Data" existem diversas áreas, e uma das mais interessantes, quer do ponto de vista económico quer do ponto de vista organizacional é a "Geo Data", ou dados georreferenciados. Este tipo de dados só se torna útil sendo possível representá-los no mapa, no entanto devido à sua grande quantidade de informação esta tarefa pode tornar-se complexa.
Para representação de dados georreferenciados existem diversas ferramentas, umas com a sua própria aplicação, onde todo o tratamento dos dados e criação das *layers* é feito na própria aplicação, como é o caso do *ArcGis*. Existem outras, que individualmente, não são tão poderosas, contudo conseguem um desempenho tão bom ou melhor, quando combinadas, como por exemplo *GeoServer* integrado com *OpenLayers*.
Atualmente falamos de dados georreferenciados, coordenadas GPS e usamos termos como georreferenciação com a maior naturalidade sem nunca ponderar como funcionam ou como surgiram. Em retrospetiva todas estas ferramentas surgiram há bem pouco tempo, mas estão diretamente ligadas à nossa história como civilização.

3.1 História da Georreferenciação

O mais antigo mapa encontrado, data de 6204 a.c., encontrado na Turquia [2]. Esta descoberta evidencia que desde há muitos anos, o ser humano tem necessidade de representar de alguma forma aquilo que vê e conhece, desde os mapas que remontam à Grécia antiga, passando pelos usados durante os Descobrimentos, até aos mapas que temos acesso no nosso telemóvel. À medida que os conhecimentos tecnológicos da humanidade iam aumentando, também a qualidade e o detalhe dos mapas produzidos aumentaram.
Um conceito geralmente associado a mapas é o de cartografia. Cartografia é a ciência de representar num mapa aquilo que é a realidade, ou seja, pode ser considerado o próprio ato de criar um mapa [3]. Uma das grandes evoluções da cartografia deu-se durante os Descobrimentos. Na época, eram necessários mapas com maior fidelidade para permitir que as viagens náuticas fossem realizadas de forma mais eficiente e segura.
Ao longo da História, as grandes evoluções estão geralmente associadas a épocas de conflito. O mesmo aconteceu na cartografia. Durante a Primeira Guerra Mundial surgiu o "reconhecimento fotográfico" (aéreo) [4], em que aviões sobrevoavam as fronteiras e recolhiam informação, da posição do inimigo, através de fotografias. Estes dados permitiram movimentos e ataques muito mais eficazes.

Mais recentemente, nos anos da "Guerra Fria", surgem os primeiros satélites durante a corrida ao espaço. Assim, foram criados os primeiros sistemas que antecederam o GPS que conhecemos atualmente [5]. Nos seus primórdios, estes sistemas conseguiram resolver um dos principais defeitos dos sistemas de localização até à data, a falta de precisão. Anteriormente usavam-se métodos menos eficazes, mas a triangulação com base nos sinais de vários satélites veio permitir localizações com uma margem de erro de metros

.

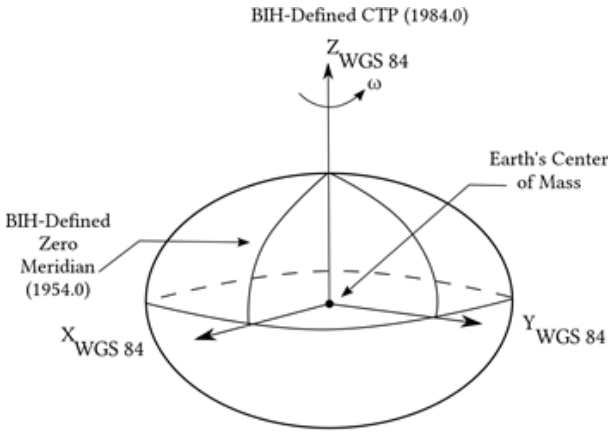


Figura 5- Representação do sistema geodésico WGS 84 [6].

Atualmente o sistema GPS encontra-se muito mais desenvolvido. Existe um conjunto de satélites que cobre todo o mundo como uma malha, o que garante a total cobertura. Existem ferramentas como o Google Earth que são capazes de mostrar todo mundo com grande precisão. O sistema GPS fornece as coordenadas segundo o WGS 84 (*World Geodetic System*), considerado como norma para os principais sistemas de georreferenciação [7].

Sommer e Wade definem georreferenciação como "aligning geographic data to a known coordinate system so it can be viewed, queried, and analysed with other geographic data" [8].

Um dos mais frequentes exemplos de uso da georreferenciação está associado aos aparelhos de GPS. Estes aparelhos tinham as mesmas funcionalidades encontradas atualmente em qualquer aplicação de navegação para o telefone ou carro. Usa a posição triangulada do objeto em causa e coloca-a sobre um mapa, permitindo ao utilizador saber a sua posição exata no mapa e calcular rotas até uma outra localização. As versões mais recentes fornecem, em casos de viagem a velocidade, e no caso de estradas, limite de velocidades associadas e até avisos de acidentes. Tudo isto é possível através da integração de informação de diversas fontes e associação com base na localização geográfica [9].

3.2 Ferramentas de georreferenciação

Tendo em conta que nos dias de hoje a grande maioria da informação tem associado dados de georreferenciação, torna-se imperativo, para quem quer tirar o maior proveito possível da mesma, usar ferramentas de georreferenciação, GIS (*Geographic Information System*). O software GIS consiste em ferramentas capazes de combinar mapas digitais com informação georreferenciada [10]. Existem várias ferramentas GIS tais como:

ArcGIS

ArcGIS é uma aplicação desenvolvida para trabalhar com mapas e informação georreferenciada. Esta pode ser integrada com uma base de dados. Permite a partilha dos mapas criados entre aplicações e páginas web. A sua primeira versão foi lançada em 1999 com o nome "ArcMap" até 2001 onde mudou o nome para "ArcGIS" com a versão 8.1. A aplicação pertence ao instituto ESRI (*Environmental Systems Research Institute*), fornecedor de vários exemplos de sotware GIS, o que lhes garante uma grande fatia do mercado deste tipo de ferramentas [11][12].

QGIS

QGIS, também conhecida como *Quantum GIS*, é uma aplicação *open-source*, que permite ver, editar e analisar informação georreferenciada. Pode ser integrado com bases de dados e suporta integração com outros softwares *open-source*. O seu desenvolvimento inicial deve-se a Gary Sherman, em 2002, passando a um projeto suportado pela *OSGeo* (*Open Source Geospatial Foundation*) em 2009, uma instituição não governamental e sem quaisquer fins lucrativos, que incentiva inovações e desenvolvimentos na área dados e tecnologias geoespaciais [13][14].

Leaflet

Leaflet é uma biblioteca de *Javascript* concebida para representação de objetos no mapa. Foi criada por Vladimir Agafonkin em 2010 como "Web Maps API" para a empresa CloudMade, mas conta, atualmente, com uma grande comunidade de "Contributors". Foi concebida com base nos conceitos de simplicidade, performance e usabilidade, sendo atualmente líder para mapas "mobile-friendly" [15].

Mapbox GL JS

Mapbox GL JS é uma biblioteca *javascript* para representações no mapa com base em vetores. Em sentido lato, esta biblioteca transforma a informação a representar no mapa em vetores, usados posteriormente por *WebGL*[16]. A biblioteca pertence ao grupo *Mapbox*. O grupo surgiu em 2010 e tem crescido até aos dias de hoje onde está avaliado em um bilião de dólares. Muitas das funcionalidades disponibilizadas por esta biblioteca têm como base *Leaflet*, projeto que a empresa apoiou no seu início pelo que não surpreende que atualmente o seu criador também pertença ao grupo *Mapbox*,

OpenLayers

OpenLayers é uma biblioteca *javascript* criada com o intuito de gerar mapas dinâmicos para páginas web. Desde a sua criação foi melhorada, até à atualidade, permitindo a utilização de qualquer tipo de dados georreferenciados. *Openlayers* foi criada pela empresa *MetaCarta* em 2005, sendo disponibilizada como *open-source* no ano de 2006. No ano de 2007 tornou-se um projeto suportado pela OSGeo (*Open Source Geospatial Foundation*) [17].

De modo a integrar este tipo de ferramentas com o produto Altaia da Altice Labs, uma vez que este funciona com base numa aplicação web, é necessário usar uma abordagem mais específica, que permita um funcionamento semelhante ao de aplicações poderosas como o *ArcGIS*, embora recorrendo a ferramentas mais simples, mas com desempenho adequado ao problema.

3.3 Ferramentas complementares

A principal funcionalidade das ferramentas referidas anteriormente é a representação de dados e para funcionarem corretamente estas precisam de ter disponíveis os dados. A disponibilização destes dados pode ser feita a partir de ficheiros ou servidores.
Para além da necessidade de fornecedores de dados a ferramentas de georreferenciação são muitas vezes integradas com outras ferramentas de desenvolvimento web, muitas das vezes para permitir aos utilizadores decidir/interagir com a representação.
Seguindo a estratégia anteriormente usada no Altaia, para armazenar / consultar informação usou-se o GeoServer e Angular como ferramenta de *frontend*.

GeoServer

O GeoServer é um software *open-source* usado para um desenvolvimento mais simples de soluções de georreferenciação. Permite a integração de vários repositórios independentes de dados georreferenciados de forma simples e eficaz.
Tratando-se de um servidor web, é capaz de publicar os dados segundo os formatos standard do *Open Geospatial Consortium* (OGC). Os formatos mais comuns são *Web Feature Service* (WFS), *Web Map Service* (WMS) e *Web Coverage Service* (WCS), no entanto o GeoServer permite outros formatos, como *Web Processing Service* (WPS), estando disponíveis através de extensões.
O GeoServer é desenvolvido, testado e mantido por uma comunidade constituída por grupos de indivíduos e organizações de todo o mundo. É um projeto reconhecido pela OSGeo (*Open Source Geospatial Foundation*).

Angular

Angular é uma ferramenta de desenvolvimento de páginas web que usa como base TypeScript.
"TypeScript é uma linguagem de programação de código aberto desenvolvida pela Microsoft. É um superconjunto sintático estrito de JavaScript e adiciona tipagem estática opcional à linguagem. Tipos fornecem uma maneira de descrever a forma de

um objeto, fornecendo melhor documentação e permitindo que o TypeScript valide o código está a funcionar corretamente" [22].
Como plataforma de desenvolvimento, Angular fornece aos seus utilizadores uma ferramenta à base de componentes, ideal para aplicações web escaláveis, um conjunto vasto de bibliotecas capazes de responder às necessidades mais comuns como gestão de formulários ou comunicação entre cliente e servidor, e um conjunto de ferramentas que facilitam o desenvolvimento, teste e atualização do código [23].
Devido às suas características, Angular é uma das ferramentas mais usadas para o desenvolvimento de páginas web, seja por empresas ou *freelancers*.

Quarkus

Quakus é uma ferramenta de Java nativo em *Kubernetes* e *full-stack*, desenvolvido para JVM's (máquinas virtuais Java) e compilação nativa. Otimiza Java para *containers*, conseguindo assim ser uma plataforma eficaz para *serverless*, *cloud* e *Kubernetes*.
O Quarkus está estruturado para funcionar com os padrões Java e as suas principais bibliotecas, como algumas das usadas no presente trabalho, nomeadamente, RESTEasy (JAX-RS), Hibernate ORM (JPA), e muitas outras.
Conta também com uma solução para injeção de dependências baseada em CDI (Contexts and Dependency Injection). Para além disso, a solução inclui uma framework de extensão para expandir a funcionalidade e ao mesmo tempo configurar, incializar e integrar frameworks na aplicação. O processo de adicionar uma extensão torna-se tão simples como adicionar uma dependência, ou através das ferramentas disponibilizadas pelo Quarkus [24].

Dentro das várias bibliotecas disponibilizadas as seguintes são aquelas que foram consideradas essenciais para o desenvolvimento do projeto referido no capítulo 6.

quarkus-hibernate-orm-panache

A biblioteca Hibernate ORM (*Object Relational Mapper*) é na realidade uma implementação de JPA (Java Persistence API) que disponibiliza as ferramentas necessárias para o mapeamento de uma base de dados relacional. Em conjunto com Panache torna a criação de entidades relativamente simples [25]. Panache é uma biblioteca específica do Quarkus usada para simplificar o desenvolvimento da camada de persistência do Hibernate [26].
Esta biblioteca permite assim mapear a base de dados com recurso às anotações da JPA usando Panache para simplificar as configurações necessárias à interação com a base de dados.

quarkus-jdbc-postgresql

Esta biblioteca utiliza Agroal[1]. Agroal é uma implementação moderna e leve de "conection pools", projetada para alto desempenho e escalabilidade. Apresenta uma

[1] Uma vez que se está a usar Hibernate ORM, não é necessário incluir a extensão Agroal de forma explícita.

integração de qualidade com os outros componentes do Quarkus, tais como segurança, gestão de transações, integridade e métricas.
Ao especificar PostgresSQL como tipo de BD, permite a omissão do seu tipo e da sua "driver" ao configurar a BD [27].

quarkus-rest-client and quarkus-rest-client-jackson

Estas duas bibliotecas são usadas em conjunto para a utilização do cliente REST de modo a interagir com outras API's REST. A anotação "@RegisterRestClient" é instrumental para o Quarkus saber que a interface irá funcionar como cliente REST. As restantes são anotações padrão do JAX-RS (*Jakarta RESTful Web Services*) e especificam como aceder ao serviço. Usando a biblioteca "quarkus-rest-client-jackson", o tipo de dados que é assumido para a maioria dos pedidos é JSON. No entanto para agilizar o processo de descodificação é aconselhado o uso da anotação "@Produces" e o tipo de dados pretendido, assim como o uso do modelo de dados através de modelos [28].

Capítulo 4
Análise de Ferramentas de Representação Geográfica

Para análise das ferramentas, foram escolhidas aquelas que respeitam condições iniciai, evidenciadas na secção 4.1, e criado um conjunto de critérios para as avaliar, apresentados na secção 4.2. Foi usando um *dataset* básico, com informação de vários pontos, para realização dos testes no âmbito dos critérios criados para comparação. O *dataset* usado aproxima-se a um conjunto de dados aproximados dos utilizados nos testes finais. Não se usou um dataset de teste maior para comparação, uma vez que esperava ser possível agregar os dados, e esse processo não é da responsabilidade da ferramenta de representação geográfica, mas sim da base de dados.

4.1 Requisitos iniciais

Dentro das várias ofertas de ferramentas de representação foi necessário que estas respeitassem os requisitos inicialmente estabelecidos. Era necessária uma ferramenta capaz de ser integrada com o produto Altaia, o que implicava que fosse compatível com páginas web. Os recursos que esta aplicação poderia consumir não deveriam impactar o desempenho do PC, logo estabeleceu-se que o ambiente de teste fosse o PC disponibilizado pela empresa.
O PC possui um processador Intel(R) Core(TM) i5-4300M CPU @ 2.60GHz 2.60GHz, um total de 8,00 GB de RAM e sistema operativo Windows 10 Enterprise versão de 64-bit.
Dentro das várias possibilidades optei por testar três bibliotecas que apresentavam um conjunto de recursos capazes de satisfazer as necessidades do problema: LeafLet, Mapbox GL JS e OpenLayers.

4.2 Critérios de comparação

Para escolher entre estas ferramentas, tendo como vista a integração em meio empresarial, foi necessário estabelecer alguns critérios de comparação. Os critérios abordam seis áreas:

- **Desempenho**: Relaciona-se com a celeridade de representação no mapa. Para efeitos de cálculo, usa-se o relatório devolvido pela extensão do *Google Chrome* "lighthouse". Para tal é necessário inicializar um servidor local para a página a testar. No relatório, gerado para a página web em desktop apenas, consta o tempo de carregamento até poder ser utilizada.
- **Compatibilidade**: Relaciona-se com as integrações possíveis e/ou viáveis com os diversos browsers. Teste da *framework* em vários browsers com resultado binário, “suporta” ou “não suporta”. Foram usados os browsers: “Mozilla Firefox

versão 78.6.1esr", "Google Chrome versão 87.0.4280.141" e "Microsoft Edge versão 87.0.664.75".

- **Extensibilidade**: Relaciona-se com a curva de aprendizagem necessária para poder usar as ferramentas disponibilizadas pela *framework*. Este critério depende da opinião pessoal do autor e situa-se na escala de 1 a 5, "Muito difícil" e "Muito fácil" respetivamente.
- **Documentação**: Relaciona-se com a quantidade e qualidade da documentação oficial. Este critério depende também da experiência do utilizador e varia entre dois planos. O primeiro plano depende do quão explícita é a informação apresentada para a resolução de um problema detetado. Este oscila entre 1 e 3, "Pouco explícito" e "Muito explícito" respetivamente. O segundo depende da complexidade da informação presente na documentação. Este varia entre 1 e 2, "Muito Complexa" e "Pouco Complexa" respetivamente. O resultado do critério varia entre 2 e 5, soma dos valores anteriores.
- **Comunidade**: Relaciona-se com o número de utilizadores da *framework* e consequentemente a sua atividade em fóruns públicos. Este critério vai depender das estatísticas fornecidas pelos principais fóruns.
- **Preçário**: Relaciona-se com o custo que se aplica ao uso da *framework* em grande escala. Este critério é binário e varia entre "Aplicável" e "Não Aplicável".

O objetivo destes critérios é fornecer à empresa informação credível sobre as ferramentas de representação de dados, e com isso permitir a sua escolha de forma consciente e fundamentada. Visam também responder a todas as perguntas mencionadas no subcapítulo "Problema" referentes ao primeiro problema.

4.3 Dataset de Teste

De modo a garantir que os testes não têm influência dos dados utilizados, foi utilizado o mesmo *dataset* para todas as ferramentas testadas.

O *dataset* original usado foi dataset_TIST2015 [19][20]. Este *dataset* continha informação recolhida pelo Foursquare ao longo de 18 meses sobre check-ins numa escala global. Foi reduzido apenas para dados referentes a Portugal, nomeadamente as cidades do Porto e Lisboa. Após a redução, os dados foram convertidos para o formato *GeoJSON*. Foi adicionada também alguma informação, número de visitas em cada ponto, para que esta possa ser representada.

O uso do formato *GeoJSON* prende-se com a transversalidade. Todas as ferramentas a testar são capazes de interpretar este tipo de dados. Assim garantimos que resultados apresentados não dependem diretamente dos dados, mas apenas das bibliotecas usadas.

4.4 Resultados

Para cada *framework* em análise foi criada uma implementação base sobre a qual foram testados os critérios aplicáveis presentes na secção 3.2. Da implantação base fazem parte os ficheiros HTML, CSS e JS necessários ao funcionamento da página. Os seguintes resultados encontram-se divididos por *frameworks*.

4.4.1 Leaflet

No que concerne ao desempenho, o carregamento da página web é "lento". Segundo o "Lighthouse" demorou 26.2 segundos e foi devolvido um "erro" avisando que a medição poderia não estar completa uma vez que ultrapassou o tempo definido para os testes. Considera-se assim que o carregamento demora mais de 26.2 segundos.

Tabela I - Tabela de compatibilidades da framework *Leaflet*.

	Mozilla Firefox	Google Chrome	Microsoft Edge
LeafLet	Compatível	Compatível	Compatível

No que diz respeito à compatibilidade, a tabela I mostra os browsers nos quais a implementação de teste funcionou.

Relativamente à extensibilidade, para a construção da implementação base, não existiu grande dificuldade. O nível de conhecimento para implementar a visualização dos pontos é relativamente simples, exigindo apenas um conhecimento básico de *javascript* e alguma pesquisa relativamente à *framework*. A nota atribuída a este critério é 4, sendo "Fácil" a aprendizagem essencial da *framework*.

A documentação presente no site oficial está muito bem distribuída por categorias. A informação contida na documentação é apresentada de forma muito explícita, com descrições detalhadas das múltiplas funcionalidades e os seus exemplos de implementação. A nota atribuída a este critério é 5, estando dividida em 3 para o primeiro plano, sendo a documentação "Muito Explícita", e 2 para o segundo plano, sendo a informação apresentada na documentação "Pouco Complexa".

No que diz respeito à comunidade, segundo o gráfico fornecido pelo fórum Stackoverflow, é possível verificar um aumento da atividade de utilizadores desde 2012. Apesar de algumas oscilações a tendência geral é de crescimento (consultar Figura 6).

Figura 6- Gráfico demonstrativo das pesquisas feitas sobre a tag "leaflet" [21].

A *framework Leaflet* é totalmente grátis.

4.4.2 Mapbox GL JS

A respeito da compatibilidade a tabela II mostra os browsers nos quais a implementação de teste funcionou.

Tabela II - Tabela de compatibilidades da *framework Mapbox GL JS.*

	Mozilla Firefox	Google Chrome	Microsoft Edge
Mapbox GL JS	Compatível	Não Compatível	Compatível

A incompatibilidade com o *Google Chrome* deve-se ao facto de, no computador onde foi testado, não ser capaz de utilizar *WebGL*.
Relativamente ao desempenho, o carregamento da página web é "rápido". Desta vez devido à incompatibilidade foi testado no *Microsoft Edge* através da mesma extensão. Segundo o relatório fornecido pelo "Lighthouse" demorou 3.9 segundos.
Em relação à extensibilidade, para a construção da implementação base, não existiu muita dificuldade. O nível de conhecimento para implementar a visualização dos pontos é relativamente simples, exigindo apenas um conhecimento básico de *javascript* e alguma pesquisa relativamente à *framework*. A nota atribuída a este critério é 3, sendo "Normal" a aprendizagem base da *framework*.
A documentação presente no site oficial está muito bem distribuída por categorias. A informação contida na documentação é apresentada de forma muito explícita, com descrições detalhadas das múltiplas funcionalidades e com um *dropdown* onde são apresentados os parâmetros das funções, o resultado e um exemplo de implementação. A nota atribuída a este critério é 4, estando dividida em 2 para o primeiro plano, sendo

a documentação "Explícita", e 2 para o segundo plano, sendo a informação apresentada na documentação "Pouco Complexa".

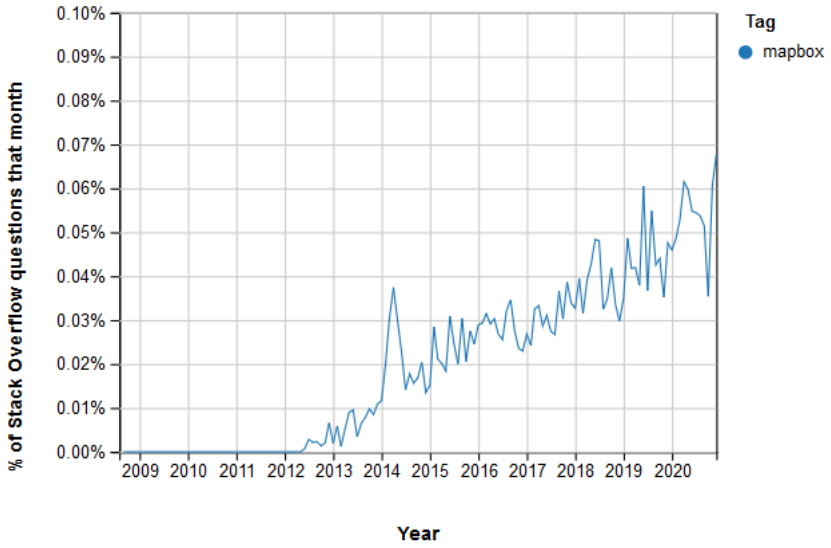


Figura 7 - Gráfico demonstrativo das pesquisas feitas sobre a tag "mapbox" [21].

No que diz respeito à comunidade, uma vez que a biblioteca e o seu uso estão diretamente relacionados com o uso das *frameworks* desenvolvidas pela Mapbox, considera-se o gráfico fornecido pelo fórum Stackoverflow. É possível verificar um aumento da atividade de utilizadores desde 2012. Apesar de algumas oscilações a tendência geral é de crescimento (ver Figura 7).
A *framework Mapbox GLJS* não é gratuita. Apesar da biblioteca ser referida como *open-source* em várias fontes, é possível ver o custo na página oficial. Para efeitos de teste permite uma quantidade substancial de utilizações de forma gratuita.

4.4.3 OpenLayers

Ao contrário das restantes *frameworks* o servidor local usado para carregar a página de Openlayers é à base de *nodeJS*. Tendo em conta o que foi anteriormente referido o tempo de carregamento da página web é "rápido". O relatório do "Lighthouse" devolve como tempo de carregamento 5.5 segundos.
A respeito da compatibilidade, a tabela III mostra os browsers nos quais a implementação teste funcionou.

Tabela III - Tabela de compatibilidades da *framework OpenLayers*

	Mozilla Firefox	Google Chrome	Microsoft Edge
OpenLayers	Compatível	Compatível	Compatível

Em relação à extensibilidade, para a construção da implementação base, conforme anteriormente referido, para além dos típicos ficheiros de *js* e *html*, também é necessário implementar um servidor local com base em node *js*, no qual é utilizado o método "parcel". O nível de conhecimento para implementar a visualização dos pontos

é relativamente simples, exigindo apenas um conhecimento básico de *js* e *html.* No entanto para inicializar o servidor e, posteriormente, carregar os dados exige uma pesquisa considerável para perceber o funcionamento. A nota atribuída a este critério é 2, sendo "Difícil" a aprendizagem base da *framework*.

A documentação presente no site oficial está bem distribuída por categorias. A informação contida na documentação é apresentada de forma explícita. Apesar da documentação ser essencialmente à base de exemplo das várias funcionalidades que existem (de forma semelhante às anteriores), no que diz respeito aos métodos para instalar, importar e criar o servidor local não existe muita informação. A nota atribuída a este critério é 3, estando dividida em 2 para o primeiro plano, sendo a documentação "Explícita", e 1 para o segundo plano, sendo a informação apresentada na documentação, considerando ambas as partes da implementação, "Muito Complexa".

No que diz respeito à comunidade, segundo o gráfico fornecido pelo fórum Stackoverflow para o uso da *tag* "openlayers", é possível verificar um aumento da atividade de utilizadores desde 2009. Sendo umas das *frameworks* mais crescida e consolidada, tem vindo a manter a sua taxa de utilização ao longo dos anos, ver Figura 8.

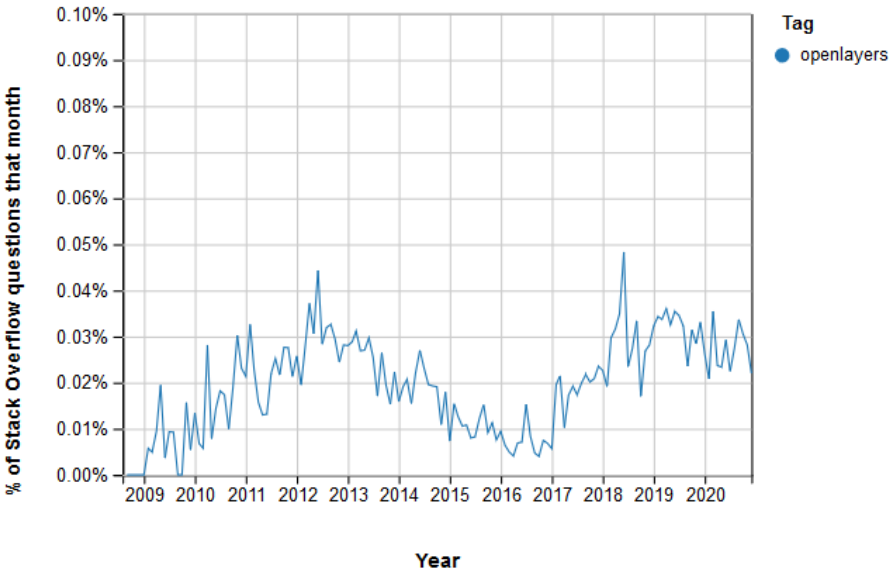


Figura 8 - Gráfico demonstrativo das pesquisas feitas sobre a *tag* "openlayers" [21].

Assim como o Leafleat também é uma *framework* grátis.

4.5 Discussão dos Resultados

Com base nos resultados apresentados na secção anterior para cada uma das *frameworks* testadas, para efeito de síntese foi construída a tabela IV. Assim é possível fazer uma comparação mais rápida entre elas e de forma menos exaustiva.

Tabela IV - Tabela síntese dos resultados de avaliação das *frameworks*.

	Leafleat	Mapbox GL	Openlayers
Desempenho (s)	26.2	3.9	5.5
Compatibilidade (%)	100	66.6	100
Extensibilidade	4	3	2
Documentação	5 (3 + 2)	4 (2 + 2)	3 (2 + 1)
Comunidade	Crescente	Crescente	Estável
Preçário	Não aplicável	Aplicável	Não aplicável

Com base nas informações presentes na tabela IV a *framework* OpenLayers apresenta um melhor desempenho. A nível de compatibilidade está equiparada com as restantes, tendo em conta os browsers utilizados. Comparativamente com as restantes, Openlayers, apresenta uma curva de aprendizagem maior que as restantes *frameworks*, o que se deve ao processo de inicialização conforme explicado na respetiva secção. Relativamente à documentação, entre as três, Leaflet apresenta a melhor documentação, é mais explícita e de mais fácil compreensão. A nível da comunidade apenas Openlayers não apresenta crescimento. Das três a única que tem custo associados é Mapbox GL JS.

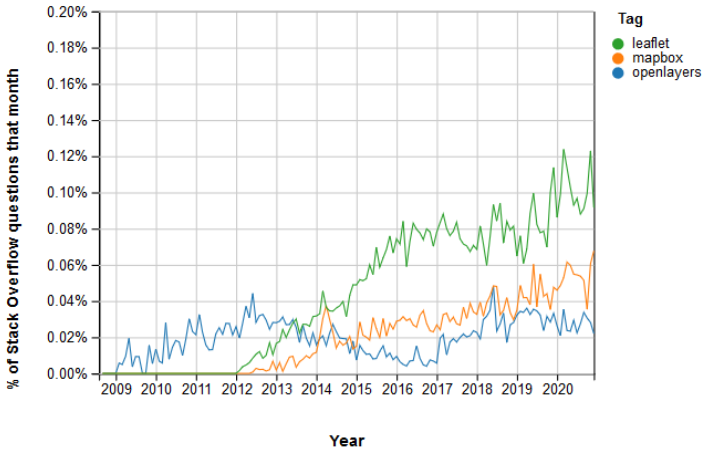


Figura 9 -Gráfico comparativo das pesquisas feitas sobre as *tags* "openlayers", "mapbox" e "leaflet"

Apesar de ambas *frameworks*, Leaflet e Mapbox, apresentarem um crescimento na comunidade, consultando a figura 9 podemos verificar que o Leaflet apresenta uma maior taxa de utilização entre as três. Apesar da taxa de uso do *Mapbox* ser crescente, esta é semelhante à do *OpenLayers*.

Comparando os resultados obtidos com outras fontes online como o site da Geoapify, podemos constatar que o *Leaflet* é a *framework* que "está mais na moda". A meu ver isto deve-se à facilidade de utilização e ausência de custo. No entanto, à vista dos

resultados não é uma ferramenta que lide muito bem com muitos dados, levando a tempos de carregamento elevados face às restantes. Apesar de este problema poder ser colmatado com métodos de *clustering,* por exemplo, nem sempre é pretendido fazê-lo. Tendo em conta estes fatores, para o problema em causa, não é a ferramenta mais adequada.

Relativamente às duas restantes ambas apresentam boa performance. Apesar de considerar o Mapbox Gl JS mais fácil de utilizar que o OpenLayers, ele tem custos associados. Um fator que leva a minha opinião a tender para o OpenLayers é o facto de este ser uma *framework* mais estável, e apesar de não apresentar crescimento, "sobreviveu" ao aparecimento de outras *frameworks*, não baixando a taxa de uso.

Capítulo 5
Dinamização do processo

5.1 Requisitos

A proposta de estágio inicial contemplava um conjunto de tarefas/requisitos para o projeto durante a fase de desenvolvimento. As propostas iniciais constituíam o objetivo ideal do projeto. No entanto, após os desenvolvimentos iniciais estes foram revistos de modo a conseguir tirar melhor partido do tempo de trabalho restante e garantir que os objetivos são atingidos.

As propostas resultantes dessa discussão foram:

• Gerar dinamicamente *layers* [2] de visualização a partir de meta-modelos de informação;

• Implementar a solução proposta para visualização de dados com contexto temporal

• Implementar a solução proposta para representação simultânea de indicadores co-localizados;

• Implementar uma ou mais das soluções propostas para visualização alternativas de informação georreferenciada;

Os restantes requisitos iniciais foram colocados como opcionais ou propostas de desenvolvimento futuro. Foram também adicionados outros que foram considerados úteis.

5.2 Conceptualização da solução

A solução encontrada para o problema de automatização do processo de consulta de informação georreferenciada passa pela criação de um serviço REST capaz de comunicar tanto com a base de dados como com o GeoServer. O utilizador interage com a interface, que por sua vez comunica com o serviço através dos *endpoints* disponibilizados para cada tarefa. A Figura 10 evidencia a sequência lógica da interação do utilizador com a interface web do módulo GeoIndicators

[2] *Layer*: Objeto gerado no Geoserver que contém a representação geográfica dos dados consultados da base de dados (neste caso). A consulta é feita segundo uma *query*.

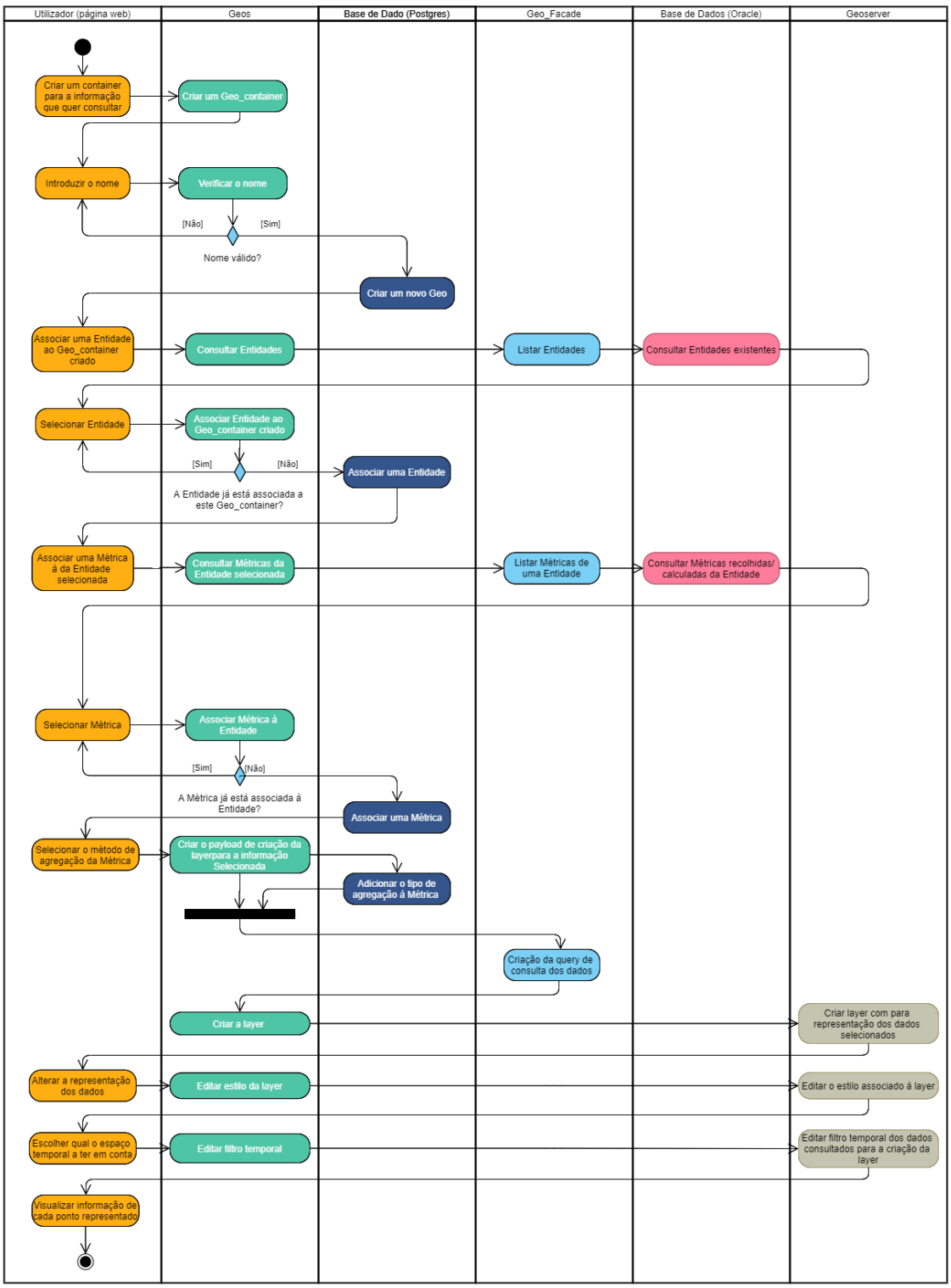


Figura 10- Diagrama de atividade representativo de uma interação de um utilizador com o sistema já com recurso aos serviços criados.

5.3 Faseamento da implementação

Dada a complexidade do problema foi decidido estabelecer metas entre as fases de desenvolvimento. Para isso recorreu-se a casos de uso, possuindo cada caso de uso uma complexidade superior ao anterior e exigindo que o anterior estivesse concluído para a sua viabilização.
Os casos de uso criados foram:

- Caso de uso 1: "Adicionar / Editar / Remover Métricas" - Compreendia a criação de uma interface que permitisse ao utilizador fazer a gestão da informação sobre a qual pretendia obter a representação geográfica.

- Caso de uso 2: "Representação de uma *layer* simples" - Comtempla a criação da *layer* de forma automática no GeoServer e consulta da *layer* por parte da interface. Inclui a agregação dos dados num valor único, criação da *view* para consulta.

- Caso de uso 3: "Consulta detalhada na informação num ponto" - Pressupõe a filtragem temporal dos dados, a agregação dos dados segundo o método adequado à métrica escolhida, várias opções de representação, consulta individual da informação no ponto representado através do clique no mapa, representação de múltiplas *layers* em simultâneo.

Capítulo 6
Funcionalidades da solução

A solução desenvolvida passa por um conjunto de componentes interligados, cada um com as suas funcionalidades e tarefas. Esta conta com diversos componentes como representação web, em Angular, com recurso à biblioteca OpenLayers, gestão e fornecimento de dados georreferenciados para representação, através do GeoServer, base de dados PostgresSQL para armazenar os dados da solução durante a sua utilização, e serviços para comunicação entre o GeoServer, a representação web e a base de dados que contém os dados.

6.1 Bases de dados

O Altaia possui uma base de dados com a informação recolhida e georreferenciada. Usa Oracle e apresenta uma estrutura em modo de estrela com diferentes níveis de granularidade.

Para o módulo GeoIndicators desenvolvido, foi decidido criar uma nova base de dados de forma a acomodar as necessidades da solução sem interferir com o sistema já existente. Existem então duas bases de dados, uma com os dados a consultar, a DBN1, e outra de suporte ao *frontend*, a BD aplicacional.

A BD aplicacional foi desenvolvida em PostgresSQL segundo a estrutura apresentada na Figura 11.

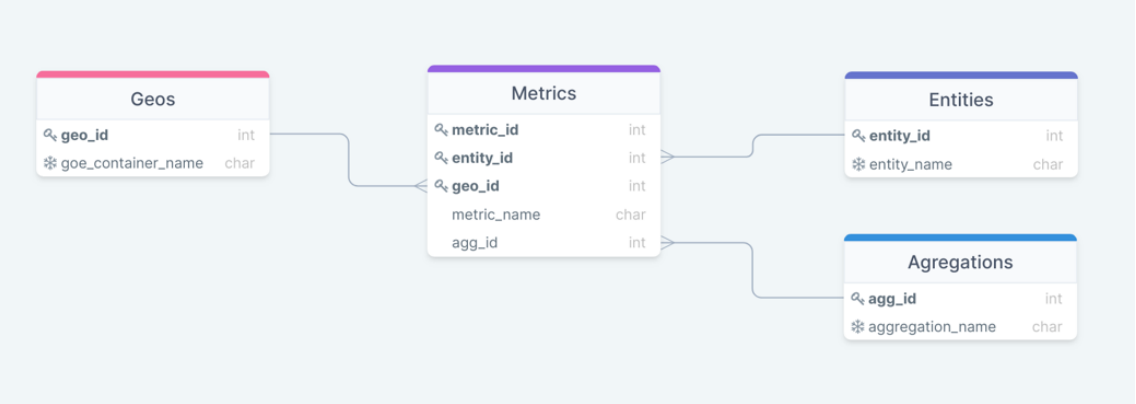


Figura 11 –Modelo relacional da base de dados PostgresSQL para o módulo GeoIndicators.

Esta estrutura permite que sejam criados múltiplos geo's[3], cada um com as métricas[4] e respetivos tipos de entidade[5] associados. Para cada métrica é também guardado o método de agregação a ser usado na criação da *layer* no GeoServer e consequentemente na sua representação.
A classe "Geos" irá conter todos os geo's que sejam criados. A classe "Entities" contém todos os tipos de entidade possíveis assim como a classe "Aggregations". Todas as possibilidades para ambas as classes estão disponíveis à partida, pois foi optado pela sua normalização.
A classe "Metrics" irá conter as associações de entidades e respetivas métricas com cada geo.
A divisão da informação por geos criados pelo utilizador vem disponibilizar a filtragem da informação personalizada dentro dos vários campos existentes, consultando apenas métricas específicas.
Esta estrutura é criada automaticamente ao inicializar o serviço "geos" pela primeira vez usando Quarkus com recurso a "Hibernate ORM *with* Panache".

6.2 Serviços

Nesta solução existem dois serviços, desenvolvidos em Quarkus, que suportam as comunicações entre os diversos componentes. O serviço "geos" é aquele que dá suporte a todas as interações que o utilizador pode ter com página web de representação dos dados comunicando com os componentes necessários às várias tarefas, e o serviço "geo_facade" funciona como suporte das operações do "geos".
No presente contexto quando é feito um pedido, a API transfere informação do estado do recurso ao *endpoint* utilizado, via HTTP, usando JSON como principal formato. O formato JSON é dos mais comuns devido à abstração relativamente à linguagem de programação e facilidade de leitura. Uma das vantagens da utilização desta arquitetura prende-se com as suas várias camadas, as quais permitem uma abstração relativamente aos componentes utilizados (bases de dados, API's, *web-clients*, etc.) e ao mesmo tempo, escalabilidade do serviço assim como fácil adição e/ou remoção de *endpoints*.

Geo_Facade

O serviço "geo_facade", faculta ao "geos" informações que este necessita para várias das suas funcionalidades. Este permite consultar as características de cada tipo de entidade, nomeadamente, os atributos que a identificam e as métricas existentes, bem como, dado um conjunto de métricas, obter uma *query* para consulta dos valores destas métricas na DBN1.

[3] Geo: Abreviação do termo Geo_container. Trata-se de um *container* com informação dos objetos a representar na *layer*
[4] Métrica: Conjunto de dados recolhidos/calculados com uma determinada periodicidade, relativos a um tipo de entidade.
[5] Tipo de entidade: Cada entidade representa o tipo de dados que estão a ser consultados, mais concretamente o tipo de rede de telecomunicação.

Geos

O serviço "geos", vem capacitar a automatização das tarefas anteriormente manuais de criação de *views* na base de dados e *layers* no GeoServer. Para além disso permite a gestão de informação relativamente aos dados visualizados, sendo capaz de criar, editar e/ou remover as representações geradas pelo utilizador.
O "geos" como serviço, apresenta uma estrutura com base em camadas. A "camada de aplicacional" consiste na interpretação dos pedidos feitos ao sistema através da interface descrita na secção 6.4. A "camada de negócio" é onde se encontra a parte lógica de todos os métodos criados, nomeadamente funções, para a resolução do problema. Na "camada de persistência" é feito o acesso à base de dados. Na "camada de transporte" é feita a comunicação com outros componentes, neste caso, serviços com recuso a API's REST disponibilizadas para o efeito.
A existência destas camadas permite uma maior abstração relativamente às tecnologias utilizadas. Havendo necessidade de alteração das ferramentas utilizadas, em teoria, o fluxo de dados mantém-se, sendo apenas necessário alterar a camada que interage com a ferramenta alterada e ajustar a sua parte referente à camada de negócio se esta não for compatível.

As transições entre camadas podem ser vistas no exemplo da Figura 12. Em casos em que seja necessário usar outros serviços, a camada de negócio é responsável por incluir na sua lógica a função referente ao pedido feito a esse serviço, e este por sua vez é realizado pela camada de transporte.

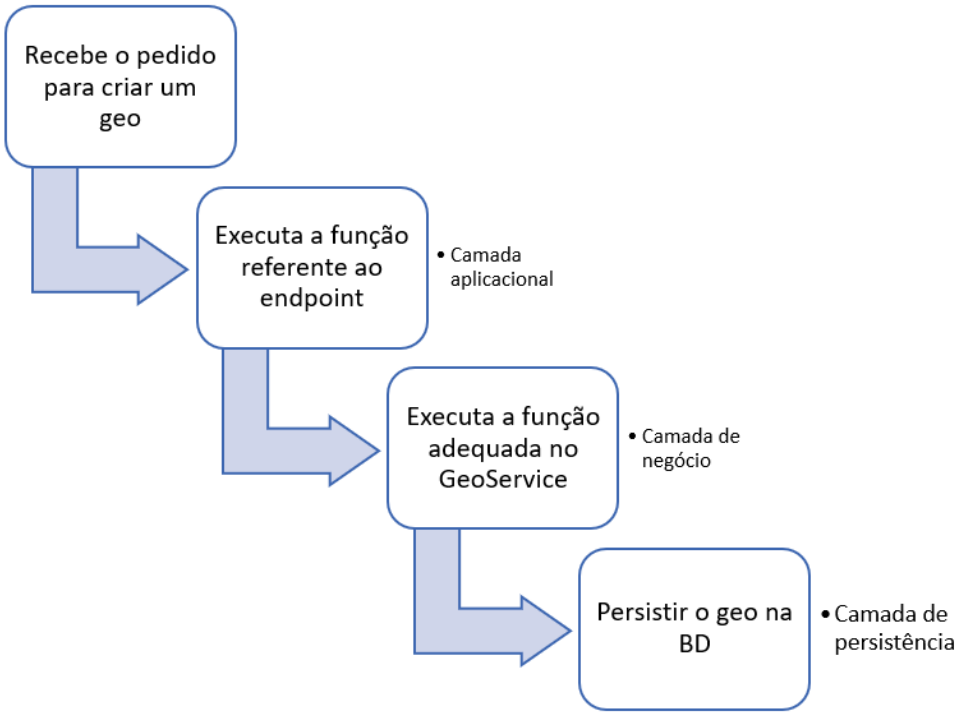


Figura 12 -Exemplo da transição entre camadas para um pedido como "criar Geo".

A camada aplicacional é responsável por relacionar o pedido recebido, com a camada de negócio, mapeando as funções associadas a cada *endpoint*. A Figura 13 mostra alguns dos *endpoints* presentes no "geos".

```java
@Path("/geo")
@RegisterProvider(LoggingFilter.class)
public class GeoResource {

    @Inject
    GeoService geoService;


    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Response getall() {
        return geoService.listall();
    }

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    public Response Creategeo(Geos geo){
        return geoService.createGeo(geo);

    }

    @GET
    @Path("/entity")
    @Produces(MediaType.APPLICATION_JSON)
    public Response getAllEntities() {
        return geoService.getAllEntities();
    }
```

Figura 13 – Exemplo de *endpoints* usados no serviço "geos".

Todas as tarefas são posteriormente geridas pela camada de negócio. Por cada chamada executada pela camada aplicacional, a camada de negócio é responsável por executar as funções para a tarefa. Essa gestão das funções a executar é feita na classe GeoService, conforme a Figura 14 (parte 1).

Algumas das tarefas são relativamente simples, pelo que o comando a executar é simples uma vez que as funções são por si só a resposta ao pedido recebido, comforme a Figura 14 (parte 2). Estas situações correspondem à camada de persistência. Esta inclui ainda a criação de *views* usando *statements* com a *query* gerada para o efeito. Um exemplo desta pode ser consultado no apêndice A.

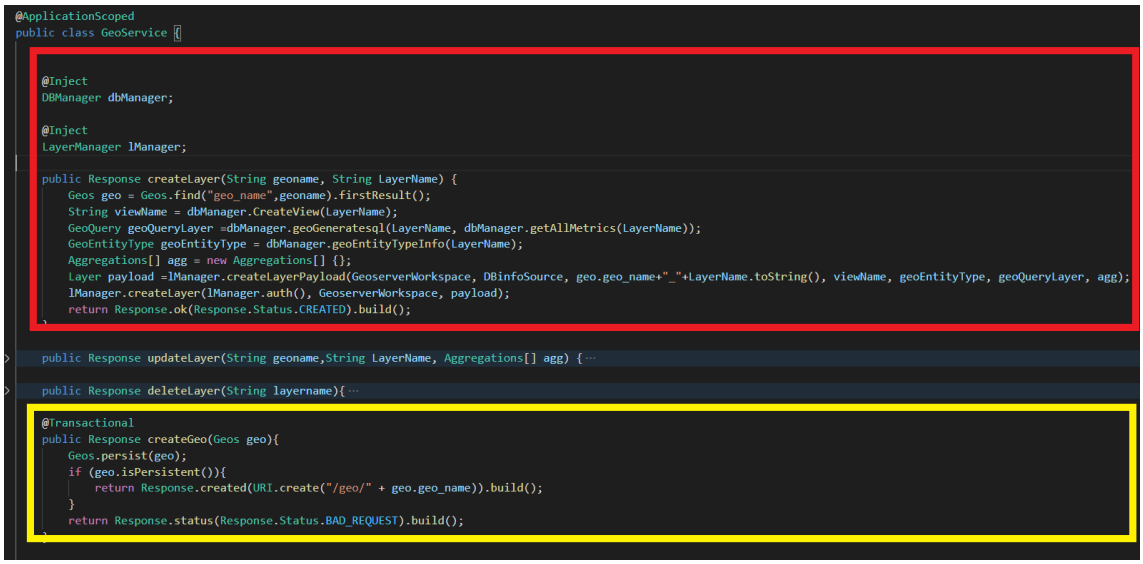


Figura 14 - Exemplo do código usado para gestão das tarefas a executar (1, a vermelho) e exemplo do código usado para a função de criação de um Geo (2, a amarelo), presentes na classe GeosService.

No entanto, para tarefas mais complexas a lógica está separada por componentes. Uma parte trata os dados obtidos da DBN1 e outra trata os dados relativamente ao GeoServer. Esta divisão acontece devido à necessidade de recursos externos ao serviço. Aqui entra em funcionamento a camada de transporte. Esta diz respeito a todas as comunicações feitas com outros serviços. Para estas comunicações serem possíveis é necessário que os componentes tenham uma API para o efeito. No caso do GeoServer este disponibiliza uma API REST para as suas funcionalidades. No caso de interação com a DBN1 foi concebido o "geo_facade".

Tratando-se de serviços o processo usado passou por importar as funções do serviço e atuar como um cliente. Para tal é necessário usar as bibliotecas "quarkus-rest-client" e "quarkus-rest-client-jackson". Estas permitem importar os *endpoints* que se pretendem utilizar e interagir com eles como um cliente através da sua identificação com a anotação "@RegisterRestClient", conforme a Figura 15.

Figura 15 - Exemplo do código utilizados para importar os *endpoints* do serviço "geo_facade" de modo a poder comunicar com o mesmo.

De modo a simplificar a descodificação da informação usam-se modelos semelhantes ao da Figura 16, para leitura da informação recebida, evitando assim erros na sua interpretação.

```java
package com.alticelabs.altaia.geos.core.models.layer.components;

import com.fasterxml.jackson.annotation.JsonPropertyOrder;

@JsonPropertyOrder({"name", "nativeName","namespace","title","keywords","nativeBoundingBox","srs","projectionPolicy","enabled","m
public class FeatureType {
    private String name;
    private String nativeName;
    private Namespace namespace;
    private String title ;
    private Keywords keywords;
    private NativeBoundingBox nativeBoundingBox;
    private String srs;
    private String projectionPolicy;
    private boolean enabled;
    private Metadata metadata;
    private boolean overridingServiceSRS;
    private Attributes attributes;

    public FeatureType(){
    }

    public FeatureType(String name,String nativeName, Namespace namespace,String title, Keywords keywords,NativeBoundingBox nativ
        this.name= name;
        this.nativeName = nativeName;
        this.namespace= namespace;
        this.title=title;
        this.keywords=keywords;
        this.nativeBoundingBox =nativeBoundingBox;
        this.srs= srs;
        this.projectionPolicy=projectionPolicy;
```

Figura 16 -Exemplo de um modelo de dados de uma "FeatureType" usada na criação do *payload* de uma *layer*. É usado para comunicar com o GeoServer.

Ao implementar um cliente REST é necessário indicar qual o endereço em que o serviço está alojado. Para simplificação do código usa-se a propriedade "configKey" e especifica-se o endereço nas propriedades da aplicação como mostra a Figura 17, mantendo assim o código independentemente do endereço usado.

```
view-db/mp-rest/url=http://localhost:9010/
```

Figura 17- Exemplo de como especificar o endereço de um cliente com recuso ao parâmetro "configKey".

Depois de criada a interface com os *endpoints* do serviço, é necessário importar o serviço como mostra a Figura 18 usando as anotações "Inject" e "RestClient".

```java
    @Inject
    @RestClient
    GeoDBViewServices geoDBViewServices;
```

Figura 18 - Exemplo do código utilizado para importar um cliente REST.

A comunicação feita com o GeoServer é em todo semelhante à feita com o serviço "geo_facade" uma vez que este disponibiliza um serviço REST para a sua configuração, sem necessidade de aceder à página web do mesmo.

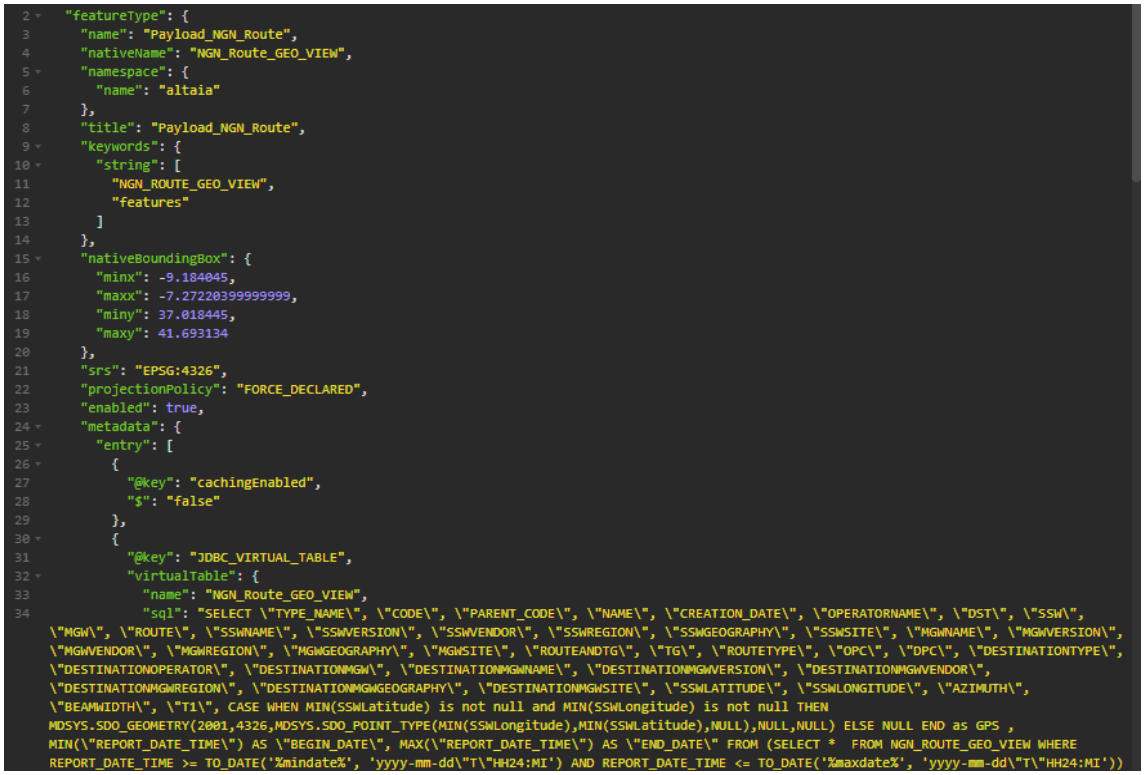
No que diz respeito à gestão de informação, tratando-se de um serviço REST pressupõe-se que nenhuma informação do cliente é armazenada entre pedidos, pelo que é necessário criar uma estrutura capaz de guardar as consultas efetuadas, na BD aplicacional descrita na secção 6.1.
Usando Quarkus com recurso a bibliotecas como "quarkus-hibernate-orm-panache" e "quarkus-jdbc-postgresql" é possível desenhar uma estrutura de base de dados através do código desenvolvido. Como demostrado na Figura 19 é possível usar a anotações de várias formas, seja para estabelecer o nome da tabela ou de um atributo ou para definir regras. É possível ainda estabelecer as relações entre tabelas podendo criar assim os modelos relacionais.

```
@Entity
@Table(name ="geos")
public class Geos extends PanacheEntityBase{

    //>>>>>>>>>>>>>>> Constructor >>>>>>>>>>>>>>>>//

    public Geos(){

    }
    public Geos(String geoname) {
        this.geo_name= geoname;
    }

    //>>>>>>>>>>>>>>>> Atributes >>>>>>>>>>>>>>>>//

    @Id
    @Column(name ="geo_id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer geo_ID;

    @Column(name = "geo_name", nullable = false, unique = true)
    public String geo_name;
```

Figura 19- Exemplo do código utilizado para a criação da tabela "Geos" na base de dados, recorrendo a anotações JPA utilizadas pela "quarkus-hibernate-orm-panache".

6.3 GeoServer

O GeoServer possui uma interface gráfica, a partir da qual é possível configurar e atualizar a informação das *layers*, assim como dos *datasources* utilizados. Uma vez que existem várias *datasources*, nem todas estão disponíveis por omissão.
Uma vez que o GeoServer apresenta por omissão alguns exemplos, começou-se por configurar um *workspace* novo, ao qual se atribuiu o nome "altaia". Posteriormente dentro do mesmo foi criado o *store* relativo á "DBN1".
Neste caso, foi necessário incluir uma extensão de modo a garantir a compatibilidade com a "DBN1". Sendo esta uma base de dados em Oracle, foi necessário instalar a respetiva extensão e configurar a conexão. Para além disso, foi necessário configurar o ficheiro "context.xml" uma vez que o GeoServer foi instalado através do Tomcat.
Após a configuração inicial pretende-se que não sejam necessárias mais alterações de forma direta. A partir deste ponto, no contexto da solução, o GeoServer funciona como repositório de *layers*. A criação destas *layers* é feita pelo serviço "geos", que interage com o GeoServer via REST sem necessidade de configuração manual. O "geos" cria um payload com a informação necessária à criação da layer, enviado através de um pedido REST, via HTTP, seguindo um conjunto de regras. O mesmo acontece para a edição e remoção das *layers*. A Figura 20 mostra parte do *payload* utilizado.



```
"featureType": {
  "name": "Payload_NGN_Route",
  "nativeName": "NGN_Route_GEO_VIEW",
  "namespace": {
    "name": "altaia"
  },
  "title": "Payload_NGN_Route",
  "keywords": {
    "string": [
      "NGN_ROUTE_GEO_VIEW",
      "features"
    ]
  },
  "nativeBoundingBox": {
    "minx": -9.184045,
    "maxx": -7.27220399999999,
    "miny": 37.018445,
    "maxy": 41.693134
  },
  "srs": "EPSG:4326",
  "projectionPolicy": "FORCE_DECLARED",
  "enabled": true,
  "metadata": {
    "entry": [
      {
        "@key": "cachingEnabled",
        "$": "false"
      },
      {
        "@key": "JDBC_VIRTUAL_TABLE",
        "virtualTable": {
          "name": "NGN_Route_GEO_VIEW",
          "sql": "SELECT \"TYPE_NAME\", \"CODE\", \"PARENT_CODE\", \"NAME\", \"CREATION_DATE\", \"OPERATORNAME\", \"DST\", \"SSW\",
\"MGW\", \"ROUTE\", \"SSWNAME\", \"SSWVERSION\", \"SSWVENDOR\", \"SSWREGION\", \"SSWGEOGRAPHY\", \"SSWSITE\", \"MGWNAME\", \"MGWVERSION\",
\"MGWVENDOR\", \"MGWREGION\", \"MGWGEOGRAPHY\", \"MGWSITE\", \"ROUTEANDTG\", \"TG\", \"ROUTETYPE\", \"OPC\", \"DPC\", \"DESTINATIONTYPE\",
\"DESTINATIONOPERATOR\", \"DESTINATIONMGW\", \"DESTINATIONMGWNAME\", \"DESTINATIONMGWVERSION\", \"DESTINATIONMGWVENDOR\",
\"DESTINATIONMGWREGION\", \"DESTINATIONMGWGEOGRAPHY\", \"DESTINATIONMGWSITE\", \"SSWLATITUDE\", \"SSWLONGITUDE\", \"AZIMUTH\",
\"BEAMWIDTH\", \"T1\", CASE WHEN MIN(SSWLatitude) is not null and MIN(SSWLongitude) is not null THEN
MDSYS.SDO_GEOMETRY(2001,4326,MDSYS.SDO_POINT_TYPE(MIN(SSWLongitude),MIN(SSWLatitude),NULL),NULL,NULL) ELSE NULL END as GPS ,
MIN(\"REPORT_DATE_TIME\") AS \"BEGIN_DATE\", MAX(\"REPORT_DATE_TIME\") AS \"END_DATE\" FROM (SELECT *  FROM NGN_ROUTE_GEO_VIEW WHERE
REPORT_DATE_TIME >= TO_DATE('%mindate%', 'yyyy-mm-dd\"T\"HH24:MI') AND REPORT_DATE_TIME <= TO_DATE('%maxdate%', 'yyyy-mm-dd\"T\"HH24:MI'))
```


Figura 20 – Exemplo parcial do *payload* utilizado para a criação de uma *layer* no GeoServer.

Na Figura 20, estão presentes apenas alguns dos parâmetros configurados. Destes existem alguns essenciais, indispensáveis à criação correta da layer, e uns complementares, que podem ser omitidos, no entanto, caso sejam omitidos aumentaria o tempo necessário ao processo de criação da *layer*. Dentro dos essenciais, estão, com todos os parâmetros que os constituem, "name", "nativeName", "namespace", "title", "keywords", "srs", "metadata" com as "entry"' evidenciadas na Figura 20 e com a "store". O *payload* completo pode ser consultado no apêndice D.

Na última linha da Figura 20, é possível perceber que a *query* utilizada contém parâmetros variáveis. Para a utilização de variáveis é necessário fazer a sua declaração, assim como a apresentação de um valor *default,* conforme a Figura 21.

```
"parameter": [
  {
    "name": "mindate",
    "defaultValue": "2021-07-16T09:45"
  },
  {
    "name": "maxdate",
    "defaultValue": "2021-07-20T09:45"
  }
]
```


Figura 21- Parte do *paylod* de criação de uma *layer* referente à declaração de parâmetros variáveis.

Depois de criada a *layer*, esta fica disponível para consultada na página web com recurso ao OpenLayers.

6.4 Angular

A utilização de Angular como ferramenta de criação da página web deveu-se essencialmente a dois fatores. O primeiro associa-se com o facto de o Altaia já usar Angular, pelo que fez todo o sentido usar a mesma ferramenta para desenvolver um módulo no qual este pode ser adicionado. Mesmo que depois, a nível de interface, tenha de sofrer alterações, já não existe a necessidade de uma conversão entre ferramentas. O segundo fator foi a inexperiência na área de desenvolvimento web, pelo que o facto de ter na equipa de trabalho membros com experiência na utilização de Angular permitiu uma aprendizagem muito mais rápida da ferramenta.
A página web desenvolvida contempla duas partes essenciais: o menu e o mapa. O menu permite ao utilizador escolher entre as várias opções disponíveis o que pretende fazer, sendo que as opções dadas ao utilizador visam responder aos requisitos iniciais. O mapa vem responder às opções escolhidas no menu, mostrando ao utilizador os dados georreferenciados das opções que este escolheu, como mostra o exemplo da figura 22.

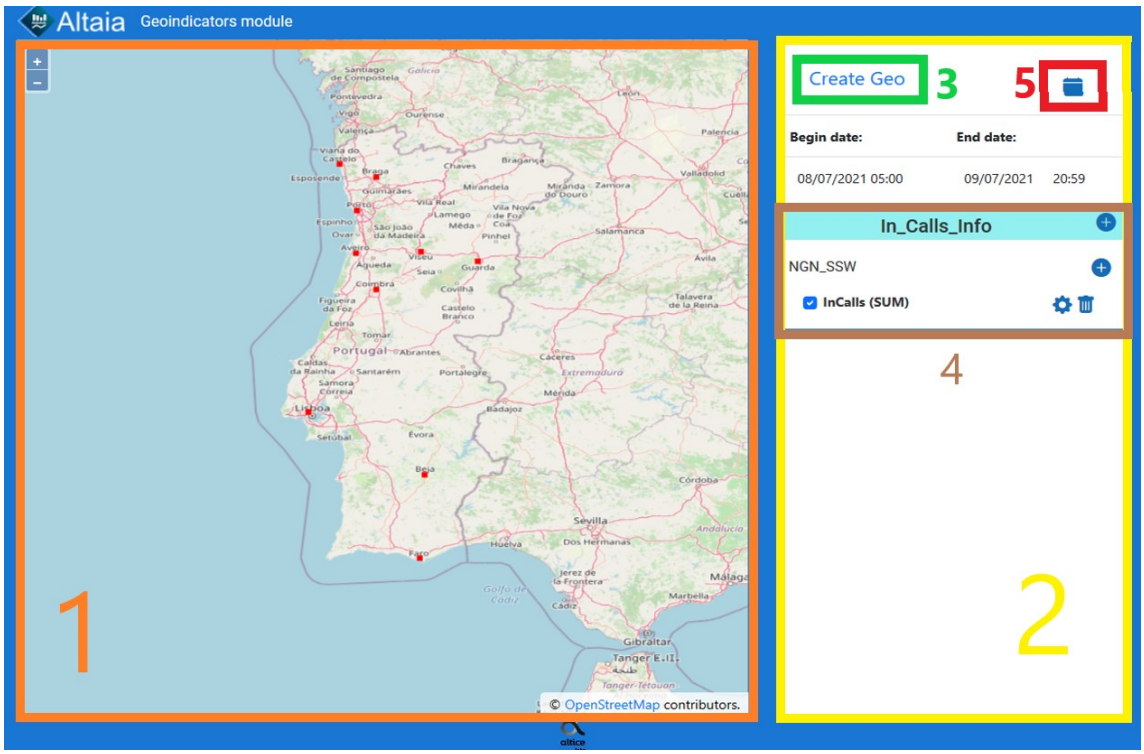


Figura 22 - Imagem demonstrativa da representação de uma métrica selecionada no menu. O mapa corresponde à parte 1, a laranja, o menu corresponde à parte 2, a amarelo. Dentro do menu está o botão para a adição de "Geo's", parte 3 a verde, o submenu para a associação de entidades tipo, métricas e configuração das representações no mapa, parte 4 a castanho, e o botão para escolher as datas (filtro temporal), parte 5 a vermelho.

O menu permite ao utilizador criar um geo, figura 22 parte 3, ao qual pode associar entidades tipo e, por sua vez, métricas (Figura 22 parte 4). Durante a associação das métricas permite escolher qual o tipo de agregação da métrica que escolheu. Todo o menu está diretamente ligado à BD aplicacional, pelo que todas as alterações efetuadas no menu são persistidas na base de dados, através do serviço "geos", guardando assim todas as consultas criadas.

Depois de associada permite a representação da mesma no mapa com recurso ao OpenLayers. Este, para cada métrica selecionada cria uma Tile[6], a qual contém a sua informação, e representa-a no mapa. Permite, além disso, a alteração da representação escolhida. Em qualquer altura o utilizador pode definir o espaço temporal da informação que está a consultar recriando assim a tile representada (Figura 22 parte 5).

A Figura 23 mostra o código utilizado para a consulta das *layers* no GeoServer e criação das respetivas tiles. É possível observar que é utilizado "VIEWPARAMS", permitindo assim substituir parâmetros da *query* usada na criação da *layer* no GeoServer usando as variáveis em detrimento dos valores *default* apresentados anteriormente na Figura 21. A função para criação da tile pertence ao OpenLayers.

```
generate_layers(value: Array<TileLayer_conf>) {
  this.layers = [this.ground_map];

  value.forEach(element => {
    const new_tile = new TileWMS({
      url: '/geoserver/altaia/wms',
      params: {
        'LAYERS': 'altaia:' + element.layer,
        'VERSION':'1.1.0',
        'TILED': true,
        'STYLES': element.style,
        'VIEWPARAMS': "mindate:" + element.mindate + ";maxdate:" + element.maxdate,
      },
      serverType: 'geoserver',
      crossOrigin: 'anonymous',
    });
    this.layers.push(new TileLayer({ source: new_tile, name: element.layer + ":" + element.metric, visible: element.visible }));

  });

  this.map.setTarget(null)
  this.map = new Map({
    layers: this.layers,
    view: this.view,
    target: 'ol-map'
  });
}
```

Figura 23 – Exemplo do código utilizado para criação de tiles no OpenLayers.

Após representação da informação das métricas no mapa, através de pontos, é possível ao clicar em qualquer um deles e obter informação detalhada da métrica nesse mesmo ponto. A informação apresentada é agrupada por tipo de entidade para todos os pontos representados no mapa, no momento da consulta, permitindo assim a consulta de várias métricas em simultâneo.
A Figura 24 mostra parte do código utilizado para a recolha dessa informação. Embora o OpenLayer disponibilize a consulta de toda a informação das suas *layers* para a coordenada selecionada, a parte de tratamento desses dados, tem de ser feita posteriormente, uma vez que se trata de dados em bruto (JSON). Para o formato de apresentação sugerido é necessário agrupar por geo e posteriormente por entidade, mantendo apenas alguns dos parâmetros considerados essenciais para identificação.

[6] *Tile*: Camada criada no OpenLayers para representar os dados de cada métrica. Cada métrica associada a um geo possuí uma *tile*. No OpenLayers uma *layer* é constituída por uma ou mais *tiles*. (diferente do conceito de *layer* no geoserver).

```
getCoord(event: any) {

  let observables = [];
  var clicked_metrics = [];
  var clicked_geo = [];
  var featurelist = [];
  var coordinates = this.map.getEventCoordinate(event);

  var view = this.map.getView();
  var viewResolution = view.getResolution();


  for (let i = 1; i < this.layers.length; i++) {
    const element = this.layers[i];
    if (element.get('visible')) {
     var source = element.getSource()
      console.log(source);

      var url = source.getFeatureInfoUrl(
        coordinates, viewResolution, view.getProjection(),
        { 'INFO_FORMAT': 'application/json', 'FEATURE_COUNT': 40, 'QUERY_LAYERS':source.params_.LAYERS});
      if (url) {
        observables.push(this.http.get<FeatureCollection>(url));
        clicked_metrics = clicked_metrics.concat(String(this.TileLayers[i - 1].metric).toUpperCase())
        clicked_geo = clicked_geo.concat(String(this.TileLayers[i - 1].geo))
      }
    }
  }
  var allfeatures= []

  forkJoin(observables).toPromise().then(data=> {
    console.log(data)
    this.Compileinfo(clicked_metrics, data, clicked_geo)

    clicked_metrics = [].concat([])
    clicked_geo = [].concat([])
    featurelist = [].concat([])
  });

}
```

Figura 24 – Parte do código utilizado para consulta dos dados apresentados após seleção de um ponto no mapa.

6.5 Processo de Inicialização

O processo de inicialização do módulo GeoIndicators é feito por vários passos sobre o pressuposto que está ligado à base de dados DBN1 do Altaia e que o serviço GeoServer está instalado e a correr através do recurso ao Tomcat. Todo este processo é feito localmente.

O primeiro passo é iniciar o serviço “geo_facade” através do comando “./mvnw compile quarkus:dev” executado na pasta do serviço “geo_facade”.

O segundo passo é iniciar o serviço “geos” através do mesmo comando “./mvnw compile quarkus:dev” mas na pasta do serviço “geos”. Ao ser executado este inicializa a base de dados PostgreSQL com recursos a um Docker.

O terceiro e último passo consiste em iniciar a página do GeoIndicators através do comando “npm start” na pasta do webservice que por sua vez executa o comando “ng serve --proxy-config proxy.config.json” presente no ficheiro “package.json”.

Capítulo 7
Testes

Após o desenvolvimento da solução é necessário fazer um conjunto de testes para averiguar se esta responde a todos os requisitos compilados na fase inicial. Deste modo foi criado um conjunto de situações que a solução teria de enfrentar e ser capaz de solucionar.

User Stories

1. Como utilizador, posso criar um geo para introduzir objetos a representar no mapa com um nome à minha escolha de modo a organizar melhor a informação que estou a consultar.
2. Como utilizador, posso selecionar entidades tipo para associar ao meu geo, de modo a saber qual o tipo de informação que estou a consultar.
3. Como utilizador, posso selecionar as métricas a associar às entidades tipo do meu geo para as poder visualizar, de modo a conseguir escolher apenas informação relevante para consulta.
4. Como utilizador, posso escolher quais as métricas a visualizar em simultâneo de modo a perceber a sua distribuição geográfica.
5. Como utilizador, posso escolher o filtro de tempo para localizar temporalmente a informação representada.
6. Como utilizador, posso escolher um estilo de representação para diferenciar a representação no mapa
7. Como utilizador, posso consultar a informação de um ponto representado no mapa para obter uma informação detalhada sobre o mesmo.

Para cada User Story, foi então desenvolvido um guião de testes, de modo a garantir que caso exista algum erro, este esteja relacionado com a funcionalidade e não com a interface apresentada, uma vez que esta tem apenas como intuito permitir visualizar as funcionalidades em funcionamento.
Para todos os testes efetuados, assume-se que todos os serviços estão em execução, nomeadamente o "geos", o "geo_facade" e o "GeoServer". Para além disso para cenário de testes é necessária a conexão ao ambiente de desenvolvimento e à base de dados "DBN1" na qual estão armazenados os dados de teste, conforme o processo de inicialização descrito na secção 6.5.

Teste 1

- Ator: Utilizador
- Objetivo: Criar um geo
- Pré-condições:
  - Estar na página “localhost:4200”
- Pós-condições:
  - Ter um geo criado
- Cenário Principal:
  1. Abrir o endereço “localhost:4200”
  2. Clicar no botão “Add Geo”
  3. Introduzir o nome para o Geo
  4. Clicar em “Save”
- Cenário Alternativo:

  4a. Já existe um geo com esse nome

  4a1. Introduzir um novo nome

O resultado do teste 1 pode ser visto na Figura 25. Os restantes passos podem ser consultados no apêndice E, constando todos os passos pela ordem cronológica descrita no cenário principal assim como os possíveis erros.

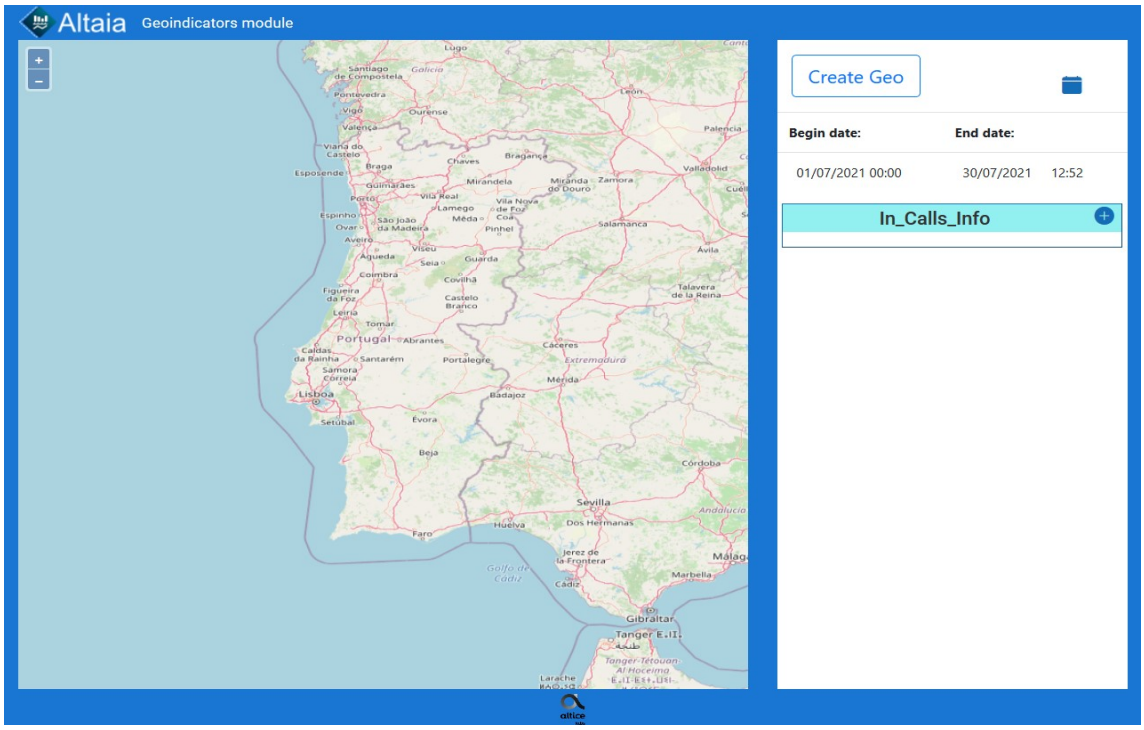


Figura 25 -Resultado final do teste 1.

Teste 2

- Ator: Utilizador
- Objetivo: Associar um tipo de entidade a um geo
- Pré-condições:
  - Estar na página "localhost:4200"
  - Ter um geo criado
- Pós-condições:
  - Ter um tipo de entidade associado ao geo
- Cenário Principal:
  1. Clicar no botão "+" ao lado do nome do geo
  2. Clicar na *droplist*
  3. Selecionar um tipo de entidade
  4. Clicar em "Save"
- Cenário Alternativo:
  - 4a. O tipo de entidade selecionada já está associado ao geo
    - 4a1. Prosseguir para o passo seguinte (Ex.: Associar métrica)

O resultado do teste 2 pode ser visto na Figura 26. Os restantes passos podem ser consultados no apêndice F, constando todos os passos pela ordem cronológica descrita no cenário principal.

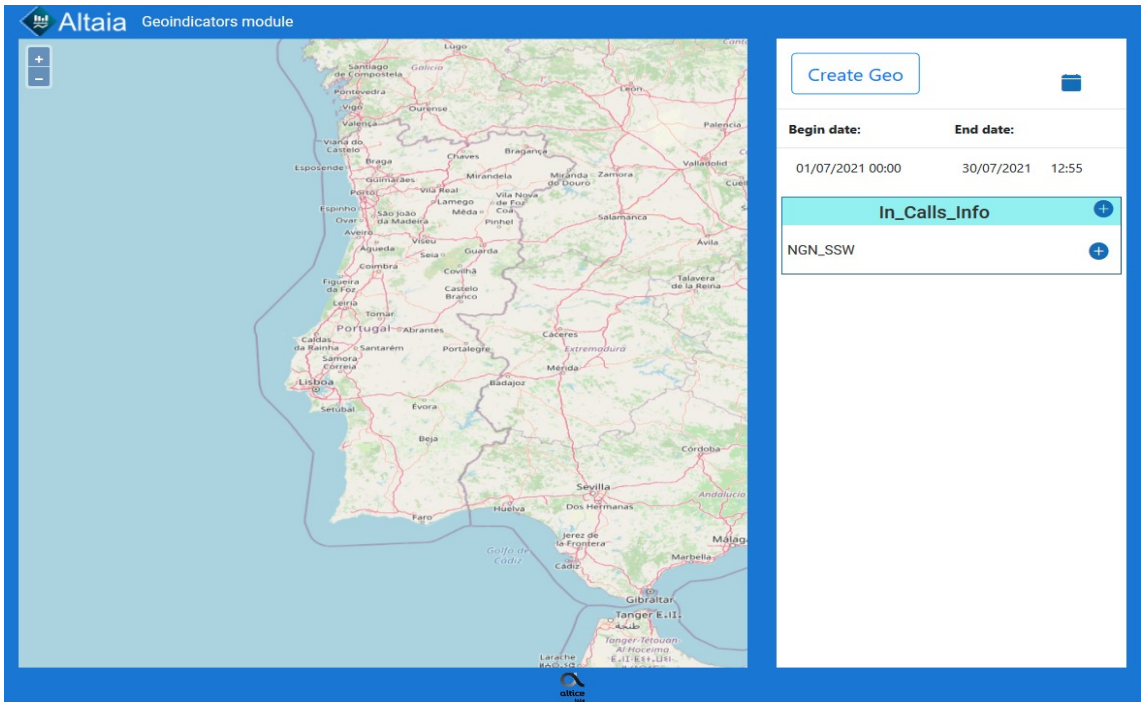


Figura 26 - Resultado final do teste 2.

Teste 3

- Ator: Utilizador
- Objetivo: Associar uma métrica ao geo
- Pré-condições:
  - Estar na página “localhost:4200”
  - Ter um geo criado
  - Ter o tipo de entidade à qual a métrica pertence associado
- Pós-condições:
  - Ter uma métrica associada
- Cenário Principal:
  1. Clicar no botão “+” ao lado do nome do tipo de entidade.
  2. Clicar na *droplist*
  3. Selecionar uma métrica
  4. Clicar na aba “Agregation”
  5. Clicar na droplist
  6. Selecionar uma agregação.
  7. Clicar em “Save”
- Cenário Alternativo:
  - 3a. Pretende associar mais que uma métrica
    - 3a1. Repetir o processo para cada métrica
  - 7a. A métricas selecionada já está associada
    - 7a1. Selecionar outra métrica

O resultado do teste 3 pode ser visto na Figura 27. Os restantes passos podem ser consultados no apêndice G, constando todos os passos pela ordem cronológica descrita no cenário principal. Independentemente do tipo de agregação, uma métrica só pode estar associada uma vez a cada geo.

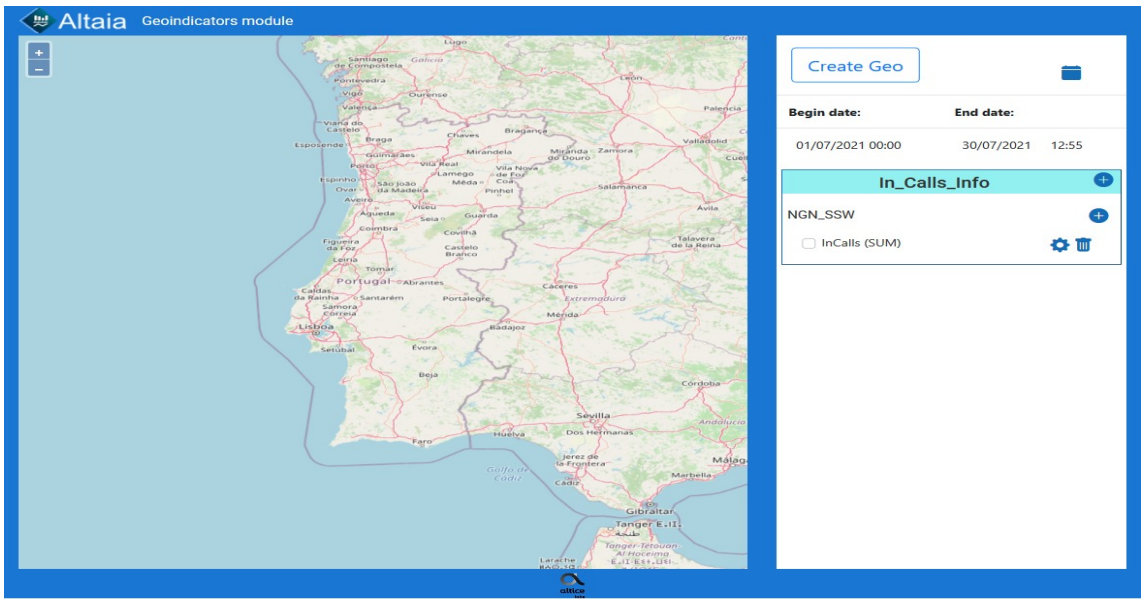


Figura 27- Resultado final de teste 3.

Teste 4

- Ator: Utilizador
- Objetivo: Localizar no tempo a visualização pretendida
- Pré-condições:
  - Estar na página "localhost:4200"
  - Ter um geo criado
  - Ter o tipo de entidade à qual a métrica pertence associado
  - Ter a métrica associada
- Pós-condições:
  - Os dados da representação são filtrados no tempo
- Cenário Principal:
  1. Clicar no botão do calendário ao lado do botão para criar um geo
  2. Selecionar uma data de início e uma data de fim
  3. Clicar em "Save"
- Cenário Alternativo:

  4a. Não selecionou nenhuma data

  4a1. Por omissão, assume-se como data de início o dia 1 do mês atual e como data de fim o dia atual.

O resultado do teste 4 pode ser visto na Figura 28. Os restantes passos podem ser consultados no apêndice H, constando todos os passos pela ordem cronológica descrita no cenário principal.

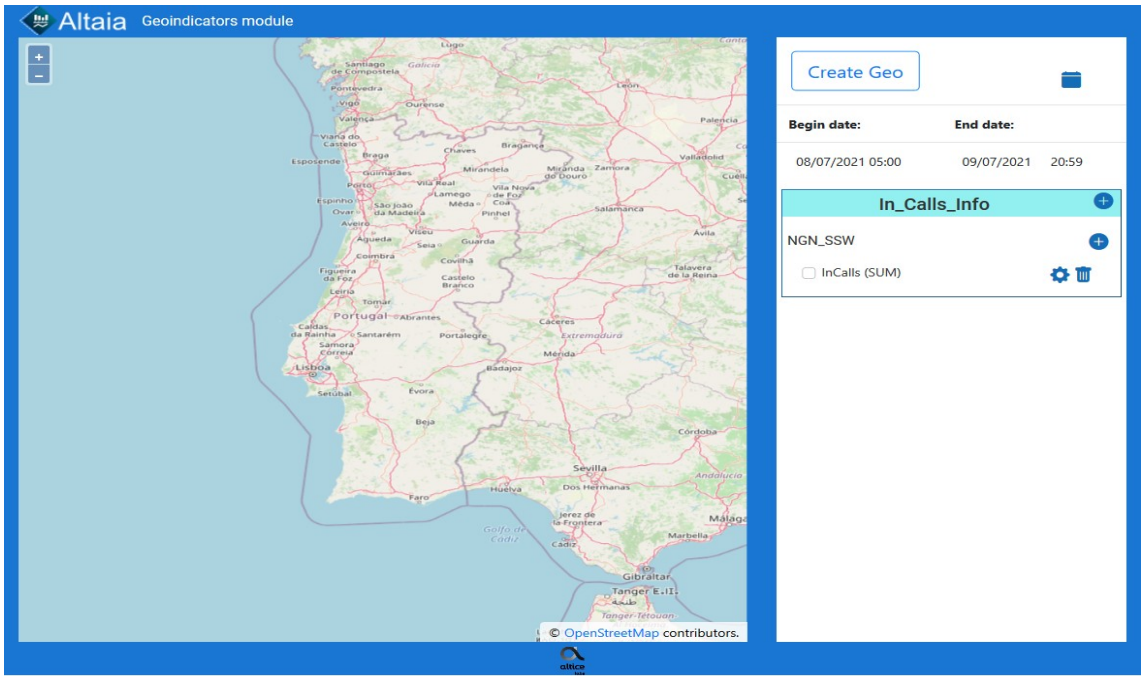


Figura 28 -Resultado final do teste 4.

Teste 5

- Ator: Utilizador
- Objetivo: Visualizar a representação de uma métrica e alterar o estilo
- Pré-condições:
  - Estar na página "localhost:4200"
  - Ter um geo criado
  - Ter o tipo de entidade à qual a métrica pertence associado
  - Ter a métrica associada
- Pós-condições:
  - Representação dos pontos no mapa
- Cenário Principal:
  1. Clicar na "checkbox" antes do nome da métrica
  2. Clicar no botão de configuração (roldana)
  3. Clicar na droplist
  4. Selecionar estilo
  5. Clicar em "Save"
- Cenário Alternativo:
  - 1a. Não aconteceu nada
    - 1a1. Esperar cerca de 60 segundos
    - 1a2. Atualizar a página e repetir o processo.
  - 1b. Não aconteceu nada após 2 min
    - 1b1. Contactar o administrador do serviço

O resultado do teste 5 pode ser visto na Figura 29. Os restantes passos podem ser consultados no apêndice I, constando todos os passos pela ordem cronológica descrita no cenário principal.

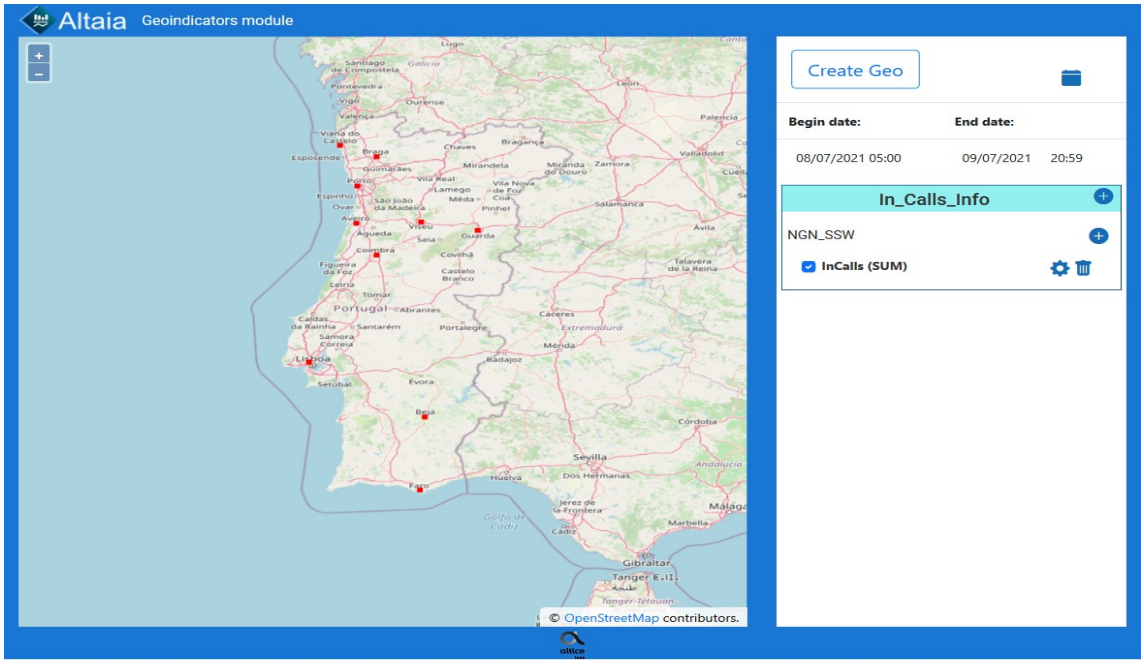


Figura 29 – Resultado final do teste 5.

Teste 6

- Ator: Utilizador
- Objetivo: Visualizar a informação dos pontos representados
- Pré-condições:
  - Estar na página “localhost:4200”
  - Ter um geo criado
  - Ter o tipo de entidade à qual a métrica pertence associado
  - Ter a métrica associada
- Pós-condições:
  - Apresentação da informação relativa à métrica no ponto
- Cenário Principal:
  1. Tornar a métrica a visualizar no mapa
  2. Clicar no ponto no mapa
  3. Visualizar a informação do ponto
- Cenário Alternativo:

  2a. O ponto não contém dados (pontos no mapa)

  2a1. Escolher um ponto com dados (com um símbolo/forma)

O resultado do teste 6 pode ser visto na figura 30. Os restantes passos podem ser consultados no apêndice J, constando todos os passos pela ordem cronológica descrita no cenário principal.

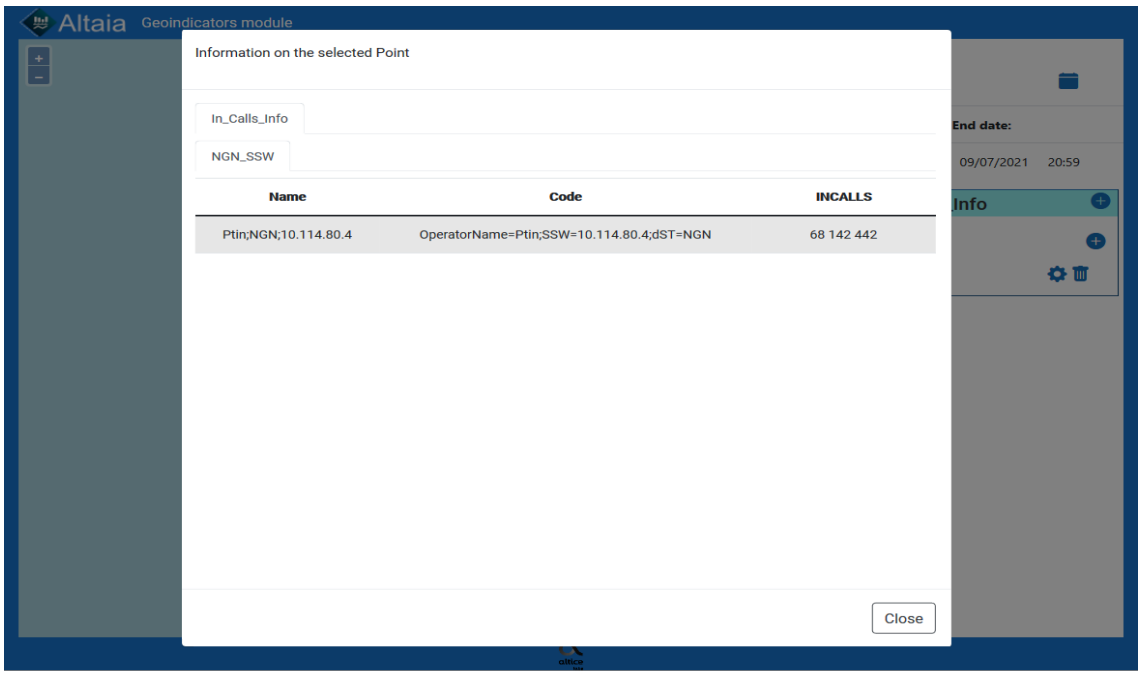


Figura 30 - Resultado final do teste 6.

Teste 7

- Ator: Utilizador
- Objetivo: Visualizar a informação dos pontos representados de múltiplas métricas
- Pré-condições:
  - Estar na página “localhost:4200”
  - Ter um geo criado
  - Ter o tipo de entidade à qual a métrica pertence associado
  - Ter várias métricas associadas
- Pós-condições:
  - Apresentação da informação de todas as métricas (checked) no ponto
- Cenário Principal:
  1. Tornar as métricas a visualizar no mapa visíveis (check)
  2. Clicar no ponto no mapa
  3. Visualizar a informação
- Cenário Alternativo:
  2a. O ponto não contém dados (pontos no mapa)
  2a1. Escolher um ponto com dados
  2b. Não acontece nada ao clicar nos pontos
  2b1. Verificar se existe alguma representação sem dados visível (Exemplo: Dentro do espaço de tempo escolhido não haver informação para essa métrica)
  2b2. Ocultar a tile que não tem dados removendo a seleção da métrica

O resultado do teste 7 pode ser visto na figura 31. Os restantes passos podem ser consultados no apêndice K, constando todos os passos pela ordem cronológica descrita no cenário principal. Ainda no apêndice K constam também com mais exemplos de múltiplos pontos, de diferentes geo’s e tipo de entidade.

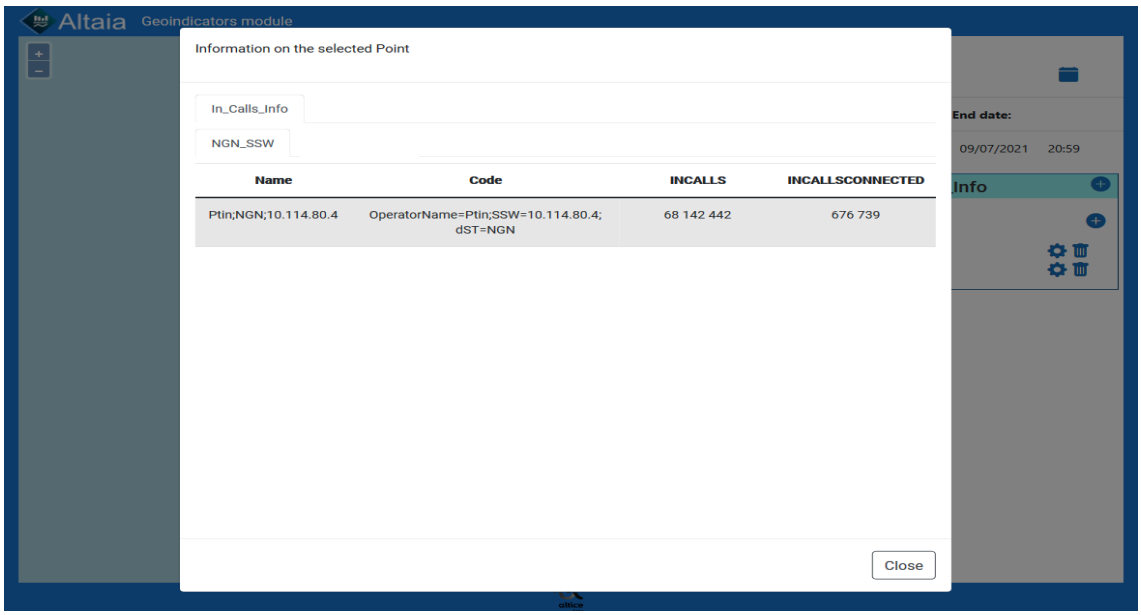


Figura 31 - Resultado final do teste 7.

Capítulo 8
Conclusões

O presente relatório descreve o trabalho realizado no âmbito do estágio "2020_N20 Representação georreferenciada de dados de desempenho e qualidade de serviço em redes de telecomunicações ". O principal objetivo do estágio era o desenvolvimento do módulo GeoIndicators para o Altaia. As principais inovações pretendidas para o módulo eram:

- Gerar dinamicamente *layers* de visualização a partir de meta-modelos de informação;
- Permitir a agregação espacial dos dados (métricas) consultados;
- Permitir a localização temporal dos dados consultados;
- Permitir a representação simultânea de indicadores co-localizados;
- Permitir o uso de uma ou mais das soluções propostas como alternativa para visualização de informação georreferenciada;

Durante o 1º Semestre foram realizadas as etapas relativas à Identificação do problema, Estado da Arte, Levantamento de Requisitos e Análise de Ferramentas de Representação geográfica. A principal dificuldade revelada nesta fase foi a identificação do processo de dinamização adequado a aplicar para diminuir a interação humana (do programador) durante a criação das *layers*, tendo sido enviesada pela área de estudos (Sistemas Inteligentes), e consequentemente levou a um desvio do objetivo principal do estágio.
Nesta fase ainda não existia um plano para a dinamização pelo que a ideia inicial era a aplicação de um modelo de inteligência artificial.
Durante o 2º Semestre, perante os resultados obtidos anteriormente e as dificuldades apresentadas, foi revista a abordagem ao problema, optando assim por abandonar a ideia de aplicação de um modelo de IA para a dinamização, uma vez que não existia fundamento para o aplicar após um melhor entendimento do problema. Foram também revistos os requisitos e metas para o estágio.
Após várias revisões do problema optou-se pelo desenho de um serviço REST. Este deveria comunicar com a Base de Dados, com o GeoServer e com a interface web do módulo. Era também necessário guardar os dados das informações consultadas no módulo, para acessos posteriores.
A principal dificuldade nesta fase foi a interação com o GeoServer via REST, tanto para consulta das *layers* como a sua criação. Isto deveu-se a o facto de a documentação existente ser muito limitada e nem sempre se encontrar no sítio mais expectável. Algumas descobertas foram feitas com base em tentativas-erro, o que levou a um atraso do desenvolvimento.
No fim do desenvolvimento todas as funcionalidades do módulo foram testadas garantido assim que todos os requisitos tinham sido atingidos.

Trabalho futuro

Apesar de atingidos os objetivos, existe sempre espaço de melhoria. No futuro, relativamente à representação geográfica, pode-se apresentar representações mais complexas, além de pontos, começando por linhas evidenciando por exemplo a direção da comunicação, gerando assim visualizações alternativas para os dados. Pode-se tentar aplicar um método de *drill down* para representações com maior densidade de pontos georrefenciados.

Numa outra perspetiva após a implementação do módulo, dever-se-ia proceder à recolha dos seus dados de utilização, permitindo, assim, após a recolha de dados suficientes para um *dataset* de treino significativo, aplicar posteriormente um sistema de recomendação capaz de sugerir visualizações consoante o histórico, ou até sugerir à partida (para novos utilizadores), as visualizações mais consultadas.

Considerações Finais

Em suma, considera-se que os objetivos principais do estágio foram atingidos, uma vez que foram considerados todos os requisitos acordados para o módulo GeoIndicators. Durante o estágio foram exploradas ferramentas novas, permitindo alargar horizontes e aumentar o conhecimento. O contacto com uma equipa de trabalho, permitiu a aprendizagem e execução de técnicas de desenvolvimento, assim como melhoria da capacidade de trabalho em equipa. Considera-se que todos os conhecimentos adquiridos, são bastante importantes no currículo de qualquer engenheiro informático, quer a nível de ferramentas de trabalho, quer a nível de metodologias.

Referências

[1] Altaia - End-to-End Assurance Solution. 2021. Altaia. [online] Available at: <https://www.alticelabs.com/site/altaia/> [Accessed 28 October 2021].

[2] Pt.wikipedia.org. Mapa. [online] Available at: <https://pt.wikipedia.org/wiki/Mapa> [Accessed 28 October 2021].

[3] En.wikipedia.org. Cartography. [online] Available at: <https://en.wikipedia.org/wiki/Cartography#Early-Modern_Period> [Accessed 28 October 2021].

[4] Pt.wikipedia.org. Aviação Na Primeira Guerra Mundial. [online] Available at: <https://pt.wikipedia.org/wiki/Aviação_na_Primeira_Guerra_Mundial> [Accessed 28 October 2021].

[5] En.wikipedia.org. Global Positioning System. [online] Available at: <https://en.wikipedia.org/wiki/Global_Positioning_System#Basic_concept> [Accessed 28 October 2021].

[6] Commons.wikimedia.org. File:WGS 84 Reference Frame (Vector Graphic).Svg - Wikimedia Commons. [online] Available at: <https://commons.wikimedia.org/wiki/File:WGS_84_reference_frame_(vector_graphic).svg > [Accessed 28 October 2021].

[7] Earth-info.nga.mil. NGA Geomatics - WGS 84. [online] Available at: <https://earth-info.nga.mil/GandG/update/index.php?dir=wgs84&action=wgs84> [Accessed 28 October 2021].

[8] Andreas Hackeloeer, Klaas Klasing, Jukka M. Krisp & Liqiu Meng (2014) Georeferencing: a review of methods and applications, Annals of GIS, 20:1, 61-69, DOI: 10.1080/19475683.2013.868826

[9] En.wikipedia.org. Georeferencing. [online] Available at: <https://en.wikipedia.org/wiki/Georeferencing> [Accessed 28 October 2021].

[10] Goodchild, MF 2010, 'Twenty years of progress: GIScience in 2010', Journal of Spatial Information Science, vol. 1, no. 2010, pp. 3-20. https://doi.org/10.5311/JOSIS.2010.1.2

[11] Esri.com. About Arcgis | Mapping & Analytics Platform. [online] Available at: <https://www.esri.com/en-us/arcgis/about-arcgis/overview> [Accessed 28 October 2021].

[12] Esri.com. About Esri | The Science Of Where. [online] Available at: <https://www.esri.com/en-us/about/about-esri/overview> [Accessed 28 October 2021].

[13] Wiki.osgeo.org. Annual Report 2007 Compiled - Osgeo. [online] Available at: <https://wiki.osgeo.org/wiki/Annual_Report_2007_Compiled> [Accessed 28 October 2021].

[14] GitHub. Qgis/QGIS. [online] Available at: <https://github.com/qgis/QGIS> [Accessed 28 October 2021].

[15] Leafletjs.com. Leaflet — An Open-Source Javascript Library For Interactive Maps. [online] Available at: <https://leafletjs.com> [Accessed 28 October 2021].

[16] Mapbox. API Reference | Mapbox GL JS. [online] Available at: <https://docs.mapbox.com/mapbox-gl-js/api/> [Accessed 28 October 2021].

[17] Web.archive.org. Openlayers Achève Sa Phase D'incubation | Osgeo.Org. [online] Available at: <https://web.archive.org/web/20090331064151/http://www.osgeo.org/node/462> [Accessed 28 October 2021].

[18] Geoapify. Map Libraries Popularity: Leaflet Vs Mapbox GL Vs Openlayers. [online] Available at: <https://www.geoapify.com/map-libraries-comparison-leaflet-vs-mapbox-gl-vs-openlayers-trends-and-statistics> [Accessed 28 October 2021].

[19] Yang, D., Zhang, D., Chen, L., and Qu, B. 2015. NationTelescope: Monitoring and visualizing large-scale collective behavior in LBSNs. Journal of Network and Computer Applications, 55, p.170–180.

[20] Yang, D., Zhang, D., and Qu, B. 2015. Participatory cultural mapping based on collective behavior in location based social networks. ACM Transactions on Intelligent Systems and Technology.

[21] Insights.stackoverflow.com. Stack Overflow. [online] Available at: <https://insights.stackoverflow.com/trends?tags=> [Accessed 28 October 2021].

[22] Typescriptlang.org. [online] Available at: <https://www.typescriptlang.org> [Accessed 28 October 2021].

[23] Angular.io. Angular. [online] Available at: <https://angular.io/guide/what-is-angular> [Accessed 28 October 2021].

[24] Redhat.com. What is a REST API? [online] Available at: <https://www.redhat.com/en/topics/cloud-native-apps/what-is-quarkus> [Accessed 28 October 2021].

[25] Quarkus.io. Simplified Hibernate ORM with Panache. [online] Available at: <https://quarkus.io/guides/hibernate-orm-panache> [Accessed 28 October 2021].

[26] Thorben Janssen. 2021. Introduction to Panache. [online] Available at: <https://thorben-janssen.com/introduction-panache/> [Accessed 28 October 2021].

[27] Quarkus.io. *Datasources*. [online] Available at: <https://quarkus.io/guides/datasource> [Accessed 28 October 2021].

[28] Quarkus.io. 2021. *Using the REST Client*. [online] Available at: <https://quarkus.io/guides/rest-client> [Accessed 28 October 2021].

Apêndices

Apêndice A

No Apêndice A é apresentada a *query* gerada para a criação das *views*, às quais o GeoServer acede para consulta da informação.

Preview ▾ Header 2 Cookie Timeline

```
CREATEORREPLACEFORCEEDITIONABLEVIEWNGN_Route_GEO_VIEW("TYPE_NAME",
"CODE",
"PARENT_CODE",
"NAME",
"CREATION_DATE",
"OPERATORNAME",
"DST",
"SSW",
"MGW",
"ROUTE",
"SSWNAME",
"SSWVERSION",
"SSWVENDOR",
"SSWREGION",
"SSWGEOGRAPHY",
"SSWSITE",
"MGWNAME",
"MGWVERSION",
"MGWVENDOR",
"MGWREGION",
"MGWGEOGRAPHY",
"MGWSITE",
"ROUTEANDTG",
"TG",
"ROUTETYPE",
"OPC",
"DPC",
"DESTINATIONTYPE",
"DESTINATIONOPERATOR",
"DESTINATIONMGW",
"DESTINATIONMGWNAME",
"DESTINATIONMGWVERSION",
"DESTINATIONMGWVENDOR",
"DESTINATIONMGWREGION",
"DESTINATIONMGWGEOGRAPHY",
"DESTINATIONMGWSITE",
"REPORT_DATE_TIME",
"SSWLATITUDE",
"SSWLONGITUDE",
"AZIMUTH",
"BEAMWIDTH",
"T1",
"GPS",
"ALTAIA_1198_THE_VERY_LARGE_NAMED_REPORTING_METRIC_CONSTANT_VALUE",
"AVGTOTALCALLS",
"CALLSREPORTINGTEST",
"CALLSREPORTINGTEST2",
"CONDITIONALMETRIC",
"DAILYTOTALCALLS",
"F_TEST_R_01",
"F_TEST_R_02",
```

Figura 32 – Exemplo de *query* usada para criar a *view*. (Parte 1)

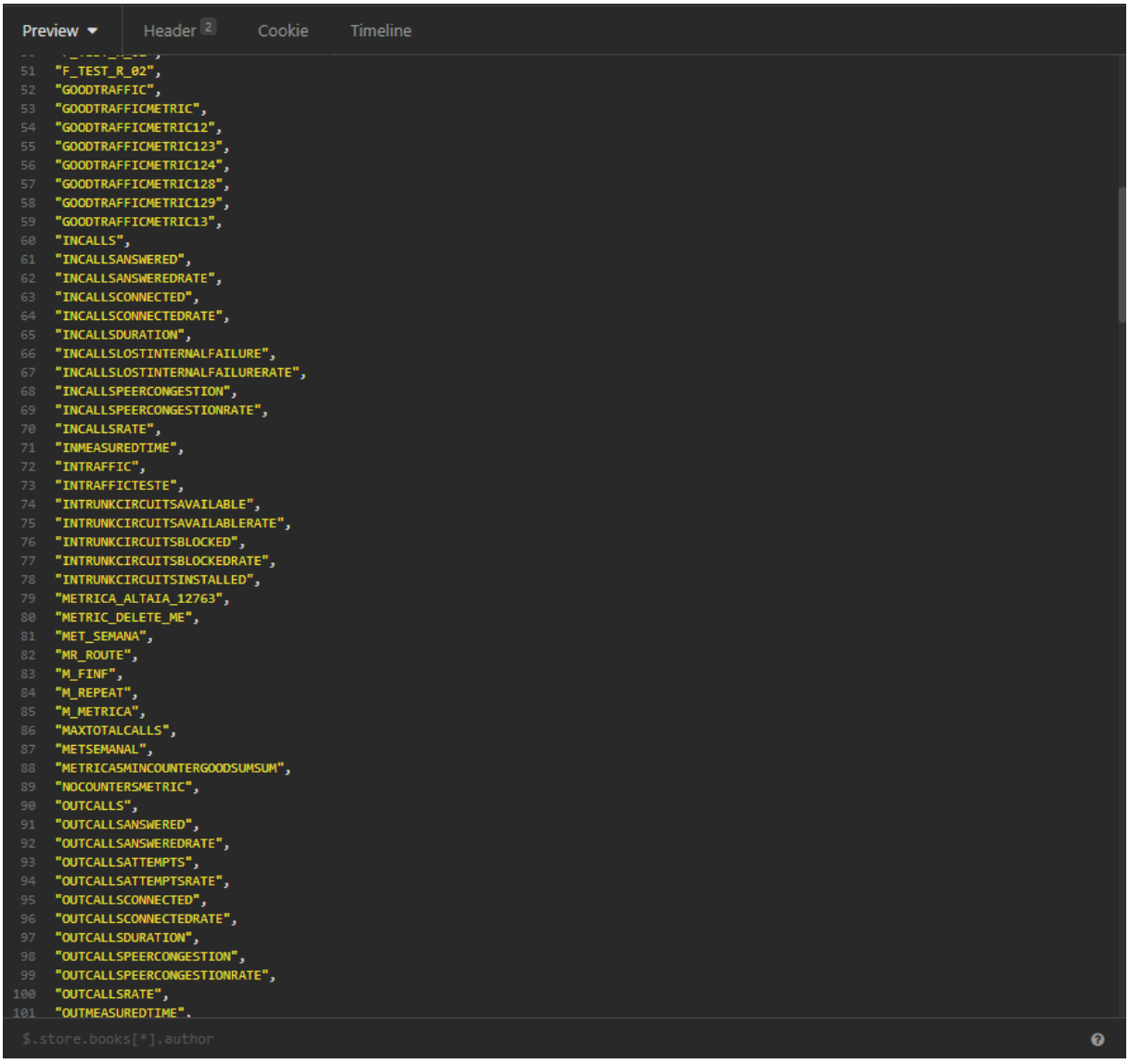


Figura 33 - Exemplo de *query* usada para criar a *view*. (Parte 2)

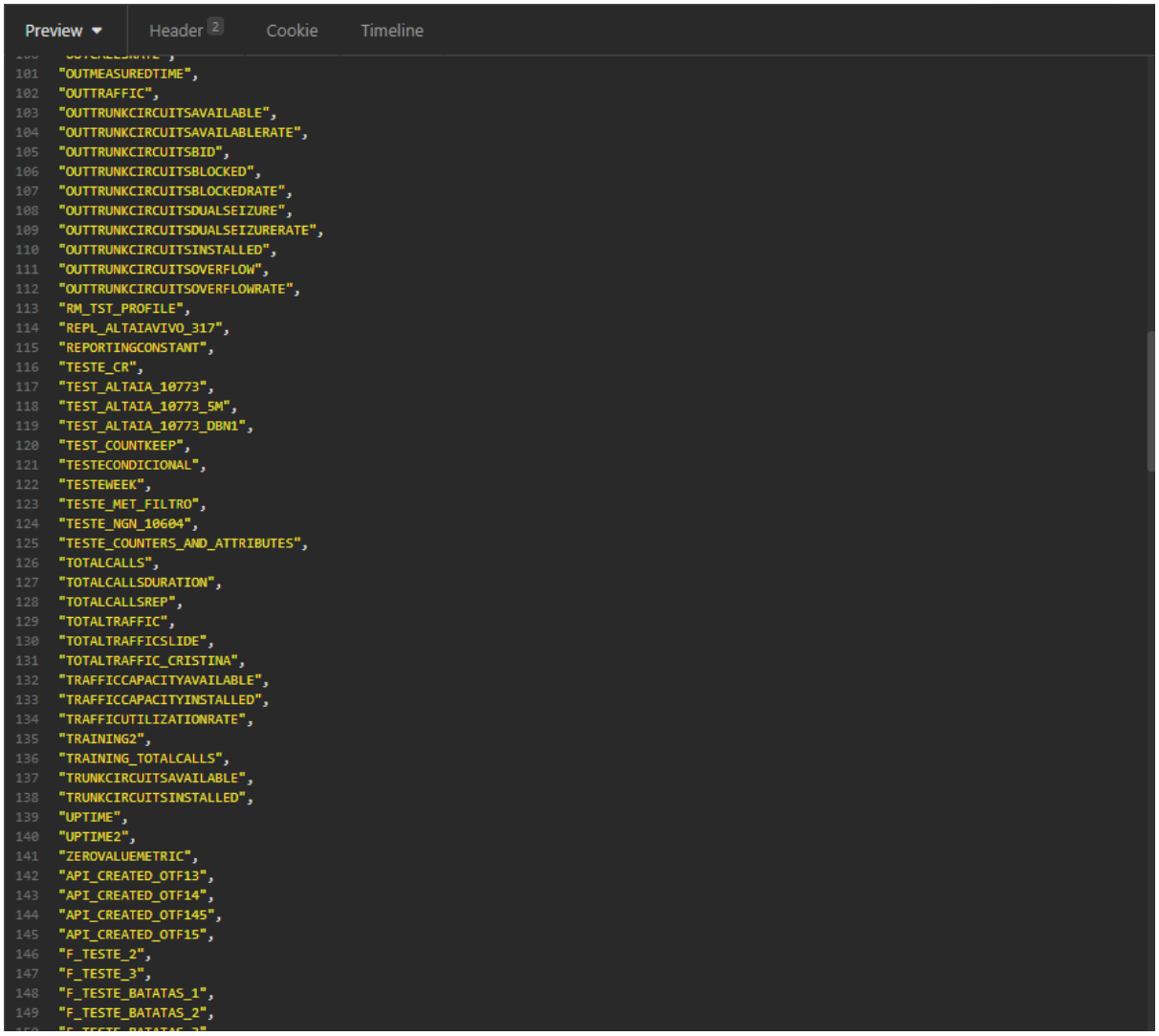


Figura 34 - Exemplo de *query* usada para criar a *view*. (Parte 4)

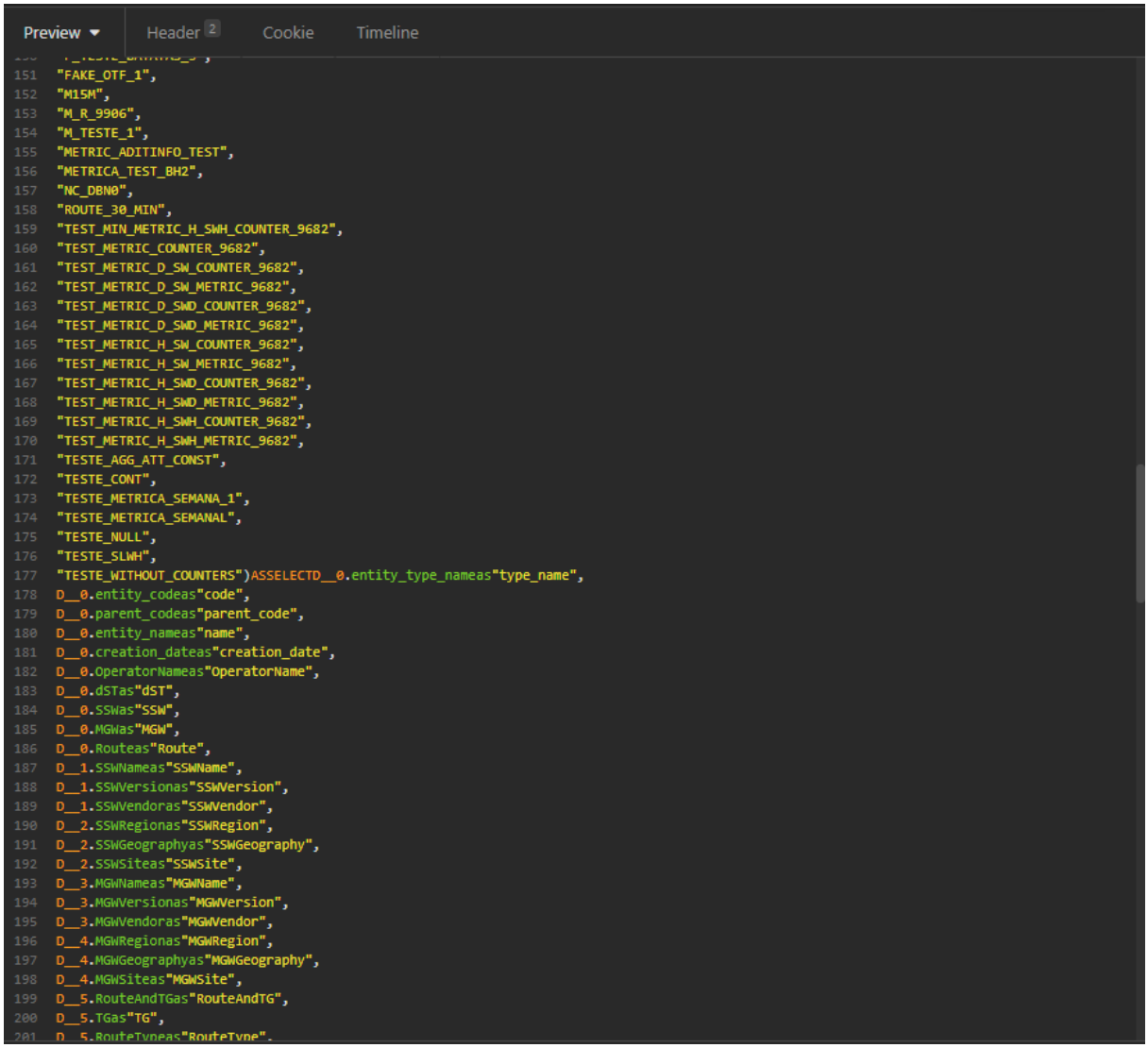
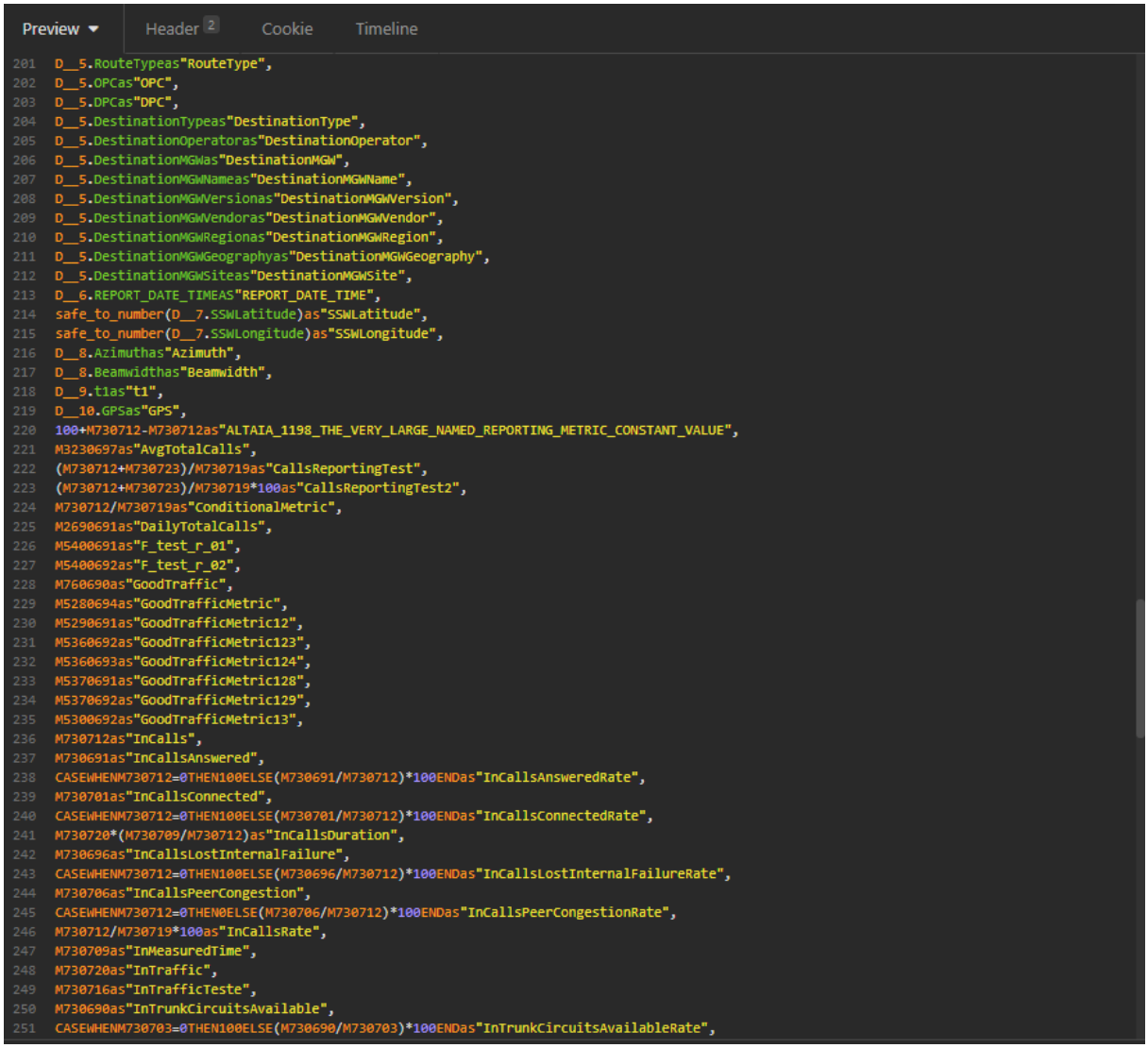


Figura 35 – Exemplo de *query* usada para criar a *view* (Parte 4)



```
Preview ▾   Header 2   Cookie   Timeline
201  D__5.RouteTypeas"RouteType",
202  D__5.OPCas"OPC",
203  D__5.DPCas"DPC",
204  D__5.DestinationTypeas"DestinationType",
205  D__5.DestinationOperatoras"DestinationOperator",
206  D__5.DestinationMGWas"DestinationMGW",
207  D__5.DestinationMGWNameas"DestinationMGWName",
208  D__5.DestinationMGWVersionas"DestinationMGWVersion",
209  D__5.DestinationMGWVendoras"DestinationMGWVendor",
210  D__5.DestinationMGWRegionas"DestinationMGWRegion",
211  D__5.DestinationMGWGeographyas"DestinationMGWGeography",
212  D__5.DestinationMGWSiteas"DestinationMGWSite",
213  D__6.REPORT_DATE_TIMEAS"REPORT_DATE_TIME",
214  safe_to_number(D__7.SSWLatitude)as"SSWLatitude",
215  safe_to_number(D__7.SSWLongitude)as"SSWLongitude",
216  D__8.Azimuthas"Azimuth",
217  D__8.Beamwidthas"Beamwidth",
218  D__9.t1as"t1",
219  D__10.GPSas"GPS",
220  100+M730712-M730712as"ALTAIA_1198_THE_VERY_LARGE_NAMED_REPORTING_METRIC_CONSTANT_VALUE",
221  M3230697as"AvgTotalCalls",
222  (M730712+M730723)/M730719as"CallsReportingTest",
223  (M730712+M730723)/M730719*100as"CallsReportingTest2",
224  M730712/M730719as"ConditionalMetric",
225  M2690691as"DailyTotalCalls",
226  M5400691as"F_test_r_01",
227  M5400692as"F_test_r_02",
228  M760690as"GoodTraffic",
229  M5280694as"GoodTrafficMetric",
230  M5290691as"GoodTrafficMetric12",
231  M5360692as"GoodTrafficMetric123",
232  M5360693as"GoodTrafficMetric124",
233  M5370691as"GoodTrafficMetric128",
234  M5370692as"GoodTrafficMetric129",
235  M5300692as"GoodTrafficMetric13",
236  M730712as"InCalls",
237  M730691as"InCallsAnswered",
238  CASEWHENM730712=0THEN100ELSE(M730691/M730712)*100ENDas"InCallsAnsweredRate",
239  M730701as"InCallsConnected",
240  CASEWHENM730712=0THEN100ELSE(M730701/M730712)*100ENDas"InCallsConnectedRate",
241  M730720*(M730709/M730712)as"InCallsDuration",
242  M730696as"InCallsLostInternalFailure",
243  CASEWHENM730712=0THEN100ELSE(M730696/M730712)*100ENDas"InCallsLostInternalFailureRate",
244  M730706as"InCallsPeerCongestion",
245  CASEWHENM730712=0THEN0ELSE(M730706/M730712)*100ENDas"InCallsPeerCongestionRate",
246  M730712/M730719*100as"InCallsRate",
247  M730709as"InMeasuredTime",
248  M730720as"InTraffic",
249  M730716as"InTrafficTeste",
250  M730690as"InTrunkCircuitsAvailable",
251  CASEWHENM730703=0THEN100ELSE(M730690/M730703)*100ENDas"InTrunkCircuitsAvailableRate",
```

Figura 36 - Exemplo de *query* usada para criar a *view*. (Parte 5)

Preview ▾ | Header 2 | Cookie | Timeline

```
CASEWHENM730703=0THEN100ELSE(M730690/M730703)*100ENDas"InTrunkCircuitsAvailableRate",
M730692as"InTrunkCircuitsBlocked",
CASEWHENM730703=0THEN0ELSE(M730692/M730703)*100ENDas"InTrunkCircuitsBlockedRate",
M730703as"InTrunkCircuitsInstalled",
M3800693as"METRICA_ALTAIA_12763",
M3270691as"METRIC_DELETE_ME",
M2090691as"MET_SEMANA",
M2700691as"MR_ROUTE",
M3460691as"M_FinF",
M1350690as"M_REPEAT",
M4190691as"M_metrica",
M3040691as"MaxTotalCalls",
M3980694as"MetSemanal",
M1236591as"Metrica5minCounterGoodSumSum",
M3230691as"NoCountersMetric",
M730723as"OutCalls",
M730697as"OutCallsAnswered",
CASEWHENM730699=0THEN100ELSE(M730697/M730699)*100ENDas"OutCallsAnsweredRate",
M730698as"OutCallsAttempts",
CASEWHENM730699=0THEN100ELSE(M730698/M730699)*100ENDas"OutCallsAttemptsRate",
M730708as"OutCallsConnected",
CASEWHENM730699=0THEN100ELSE(M730708/M730699)*100ENDas"OutCallsConnectedRate",
M730693*(M730702/M730723)as"OutCallsDuration",
M730711as"OutCallsPeerCongestion",
CASEWHENM730699=0THEN0ELSE(M730711/M730699)*100ENDas"OutCallsPeerCongestionRate",
CASEWHENM730699=0THEN100ELSE(M730723/M730699)*100ENDas"OutCallsRate",
M730702as"OutMeasuredTime",
M730693as"OutTraffic",
M730717as"OutTrunkCircuitsAvailable",
CASEWHENM730707=0THEN100ELSE(M730717/M730707)*100ENDas"OutTrunkCircuitsAvailableRate",
M730699as"OutTrunkCircuitsBid",
M730722as"OutTrunkCircuitsBlocked",
CASEWHENM730703=0THEN0ELSE(M730722/M730707)*100ENDas"OutTrunkCircuitsBlockedRate",
M730694as"OutTrunkCircuitsDualSeizure",
CASEWHENM730699=0THEN100ELSE(M730694/M730699)*100ENDas"OutTrunkCircuitsDualSeizureRate",
M730707as"OutTrunkCircuitsInstalled",
M730714as"OutTrunkCircuitsOverflow",
CASEWHENM730699=0THEN0ELSE(M730714/M730699)*100ENDas"OutTrunkCircuitsOverflowRate",
M980693+10as"RM_TST_PROFILE",
M5430691as"Repl_ALTAIAVIVO_317",
M730719-M730719+8888as"ReportingConstant",
M3910691as"TESTE_CR",
M2560691as"Test_ALTAIA_10773",
M2560697as"Test_ALTAIA_10773_5m",
M2560692as"Test_ALTAIA_10773_DBN1",
M2720694as"Test_CountKeep",
(M730712/M730723)*100as"TesteCondicional",
M5870691as"TesteWeek",
M3340691as"Teste_Met_Filtro",
M2700699as"Teste_NGN_10604",
M3230692as"Teste_counters_and_attributes",
```

Figura 37 - Exemplo de query usada para criar a view. (Parte 6)

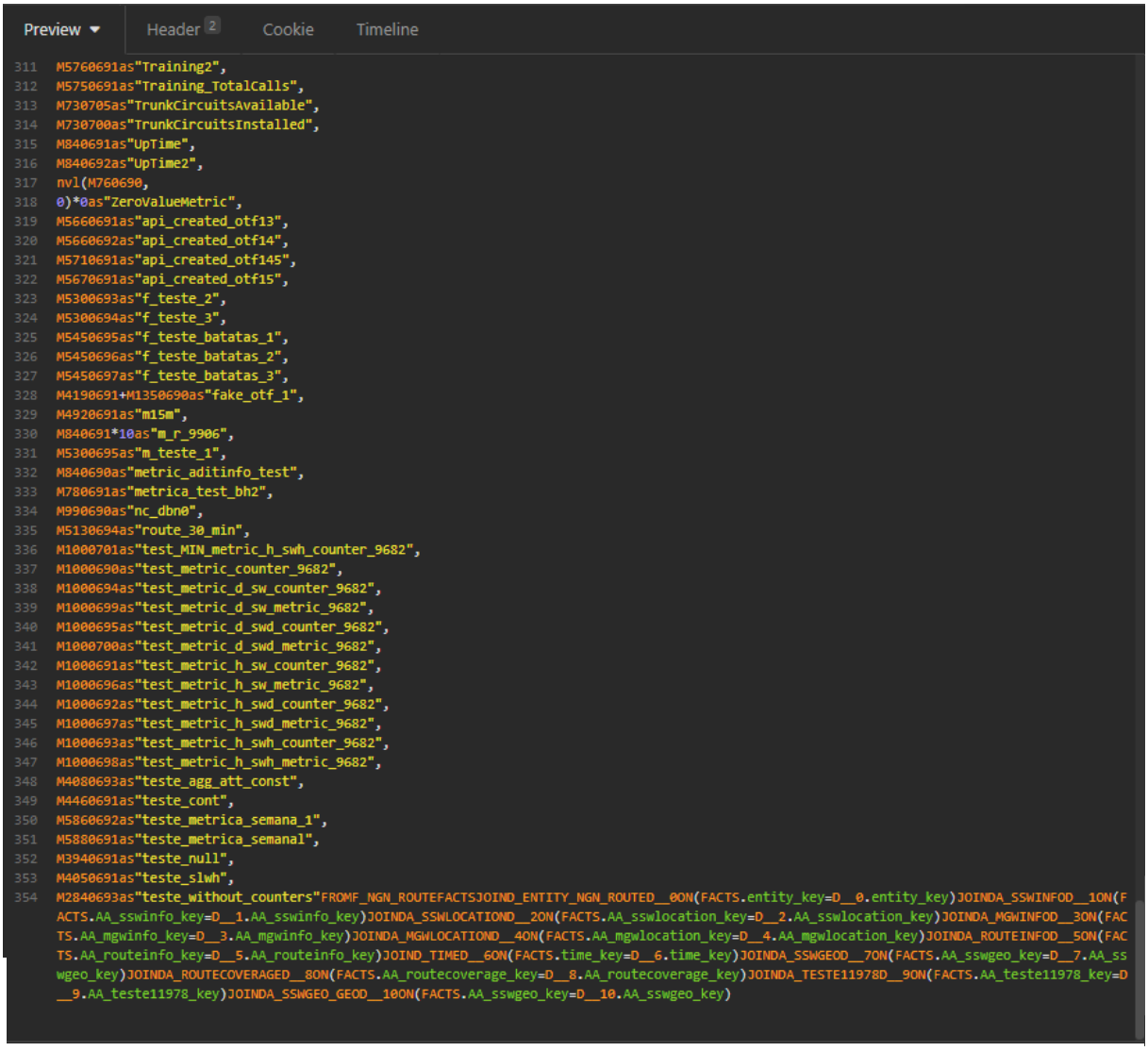


Figura 39 - Exemplo de query usada para criar a view. (Parte 8)

Apêndice B

No Apêndice B são descritos os *Enpoints* do API do serviço "geos". Contem a lista dos *endpoints* e os parametros de cada um deles.

Figura 40 - *Endpoints* do serviço geos.

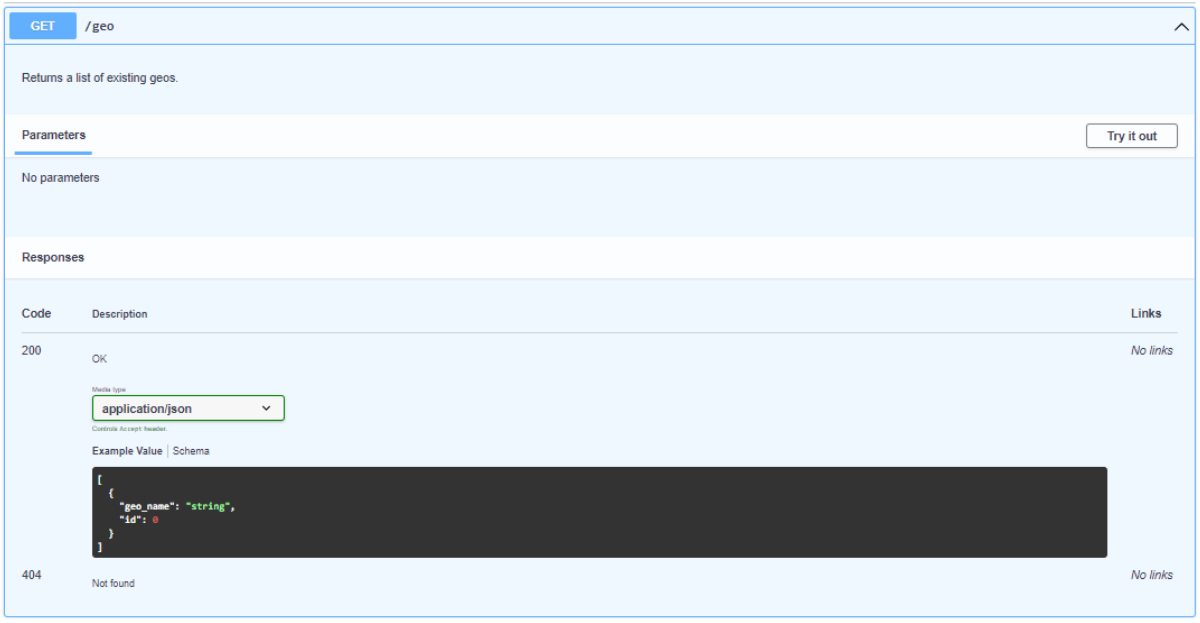


Figura 41 - *Endpoint* para mostrar todos os geos existentes.

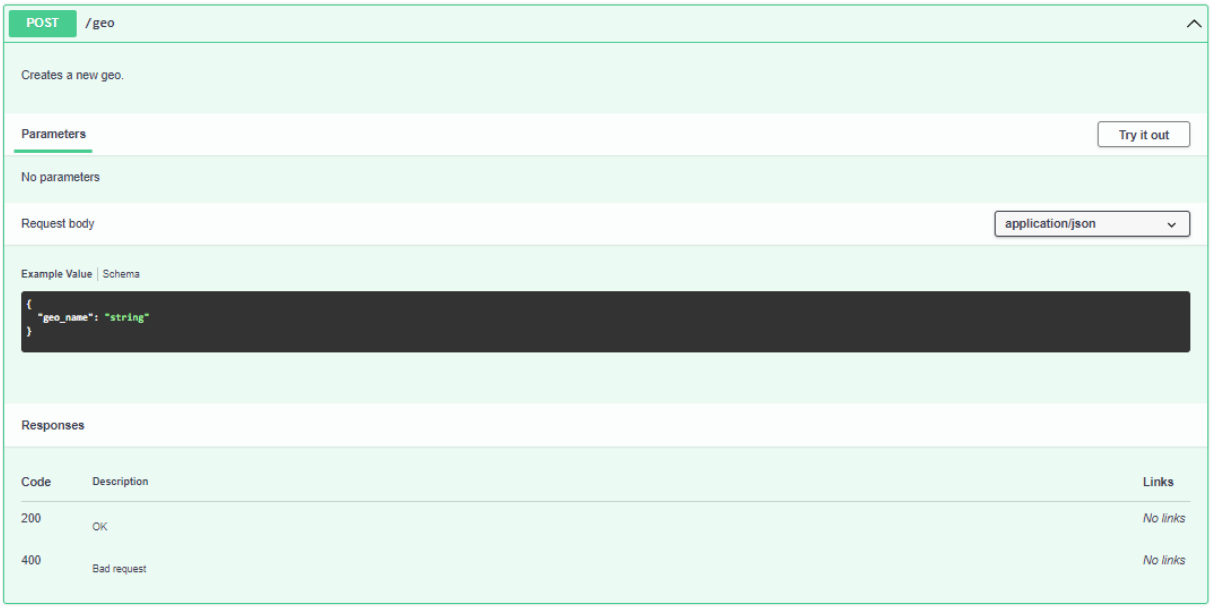


Figura 42 - *Endpoint* para criação de um novo geo.

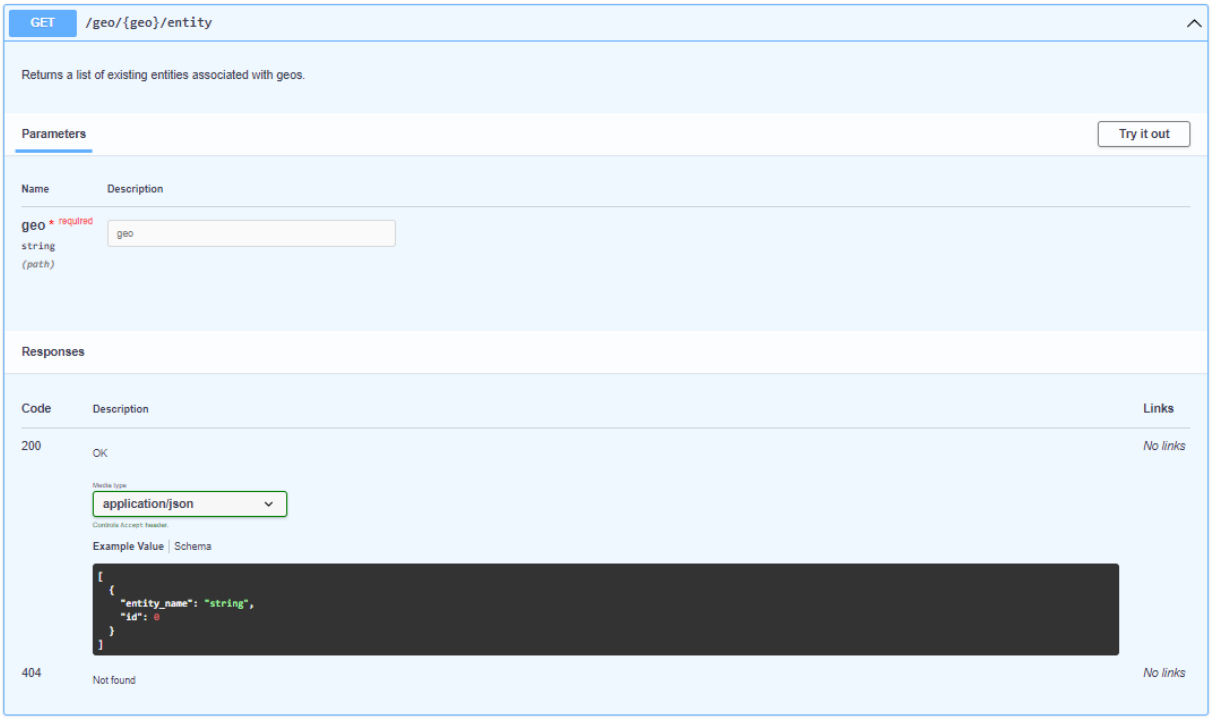


Figura 43 - *Endpoint* para mostrar os tipos de entidade associadas ao geo

Figura 44 - *Endpoint* para associação do tipo de entidade a um geo.

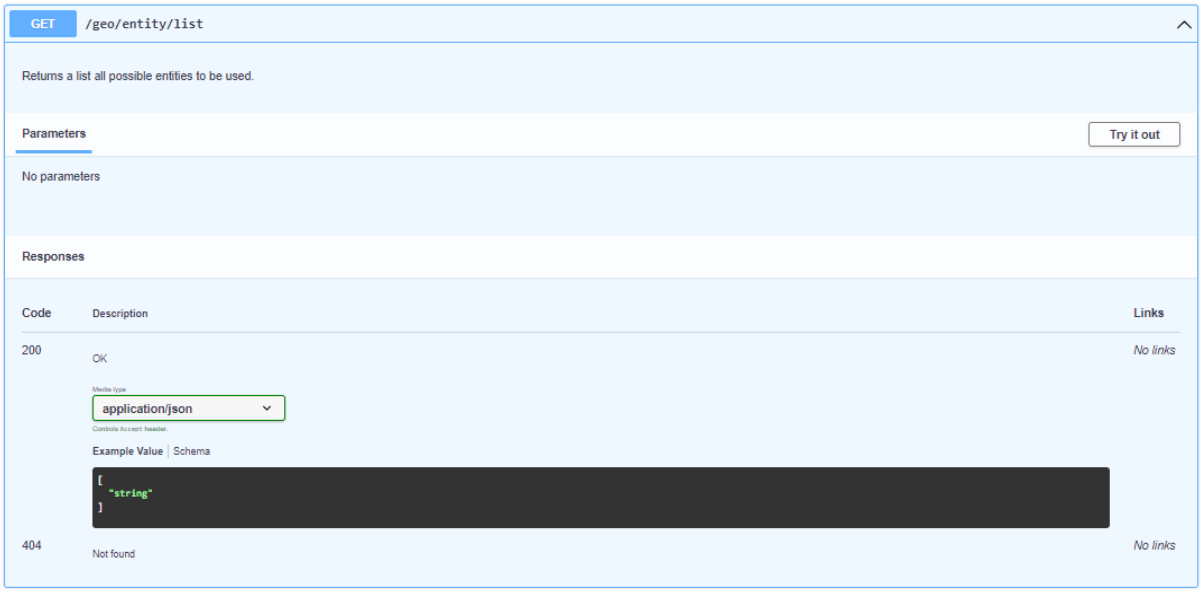


Figura 45 - Endpoint para listagem de todas os tipos de entidade possíveis.

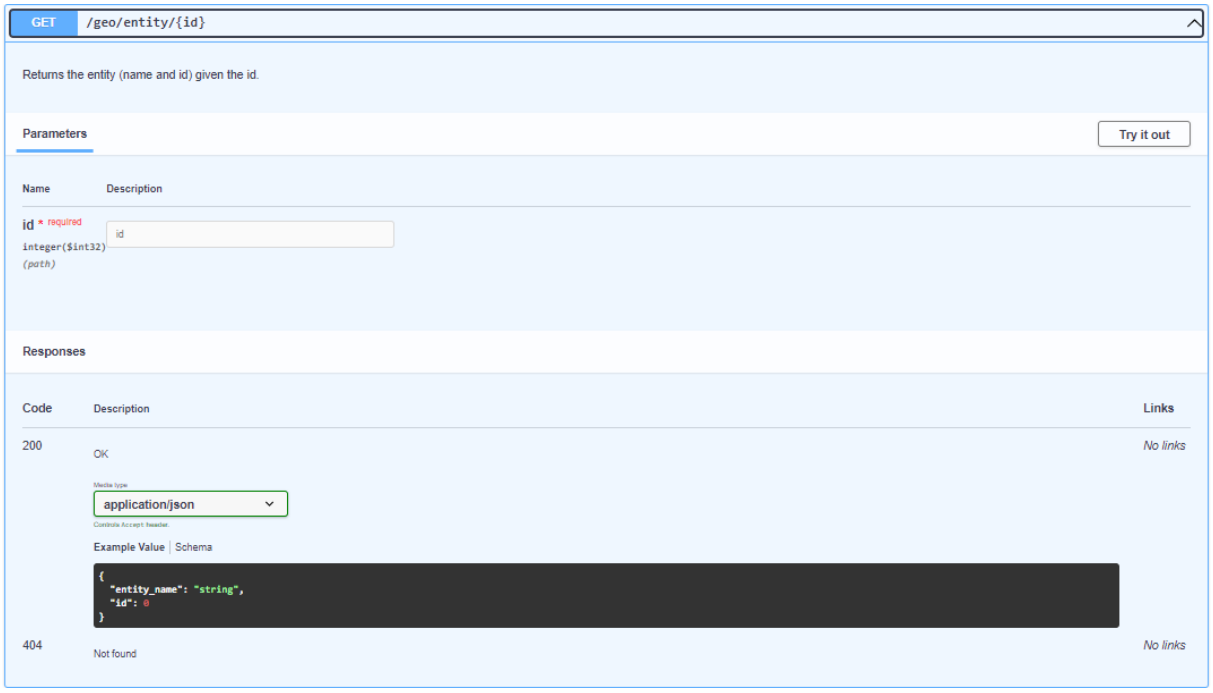


Figura 46 - Endpoint para consulta da informação do tipo de entidade identificada pelo id.

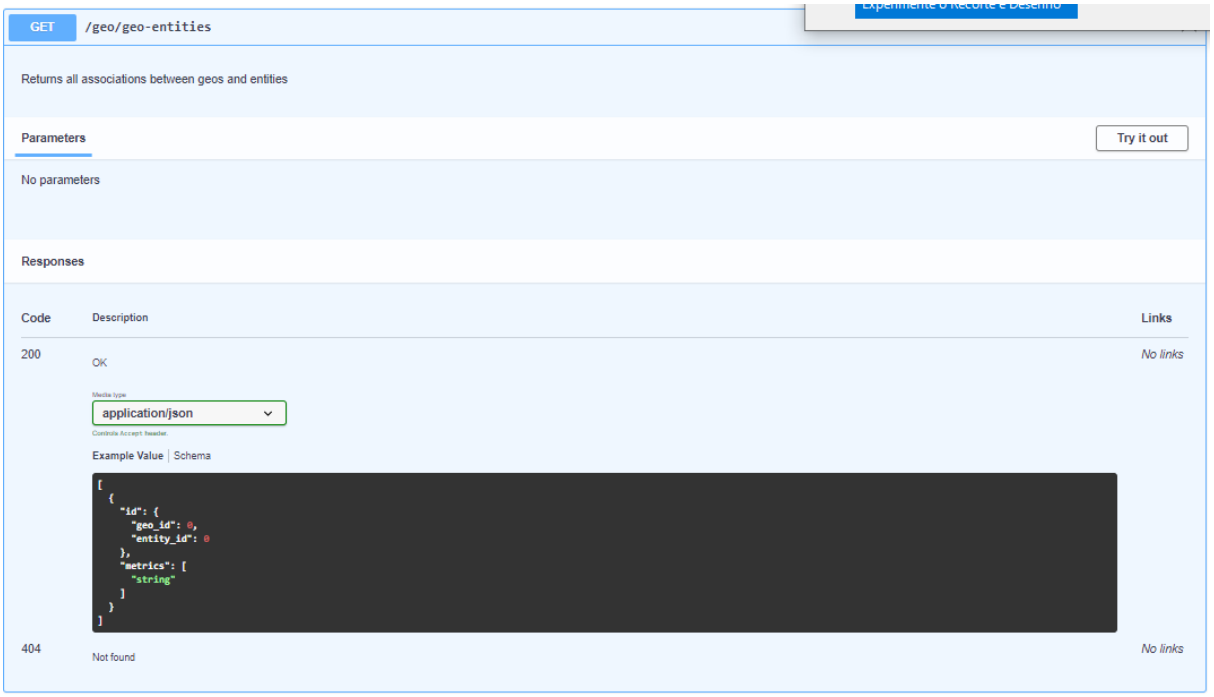


Figura 47 - *Endpoint* para a consulta das associações entre geos e tipo de entidade existentes.

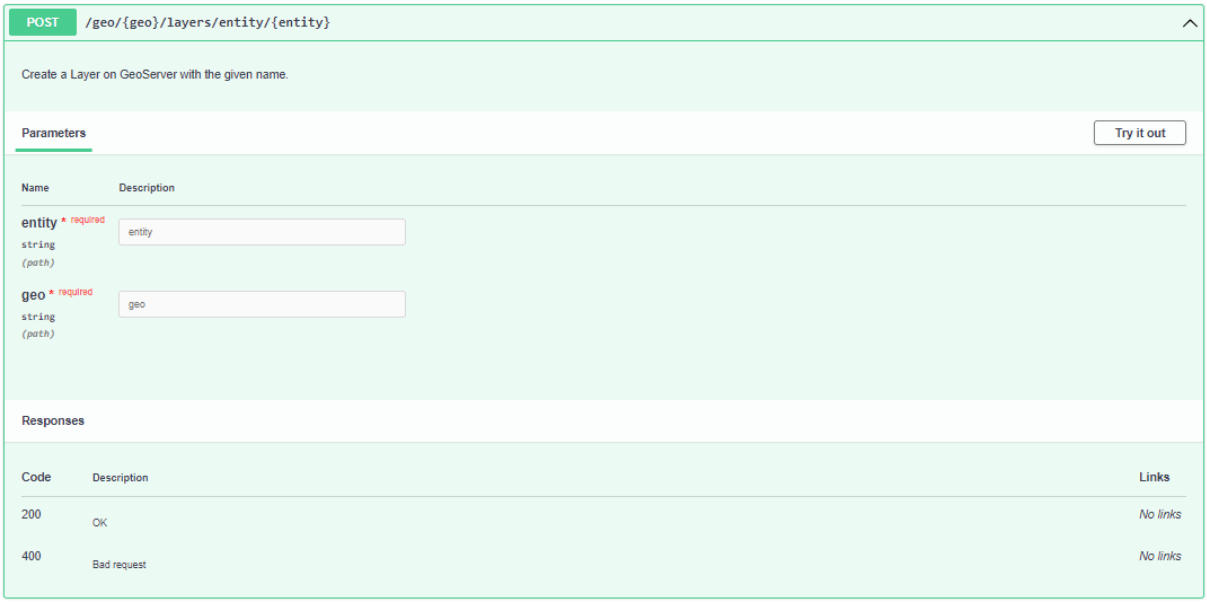


Figura 48 - *Endpoint* para criação da *layer* no Geoserver.

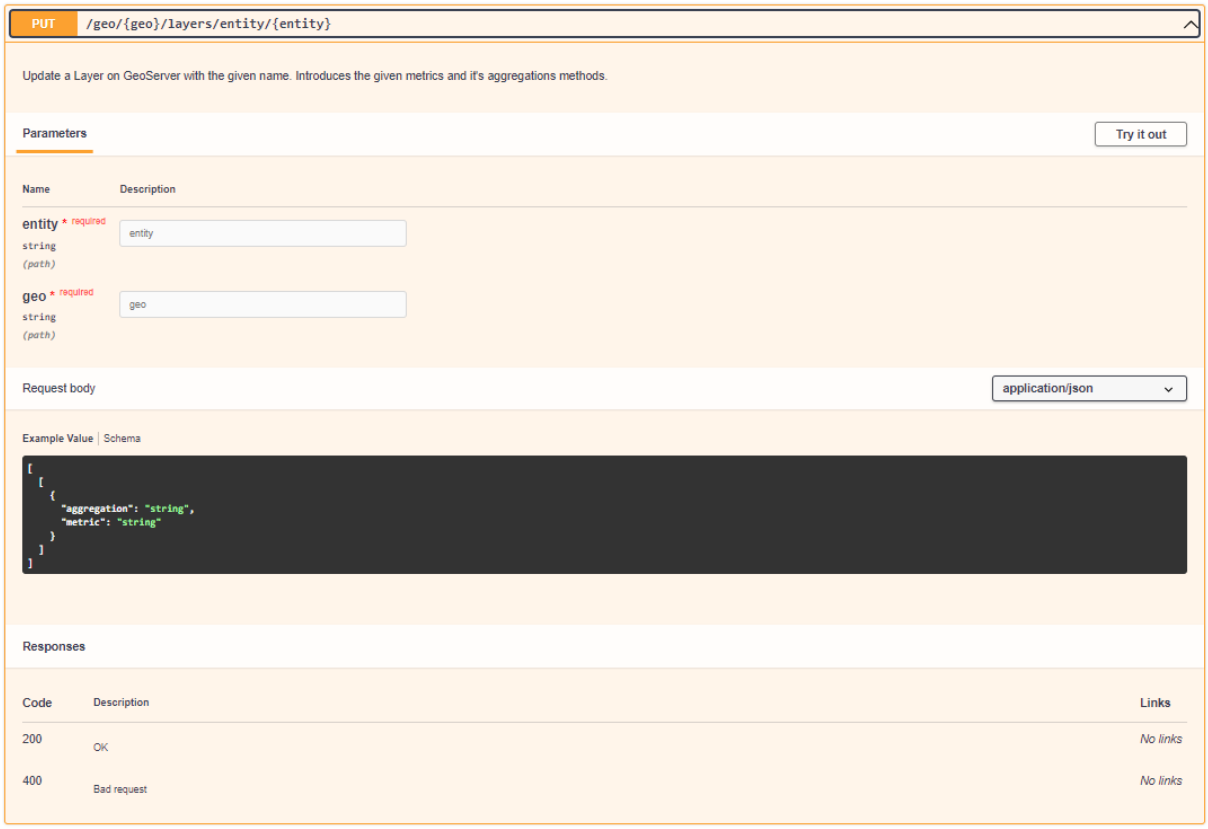


Figura 49 - Endpoint para a edição da layer.

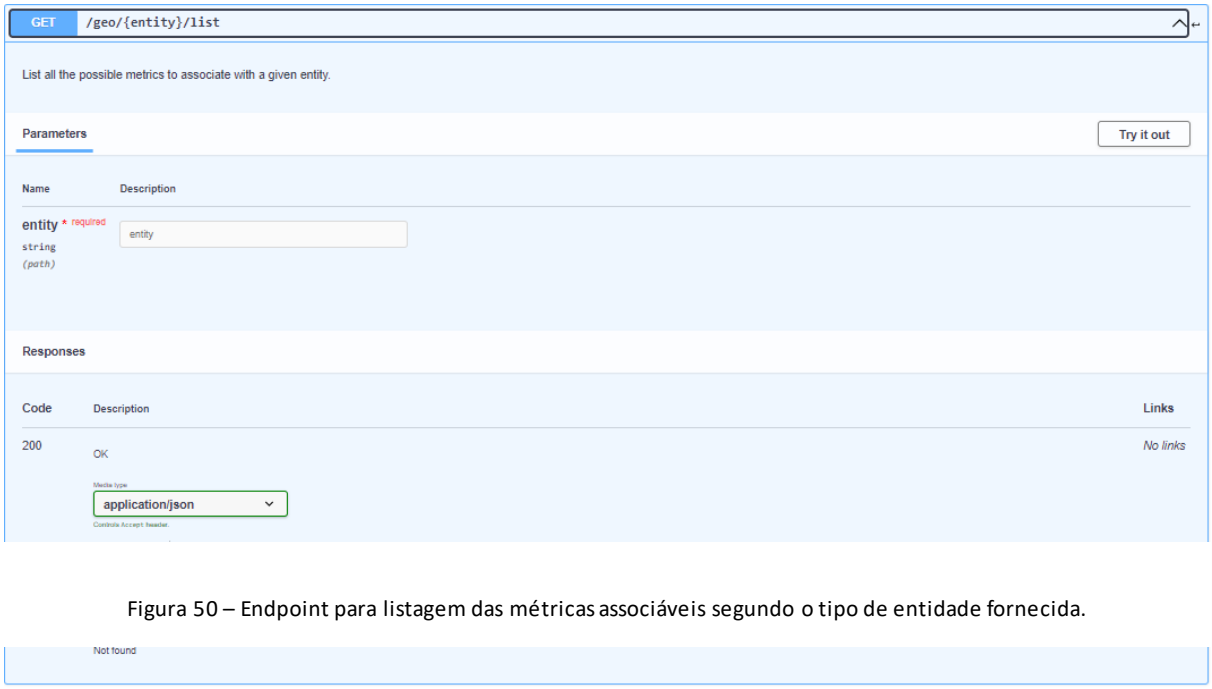


Figura 50 – Endpoint para listagem das métricas associáveis segundo o tipo de entidade fornecida.

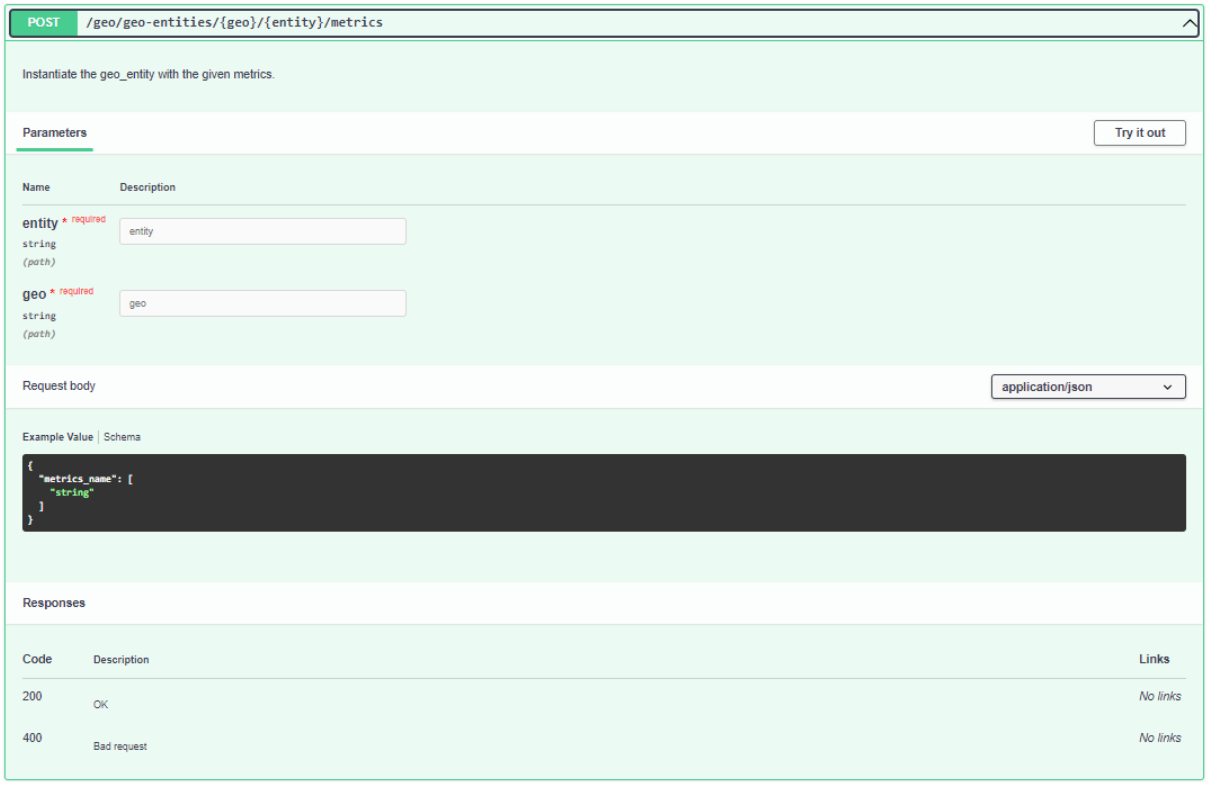


Figura 51 - Endpoint para instanciar o Geo_Entities.

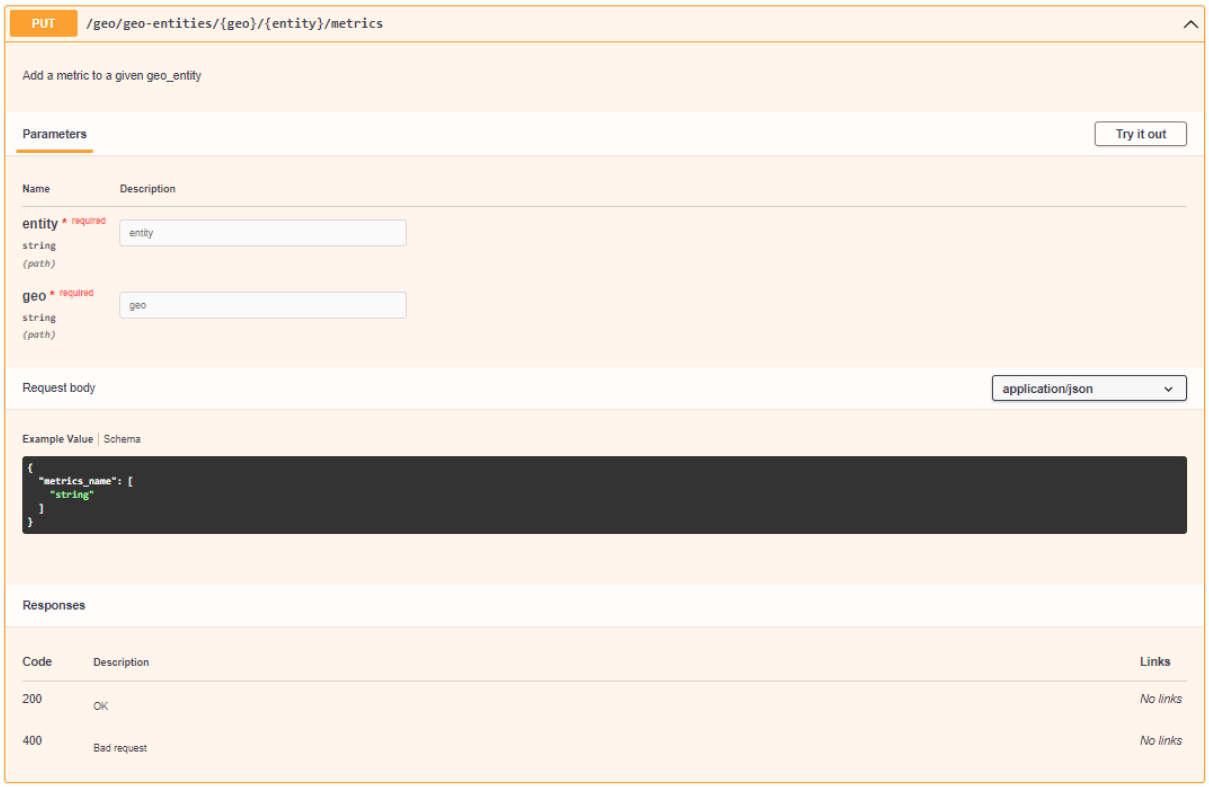


Figura 52 - *Endpoint* para adição de Métricas ao Geo_Entity.

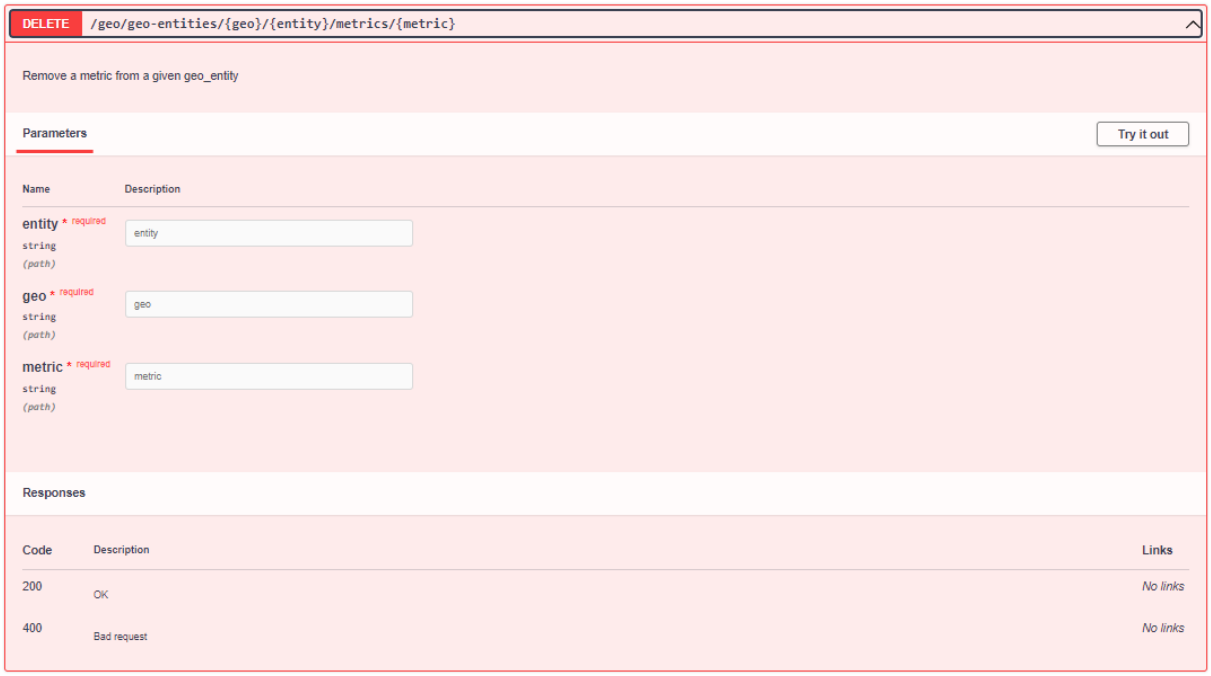


Figura 53 - *Endpoint* para a remoção de Métricas do Geo_Entity.

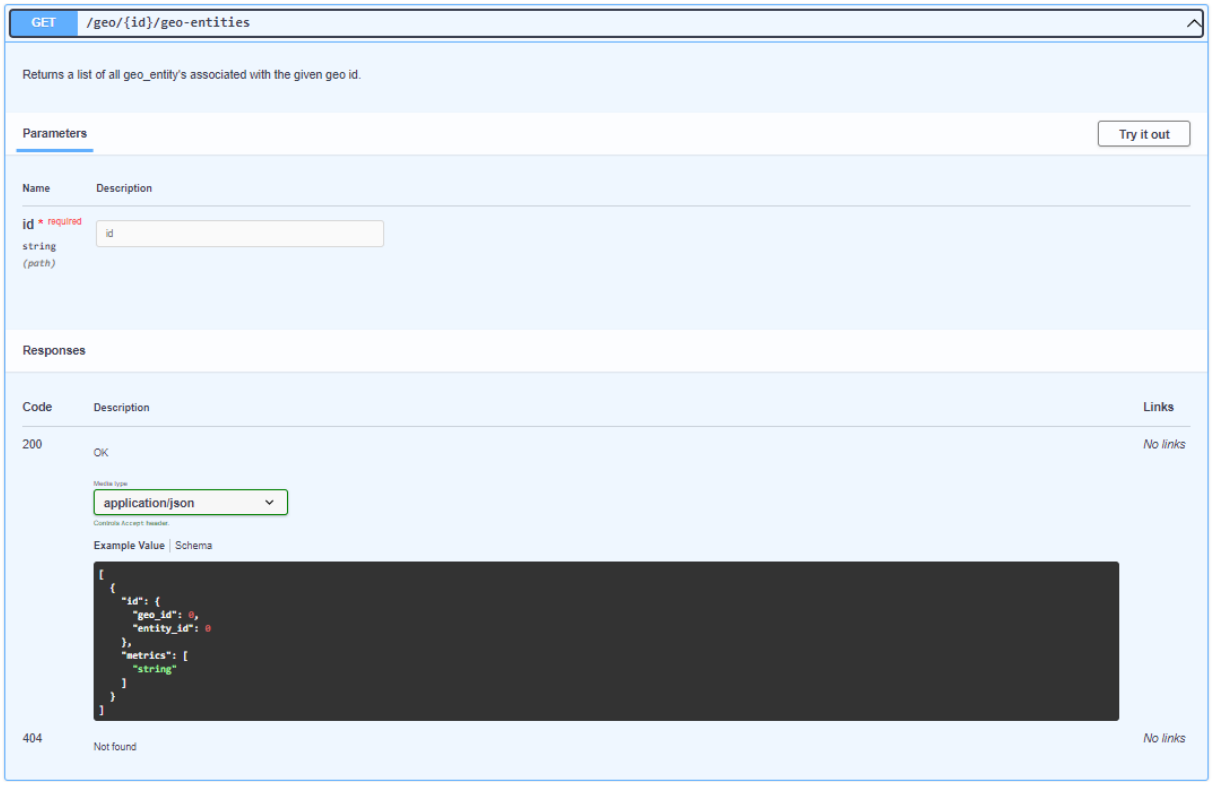


Figura 54 - *Endpoint* para listar os Geo_Entities associados a um geo.

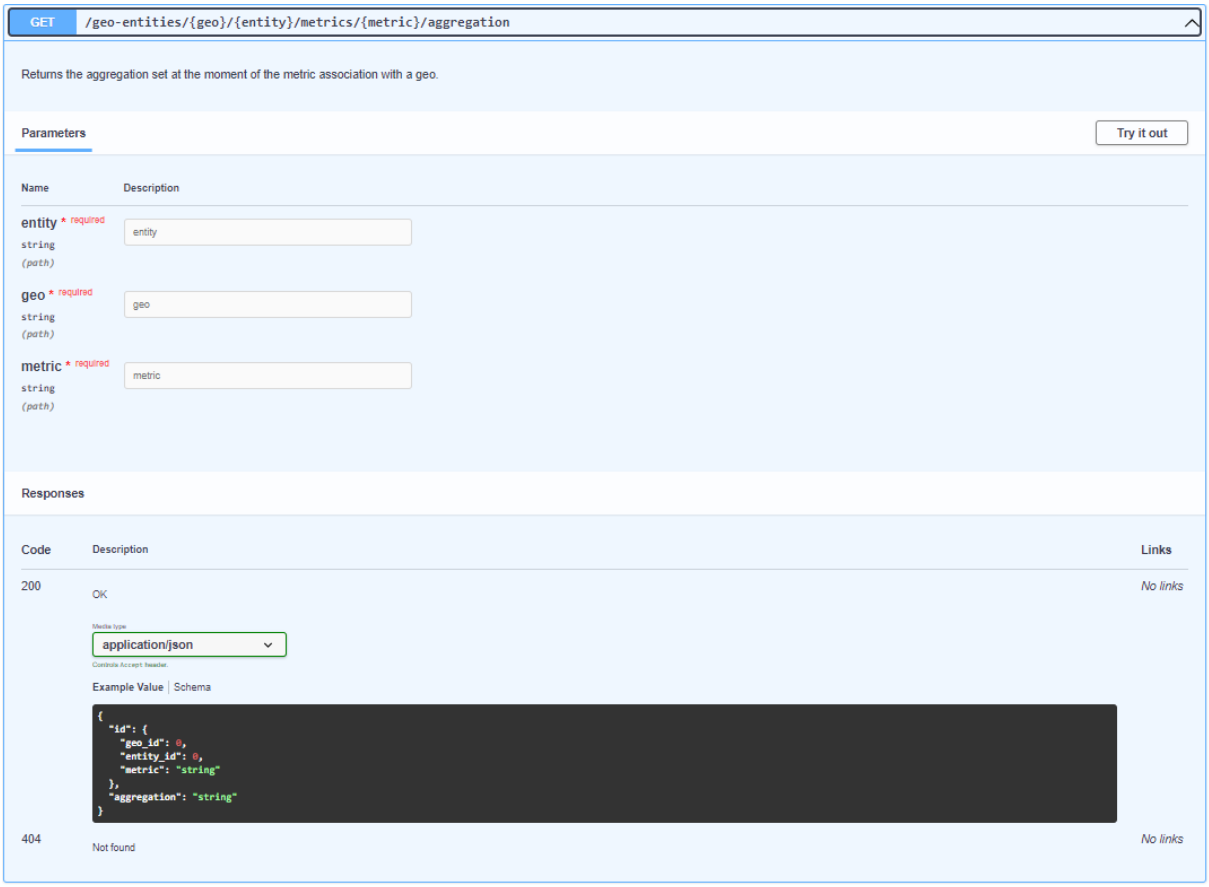


Figura 55 - *Endpoint* para consulta do método de agregação utilizado na associação da métrica ao geo.

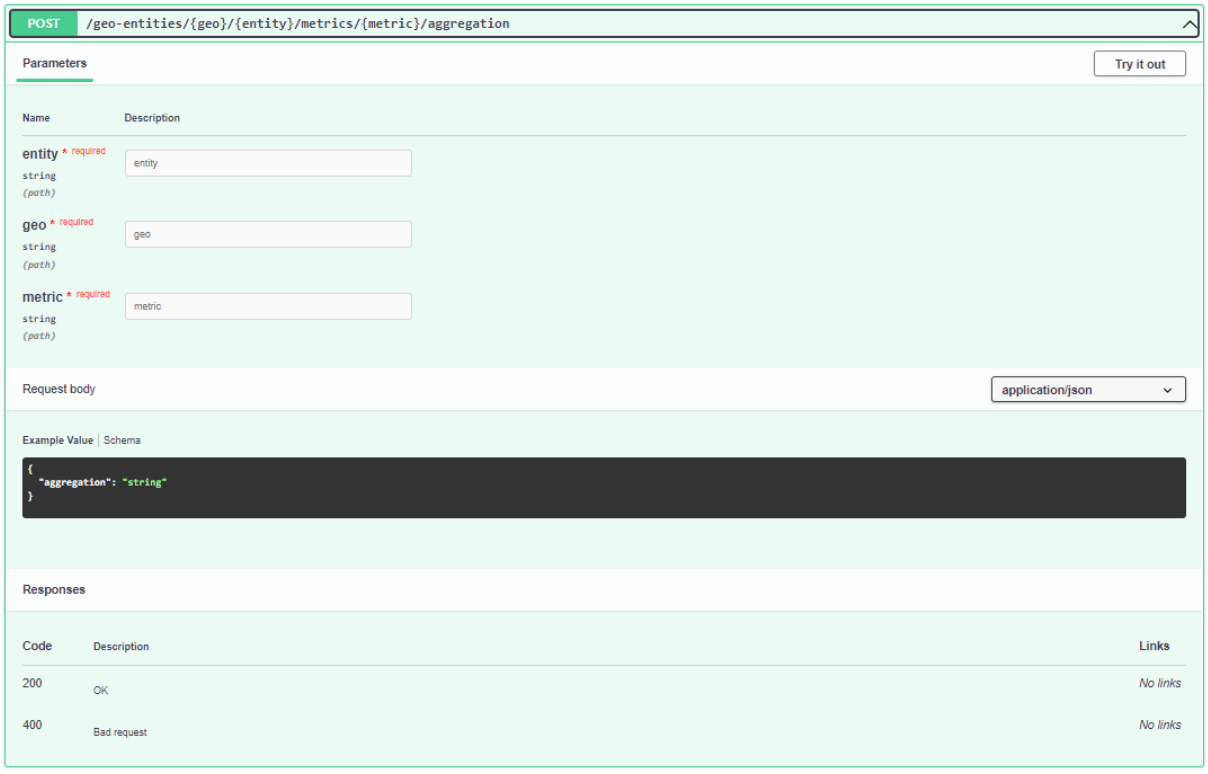


Figura 56 - *Endpoint* para adição do método de agregação utilizado na associação da métrica ao geo.

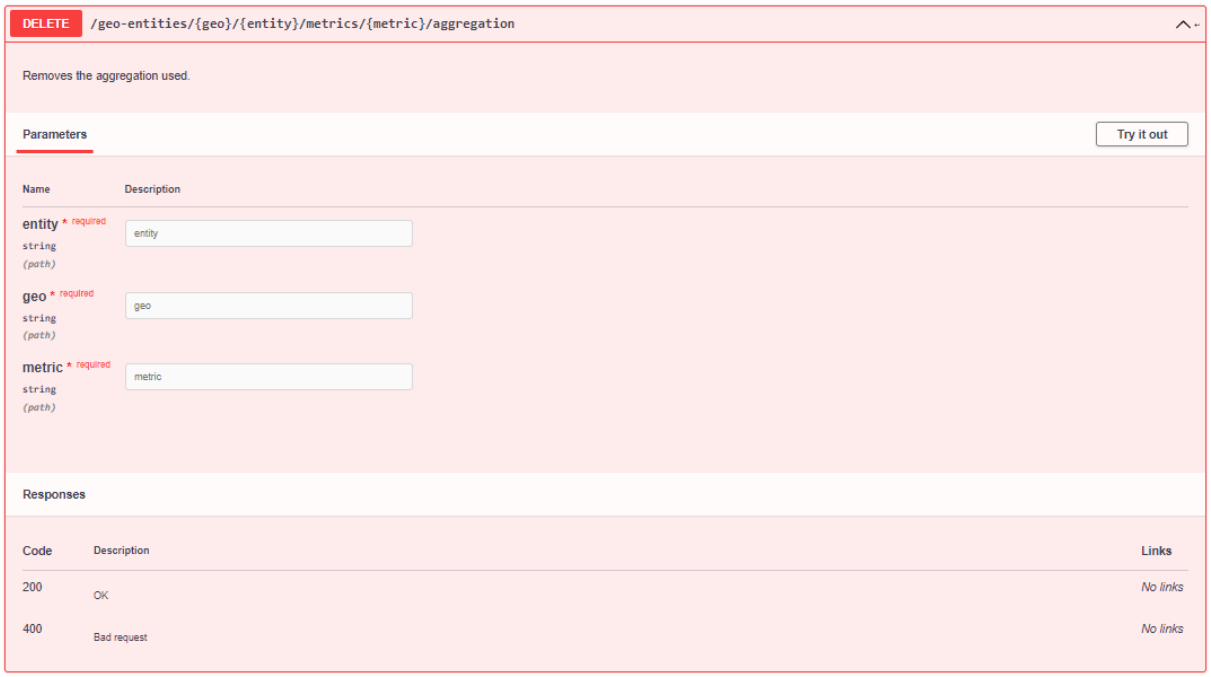
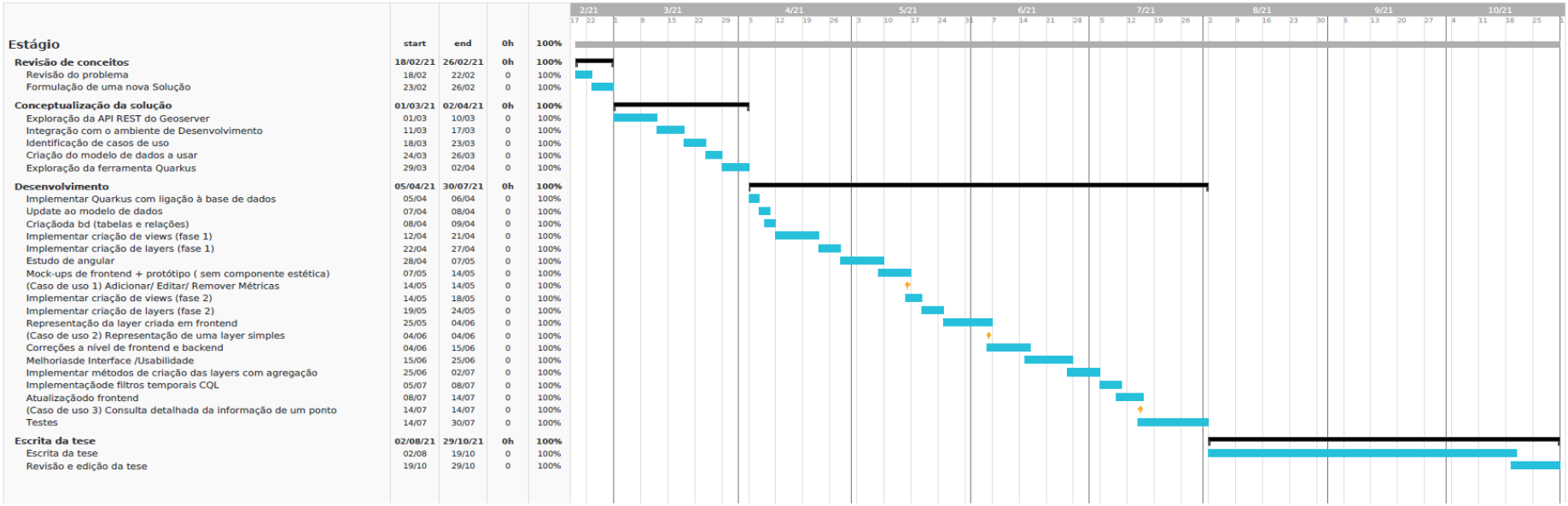


Figura 57 - *Endpoint* para remoção do método de agregação utilizado na associação da métrica ao geo.

Apêndice C

No Apêndice C é apresentado o Mapa de Gantt relativo ao segundo semestre, em versão "alargada".



Estágio	start	end	0h	100%
Revisão de conceitos	**18/02/21**	**26/02/21**	**0h**	**100%**
Revisão do problema	18/02	22/02	0	100%
Formulação de uma nova Solução	23/02	26/02	0	100%
Conceptualização da solução	**01/03/21**	**02/04/21**	**0h**	**100%**
Exploração da API REST do Geoserver	01/03	10/03	0	100%
Integração com o ambiente de Desenvolvimento	11/03	17/03	0	100%
Identificação de casos de uso	18/03	23/03	0	100%
Criação do modelo de dados a usar	24/03	26/03	0	100%
Exploração da ferramenta Quarkus	29/03	02/04	0	100%
Desenvolvimento	**05/04/21**	**30/07/21**	**0h**	**100%**
Implementar Quarkus com ligação à base de dados	05/04	06/04	0	100%
Update ao modelo de dados	07/04	08/04	0	100%
Criaçãoda bd (tabelas e relações)	08/04	09/04	0	100%
Implementar criação de views (fase 1)	12/04	21/04	0	100%
Implementar criação de layers (fase 1)	22/04	27/04	0	100%
Estudo de angular	28/04	07/05	0	100%
Mock-ups de frontend + protótipo (sem componente estética)	07/05	14/05	0	100%
(Caso de uso 1) Adicionar/ Editar/ Remover Métricas	14/05	14/05	0	100%
Implementar criação de views (fase 2)	14/05	18/05	0	100%
Implementar criação de layers (fase 2)	19/05	24/05	0	100%
Representação da layer criada em frontend	25/05	04/06	0	100%
(Caso de uso 2) Representação de uma layer simples	04/06	04/06	0	100%
Correções a nível de frontend e backend	04/06	15/06	0	100%
Melhoriasde Interface /Usabilidade	15/06	25/06	0	100%
Implementar métodos de criação das layers com agregação	25/06	02/07	0	100%
Implementaçãode filtros temporais CQL	05/07	08/07	0	100%
Atualizaçãodo frontend	08/07	14/07	0	100%
(Caso de uso 3) Consulta detalhada da informação de um ponto	14/07	14/07	0	100%
Testes	14/07	30/07	0	100%
Escrita da tese	**02/08/21**	**29/10/21**	**0h**	**100%**
Escrita da tese	02/08	19/10	0	100%
Revisão e edição da tese	19/10	29/10	0	100%

Figura 58- Mapa de Gantt para o segundo semestre.

Apêndice D

No Apêndice D é apresento o *Payload* utilizado para a criação das *layers* no GeoServer, contendo todos os parâmetros utilizados de modo a diminuir o máximo possível o tempo da tarefa.

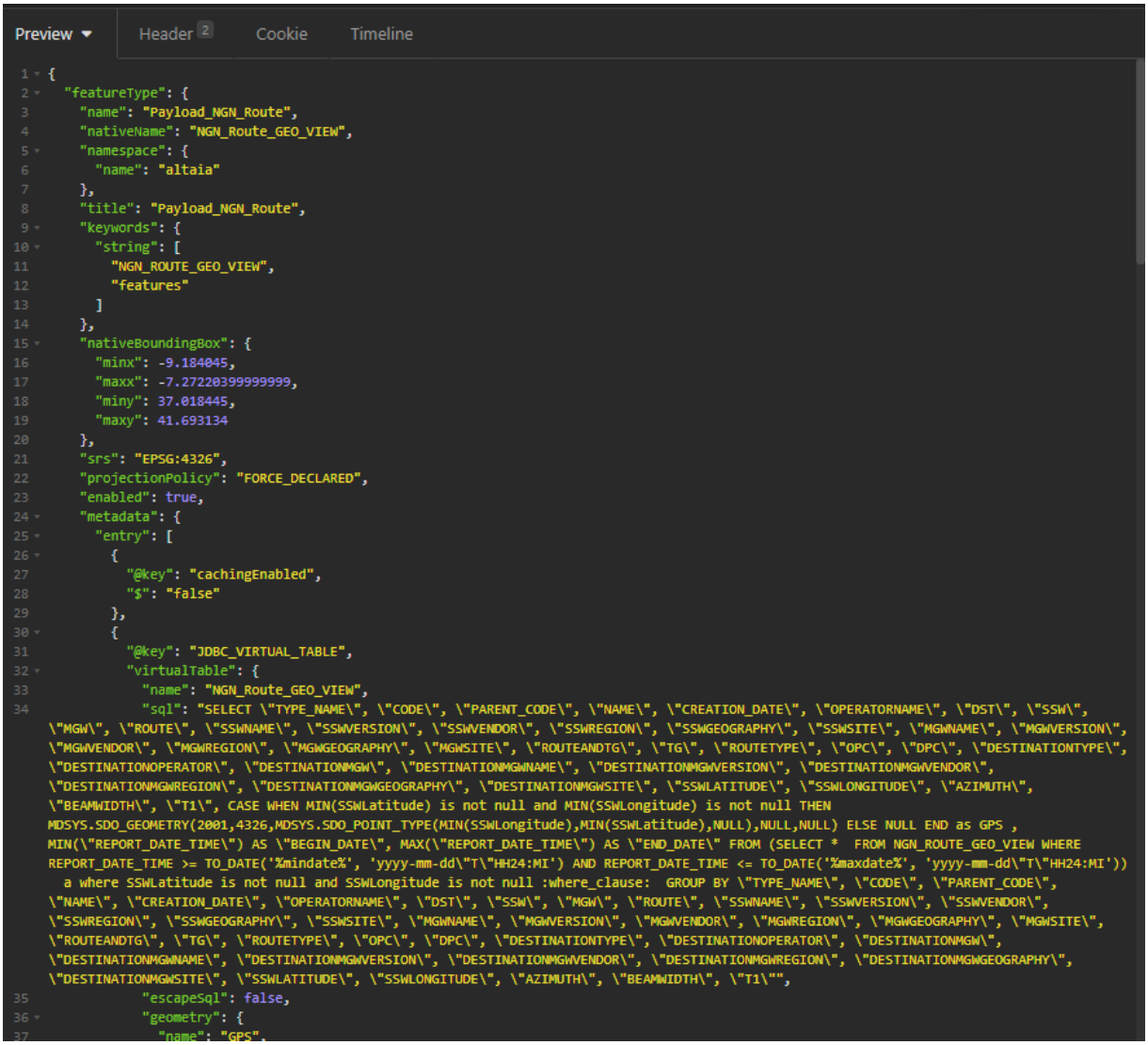


Figura 59 - Exemplo do *playload* utilizado para criar uma *layer*. (Parte 1)

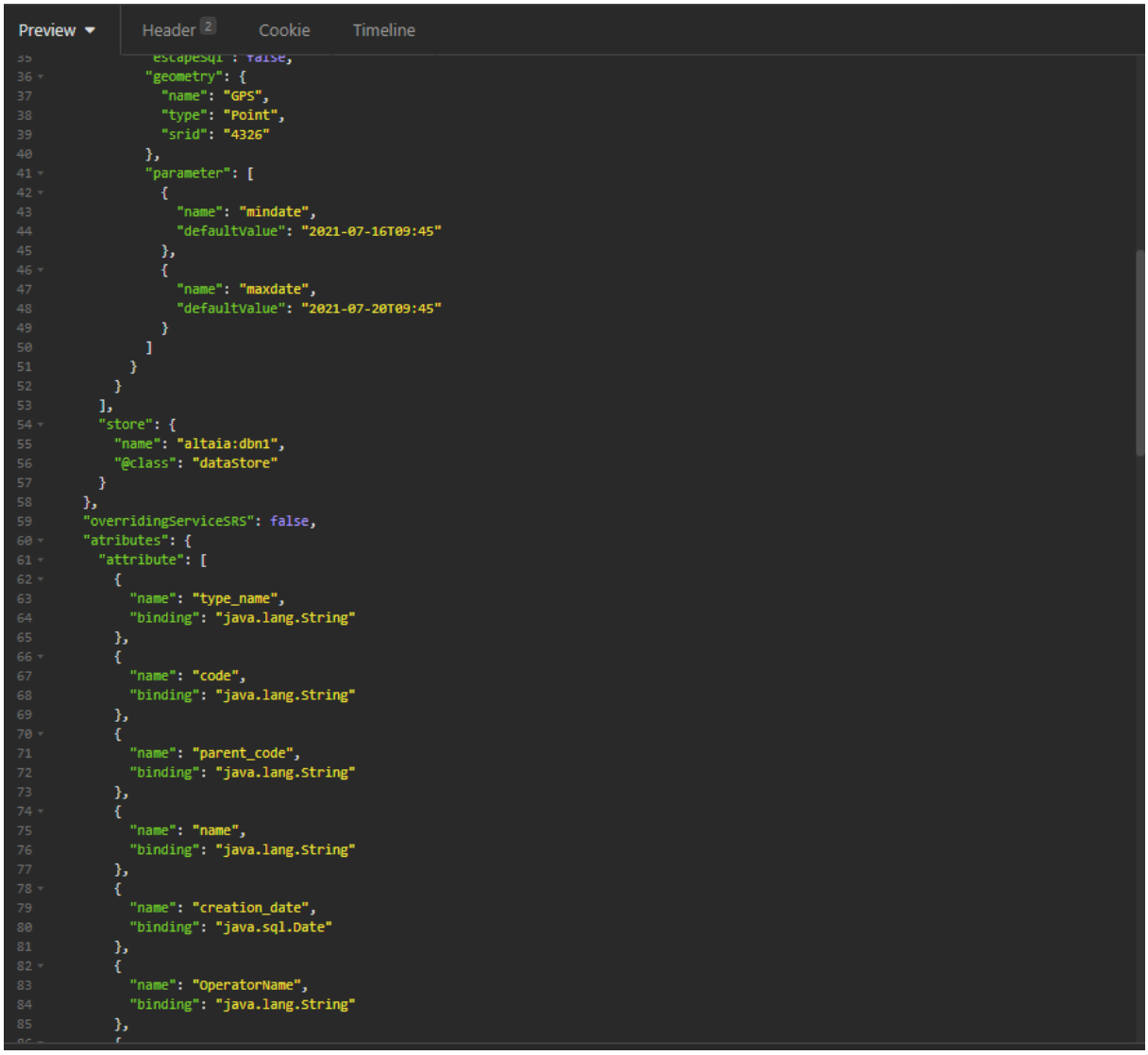


Figura 60 - Exemplo do *playload* utilizado para criar uma *layer*. (Parte 2)

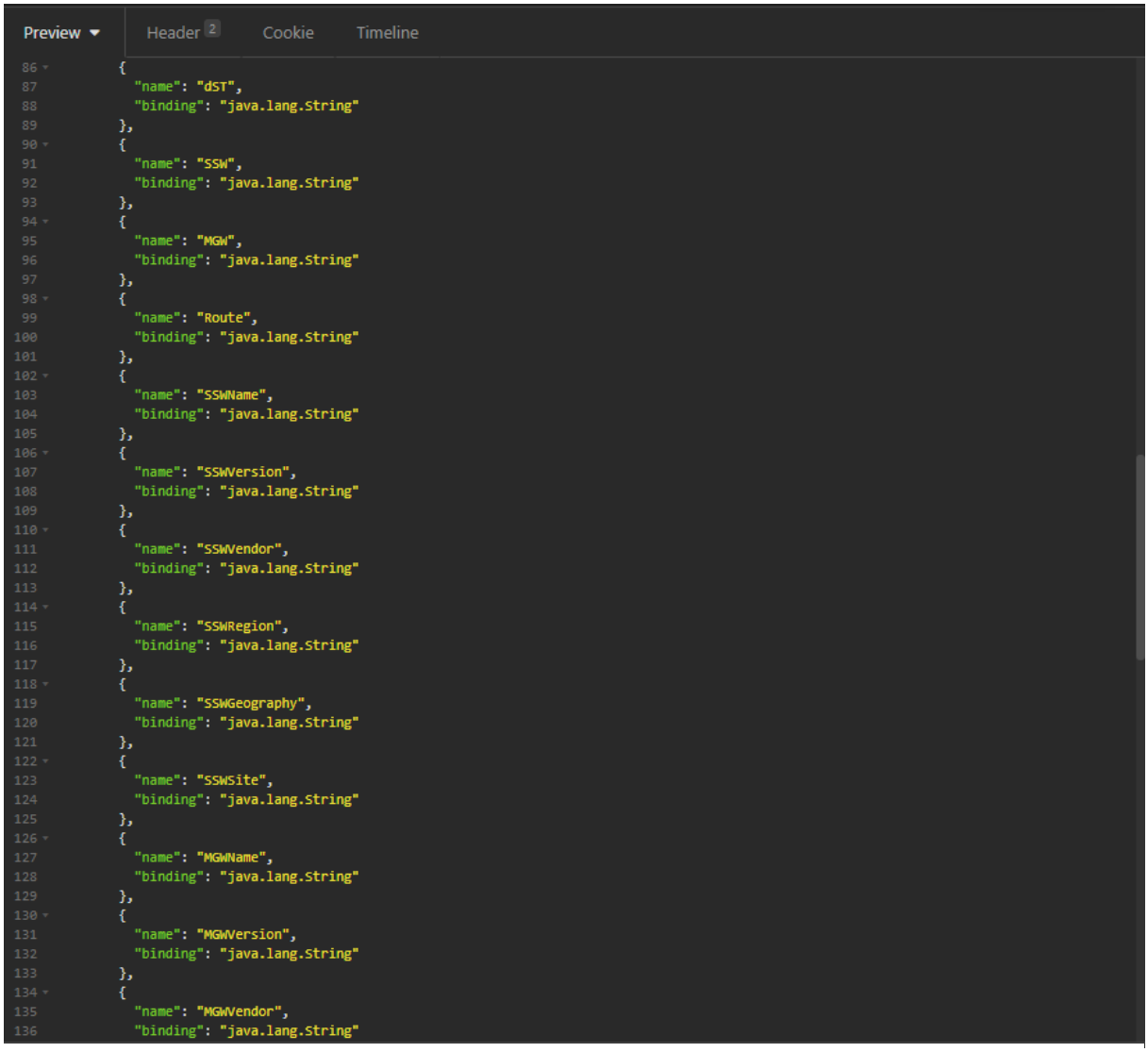


Figura 61 - Exemplo do *playload* utilizado para criar uma *layer*. (Parte 3)

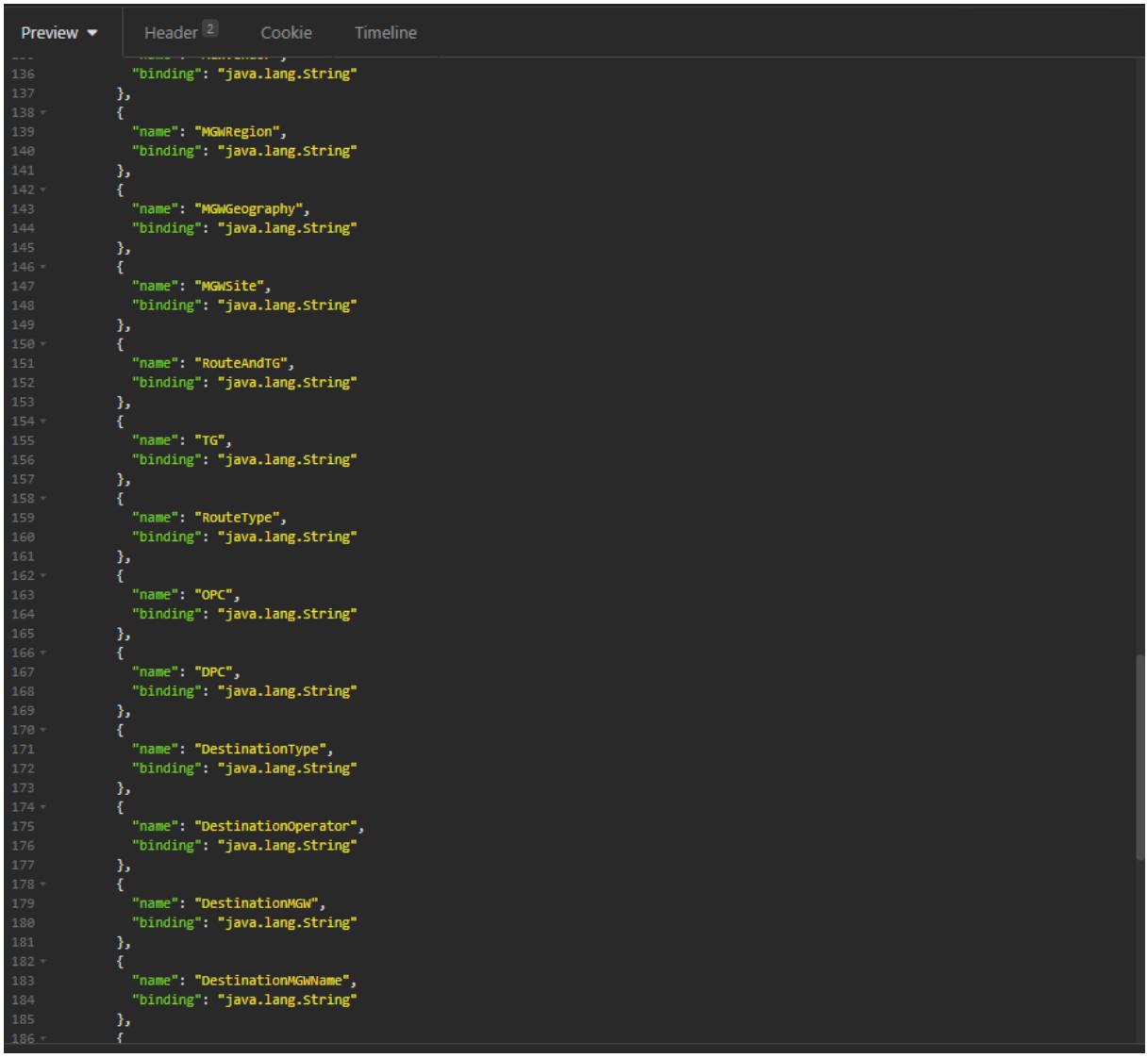


Figura 62 - Exemplo do *playload* utilizado para criar uma *layer*. (Parte 4)

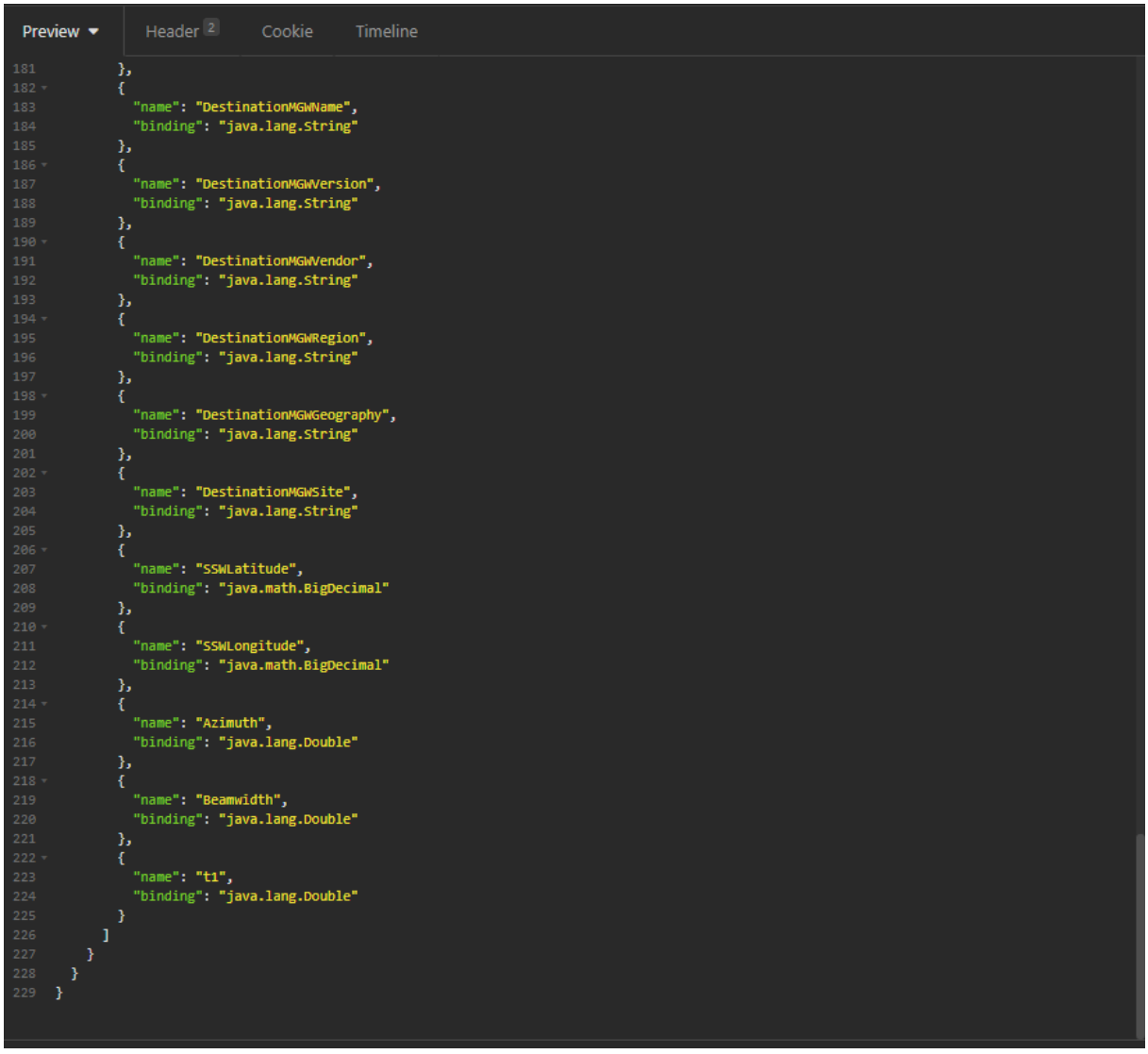


Figura 63 - Exemplo do *playload* utilizado para criar uma *layer*. (Parte 5)

Apêndice E

No Apêndice E são apresentados os passos para a execução do teste 1, criar um geo, assim como possíveis erros cometidos por parte do utilizador.

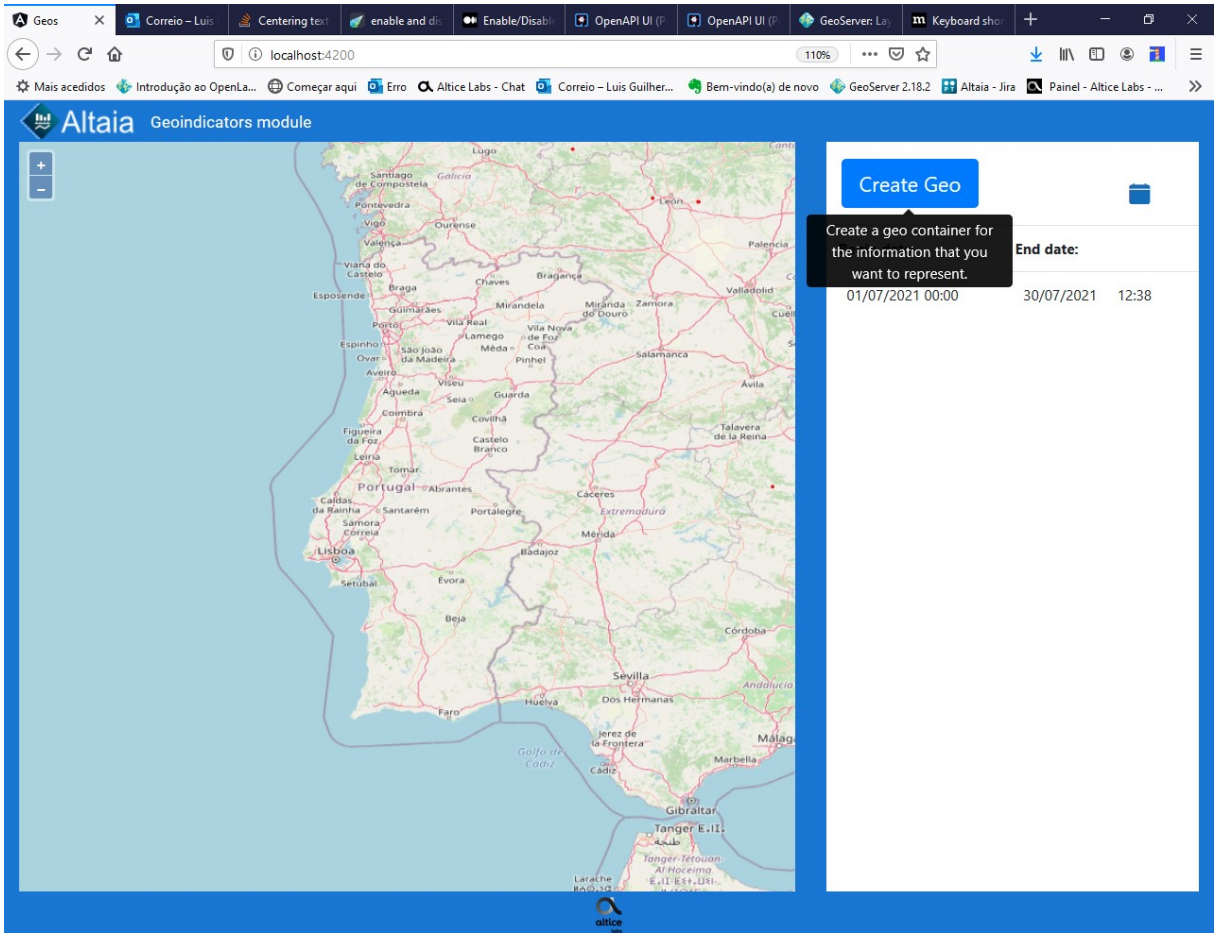


Figura 64 - Teste 1, criar um geo. Passo 1 de 4. Abrir o endereço “localhost:4200.

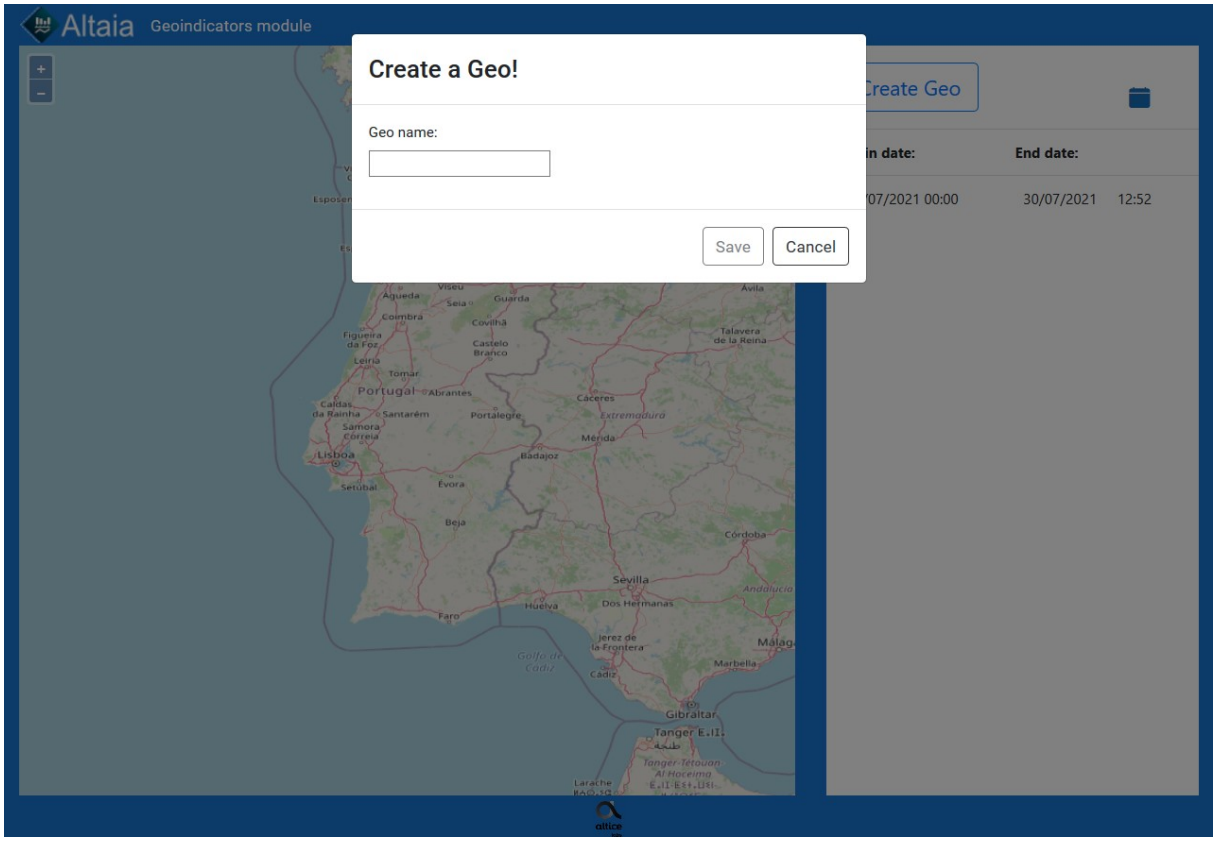


Figura 65- Teste 1, criar um geo. Passo 2 de 4. Clicar no botão “Add Geo”

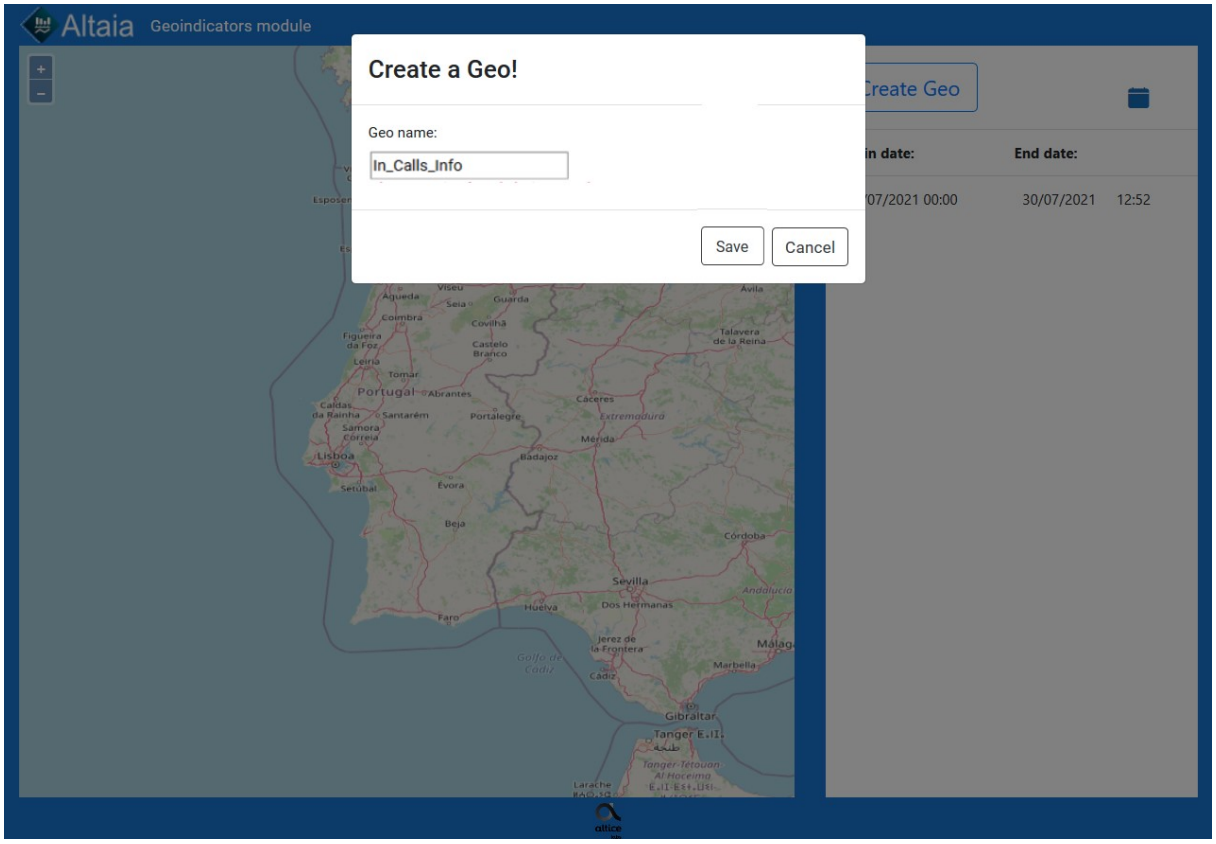


Figura 66 - Teste 1, criar um geo. Passo 3 de 4. Introduzir o nome para o Geo

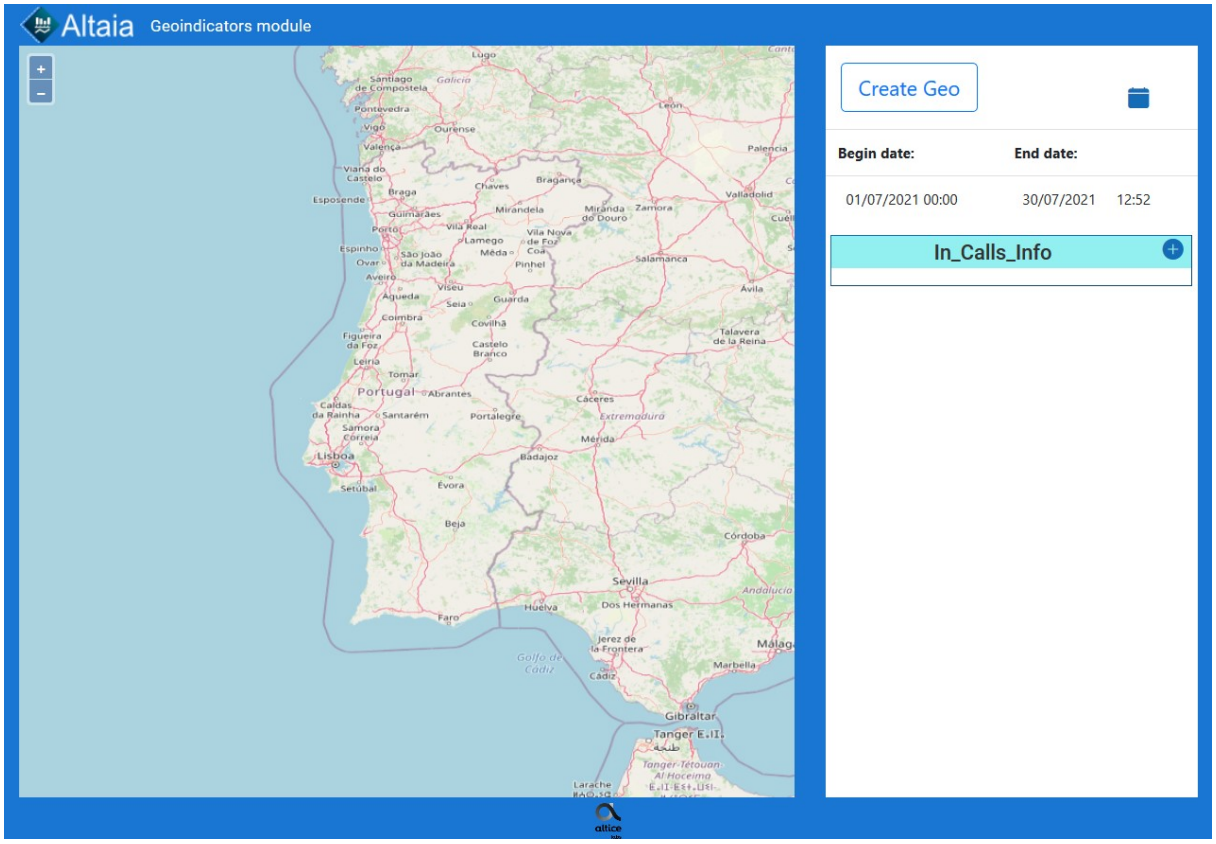


Figura 67 - Teste 1, criar um geo. Passo 4 de 4. Clicar em “Save”

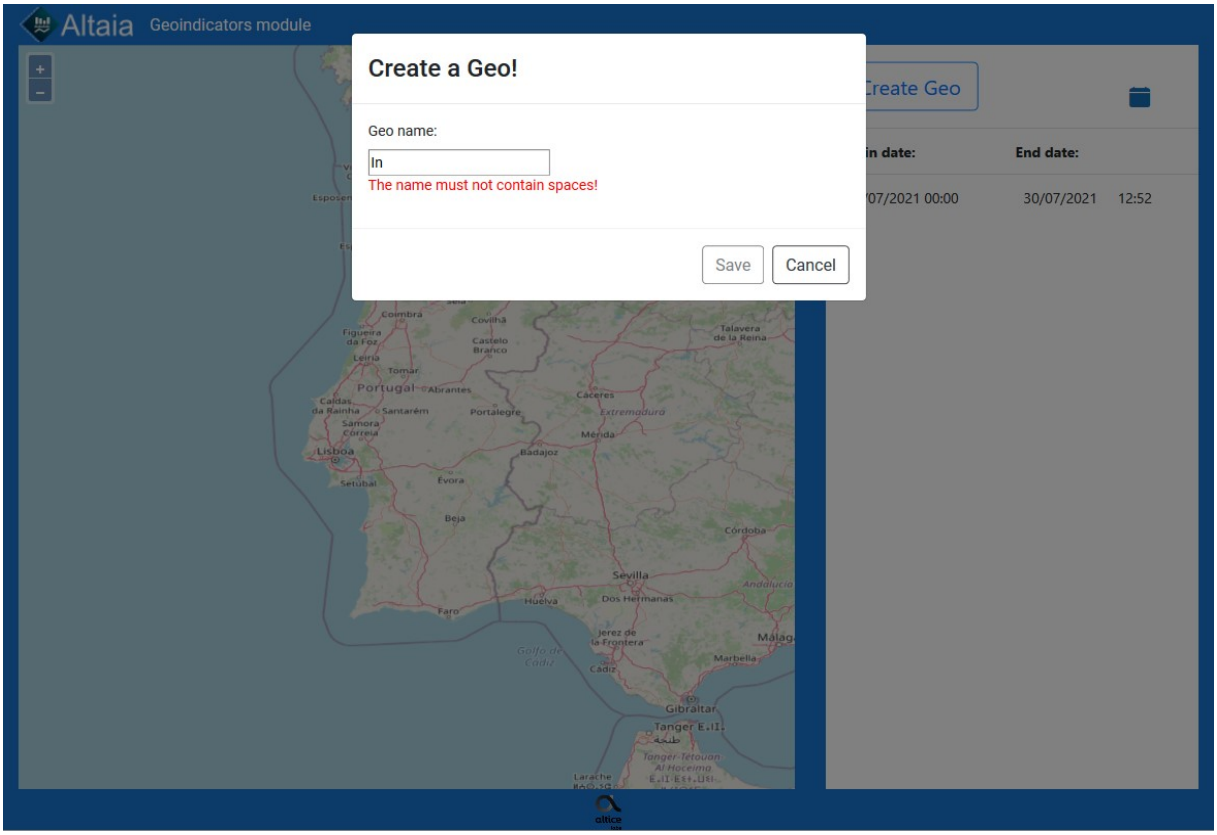


Figura 68 Criar geo - Possível erro ao introduzir espaços no nome.

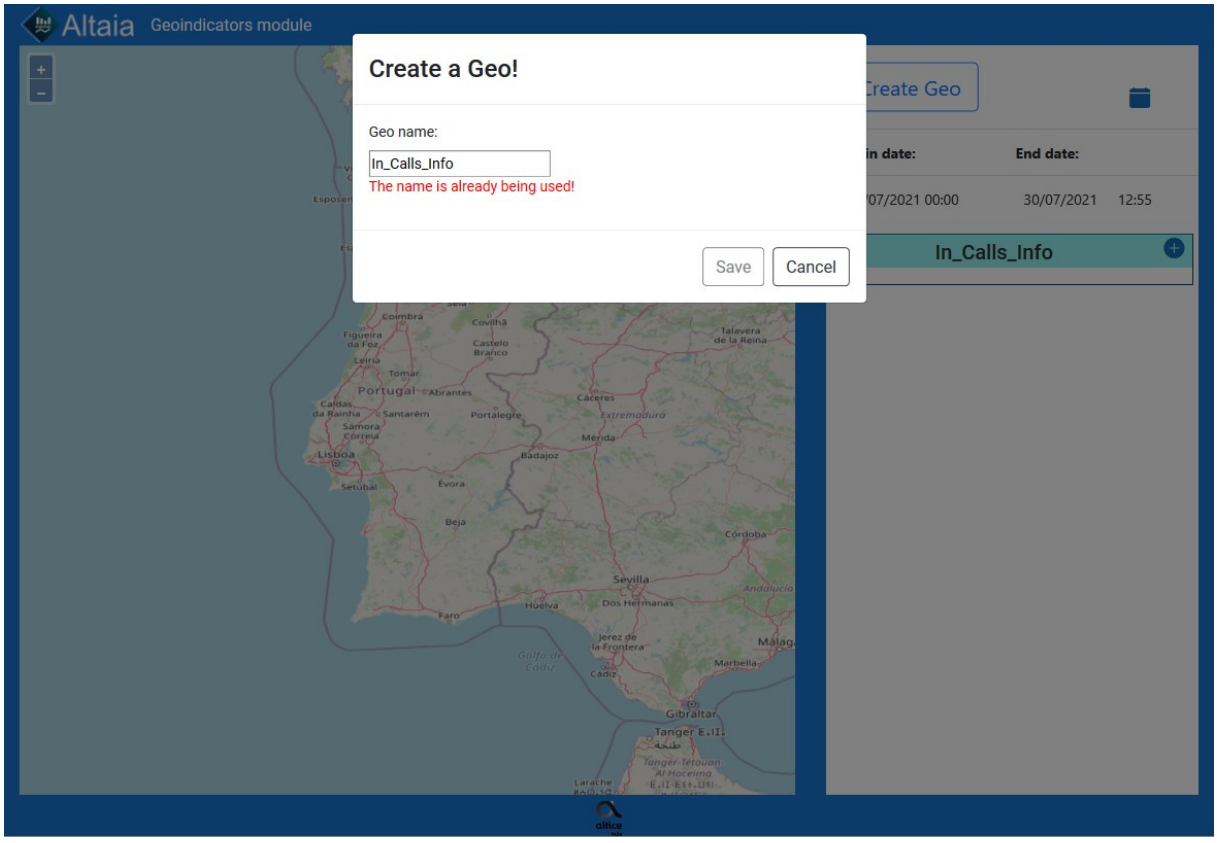


Figura 69 - Criar geo. Possível erro ao tentar criar um geo com um nome já existente.

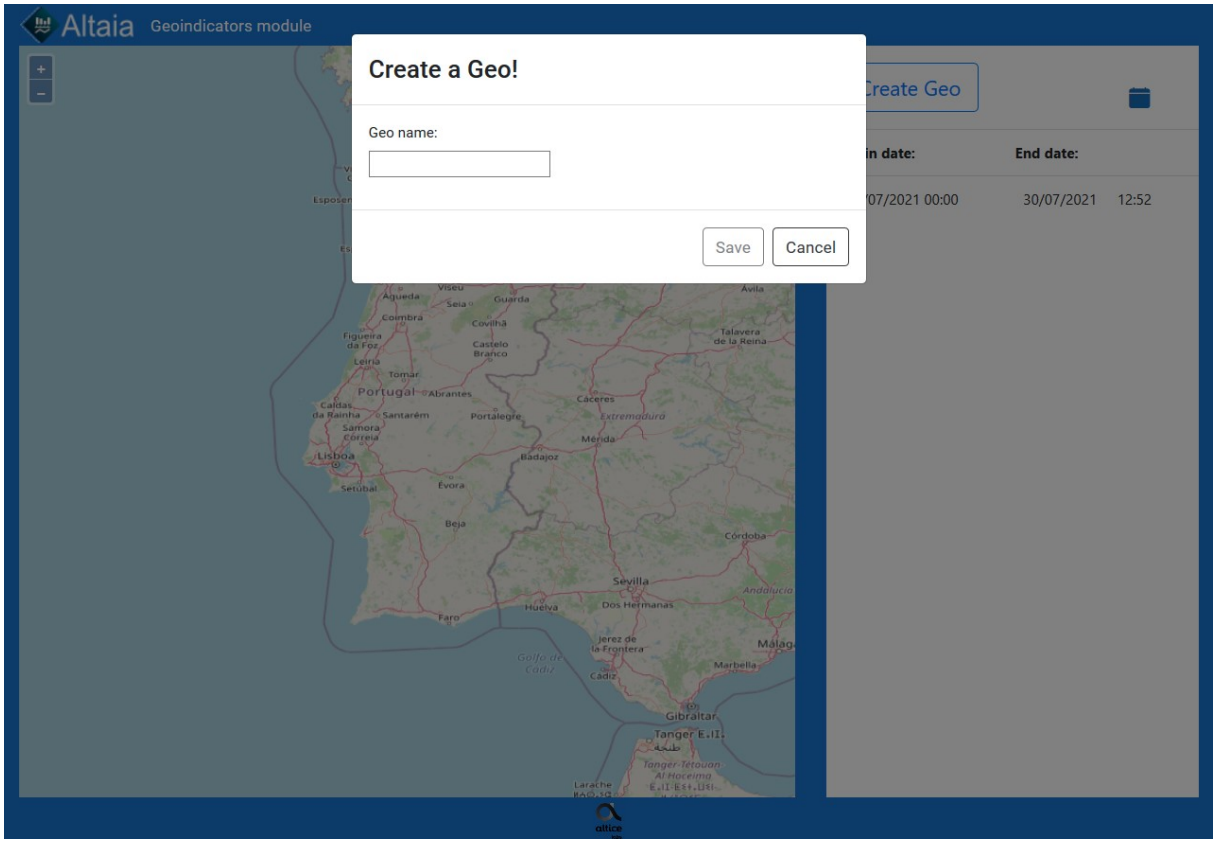


Figura 70 - Criar geo. Possivel erro ao tentar criar um geo sem nome.

Apêndice F

No Apêndice F são apresentados os passos para a execução do teste 2, associar um tipo de entidade a um geo, assim como possíveis erros cometidos pelo utilizador.

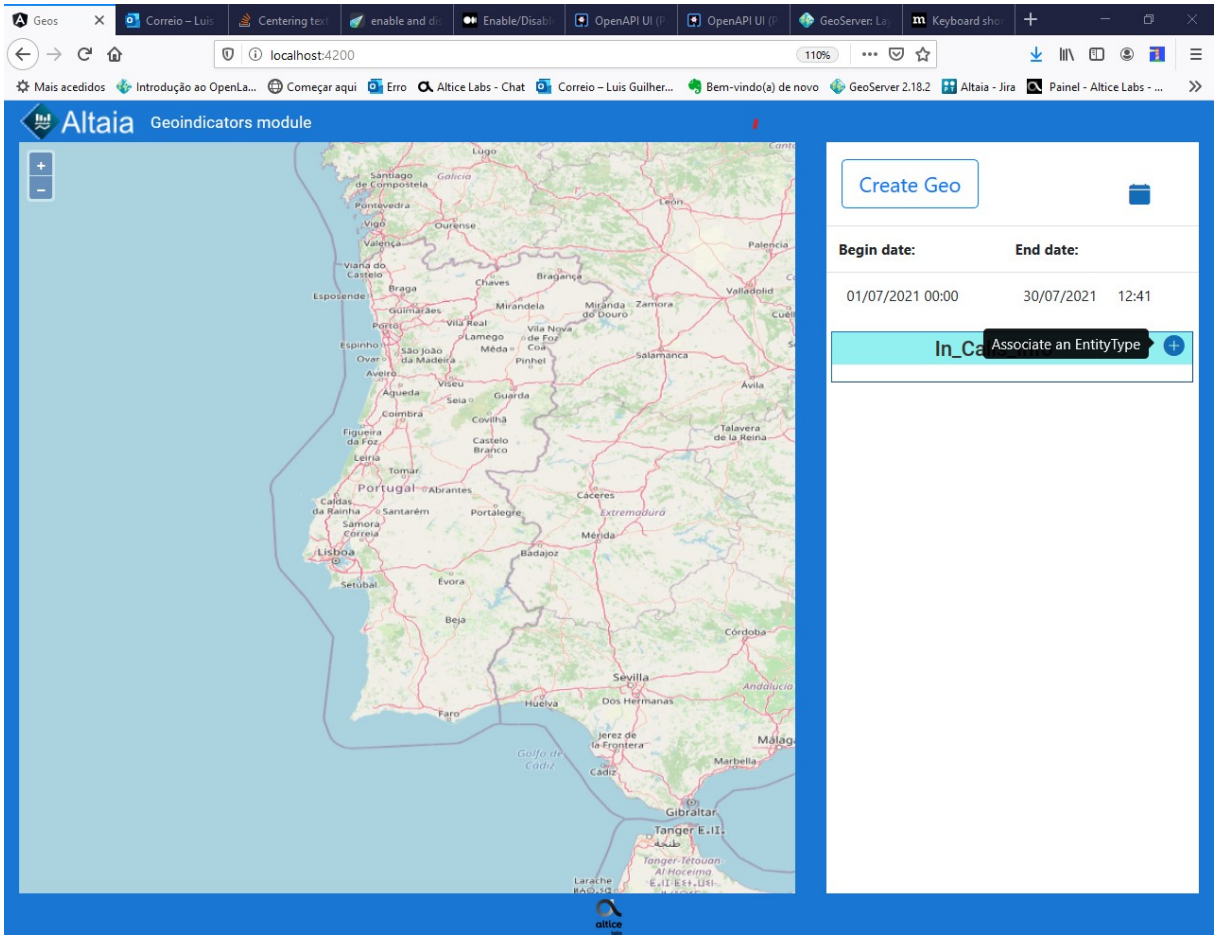


Figura 71 -Teste 2, associar um tipo de entidade a um geo. Passo 1 de 4, clicar no botão "+" ao lado do nome do geo.

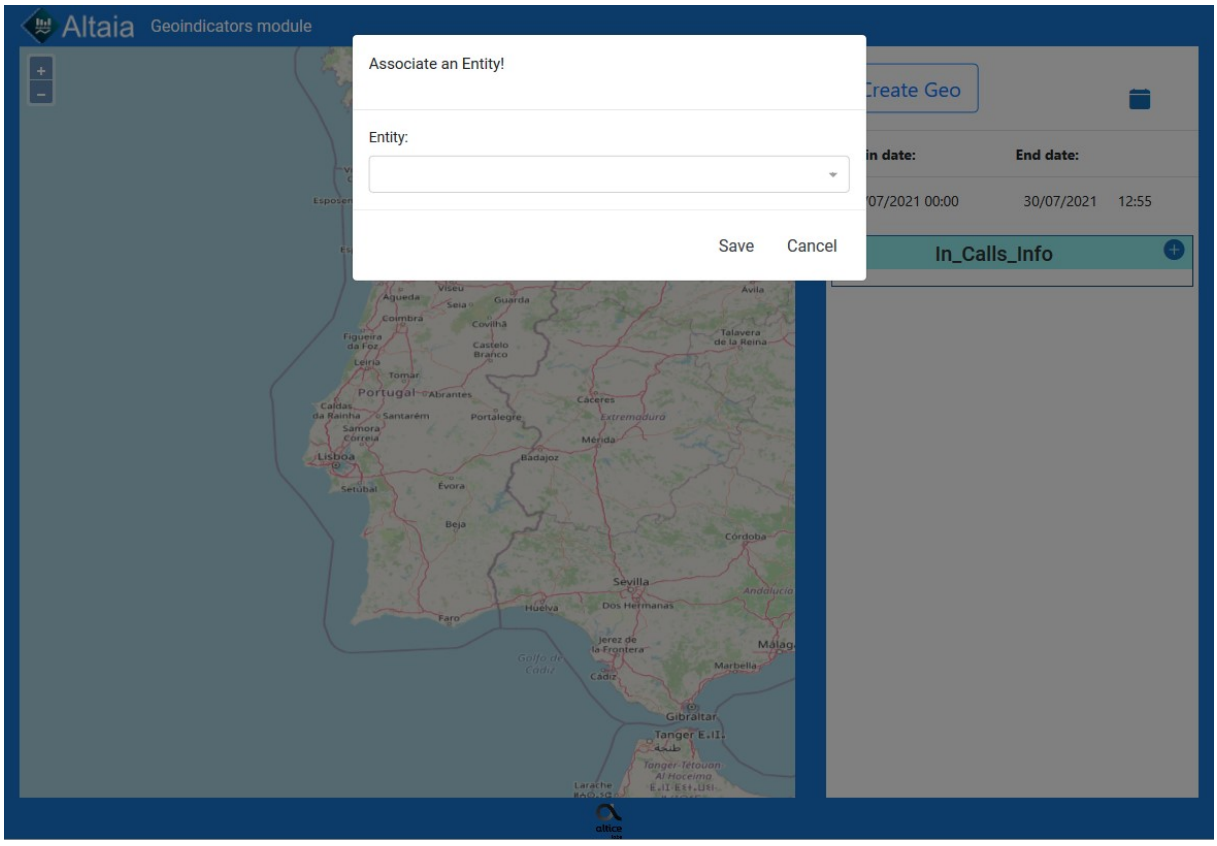


Figura 72 - Teste 2, associar um tipo de entidade a um geo. Modal para associação de tipos de entidade.

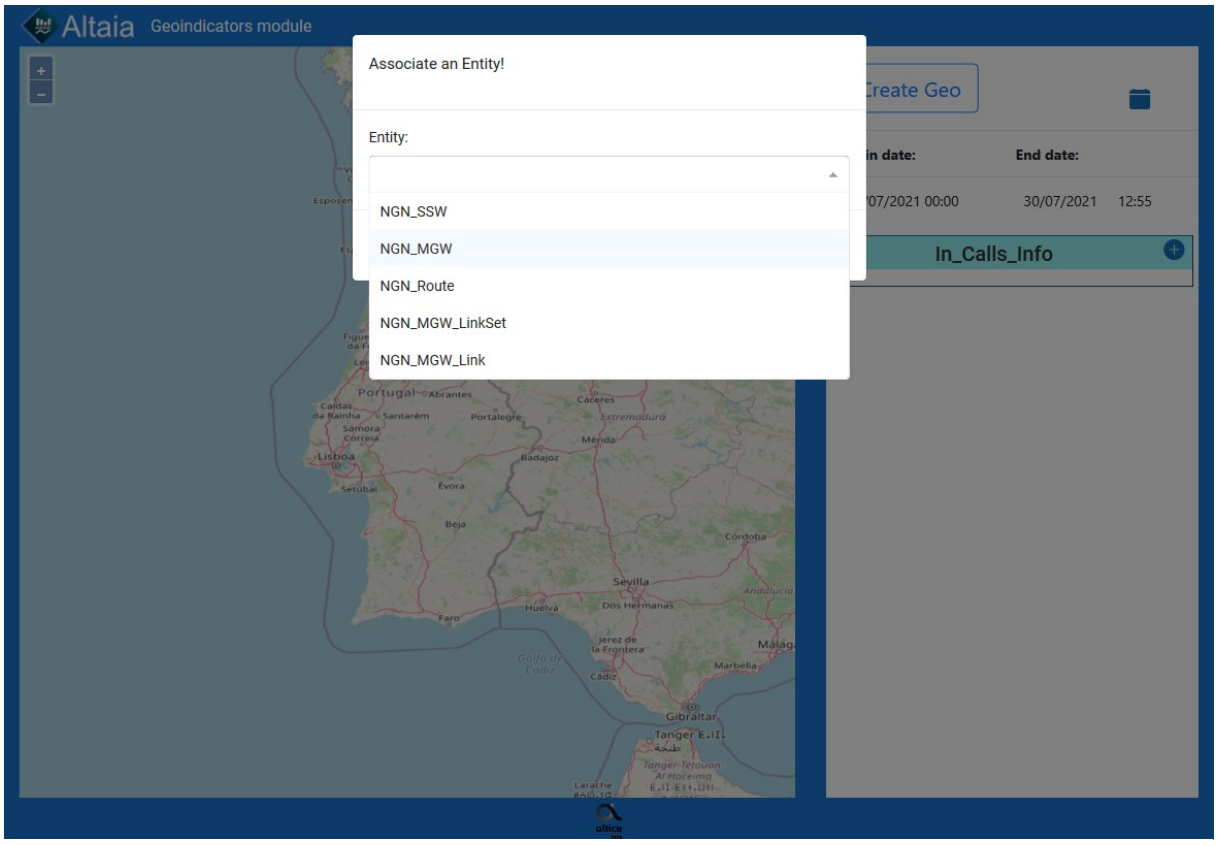


Figura 73 -Teste 2, associar um tipo de entidade a um geo. Passo 2 de 4, clicar na *droplist*.

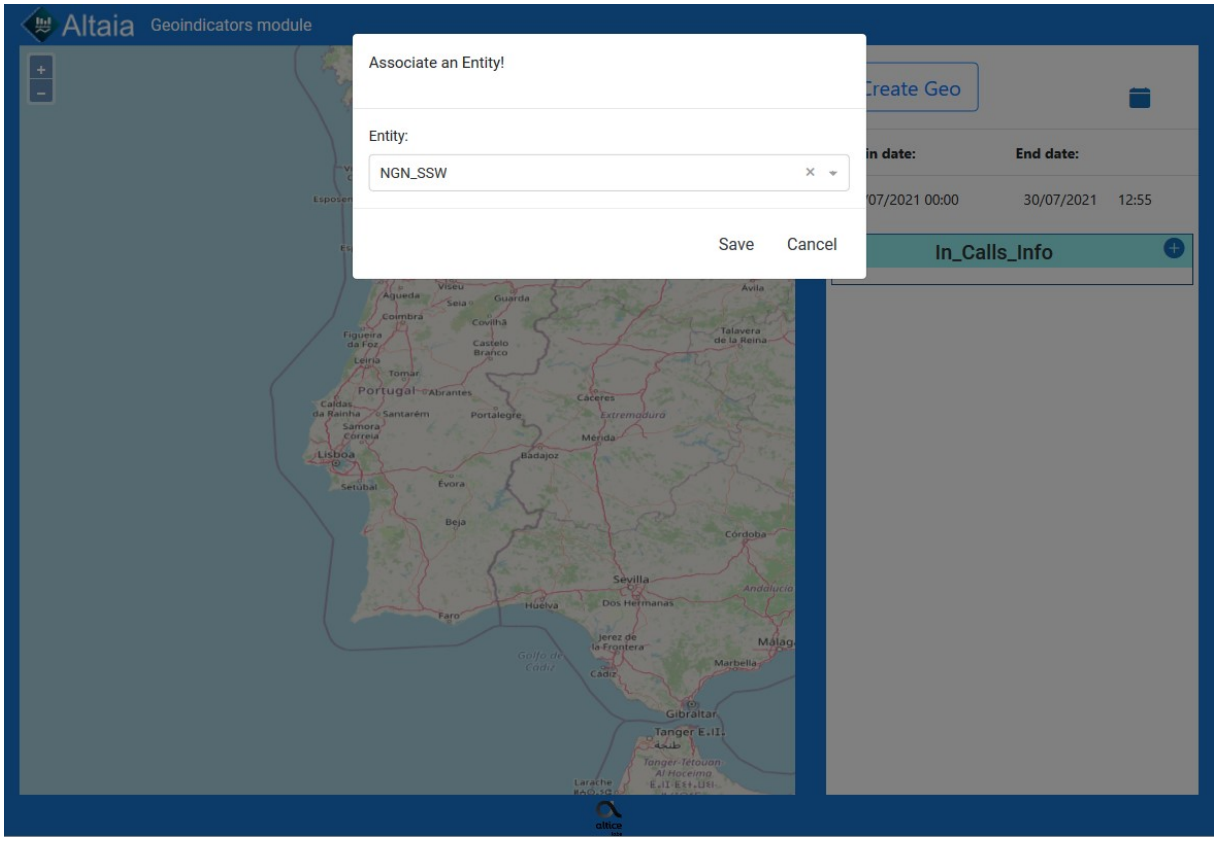


Figura 74 - Teste 2, associar um tipo de entidade a um geo. Passo 3 de 4, selecionar um tipo de entidade

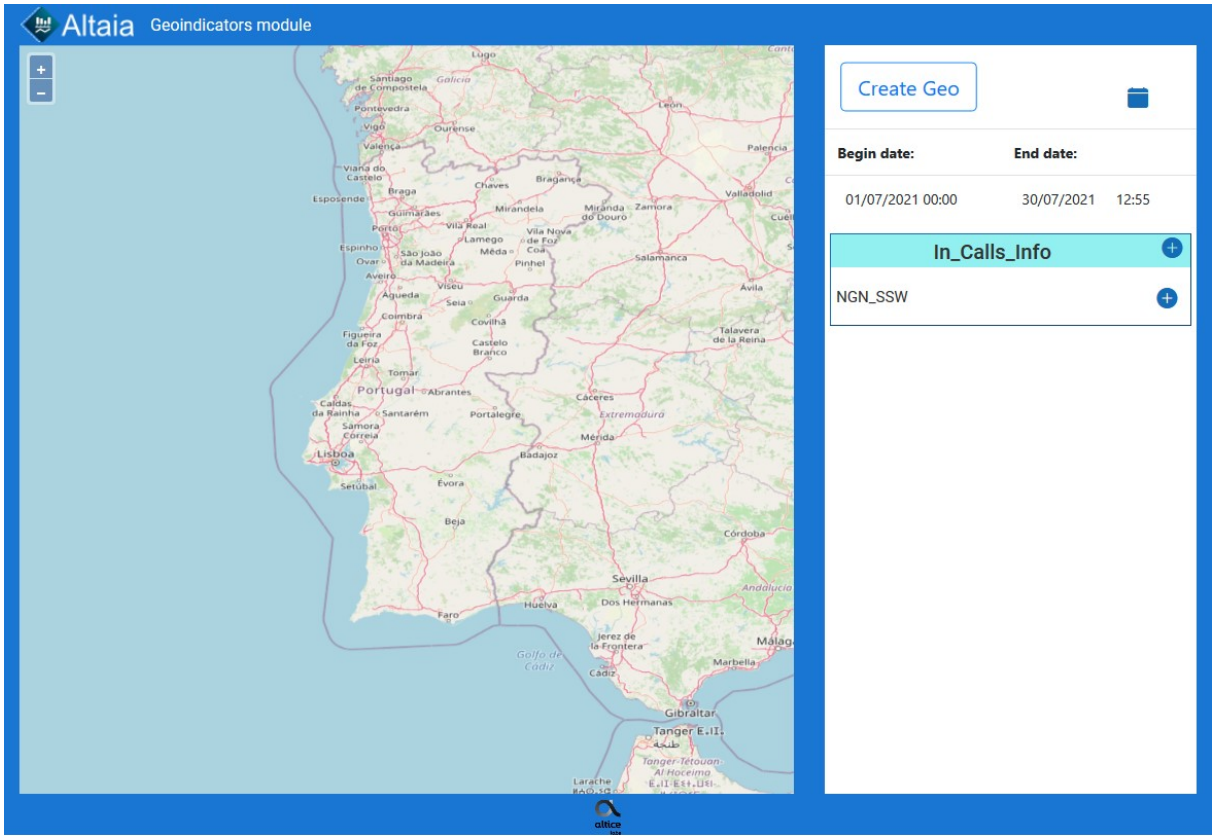


Figura 75 - Teste 2, associar um tipo de entidade a um geo. Passo 4 de 4, clicar em “Save”

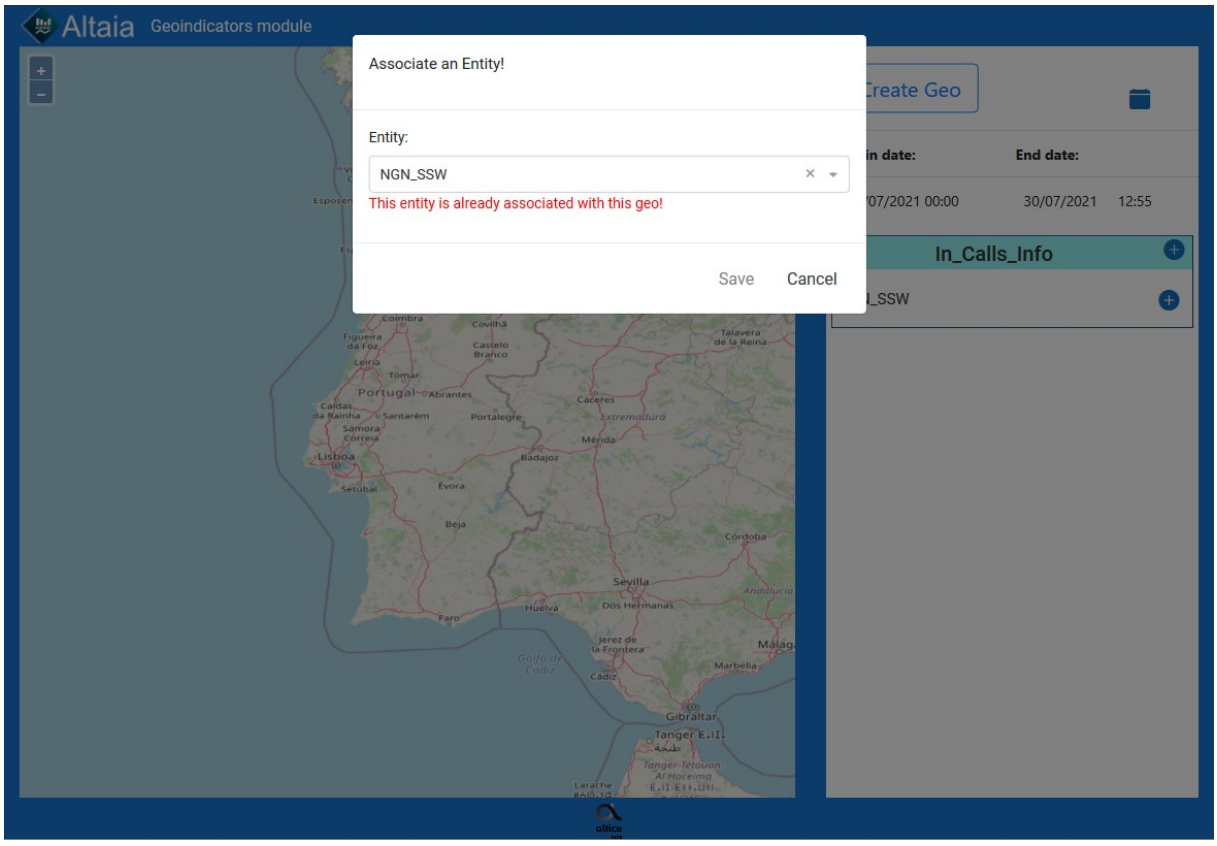


Figura 76 - Associar um tipo de entidade a um geo. Possível erro ao tentar associar novamente o mesmo tipo de entidade.

Apêndice G

No Apêndice G são apresentados os passos para a execução do teste 3, associar uma métrica ao geo.

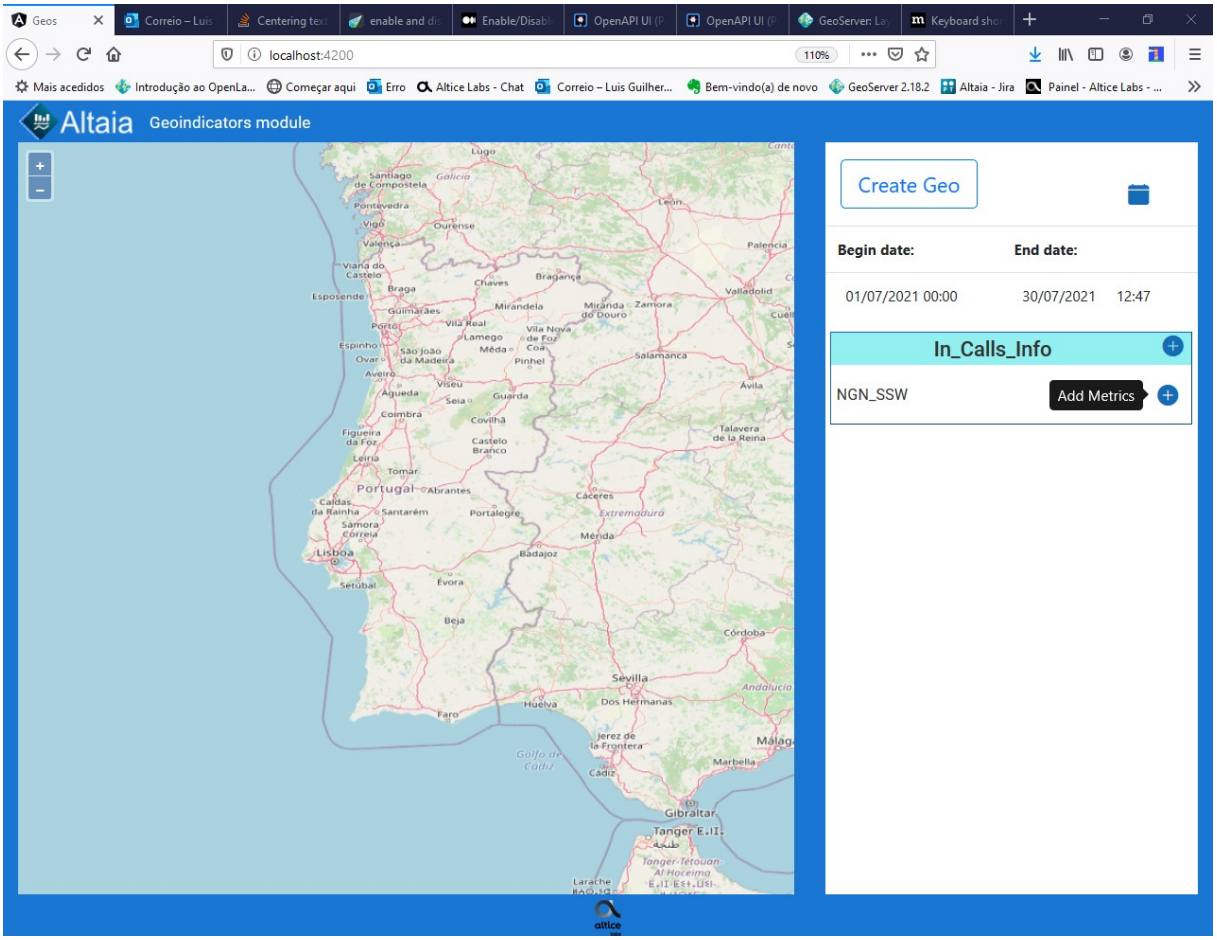


Figura 77 – Teste 3, associar uma métrica ao geo. Passo 1 de 7, clicar no botão "+" ao lado do nome do tipo de entidade.

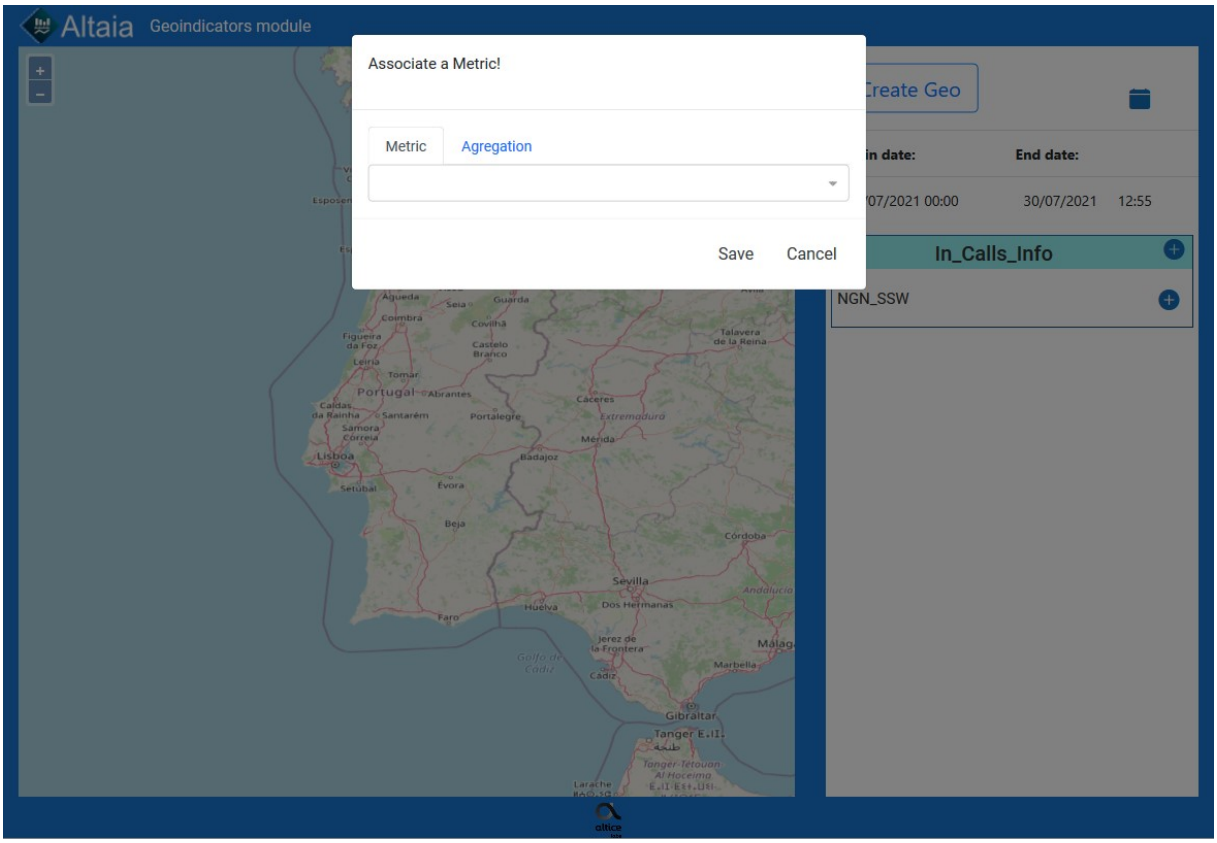


Figura 78 - Teste 3, associar uma métrica ao geo. Modal para associação de métricas.

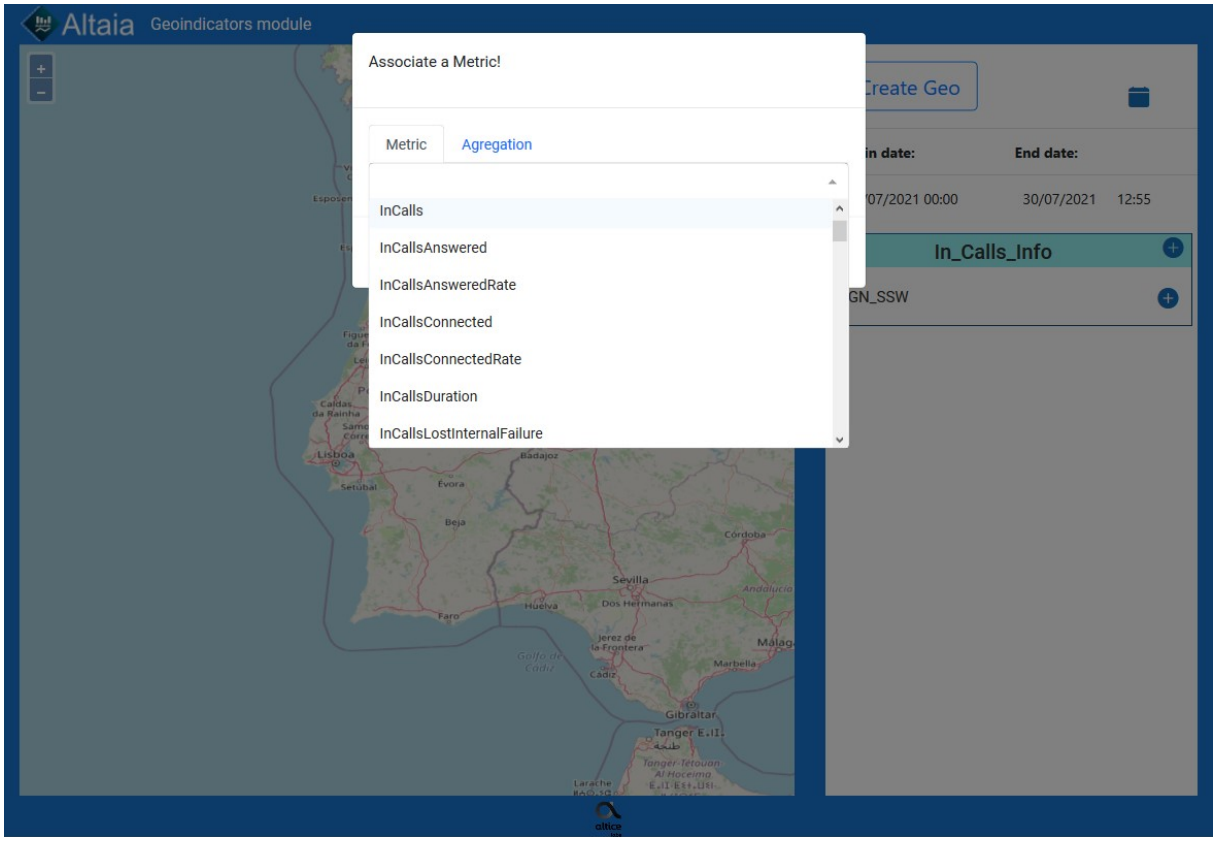


Figura 79 - Teste 3, associar uma métrica ao geo. Passo 2 de 7, clicar na *droplist*.

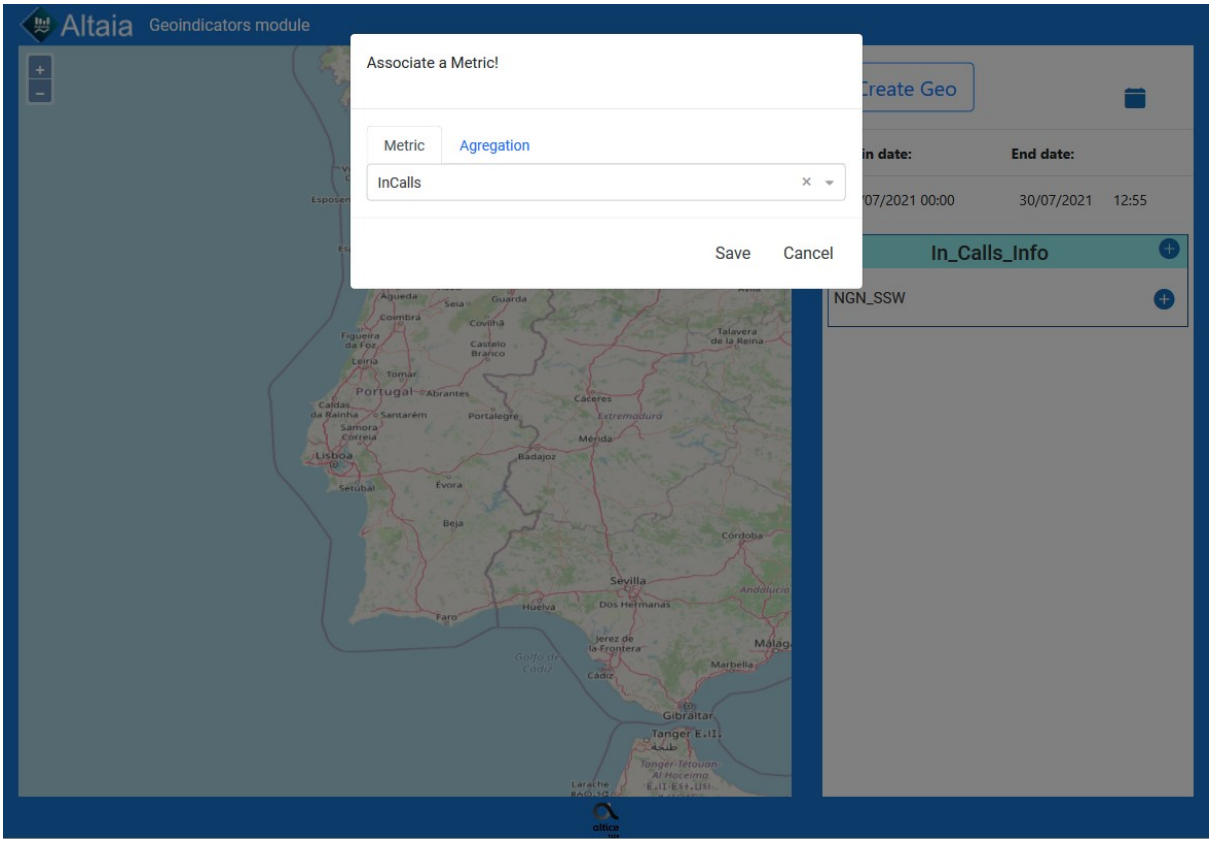


Figura 80 - Teste 3, associar uma métrica ao geo. Passo 3 de 7, selecionar uma métrica.

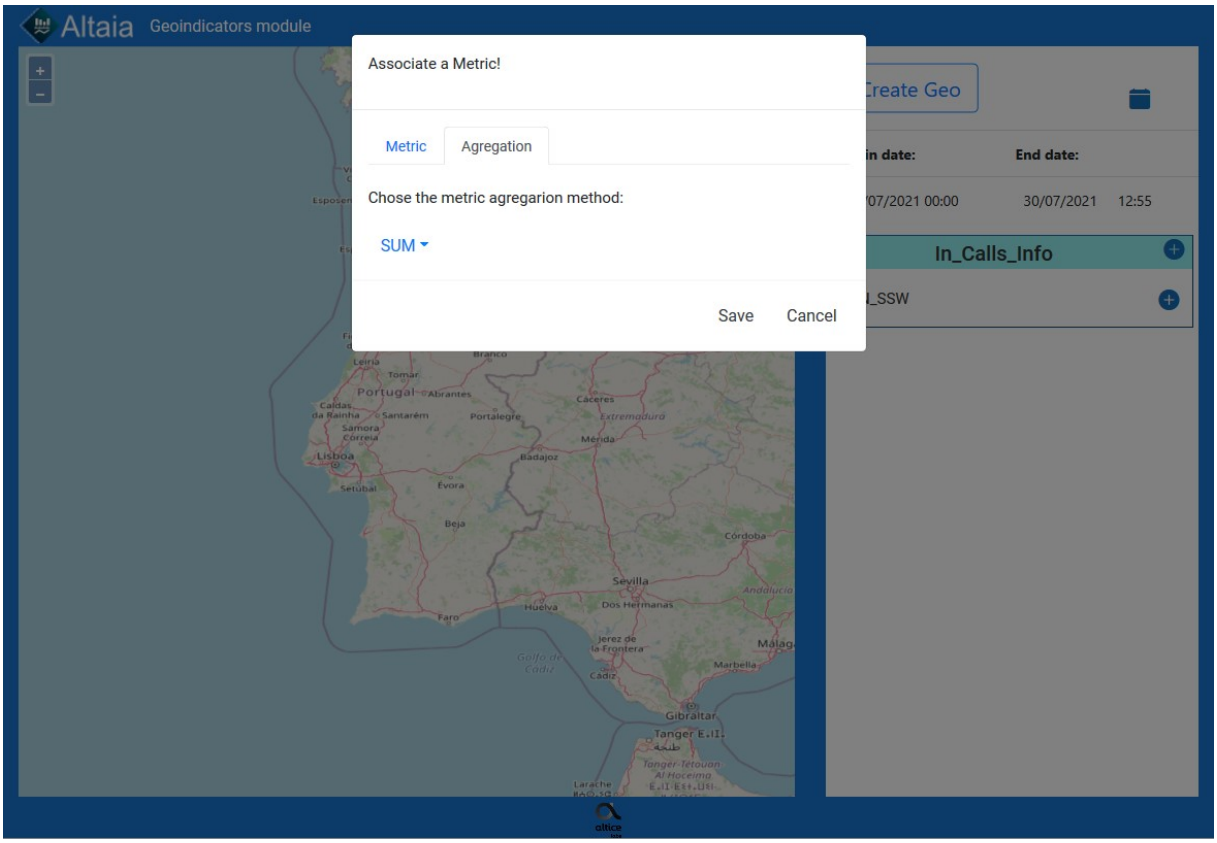


Figura 81 - Teste 3, associar uma métrica ao geo. Passo 4 de 7, clicar na aba “Agregation”.

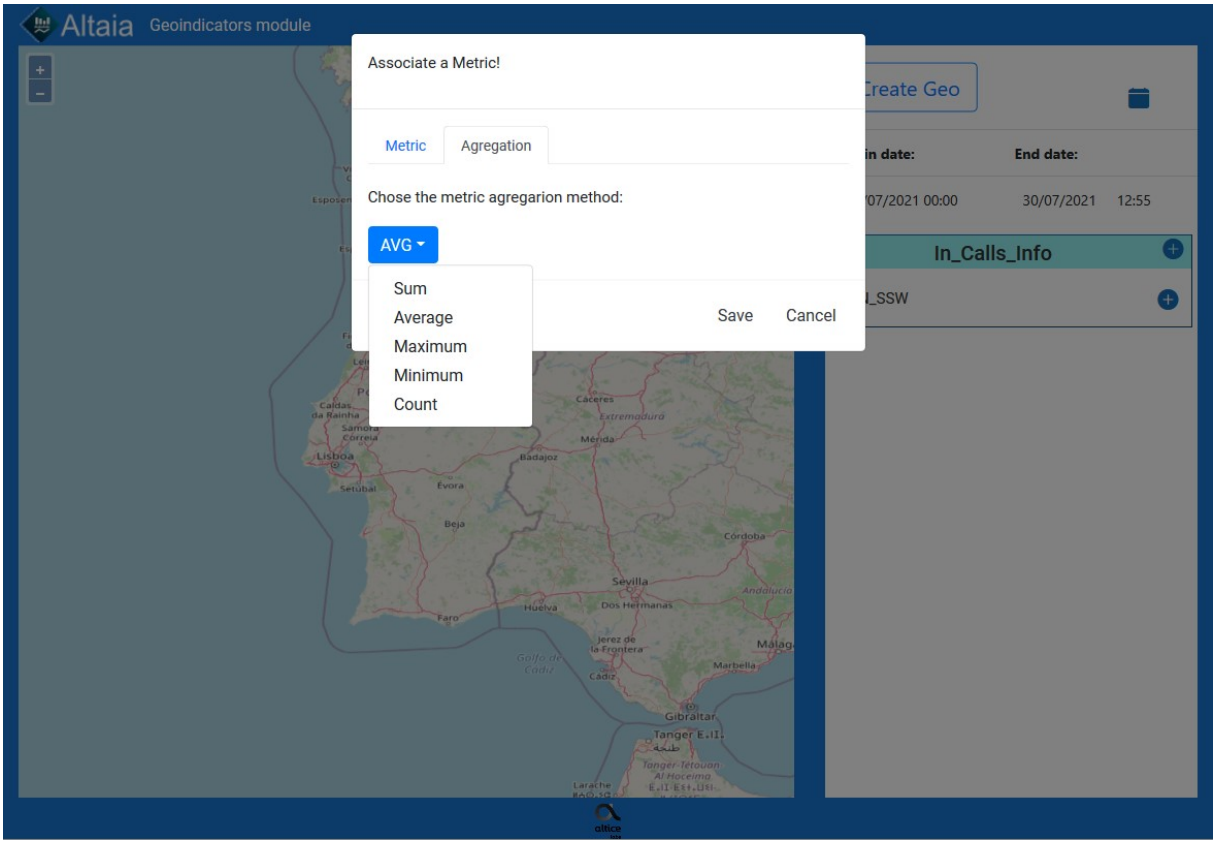


Figura 82 - Teste 3, associar uma métrica ao geo. Passo 5 de 7, clicar na *droplist*.

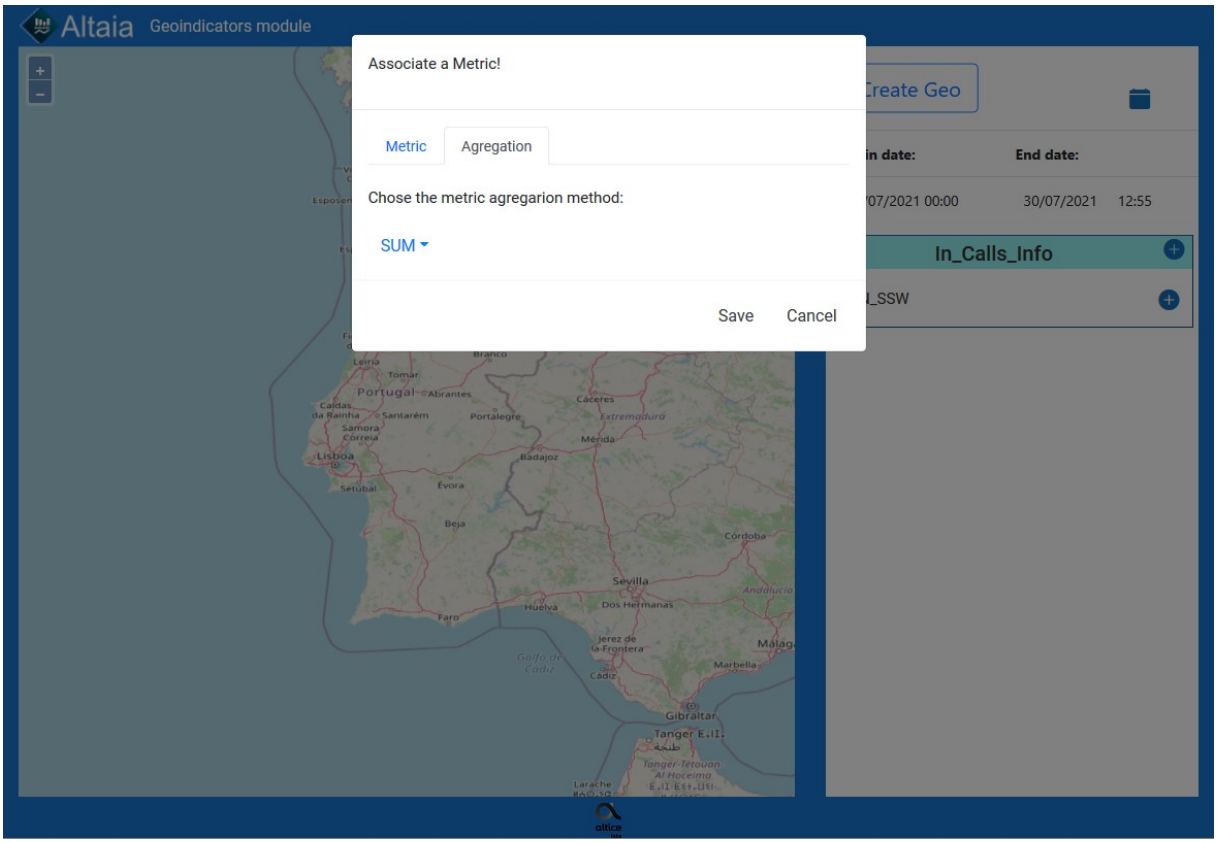


Figura 83 - Teste 3, associar uma métrica ao geo. Passo 6 de 7, selecionar uma agregação.

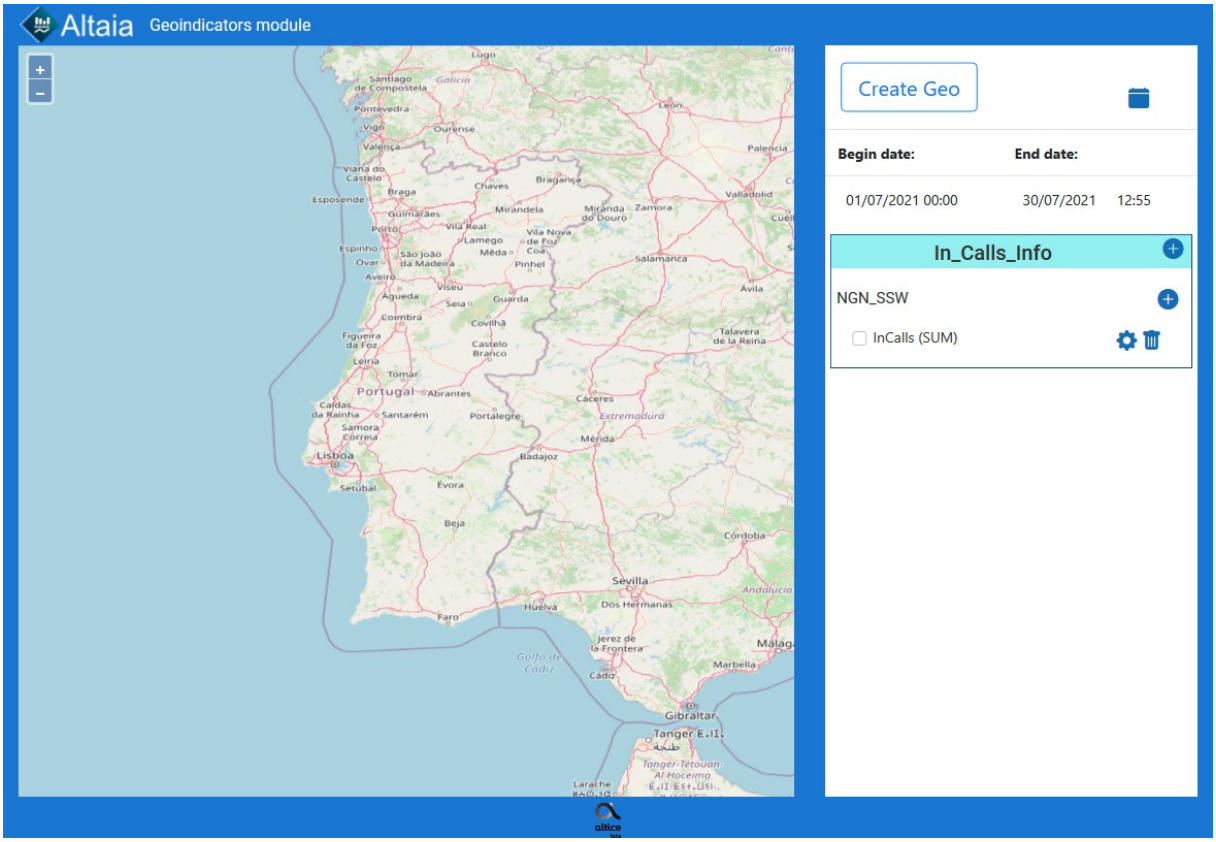


Figura 84 - Teste 3, associar uma métrica ao geo. Passo 7 de 7, clicar em “Save”.

Apêndice H

No Apêndice H são apresentados os passos para a execução do teste, localizar no tempo a visualização pretendida.

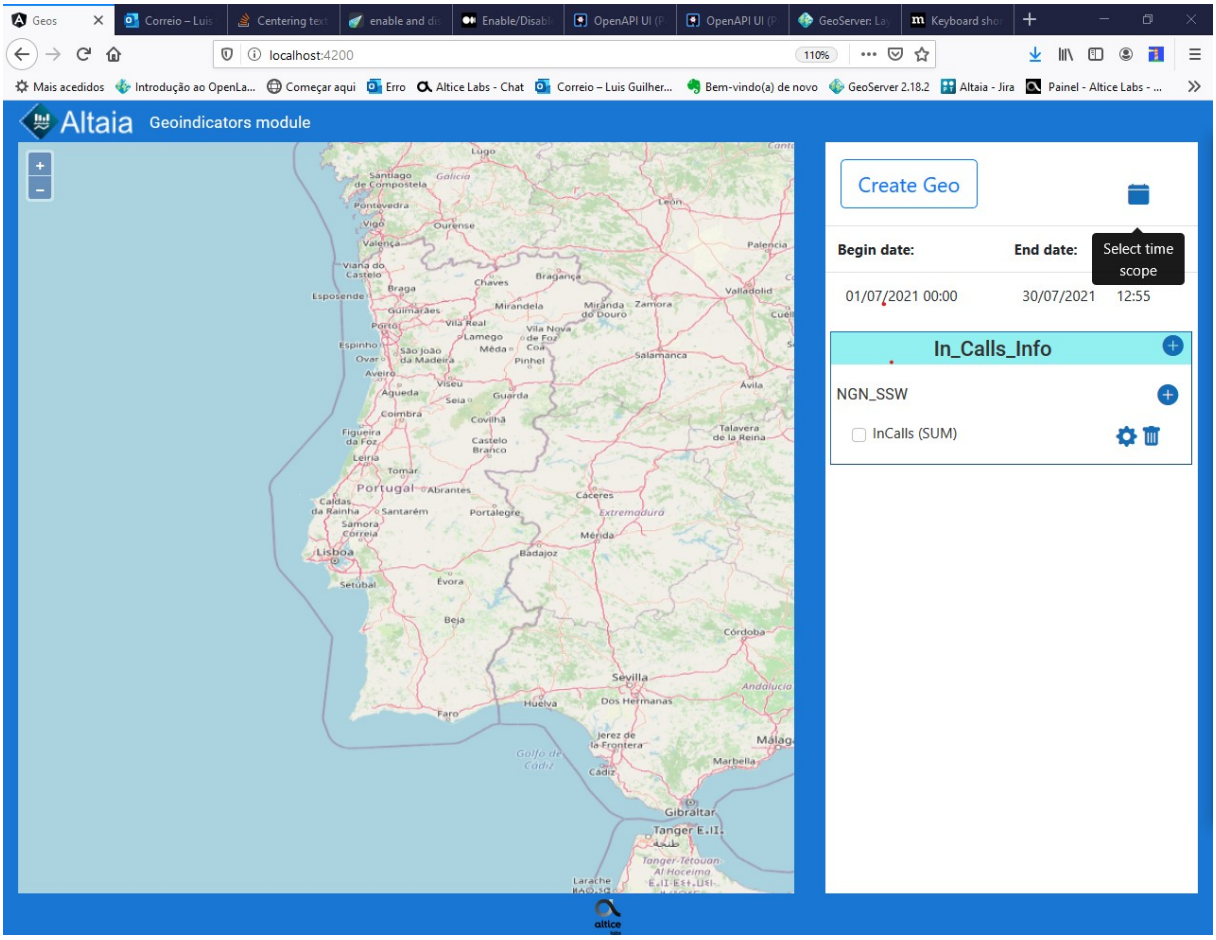


Figura 85 - Teste 4, localizar no tempo a visualização pretendida. Passo 1 de 3, clicar no botão do calendário ao lado do botão para criar um geo.

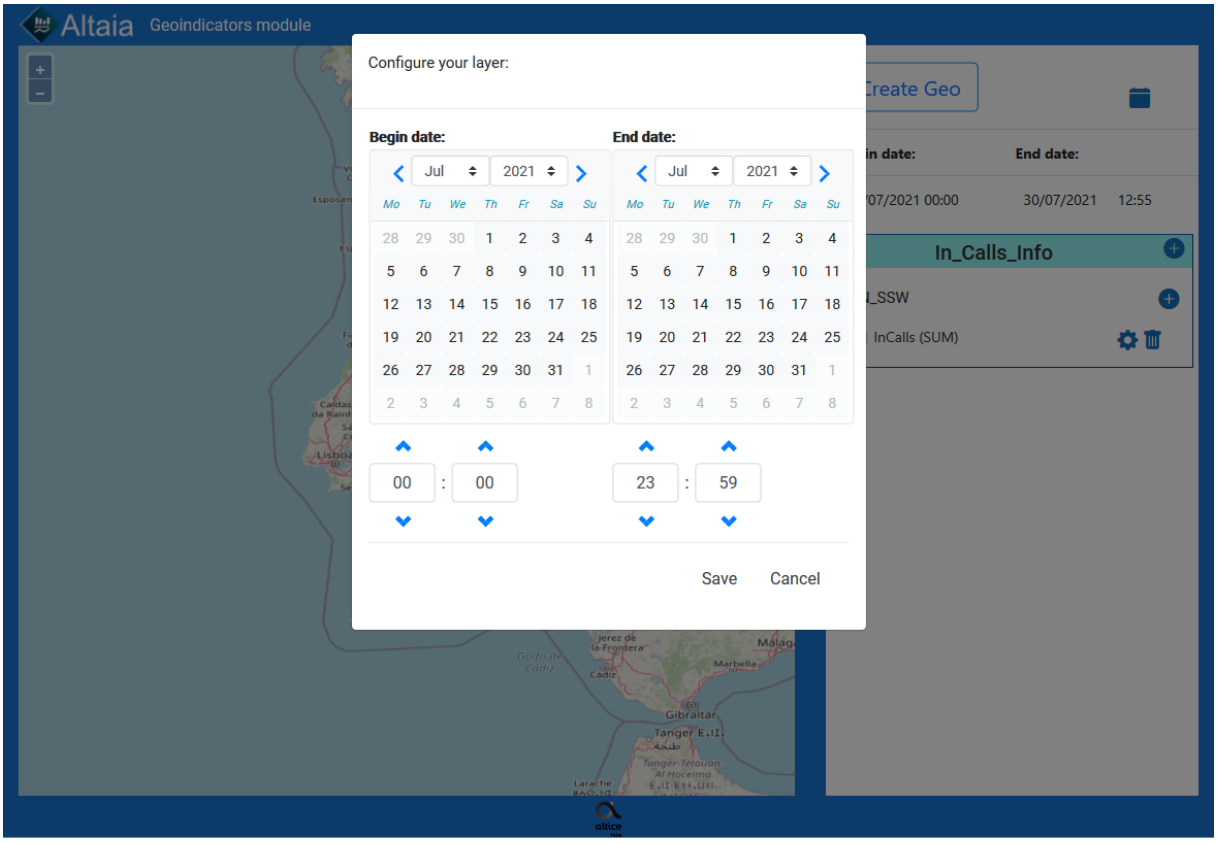


Figura 86 - Teste 4, localizar no tempo a visualização pretendida. Modal para a seleção das datas a considerar.

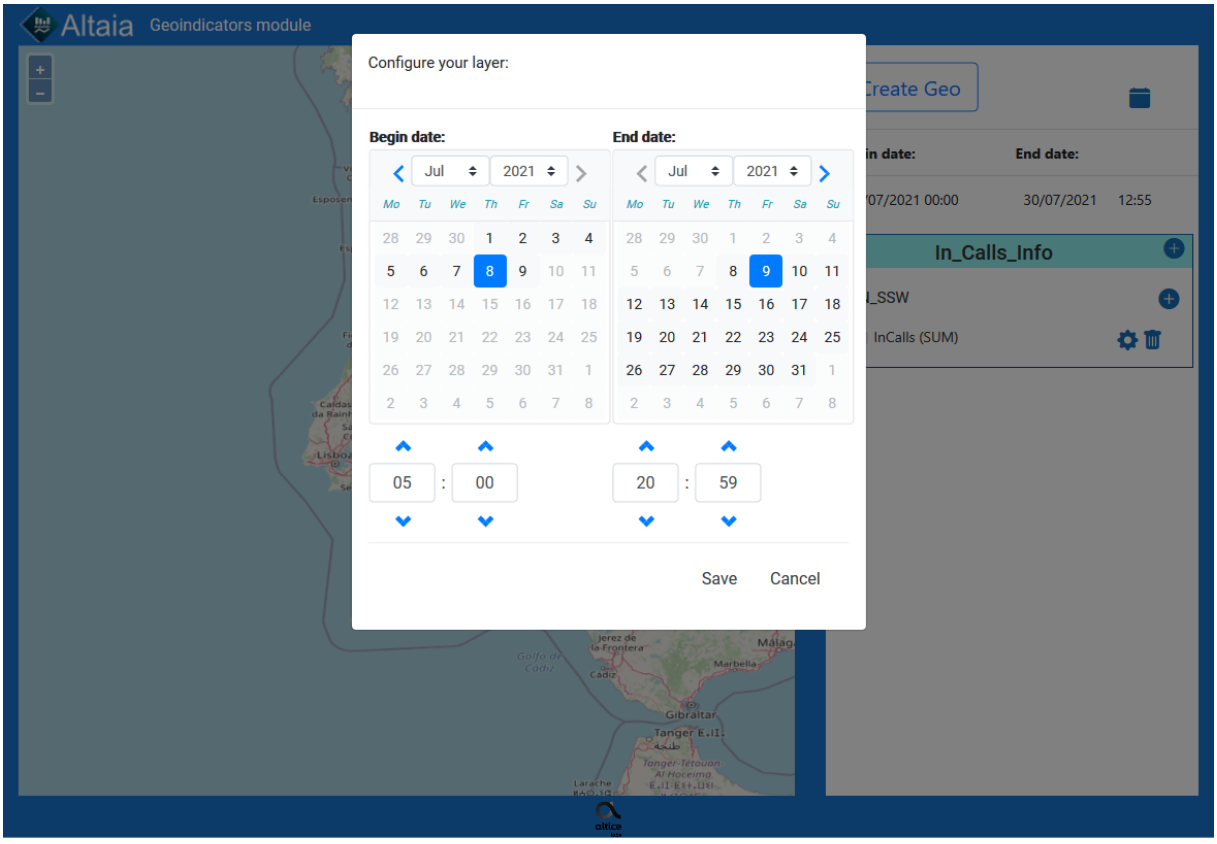


Figura 87 - Teste 4, localizar no tempo a visualização pretendida. Passo 2 de 3, selecionar uma data de início e uma data de fim.

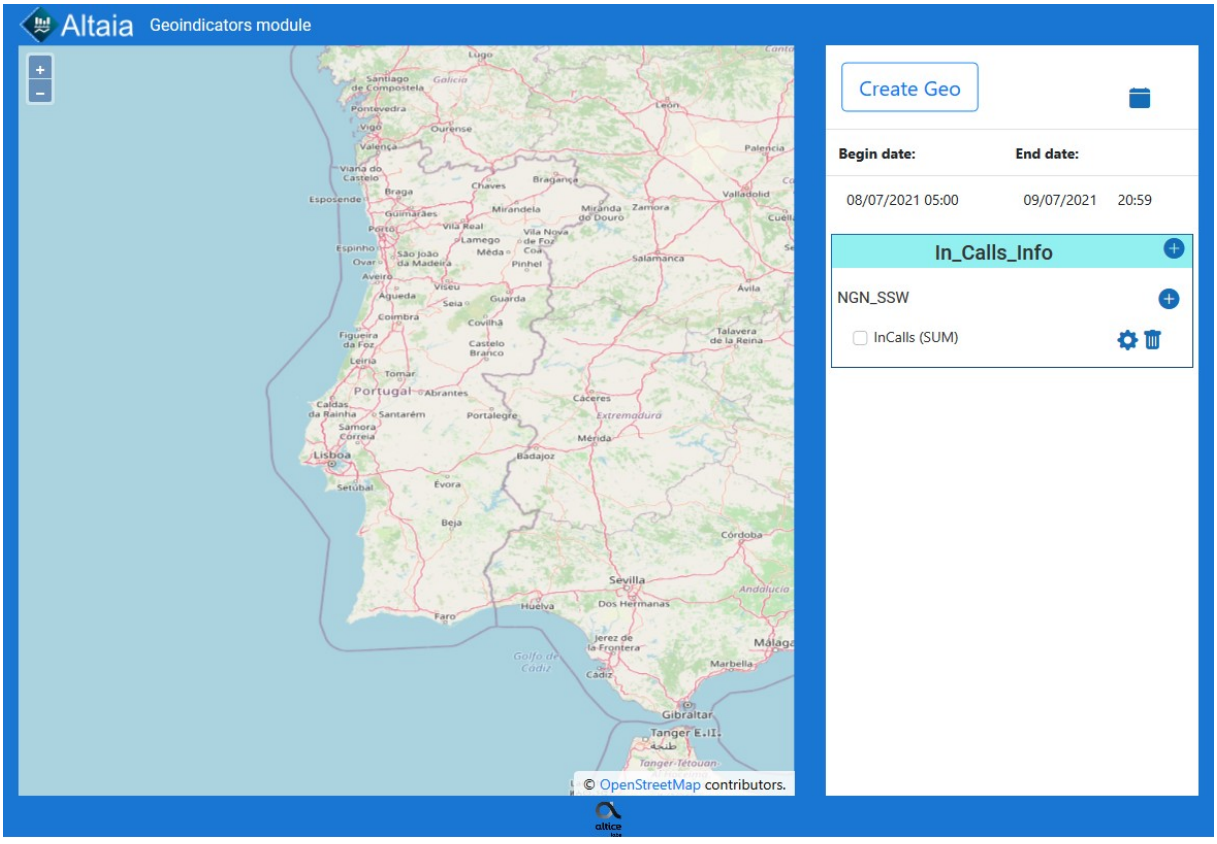


Figura 88 - Teste 4, localizar no tempo a visualização pretendida. Passo 3 de 3, clicar em “Save”.

Apêndice I

No Apêndice I são apresentados os passos para execução do teste 5, visualizar a representação de uma métrica e alterar o estilo. Incluí também a opção de esconder a visualização.

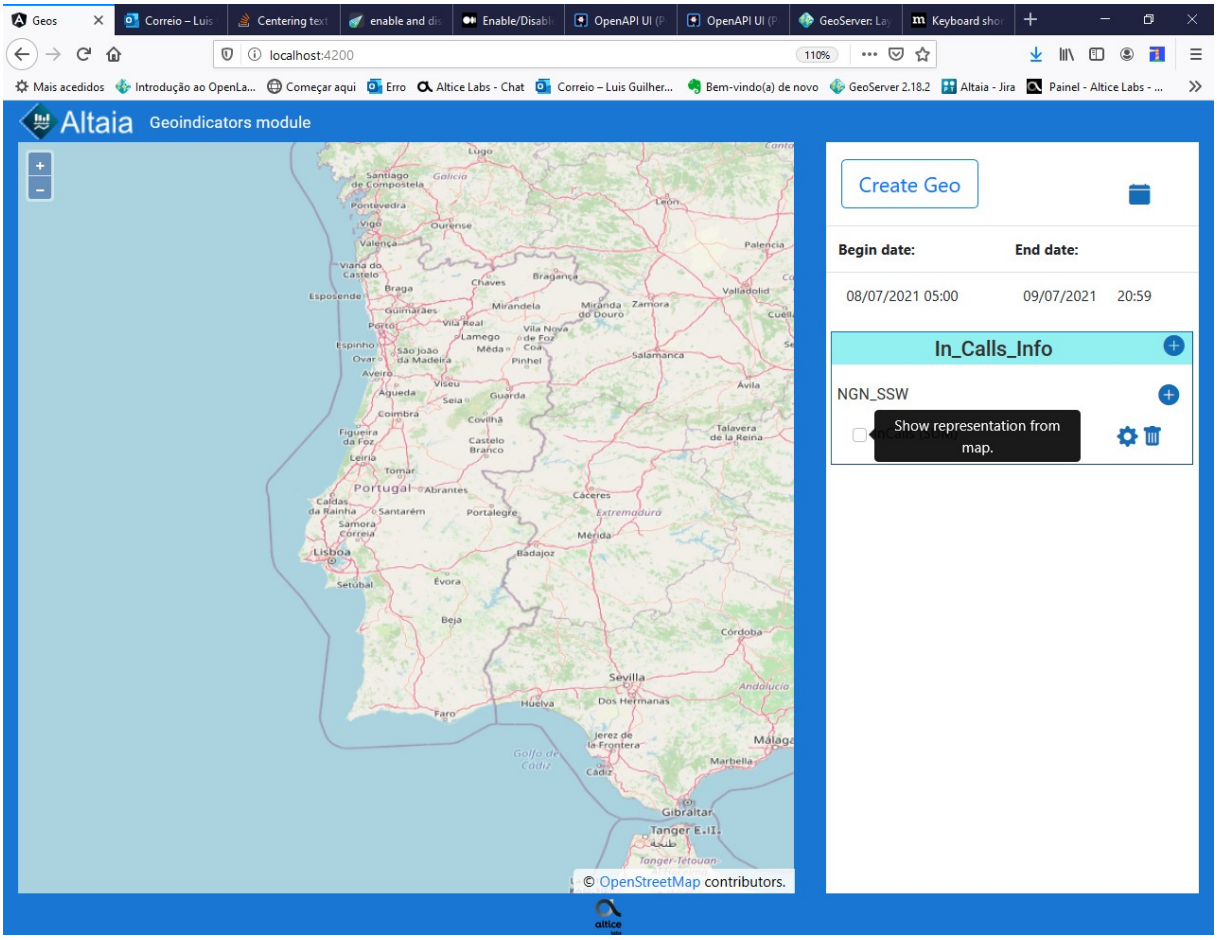


Figura 89 - Teste 5, visualizar a representação de uma métrica e alterar o estilo. Passo 1 de 5 clicar na "*checkbox*" antes do nome da métrica.

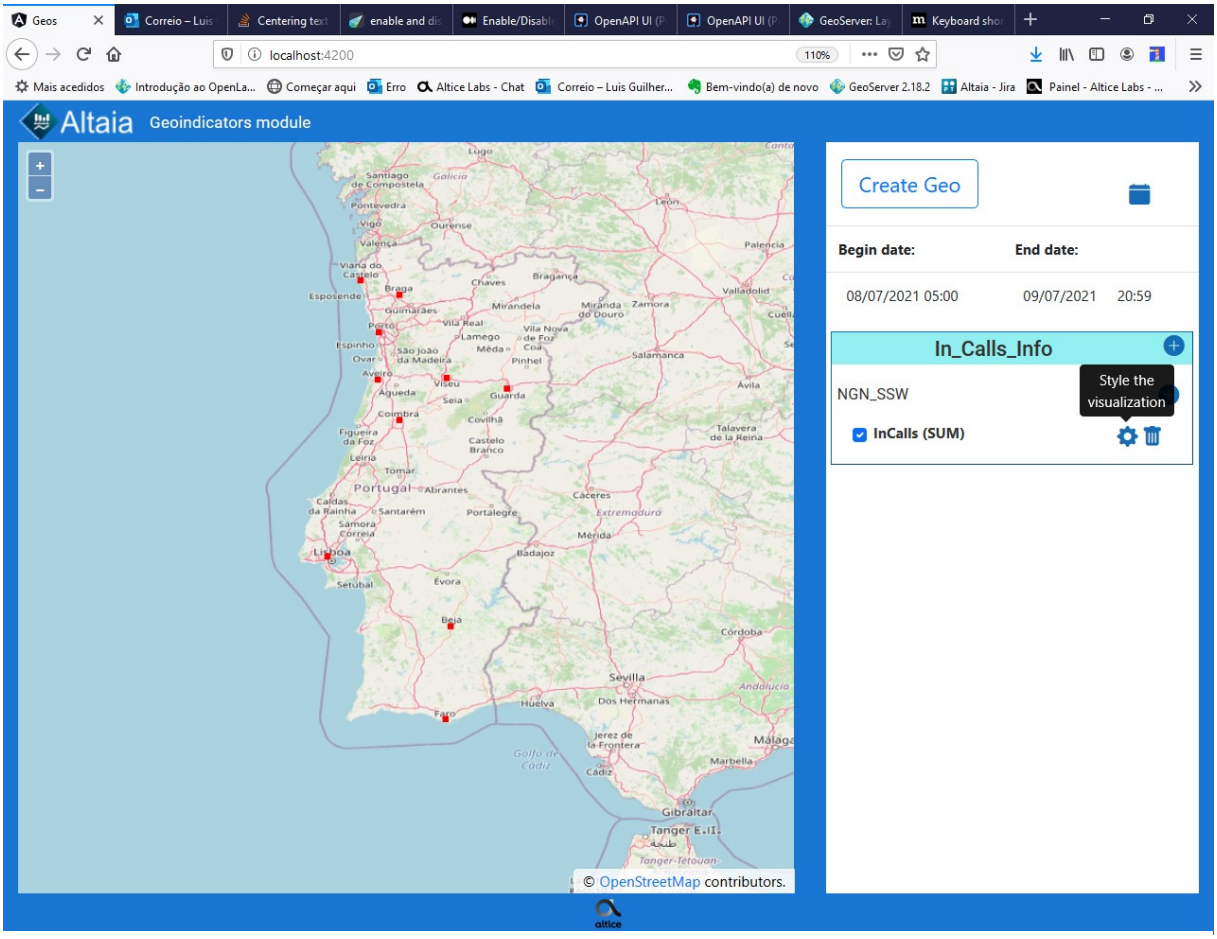


Figura 90 - Teste 5, visualizar a representação de uma métrica e alterar o estilo. Passo 2 de 5, clicar no botão de configuração (roldana).

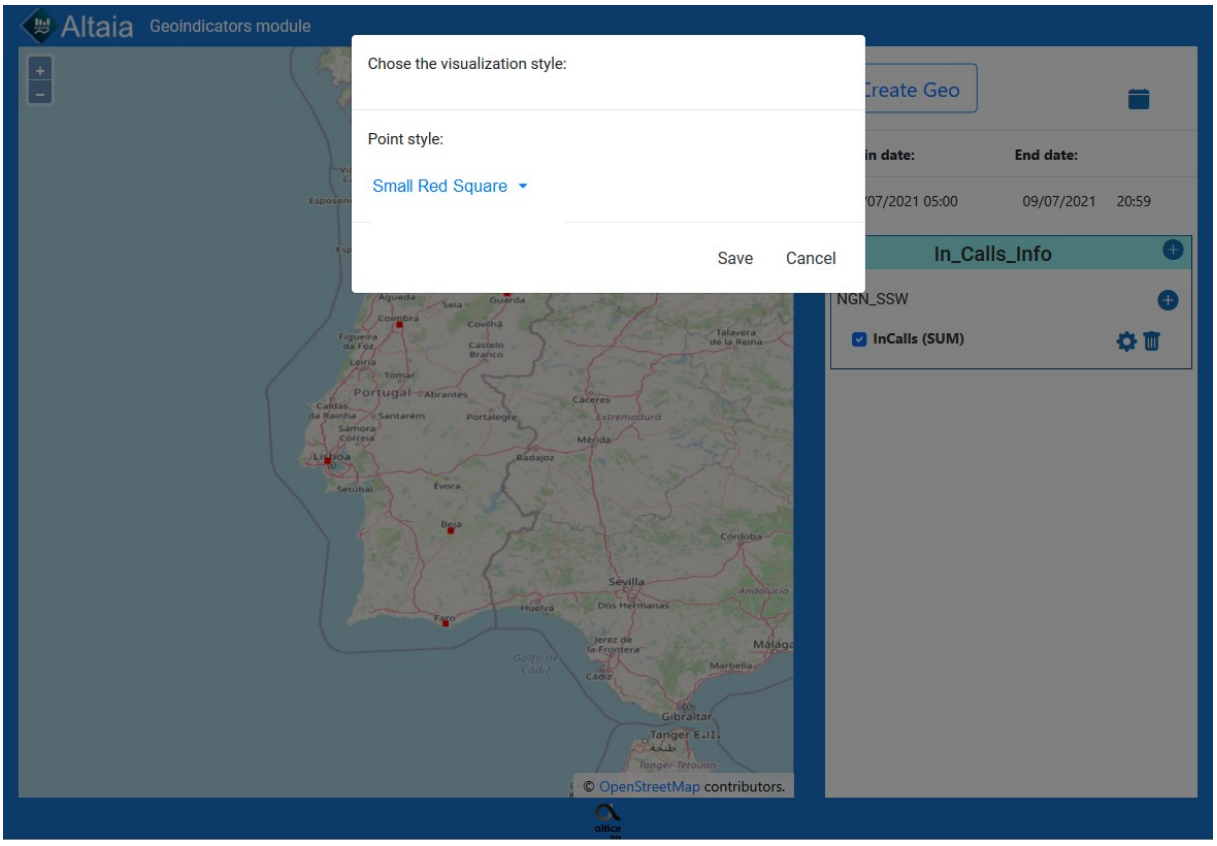


Figura 91 - Teste 5, visualizar a representação de uma métrica e alterar o estilo. Modal para edição do estilo da representação.

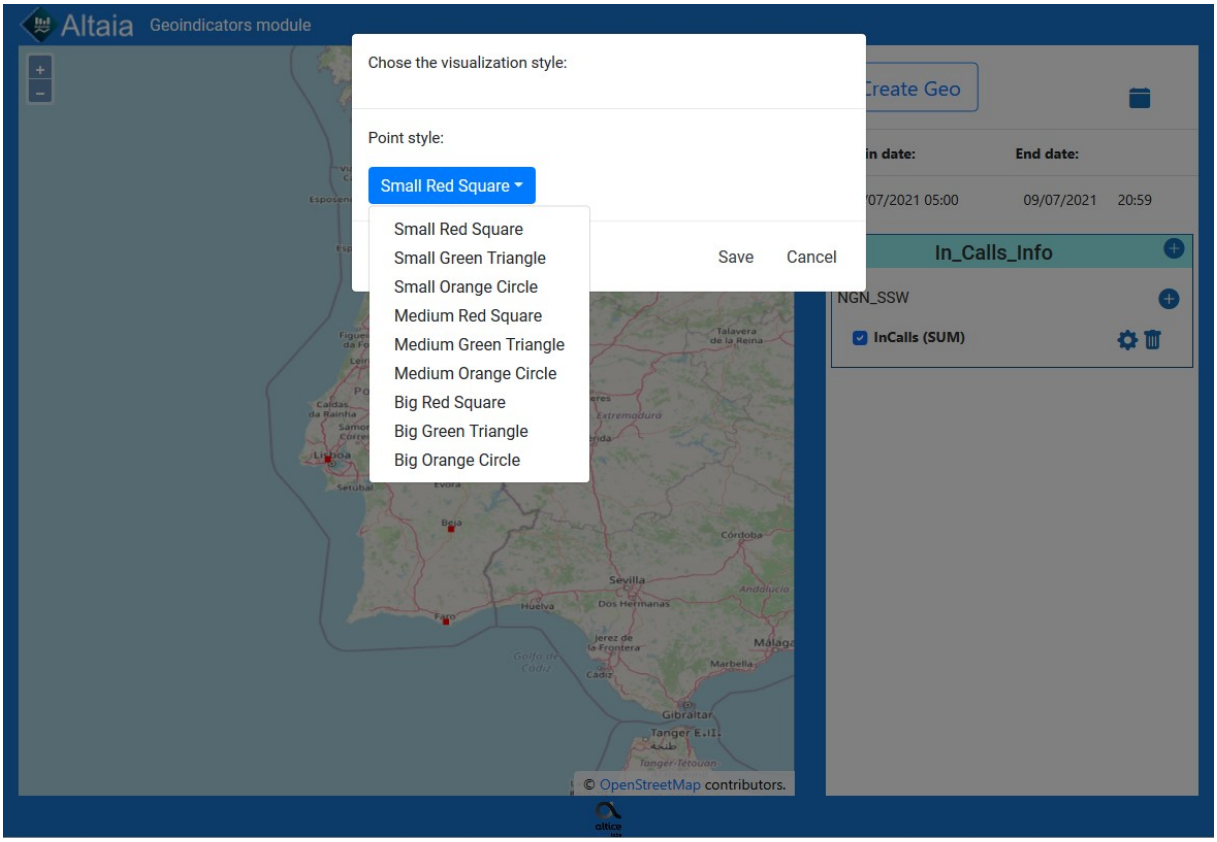


Figura 92 – Teste 5, visualizar a representação de uma métrica e alterar o estilo. Passo 3 de 5, clicar na *droplist*.

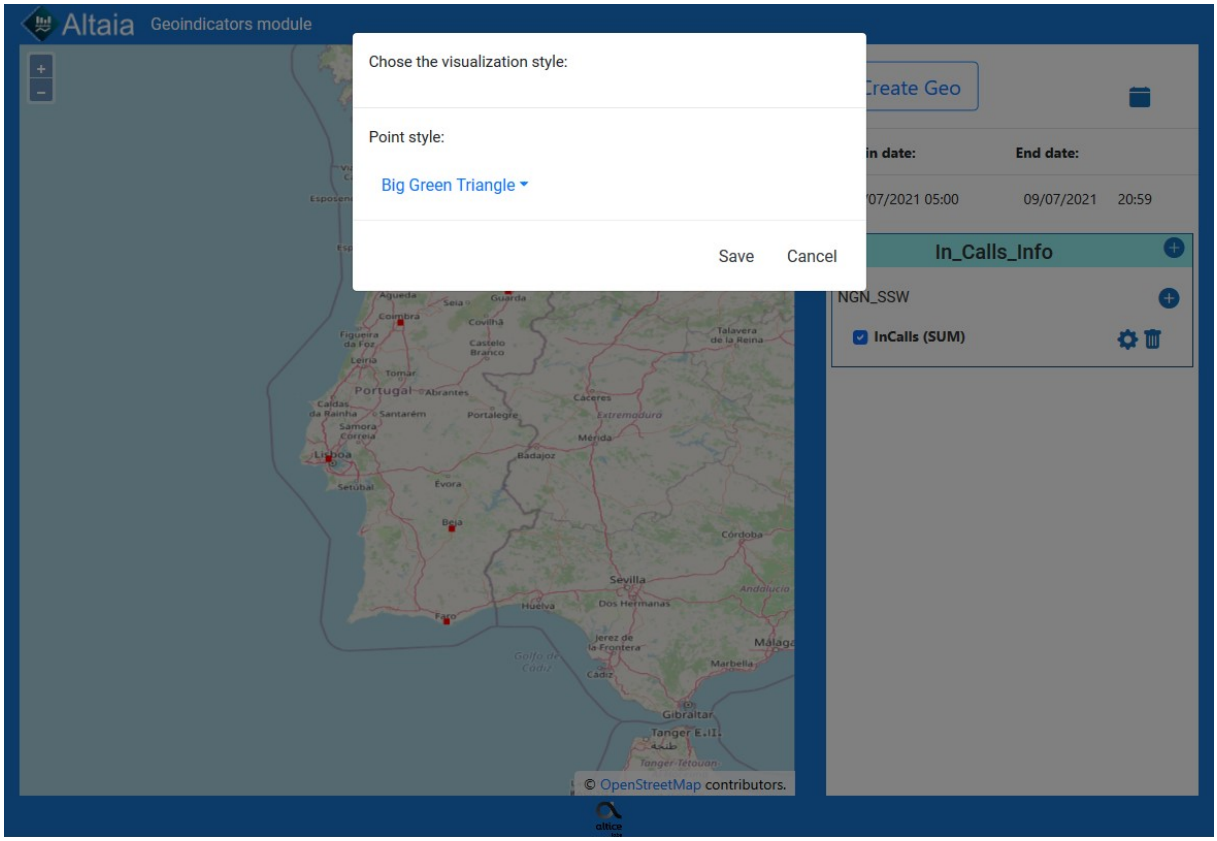


Figura 93 - Teste 5, visualizar a representação de uma métrica e alterar o estilo. Passo 4 de 5, selecionar estilo.

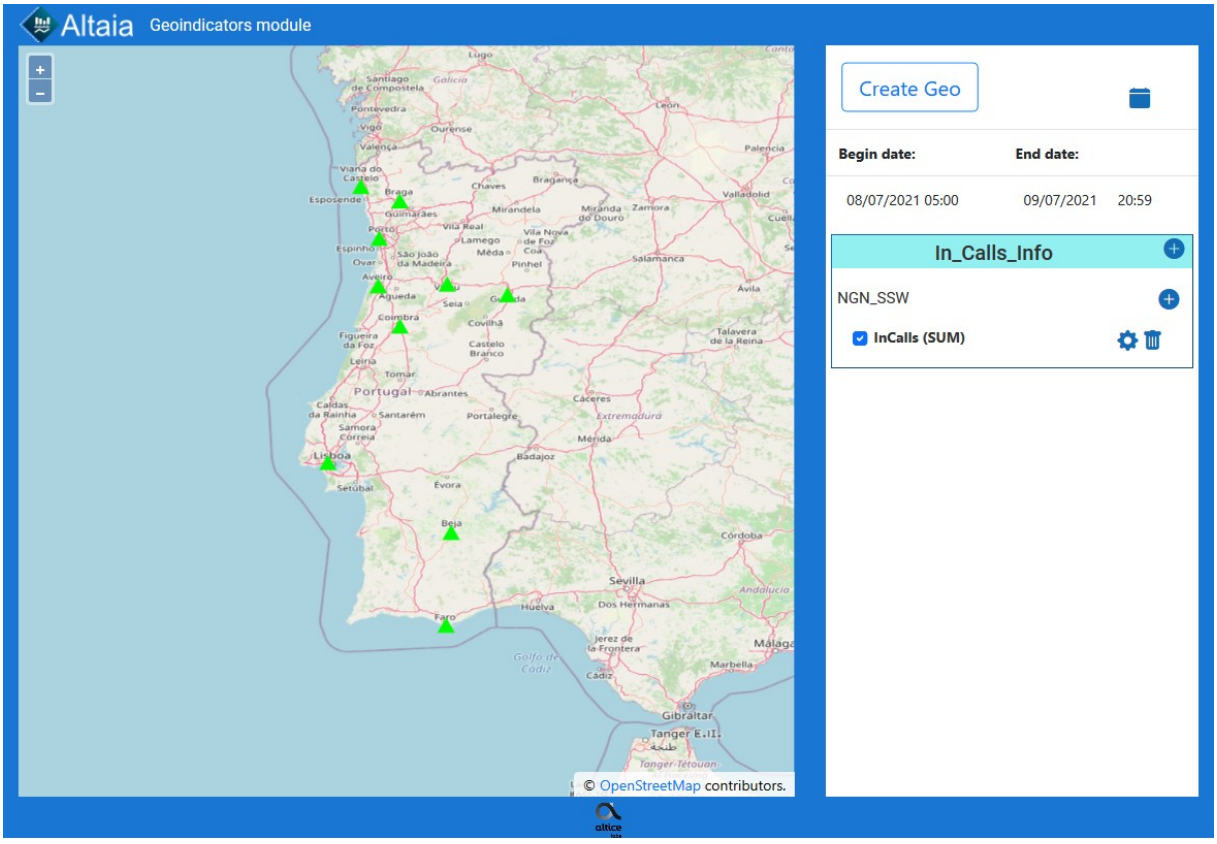


Figura 94 - Teste 5, visualizar a representação de uma métrica e alterar o estilo. Passo 5 de 5, clicar em “Save”.

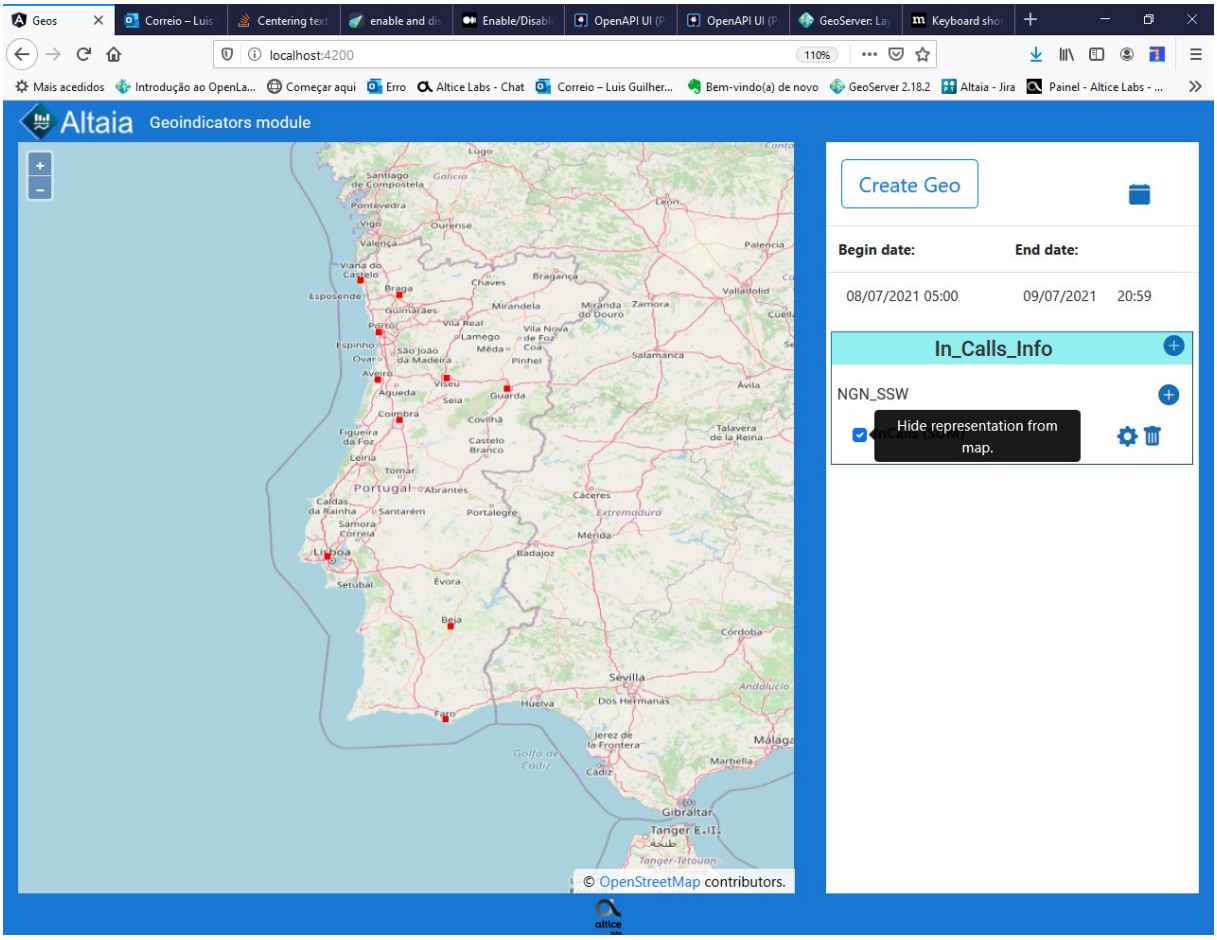


Figura 95 - Visualizar a representação de uma métrica. Operação de esconder a representação da métrica.

Apêndice J

No Apêndice J são apresentados os passos para a execução do teste 6, visualizar a informação dos pontos representados.

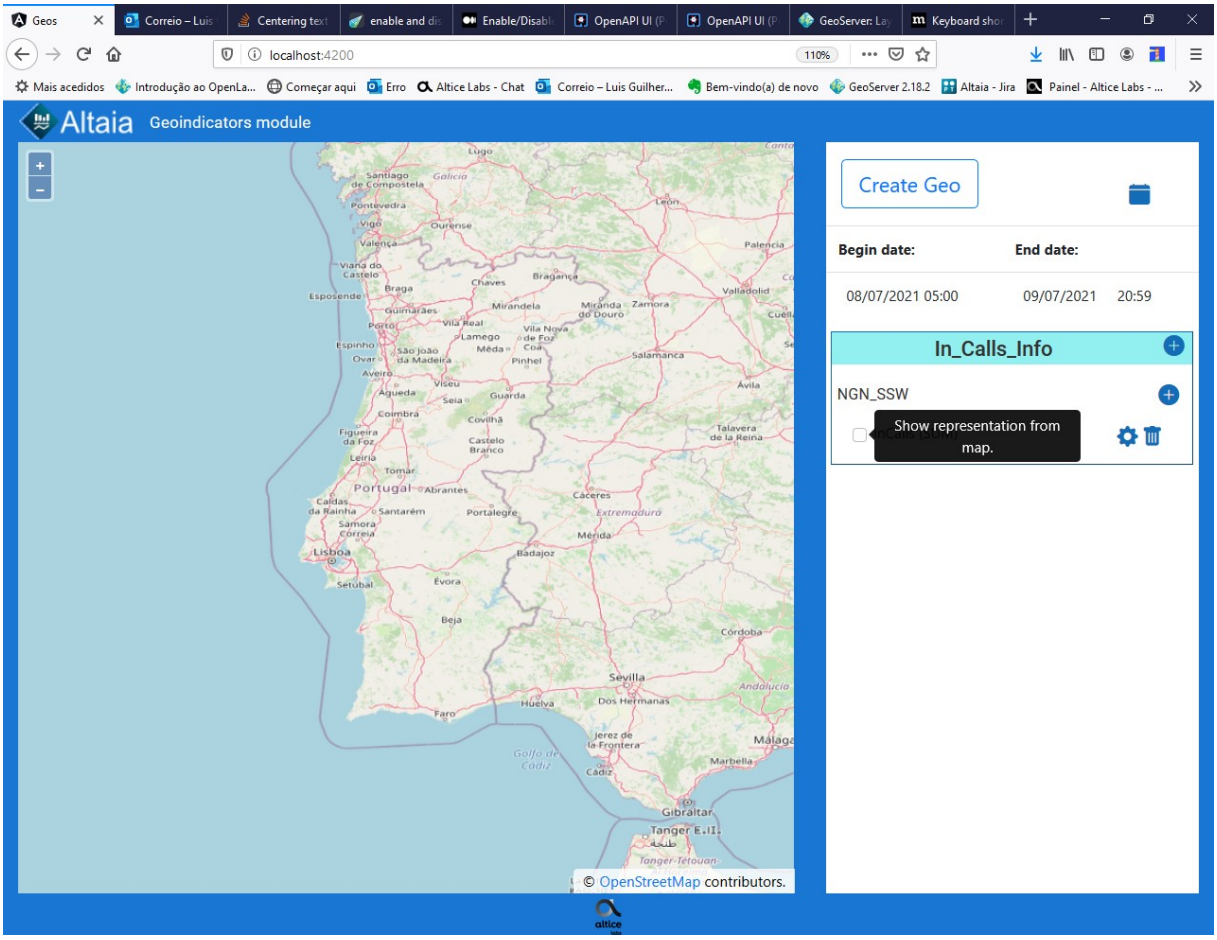


Figura 96 - Teste 6, visualizar a informação dos pontos representados. Passo 1 de 3, tornar a métrica a visualizar no mapa.

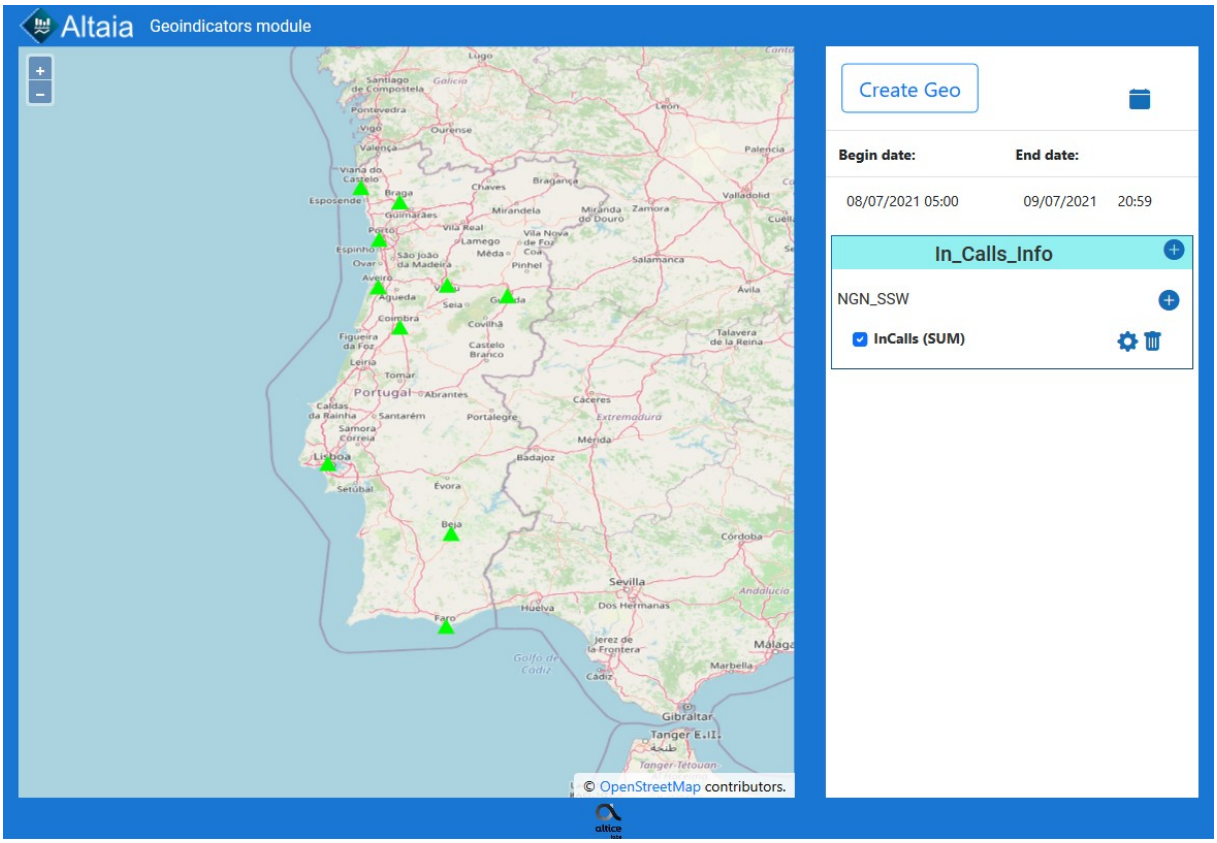


Figura 97 - Teste 6, visualizar a informação dos pontos representados. Passo 2 de 3, clicar no ponto no mapa.

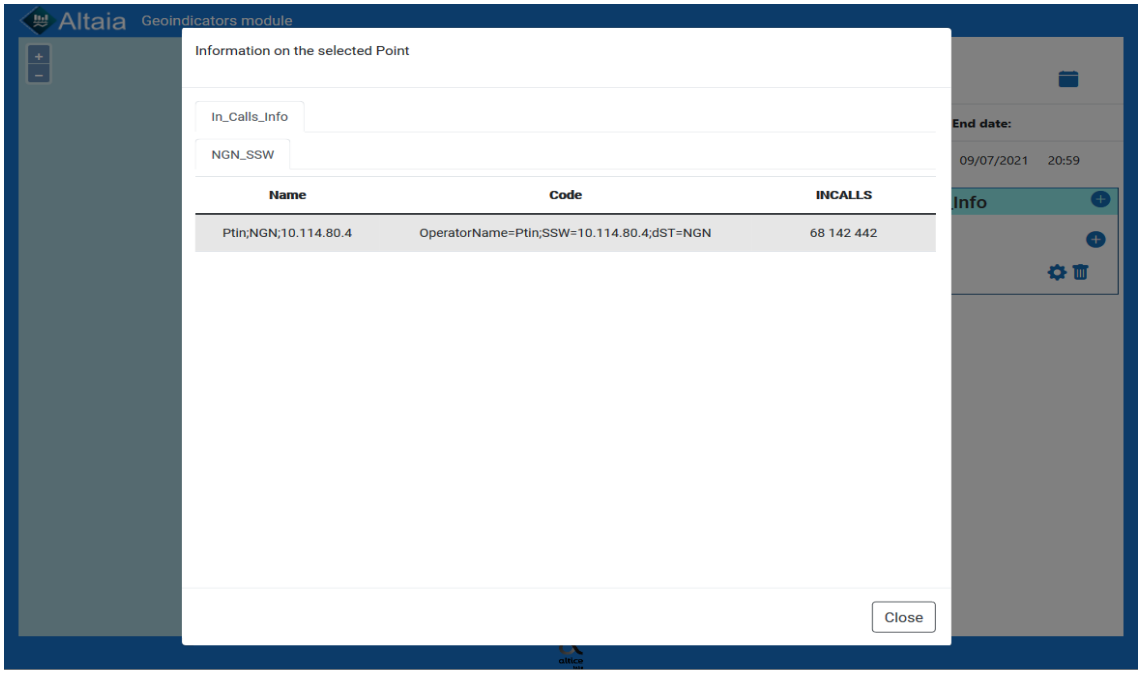


Figura 98 - Teste 6, visualizar a informação dos pontos representados. Passo 3 de 3, visualizar a informação do ponto.

Apêndice K

Passos para execução do teste 7, visualizar a informação dos pontos representados de múltiplas métricas.
Incluí a forma de representação na situação:

- Representação de um geo com 2 métricas pertencentes a diferentes tipos de entidade e ainda um geo com a mesma métrica, mas com uma agregação diferente.

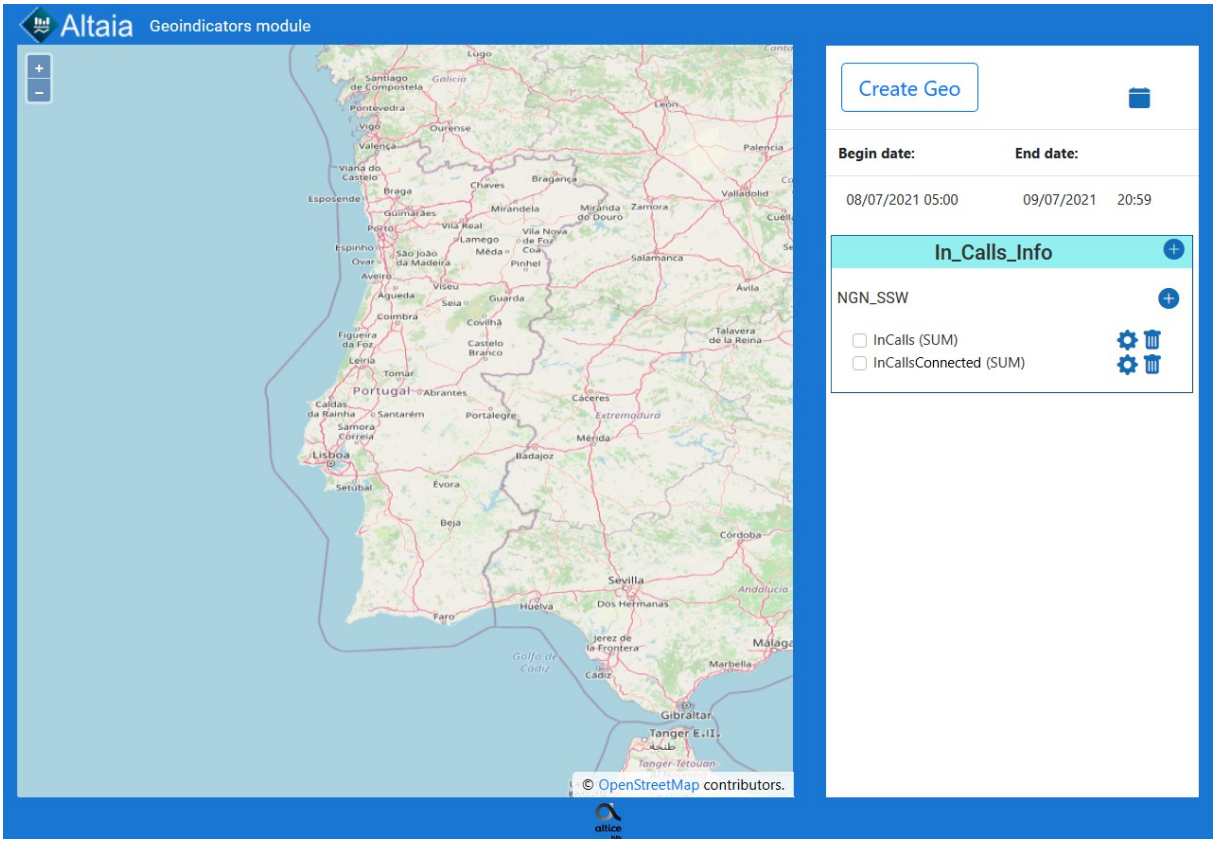


Figura 99 - Teste 7, visualizar a informação dos pontos representados de múltiplas métricas. Passo 1 de 3, tornar as métricas a visualizar no mapa visíveis (check).

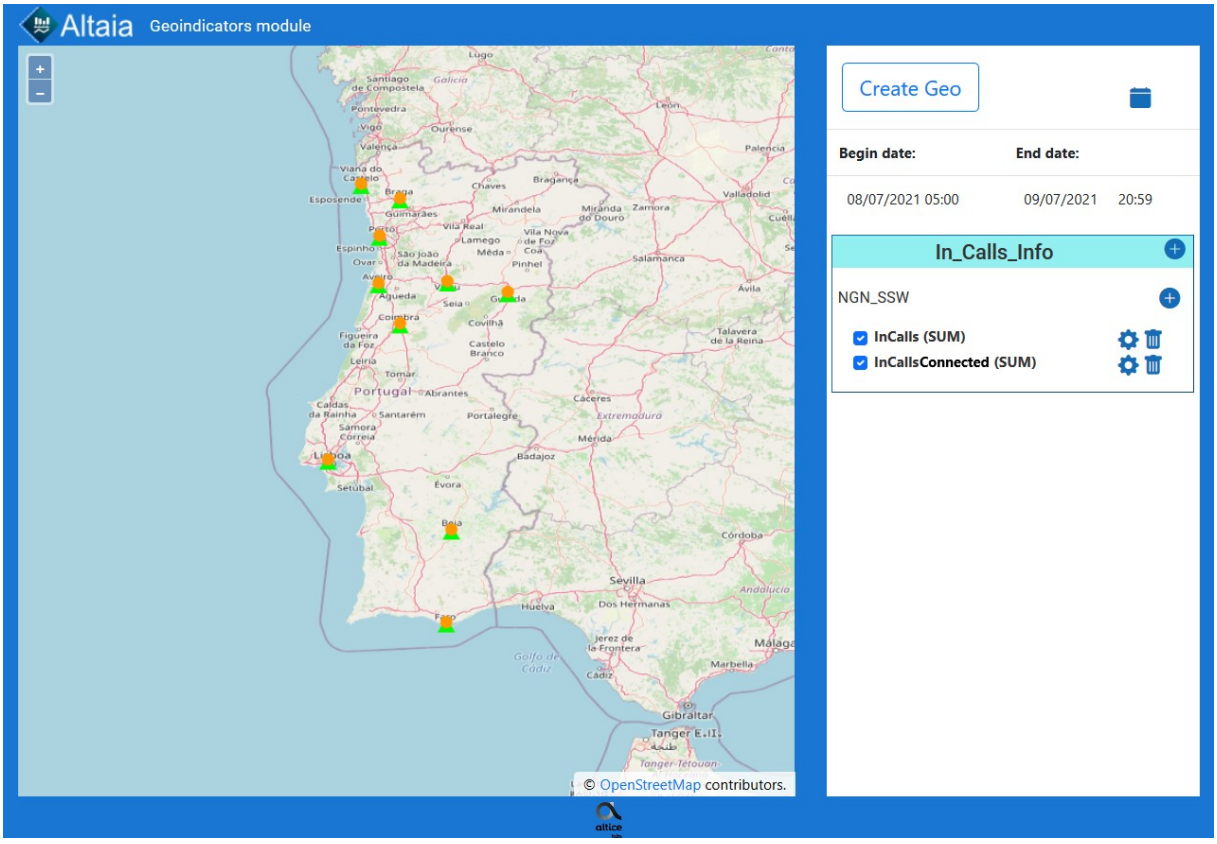


Figura 100 - Teste 7, visualizar a informação dos pontos representados de múltiplas métricas. Passo 2 de 3, clicar no ponto no mapa.

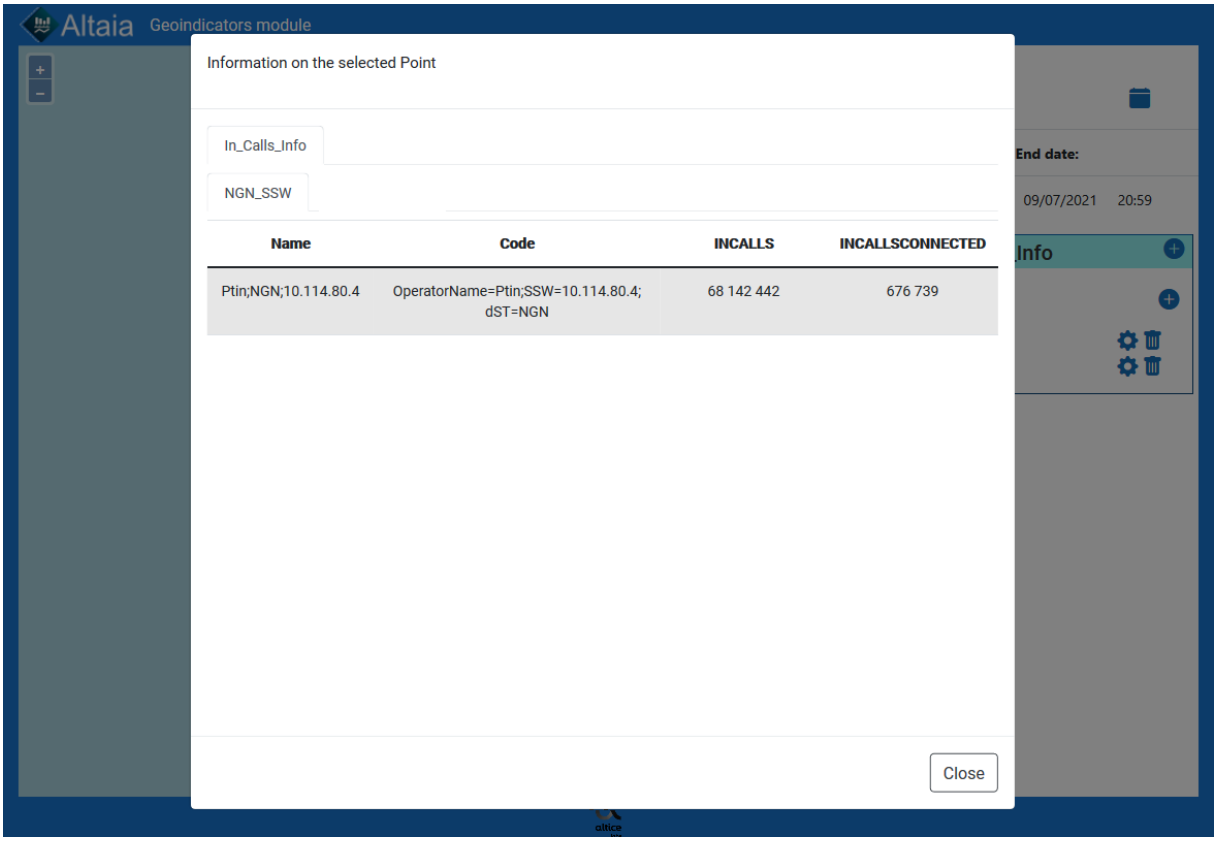


Figura 101 - Teste 7, visualizar a informação dos pontos representados de múltiplas métricas. Passo 3 de 3, visualizar a informação.

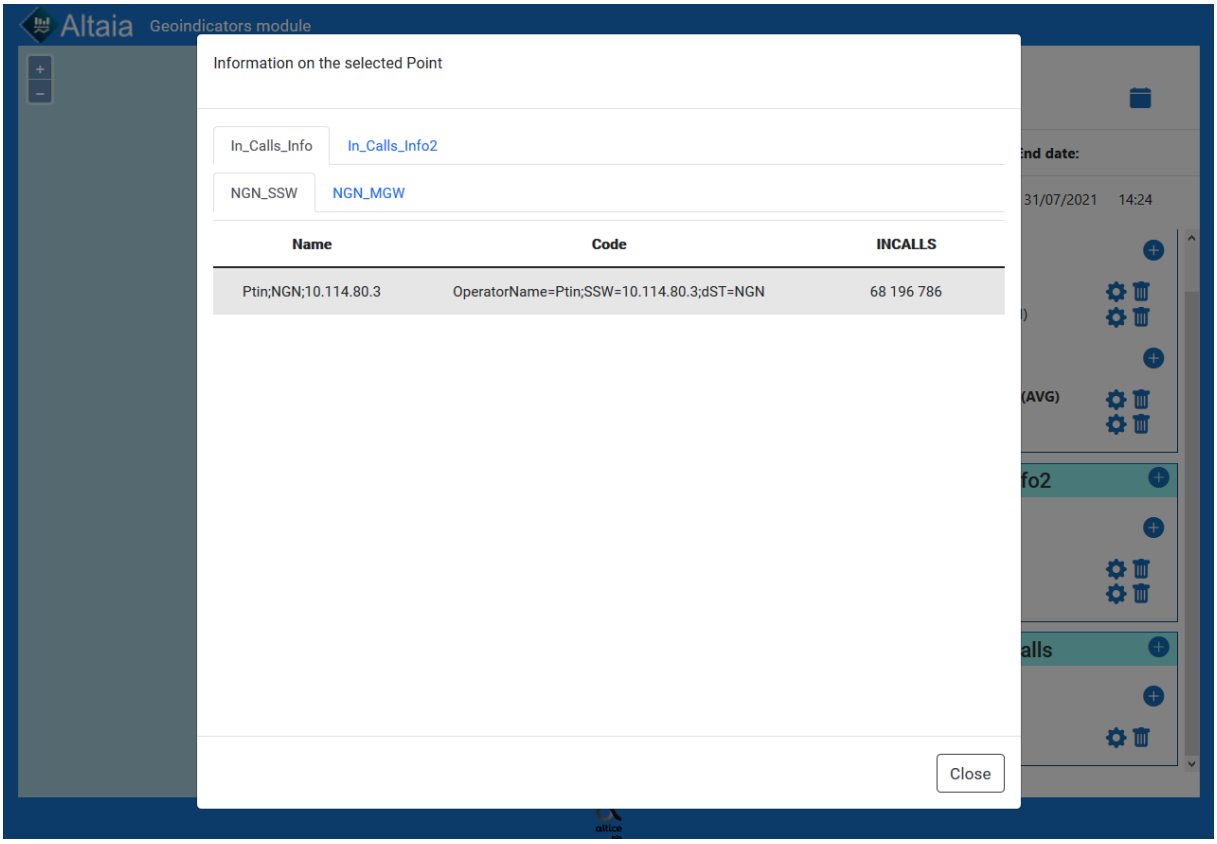


Figura 102 - Visualizar a informação dos pontos representados de múltiplas métricas. Combinação de múltiplos geos, múltiplos tipos de entidade e a mesma métrica (em geos diferentes) com diferentes agregações (Parte 1).

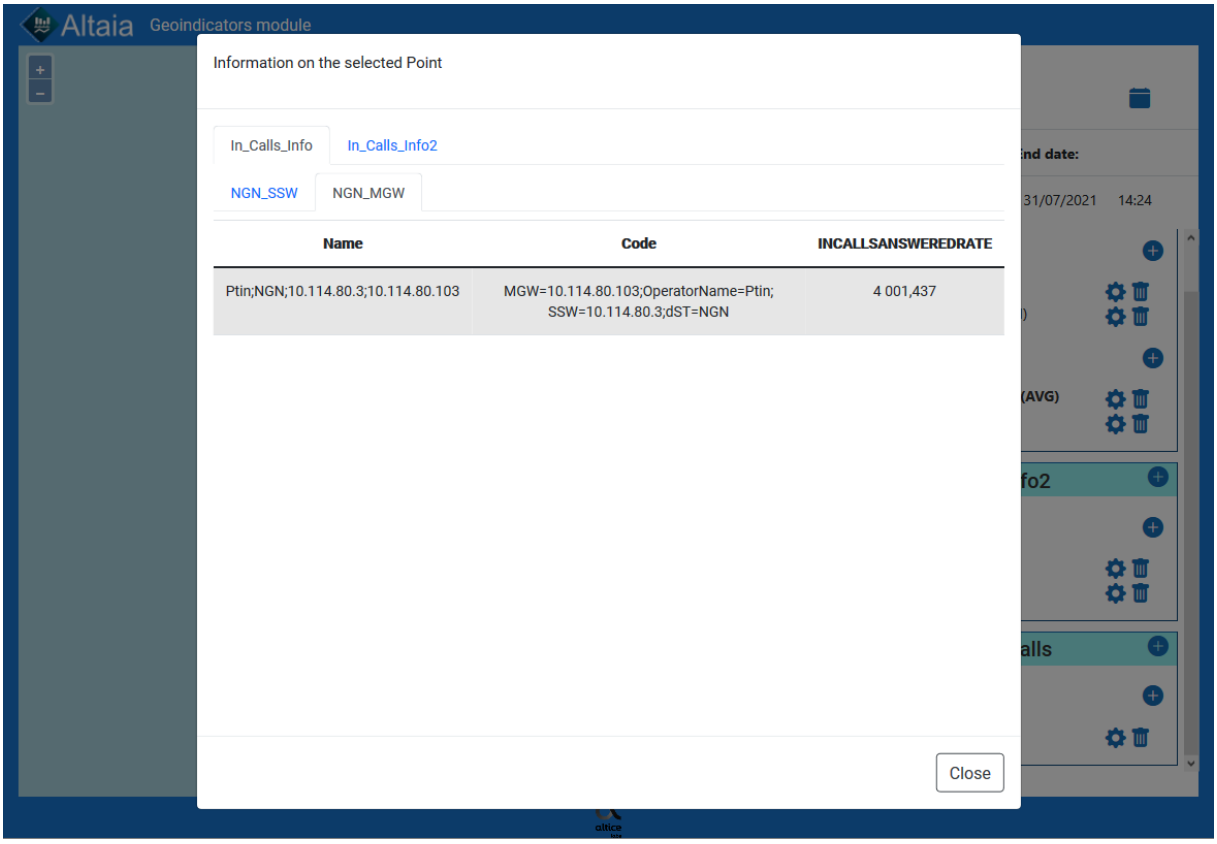


Figura 103 - Visualizar a informação dos pontos representados de múltiplas métricas. Combinação de múltiplos geos, múltiplos tipos de entidade e a mesma métrica (em geos diferentes) com diferentes agregações (Parte 2).

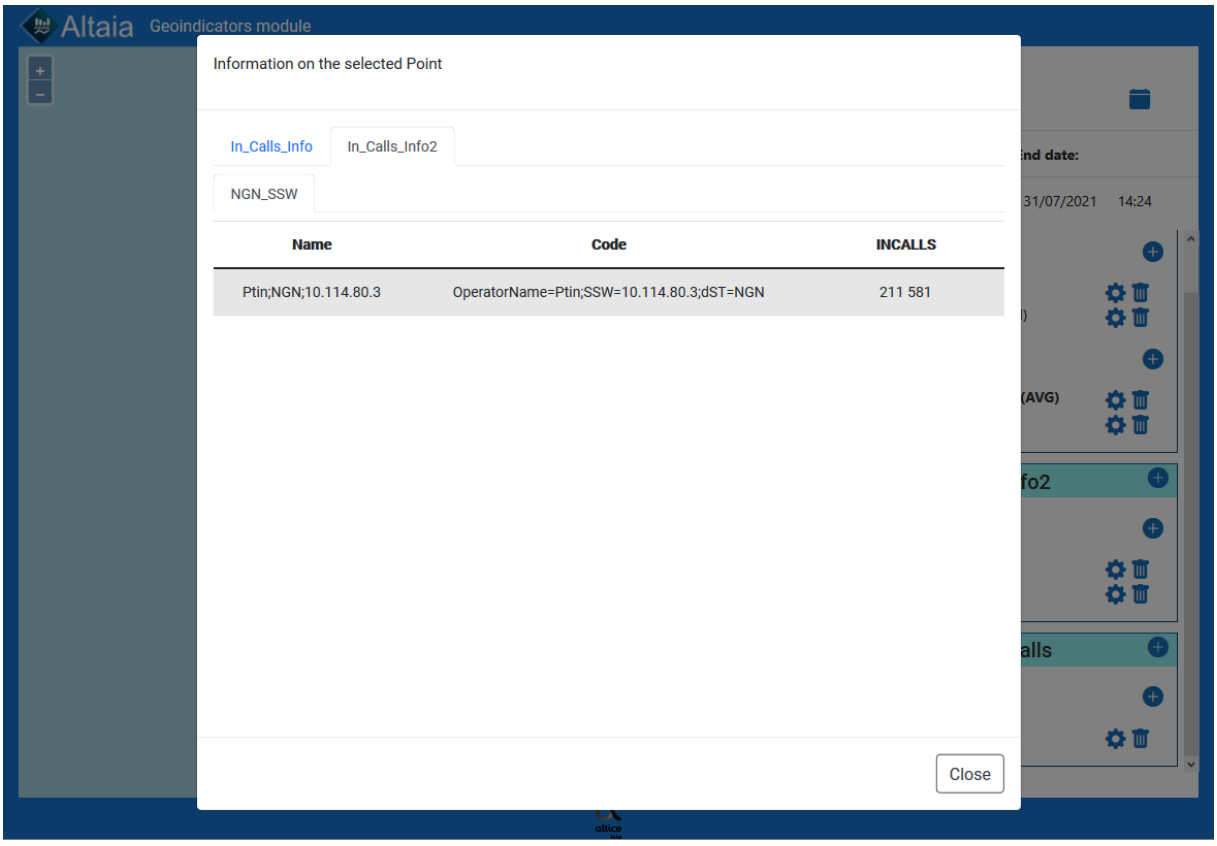


Figura 104 - Visualizar a informação dos pontos representados de múltiplas métricas. Combinação de múltiplos geos, múltiplos tipos de entidade e a mesma métrica (em geos diferentes) com diferentes agregações (Parte 3).