UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

WESLEY CLAUDIO DIAS
Orientador: Tiago Garcia de Senna Carneiro

ARQUITETURA DE MICROSSERVIÇO PARA APLICAÇÕES DE INTELIGÊNCIA ARTIFICIAL GEOESPACIAL

Ouro Preto, MG
2022

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

WESLEY CLAUDIO DIAS

ARQUITETURA DE MICROSSERVIÇO PARA APLICAÇÕES DE INTELIGÊNCIA ARTIFICIAL GEOESPACIAL

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Tiago Garcia de Senna Carneiro

Ouro Preto, MG
2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

D541a Dias, Wesley Claudio.
Arquitetura de Microsserviço para aplicações de Inteligência Artificial Geoespacial. [manuscrito] / Wesley Claudio Dias. - 2022.
31 f.: il.: color..

Orientador: Prof. Dr. Tiago Carneiro.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. GeoAI. 2. Inteligência Artificial Geoespacial. 3. DSS. 4. Sistema de Apoio à Decisão. 5. GIS. 6. Sistema de Informação Geográfica. 7. IA. 8. Inteligência Artificial. I. Carneiro, Tiago. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004

Bibliotecário(a) Responsável: Luciana De Oliveira - SIAPE: 1.937.800

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Wesley Claudio Dias

Arquitetura de Microsserviço para Aplicações de Inteligência Artificial Geoespacial

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 01 de Novembro de 2022.

Membros da banca

Tiago Garcia de Senna Carneiro (Orientador) - Doutor - Universidade Federal de Ouro Preto
Igor Muzetti Pereira (Examinador) - Doutor - Universidade Federal de Ouro Preto
Maycon José Jorge Amaro (Examinador) - Bacharel - Universidade Federal de Ouro Preto

Tiago Garcia de Senna Carneiro, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 01/11/2022.

Documento assinado eletronicamente por **Tiago Garcia de Senna Carneiro**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 03/11/2022, às 20:37, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0416029** e o código CRC **B73DB4D1**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Resumo

A Inteligência Artificial Geoespacial é um campo interdisciplinar que tem conquistado enorme atenção da academia e da indústria nos últimos anos. O avanço tecnológico, principalmente da Inteligência Artificial, observado nos últimos anos, possibilitou o surgimento deste novo campo de estudo. Seu objetivo principal é desenvolver programas computacionais para fornecer uma solução promissora para problemas geoespaciais intensivos em dados ou em computação. Quando estamos desenvolvendo sistemas novos, pensar em sua arquitetura é essencial para o futuro do projeto. Um estilo de arquitetura que tem ganhado cada vez mais espaço é a Arquitetura de Microsserviço. Neste trabalho, foram analisadas as diferentes abordagens para o desenvolvimento de uma arquitetura de microsserviços para a construção de aplicações em Inteligência Artificial Geoespacial. Após a análise e escolha da abordagem mais adequada, foi proposta uma arquitetura de microsserviços para o desenvolvimento de sistemas de apoio à tomada de decisão para as áreas de geomarketing ou de gestão ambiental.

Palavras-chave: GeoAI; Inteligência Artificial Geoespacial; DSS; Sistema de Apoio à Decisão; GIS; Sistema de Informação Geográfica; IA; Inteligência Artificial.

Abstract

Geospatial Artificial Intelligence is an interdisciplinary field that has gained enormous attention from academia and industry in recent years. Technological advances, especially Artificial Intelligence, observed in recent years, enabled the emergence of this new field of study. Its main objective is to develop computer programs to provide a promising solution to data- or computation-intensive geospatial problems. When we are developing new systems, thinking about their architecture is essential for the future of the project. An architecture style that has gained more and more space is the Microservice Architecture. In this work, the different approaches for the development of a microservices architecture for the construction of applications in Geospatial Artificial Intelligence were analyzed. After analyzing and choosing the most appropriate approach, a microservices architecture was proposed for the development of decision-making support systems for the areas of geomarketing or environmental management.

Keywords: GeoAI, DSS, Geographical Artifical Inteligence. Decision Support System. GIS. Geographic Information System.

Lista de Ilustrações

Figura 3.1 – Arquitetura de GIS 15
Figura 3.2 – Arquitetura de um DSS híbrido 15
Figura 4.1 – Hierarquia de Atores 19
Figura 4.2 – Microsserviços para um sistema GeoAI 20
Figura 4.3 – Interface Data Mapper 21
Figura 4.4 – Planning 21
Figura 4.5 – AbstractParam 21
Figura 4.6 – AbstractDataRange 22
Figura 4.7 – Remote Sensor Data Processing 23
Figura 4.8 – Network 23
Figura 4.9 – Geospatial Analysis and Geostatistics 24
Figura 4.10–Influence Area 24
Figura 4.11–Terrain Modeling 25
Figura 4.12–Geocoding 25
Figura 4.13–Model Management 26
Figura 4.14–User Management 26
Figura 4.15–Knowledge Base 27
Figura 4.16–SpatioTemporal Data Management 27
Figura 4.17–Object Management 28
Figura 4.18–Access Control 28
Figura 4.19–Authentication 28

Lista de Abreviaturas e Siglas

AI	Artificial Intelligence
API	Interface de Programação de Aplicação
BI	Business Intelligence
DDD	Domain-Driven Design
DECOM	Departamento de Computação
DSS	Sistema de Apoio à Decisão
GeoAI	Inteligência Artificial Geoespacial
GIS	Sistema de Informação Geográfica
MDD	Model-Driven Development
MNT	Modelo Numérico de Terreno
MVP	Produto Mínimo Viável
UFOP	Universidade Federal de Ouro Preto
UML	Unified Modeling Language

Sumário

1 Introdução 1
1.1 Justificativa 1
1.2 Objetivos 2
1.2.1 Objetivos Gerais 2
1.2.2 Objetivos Específicos 2
1.3 Organização do Trabalho 3
2 Revisão Bibliográfica 4
2.1 Fundamentação Teórica 4
2.1.1 Inteligência Artificial Geoespacial 4
2.1.2 Sistema de Informação Geográfica 5
2.1.3 Sistema de Apoio à Decisão 6
2.1.4 Arquitetura de Microsserviços 7
2.2 Trabalhos Relacionados 8
2.2.1 Extração de microsserviços a partir de arquiteturas monolíticas 8
2.2.2 Desafios do projeto dirigido por domínio de microsserviços: Uma perspectiva dirigida por modelo 10
3 Desenvolvimento 13
3.1 Desenvolvendo projetos em Microsserviços 13
3.2 GeoAI: compreendendo seu desenvolvimento 14
3.3 Arquitetura de Microsserviços e GeoAI: análise do seus requisitos 16
3.4 Projetando uma Arquitetura de Microsserviços para Sistema de Inteligência Artificial Geoespacial 16
4 Resultados 18
4.1 Atores 18
4.2 Serviços 18
4.2.1 Primeira Camada 19
4.2.2 Segunda Camada 22
4.2.3 Terceira Camada 25
4.2.4 Quarta Camada 27
5 Conclusão 29
5.1 Trabalhos Futuros 29

Referências 30

1 Introdução

Ao longo da trajetória histórica da humanidade, os indivíduos vêm utilizando de diversas técnicas e tecnologias para analisar, entender e intervir no meio natural. Tais objetos e técnicas, como arado, binóculo, estradas, foice e bússola, tinham como objetivo auxiliar o ser humano na sua interação com o mundo natural e gerar meios para que o indivíduo o alterasse, conforme as necessidades de cada contexto. A busca por um entendimento acerca de como ocorrem essas interações e o domínio da natureza surgem como formas de atender as necessidades do homem. Essas necessidades, que podem ser exemplificadas com o desenvolvimento de formas de alimentação, abrigo e proteção, são regidas por uma série de técnicas, entendidas como a principal forma de relação do homem com o meio. Tais técnicas são formas instrumentais e sociais onde os homens organizam suas vidas e os espaços nos quais habitam (SANTOS, 2002). Com o intuito de compreender melhor a relação entre homem, a natureza e as técnicas desenvolvidas, se instituiu o estudo da Geografia.

Os avanços tecnológicos, principalmente a partir do surgimento dos primeiros computadores eletrônicos, permitiram ampliar o campo de estudo e aplicação da Geografia, desenvolvendo ferramentas que auxiliam na compreensão dos meios de forma ainda mais detalhada. A partir de tal desenvolvimento, podemos observar uma nova técnica que vem sendo estudada nos últimos anos, sendo essa a junção da Geografia com a Inteligência Artificial (AI — do inglês, Artificial Intelligence). Esse estudo levou o nome de Inteligência Artificial Geoespacial (GeoAI — inglês, Geographical Artificial Intelligence), e ele visa utilizar técnicas da AI que permitem auxiliar a resolução de problemas ambientais, sejam eles sociais, econômicos ou naturais (GAO, 2021).

A criação de novos modelos de AI e outros métodos computacionais desenvolvidos especialmente para problemas geográficos favoreceram o surgimento da GeoAI e suas demais aplicações. Neste trabalho, abordaremos a interface entre AI e Geografia ao estudarmos a integração entre os serviços oferecidos pelos Sistemas de Apoio à Decisão (DSS — do inglês, Decision Support System) (SHIM et al., 2002), que visam auxiliar e melhorar a tomada de decisão de um grupo ou pessoa, e os serviços oferecidos pelos Sistemas de Informação Geográfica (GIS — do inglês, Geographic Information System). Um GIS é um sistema computacional especializado em armazenar, recuperar e analisar dados geoespaciais (CHANG, 2008). Nesta interface, tais dados podem ser utilizados para treinar os modelos de GeoAI capazes de solucionar os problemas ambientais, oriundo das interações entre o homem e a natureza.

1.1 Justificativa

Atualmente, o mundo vivencia grandes problemas de natureza espacial como, por exemplo, degradação da qualidade ambiental, desastres naturais ocorrendo mais frequentemente,

doenças novas e reemergentes, entre outros. Entre os inúmeros fatores, tanto naturais, quanto humanos que podem ser apontados para compreender o contexto de tais ocorrências, podemos entender que a maioria destas questões globais se devem ao rápido crescimento populacional e econômico, ao consumo excessivo de recursos naturais e aos níveis crescentes de desigualdade social (LI, 2020). Os sistemas que lidam com a coleta, armazenamento, análise e visualização de dados geoespaciais, desempenharam um papel essencial para prevenir e responder a problemas atuais e emergentes, já que estes auxiliam na geração e compreensão de dados utilizados pelos grupos e/ou sistemas que atuam diretamente sobre estes problemas. Pensando então o cenário descrito e em sua massiva importância para o quadro de vida dos indivíduos de maneira geral, é essencial pensar em meios para que métodos, técnicas e ferramentas em GeoAI sejam desenvolvidos em maior volume, de forma mais rápida e eficiente. Neste contexto, para a evolução da GeoAI é preciso pensar em um conjunto mínimo, suficiente e necessário de funcionalidades computacionais que poderiam ser reutilizadas na construção de novas aplicações neste tema. Surge então a necessidade de pensarmos como estas funcionalidades poderiam ser agrupadas em serviços e como estes serviços deveriam ser organizados em uma arquitetura de software reutilizável. Uma arquitetura de alta qualidade leva a uma entrega mais rápida de novos recursos, porque *"há menos sujeira para atrapalhar"* (FOWLER, 2019).

1.2 Objetivos

Esta seção de texto apresenta o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivos Gerais

O presente trabalho planeja analisar as diferentes abordagens para o desenvolvimento de Arquiteturas de Microsserviços. Após a análise, o trabalho tem como objetivo geral propor uma Arquitetura de Microsserviços de modo a auxiliar à construção de Sistemas de Inteligência Artificial Geoespacial.

1.2.2 Objetivos Específicos

Para a execução desta análise enunciada como objetivo geral, será necessário que os seguintes objetivos específicos sejam alcançados:

- Realizar uma revisão da literatura em busca de diferentes estratégias para o projeto, a implementação e avaliação de Arquiteturas de Microsserviços. É essencial identificar requisitos mínimos das Arquiteturas de Microsserviço e as vantagens e desvantagens das diferentes estratégias para seu desenvolvimento.
- Realizar uma revisão sobre estudos que descrevem o desenvolvimento de Sistemas de inteligência Artificial Geoespacial, especificamente, aqueles que discutem os requisitos

mínimos e as diferentes arquiteturas de Sistemas Espaciais de Suporte a Tomada e Decisões e de Sistemas de Informação Geográfica.

- Identificar mediante uma análise criteriosa os requisitos de software de uma Arquitetura de Microsserviço destinada ao desenvolvimento de Sistema de Inteligência Artificial Geográfica, justificando os requisitos funcionais e não funcionais desta arquitetura.
- Estabelecer mediante uma análise criteriosa qual a abordagem arquitetônica é mais indicada para o projeto e implementação de uma Arquitetura de Microsserviços destinada à construção de Sistemas de Informação Geográfica, determinando quais funcionalidades devem estar presentes na arquitetura e como estas funcionalidades devem ser agrupadas em microsserviços altamente coesos e fracamente acoplados.
- Propor uma Arquitetura de Microsserviços destinada à construção de Sistema de Inteligência Artificial Geográfica, respeitando os requisitos levantados, justificando cada microsserviço e suas funcionalidades básicas.

1.3 Organização do Trabalho

Os demais capítulos deste texto se organizam da seguinte maneira. O Capítulo 2 apresenta a fundamentação teórica necessária ao entendimento deste trabalho, assim como os trabalhos encontrados na literatura que a ele são correlatos. O Capítulo 3 apresenta os métodos e técnicas utilizados para a execução deste trabalho. O Capítulo 4 apresenta e explica uma proposta de Arquitetura de Microsserviços para Sistema de Inteligência Artificial Geográfica e no Capítulo 5 são apresentadas as considerações finais e as propostas de trabalhos futuros.

2 Revisão Bibliográfica

Este capítulo aborda uma detalhada revisão bibliográfica e encontra-se organizado da seguinte forma: a seção 2.1 apresenta a fundamentação teórica necessária para o entendimento deste trabalho e a seção 2.2 apresenta uma breve descrição de trabalhos relacionados que guiaram o desenvolvimento deste projeto.

2.1 Fundamentação Teórica

Nesta seção, serão discutidos os principais fundamentos para o bom entendimento do trabalho proposto.

2.1.1 Inteligência Artificial Geoespacial

A Inteligência Artificial Geoespacial (GeoAI) é um campo interdisciplinar que tem como objetivo, através da integração dos estudos geoespaciais e da AI, desenvolver programas computacionais para fornecer uma solução promissora para problemas geoespaciais intensivos em dados ou em computação (LI, 2020). O surgimento da GeoAI foi possível graças a três fatores, sendo eles, primeiro a crescente disponibilidade de dados geoespaciais em grande escala. Esses conjuntos de dados grandes e heterogêneos permitem o treinamento de modelos de AI para abordar uma grande variedade de problemas (HU et al., 2019). O segundo fator são os recentes avanços em AI, mais especificamente em aprendizado profundo, permitindo analisar, extrair informações e visualizar dados geoespaciais massivos, o que é difícil por meio dos métodos tradicionais de análise espacial (HU et al., 2019). O terceiro fator é o aumento na disponibilidade dos recursos computacionais de alto desempenho, como Unidades de Processamento Gráfico (GPUs), possibilitando o treinamento eficaz de modelos de aprendizado de máquina com big data (LI, 2020).

A GeoAI possui duas classes de abordagens principais. A primeira delas é a orientada a dados, conhecida como abordagem bottom-up. Esta abordagem utiliza de aprendizado de máquina para aprender a fazer previsões a partir de grandes quantidades de dados sem a necessidade de regras analíticas e explícitas (LI, 2020). A segunda é a orientada a conhecimento, nomeada como abordagem top-down. Diferente da outra abordagem, a top-down depende de uma base de conhecimento para fornecer definições semânticas de entidades do mundo real e suas interrelações (LI, 2020). Atualmente os tópicos mais pesquisados em GeoAI são processamento de imagens geoespaciais, modelagem e análise de transportes e análise de mídias sociais e geotexto (HU et al., 2019).

2.1.2 Sistema de Informação Geográfica

O Sistema de Informação Geográfica (GIS) é um sistema computacional capaz de capturar, armazenar, consultar, analisar e visualizar dados geoespaciais (CHANG, 2008). Segundo Câmara e Medeiros (1996) os modelos de dados manipulados por um GIS são:

- **Mapas Temáticos**: são dados obtidos por levantamento de campo e inseridos no sistema por digitalização ou a partir de classificação de imagens obtidas por sensores remotos. Esta categoria de dados descreve a distribuição espacial de uma grandeza geográfica, expressa qualitativamente. Por exemplo, mapa de solos ou mapa de vegetação.

- **Mapas Cadastrais**: cada elemento nos Mapas Cadastrais é um objeto geográfico que possui atributos e pode estar associado à várias representações gráficas. É de extrema importância, para caracterizar adequadamente os dados, saber distinguir entre os Mapas Temáticos e os Mapas Cadastrais. Os Mapas Temáticos, geralmente, lidam com informações imprecisas como, por exemplo, em um mapa temático de solos, onde os limites indicados no mapa são uma aproximação da realidade. Nos Mapas Cadastrais as medidas são precisas e determinadas. Um exemplo de mapa cadastral é um mapa de unidades domiciliares contendo, para cada domicílio, um conjuntos de atributos que o descreve.

- **Redes**: quando se trata de Redes, cada objeto geográfico possui uma localização geográfica exata e está sempre associado à atributos descritivos. Diferente dos outros tipos de dados, os objetos geográficos representam um nó em uma rede que organiza-se como uma estrutura de grafo, de forma que de cada nó, partem e chegam diversas arestas que representam fluxos reais de energia, matéria ou informação entre estes nós. A cada aresta está associada uma capacidade máxima de fluxos. Exemplos deste tipo de dados são redes de estradas, redes de eletrificação, redes de esgoto, etc. Estas estruturas de dados são essenciais para a computação de rotas ou capacidade de fluxo entre duas localidades.

- **Imagens**: as imagens são dados de natureza matricial que representam formas de captura indireta de informações espaciais. Elas são obtidas por meio de sensores a bordo de satélites, fotografias aéreas ou "scanners"aerotransportados. Pelas formas que ocorrem a aquisição de uma imagem é necessário recorrer às técnicas de fotointerpretação e de classificação, para alcançar o objetivo de individualizar os objetos geográficos contidos na imagem.

- **Modelo Numérico de Terreno**: também conhecido como MNT, esse dado é utilizado para indicar a representação quantitativa de uma grandeza que varia continuamente no espaço. Os MNT são, geralmente, associados à altimetria para modelar o relevo de um terreno, mas também podem ser aplicados para modelar outras grandezas, como superfícies que descrevem a variação de temperatura ou umidade.

2.1.3 Sistema de Apoio à Decisão

Segundo Shim et al. (2002) os Sistemas de Apoio à Decisão (DSS) são sistemas computacionais que podem ser utilizados para apoiar a tomada de decisões complexas e a resolução de problemas. O DSS clássico é composto pelos seguintes componentes: (i) Data Management, que é responsável pelo gerenciamento de dados internos e externos; (ii) Model Management, que é responsável por guardar e executar os modelos matemático-computacionais analíticos ou de simulação que possuem as regras que capturam e/ou descrevem o comportamento das entidades que participam do problema em estudo; (iii) User Interface, que representa a interface gráfica que permite consultas interativas, relatórios e visualizações gráficas (SHIM et al., 2002).

Ainda segundo Shim et al. (2002) esses componentes podem ser alterados conforme o tipo do DSS. Como a maioria dos DSS são implementados para resolver problemas específicos, novas categorias surgem a todo momento, de acordo com a necessidade de uma cenário específico. Segundo Power (2002) as categorias de DSS mais comuns são:

- **Data-driven DSS**: O DSS orientado a dados enfatiza a análise de grandes quantidades de dados estruturados. Esses sistemas incluem drawer and management reporting systems, data warehousing e analytical systems. Podemos apresentar os sistemas de Business Intelligence (BI) como exemplos de DSS orientado a dados. Um DSS orientado a dados fornece acesso e funcionalidades para manipulação e análise de grandes bancos de dados estruturados.

- **Document-driven DSS**: O DSS orientado a documentos são projetados para ajudar os gerentes a coletar, recuperar, classificar e gerenciar documentos não estruturados, incluindo páginas da Web. O DSS orientado a documentos integra várias tecnologias de armazenamento e processamento para fornecer recuperação e análise completas de documentos. Um mecanismo de busca é uma poderosa ferramenta associada a um DSS orientado a documentos.

- **Model-driven DSS**: O DSS orientado a modelo enfatiza o acesso e a manipulação de modelos matemático-computacional analíticos e de simulação, entre eles, modelos representacionais, modelo preditivos e modelos de otimização. O DSS orientado por modelo usa dados e parâmetros fornecidos pelos tomadores de decisão para ajudá-los a analisar uma situação real ou cenário hipotético.

- **Knowledge-driven DSS**: O DSS orientado ao conhecimento utiliza regras de negócios e bases de conhecimento para sugerir ou recomendar ações a decisores. Esses DSS são sistemas pessoa-computador com experiência especializada em solução de problemas. A "expertise" consiste em conhecimento sobre um domínio específico, compreensão dos problemas dentro desse domínio e "habilidade" para resolver alguns desses problemas. As ferramentas de mineração de dados podem ser usadas para criar DSS híbrido orientado por dados e orientado por conhecimento.

2.1.4 Arquitetura de Microsserviços

Para Garlan (2008), a arquitetura de software de um sistema é o conjunto de estruturas necessárias para seu desenvolvimento, que abrange os elementos do software, as relações entre eles e suas propriedades. Ou seja, o projeto de arquitetura é como o sistema será concebido, organizado e utilizado. A arquitetura de um software determina como as funcionalidades que serão providas são distribuídas entre os componentes que formam o software. Assim, a arquitetura determina como se dá a interdependência entre esses componentes para que esses possam realizar suas responsabilidades.

Segundo Fowler e Lewis (2015), para começar a explicar melhor a arquitetura de microsserviços, é mais efetivo compará-la com a arquitetura monolítica. Um monolito é um software onde os módulos não podem ser executados independentemente (DRAGONI et al., 2017), existe apenas um único componente central que oferece todas as funcionalidades previstas. No começo a manutenção ou evolução de um arquitetura monolítica pode não apresentar problemas, mas, à medida que novas funcionalidades são adicionadas e ocorre o crescimento do monolito, surgem dificuldades para que a correção ou a evolução de determinadas funcionalidades não impactem o funcionamento de outras, introduzindo falhas nestas últimas mesmo quando não tiveram seu código alterado.

Além de ser difícil rastrear as falhas, quando as encontramos, qualquer alteração em um módulo específico requer a reinicialização de toda a aplicação. É importante considerar que, para grandes projetos, reiniciar envolve um tempo considerável de inatividade, dificultando assim o desenvolvimento, teste e manutenção do mesmo (DRAGONI et al., 2017). Quando o assunto é escalabilidade, os monolitos também sofrem com limitações. Geralmente a estratégia para lidar com o aumento de solicitação de entrada é criar instâncias da mesma aplicação e dividir a carga. Entretanto, uma situação como essa pode ser o caso onde o aumento de tráfego estressa apenas um subconjunto dos módulos, tornando a nova alocação inconveniente (DRAGONI et al., 2017).

Segundo Dmitry e Manfred (2014) outro grande problema é a pilha de tecnologia. Em uma aplicação monolítica existe um bloqueio de tecnologia para os desenvolvedores, que são obrigados a usar a mesma linguagem e estrutura da aplicação original. Compreendendo então todas as limitações apresentadas e com o intuito de solucionar os problemas da arquitetura de monolíticos, surgiu a arquitetura de microsserviços.

O termo “microsserviços” foi introduzido pela primeira vez em um workshop de arquitetura em 2011. Antes disso, as abordagens semelhantes eram conhecidas por outros nomes (DRAGONI et al., 2017). É possível compreender que uma arquitetura de microsserviços é uma aplicação descentralizada, onde todos os seus componentes são microsserviços. Segundo Thönes (2015) um microsserviço é um pequeno conjunto de funcionalidades agrupadas em um serviço que pode ser implantado, escalado e testado independentemente e que tem uma única responsabilidade, o que gera solução para algumas limitações apresentadas na arquitetura

monolítica.

De acordo com Newman (2021) ao implementar os microsserviços devemos lembrar de dois termos muito utilizados no contexto de sistema orientado a objetos. O primeiro termo é o fraco acoplamento, que é definido pelo fato de que uma mudança em um determinado serviço não deve exigir que ocorra uma mudança em outro. O segundo termo utilizado se refere a alta coesão, que diz que funcionalidades relacionadas devem ficar juntas e as funcionalidades não relacionados devem permanecer alocadas em lugar diferentes. Assim, serviços altamente coesos oferecem todas as funcionalidades necessárias para o cumprimento de sua missão. Funcionalidades relacionadas somente com cumprimento da missão de outro serviço, devem pertencer a este outro serviço. Assim, a alta coesão promove o fraco acoplamento entre diversos serviços.

Outra característica apresentada é que por ser totalmente desacoplado, cada microsserviço pode ser desenvolvido, implantado e atualizado independentemente. Assim, a correção de uma falha ou a adição de uma nova funcionalidade, não causará impacto algum nos outros serviços, em um cenário em que a forma de comunicação entre eles não foi alterada. Outra vantagem dos microsserviços serem independentes é que ao desenvolver uma aplicação em microsserviço as tecnologias utilizadas em cada um deles podem ser escolhidas separadamente (FOWLER; LEWIS, 2015).

2.2 Trabalhos Relacionados

Esta seção de texto descreve dois trabalhos em que foram propostas estratégias para o projeto de arquitetura de microsserviços, buscando agrupar funcionalidades em serviços altamente coesos e fracamente acoplados.

2.2.1 Extração de microsserviços a partir de arquiteturas monolíticas

O artigo de Mazlami, Cito e Leitner (2017) apresenta um modelo formal de extração de grupamentos de funcionalidades, a partir de aplicações monolíticas já existentes, que permite gerar recomendações de candidatos a microsserviços em um cenário de refatoração e migração. Para isso, são apresentadas três estratégias formais de acoplamento. Estas estratégias são incorporadas em um algoritmo de agrupamento baseado na estrutura de dados denominada grafo. As estratégias de extração dependem de informações contidas no repositório de código do monolito para construir tal grafo.

Todas as estratégias de extração são compostas por três estágios: o estágio de monolito, o estágio de grafo e o estágio de microsserviços. Entre esses estágios existe a etapa de construção, onde é utilizado uma das estratégias de acoplamento para extrair informações do monolito e criar um grafo não direcionado e ponderado. No grafo, cada vértice representa uma classe do monolito. Os pesos das arestas são determinados por quão forte é o acoplamento entre os vizinhos, em cada estratégia de acoplamento este peso é calculado de forma distinta.

As três estratégias de extração são Estratégia de Acoplamento Lógico, Estratégia de Acoplamento Semântico e Estratégia de Acoplamento do Contribuinte. Na Estratégia de Acoplamento Lógico os arquivos de classe que mudam juntos devem permanecer juntos e, para isso, é utilizado o histórico de revisões do código fonte da aplicação. O peso da aresta entre duas classes distintas é calculado verificando, no histórico de alterações, quantas vezes as classes foram alteradas no mesmo evento de mudança. O grafo é construído com pesos maiores em arestas que são alteradas mais vezes juntas. A estratégia busca garantir limites de módulos fortes. Assim, quando o desenvolvedor necessitar fazer alterações em um sistema, ele precisa conhecer apenas o módulo a ser alterado.

A Estratégia de Acoplamento Semântico utiliza de um dos princípios do Projeto Dirigido por Domínio (DDD - do inglês, "Domain-Driven Design") para construir o grafo. O contexto delimitado é apresentado por Fowler e Lewis (2015) como uma lógica promissora para microsserviços. De acordo com essa lógica, cada microsserviço deve corresponder a um único contexto delimitado do domínio do problema. Isso resulta em microsserviços que se concentram em uma responsabilidade. Para identificar as entidades de domínio de problema e seu contexto delimitado, a estratégia analisa todas as classes distintas e extrai um conjunto de palavras. Após isso, para cada par de classes é realizado um procedimento para calcular a semelhança entre elas. Dessa forma, a estratégia acopla as classes que contêm código sobre as mesmas "coisas", ou seja, mesmas entidades de modelo de domínio.

A Estratégia de Acoplamento do Contribuinte tem como objetivo incorporar as características advindas de equipes multifuncionais que são organizadas em torno de recursos de domínios e negócios para arquitetura de microsserviços (FOWLER; LEWIS, 2015). A estratégia realiza uma análise no histórico de alterações do monolito procurando pelos autores das modificações nos arquivos de classe. O procedimento para calcular o acoplamento é aplicado a todos os arquivos de classe do monolito. Ao final do cálculo, para cada classe, cria-se um conjunto com todos os desenvolvedores que fizeram contribuição em arquivos de classe. O peso da aresta entre duas classes é definido como a cardinalidade da intersecção dos conjuntos de desenvolvedores que contribuíram nas respectivas classes.

Ao final da etapa de extração começa a etapa de agrupamento onde a representação de grafo é então particionada em subgrafos para obter os candidatos a microsserviços. O algoritmo apresentado prioriza arestas com o peso baixo, sendo assim, a cada interação é excluída a aresta de menor valor. Por não ser possível determinar de forma automática quando parar o particionamento do grafo, o algoritmo usa um parâmetro de entrada para definir o número de iterações.

Para avaliar a abordagem foram levantadas duas questões, sendo elas:"Qual o desempenho do protótipo implementado em relação ao tempo de execução?"e "Qual é a qualidade das recomendações de microsserviços geradas pelo protótipo?". Para responder estas questões, foram implementadas estratégias de extração de informação de código fonte na linguagem Java e, para testar essas estratégias, um conjunto de projetos com código aberto foi utilizado. Foi realizado

uma série de experimentos que foram separados em dois tipos. O primeiro tipo é para responder a primeira questão e, então, foram feitos experimentos de medição de desempenho para obter a resposta desejada. Já o segundo tipo de experimentos foi utilizado para responder a segunda questão e, para isso, foram levantadas métricas de qualidade de software.

Em relação à performance, todos os experimentos de desempenho demonstram níveis satisfatórios, indicando que a abordagem pode ser usada para diferentes cenários. No entanto, em alguns casos a estratégia de acoplamento semântico tem pior desempenho. Para a avaliação de qualidade foram utilizadas as métricas de redução do tamanho da equipe e redundância de domínio médio. Os resultados para a redução do tamanho da equipe indicam que a abordagem funciona muito bem, pois, o tamanho da equipe diminuiu no mínimo pela metade para todas as estratégias de extração. É possível notar também que a estratégia de acoplamento semântico apresenta os melhores resultados, superando significativamente as outras estratégias. Na redundância de domínio médio, pode-se supor intuitivamente que a estratégia de acoplamento semântico terá um desempenho muito bom porque otimiza o grafo de extração para agrupamento específicos de domínio. A melhor estratégia com redundância de domínio é a de acoplamento do contribuinte.

Com os resultados demonstrados podemos perceber que a estratégia de acoplamento semântico se saiu melhor quando as métricas são sobre alguma característica da arquitetura de microsserviço. A abordagem apresentou resultados satisfatórios, mas contem algumas limitações, duas delas apontadas pelos próprios autores. A primeira limitação é o fato de que o modelo de extração é baseado em classes como unidade atômica de computação nas estratégias e no grafo, reduzindo assim a disponibilidade ao refatorar um monolito. A segunda limitação apresentada pelos autores é que o modelo de extração necessita de um sistema auxiliar para representar as entidades dos modelos de dados como classes e, assim, serem tratados como uma classe comum pelo algorítimo de extração. Sem este sistema não é possível resolver o problema de como compartilhar ou atribuir bancos de dados preexistentes aos novos microsserviços criados.

Além das limitações apresentadas pelos autores, no intuito de atingir o objetivo desse trabalho, essa abordagem possui outra limitação. Para ser utilizada, a mesma carece de um código monolitico prévio, o que difere das determinações estabelecidas para trabalho, que necessita desde o início de métodos para a construção de sistemas em arquitetura de microsserviço.

2.2.2 Desafios do projeto dirigido por domínio de microsserviços: Uma perspectiva dirigida por modelo

O DDD fornece, entre outras coisas, os meios de modelagem para decompor domínios em contextos delimitados. Deste modo, cada contexto compõe conceitos de domínio coerentes e indica candidatos a um microsserviços. Aplicar o DDD à arquitetura de microsserviço apresenta alguns desafios pois os modelos de domínio geralmente não possuem as informações necessárias para subsidiar o projeto de uma arquitetura de microsserviço. No artigo de Rademacher, Sorgalla e

Sachweh (2018) são discutidos os vários desafios de uma perspectiva de projeto de microsserviços orientado a domínio e também apresenta maneiras de lidar com eles com base no Desenvolvimento Dirigido por Modelo (MDD - do inglês, "Model-Driven Development").

Para demonstrar os desafios do design de microsserviços orientado a domínio, é utilizado como exemplo um modelo de domínio relacionado ao transporte de carga. O modelo de domínio contém três contextos delimitados, sendo eles a carga, o cliente e a localização. Os modelos de domínio geralmente são esboçados como diagramas de classe da Linguagem Unificada de Modelagem (UML - do inglês, "Unified Modeling Language"). Os três principais desafios, segundo os autores, são a dedução de microsserviços de modelos de domínio, modelagem de componentes de infraestrutura e modelagem de domínio em equipes autônomas.

De acordo com os autores, para a implementação de um microsserviço, informações específicas sobre as características são obrigatórias, características essas como: interfaces e operações; parâmetros de operação e tipos de retorno; endpoints, protocolos e formatos de mensagem. Os modelos de domínio normalmente omitem estas informações, dificultando a dedução dos microsserviços a partir dos modelos de domínio. Nos modelos de domínio, associações entre conceitos de diferentes contextos definem relacionamentos de troca entre instâncias desses conceitos. Quando um contexto é implementado como um microsserviço, esses relacionamentos correspondem a interações entre serviços que realizam a troca de instâncias. No entanto, o modelo não especifica quais operações a interface de serviço contém e como elas podem ser invocadas, nem especificam tipos de retorno.

Os modelos de domínio não incluem componentes de infraestrutura como, por exemplo, implantação de serviço e provisionamento de infraestrutura. Ou seja, com modelo de domínio não conseguimos saber como aquele microsserviço deve ser disponibilizado ou qual tecnologia deve ser usada no seu desenvolvimento. Outro desafio está relacionado ao acesso ao modelo de domínio e gerenciamento de mudança. Deve-se decidir quais partes do modelo de domínio são visíveis para cada equipe. Uma abordagem simples é tornar o modelo de domínio completamente acessível, mas isso pode trazer alguns problemas, já que essa abordagem permite que uma equipe altere o contexto de outra equipe.

Para lidar com esses desafios são apresentadas três possíveis soluções. A primeira é a utilização dos modelos intermediários que introduzem características técnicas que intencionalmente não estão no modelo de domínio, mas que são fundamentais para o projeto da arquitetura de microsserviço. Neste artigo foi sugerido dois modelos intermediários: modelo de interface e modelo de implantação. O modelo de interface mostra microsserviços com interfaces, operações com tipos e dependências de serviço. O modelo de implantação inclui informações técnicas sobre os endpoints dos microsserviços, protocolos e formatos de mensagens, tecnologias de implantação e componentes de infraestrutura.

A segunda solução é reduzir a informalidade do modelo de domínio através de convenções de modelagem. Estas convenções são feitas entre os modeladores e os desenvolvedores e, juntos,

os dois grupos devem tomar decisões de como o modelo de domínio vai ser interpretado. Um exemplo seria determinar que os métodos de modelo de domínio com visibilidade pública de UML resultam em operações de interface ou padronizar os tipos de comunicação e de retorno.

A terceira solução é criar uma política para regular o acesso ao modelo do domínio e as alterações permitidas para cada equipe. Esta política deve considerar que as equipes irão ter acesso total ou parcial ao modelo de domínio. O acesso parcial tem como vantagem promover o baixo acoplamento, pois cada equipe tem uma visão apenas de seus contextos e um mínimo de conhecimento sobre conceitos externos a seus contextos, mas como desvantagem requer um esforço adicional para dividir o arquivo de modelo em vários arquivos contendo apenas os contextos e conceitos relevantes para a equipe. O acesso total não requer este esforço pois todas as equipes tem acesso ao modelo de domínio geral, mas aumenta o conjunto de conceitos de domínio que podem ser afetados por alguma alteração.

Uma grande limitação deste trabalho é que ele não apresenta nenhum experimento ou forma de avaliação comprovando a eficácia das soluções propostas. No entanto, como este artigo se propõe a resolver os problemas originados ao se aplicar o DDD para produção de uma arquitetura de microsserviço, as soluções nele descritas podem subsidiar as decisões e estratégias a serem adotadas neste trabalho. Especificamente, a solução que utiliza de modelos intermediários para especificar detalhes relevantes aos microsserviços é, aparentemente, uma abordagem promissora a ser usada em trabalhos futuros.

3 Desenvolvimento

Este capítulo apresenta o conhecimento e os raciocínios utilizados para a execução deste trabalho que, por enquanto, tem um caráter puramente teórico. Busca-se justificar as decisões que serão tomadas ao longo deste projeto e situar a equipe do laboratório TerraLAB, do Departamento de Computação da UFOP, dentro do estado-da-arte nos temas sob estudo. Assim, espera-se subsidiar o projeto de uma arquitetura de microsserviços destinada ao desenvolvimento de futuras aplicações em Inteligência Artificial Geoespacial, nicho de atuação em pesquisa e em desenvolvimento tecnológico deste laboratório, através do qual ele estabelece relações de cooperação com diversas empresas e instituições de pesquisas, nacionais e internacionais.

3.1 Desenvolvendo projetos em Microsserviços

Após finalizar a revisão na literatura, em busca de estratégias para como projetar, implementar e avaliar as Arquiteturas de Microsserviços, podemos notar que a maioria dos trabalhos diz respeito aos mecanismos utilizados para passar uma arquitetura de monolito para microsserviços. Segundo Fowler (2015) isso ocorre por dois motivos. Primeiro, ao começar um novo projeto, não é possível ter certeza a respeito da sua assertividade. A melhor forma de saber se o software é útil, é desenvolvendo um Produto Mínimo Viável (MVP - do inglês, Minimum Viable Product) e testá-lo. Nesta primeira fase, é importante priorizar a velocidade, para captar o máximo de feedbacks possíveis. Como a Arquiteturas de Microsserviços é mais complexa, muitos optam por começar com monolitos.

O segundo motivo para não começar com microsserviços é que eles só funcionam bem se o projetista definir limites bons e estáveis entre os serviços. Para isso, é essencial elaborar um conjunto certo de Contextos Delimitados. Mesmo os arquitetos mais experientes têm grande dificuldade em estabelecer limites, logo no início da elaboração de seus projetos. Qualquer refatoração de funcionalidade entre serviços se faz muito mais difícil que em um monolito. Por causa desses dois motivos, uma abordagem muito comum é começar com um monolito e substituí-lo gradualmente em microsserviços.

Pensando em atender os dois requisitos mínimos para o desenvolvimento de microsserviços, alta coesão e fraco acoplamento, este trabalho apresentou dois artigos com estratégias documentadas pela literatura científica. O primeiro artigo apresenta três abordagens distintas de extração de candidatos a microsserviços em aplicações monolíticas. Apesar dos seus resultados satisfatórios, na maioria dos casos, o artigo em questão também apresenta limitações que devem ser consideradas antes de optar por quaisquer destas estratégias. O segundo artigo apresenta três estratégias utilizando MDD para lidar os desafios de aplicar DDD para modelar um sistema em arquitetura de microsserviço. No entanto, este último artigo não apresenta nenhum experimento

para provar sua eficácia. Porém, do ponto de vista técnico e qualitativo, suas soluções aparentam ser abordagens promissoras, principalmente, aquela baseada em modelos intermediários.

3.2 GeoAI: compreendendo seu desenvolvimento

Para lidar com os problemas ambientais e de saúde pública mencionados no início do artigo, ou até mesmo problemas em Geomarketing, o GeoAI terá um papel fundamental na integração de grandes conjuntos de dados geoespaciais, viabilizando análises inteligentes e permitindo respostas interativas, além de fornecer recursos de suporte à tomada de decisões por parte de planejadores e entre outras partes interessadas (LI, 2020). A GeoAI é um campo de estudo recente e, por isso, possui alguns aspectos que ainda estão passando pela fase de teste e consolidação dentro do meio acadêmico. Uma evidência disso é o fato de que, durante a revisão da literatura, foi encontrado apenas um trabalho que descreve ou discute uma arquitetura para sistemas de GeoAI. Mesmo assim, a arquitetura é abordada de uma forma muito superficial. Esta trabalho é um relatório do Departamento de Defesa dos Estados Unidos que discute a aplicação destes sistemas em operações militares (MEILLÓN, 2008). Assim, como de costume, o Departamento de Defesa realiza pesquisas no campo indústria de software, mas mantém detalhes de suas pesquisas sob confidencialidade. Como forma de resolver a escassez de referencial, neste trabalho os autores buscaram estudar, então, as arquiteturas gerais dos GIS e DSS.

Ao desenvolver um GIS, é preciso garantir que o usuário seja capaz de capturar, armazenar, analisar e visualizar todos os tipos de dados espaciais citados na revisão bibliográfica. A Figura 3.1 representa uma arquitetura monolitica proposta por Câmara e Medeiros (1996). Nela podemos observar os serviços oferecidos por um GIS e uma estrutura em camadas estabelecidas entre eles. Infelizmente, não há clareza se as setas indicam relações de dependências de controle entre esses serviços ou se indicam direções de fluxos de dados. A interface aparece no nível mais próximo ao usuário. Nos dois próximos níveis, é onde todos os serviços que manipulam e transformam dados espaciais em informação útil são implementados.

A arquitetura de um DSS depende da categoria onde o mesmo se encaixa. Para este trabalho vamos considerar um DSS híbrido, orientado por dados e orientado por conhecimento pois, como mencionado, o GeoAI lida com grandes quantidades de dados de um domínio específico. A arquitetura que melhor representa um DSS híbrido é ilustrada na Figura 3.2 conforme sugerida por Delgado et al. (2013). No DSS, o serviço mais próximo do usuário também é a interface que auxilia o usuário interagir com o sistema. Após a interface, três serviços são responsáveis por todo o processo de resolução do problema. O Knowledge Management é responsável pela distribuição, acesso e recuperação dos conhecimentos sobre um domínio específico. Ele pode atuar de forma independente ou fornecer suporte a qualquer outro componente. O Data Management é responsável por realizar o gerenciamento dos dados internos e externos. Finalmente, o Model Management é responsável por guardar, executar e integrar os modelos matemático-computacionais,

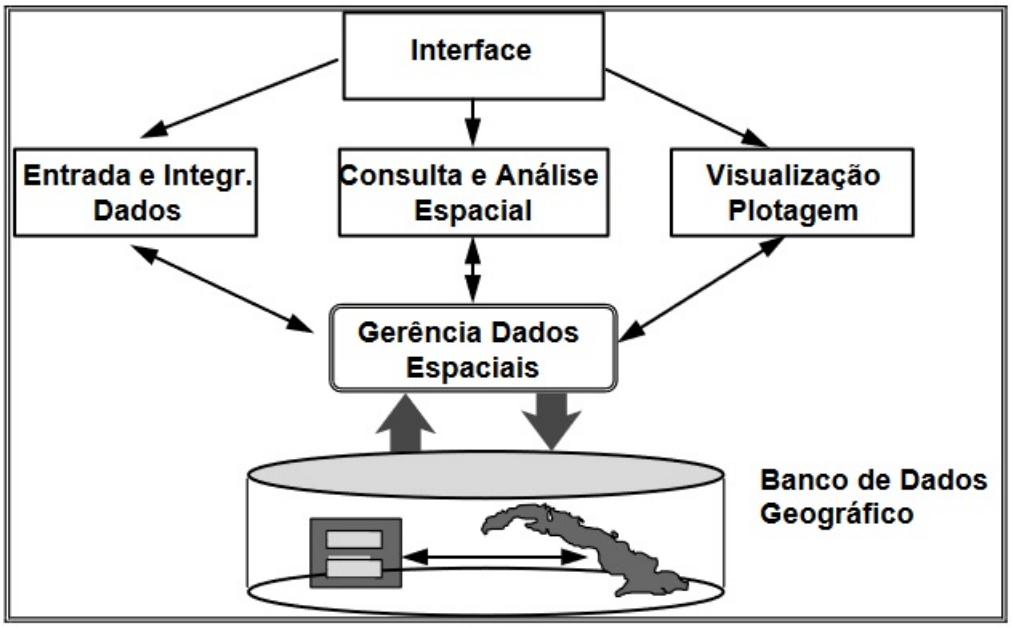

Figura 3.1 – Arquitetura de GIS

(CÂMARA; MEDEIROS, 1996)

sejam analíticos ou de simulação. Novamente, não há clareza se as setas indicam relações de dependências de controle entre esses serviços ou se indicam direções de fluxos de dados.

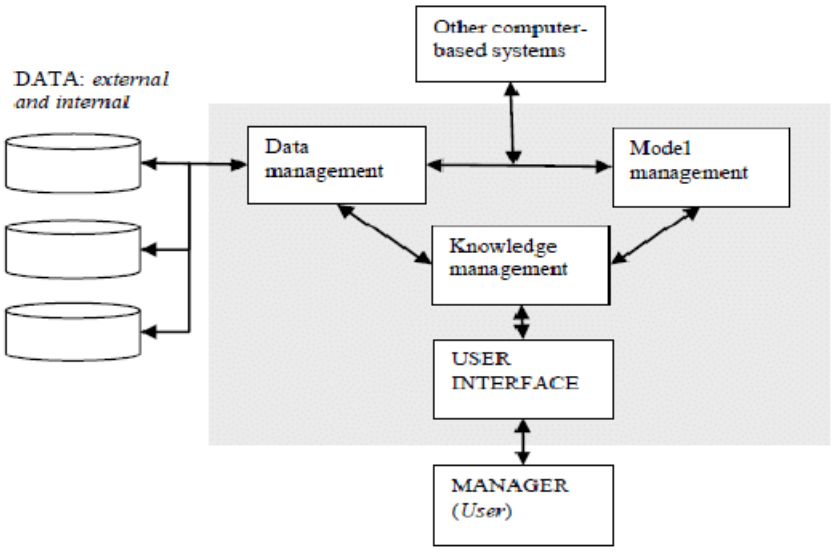

Figura 3.2 – Arquitetura de um DSS híbrido

(DELGADO et al., 2013)

3.3 Arquitetura de Microsserviços e GeoAI: análise do seus requisitos

Ao pensar uma Arquitetura de Microsserviços, os principais requisitos não funcionais que é preciso ter em mente é que os microsserviços precisam ser altamente coesos e fracamente acoplados. Porém, existem outros requisitos extremamente importantes que precisam ser satisfeitos pelo projeto. Os microsserviços precisam ser implantados independente e automaticamente, ou seja, ao fazer alguma alteração em um microsserviço que está na nuvem, esta alteração tem que ser implantada sem precisar modificar outros microsserviços e sem que o programador coloque ela de forma manual, evitando-se assim erros humanos durante o processo de implantação.

O armazenamento de dados em uma Arquitetura de Microsserviços precisa ser descentralizado, ou seja, cada microsserviço deve ter seu próprio banco de dados e somente ele pode ter acesso ao mesmo. Se outro serviço depender dos dados gerenciados por um determinado serviço, ele deve solicitar os dados via a Interface de Programação de Aplicação (API - do inglês, Application Programming Interface) deste último. Isto deve ocorrer porque somente um microsserviço pode conhecer seu contexto.

Outro requisito não funcional é utilizar protocolos RESTish simples ao fazer a comunicação entre os microsserviços. Esta comunicação, apesar de simples, precisa ser clara. Um microsserviço precisa saber qual endpoit, além de quais dados de envio e retorno são esperados pelo outro microsserviço ao fazer uma comunicação. De outra forma, as APIs dos microsserviços funcionam como contratos que estabelecem a maneira na qual a comunicação deve ocorrer.

De uma forma resumida, os requisitos funcionais de um sistema GeoAI são:

- Guardar, editar, excluir e visualizar dados geoespaciais;
- Analisar grandes quantidades de dados geoespaciais;
- Sugerir ou recomendar ações a decisores;
- Acessar e Manipular modelos matemáticos, estatísticos, entre outros;
- Guardar e utilizar regras e bases de conhecimento, fornecidas pelo usuários ou um sistema externo;

3.4 Projetando uma Arquitetura de Microsserviços para Sistema de Inteligência Artificial Geoespacial

Após realizar a análise dos trabalhos apresentado anteriormente, é possível concluir que, diante da ausência de um monolito para ser estudado, não é viável a extração de microsserviços baseada no grupamento de grafos. Logo, avaliaremos a abordagem baseada em DDD. Contudo,

como os modelos de dados também não estão disponíveis para serem utilizados na perspectiva MDD sugerida. Observando também que este trabalho inicia o projeto arquitetural partindo-se do zero, mesmo diante dos alertas encontrados na literatura, então, considera-se uma alternativa viável basear o projeto em análises de GIS e DSS livres, para determinar assim o contexto de cada microsserviço, e em entrevistas com especialistas nesses domínios, para validar as decisões arquiteturais avaliadas.

Ao final, espera-se ter projetos de APIs e modelo de implantação com todas as funcionalidades necessárias para realizar a captura, armazenamento, consulta, análise e visualização de grandes quantidades de todos os tipos de dados geoespacial apresentado anteriormente. Além disso, também é desejado que os projetos de APIs e o modelos de implantação também gerem a possibilidade de apoiar a tomada de decisões acerca de problemas espaciais complexos.

4 Resultados

Para a finalização deste trabalho, foi realizada uma entrevista com um especialista no domínio dos GIS e análise de GIS e DSS com código livre. Com isso, foi possível definir quais os contextos delimitados e possíveis microsserviços para o desenvolvimento de um sistema GeoAI. Também conseguimos identificar quais são os atores que interagem com um sistema GeoAI. Os resultados destes levantamentos podem ser observados nas seções a seguir.

4.1 Atores

O ator é qualquer entidade que interage com as funcionalidades do sistema. Esses atores podem ser usuários, organizações ou sistemas externos, que geram informações ou necessitam de informações geradas pelo sistema. Em suma, os atores principais em um sistema GeoAI são:

Administrador: é o responsável por efetuar manutenção no sistema (criar e recuperar backup) e por gerenciar os usuários (criar, editar e excluir contas).

Analista Geoespacial: é o responsável por coletar, armazenar e manipular dados para gerar informação geoespacial, utilizando de pensamento crítico, raciocínio geoespacial e técnicas analíticas.

Analista de Inteligência: é o responsável por projetar, implementar e manter a base de conhecimento, incluindo os modelos matemático-computacionais necessários para a resolução de problemas.

Decisor: é o responsável por informar o problema e analisar as informações geradas e avaliar as decisões sugeridas, de modo a tomar decisões mais apropriadas.

Na Figura 4.1 podemos verificar a hierarquia desses atores em um sistema GeoAI.

4.2 Serviços

Após a entrevista com esse especialista, análise de GIS e DSS com código livre e considerar os requisitos fundamentais para um sistema GeoAI foram identificados neste trabalho os candidatos a microsserviços, a Figura 4.2[1] mostra os microsserviços sugeridos para o desenvolvimento de um sistema GeoAI. Para um melhor entendimento dividimos os microsserviços em 4 camadas. Os microsserviços da camada superior podem acessar os que estão na camada inferior.

[1] Uma versão online e escalável da imagem está disponível no link <https://tinyurl.com/2ptpbd37>

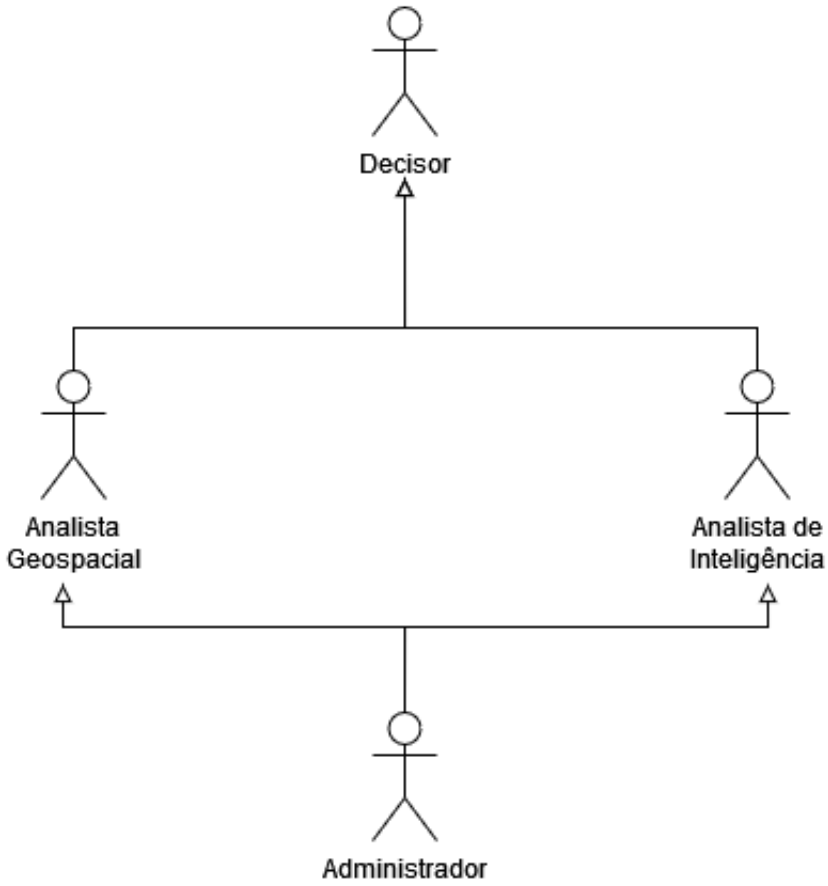

Figura 4.1 – Hierarquia de Atores

Antes de falarmos dos microsserviços, apresentaremos alguns padrões de projeto que escolhemos para este trabalho. O primeiro é o Factory; sendo um padrão criacional, ele fornece uma interface para criar um objeto, sendo que o principal objetivo deste padrão é evitar um forte acoplamento entre o criador e as classes que utilizam o objeto (GAMMA et al., 1995). Pensando no fraco acoplamento entre os microsserviços, este padrão mostrou-se necessário. Sendo assim, todos os microsserviços tem um método de criação próprio e dependem apenas das abstrações dos outros microsserviços.

Outro padrão de projeto importante para o fraco acoplamento, não só entre os microsserviços, mas das camadas de cada microsserviço é o Data Mapper. Este padrão surgiu com o intuito de desacoplar a camada de domínio da camada de infraestrutura, realizando transferências bidirecionais de dados entre os objetos na memória e o banco de dados, mantendo assim uma independência um do outro (FOWLER, 2009). Foi construído então uma interface abstrata, Figura 4.3, na qual todas as interfaces de microsserviços herdarão.

4.2.1 Primeira Camada

Voltando aos microsserviços, na primeira camada contém apenas um microsserviço, o Planning. Utilizando dos microsserviços das camadas abaixo, o Planning é responsável por

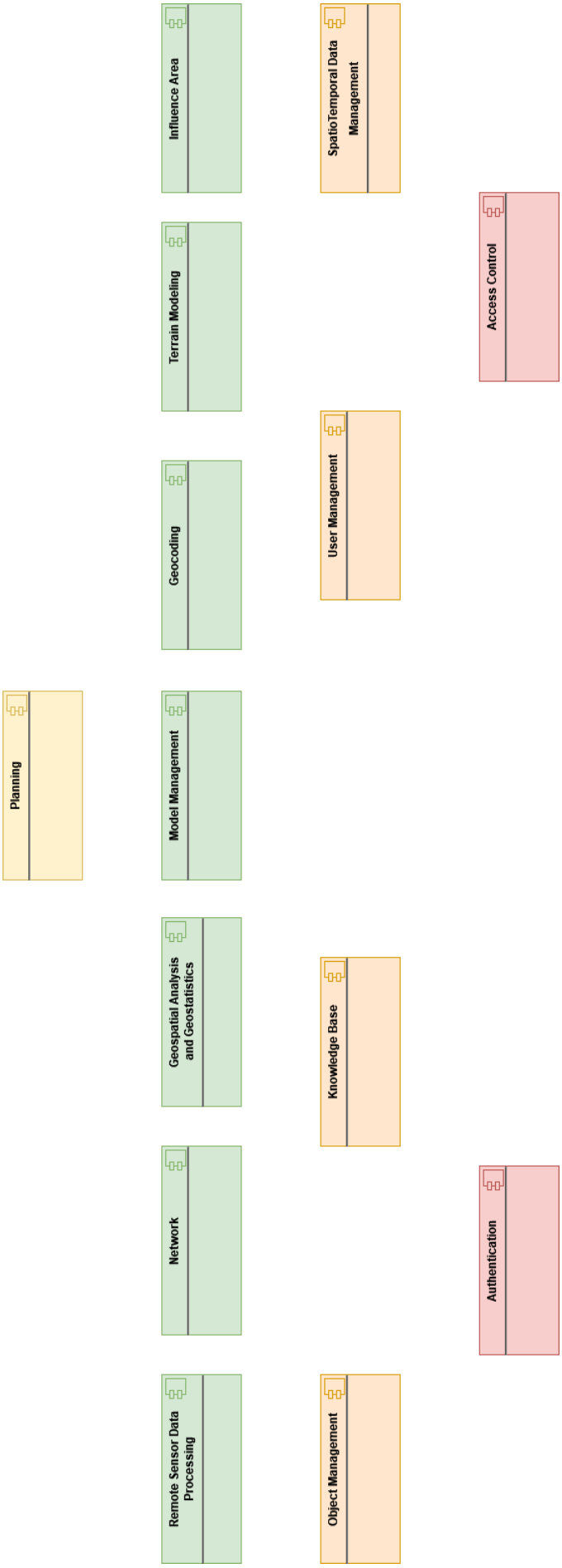

Figura 4.2 – Microsserviços para um sistema GeoAI

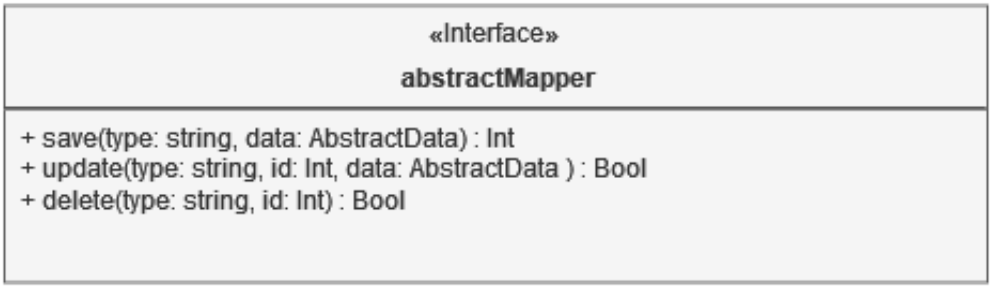

Figura 4.3 – Interface Data Mapper

encontrar e exibir os valores ótimos para os cenários e indicadores passados. Como podemos observar na Figura 4.4, ele expõe cinco métodos que serão necessários para cumprir seu objetivo. O primeiro método é chamado de `createScenario`, responsável por criar cenários. Para a criação de um cenário precisamos passar quais parâmetros vão variar e, para cada um deles, a faixa de variação. Para isso, o método `createScenario` recebe um conjunto de pares de `AbstractParam` e `AbstractDataRange`. O `AbstractParam`, Figura 4.5, é um objeto contendo o nome do parâmetro e seu valor. A faixa de variação é representada pelo`AbstractDataRange` , como podemos observar na Figura 4.6, um`AbstractDataRange` pode ser ou um `ContinuousDataRange` que contém um valor mínimo e máximo, ou um `DiscreteDataRange` que contém uma categoria. O `createScenario` retorna um `AbstractScenario`.

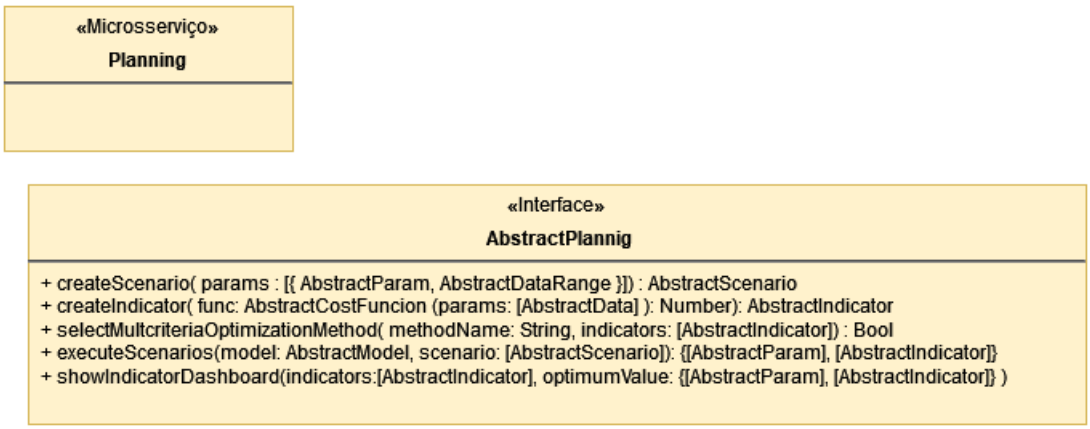

Figura 4.4 – Planning

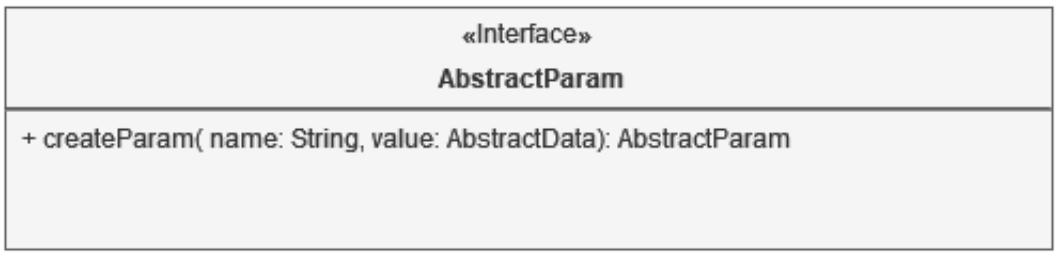

Figura 4.5 – AbstractParam

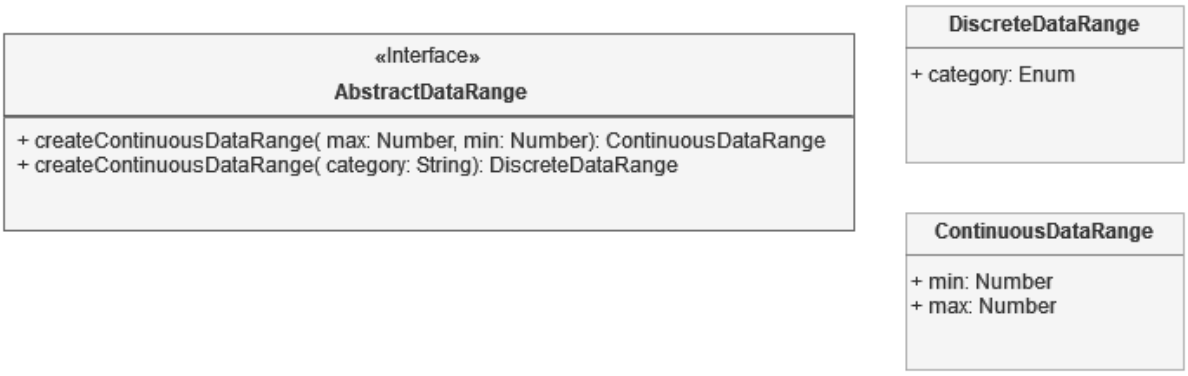

Figura 4.6 – AbstractDataRange

O segundo método descrito na Figura 4.4 é o `createIndicator`, que é responsável por criar indicadores. O método recebe como parâmetro uma função de custo; esta função recebe como parâmetro um conjunto de dados e retorna um `Number`. Por ser uma fábrica, o método retorna um `AbstractIndicator`. O próximo método é o `selectMultcriteriaOptimizationMethod`, cujo objetivo é selecionar e configurar um método de otimização de multicritério. Para isso, os parâmetros do `selectMultcriteriaOptimizationMethod` são o nome do método que deseja selecionar e um conjunto de indicadores para configurar o método. Se conseguir selecionar e configurar o `selectMultcriteriaOptimizationMethod` retorna `true`, caso contrario retorna false.

O método `executeScenarios` executa um conjunto de cenários para um determinado modelo, seu retorno é uma tupla com um conjunto parâmetro e um conjunto de indicadores para o ponto de ótimo das funções de custo, o usuário utiliza desses indicadores para tomar decisões. E por último o método `showIndicatorDashboard` que exibe o valor ótimo até o momento e os valores atuais de todos os indicadores.

4.2.2 Segunda Camada

A segunda camada pode ser chamada de camada de análise. Nela estão contidos os microsserviços responsáveis por analisar os dados que serão usados no microsserviço de Plannig. O microsserviço Remote Sensor Data Processing, Figura 4.7, é responsável pelos dados em formato matricial obtidos por sensores a bordo de satélites, fotografias aéreas ou ”scanners” aerotransportados. Seguindo o padrão de projeto, o primeiro método é a fábrica; nele passamos o tipo e o dado a ser criado e será retornado um `AbstracRemoteSensorData`. A variável `type` foi criada porque existem vários tipos de dados de sensores remoto e é função da fábrica saber como criar cada um deles. O segundo método é para visualizar um dado de sensor remoto. No método `analyze` roda algum modelo usando um conjunto de parâmetros visando entender melhor os dados, um exemplo de modelo e a classificação. Por último, temos o método `operation`; este método é usado para fazer alguma manipulação em um ou mais dados como, por exemplo, cortar uma imagem ou uma interseção entre duas imagens.

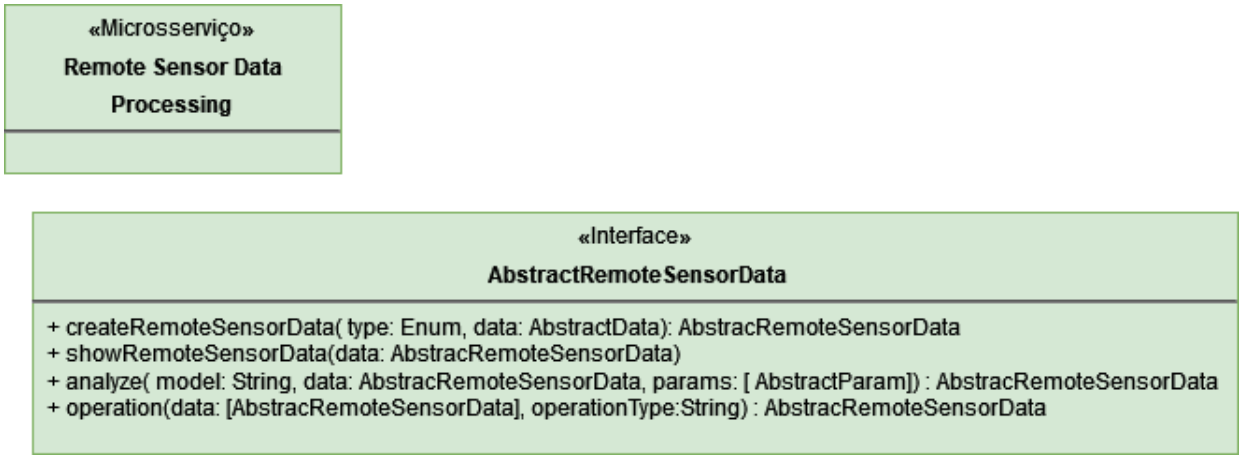

Figura 4.7 – Remote Sensor Data Processing

Como podemos ver nas imagens Figura 4.8, Figura 4.9, Figura 4.10 e Figura 4.11, os microsserviços Network, Geospatial Analysis and Geostatistics, Influence Area e Terrain Modeling contem os mesmo métodos, mudando, é claro, seus domínios. O microsserviço Network é responsável por analisar, processar e extrair informações de dados de redes. O Geospatial Analysis and Geostatistics é responsável por analisar, processar e extrair informações a partir dos dados espaciais em formato vetorial. O Influence Area é responsável por analisar, processar e extrair informações a partir de isócronas e Terrain Modeling é responsável por gerar, analisar, processar e extrair informações dos dados de terreno.

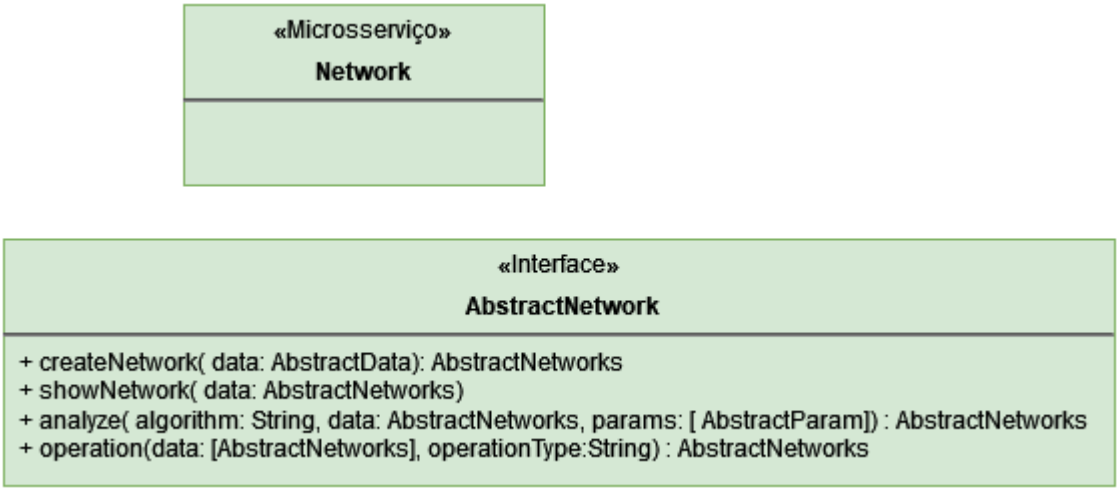

Figura 4.8 – Network

O microsserviço Geocoding é responsável pelo processo de transformar dados de localização de endereço em dados espaciais ou dados espaciais em localização de endereço. Como podemos observar na Figura 4.12, os dois primeiros métodos são fábrica e visualização. O método `autocomplete` é invocado passando uma parte do endereço e retorna um conjunto de sugestões

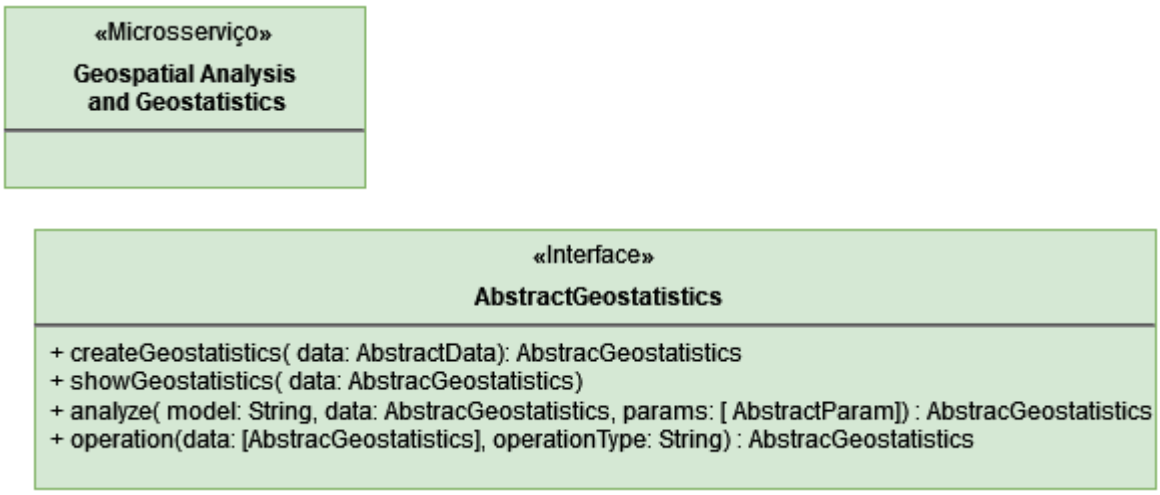

Figura 4.9 – Geospatial Analysis and Geostatistics

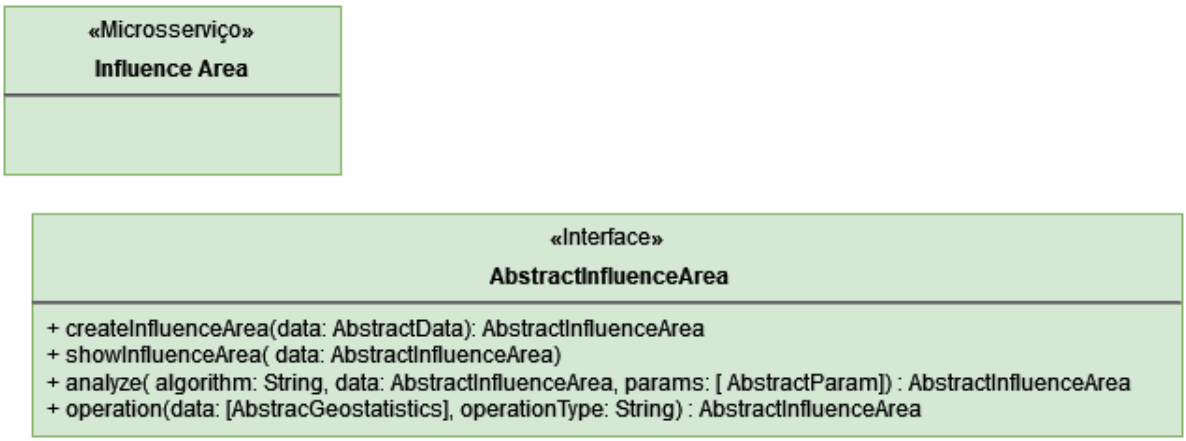

Figura 4.10 – Influence Area

de complemento para o endereço. O processo de transformação de localização em dados espaciais é realizado no método `geocode` e o processo reverso no `reverseGeocode`.

O último microsserviço dessa camada é o Model Management, responsável por permitir a formulação de problemas e suas soluções; por gerenciar as representações de modelos matemático-computacionais, que implementam tais formulações; por executar tais modelos e, finalmente, por produzir soluções alternativas desses problemas. A fábrica deste microsserviço é um pouco diferente das outras. Cada modelo pode ser implementado de formas, com linguagens e paradigmas diferentes. Por esse motivo, a fábrica recebe um código-fonte do modelo a ser criado. Para rodar um modelo, precisamos passar qual modelo queremos rodar e os parâmetros necessários, o método `run` executa essa tarefa e devolve a resposta do modelo. As assinaturas dos métodos podem ser observadas na Figura 4.13.

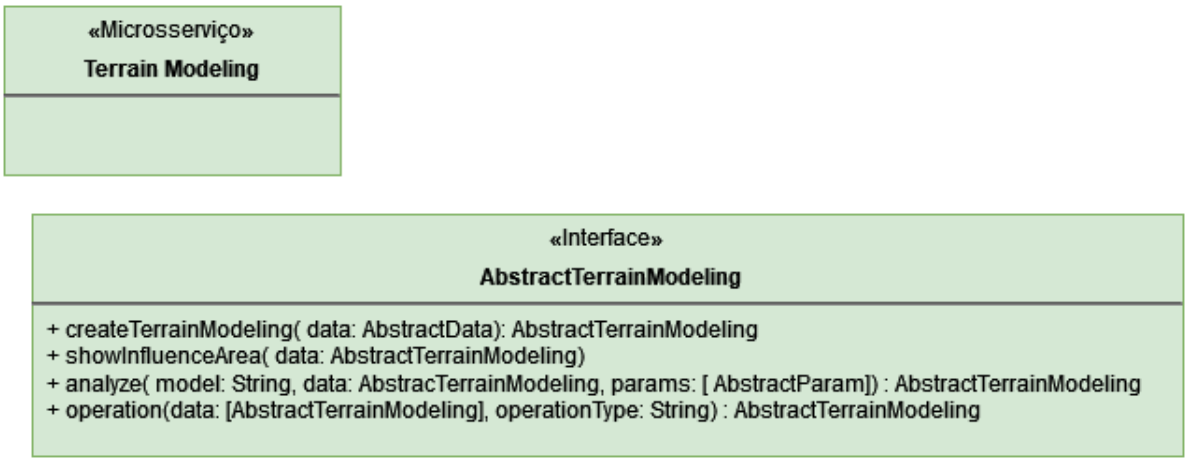

Figura 4.11 – Terrain Modeling

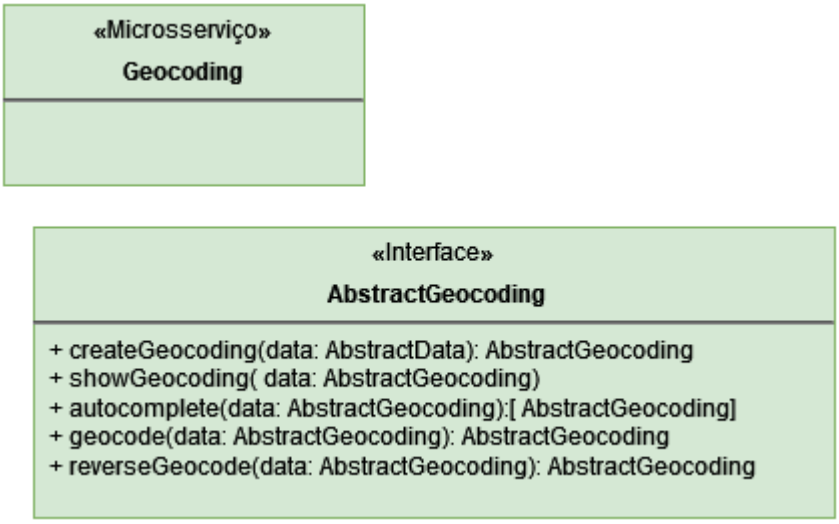

Figura 4.12 – Geocoding

4.2.3 Terceira Camada

Os microsserviços que estão nesta camada são responsáveis por gerenciar os dados que o cliente ou os microsserviços da camada acima precisam. A Figura 4.14 descreve o microsserviço User Management, como o próprio nome diz, é o serviço que gerencia os usuários do sistema. Nele podemos encontrar um usuário passando um identificador e criar um usuário passando o nome, o e-mail e o papel que aquele usuário tem no sistema, os métodos são `findById` e `createUser` respectivamente.

A base de conhecimento do sistema é gerenciada pelo microsserviço Knowledge Base, Figura 4.15. Os dois primeiros métodos do microsserviço são responsáveis por criar uma regra, passando o código-fonte e por verificar se os parâmetros passados atende algum critério de uma determinada regra. A ontologia do sistema é criada ao passar um termo e seu significado no contexto do problema que está sendo resolvido, e quando precisamos buscar o significado de um

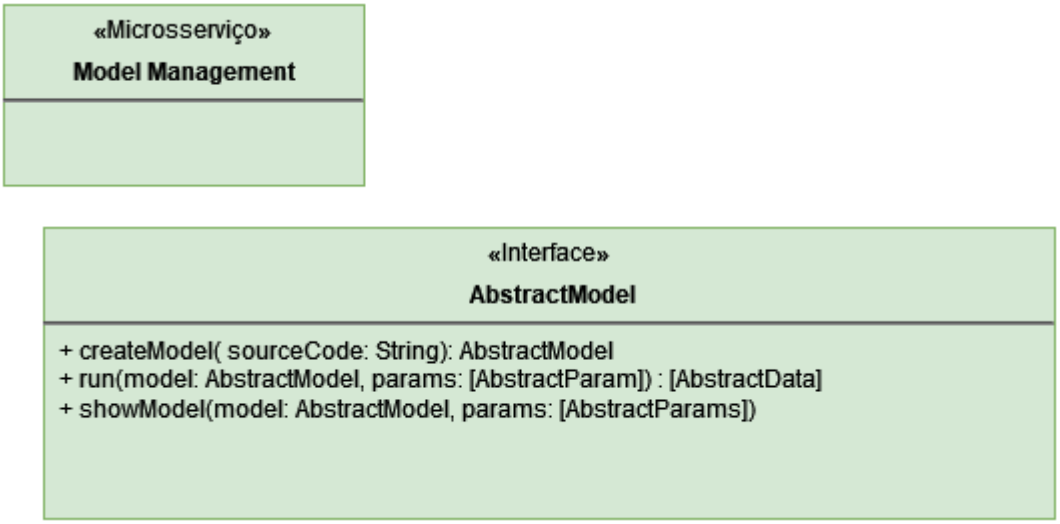

Figura 4.13 – Model Management

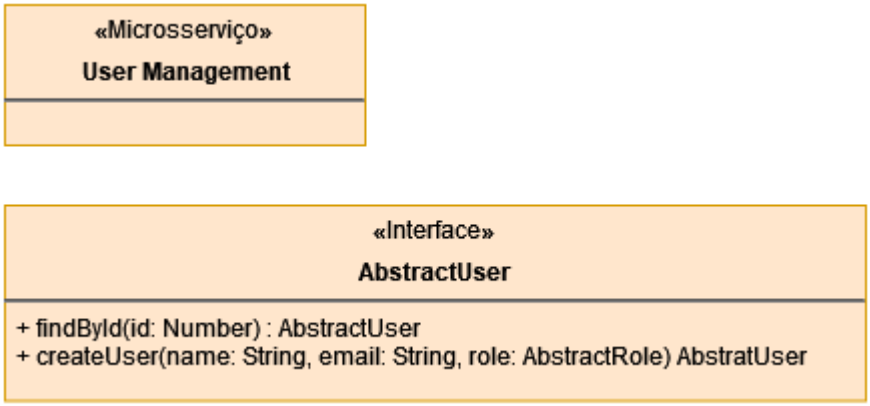

Figura 4.14 – User Management

determinado termo basta passar ele no método `concepts`.

O microsserviço SpatioTemporal Data Management é responsável por armazenar e gerir todo o tipo de dados geoespaciais necessários para o sistema. Pode parecer que estamos ferindo um dos princípios da Arquitetura de Microsserviços, gerenciamento de dados descentralizado, mas esse não é o caso. Ao utilizar o sistema, o cliente pode precisar ou querer armazenar um conjunto de dados geoespaciais diferentes, mas que são interlinhados. A ideia é esse microsserviço armazenar os dados brutos e quando os microsserviços da camada superior precisar destes dados buscar nele e após isso guardar nos seus próprios bancos de dados. Como podemos conferir na Figura 4.16, precisou adicionar uma variável tipo para que o microsserviço saiba que tipo de dados estão sendo armazenados.

Quando trabalhamos com problemas espaciais, algumas informações não estão em mapas ou imagens, mas são de suma importância para a resolução do problema. Para armazenar essas informações foi criado o microsserviço Object Management, Figura 4.17. Os métodos dele são semelhantes ao SpatioTemporal Data Management, a diferença é que os objetos serão salvos

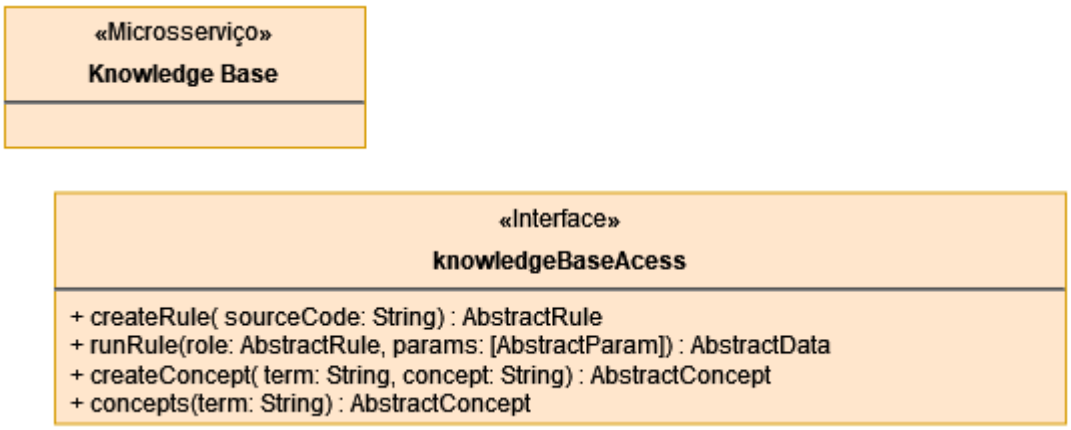

Figura 4.15 – Knowledge Base

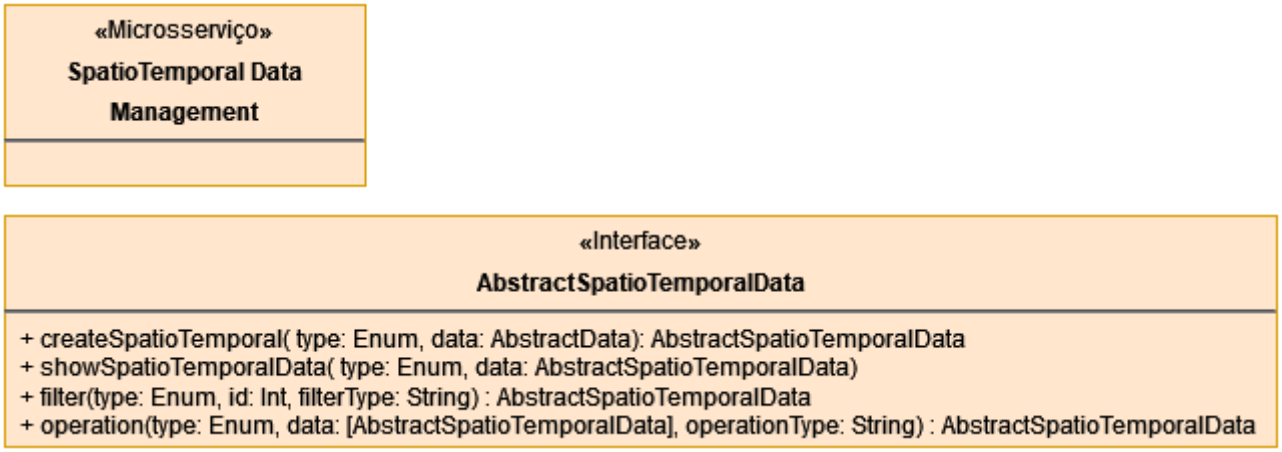

Figura 4.16 – SpatioTemporal Data Management

como em um banco de dados objeto-relacional.

4.2.4 Quarta Camada

Na quarta e última camada, temos dois microsserviços e podemos chamá-la de camada de acesso. O microsserviço Authentication é responsável por criar uma autenticação para o usuário, autenticar, autorizar e alterar senha de um usuário, na Figura 4.18 contém as assinaturas dos métodos. Com o intuito de controlar as funcionalidades no sistema que cada usuário tem acesso, foi criado o microsserviço Access Control, Figura 4.19. Nele é possível criar um papel passando as permissões que ele terá, mostrar um papel e buscar todos os papéis.

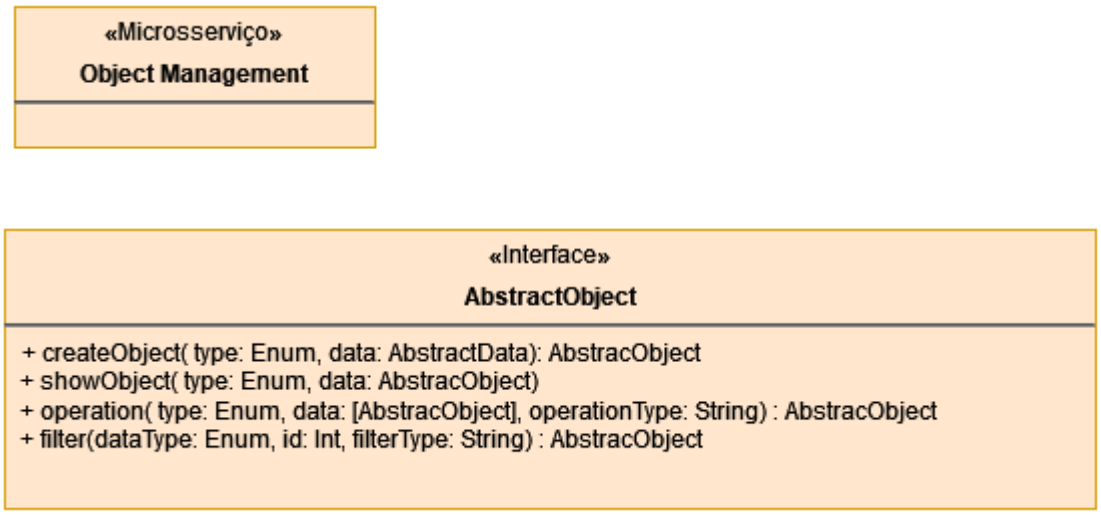

Figura 4.17 – Object Management

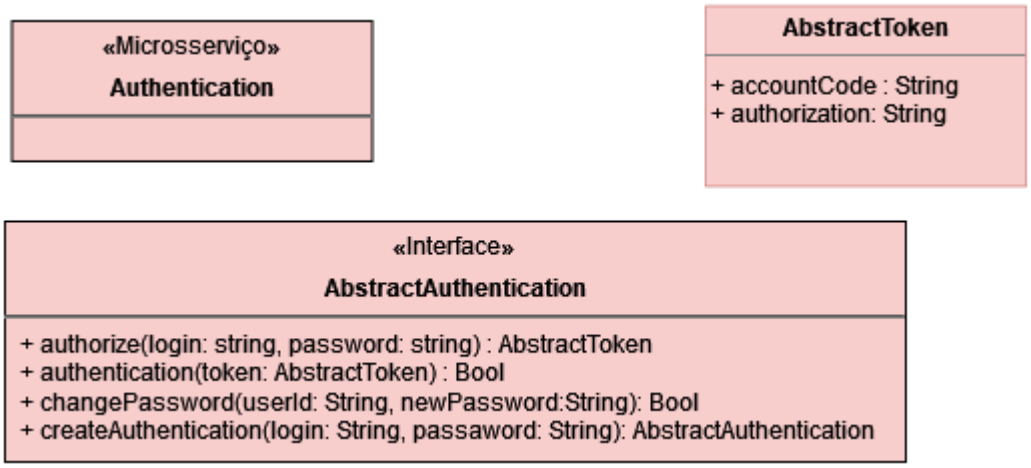

Figura 4.18 – Access Control

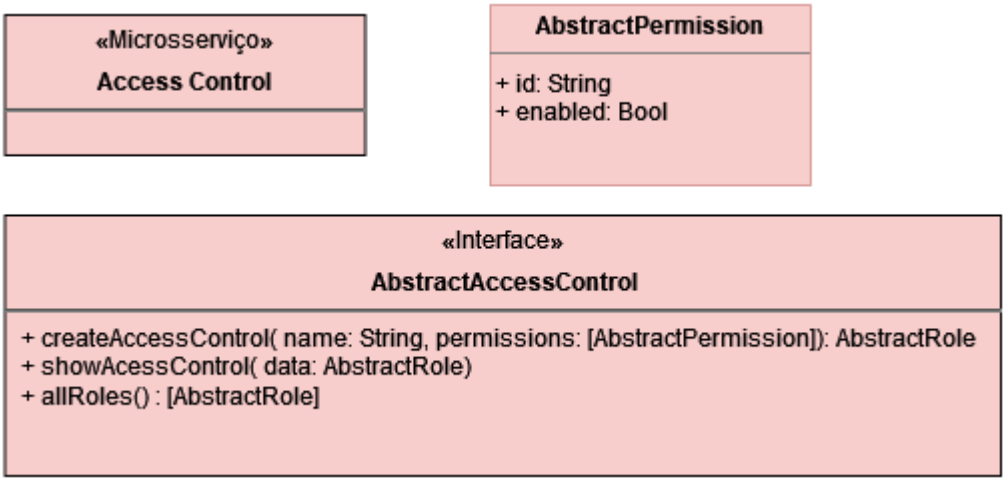

Figura 4.19 – Authentication

5 Conclusão

Este trabalho apresentou uma proposta de arquitetura de microsserviços para o desenvolvimento de aplicações de Inteligência Artificial Geoespacial. Primeiramente, foi realizada uma revisão bibliográfica e apresentou-se conceitos sobre Inteligência Artificial Geoespacial, Sistema de Informação Geográfica, Sistema de Apoio à Decisão e Arquitetura de Microsserviços. Posteriormente, foram efetuadas buscas a fim de encontrar arquiteturas de sistemas de GeoAI, mas, nesta busca, foi encontrado apenas um trabalho. Ocorreu então uma alteração nas buscas, passando a procurar por métodos para o desenvolvimento de arquitetura de microsserviços para qualquer domínio.

Após realizar as pesquisas estabelecidas com o novo filtro, foi possível selecionar uma nova amostragem apresentada no capítulo ”Revisão Bibliográfica”. Após a escolha de dois trabalhos dentro da nova temática estabelecida, realizou-se uma análise e foi decidido que a abordagem a ser usada para propor uma Arquitetura de Microsserviços para aplicações de GeoAI, será a aplicação de DDD com modelos intermediários, já que a mesma apresenta viabilidade aceitável.

Os métodos escolhidos para auxiliar a determinar quais contextos delimitados de cada microsserviço foram as análises de GIS e DSS livres e as entrevistas com especialistas nestes domínios. Ao final das análises, foi possível definir os contextos, microsserviço e algumas funcionalidades necessárias para o sistema. Outro resultado identificado foi identificar os atores dos sistemas e como eles irão interagir com os mesmos. Os resultados foram apresentados utilizando diagramas UML. O primeiro diagrama contendo os microsserviços separados em camadas para auxiliar a compreensão dos papéis que cada um deles exercerá no sistema. Para demonstrar as funcionalidades básicas dos microsserviços, assim como interfaces auxiliares para o desenvolvimento do sistema, criou-se um segundo diagrama. Para uma melhor visualização os diagramas podem ser acessados neste https://tinyurl.com/2ptpbd37.

5.1 Trabalhos Futuros

Como trabalhos futuros se apresenta como proposta a implementação dos microsserviços elaborados neste artigo, possibilitando-se a validação dos processos apresentados neste trabalho. A partir disso, sera possível corroborar ou não a proposta veiculada.

Referências

CÂMARA, G.; MEDEIROS, J. S. *Geoprocessamento para projetos ambientais*. [S.l.]: INPE São José dos Campos, 1996.

CHANG, K.-T. *Introduction to geographic information systems*. [S.l.]: McGraw-Hill Boston, 2008. v. 4.

DELGADO, T.; GONZÁLEZ, G.; MIRANDA, G.; NAVARRO, D. G.; GRAVERÁN, A. Context-aware spatial decision support systems (ca-sdss): Articulating decision support systems, business intelligence and recommender systems considering the geospatial component. In: *Fourth International Workshop on Knowledge Discovery, Knowledge Management and Decision Support. Atlantis Press*. [S.l.: s.n.], 2013. p. 358–367.

DMITRY, N.; MANFRED, S.-S. On micro-services architecture. *International Journal of Open Information Technologies*, v. 2, n. 9, 2014.

DRAGONI, N.; GIALLORENZO, S.; LAFUENTE, A. L.; MAZZARA, M.; MONTESI, F.; MUSTAFIN, R.; SAFINA, L. Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering*, Springer, p. 195–216, 2017.

FOWLER, M. *Padrões de arquitetura de aplicações corporativas*. [S.l.]: Bookman, 2009.

FOWLER, M. Monolithfirst. 2015. Acessado: 09 Junho. 2022. Disponível em: <https://martinfowler.com/bliki/MonolithFirst.html>.

FOWLER, M. Software architecture guide. 2019. Acessado: 05 Junho. 2022. Disponível em: <https://martinfowler.com/architecture>.

FOWLER, M.; LEWIS, J. Microservices guide. 2015. Acessado: 21 Março. 2022. Disponível em: <https://martinfowler.com/microservices>.

GAMMA, E.; HELM, R.; JOHNSON, R.; JOHNSON, R. E.; VLISSIDES, J. et al. *Design patterns: elements of reusable object-oriented software*. [S.l.]: Pearson Deutschland GmbH, 1995.

GAO, S. *Geospatial Artificial Intelligence (GeoAI)*. [S.l.]: Oxford University Press, 2021.

GARLAN, D. Software architecture. Carnegie Mellon University, 2008.

HU, Y.; GAO, S.; LUNGA, D.; LI, W.; NEWSAM, S.; BHADURI, B. Geoai at acm sigspatial: progress, challenges, and future directions. *Sigspatial Special*, ACM New York, NY, USA, v. 11, n. 2, p. 5–15, 2019.

LI, W. Geoai: Where machine learning and big data converge in giscience. *Journal of Spatial Information Science*, n. 20, p. 71–77, 2020.

MAZLAMI, G.; CITO, J.; LEITNER, P. Extraction of microservices from monolithic software architectures. In: IEEE. *2017 IEEE International Conference on Web Services (ICWS)*. [S.l.], 2017. p. 524–531.

MEILLÓN, S. Geospatial intelligence and geospatial information systems. *NPS-Naval Postgraduation School: Monterey, CA, USA*, 2008.

NEWMAN, S. *Building microservices*. [S.l.]: "O'Reilly Media, Inc.", 2021.

POWER, D. J. *Decision support systems: concepts and resources for managers*. [S.l.]: Greenwood Publishing Group, 2002.

RADEMACHER, F.; SORGALLA, J.; SACHWEH, S. Challenges of domain-driven microservice design: A model-driven perspective. *IEEE Software*, IEEE, v. 35, n. 3, p. 36–43, 2018.

SANTOS, M. *A natureza do espaço: técnica e tempo, razão e emoção*. [S.l.]: Edusp, 2002. v. 1.

SHIM, J. P.; WARKENTIN, M.; COURTNEY, J. F.; POWER, D. J.; SHARDA, R.; CARLSSON, C. Past, present, and future of decision support technology. *Decision support systems*, Elsevier, v. 33, n. 2, p. 111–126, 2002.

THÖNES, J. Microservices. *IEEE software*, IEEE, v. 32, n. 1, p. 116–116, 2015.